

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці**

ОСВІТНЬО ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»

галузь знань 12 «Інформаційні технології»

спеціальність 122 «Комп'ютерні науки»

Форма навчання: денна

КВАЛІФІКАЦІЙНИЙ БАКАЛАВРСЬКИЙ ПРОЄКТ

на тему

**ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ВЕДЕННЯ
ОБЛІКУ ОСНОВНИХ ЗАСОБІВ І НЕМАТЕРІАЛЬНИХ АКТИВІВ**

здобувача Шахрая Дмитра Ігоровича

Науковий керівник:

к.е.н., доцент

Ситник Н.В

**Проект допущений до захисту перед екзаменаційною комісією з
атестації здобувачів вищої освіти**

завідувач кафедри:

к.е.н., доцент.

Тішков Б.О.

Київ 2024

Міністерство освіти і науки України
Київський національний економічний університет імені Вадима Гетьмана
Навчально-науковий інститут «Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

ПОГОДЖЕНО:

Керівник проектної групи(гарант)
освітньо-професійної програми

_____Іванченко Г.Ф.
« ____ » _____ 2024 р.

ЗАТВЕРДЖУЮ:

завідувач кафедри
_____Тішков Б.О.

“ ____ ” _____ 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувача вищої освіти **Шахрай Дмитра Ігоровича** *очної (денної) форми навчання*

на підготовку кваліфікаційного бакалаврського проєкту
на тему: **«Інформаційна система з обліку основних засобів та нематеріальних активів»**

Тему затверджено наказом ректора Університету від «11» березня 2024 р. № 529-ст.

Кваліфікаційний бакалаврський проєкт виконується на матеріалах наукових публікацій та матеріалів з мережі інтернет

План кваліфікаційного бакалаврського проєкту

Розділ I Характеристика та аналіз предметної області

Розділ II Розробка вимог і моделювання інформаційної системи.

Розділ III Проектування та реалізація компонентів системи.

Об'єкт дослідження інформаційні процеси, які характеризують процеси обліку основних засобів та нематеріальних активів.

Предмет дослідження сукупність теоретичних, методичних і прикладних аспектів з моделювання, проектування та програмування модулів для підвищення ефективності обліку та аналізу основних засобів та нематеріальних активів.

Мета кваліфікаційного бакалаврського проєкту систематизація і узагальнення сучасного досвіду розробки нових рішень та розробка інформаційної системи з обліку та аналізу основних засобів та нематеріальних активів.

Конкретні завдання, які студент повинен виконати для досягнення поставленої мети:

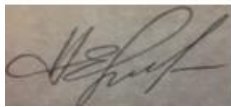
У розділі I Описати предметну галузь з обліку та аналізу основних засобів та нематеріальних активів. Навести характеристику об'єкта дослідження. Подати результати аналізу літературних джерел та існуючих варіантів інформаційних систем і технологій з обліку та аналізу основних засобів та нематеріальних активів.

У розділі II Визначити бізнес-вимоги, функціональні та нефункціональні вимоги до проєктованої системи та подати їх у формі діаграм і проєктної специфікації, розроблених з використанням CASE-засобів та мови SysML. Подати постановку та алгоритм розв'язання задачі автоматизації обліку та аналізу основних засобів та нематеріальних активів. Розробити моделі системи в специфікації мови UML

У розділі III Виконати проєктування компонентів інформаційної системи, зокрема інформаційного, технічного та програмного забезпечення. Виконати реалізацію програмного забезпечення задачі з обліку та аналізу основних засобів та нематеріальних активів, привести відповідні результати, вказавши можливість та доцільність застосування їх на практиці.

Завдання підготував

науковий керівник



Ситник Ніна Василівна

“14”березня 2024р.

**Завдання одержав
здобувач**



Шахрай Дмитро Ігорович

“14”березня 2024р.

В і д г у к
на кваліфікаційний бакалаврський проєкт
здобувача навчально-наукового інституту «Інститут інформаційних
технологій в економіці»
галузі знань 12 «Інформаційні технології»
спеціальності 122 «Комп'ютерні науки»

Шахрая Дмитра Ігоровича

на тему: «Інформаційна система з обліку основних засобів та нематеріальних активів»

Актуальність теми: Тема кваліфікаційного бакалаврського проєкту є актуальною, оскільки присвячена важливим питанням розробки інформаційної системи з обліку основних засобів та нематеріальних активів в сучасних умовах, які характеризуються досить частими змінами норм та правил в порядку обліку необоротних активів.

2. Позитивні риси кваліфікаційного бакалаврського проєкту: Моделювання інформаційної системи виконано з застосуванням сучасних засобів автоматизованого інжинірингу. Слід зазначити, що в проєкті розглянуто не лише традиційний облік основних засобів, а також облік нематеріальних активів, які включають патенти, авторські права, програмне забезпечення та інші необоротні активи, які характеризують інтелектуальну складову об'єкта управління, що впливає на його інвестиційну привабливість

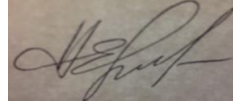
3. Наявність самостійних розробок автора: Проєкт розроблений самостійно з орієнтацією на використання в умовах різного виду об'єктів управління, які використовують різні методи нарахування амортизації та експлуатують різні операційні системи. Розроблене здобувачем програмне забезпечення структуроване та організоване за модульним принципом, що забезпечить його гнучкості та можливість розширення та підтримки майбутніх змін

4. Цінність теоретичних висновків та практичних рекомендацій: Пропозиції, які надані в проєкті, є безперечно цікавими і практично зорієнтованими. Крос-платформенна сумісність робить систему гнучкою та легко адаптованою до використання в умовах різних об'єктів управління, які потребують автоматизації обліку основних засобів та нематеріальних активів.

5. Наявність недоліків: В роботі зроблений акцент на різні методи нарахування амортизації і менше приділено уваги іншим обліковим функціям. В схемах алгоритмів використані графічні позначення, які не відповідають державному стандарту "Схеми алгоритмів, програм, даних і систем".

6. Загальна оцінка кваліфікаційного бакалаврського проєкту та його допущення до захисту перед ЕК: У цілому, представлений кваліфікаційний бакалаврський проєкт відповідає встановленим вимогам методичних вказівок щодо структури, обсягу та змісту та рекомендується до захисту з **високою оцінкою**, а **Шахраю Дмитру Ігоровичу** може бути присвоєний освітньо-кваліфікаційний рівень «бакалавр» галузі знань 12 «Інформаційні технології» за спеціальністю 122 «Комп'ютерні науки».

Науковий керівник: к.е.н., професор кафедри ІСЕ



Ситник Н.В.

“12” червня 2024 р.

Рецензія

на кваліфікаційний бакалаврський проєкт

Шахрая Дмитра Ігоровича

на тему: **«Проектування інформаційної системи з обліку основних засобів та нематеріальних активів»**

Актуальність теми кваліфікаційного бакалаврського проєкту і доцільність його розроблення

Тема проєкту є надзвичайно актуальною в сучасних умовах, коли ефективне управління основними засобами та нематеріальними активами стає критично важливим для функціонування та розвитку будь-якої організації. Сучасні інформаційні системи дозволяють автоматизувати процеси обліку, зменшуючи кількість помилок та забезпечуючи точність і своєчасність інформації, що є необхідною для прийняття обґрунтованих управлінських рішень. Розроблення такої системи є доцільним, оскільки вона сприятиме підвищенню ефективності управління активами, що, в свою чергу, може позитивно вплинути на фінансові показники організації.

Якість проведеного дослідження

Дослідження, проведене автором, відзначається глибиною аналізу та систематичним підходом до вирішення поставлених завдань. В першому розділі проєкту автор надає всебічну характеристику предметної галузі, аналізує існуючі програмні засоби, що використовуються в цій сфері, та визначає задачі, які потребують вирішення. Другий розділ, присвячений проектуванню системи, демонструє ґрунтовний підхід до розробки архітектури та алгоритмів функціонування системи. У третьому розділі автор детально розглядає технічне, програмне та організаційне забезпечення, необхідне для реалізації системи. Такий підхід свідчить про високу якість проведеного дослідження та належний рівень підготовки автора.

Позитивні риси кваліфікаційного бакалаврського проєкту

Проєкт має низку позитивних рис, зокрема:

- Комплексний підхід до вирішення проблеми автоматизації обліку основних засобів та нематеріальних активів.

- Детальне обґрунтування вибору методів та засобів розробки системи.

- Ретельний аналіз існуючих рішень та виявлення їхніх недоліків.

- Розробка чіткої та логічно побудованої архітектури інформаційної системи.

- Використання сучасних технологій та інструментів розробки, таких як Qt, C++ та SQLite.

- Практичні рекомендації щодо впровадження системи в реальних умовах.

Зауваження

Попри високу якість проєкту, є зауваження, яке варто врахувати:

- Необхідно надати більше уваги питанням безпеки даних, особливо в умовах зростаючих кіберзагроз.

Практична значимість висновків і рекомендацій

Висновки та рекомендації, наведені у кваліфікаційному бакалаврському проєкті, мають значну практичну цінність. Розроблена інформаційна система може бути успішно впроваджена в будь-якій організації для підвищення ефективності управління активами. Використання сучасних технологій та інструментів забезпечує гнучкість та масштабованість системи, що дозволяє адаптувати її до специфічних потреб конкретного підприємства. Рекомендації щодо впровадження системи є реалістичними та ґрунтуються на практичному досвіді автора, що підвищує їх цінність для потенційних користувачів.

Кваліфікаційний бакалаврський проєкт **Шахрая Дмитра Ігоровича** на тему «Проектування інформаційної системи з обліку основних засобів та нематеріальних активів» є актуальним, ґрунтовно виконаним та має значну практичну цінність. Автор продемонстрував високий рівень знань та вмінь у

сфері інформаційних технологій, що дозволяє рекомендувати проєкт до **високої оцінки** та впровадження в практичну діяльність.

Місце роботи та посада рецензента

ТОВ «Агро вест»,
директор підприємства,
керівник ІТ-відділу.

Науковий ступінь, учене звання (за наявності) _____

Ренчковський В.Л. _____



Місце печатки організації, де працює рецензент

АНОТАЦІЯ

кваліфікаційного бакалаврського проєкту студента 4 курсу
Навчально-наукового інституту «Інститут інформаційних технологій в
економіці»

Шахрая Дмитра Ігоровича,

на тему: **«Проектування інформаційної системи з обліку основних
засобів та нематеріальних активів»**,

Київ: кафедра інформаційних систем в економіці, 2024р.

Кваліфікаційний бакалаврський проєкт присвячений актуальній проблемі автоматизації бухгалтерського обліку основних засобів та нематеріальних активів, яку пропонується розв'язувати та удосконалювати з використанням сучасних інформаційних технологій.

Проект складається з трьох розділів, логічно пов'язаних між собою.

У першому розділі дана характеристика предметної галузі й об'єкта дослідження, наведено аналіз задач, що розв'язуються, а також наводиться перелік існуючих програмних засобів даної предметної галузі. Особлива увага приділена аналізу поточного стану бухгалтерського обліку основних засобів та нематеріальних активів, виявленню проблем та потреб, що виникають у процесі управління цими активами. Розглянуто різні програмні рішення, які використовуються для автоматизації цих процесів, їх переваги та недоліки.

Другий розділ є проєктним і присвячений обґрунтуванню методу проектування системи, розробленню її архітектури, виконанню постановки та розробленню алгоритму розв'язання задач. Тут описано процеси проектування системи бухгалтерського обліку основних засобів та нематеріальних активів, включаючи вибір методології, визначення вимог до системи, проектування бази даних, створення моделей даних та алгоритмів обробки інформації. Розроблено детальну архітектуру системи, що включає її основні компоненти та їх взаємодію.

Третій розділ – конструктивний. Тут наведено інформаційне, технічне, програмне та організаційне забезпечення для системи бухгалтерського обліку основних засобів та нематеріальних активів. Висвітлені питання організації інформаційного забезпечення: розроблена структура інформаційного забезпечення, описані та наведені в додатках форми вхідних та вихідних

документів. Обґрунтований комплекс технічних засобів, а також програмне забезпечення, що використовується для створення системи бухгалтерського обліку. Вказані структури інформаційних масивів, які використовуються під час вирішення задачі. Особлива увага приділена питанням інтеграції з існуючими системами та забезпечення безпеки даних.

Висновки містять рекомендації щодо доцільності розроблення та впровадження системи бухгалтерського обліку основних засобів та нематеріальних активів. Зроблено висновок про економічну та організаційну ефективність запропонованого рішення, його вплив на покращення управління активами, зниження витрат та підвищення прозорості обліку. Розглянуто можливості подальшого розвитку системи, включаючи розширення її функціональності та інтеграцію з іншими інформаційними системами бухгалтерського обліку.

РЕФЕРАТ

Кваліфікаційний бакалаврський проєкт містить __210__ сторінок, __10__ таблиць, __76__ рисунків, список літератури з __16__ найменувань, __4__ додатків.

«Проектування інформаційної системи з обліку основних засобів та нематеріальних активів»

Перелік ключових слів : Інформаційна система, облік основних засобів, нематеріальні активи, автоматизація, бухгалтерський облік, управління активами, проектування системи, база даних, SQLite, QT, C++, інтеграція, безпека даних, фінансовий моніторинг, технічне забезпечення.

Предметом дослідження є інформаційні системи для бухгалтерського обліку основних засобів та нематеріальних активів .

Об'єктом дослідження виступає процес автоматизації бухгалтерського обліку основних засобів та нематеріальних активів за допомогою інформаційних систем.

Мета кваліфікаційного бакалаврського проєкту полягає в розробці та впровадженні інформаційної системи для автоматизації бухгалтерського обліку основних засобів та нематеріальних активів, що сприятиме покращенню управління активами, підвищенню ефективності облікових процесів та зниженню ризиків.

Завданнями випускного бакалаврського проєкту є:

- Аналіз предметної галузі та виявлення основних проблем і потреб у сфері бухгалтерського обліку активів.
- Дослідження існуючих програмних засобів для автоматизації бухгалтерського обліку активів.
- Проектування архітектури інформаційної системи.
- Розробка бази даних та алгоритмів обробки інформації.
- Розробка програмного забезпечення для бухгалтерського обліку основних засобів та нематеріальних активів.
- Проведення тестування та оцінка ефективності розробленої системи.

Апаратні засоби: комп'ютери з операційною системою Windows, сервери для розміщення бази даних та програмного забезпечення. Програмні засоби: мова програмування C++, середовище розробки QT, система керування базами

даних SQLite.

Результати досягнуті в процесі роботи включають:

- Розроблена і впроваджена інформаційна система для бухгалтерського обліку основних засобів та нематеріальних активів.
- Знижені витрати часу та ресурсів на облік активів.
- Підвищена точність і достовірність облікових даних.
- Запропоновані методи контролю та захисту даних.

Новизна розробленої системи полягає у використанні сучасних інформаційних технологій для автоматизації процесів обліку активів, що дозволяє підвищити ефективність управління активами та знизити ризики.

Одержані результати можуть бути використані бухгалтерії для автоматизації обліку основних засобів та нематеріальних активів, що сприятиме підвищенню ефективності управління активами, зниженню ризиків і витрат. Розроблена система також може бути адаптована для використання в інших фінансових установах та організаціях, які потребують ефективного управління активами.

Рік виконання випускного бакалаврського проєкту 2024

Рік захисту випускного бакалаврського проєкту 2024

Перелік умовних позначень, символів, одиниць величин і термінів

IC	Інформаційна система
OЗ	Основні засоби
HA	Нематеріальні активи
БД	База даних
СКБД	Система керування базами даних
GUI	Графічний інтерфейс користувача (Graphical User Interface)
API	Програмний інтерфейс застосунків (Application Programming Interface)
SQL	Мова структурованих запитів (Structured Query Language)
QT	Кросплатформовий інструментарій для розробки графічних інтерфейсів користувача
C++	Мова програмування загального призначення.
SQLite	Легка автономна реляційна система керування базами даних.
Основні засоби (OЗ).	Матеріальні активи підприємства, які використовуються в господарській діяльності більше одного року
Нематеріальні активи (HA)	Активи, що не мають фізичної форми, але приносять економічні вигоди протягом тривалого часу
Амортизація	Це процес розподілу вартості основних засобів протягом їх корисного терміну використання.
CFO	Фінансовий директор
ERP	Планування ресурсів підприємства
KPI	Показники ефективності
IT	Інформаційні технології

OPD	Об'єктно-процесна діаграма
Use Case	Прецеденти
UML	Єдина мова моделювання
BDD	Діаграма визначення блоків
CRUD	Основні операції з даними (створення, читання, оновлення, видалення)
IBD	Розгорнута діаграма блоків
Стан idle	Стан бездіяльності
НД	Необхідні дані (документ)

ЗМІСТ

ВСТУП.....	1
РОЗДІЛ 1 ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ	4
1.1 Характеристика предметної галузі та об'єкта дослідження.....	4
1.2 Аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі.....	6
РОЗДІЛ 2 РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ.....	18
2.1 Аналіз і специфікація вимог до інформаційної системи	18
2.2 Постановка задачі та алгоритм розв'язання задачі	26
2.3 Моделювання інформаційної системи.....	46
РОЗДІЛ 3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ.....	66
3.1 Інформаційне забезпечення	66
3.2 Технічне забезпечення.....	78
3.3 Програмне забезпечення	81
3.4 Результати реалізації інформаційної системи.....	91
ВИСНОВКИ	93
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ	96

ВСТУП

В сучасному світі багато професій змінюються на інші, які все більше відносяться до комп'ютерних технологій і все більш зачіпляють тему максимальної автоматизації процесів, найпростішим буде приклад скорочення робочих місць на підприємствах, наприклад фабрики та цехи по виготовленню товарів. Ми живемо у той час коли працівників таких підприємств замінюють на автоматизовану робочу силу, адже технології дозволяють виключити людський фактор та пришвидшити роботу.

При цих змінах залишаються професії в яких залишається дуже важливим наявність людини – однією з таких професій є бухгалтер, до задач якого відносяться не тільки контроль виплат заробітних плат працівників, а й облік усіх грошових одиниць у компанії, перевірка, контроль та облік усіх процесів, працівників та усього обладнання .

Задля цих задач бухгалтери використовують специфічні програми для обліку усього на підприємстві, включаючи людей, їх заробітну плату, механічне оснащення яке використовує компанія і навіть такі дрібні на перший погляд речі як ліцензії програм та авторські права.

При створенні кваліфікаційного дипломного проєкту, паралельно вглиблюючись у роботу бухгалтера стало зрозуміло наскільки важливим етапом є облік усіх використаних та зароблених фінансів для детального аналізу, що приведе до подальшого розвитку компанії.

На даний момент основними інструментами для роботи бухгалтера є комп'ютерні програми які допомагають вести повний облік, аналізувати витрати та доходи компанії, створювати по цій інформацію звіти які б це відображали детально, щоб у подальшому керівник підприємства міг виконувати дії, які б сприяли розвитку та процвітанню його компанії

Бухгалтерський облік поняття комплексне й дуже обширне і далеко не всі програмні рішення є задовільними для цієї важливої роботи, на даний момент проаналізувавши роботу бухгалтера, було підсумовано що українських програм, які б підходили для вирішення усіх задач дуже мало і нажаль всюди

можна зустріти або російські аналоги або неповноцінні та не достатньо налаштовані комп'ютерні програми для обліку хоч однієї з частин роботи.

Тому було визначено що на розробку української інформаційної системи, що буде вирішувати задачі поставленні перед нею, що буде створена саме із урахуванням побажань самих бухгалтерів, буде великий запит.

Інформаційна система, що буде описана у цій дипломній роботі відповідає для основні запити користувачів для обліку нематеріальних активів та основних засобів, допоможе формувати звіти та має потенціал і буде розвиватись до рівня коли зможе конкурувати з програмними засобами які на даний момент використовуються.

Основними цілями для розробленої системи буде:

- Розвиток та збільшення функціоналу, що буде зачіпати увесь бухгалтерський облік ;
- Адаптивність до різних мов та законів різних держав;
- Аналіз та оцінка ефективності розробленої інформаційної системи;
- Реалізація розробленої системи для тестування та виправлення виявлених у подальшому помилок та побажань користувачів.

Об'єктом дослідження є інформаційні системи для ведення бухгалтерського обліку, зокрема обліку нематеріальних активів та основних засобів. Предметом дослідження є процеси автоматизації та оптимізації бухгалтерського обліку з використанням розробленої інформаційної системи.

У вступі проведено огляд літературних джерел (див. сторінка 94) за темою дослідження, включаючи найбільш цінні та актуальні роботи. За результатами огляду можна зробити висновок про необхідність подальшого поглиблення досліджень у цій галузі, зокрема щодо оптимізації структури активів та капіталу з урахуванням специфічних галузевих особливостей. Дана інформаційна система, яка буде описана в дипломній роботі, покликана задовольнити основні запити користувачів, щодо обліку нематеріальних активів та основних засобів, допоможе сформувати звітність та має потенціал для розвитку до рівня, коли вона зможе конкурувати з програмними засобами, що використовуються на даний момент.

Таким чином, вибір теми дослідження є обґрунтованим та актуальним, оскільки спрямований на вирішення важливої проблеми, що має значний соціальний та економічний ефект.

РОЗДІЛ 1 ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Характеристика предметної галузі та об'єкта дослідження

Активи підприємства визначаються як ресурси, створені минулими операціями або подіями, які належать або контролюються підприємством і мають потенціал принести економічні вигоди в майбутньому. Активи класифікуються за різними критеріями. Наприклад, за терміном використання активи поділяються на поточні та довгострокові, а за наявністю фізичної форми – на матеріальні та нематеріальні.

У цьому розділі ми детально розглянемо концепцію активів, їх класифікацію, а також важливість оптимальної структури активів для стабільного розвитку підприємства. Це буде основою для подальшого дослідження та розробки інформаційної системи для обліку нематеріальних активів та основних засобів, що дозволить ефективно управляти активами підприємства.

Для стабільного розвитку підприємство повинно раціонально планувати свою структуру активів. Це передбачає створення такої комбінації активів, яка дозволяє досягти синергії та оптимального співвідношення між ризиками і доходами. Правильно спланована структура активів сприяє здоровому і стабільному розвитку підприємства, забезпечуючи ефективне використання ресурсів та зменшення операційних ризиків.

Поточні активи – це активи, які підприємство планує використати або реалізувати протягом одного операційного циклу або одного року. До них відносяться грошові кошти, дебіторська заборгованість, товарно-матеріальні запаси та інші активи, які легко конвертуються в готівку.

Довгострокові активи – це активи, які плануються до використання на довгий термін, понад один рік. Вони включають основні засоби, такі як будівлі, обладнання, машини, а також нематеріальні активи, як патенти, ліцензії та авторські права.

Матеріальні активи – це активи, що мають фізичну форму. Вони включають землю, будівлі, обладнання, транспортні засоби тощо. Ці активи

використовуються в операційній діяльності підприємства і поступово зношуються.

Нематеріальні активи – це активи, які не мають фізичної форми, але мають вартість і можуть приносити економічні вигоди. До них відносяться патенти, торгові марки, авторські права, ліцензії, програмне забезпечення та інші інтелектуальні активи.

Будь-яке підприємство, незалежно від його бізнес-операцій, масштабу або бізнес-моделі, має свою унікальну структуру активів. Це розподіл і частка різних видів активів у загальній сумі активів підприємства. Структура активів тісно пов'язана з умовами діяльності підприємства і має безпосередній вплив на його операційні ризики та прибутковість.

Структура активів і структура капіталу мають тісний взаємозв'язок. З формальної точки зору, вони розташовані по обидва боки балансу, відображаючи фінансовий стан кожного підприємства; з точки зору змісту, вони є джерелом фінансування та способом використання коштів. Виробничі компанії, акції яких котируються на біржі, можуть оптимізувати свою структуру капіталу. У багатьох дослідженнях науковці використовують частку оборотних активів у загальній сумі активів та частку матеріальних активів у загальній сумі активів як показники для вимірювання структури активів, або використовують співвідношення боргу до активів та співвідношення власного капіталу як показники для вимірювання структури капіталу.

Важливим аспектом оптимізації є також управління ризиками. Підприємства повинні не лише враховувати поточний стан своїх активів і капіталу, але й прогнозувати майбутні потреби та ризики. Наприклад, у періоди економічної нестабільності може бути доцільним збільшити частку ліквідних активів, щоб забезпечити фінансову стійкість.

Проаналізувавши взаємозв'язок між капіталом, структурою активів та операційною діяльністю компаній, а також висунувши пропозиції щодо оптимізації структури капіталу та активів з урахуванням галузевих особливостей. Багато науковців зосереджували свою увагу на взаємозв'язку між структурою активів та операційною діяльністю компаній, однак більшість

з них лише досліджували кореляцію між цими двома показниками, не пропонуючи конкретних заходів з оптимізації. Це свідчить про необхідність подальшого поглиблення досліджень.

Однак, незважаючи на існуючі дослідження, питання оптимізації структури капіталу та активів залишається актуальним. Зокрема, варто приділити більше уваги специфічним галузевим особливостям, які впливають на ці структури. Наприклад, у високотехнологічних галузях може бути більша потреба у нематеріальних активах, таких як інтелектуальна власність та програмне забезпечення, тоді як у виробничих галузях більша частка може припадати на матеріальні активи, такі як машини та обладнання.

Таким чином, для досягнення оптимальної структури активів та капіталу необхідно здійснювати комплексний підхід, який враховує як внутрішні показники підприємства, так і зовнішні економічні умови. Тільки за таких умов можна досягти синергії між різними видами активів та капіталу, що сприятиме стабільному та ефективному розвитку підприємства.

1.2 Аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі

Проаналізувавши міжнародні стандарти бухгалтерського обліку можна визначати нематеріальні активи більш точно і науково обґрунтовано, ніж старий стандарт «немонетарні довгострокові активи без фізичної форми, які підприємство утримує для виробництва товарів, надання послуг, здачі в оренду іншим особам або для адміністративних цілей». Стандарт передбачає, що гудвіл, створений підприємством, а також внутрішньо генеровані бренди, назви газет тощо не повинні визнаватися нематеріальними активами.

Нематеріальні активи можуть бути визнані тільки тоді, коли виконуються наступні умови:

1. Існує ймовірність того, що підприємство отримає в майбутньому економічні вигоди, пов'язані з цим активом. При визнанні нематеріальних активів підприємства повинні зробити обґрунтовану оцінку різних економічних факторів, які можуть існувати протягом очікуваного строку корисного використання нематеріальних активів, і надати чіткі докази того,

що економічні вигоди, які генеруються нематеріальними активами, ймовірно, надійдуть до підприємства, і що вони можуть бути визнані як нематеріальні активи;

2. Собівартість нематеріальних активів можна достовірно оцінити. При оцінці вартості нематеріальних активів слід чітко визначати витрати на внутрішні науково-дослідні проєкти підприємства, а також правильно розмежовувати витрати на етапі дослідження та етапі розробки.

При подальшому перегляді інформації можна винести основні методи первісної оцінки нематеріальних активів які виокремлюють наступне - первісна оцінка нематеріальних активів відноситься до балансової вартості, коли нематеріальні активи вперше придбані. Метод історичної собівартості в основному використовується для первісної оцінки нематеріальних активів. Методи придбання в основному включають зовнішнє придбання, саморозвиток та інвестиції інвесторів. Розглянемо оцінки під більш гострим кутом:

1. Оцінка придбаних нематеріальних активів. Придбані нематеріальні активи слід первісно оцінювати за собівартістю. Вона включає ціну придбання, відповідні податки та інші витрати, безпосередньо пов'язані з використанням активу за призначенням. Якщо ціна придбання нематеріального активу відстрочується понад звичайні кредитні умови і має по суті фінансовий характер, собівартість нематеріального активу визначається на основі теперішньої вартості ціни придбання. Різниця між фактичним платежем і теперішньою вартістю ціни придбання, за винятком тієї, що підлягає капіталізації, повинна включатися до складу поточних прибутків і збитків протягом кредитного періоду;

2. Оцінка нематеріальних активів, створених власними силами. Нематеріальні активи, створені власними силами, - це витрати, понесені на етапі розробки внутрішніх науково-дослідних проєктів, які можуть сформувати нематеріальні активи підприємства. Загальна сума витрат, понесених до досягнення запланованого використання, визнається

собівартістю нематеріального активу, але не здійснюється коригування витрат, які були списані на витрати в попередніх періодах.

Далі слід зазначити ще й методи подальшої оцінки нематеріальних активів. Подальша оцінка нематеріальних активів - це оцінка вартості нематеріальних активів протягом їх існування, в основному включаючи амортизацію та зменшення корисності нематеріальних активів.

Спираючись на ті ж стандарти за яких було визначено нематеріальні активи, можна виокремити наступне, основні засоби – це матеріальні активи, які підприємство утримує для використання у процесі виробництва або постачання товарів, надання послуг, для передачі в оренду іншим особам або для здійснення адміністративних і соціально-культурних функцій. Вони мають строк корисного використання (експлуатації), який перевищує один рік (або операційний цикл, якщо він довший за рік).

При аналізі цієї інформації, основні засоби можуть бути визнані тоді, коли виконуються наступні умови:

1. Існує ймовірність того, що підприємство отримає в майбутньому економічні вигоди, пов'язані з використанням цього активу;
2. Собівартість основного засобу можна достовірно оцінити.

Для того щоб провести первісну оцінку основних засобів потрібно базуватись на їхній собівартості. Собівартість включає ціну придбання, податки, мита, витрати на доставку, встановлення і монтаж, а також інші витрати, безпосередньо пов'язані з приведенням активу до стану, в якому він придатний для використання за призначенням.

В подальшій оцінці основних засобів слід застосовувати два методи: за собівартістю та за переоціненою вартістю.

1. Оцінка за собівартістю: активи обліковуються за первісною вартістю мінус накопичена амортизація та накопичені збитки від зменшення корисності;
2. Оцінка за переоціненою вартістю: активи обліковуються за переоціненою вартістю, яка є справедливою вартістю на дату переоцінки

мінус накопичена амортизація та накопичені збитки від зменшення корисності.

Амортизація основних засобів є важливим аспектом їх подальшої оцінки, як і амортизація нематеріальних активів. Обидва типи активів потребують систематичного розподілу амортизаційної вартості протягом строку їх корисного використання. Таким чином потрібно не менш пильно розглянути амортизацію нематеріальних активів та амортизацію основних засобів.

Підприємства повинні аналізувати та визначати строк корисного використання нематеріальних активів при їх придбанні. Якщо строк корисного використання нематеріального активу обмежений, слід оцінити кількість років такого строку корисного використання або кількість подібних одиниць виміру, таких як обсяг виробництва, що становлять строк корисного використання. Якщо неможливо передбачити період часу, протягом якого нематеріальний актив приносить підприємству економічні вигоди, його слід розглядати як нематеріальний актив з невизначеним строком корисного використання.

Для нематеріальних активів з обмеженим строком корисного використання сума амортизації повинна систематично та обґрунтовано амортизуватися протягом строку їх корисного використання. Метод амортизації нематеріальних активів, обраний підприємством, повинен відображати очікуваний спосіб реалізації економічної вигоди, пов'язаних з нематеріальним активом. Якщо очікуваний спосіб реалізації не може бути достовірно визначений, для нарахування амортизації слід застосовувати прямолінійний метод. Сума амортизації нематеріальних активів, як правило, повинна включатися до прибутку або збитку поточного періоду, якщо інше не передбачено іншими стандартами бухгалтерського обліку. Нематеріальні активи з невизначеним строком корисного використання не повинні амортизуватися, слід відображати лише зменшення корисності.

Сума амортизації нематеріальних активів - це сума після вирахування з їх первісної вартості очікуваної ліквідаційної вартості. Для нематеріальних активів, щодо яких створено резерв на покриття збитків від зменшення корисності, слід також відняти накопичену суму резерву на покриття збитків

від зменшення корисності нематеріальних активів, що були створені. Ліквідаційна вартість нематеріальних активів з обмеженим строком корисного використання повинна дорівнювати нулю, за винятком таких випадків: перший - третя сторона пообіцяла придбати нематеріальний актив наприкінці строку його корисного використання; другий - інформація про очікувану ліквідаційну вартість може бути отримана на основі активного ринку, і існує ймовірність того, що такий ринок існуватиме на кінець строку корисного використання нематеріального активу.

Амортизація основних засобів – це систематичний розподіл амортизаційної вартості об'єкта основних засобів протягом строку його корисного використання.

Час амортизації: період амортизації починається з моменту, коли актив стає придатним для використання, і закінчується, коли актив визнається як непридатний для подальшого використання або вибуває з експлуатації.

Методи амортизації: для амортизації основних засобів можуть використовуватися різні методи, такі як прямолінійний метод, метод зменшення залишкової вартості, метод одиниць виробництва тощо.

Ліквідаційна вартість: це сума, яку підприємство очікує отримати за актив після закінчення строку його корисного використання мінус очікувані витрати на вибуття.

Коли балансова вартість активу перевищує суму очікуваної майбутньої економічної вигоди від його використання, відбувається зменшення корисності при цьому актив має бути списаний до справедливої вартості за вирахуванням витрат на продаж.

Подальшою оцінкою нематеріальних активів нелегко керувати та контролювати. Новий стандарт передбачає лише амортизацію та зменшення корисності для подальшої оцінки нематеріальних активів і не передбачає подальших витрат. Оскільки положення нового стандарту є більш гнучкими, багато аспектів подальшої оцінки вимагають професійного судження, наприклад, період амортизації, ліквідаційна вартість, зменшення корисності

тощо, і є більш суб'єктивними, що ускладнює контроль в управлінських технологіях і збільшує простір для маніпулювання прибутком.

Усі ці висновки та правила встановлені саме після аналізу літературних джерел та практичного досвіду, і саме це є важливою складовою частиною дослідження, оскільки дозволяє оцінити поточний стан розробок у предметній галузі та визначити напрямки для подальших досліджень і вдосконалення.

Під час написання цього розділу було проаналізовано :

1. Класичні підходи до бухгалтерського обліку:

Багато наукових робіт присвячено загальним принципам бухгалтерського обліку, включаючи теоретичні основи, методи та практики. Книги та статті відомих авторів, таких як Бабінський В.В., Ковальчук В.В. (див. сторінку 96, пункт 4) забезпечують глибоке розуміння основних принципів бухгалтерського обліку та фінансового аналізу. У їхніх працях детально розглянуто питання обліку основних засобів та нематеріальних активів.

2. Сучасні технології в бухгалтерському обліку:

З розвитком інформаційних технологій, з'являються нові підходи до ведення бухгалтерського обліку. У роботах авторів Лузан О.М., Олійник А.А., Пашкевич О.Б, (див. сторінку 96, пункт 5) розглянуто використання сучасних програмних засобів для автоматизації бухгалтерських процесів. Вони наголошують на важливості інтеграції інформаційних систем у роботу бухгалтерії для підвищення ефективності та зниження ризиків.

3. Використання баз даних у бухгалтерському обліку:

Дослідження, присвячені використанню баз даних у бухгалтерії, праці Симоненко В.В., Ставнійчук О.В., Тарасенко О.В., (див. сторінку 96, пункт 6) описують переваги зберігання та обробки фінансових даних у структурованих базах даних, зокрема SQL. Вони підкреслюють важливість забезпечення надійності та безпеки даних, що є критично важливим для бухгалтерських інформаційних систем.

4. Програмне забезпечення для бухгалтерії:

Багато джерел аналізують різні програмні рішення, такі як 1C, SAP, Oracle Financials, і їх здатність задовольняти потреби бухгалтерів. У працях багатьох

відомих економістів, таких як Горохольський В.О., Ковальчук В.В. (див. сторінку 96, пункт 4). детально розглянуто функціональні можливості цих систем, їхні переваги та недоліки.

У реальних кейсах, зібраних з досвіду українських та зарубіжних підприємств, описано впровадження інформаційних систем для обліку нематеріальних активів та основних засобів. Ці кейси демонструють, як такі системи допомагають підвищити точність обліку, зменшити кількість помилок та забезпечити своєчасне формування звітності.

1. Вплив автоматизації на роботу бухгалтерії:

У статтях та звітах Міністерства фінансів України (див. сторінку 96, пункт 7) розглянуто вплив автоматизації на ефективність роботи бухгалтерських служб. Дослідження показують, що автоматизація дозволяє знизити трудомісткість облікових операцій, покращити контроль за фінансовими потоками та забезпечити більш точний фінансовий аналіз.

2. Особливості впровадження інформаційних систем в Україні:

У працях Національної академії наук (див. сторінку 96, пункт 8) розглянуто специфічні аспекти впровадження інформаційних систем в українських умовах, зокрема, адаптацію програмного забезпечення до місцевих нормативно-правових вимог та облікових стандартів.

3. Проблеми та перспективи розвитку ІС:

У роботах, присвячених проблемам і перспективам розвитку інформаційних систем для бухгалтерії, таких авторів як Лузан О.М., Олійник А.А., Пашкевич О.Б (див. сторінку 96, пункт 5) аналізуються поточні виклики, з якими стикаються підприємства при впровадженні та використанні інформаційних систем, а також перспективи розвитку цих систем у найближчому майбутньому.

На основі проведеного аналізу літературних джерел та практичного досвіду можна зробити висновок, що ефективний облік основних засобів та нематеріальних активів є критично важливим для забезпечення достовірності фінансової звітності та успішного управління підприємством. Це дослідження спрямоване на розробку системи, яка відповідатиме сучасним вимогам

автоматизації бухгалтерських процесів і сприятиме підвищенню ефективності обліку та управління активами підприємства.

Роздивимось більше детально, слід провести ідентифікацію основних користувачів та осіб, що приймають рішення, це дозволяє створити ефективну інформаційну систему, яка відповідатиме потребам усіх зацікавлених сторін. Розуміння їхніх ролей, завдань та вимог забезпечить успішну розробку та впровадження системи, що сприятиме підвищенню ефективності та точності бухгалтерського обліку на підприємстві. Ось основні визначенні кола користувачів та осіб, що приймають рішення.

Основними користувачами системи було окреслено :

1. Бухгалтери

Ролі та завдання: Ведення бухгалтерського обліку, зокрема обліку нематеріальних активів та основних засобів, складання звітності, проведення переоцінки активів, нарахування амортизації.

Вимоги: Інтуїтивно зрозумілий інтерфейс, доступ до всіх необхідних функцій обліку, генерація стандартних і налаштовуваних звітів, інтеграція з іншими бухгалтерськими програмами.

Доцільність: Висока, оскільки бухгалтери є основними користувачами системи і їх ефективність безпосередньо впливає на точність та своєчасність бухгалтерського обліку.

2. Фінансові менеджери

Ролі та завдання: Аналіз фінансового стану підприємства, планування бюджету, контроль за виконанням фінансових планів, прийняття рішень щодо інвестицій в основні та нематеріальні активи.

Вимоги: Доступ до аналітичних інструментів, можливість формування детальних фінансових звітів, інтеграція з ERP-системами для отримання актуальної інформації.

Доцільність: Висока, оскільки фінансові менеджери потребують точних даних для прийняття стратегічних рішень щодо фінансів компанії.

3. Керівники підприємств

Ролі та завдання: Прийняття стратегічних рішень щодо розвитку підприємства, затвердження бюджету, оцінка ефективності використання активів.

Вимоги: Зручний доступ до узагальненої інформації, можливість перегляду ключових показників ефективності (KPI), доступ до звітів у зручному форматі.

Доцільність: Висока, оскільки керівники підприємств потребують зрозумілої та достовірної інформації для прийняття обґрунтованих рішень.

4. Аудитори

Ролі та завдання: Перевірка достовірності фінансової звітності, контроль за дотриманням нормативних вимог та стандартів обліку, надання рекомендацій щодо покращення облікової політики.

Вимоги: Доступ до історичних даних та можливість їх аналізу, зручні інструменти для перевірки та верифікації інформації, інтеграція з іншими обліковими системами.

Доцільність: Висока, оскільки аудитори забезпечують контроль за правильністю ведення обліку та надають рекомендації для підвищення ефективності.

Особами, що приймають рішення було визначено:

1. Фінансовий директор (CFO)

Ролі та завдання: Визначення фінансової стратегії підприємства, управління фінансовими ресурсами, затвердження бюджету та контроль за його виконанням, прийняття рішень щодо інвестицій.

Характер рішень: Стратегічні та оперативні рішення щодо управління фінансами та інвестиціями.

Рівень доцільності: Високий, оскільки фінансовий директор забезпечує фінансову стабільність підприємства та його розвиток.

2. Головний бухгалтер

Ролі та завдання: Координація роботи бухгалтерії, забезпечення дотримання облікової політики, складання фінансової звітності, взаємодія з аудиторами та контролюючими органами.

Характер рішень: Оперативні рішення щодо ведення бухгалтерського обліку, коригування облікової політики, впровадження нових облікових стандартів.

Рівень доцільності: Високий, оскільки головний бухгалтер відповідає за точність та своєчасність фінансової звітності.

3. IT-директор

Ролі та завдання: Забезпечення функціонування інформаційних систем, впровадження нових технологій, підтримка інтеграції з іншими системами, забезпечення безпеки даних.

Усі ролі, об'єкти та очікуваний функціонал інформаційної системи по обліку нематеріальних активів та основних засобів зображений на рис. 1.1.

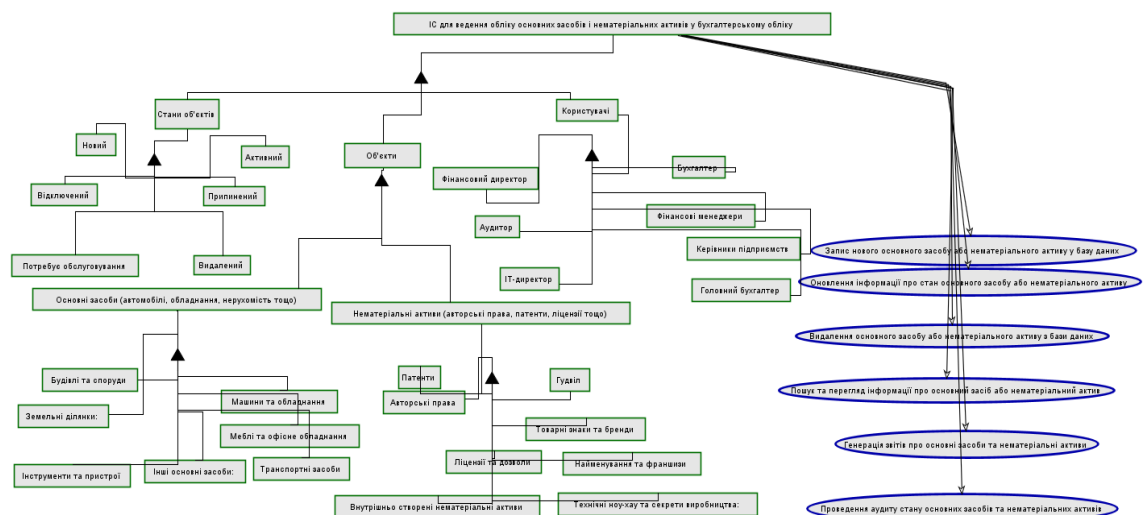


Рис. 1.1 OPD-діаграма з повним складом процесів і об'єктів з їхніми станами для інформаційної системи.

Характер рішень: Технічні рішення щодо вибору та впровадження інформаційних систем, забезпечення їх надійності та безпеки.

Рівень доцільності: Високий, оскільки IT-директор забезпечує технічну підтримку та розвиток інформаційної інфраструктури підприємства.

Як вже було вказано раніше метою даної роботи є розробка і впровадження інформаційної системи для автоматизації бухгалтерського обліку нематеріальних активів та основних засобів. Ця система повинна

забезпечити ефективне ведення обліку, в межах задач виконання кваліфікаційного дипломного проєкту, зменшення помилок, підвищення швидкості обробки даних та полегшення процесу складання фінансової звітності. Визначення мети роботи, її важливості та актуальності дозволяє чітко окреслити напрямок дослідження та розробки інформаційної системи для бухгалтерського обліку нематеріальних активів та основних засобів. Розуміння цих аспектів сприяє підвищенню ефективності обліку, забезпеченню точності та надійності фінансової звітності, що в кінцевому результаті позитивно вплине на фінансовий стан підприємства та його конкурентоспроможність на ринку.

Розробка і впровадження інформаційної системи бухгалтерського обліку нематеріальних активів та основних засобів має кілька важливих аспектів:

1. Підвищення ефективності бухгалтерського обліку: Автоматизація облікових процесів дозволяє скоротити час на ведення обліку та зменшити кількість помилок, що виникають при ручній обробці даних;
2. Забезпечення точності та надійності даних: Використання сучасних інформаційних технологій сприяє підвищенню точності облікових даних та забезпечує їх надійне зберігання та обробку;
3. Спрощення складання фінансової звітності: Система повинна автоматично генерувати необхідні звіти, а саме: звіт амортизації, руху активів, інвентарна картка, обліку активів, звіт за категоріями, що полегшує роботу бухгалтерів і фінансових менеджерів та забезпечує своєчасне подання звітності;
4. Забезпечення відповідності нормативним вимогам: Система повинна відповідати сучасним вимогам національних та міжнародних стандартів бухгалтерського обліку, що сприяє підвищенню якості фінансової звітності.

Актуальність даної роботи обумовлена наступними факторами:

1. Зростаюча роль нематеріальних активів: У сучасній економіці нематеріальні активи, такі як патенти, авторські права, торгові марки, відіграють все більшу роль. Їх облік стає важливим аспектом діяльності

підприємства, який впливає на його фінансові результати та інвестиційну привабливість;

2. Необхідність модернізації облікових систем: Багато підприємств використовують застарілі системи обліку, які не відповідають сучасним вимогам і не забезпечують необхідного рівня автоматизації. Це призводить до низької ефективності роботи бухгалтерії та збільшує ризик помилок;

3. Вимоги нормативних актів: Введення нових стандартів бухгалтерського обліку, вимагає від підприємств оновлення підходів до ведення обліку та впровадження нових інформаційних систем;

4. Конкурентні переваги: Підприємства, які використовують сучасні інформаційні системи, мають значні конкурентні переваги. Вони можуть швидше і точніше обробляти облікову інформацію, приймати обґрунтовані фінансові рішення та забезпечувати вищу якість фінансової звітності.

РОЗДІЛ 2 РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Аналіз і специфікація вимог до інформаційної системи

У цьому розділі ми розглянемо вимоги до інформаційної системи для обліку нематеріальних активів та основних засобів. Система має забезпечити автоматизацію та оптимізацію процесів обліку нематеріальних активів та основних засобів, створення звітності, а також аналізу та управління активами.

Актуальність цієї роботи обумовлена необхідністю вдосконалення систем обліку активів на підприємствах, особливо у контексті переходу на міжнародні стандарти бухгалтерського обліку, можливості використання системи, що буде спрямована на визначений аспект. Існуючі рішення не завжди задовольняють вимоги підприємств, тому розробка нової системи, що враховує сучасні вимоги та особливості українського бізнес-середовища, є важливою та актуальною. Тож сформованою метою цієї роботи буде розробка інформаційної системи, яка забезпечить високий рівень автоматизації процесів обліку нематеріальних активів та основних засобів, що, в свою чергу, сприятиме підвищенню ефективності управління цими активами в українських підприємствах. Задля досягнення подібних цілей перед створенням системи потрібно встановлювати їй необхідні вимоги, розглянемо їх більш детально. Почнемо з бізнес-вимоги до системи.

Бізнес-цілі для розробки і впровадження інформаційної системи управління обліком основних засобів та нематеріальних активів у сфері бухгалтерського обліку формують основну мету та функціональність системи з урахуванням потреб користувачів та вимог ринку. Однією з цілей є автоматизація та оптимізація управління процесами обліку, включаючи реєстрацію та облік основних засобів та нематеріальних активів. Додатковою метою є підвищення якості обслуговування користувачів, забезпечення швидкої реакції на їхні запити та поліпшення комунікації між відділами бухгалтерського обліку.

Бізнес-вимоги визначають функціональність та характеристики, необхідні для задоволення потреб користувачів та досягнення бізнес-цілей. У

створенні інформаційної системи для управління обліком бухгалтерського обліку стейкхолдери включають власників, менеджерів та адміністраторів. Кожен стейкхолдер має унікальні потреби, які відображаються у вимогах до системи. Наприклад, власник може бажати мати доступ до фінансової звітності та аналізу ефективності використання активів, тоді як адміністратор може потребувати інтерфейс для реєстрації та обліку основних засобів та нематеріальних активів. Технічна підтримка може вимагати систему для відстеження та вирішення технічних проблем, що виникають при використанні системи обліку.

Вимоги до системи

Враховуючи бізнес-цілі та потреби стейкхолдерів, вимоги до системи для ефективного ведення обліку основних засобів та нематеріальних активів у бухгалтерському обліку можуть включати наступне:

1. Реєстрація та облік активів:

Система повинна забезпечувати можливість реєстрації та обліку основних засобів та нематеріальних активів, включаючи їхні атрибути, такі як опис, вартість, дата придбання, термін служби тощо.

2. Управління життєвим циклом активів:

Система має забезпечувати можливість ведення обліку всіх етапів життєвого циклу активів, включаючи придбання, експлуатацію, ремонт та списання.

3. Забезпечення доступу до інформації:

Система повинна мати механізми доступу, які забезпечують, що користувачі мають доступ лише до необхідної для їхньої ролі інформації.

4. Генерація звітності:

Система повинна забезпечувати можливість генерації різноманітних звітів та аналітичних даних для власників, менеджерів та інших зацікавлених сторін.

5. Інтеграція з іншими системами:

Якщо це необхідно, система повинна мати можливість інтеграції з іншими бухгалтерськими системами, ERP-системами або системами управління ресурсами підприємства (ERP).

6. Забезпечення безпеки даних:

Система повинна мати механізми захисту конфіденційності, цілісності та доступності даних, включаючи контроль доступу та резервне копіювання.

Загальна діаграма вимог до інформаційної системи бухгалтерського обліку нематеріальних активів та основних засобів зображена на рис 2.1.

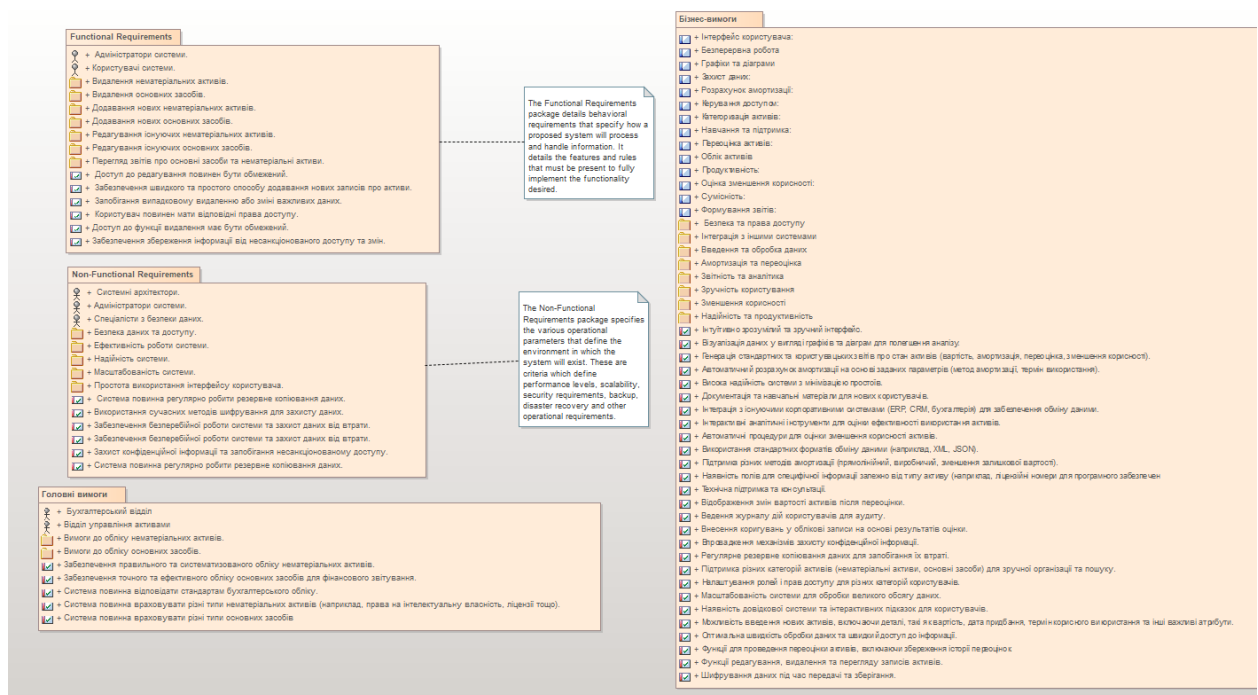


Рис. 2.1 Діаграма вимог до інформаційної системи бухгалтерського обліку нематеріальних активів та основних засобів.

Рівень доцільності рішень

1. Ефективність - автоматизація обліку активів зменшує ймовірність помилок, знижує трудовитрати та підвищує швидкість обробки даних;
2. Прозорість - система забезпечує прозорість та відстежуваність всіх операцій з активами, що важливо для фінансового контролю та аудиту;
3. Гнучкість - можливість інтеграції з іншими системами дозволяє гнучко налаштовувати інформаційну інфраструктуру підприємства відповідно до його потреб;

4. Безпека - високий рівень захисту даних знижує ризики втрати або несанкціонованого доступу до конфіденційної інформації.

Функціональні вимоги визначають основні можливості та функції, які інформаційна система для бухгалтерського обліку повинна надавати для задоволення потреб користувачів та досягнення бізнес-цілей проєкту. Вони забезпечують, що система матиме необхідний функціонал для бронювання та управління основними засобами, обліку фінансових операцій та послуг, а також для генерації звітів та аналітичної звітності.

Функціональні вимоги також передбачають наявність важливих функцій безпеки, таких як контроль доступу до даних, захист від несанкціонованого доступу та забезпечення конфіденційності інформації. Також вони включають можливість автоматизації процесів.

Характеристика головних функціональних вимог системи:

1. Реєстрація активів

Система повинна надавати можливість реєстрації нових основних засобів та нематеріальних активів, включаючи відомості про них, такі як назва, опис, вартість, дата придбання тощо;

2. Оновлення інформації про активи

Користувачі повинні мати можливість вносити зміни до інформації про існуючі активи, наприклад, змінювати їхні атрибути або відображати проведені операції з ними;

3. Пошук та фільтрація активів

Система повинна мати можливість швидкого пошуку та фільтрації активів за різними критеріями, такими як назва, категорія, вартість тощо;

4. Генерація звітності

Користувачі повинні мати можливість генерувати різноманітні звіти та аналітичні дані про стан активів, їх використання, амортизацію та інші показники;

5. Моніторинг та контроль

Система повинна забезпечувати моніторинг та контроль за станом активів, їх використанням та технічним станом для своєчасного прийняття управлінських рішень;

6. Автоматизація процесів

Система повинна автоматизувати рутинні процеси, такі як реєстрація нових активів, розрахунок амортизації, генерація звітів тощо, для зменшення навантаження на персонал та запобігання помилкам;

7. Інтеграція з іншими системами

Система повинна мати можливість інтеграції з іншими бухгалтерськими системами, ERP-системами або системами управління ресурсами підприємства (ERP) для обміну даними та забезпечення їхньої сумісності.

Детальніше переглянути функціональні вимоги можна на діаграмі, що зображена на рис. 2.2, та специфікації вимог, зроблені за допомогою менеджера специфікацій (Specification Manager) Enterprise Architect, що зображені на рис. 2.3-2.4.

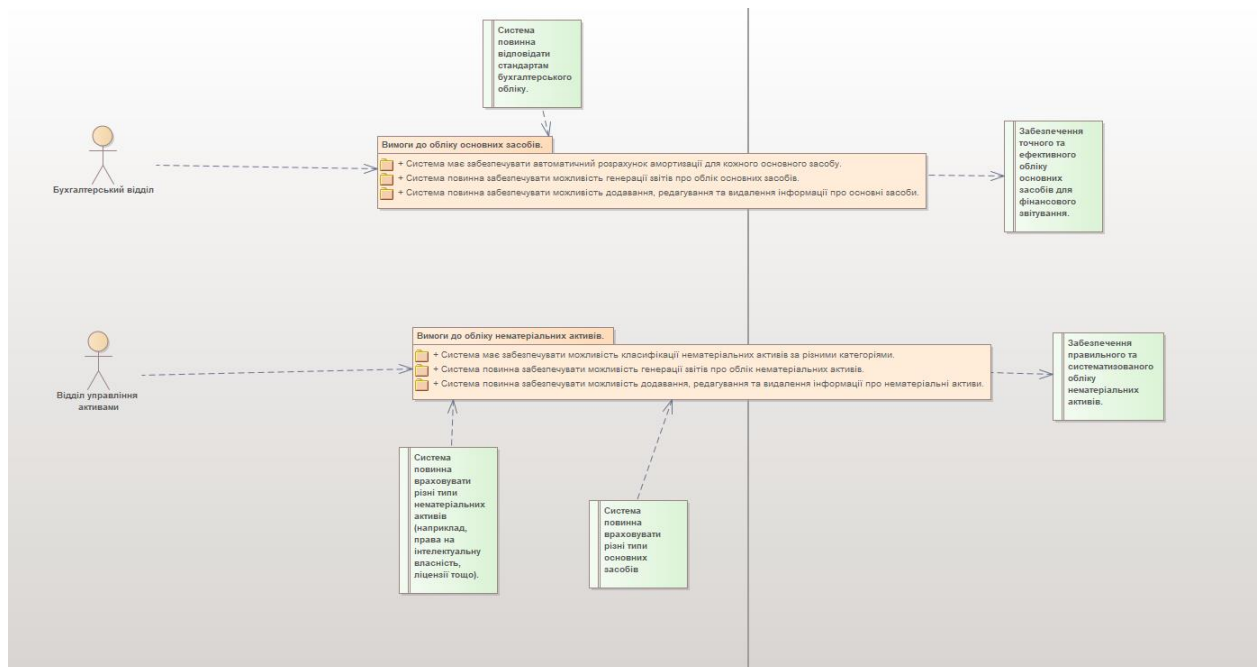


Рис. 2.2 Головні вимоги до інформаційної системи.

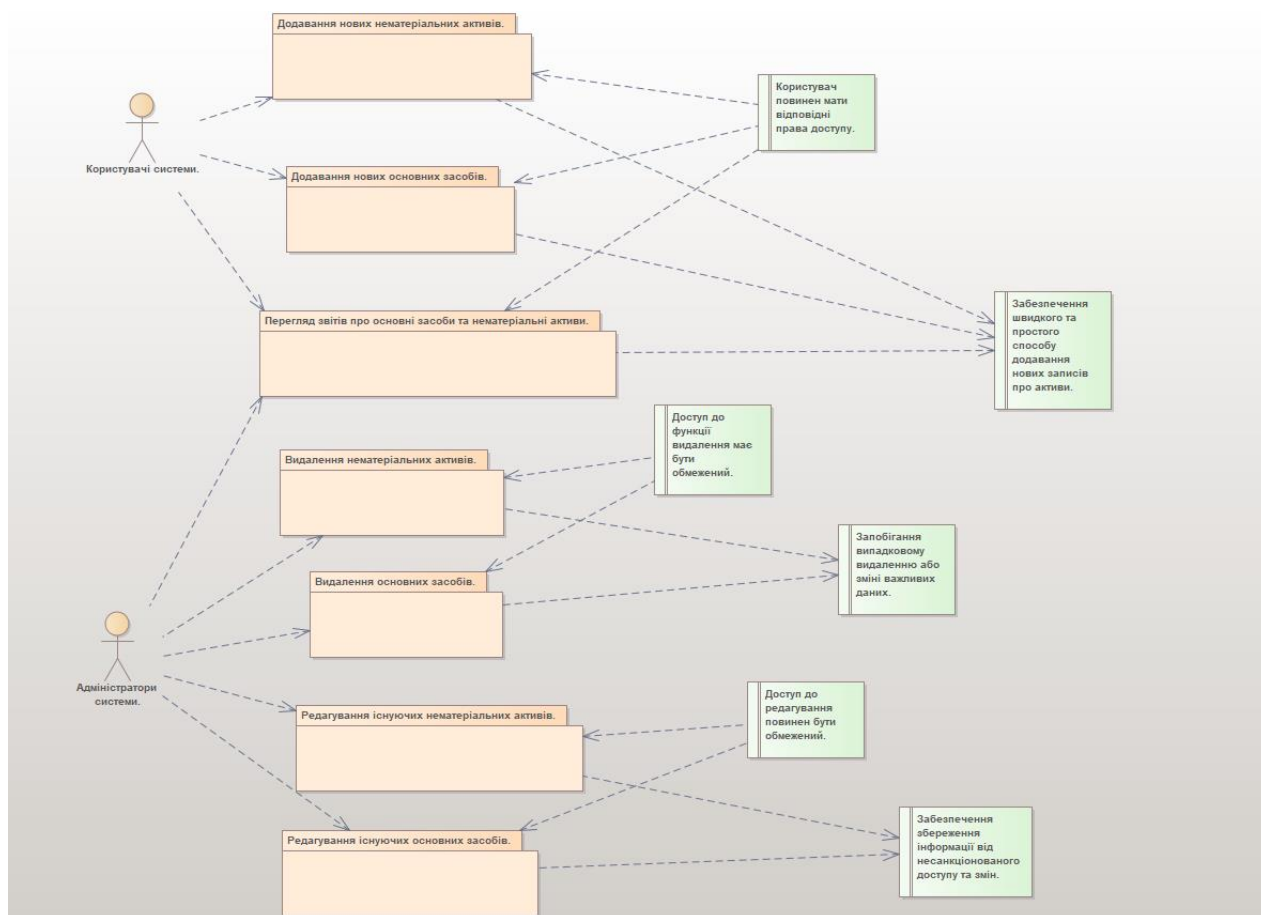


Рис. 2.3 Функціональні вимоги до інформаційної системи.

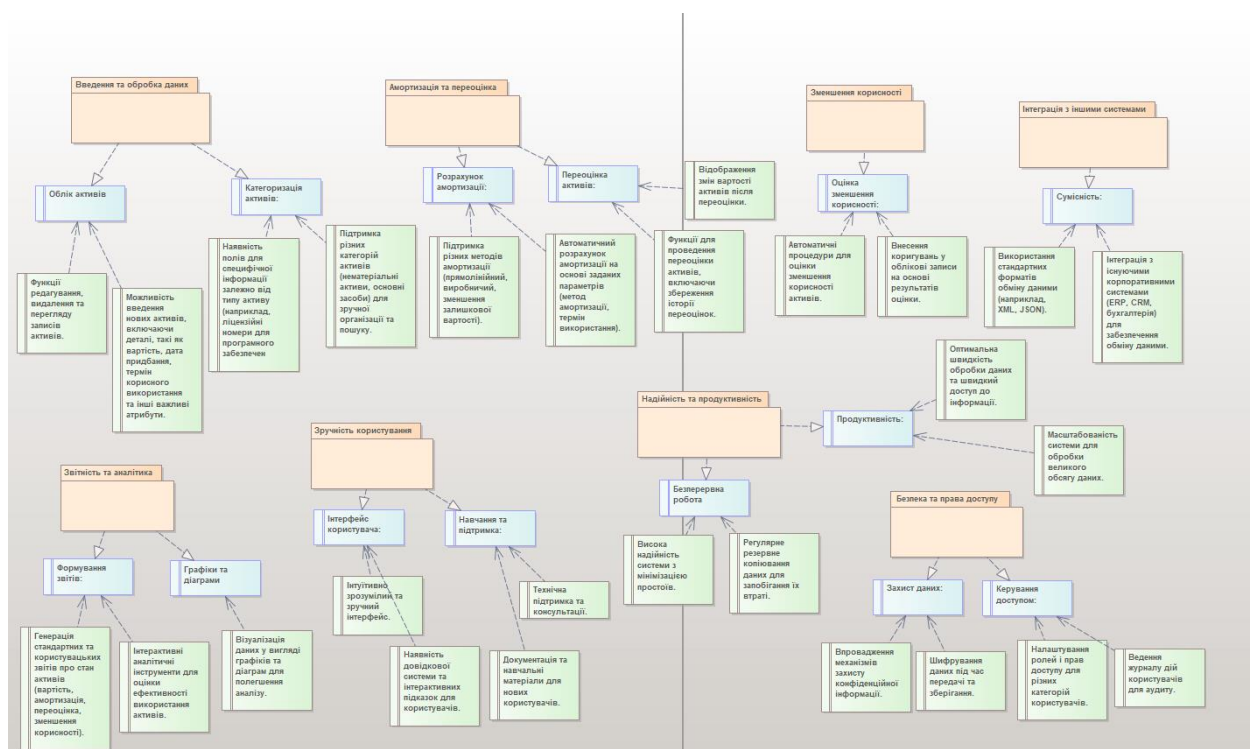


Рис. 2.4 Бізнес вимоги до інформаційної системи.

Нефункціональні вимоги до системи важливі так само, як і функціональні вимоги, оскільки вони визначають параметри якості, безпеки, надійності та інші характеристики, які повинна мати система. Ось деякі приклади нефункціональних вимог для системи ведення обліку основних засобів та нематеріальних активів:

1. Безпека:

Система повинна мати механізми автентифікації та авторизації користувачів для захисту від несанкціонованого доступу.

Вимога до шифрування конфіденційної інформації та забезпечення захисту даних під час їх передачі та зберігання.

2. Продуктивність:

Система повинна мати високу продуктивність та ефективність роботи навіть при великому обсязі даних та навантаженні.

Вимога до швидкості відповіді системи на запити користувачів та обробки інформації.

3. Надійність:

Система повинна бути стійкою до відмов та забезпечувати високий рівень доступності, уникнення втрати даних та забезпечення безперервної роботи.

Вимога до резервного копіювання даних та можливості відновлення після випадків аварійного завершення роботи системи.

4. Сумісність:

Система повинна бути сумісною з різними операційними системами, браузерами та пристроями для забезпечення універсального доступу.

Вимога до стандартів та протоколів обміну даними для забезпечення інтеграції з іншими системами.

5. Масштабованість:

Система повинна бути масштабованою, тобто здатною працювати ефективно при збільшенні обсягу даних та користувачів.

Вимога до горизонтального та вертикального масштабування системи для забезпечення росту бізнесу.

6. Документація:

Система повинна мати чітку та зрозумілу документацію, яка описує її архітектуру, функції та інші аспекти для полегшення розуміння та підтримки.

Загальні нефункціональні вимоги можна оглянути на рис. 2.6 та специфікації вимог, зроблені за допомогою менеджера специфікацій (Specification Manager) Enterprise Architect, що зображені на рис. 2.5-2.6.

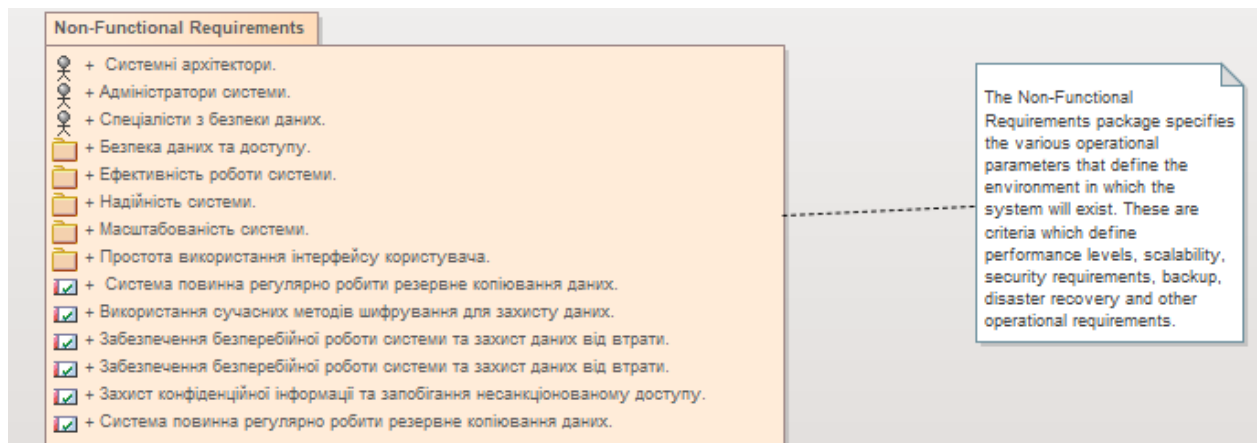


Рис. 2.5 Загальний вигляд нефункціональних вимог до інформаційної системи.

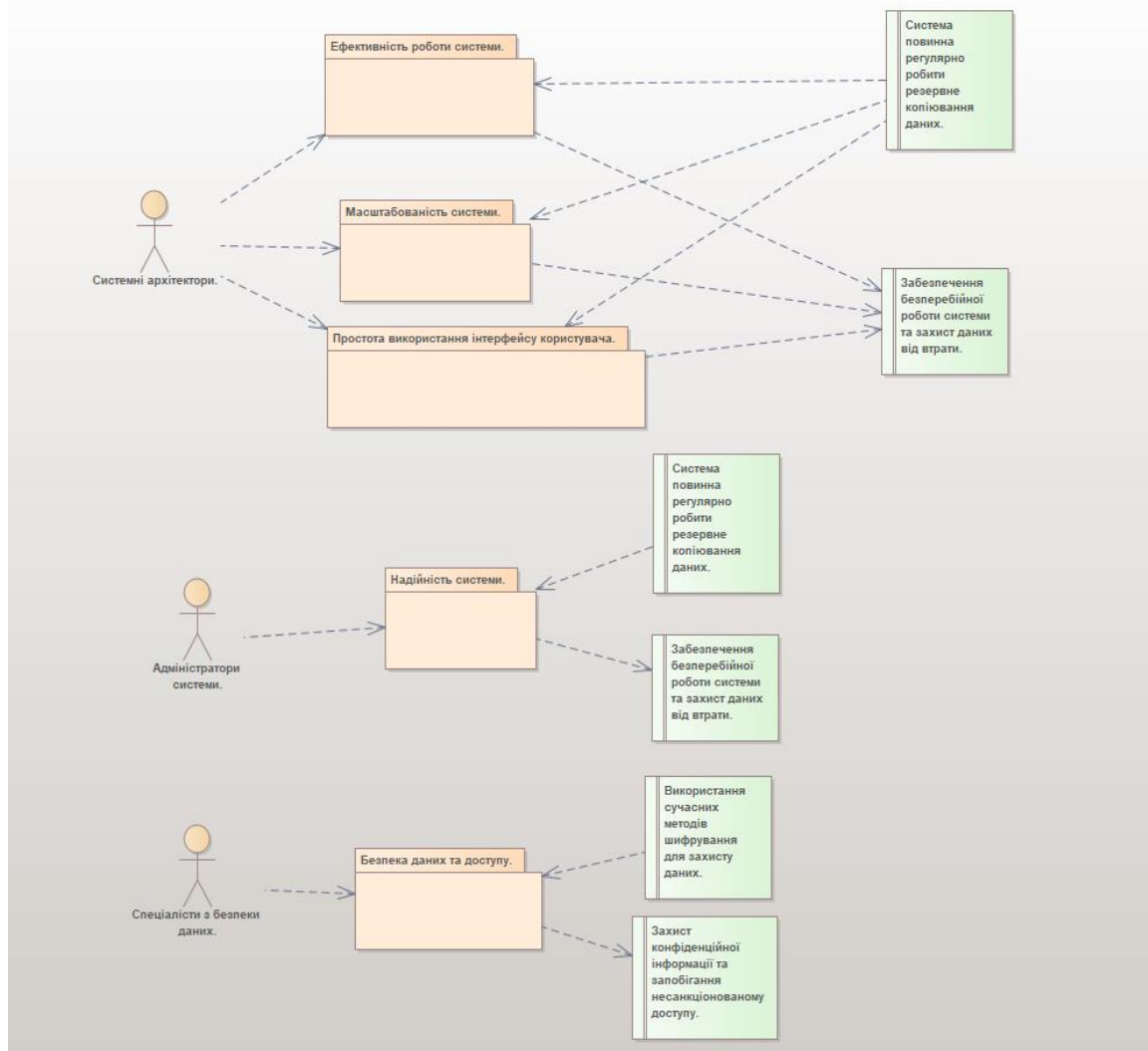


Рис. 2.6 Детальний опис нефункціональних вимог до інформаційної системи.

2.2 Постановка задачі та алгоритм розв'язання задачі

Основною метою розробки інформаційної системи обліку основних засобів та нематеріальних активів є автоматизація облікових процесів підприємства. Система покликана зменшити трудовитрати, мінімізувати помилки, забезпечити своєчасне та достовірне формування звітності, підвищити ефективність управління активами підприємства.

Техніко-економічна суть завдання полягає у створенні програмного продукту, який дозволить значно скоротити час, необхідний для ведення обліку, забезпечити достовірність даних та зменшити витрати, пов'язані з ручною обробкою інформації. Автоматизація облікових процесів дозволить

підприємству зосередитися на стратегічних завданнях, підвищити продуктивність праці та покращити якість управлінських рішень.

До переліку об'єктів, щодо яких вирішуються завдання під його управлінням, входять:

- Основні засоби (нерухомість, обладнання, транспортні засоби тощо);
- Нематеріальні активи (патенти, ліцензії, торгові марки тощо).

Вихідна інформація, що генерується системою, використовується для:

- формування фінансової та управлінської звітності для керівництва підприємства;
- аналізу ефективності використання основних засобів і нематеріальних активів;
- прийняття рішень про інвестиції в нові активи або списання існуючих активів;
- забезпечення відповідності даних вимогам внутрішнього та зовнішнього аудиту.

Система повинна підтримувати різні цикли вирішення проблем:

- Щоденний облік та моніторинг активів;
- Щомісячна генерація звітів (див. додаток Г, сторінку 112), та витрати на утримання активів.

Нижче наведена інформаційна модель системи бух. обліку, на рис.2.7.

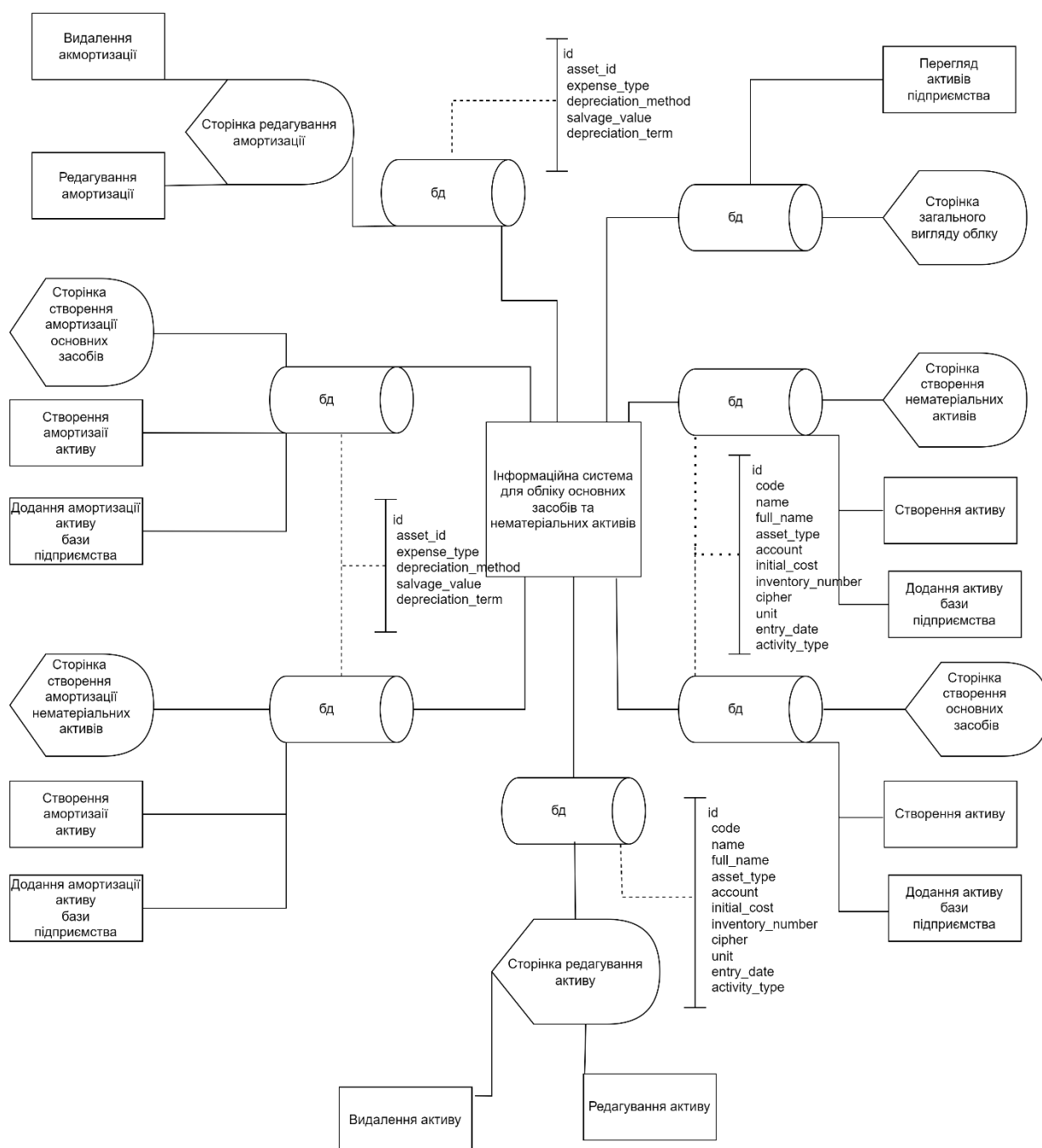


Рис. 2.7 Діаграма класів інформаційної системи.

На таблиці 2.1 що наведена нижче представлена уся інформація щодо вихідних повідомлень системи.

№з п/п	Назва вихідного повідомленн я	Ідентифі катор	Ф орма по данн я і ви моги до неї	Періо дичність видання	Тер мін видання і допусти мий час затримки	Кори стувачі інформаці ї
	Звіт про	ZATA_0	Електр	Щомісячн	На	Керів

1	амортизацію основних засобів	01	онний документ (PDF)	о, Щоквартально	початку робочого дня \ місяця \ кварталу допустима затримка - 1 год	ництво, бухгалтерія
2	Звіт про амортизацію основних засобів(прямолінійним методом)	ZATAP M_002	Електронний документ (PDF)	Щомісячно, Щоквартально	На початку робочого дня \ місяця \ кварталу допустима затримка - 1 год	Керівництво, бухгалтерія
33	Звіт про амортизацію основних засобів(методом зменшеного залишку)	ZATAM ZZ_003	Електронний документ (PDF)	Щомісячно, Щоквартально	На початку робочого дня \ місяця \ кварталу допустима затримка - 1 год	Керівництво, бухгалтерія
4	Звіт про амортизацію основних засобів(виробничий метод)	ZATAV M_004	Електронний документ (PDF)	Щомісячно, Щоквартально	На початку робочого дня \ місяця \ кварталу допустима затримка - 1 год	Керівництво, бухгалтерія
5	Звіт про амортизацію нематеріальних активів	ZAITA_005	Електронний документ (PDF)	Щомісячно, Щоквартально	На початку робочого дня \ місяця \ кварталу допустима	Керівництво, бухгалтерія

					ма затримка - 1 год	
6	Звіт про амортизацію нематеріальних активів (прямолінійним методом)	ZAITAP M_006	Електронний документ (PDF)	Щомісячно, Щоквартально	На початку робочого дня \ місяця \ кварталу допустима затримка - 1 год	Керівництво, бухгалтерія
7	Звіт про амортизацію нематеріальних активів (методом зменшеного залишку)	ZAITAM ZZ_007	Електронний документ (PDF)	Щомісячно, Щоквартально	На початку робочого дня \ місяця \ кварталу допустима затримка - 1 год	Керівництво, бухгалтерія
8	Звіт про амортизацію нематеріальних активів (виробничий метод)	ZAITAV M_008	Електронний документ (PDF)	Щомісячно, Щоквартально	На початку робочого дня \ місяця \ кварталу допустима затримка - 1 год	Керівництво, бухгалтерія
9	Звіт про стан активів	ZSA_009	Електронний документ (PDF)	Щомісячно, Щоквартально	На початку робочого дня \ місяця \ кварталу допустима затримка - 1 год	Керівництво, бухгалтерія
10	Звіт про рух активів	ZMA_010	Електронний документ	Щомісячно, Щоквартально	На початку робочого	Керівництво, бухгалтер

			нт (PDF)	льно	дня\ місяця\к варталу допусти ма затримка - 1 год	ія
11	Звіт – інвентарна картка активу	ZICA_01 1	Електр онний докуме нт (PDF)	Щомісячн о, Щокварта льно	На початку робочого дня\ місяця\к варталу допусти ма затримка - 1 год	Керівницт во, бухгалтер ія
12	Звіт по категоріям активів	ZCA_01 2	Електр онний докуме нт (PDF)	Щомісячн о, Щокварта льно	На початку робочого дня\ місяця\к варталу допусти ма затримка - 1 год	Керівницт во, бухгалтер ія
13	Загальна модель системи зображена у вигляді таблиці	GVSMТ V_013	Електр онний докуме нт (.xml, .sql, .sqlite.)	Щоденно	На початку роботи з базою підприєм ства допусти ма затримка - відсутня	Керівницт во, бухгалтер ія, аналітичн ий відділ, аудитори

Таблиця 2.1 Таблиця вихідної інформації системи бухгалтерського обліку нематеріальних активів та основних засобів

Ця таблиця демонструє основні параметри вихідних повідомлень, що генеруються системою, включаючи ідентифікатори, форми подання,

періодичність та користувачів, що отримують цю інформацію (усі звіти див. додаток Г, сторінка 112).

Важливою доповненням до таблиці, генерація звітів амортизації не стосуються такі активи: земельні ділянки, музейні фонди (пам'ятки культурної спадщини національного або місцевого значення, які внесені (підлягають внесенню) до Державного реєстру нерухомих пам'яток України, унікальні документи Національного архівного фонду України, які внесені (підлягають внесенню) до Державного реєстру національного культурного надбання, які зберігаються в бібліотеках згідно із Законом України «Про Національний архівний фонд та архівні установи», рідкісні та особливо цінні документи та колекції, які є частиною бібліотечних фондів, що внесені (підлягають внесенню) до Державного реєстру національного культурного надбання України, тощо, як об'єкти з невизначеним строком корисного використання), піддослідні тварини, багаторічні насадження та плодоносні рослини, що не досягли експлуатаційного віку, природні ресурси, незавершені капітальні інвестиції.

Детальний опис вихідної інформації.

Усі вихідні повідомлення системи можна поділити на дві групи :

- Звітна (сформовані звіти які можуть бути збережені та використанні у подальшій роботі);
- Інформаційна (інформаційна таблиця що є допоміжним інструментом при роботі з інформаційною системою).

Вихідна інформація, що генерується системою, використовується для:

- Підготовки бухгалтерської звітності;
- Проведення фінансового аналізу;
- Прийняття управлінських рішень;
- Планування та контролю бюджетів на утримання активів;

Періодичність та строки оприлюднення вихідної інформації:

- Щоденний звіт: публікується на початку робочого дня, для оперативного управління активами;

- Щомісячний звіт: публікується на початку місяця, для аналізу та планування діяльності;

- Квартальний звіт: публікується на початку кожного кварталу для внутрішнього аудиту та коригування стратегічного плану.

На таблиці 2.2 що наведена нижче представлена уся інформація щодо вхідних повідомлень системи.

з/ п	№ Назва вхідного повідомлення	Ідентифікато р	Форма подання	Термін і частота надходженн я	Джерело
1	Дані про нові основні засоби	DNTA_002	Електронни й документ	За потреби	Відділ бухгалтерії
2	Дані про амортизацію основних засобів	DATA_003	Електронни й документ	За потреби	Відділ бухгалтерії
3	Дані про нові нематеріальні активи	DNITA_004	Електронни й документ	За потреби	Відділ бухгалтерії
4	Дані про амортизацію нематеріальні активи	DAITA_005	Електронни й документ	За потреби	Відділ бухгалтерії
5	Дані для редагування основних засобів	DETA_006	Електронни й документ	За потреби	Відділ бухгалтерії
6	Дані для редагування амортизації основних засобів	DEATA_007	Електронни й документ	За потреби	Відділ бухгалтерії
7	Дані для редагування нематеріальних активів	DEITA_008	Електронни й документ	За потреби	Відділ бухгалтерії
8	Дані для редагування амортизації нематеріальних активів	DEAITA_009	Електронни й документ	За потреби	Відділ бухгалтерії

	х активів				
--	-----------	--	--	--	--

Таблиця 2.2 Таблиця вхідної інформації системи бухгалтерського обліку нематеріальних активів та основних засобів

Ця таблиця демонструє основні параметри вхідних повідомлень, що віддаються до системи, включаючи ідентифікатори, форми подання, термін та частоту надходження та джерела, що надають цю інформацію (усі вхідні повідомлення див. додаток В на стор. 117).

Детальний опис вхідної інформації.

Усі вхідні повідомлення системи можна поділити на групи якщо спиратись на скорочення CRUD:

- Вхідна інформація для створення бази даних;
- Вхідна інформація для збереження бази даних;
- Вхідна інформація для створення активів;
- Вхідна інформація для редагування активів.

Вихідна інформація, що генерується системою, використовується для:

- Підготовки бухгалтерської звітності;
- Створення нових активів для подальшої роботи з ними у інформаційній системі;
- Прийняття організаційних рішень по роботі з базою даних.

Щодо алгоритму вирішення задач, перед системою було поставлено декілька функціональних вимог, опираючись на які можна визначити наступні задачі:

- генерація звітів;

Нижче на рис. 2.8 зображений детальний алгоритм генерації звітів у інформаційній системі бухгалтерського обліку.

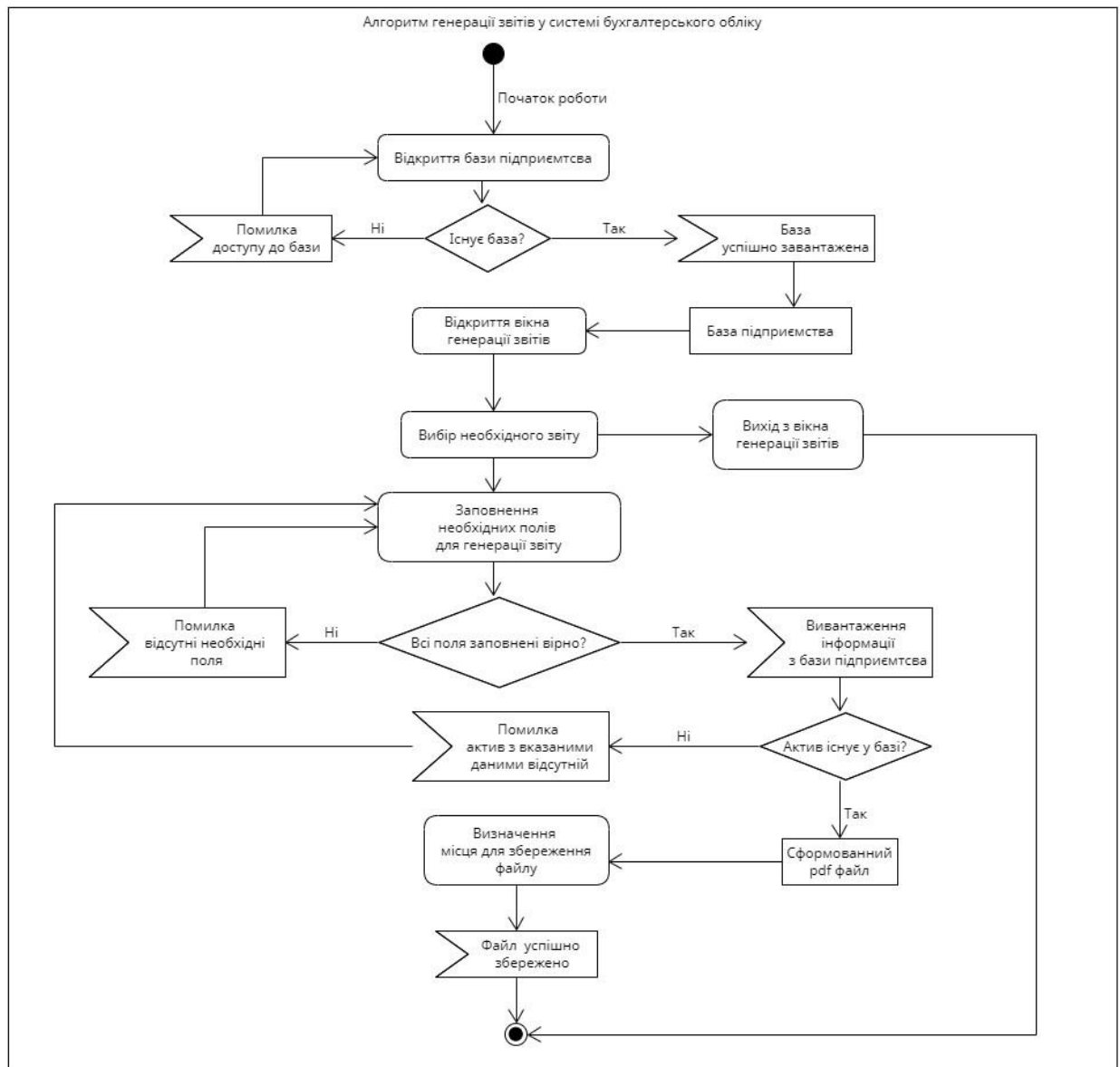


Рис. 2.8 Схема алгоритму генерації звітів інформаційної системи.

В даному алгоритмі для генерації звітів інформаційної системи можна виділити наступні кроки які виконує система:

1. Ініціалізація

- Запуск функції генерації звіту;
- Вибір користувачем параметрів звіту (тип активу, метод амортизації, період).

2. Вибір файлу для збереження звіту

- Відкриття діалогу збереження файлу;
- Вибір місця збереження імені файлу користувачем.

3. Формування запиту до бази даних

- Підготовка SQL-запиту для вибору даних активів та амортизації;
- Виконання запиту з використанням обраних користувачем параметрів.

4. Обробка результатів запиту

- Отримання результатів з бази даних;
- Обробка кожного запису для отримання необхідних даних.

5. Створення PDF-звіту

- Ініціалізація об'єкта QPdfWriter для створення PDF-файлу;
- Налаштування параметрів сторінки (розмір, поля);
- Використання QPainter для малювання тексту та таблиць у PDF.

6. Заповнення звіту даними

- Запис заголовків та загальної інформації про звіт;
- Формування таблиці з даними активів та амортизації;
- Запис кожного запису у відповідні комірки таблиці.

7. Завершення створення звіту

- Завершення роботи з QPainter;
- Повідомлення користувача про успішне збереження звіту.

Самі звіти які генерує система зображені в додатку Г сторінка 288.

- облік активів, що включає в себе створення, редагування, видалення, оновлення інформації.

Нижче на рис. 2.9-2.14 зображений детальний алгоритм обліку у інформаційній системі бухгалтерського обліку.

Рис. 2.9 Загальна схема алгоритму обліку інформаційної системи.

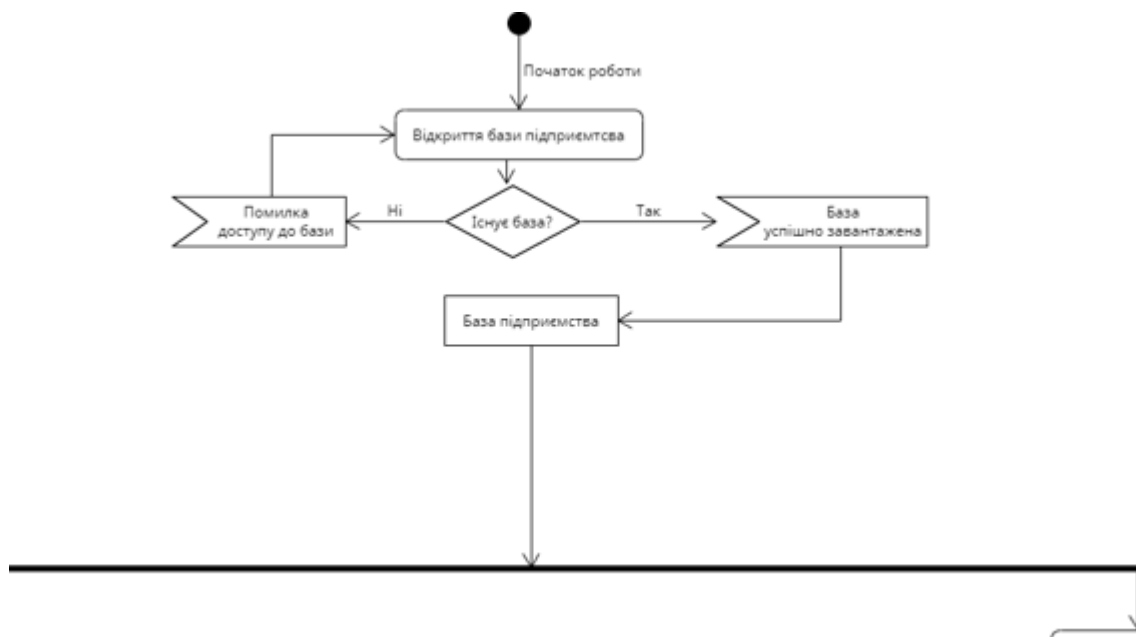


Рис. 2.10 Схема алгоритму початку обліку активів у інформаційній системі.

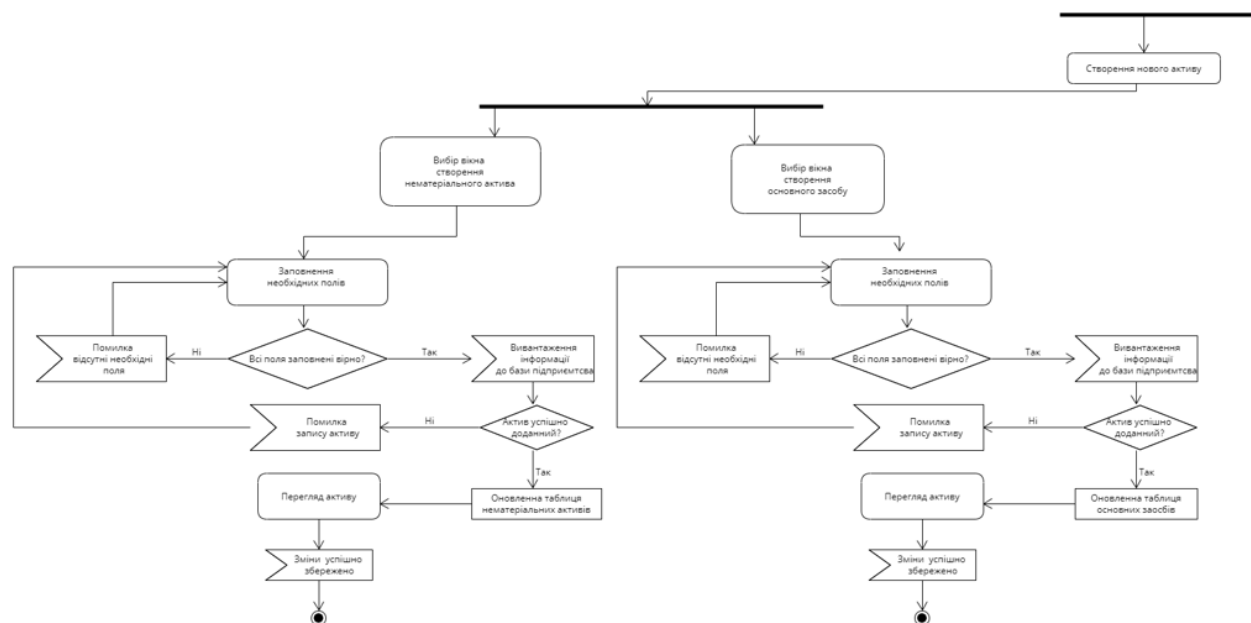


Рис. 2.11 Схема алгоритму створення активу у інформаційній системі.

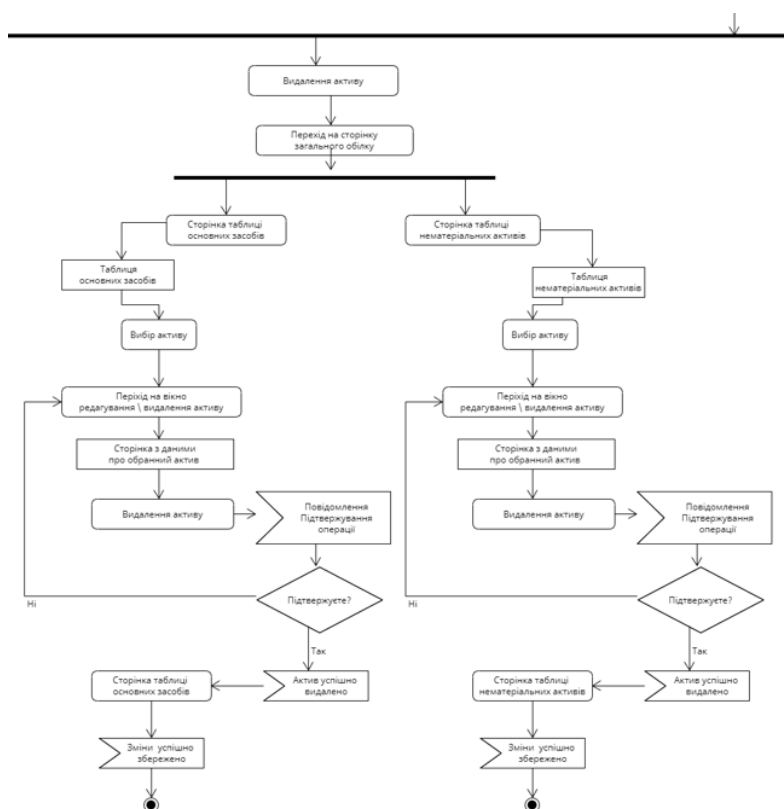


Рис. 2.12 Схема алгоритму видалення активу у інформаційній системі.

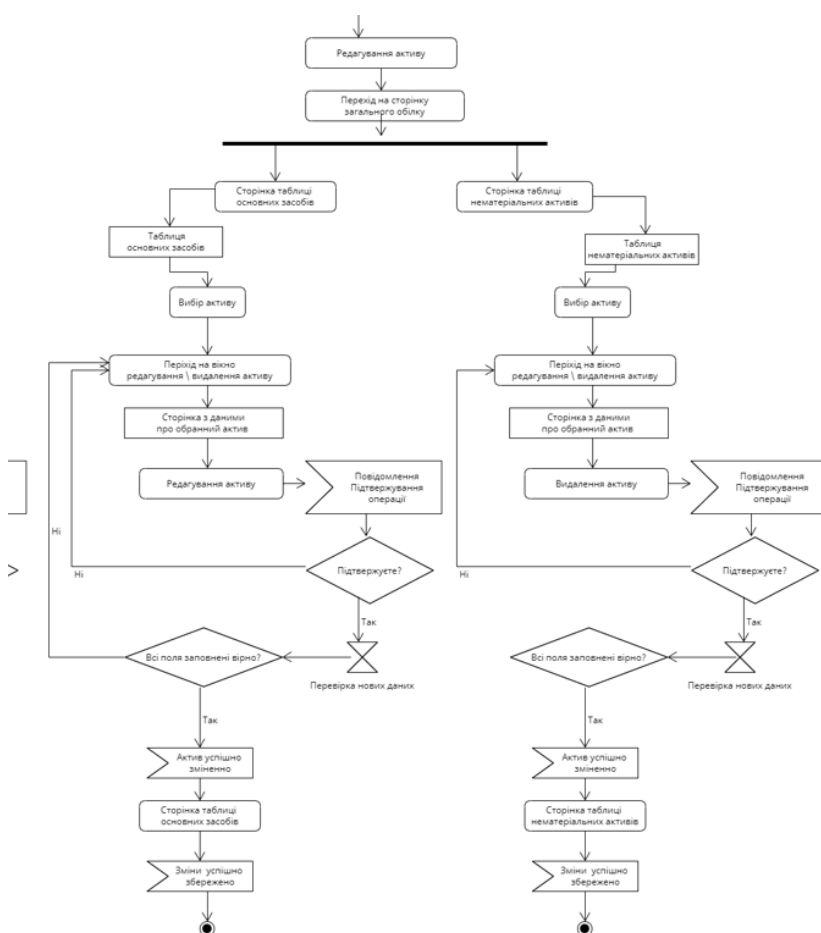


Рис. 2.13 Схема алгоритму редагування активу у інформаційній системі.

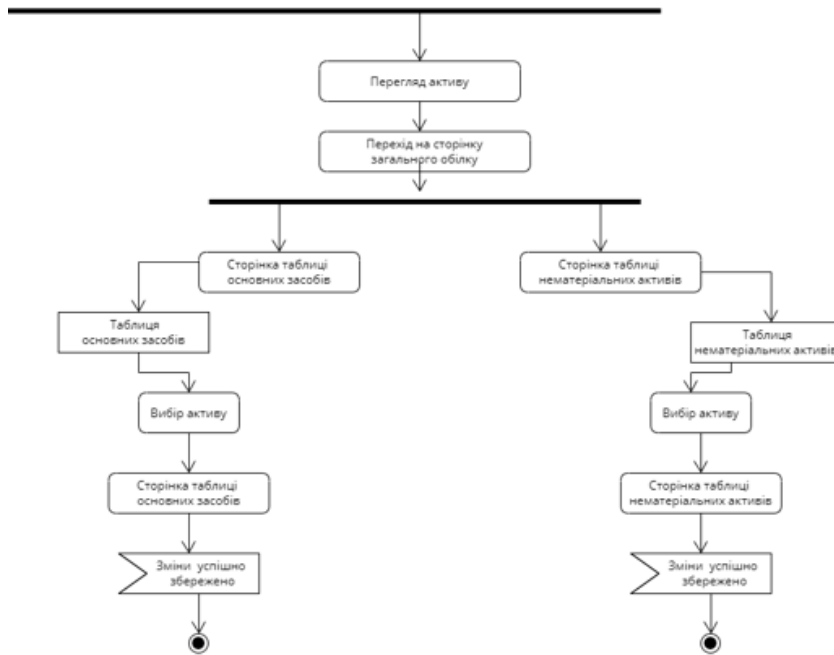


Рис. 2.14 Схема алгоритму перегляду активу у інформаційній системі.

Розглянемо алгоритм операції створення активу у інформаційній системі бухгалтерського обліку, можна виділити наступні кроки які виконує система:

1. Ініціалізація
 - Відкриття форми створення нового активу;
 - Введення користувачем необхідних даних (код, назва, тип активу тощо).
2. Перевірка вхідних даних
 - Валідація введених даних (перевірка наявності обов'язкових полів, коректність форматів).
3. Формування запиту до бази даних
 - Підготовка SQL-запиту для вставки нового запису у відповідну таблицю бази даних;
 - Виконання запиту з використанням введених користувачем даних.
4. Обробка результатів запиту
 - Перевірка результату виконання запиту;
 - Виведення повідомлення про успішне додавання активу або помилку.
5. Оновлення інтерфейсу
 - Оновлення списку активів у інтерфейсі користувача;

Нижче на рис. 2.15-2.17 зображений детальний алгоритм підрахунку амортизації у інформаційній системі бухгалтерського обліку.

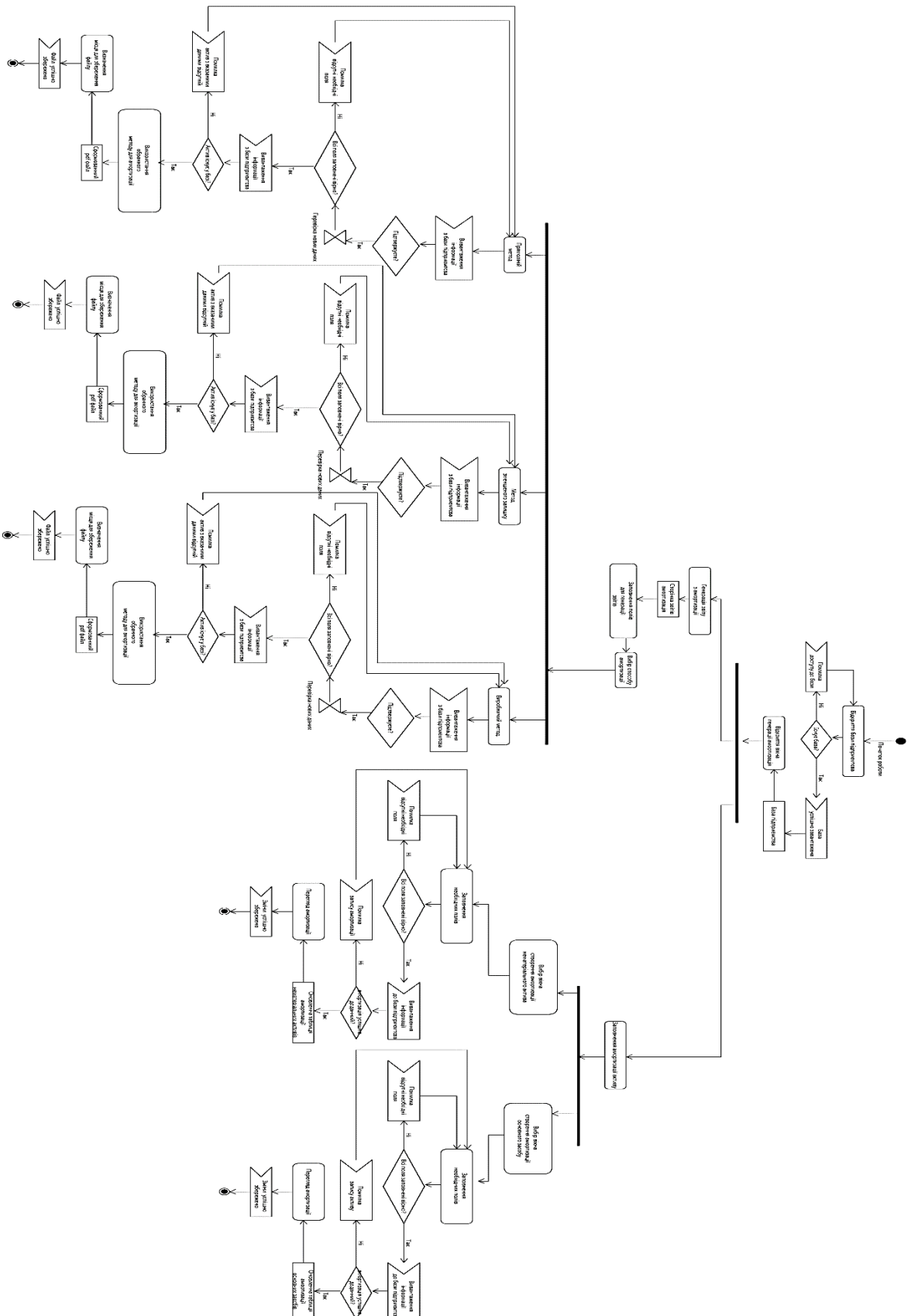


Рис. 2.15 Схема алгоритму амортизації активу інформаційної системи.

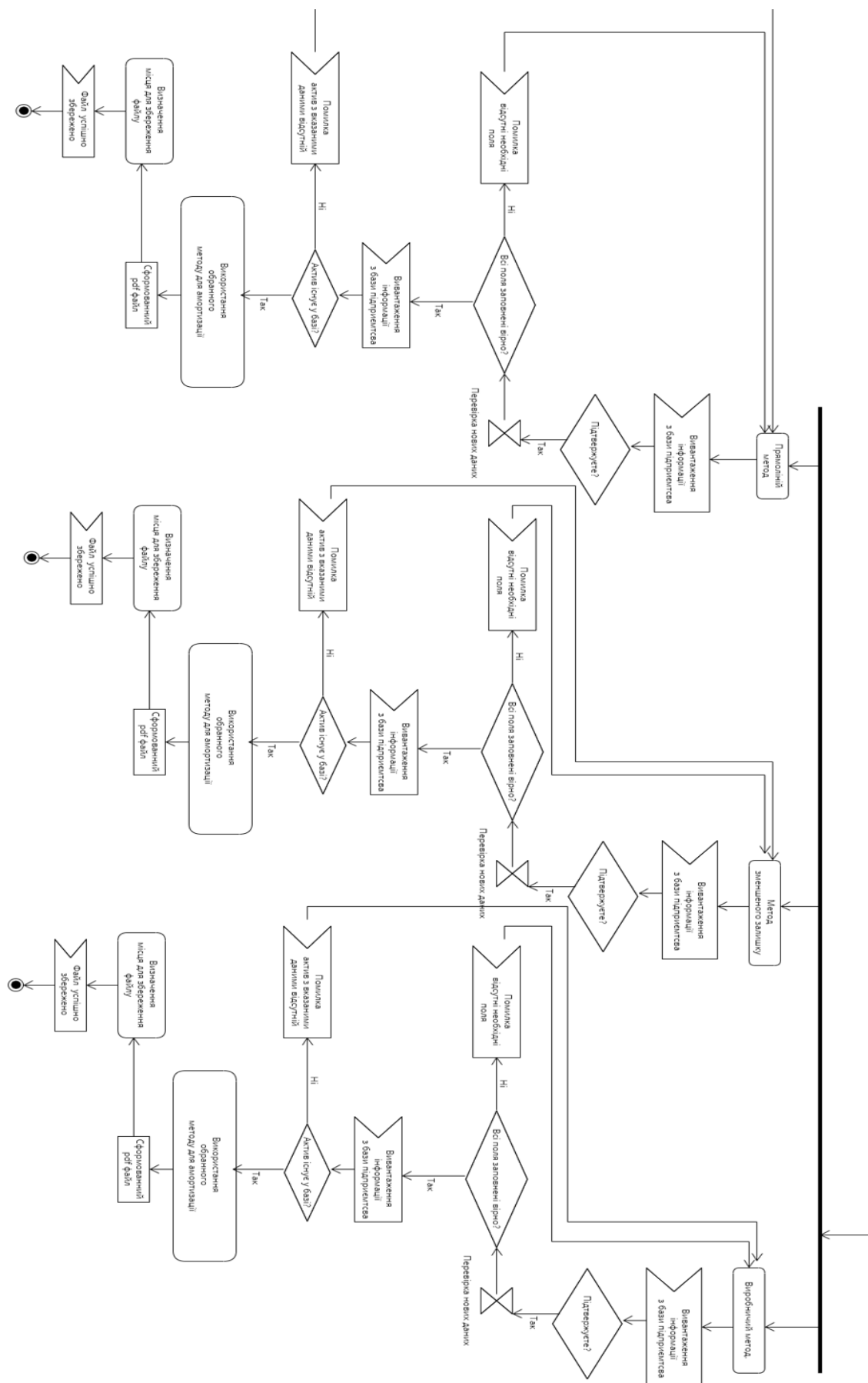


Рис. 2.16 Схема алгоритму підрахунку амортизації активу інформаційної системи.

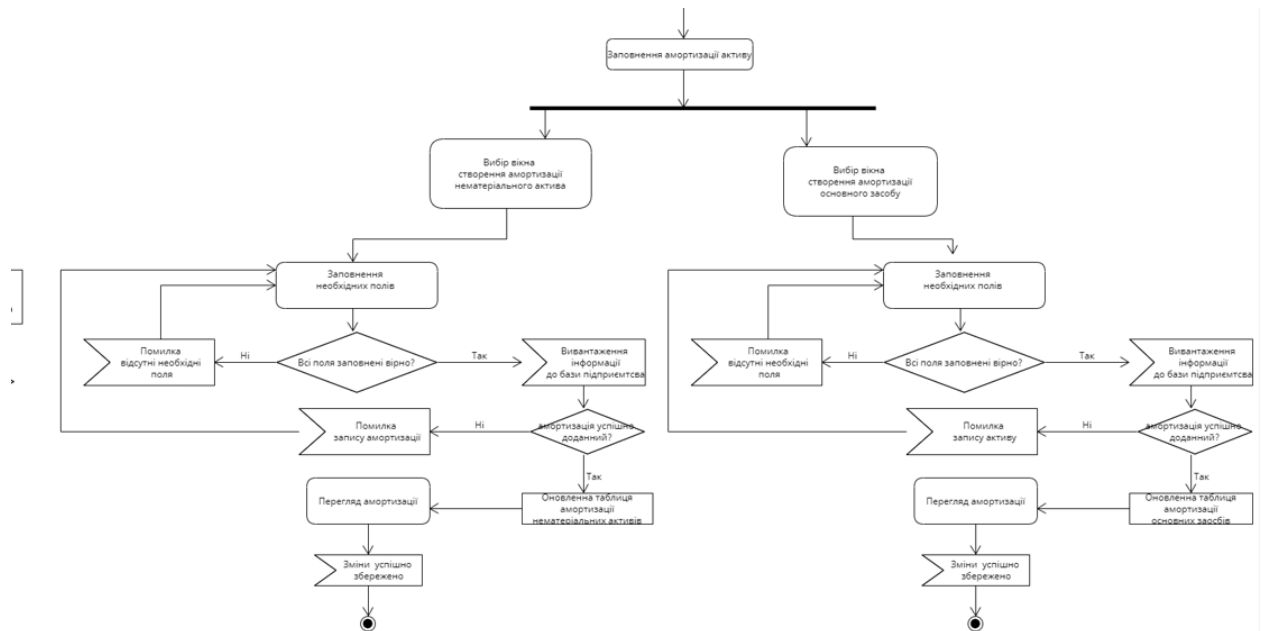


Рис. 2.17 Схема алгоритму заповнення інформації про амортизацію активу інформаційної системи.

Для розрахунку амортизації використовуються наступні вхідні дані:

- Початкова вартість (зазначимо як P);
- Ліквідаційна вартість (зазначимо як L);
- Строк корисного використання (зазначимо як n) в роках;
- Норма амортизації (зазначимо як A);
- Кількість вироблених одиниць продукції (зазначимо як U) для виробничого методу;
- Кількість років (зазначимо як n).

Загальне очікуване виробництво (зазначимо як U_{total}) для виробничого методу.

Результатом розв'язання задачі є визначення сум амортизації основних засобів за кожним з трьох методів:

- Прямолінійний метод;

Сума амортизації (D) розраховується за формулою:

$$D = \frac{P - L}{n};$$

- Метод зменшеного залишку.

Сума амортизації (D) розраховується за формулою:

$$D = (P - \sum_{i=1}^{k-1} D_i) * A ;$$

- Виробничий метод.

Сума амортизації (D) розраховується за формулою:

$$D = \frac{P - L}{U_{total}} * U;$$

Розглянемо алгоритм операції підрахунку та заповнення амортизації інформаційної системи, можна виділити наступні кроки які виконує система:

1. Ініціалізація
 - Вибір користувачем методу амортизації та періоду.
2. Формування запиту до бази даних
 - Підготовка SQL-запиту для вибору даних активів, що підлягають амортизації;
 - Виконання запиту з використанням обраних параметрів.
3. Обробка результатів запиту
 - Отримання даних про активи з бази даних;
 - Вибір методу амортизації (прямолінійний метод, метод зменшеного залишку, виробничий метод).
4. Розрахунок амортизації
 - Прямолінійний метод, формула зображена на сторінці 86, повний код реалізований додаток В, стор.117;
 - Метод зменшеного залишку формула зображена на сторінці 86, повний код реалізований додаток В, стор.117;

• Виробничий метод: формула зображена на сторінці 86, повний код реалізований додаток В, стор.117;

5. Формування даних для звіту

- Створення масиву або таблиці з розрахованими даними амортизації;
- Включення цих даних до результатів запити.

6. Створення PDF-звіту

- Ініціалізація QPdfWriter для створення PDF-файлу;
- Налаштування параметрів сторінки та використання QPainter для малювання.

7. Заповнення звіту даними

- Включення розрахованих даних амортизації до таблиці звіту;
- Запис кожного розрахунку до відповідних комірок таблиці.

8. Завершення створення звіту

- Завершення роботи з QPainter;
- Повідомлення користувача про успішне збереження звіту.

Для реалізації усіх розрахунків використовується призначення інформації, за рахунок екранних форм які користувач при роботі із системою заповнює, ми можемо сформувати окремі масиви інформації на таблиці 2.3-2.4 зображені масиви використаної інформації та масиви результативної інформації.

Масив	Ідентифікатор	Максимальна кількість записів
Масив даних про основні засоби	ADTA_001	Регулюється виділеною пам'яттю під базу даних підприємства
Масив даних про нематеріальні активи	ADITA_002	Регулюється виділеною пам'яттю під базу даних підприємства
Масив даних про амортизацію основних засобів	ADATA_003	Регулюється виділеною пам'яттю під базу даних підприємства

Масив даних про амортизацію нематеріальних активів	ADAITA_004	Регулюється виділеною пам'ятаю під базу даних підприємства
--	------------	--

Таблиця 2.3 Таблиця переліку масивів використаної інформації системи бухгалтерського обліку нематеріальних активів та основних засобів

Масив	Ідентифікатор	Максимальна кількість записів
Масив даних для формування звітності	ADRF_001	Регулюється виділеною пам'яттю під базу даних підприємства
Масив даних про основні засоби для відображення на таблиці обліку	ADTATV_002	Регулюється виділеною пам'яттю під базу даних підприємства
Масив даних про амортизацію основних засобів для відображення на таблиці обліку	ADATATV_003	Регулюється виділеною пам'яттю під базу даних підприємства
Масив даних про нематеріальні активи для відображення на таблиці обліку	ADITATV_004	Регулюється виділеною пам'яттю під базу даних підприємства
Масив даних про амортизацію нематеріальних активів для відображення на таблиці обліку	ADAITATV_005	Регулюється виділеною пам'яттю під базу даних підприємства

Таблиця 2.4 Таблиця переліку масивів результативної інформації системи бухгалтерського обліку нематеріальних активів та основних засобів

Розробка та впровадження інформаційної системи обліку основних засобів та нематеріальних активів дозволить значно підвищити ефективність управління активами, забезпечити прозорість обліку та сприятиме прийняттю обґрунтованих управлінських рішень.

2.3 Моделювання інформаційної системи

Моделювання є універсальним засобом для вивчення складних систем за допомогою їх спрощення шляхом абстрагування. Моделе-зорієнтований інжиніринг є сучасним перспективним підходом до розробки складних систем,

зокрема, інформаційних, і передбачає, що основними розроблюваними артефактами є моделі, з яких генеруються інші системні артефакти.

У цьому підрозділі розглянемо такі пункти: "Моделювання поведінки системи", "Моделювання структури системи", "Розподіл вимог за компонентами системи".

Вимоги та модель поведінки системи для інформаційної системи (ІС) для ведення обліку основних засобів та нематеріальних активів у бухгалтерському обліку включають:

1. Реєстрація користувачів:

Можливість реєстрації нових користувачів, включаючи бухгалтерів та адміністраторів системи.

Користувачі можуть вводити свої особисті дані та створювати облікові записи для доступу до системи.

2. Аутентифікація:

Після реєстрації користувачі можуть увійти в систему за допомогою ідентифікаційних даних, таких як ім'я користувача та пароль.

3. Управління активами:

Можливість додавання, редагування та видалення інформації про основні засоби та нематеріальні активи.

Функціональність включає в себе введення даних про придбання, амортизацію, вартість та інші атрибути активів.

4. Обробка даних:

Автоматизована обробка та облік даних про активи для підготовки фінансової звітності та аналізу стану підприємства.

5. Аналіз та звітність:

Надання звітної та аналітичної інформації про стан основних засобів та нематеріальних активів, їх амортизацію, вартість та інші показники.

Можливість генерації різноманітних звітів для внутрішнього та зовнішнього використання.

6. Захист даних:

Надійні механізми захисту конфіденційності, цілісності та доступності даних, включаючи шифрування, контроль доступу та резервне копіювання.

7. Інтеграція з іншими системами:

Можливість інтеграції з іншими бухгалтерськими системами або ERP-системами для обміну даними та забезпечення сумісності.

8. Аудит:

Функціонал для ведення журналу аудиту та моніторингу дій користувачів системи для забезпечення контролю та відповідності процесів.

9. Інтерфейс користувача:

Зручний та інтуїтивно зрозумілий інтерфейс для користувачів системи, який спрощує введення та обробку даних.

10. Підтримка:

Надання технічної підтримки та навчання користувачів для ефективного використання системи.

Модель поведінки системи включає взаємодію з користувачами, обробку даних, управління активами та забезпечення захисту та надійності даних. Також ця модель безпосередньо включає відображення прецедентів, розглянемо основні прецеденти до системи.

Прецедент 1: Додавання нового активу

- Опис: Користувач може додати новий актив до системи, заповнюючи необхідні поля, такі як назва, тип активу (нематеріальний актив або основний засіб), дата придбання, вартість та інші атрибути.

- Актори: Бухгалтер, адміністратор системи.

- Передумови: Користувач має відповідні права доступу.

- Результат: Новий актив збережений у базі даних.

Прецедент 2: Редагування інформації про актив

- Опис: Користувач може редагувати інформацію про існуючий актив, вносячи зміни до полів, таких як вартість, стан, місцезнаходження тощо.

- Актори: Бухгалтер, адміністратор системи.

- Передумови: Користувач має відповідні права доступу.

- Результат: Зміни збережені в базі даних.

Прецедент 3: Видалення активу

- Опис: Користувач може видалити актив із системи.
- Актори: Адміністратор системи.
- Передумови: Користувач має відповідні права доступу.
- Результат: Актив видалено з бази даних.

Прецедент 4: Автоматичний розрахунок амортизації

• Опис: Система автоматично розраховує амортизацію активів на основі заданих параметрів, таких як метод амортизації та термін корисного використання.

- Актори: Бухгалтер, фінансовий менеджер.
- Передумови: Активи введені до системи з усіма необхідними атрибутами.
- Результат: Розрахунок амортизації збережено у системі.

Прецедент 5: Переоцінка активів

• Опис: Користувач може провести переоцінку активів, вносячи нову справедливую вартість та дату переоцінки.

- Актори: Бухгалтер, фінансовий менеджер.
- Передумови: Користувач має відповідні права доступу.
- Результат: Оновлена вартість активів збережена у системі.

Прецедент 6: Оцінка зменшення корисності активів

• Опис: Система проводить автоматичну оцінку зменшення корисності активів, аналізуючи поточну ринкову вартість та інші фактори.

- Актори: Бухгалтер, фінансовий менеджер.
- Передумови: Дані про активи введені та актуалізовані.
- Результат: Оцінка зменшення корисності збережена у системі.

Прецедент 7: Генерація звітів

• Опис: Користувач може генерувати різні види звітів про стан активів, амортизацію, переоцінку, зменшення корисності тощо.

- Актори: Бухгалтер, фінансовий менеджер, керівник.
- Передумови: Дані про активи введені та актуалізовані.
- Результат: Згенерований звіт доступний для перегляду або експорту.

Прецедент 8: Візуалізація даних

• Опис: Система надає можливість візуалізувати дані у вигляді графіків та діаграм для полегшення аналізу.

- Актори: Бухгалтер, фінансовий менеджер, керівник.
- Передумови: Дані про активи введені та актуалізовані.
- Результат: Візуалізація даних доступна для перегляду.

Прецедент 9: Інтеграція з ERP-системою

• Опис: Система синхронізує дані про активи з існуючою ERP-системою підприємства.

- Актори: Адміністратор системи.
- Передумови: Налаштовані параметри інтеграції.
- Результат: Дані успішно синхронізовані між системами.

Прецедент 10: Інтеграція з бухгалтерською системою

• Опис: Система забезпечує обмін даними з бухгалтерською системою для автоматичного оновлення фінансової звітності.

- Актори: Адміністратор системи.
- Передумови: Налаштовані параметри інтеграції.
- Результат: Дані успішно синхронізовані між системами.

Прецедент 11: Управління правами доступу

• Опис: Адміністратор може налаштувати права доступу для різних користувачів та ролей у системі.

- Актори: Адміністратор системи.
- Передумови: Користувач має відповідні права доступу.
- Результат: Права доступу налаштовані та збережені.

Прецедент 12: Журнал дій користувачів

• Опис: Система веде журнал всіх дій користувачів для забезпечення аудиту та безпеки.

- Актори: Адміністратор системи.
- Передумови: Система працює в штатному режимі.
- Результат: Журнал дій користувачів доступний для перегляду.

Прецедент 13: Інтерактивні підказки та довідкова система

- **Опис:** Система надає інтерактивні підказки та довідкову систему для полегшення користування.

- **Актори:** Всі користувачі системи.
- **Передумови:** Система працює в штатному режимі.
- **Результат:** Підказки та довідкова система доступні користувачам.

Прецедент 14: Технічна підтримка

- **Опис:** Користувачі можуть звертатися за технічною підтримкою через вбудовану функцію зворотного зв'язку.

- **Актори:** Всі користувачі системи.
- **Передумови:** Система працює в штатному режимі.
- **Результат:** Технічна підтримка надає допомогу користувачам.

Ці функціональні можливості забезпечують повний цикл обліку та управління нематеріальними активами та основними засобами, відповідаючи вимогам сучасного бізнесу та забезпечуючи ефективність і точність у бухгалтерському обліку.

Взаємодія дійових осіб та функціональних можливостей системи відображена на діаграмі прецедентів дивитись на рис. 2.18.



Рис. 2.18 Діаграма прецедентів для інформаційної системи.

Діаграма прецедентів документує функціональні вимоги у вигляді варіантів використання як типової взаємодії системи з акторами. Склад та обсяг системи, що її подає діаграма прецедентів, має відповідати функціональним вимогам до системи та постановці задачі, що їх було розроблено в попередніх підрозділах.

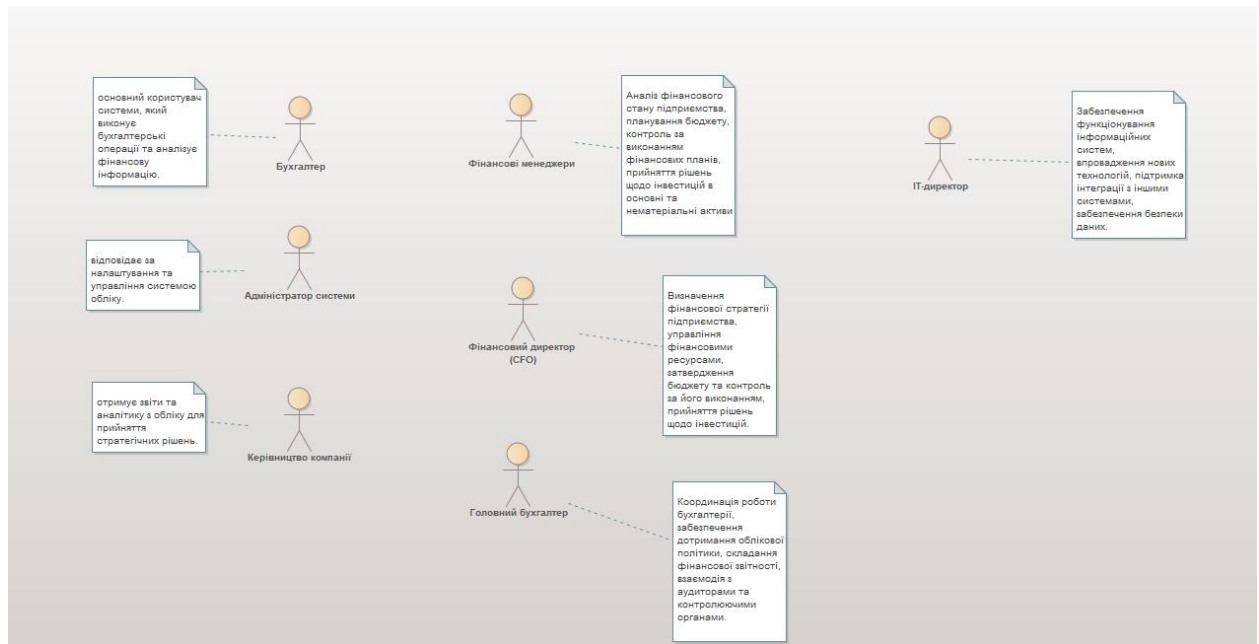


Рис. 2.19 Діаграма «Акторів» для інформаційної системи.

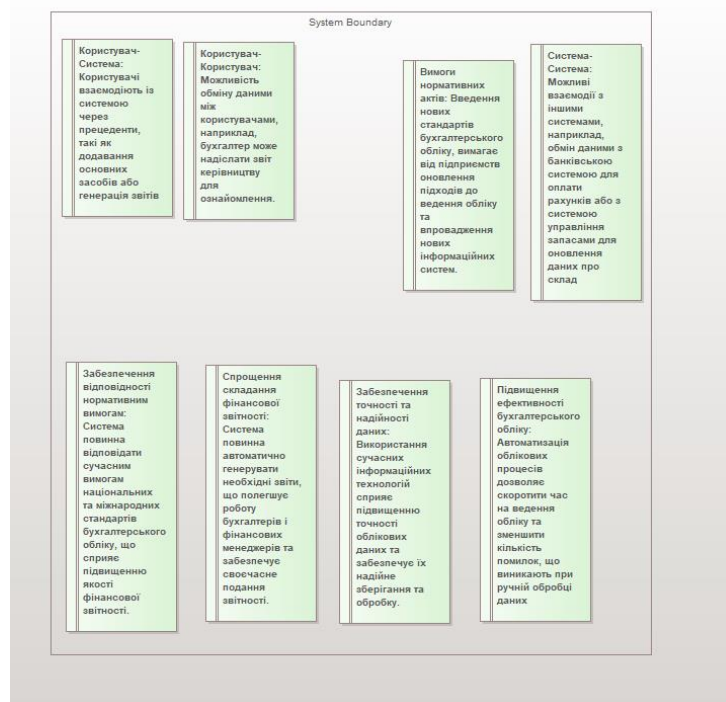


Рис2.20 Діаграма вимог до прецедентів інформаційної системи.

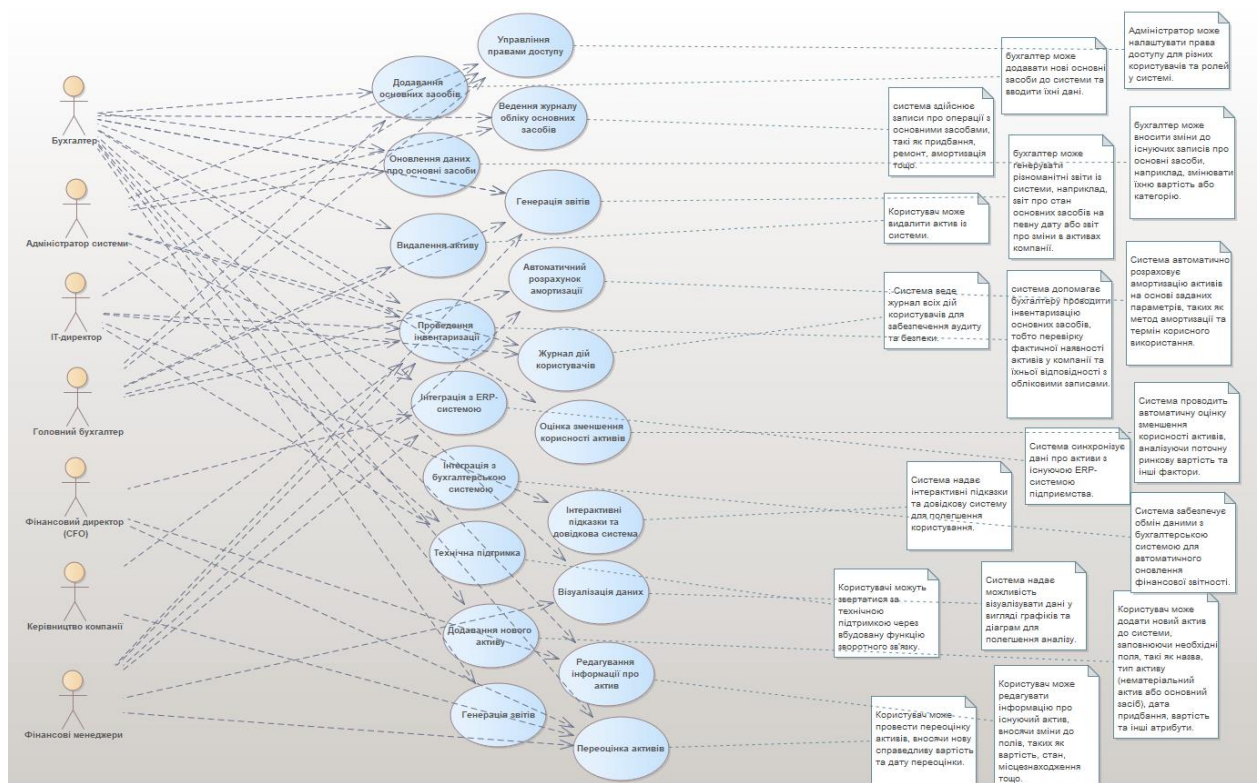


Рис. 2.21 Діаграма взаємодії акторів та прецедентів для інформаційної системи.

На діаграмі прецедентів (Use Case) для інформаційної системи бух. обліку відображені ключові функціональні можливості системи та їх взаємодія з головними користувачами. Діаграма охоплює різні сценарії, які можуть відбуватися між користувачами (бухгалтерами, адміністраторами)

Також зроблено матриці відповідності вимог і прецедентів, які продемонстровані на рис. 2.22

Source: Прецеденти (функціональні можливості)	Type: UseCase	Link Type: Dependency	Profile: <None>	Refresh
Target: Actors	Type: Actor	Direction: Both	Overlays: <None>	Options

Source	Адміністратор сист	Адмін. Бухгалтер	Адмін. Головний бухгалтер	Адмін. IT-директор	Адмін. Керівництво компанії	Адмін. Фінансовий директор	Адмін. Фінансовий менеджер
Прецеденти (функціональні можливості): Переоцінка активів	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Управління правами доступу	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Автоматичний розрахунок амортизації	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Ведення журналу обліку основних засобів	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Видалення активу	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Візуалізація даних	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Генерація звітів	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Генерація звітів	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Додавання нового активу	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Додавання основних засобів	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Журнал дій користувачів	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Інтеграція з ERP-системою	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Інтеграція з бухгалтерською системою	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Інтерактивні підказки та довідкова система	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Оновлення даних про основні засоби	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Оцінка зменшення корисності активів	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Проведення інвентаризації	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Редагування інформації про актив	↑	↑	↑	↑	↑	↑	↑
Прецеденти (функціональні можливості): Технічна підтримка	↑	↑	↑	↑	↑	↑	↑

Рис. 2.22 – Матриці відповідності вимог і прецедентів.

Діаграма відображає складність та багатозадачність інформаційної системи, підкреслюючи необхідність її інтеграції з різними обліковими системами. З її допомогою можна зрозуміти, як система буде сприяти автоматизації бізнес-процесів та підвищенню ефективності управління бухгалтерії, водночас забезпечуючи високий рівень задоволеності клієнтів через оптимізований інтерфейс та зручні функції.

Для деяких прецедентів наводимо діаграми послідовності у відповідності до сценаріїв варіантів використання в обсязі, достатньому для виокремлення всіх об'єктів (компонентів) проєктованої системи (рис 2.23-2.27).

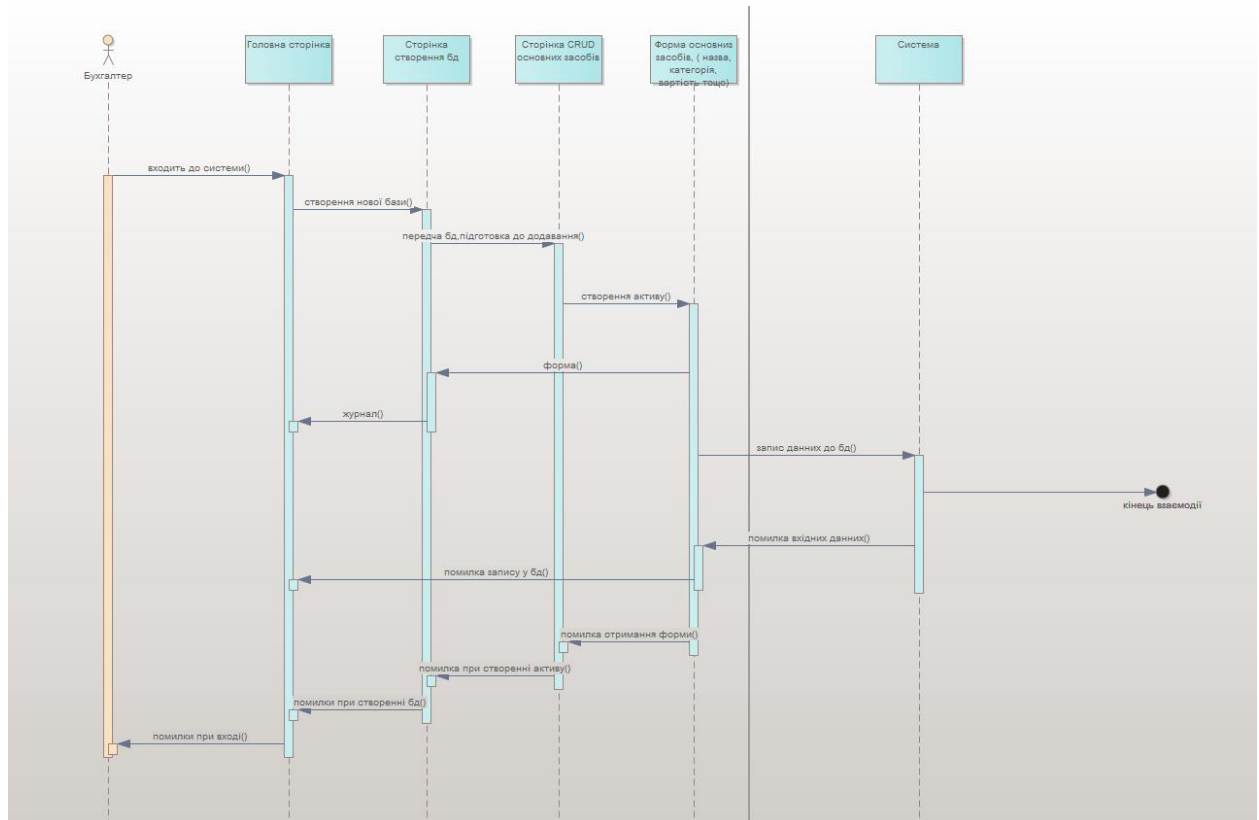


Рис 2.23 Діаграма послідовності «Ведення журналу обліку Case Model».

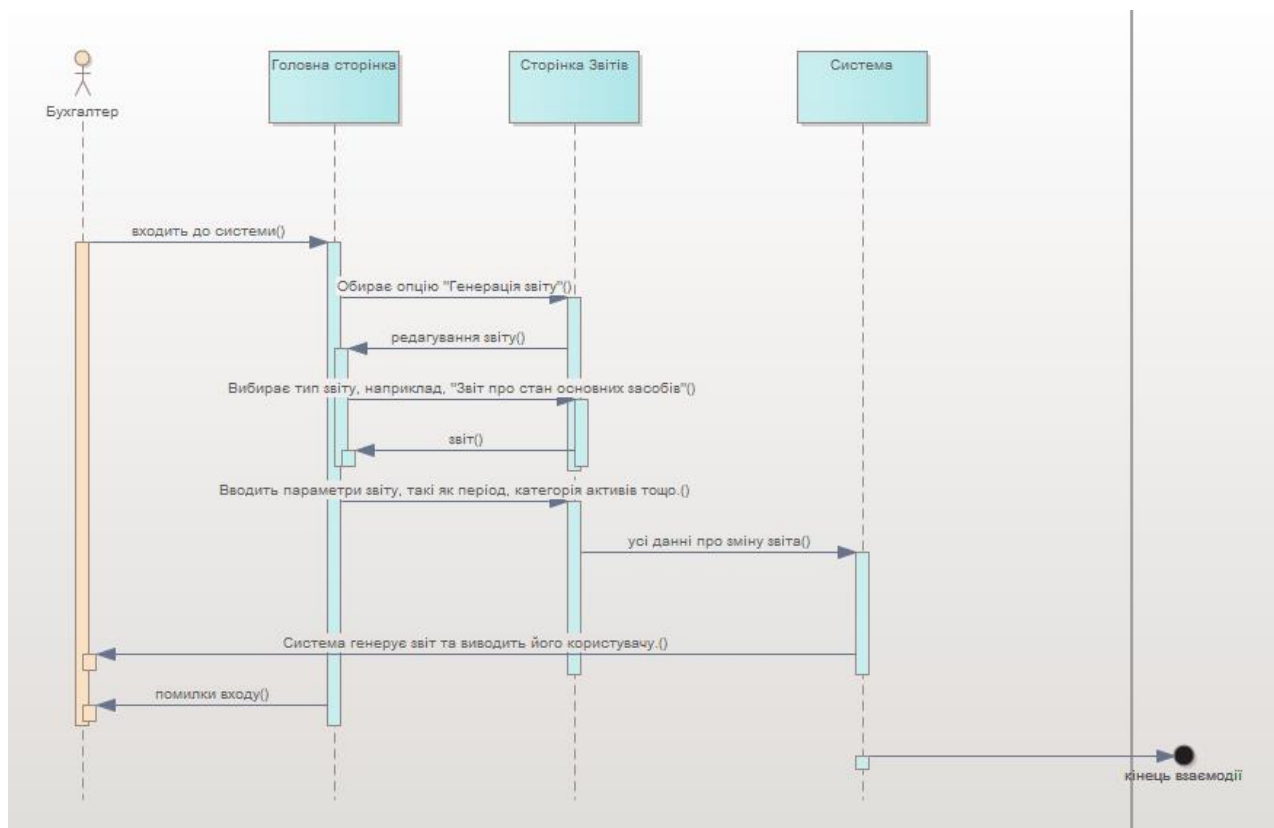


Рис 2.24 Діаграма послідовності «Генерація звітів Case Model».

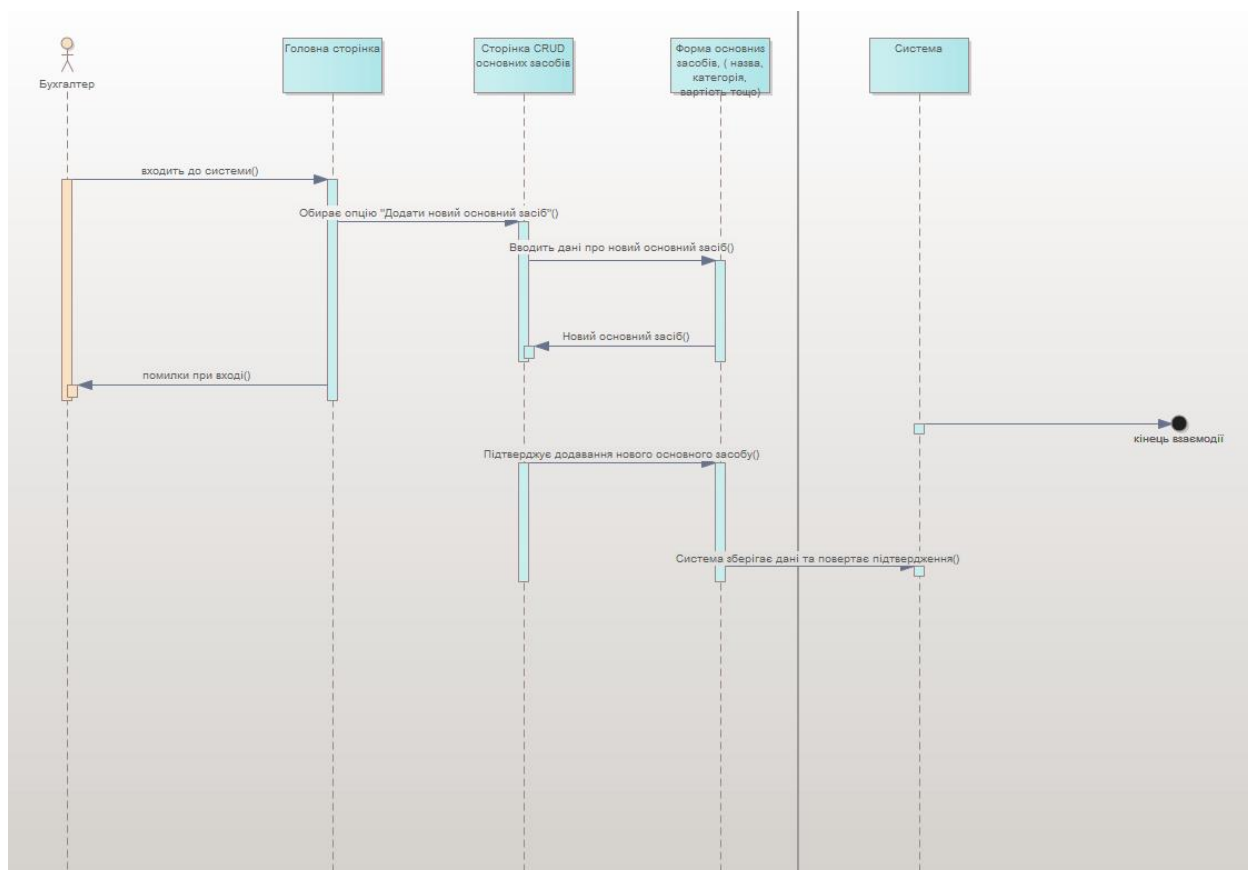


Рис 2.25 Діаграма послідовності «Додавання основних засобів Case Model».

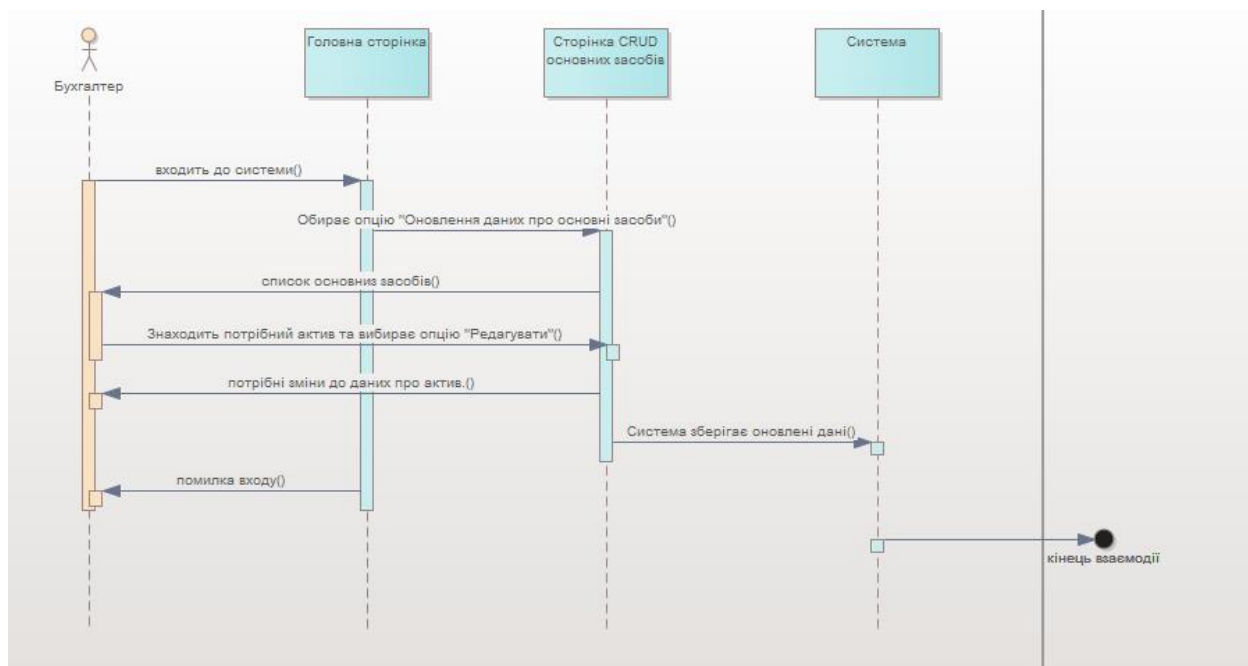


Рис 2.26 Діаграма послідовності «Оновлення даних про основних засобів Case Model».

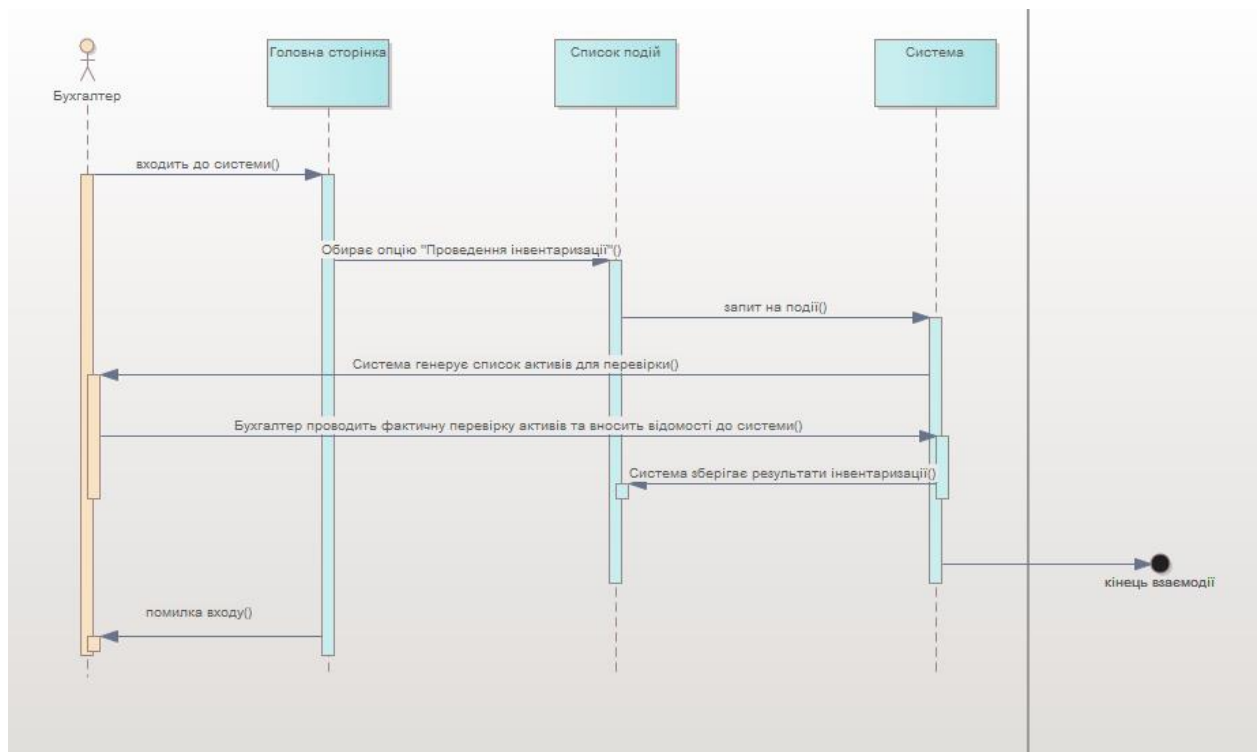


Рис 2.27 Діаграма послідовності «Проведення інвентаризації Case Model».

Основні кроки діаграми послідовності

• Діаграма послідовності «Ведення журналу обліку основних засобів» моделі випадку використання (Case Model) має такі основні кроки:

- Користувач входить в систему:
- Користувач аутентифікується в системі, вводячи свої ідентифікаційні дані.
- Обрання опції "Ведення журналу обліку основних засобів":
- Користувач обирає необхідну опцію з меню або вкладки для початку ведення журналу обліку.
- Введення даних про основний засіб:
- Користувач вводить дані про новий основний засіб, такі як назва, тип, вартість, дата придбання тощо.
- Збереження даних:
- Система зберігає введені дані про основний засіб у базі даних.
- Автоматичне оновлення журналу:

- Система автоматично оновлює журнал обліку основних засобів з урахуванням введених користувачем даних.

- Виведення повідомлення про успішне збереження:

- Система повідомляє користувача про успішне збереження даних та готовність до подальших дій.

Для систем з великою кількістю паралельних процесів наводимо опис поведінки у формі діаграм діяльності.

Модель структури системи подаємо діаграмами класів UML, що відображають статичну структуру класів системи і зв'язків між ними. Для забезпечення прозорості і зрозумілості моделі на різних діаграмах на моделі подано:

- Пакети класів і відношення між ними.
- Підмножину класів і відношення між ними.
- Підмножину класів з атрибутами та операціями класів.
- На діаграмах мають бути подані класи основних стереотипів:
- Класи-сутності, що використовуються для моделювання даних і поведінки з довгим життєвим циклом.
- Граничні класи, за допомогою яких моделюють інтерфейси системи на високому рівні.
- Класи керування (класи-менеджери), що створюються для моделювання послідовної поведінки одного або кількох прецедентів і координації подій, що реалізують закладену в ці прецеденти поведінку.

Це відмінний приклад структурування компонентів для інформаційної системи бухгалтерського обліку. На основі цих компонентів можна побудувати повну систему, яка забезпечує ефективне ведення обліку основних засобів та нематеріальних активів.

Компонент взаємодії з користувачем грає ключову роль у створенні зручного та інтуїтивно зрозумілого інтерфейсу для користувачів системи. Завдяки різноманітним інтерфейсам, таким як веб-інтерфейс, мобільні додатки та десктопні програми, користувачі можуть з легкістю взаємодіяти з системою та виконувати різноманітні завдання бухгалтерського обліку.

Компонент аналізу даних відповідає за обробку та аналіз великих обсягів даних про основні засоби та нематеріальні активи. Використовуючи різні методи аналізу, цей компонент допомагає здійснювати інформовані бізнес-рішення на основі надійної аналітики.

Компонент адміністрування бази даних відповідає за управління та адміністрування бази даних, забезпечуючи її надійну роботу та доступність. Цей компонент включає в себе створення резервних копій, відновлення даних та моніторинг продуктивності бази даних.

Компонент моніторингу та аналізу продуктивності відповідає за стеження за роботою системи та виявлення проблем з продуктивністю. Шляхом моніторингу та аналізу, цей компонент допомагає вирішувати проблеми та підтримувати оптимальну роботу системи.

Інтерфейси користувача відіграють ключову роль у забезпеченні зручного та ефективного взаємодії з інформаційною системою для бухгалтерського обліку. Їх розробка повинна здійснюватися з урахуванням потреб користувачів та вимог до функціональності системи. Зв'язуючи з вищевказаними можливостями інтеграції з іншими системами та сервісами, діаграми визначення блоків для подання внутрішньої структури системи у вигляді ієрархії блоків різних типів є критичним інструментом проєктування та розробки інформаційної системи для бухгалтерського обліку. Ці діаграми дозволяють:

Визначення компонентів системи: Шляхом розгляду різних типів блоків (наприклад, інтерфейси користувача, модулі обробки даних, інтеграційні шари) можна чітко визначити компоненти системи, їх ролі та взаємозв'язки.

Уточнення функціональності: Розділення системи на окремі блоки допомагає уточнити функціональні обов'язки кожного компонента. Наприклад, окремі блоки можуть бути призначені для обробки платежів, взаємодії з іншими системами, а також для взаємодії з користувачем.

Опис архітектури: Ієрархічна структура блоків дозволяє описати архітектуру системи на рівні блоків, які можуть бути легко ідентифіковані та зрозумілі як частина загальної системи.

Визначення точок інтеграції: Розгляд різних типів блоків дозволяє визначити точки інтеграції з іншими системами та сервісами. Це допомагає у визначенні потреб у взаємодії та уточненні вимог до інтеграційних рішень.

Планування розробки: діаграми визначення блоків допомагають розробникам створити план розробки системи, розподіливши роботу на окремі компоненти та блоки, що спрощує процес розробки та тестування.

Таким чином, діаграми визначення блоків є важливим інструментом у проєктуванні та розробці інформаційної системи для бухгалтерського обліку, допомагаючи зрозуміти структуру системи, уточнити функціональність та планувати розробку та інтеграцію з іншими сервісами.

Також, наведено приклад стислої роботи класів в інформаційній системі бух. обліку (рис. 2.28) в системі моделює структуру з точки зору головних об'єктів, які в ній присутні, їхні взаємовідносини та властивості.

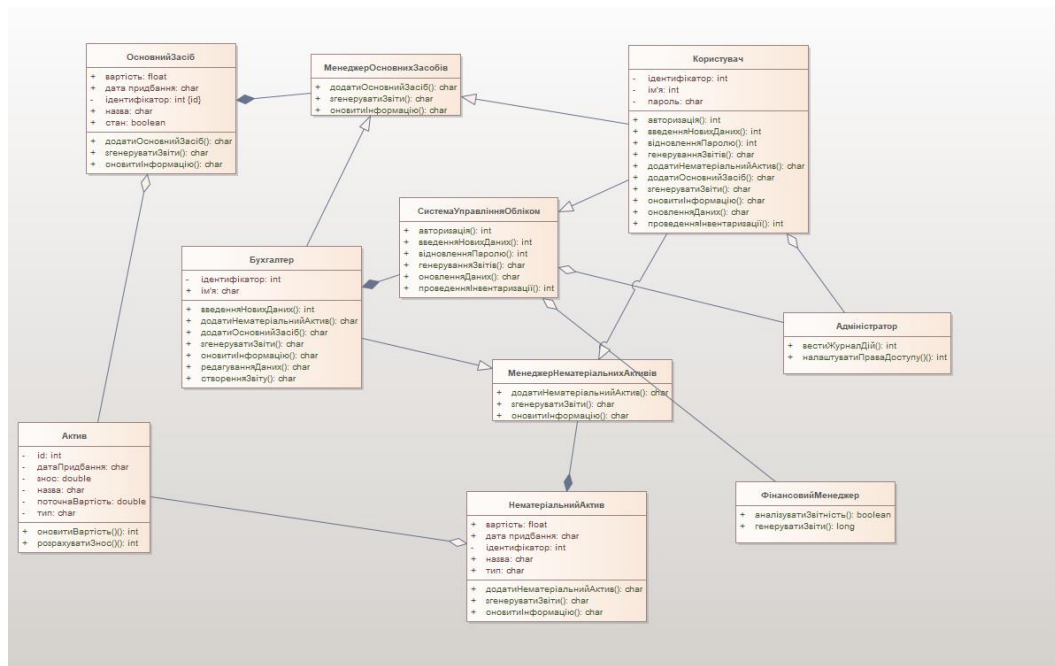


Рис. 2.28 Діаграма класів.

Розподіл вимог за компонентами системи виконується за допомогою діаграми визначення блоків для подання внутрішньої структури системи у вигляді ієрархії блоків різних типів відображає компоненти, які складають систему та їхні взаємозв'язки. Для нашої інформаційної системи для бух. обліку така діаграма відображатиме різні рівні абстракції та компоненти, що складають систему. На верхньому рівні ієрархії блоків можуть бути такі

компоненти, як "Система ведення обліку", "Модуль обліку основних засобів", "Модуль нематеріальних активів". Кожен з цих блоків має свої власні підкомпоненти та залежності, що відображають структуру та взаємозв'язки між різними частинами системи (див. рис. 2.29).

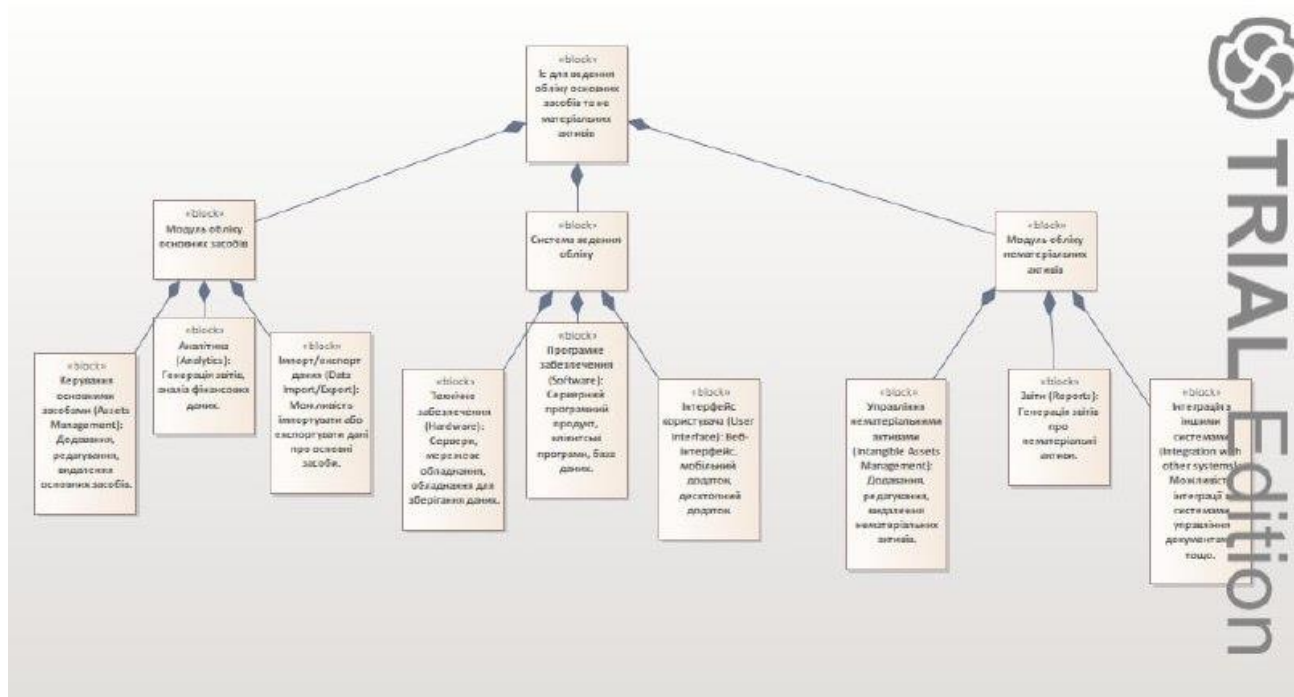


Рис. 2.29 – Діаграма визначення блоків для подання внутрішньої структури системи (BDD).

На рисунку 2.30 зображено приклад роботи API компонент IC. На цій діаграмі показано фрагмент залежності компонентів системи, для роботи з основними засобами та нематеріальними активами та редагування даних в базі даних. API приймає запит з вказаними даними і це все надсилається в базу та виводиться на користувальницькому інтерфейсі. Кожен з цих блоків може мати внутрішні компоненти, які відображають конкретні функціональні частини. Зв'язки між цими блоками можуть бути визначені через порти, які відповідають за обмін даними та виклики функцій між різними компонентами.

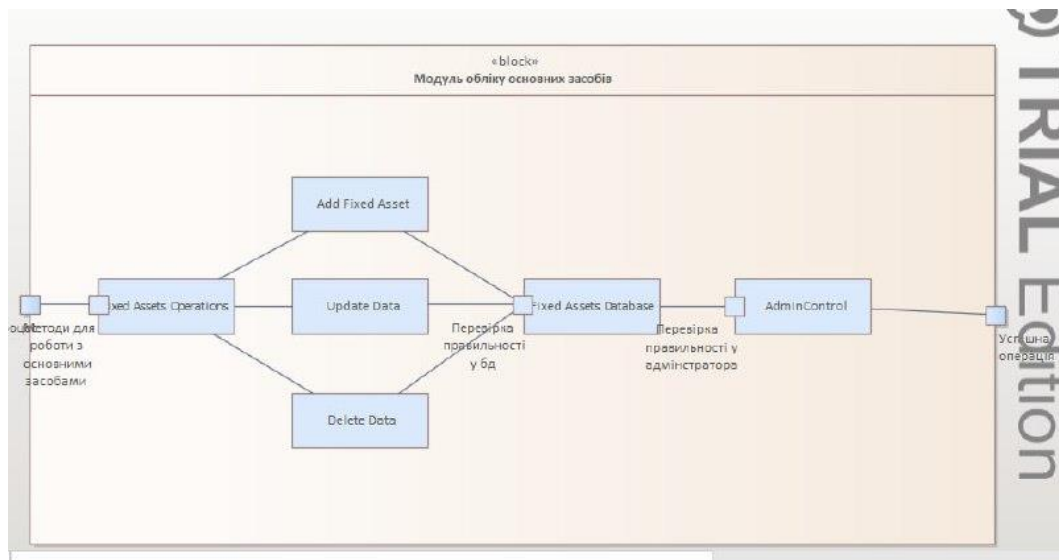


Рис. 2.30 – Процес обліку основних засобів(ІВД).

Спочатку система перебуває в стані *idle* (бездіяльності), поки не отримає запит на доступ від користувача. Після отримання запиту, система переходить в стан перевірки (верифікація), де вона перевіряє, чи доступне НД. Якщо НД доступне, система переходить в стан «Активний», де вона здійснює запит користувача. Якщо НД не доступне, система повертається в стан «Відмова» і відхиляє запит користувача. У стані «Активний», система може отримати дві події: скасування запиту або підтвердження. У випадку скасування запиту, система повертається в стан *idle*. У випадку підтвердження, система переходить в стан виконання запиту, де вона завершує запит користувача і повертається в стан *idle*.

Трасування вимог є важливим кроком у процесі розробки програмного забезпечення. Потрібно для визначення взаємозв'язків між вимогами, прецедентами, тестовими кейсами та елементами користувацького інтерфейсу і допомагає гарантувати, що реалізація функціональності програмного забезпечення є повною і відповідає вимогам користувачів та бізнес-цілям.

Трасування вимог – це процес відстеження вимог від початку до реалізації та тестування. В інформаційній системі трасування системних вимог допомагає забезпеченню відповідності функціональності системи потребам користувачів і бізнес-цілям проєкту.

Характеристика використання діаграм відстеження та матриць взаємозв'язків для нашої ІС:

Діаграма трасування вимог дозволяє візуалізувати взаємозв'язки між різними елементами системи та вимогами до неї. Для інформаційної системи для ведення обліку основних засобів і нематеріальних активів у бухгалтерському обліку можна використовувати такі типи зв'язків:

Зв'язок "Trace":

Трасування вимог до системи (функціональних або нефункціональних) до відповідних прецедентів або сценаріїв використання. Наприклад, вимога на ведення журналу інвентаризації може бути пов'язана з прецедентом "Внесення нового основного засобу".

Зв'язок "Realization":

Показує, які компоненти або модулі програмного забезпечення відповідають за реалізацію конкретних прецедентів або вимог. Наприклад, клас "InventoryManagement" може реалізовувати прецедент "Редагування даних про основний засіб".

Зв'язок "Abstraction":

Показує абстрактні класи або інтерфейси, які пов'язані з конкретними компонентами системи, які їх реалізують. Наприклад, абстрактний клас "Asset" може бути пов'язаний з класом "FixedAsset", який представляє основний засіб.

Використання цих зв'язків допомагає забезпечити чітке розуміння того, як кожна вимога впливає на функціональність системи та які компоненти системи відповідають за її реалізацію. Такий підхід дозволяє покращити якість та відповідність розробленої інформаційної системи вимогам користувачів і бізнесу.

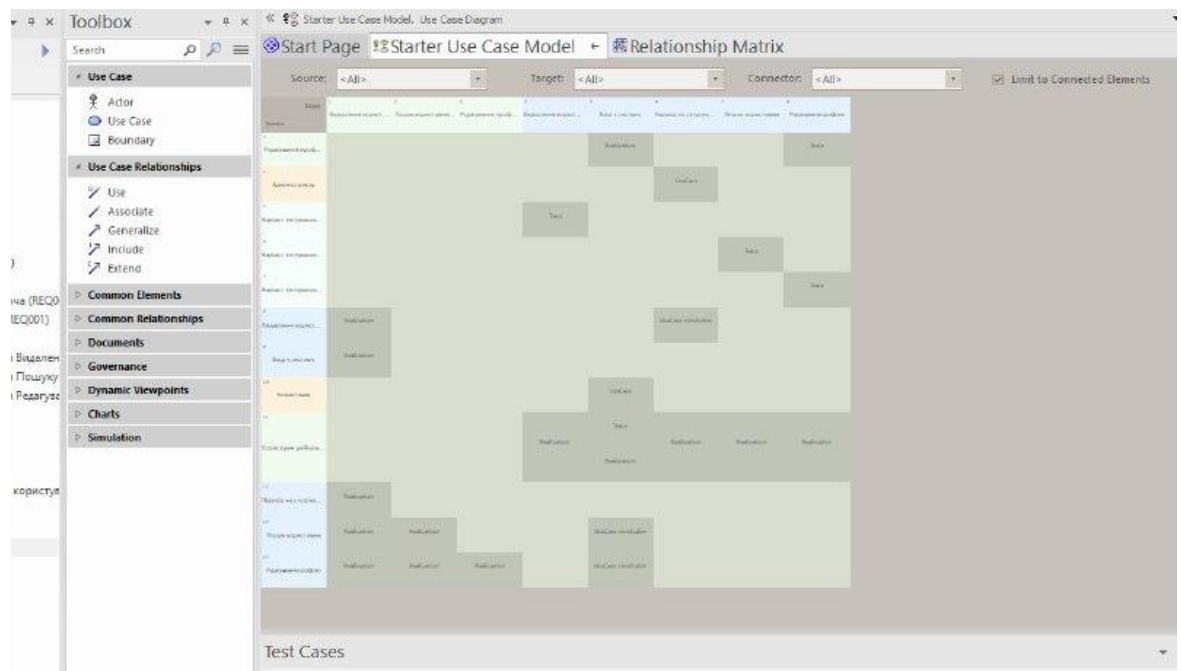


Рис. 2.31 – Діаграма трасування в системі.

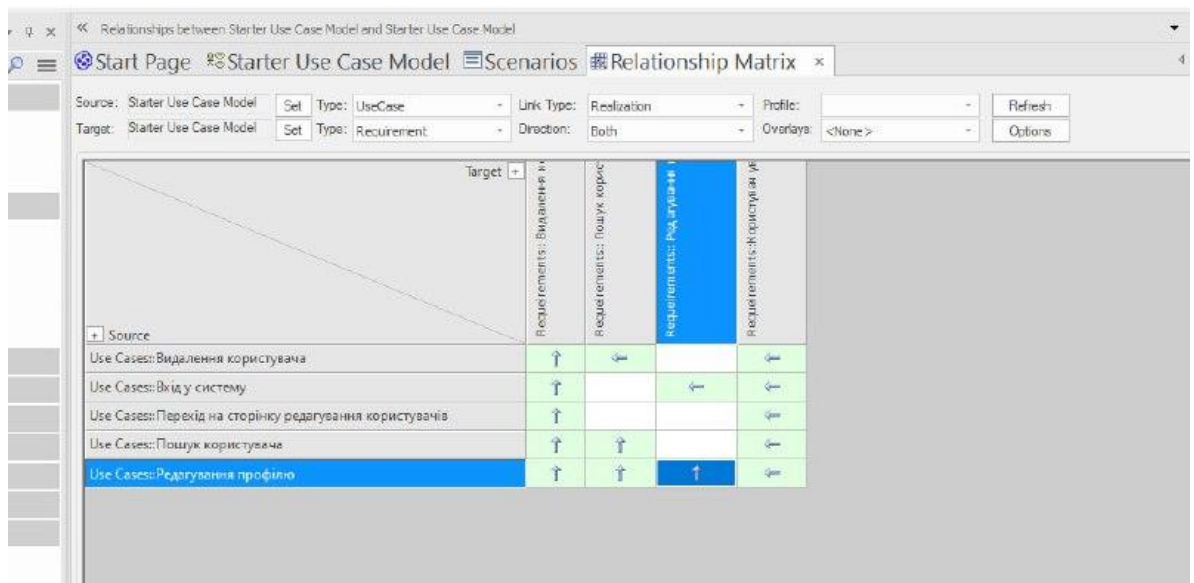


Рис. 2.32 – Матриця взаємозв'язків в системі «Realization».

РОЗДІЛ 3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

3.1 Інформаційне забезпечення

Система управління активами використовує реляційну базу даних як центральне сховище інформації. Ця база даних спеціально розроблена для ефективного зберігання, керування та доступу до даних про активи. Таким чином ви отримуєте повний контроль над своїм майном.

Використання SQLite в зв'язці з C++, технологією QT забезпечує потужний інструментарій для роботи з базами даних. Інтеграція з C++ відбувається за допомогою QT SQLApplication, що дозволяє взаємодіяти з базою даних для виконання запитів, оновлень та отримання даних у C++-додатках.

У рамках інформаційної системи бухгалтерського обліку приділено велику увагу розробці користувацького інтерфейсу з метою забезпечення ефективності та зручності користування системою. Вхідні форми, такі як форма реєстрації, форма введення фінансових операцій та форма звітності, розроблені таким чином, щоб мінімізувати необхідність введення даних вручну та забезпечити швидке та безпомилкове внесення інформації. Наприклад, форма введення фінансових операцій може включати випадальні списки для вибору категорії операцій та автоматичні перевірки на валідність введених даних.

База даних складається з шести взаємозалежних таблиць:

Нематеріальні активи: У цій таблиці зберігається інформація про програмне забезпечення, патенти, авторські права та інші нематеріальні активи. Кожен запис включає унікальний ідентифікатор, назву активу, код, дату введення в експлуатацію, вартість, термін служби та інші атрибути. Нижче наведений рис. 3.1 який зображає код для створення таблиці нематеріальні активи, позначення полів, їх визначення зображено на таблиці 3.1. Код для створення взаємозв'язків таблиць системи вказані у додатку Б, сторінка 115.

```

if (!query.exec("CREATE TABLE IF NOT EXISTS intangible_assets ( "
    "id INTEGER PRIMARY KEY AUTOINCREMENT, "
    "code TEXT, "
    "name TEXT, "
    "full_name TEXT, "
    "asset_type TEXT, "
    "account TEXT, "
    "initial_cost REAL, "
    "inventory_number TEXT, "
    "cipher TEXT, "
    "unit TEXT, "
    "entry_date DATE, "
    "activity_type TEXT)")) {
    QMessageBox::critical(this, "Error", "Failed to create table: " + query.lastError().text());
    qDebug() << "Failed to create table: " << query.lastError().text();
}

```

Рис. 3.1 – Код для створення таблиці «Нематеріальні активи».

Поле	Тип даних	Визначення
id	INT PRIMARY KEY IDENTITY	Унікальний ідентифікатор, автоматично збільшується
code	NVARCHAR(50)	Текстовий код активу
name	NVARCHAR(255)	Назва активу
full_name	NVARCHAR(255)	Повна назва або опис активу (опціонально)
asset_type	NVARCHAR(50)	Тип активу
account	NVARCHAR(50)	Код рахунку для фінансових цілей
initial_cost	MONEY	Початкова вартість активу
inventory_number	NVARCHAR(50)	Інвентарний або відстежувальний номер (опціонально)
cipher	NVARCHAR(50)	Внутрішній код або ідентифікатор (опціонально)
unit	NVARCHAR(50)	Одиниця виміру
entry_date	DATE	Дата придбання активу

activity_type	NVARCHAR(50)	Тип діяльності, пов'язаної з активом
---------------	--------------	--------------------------------------

Таблиця 3.1 Таблиця нематеріальних активів.

Основні засоби: У цій таблиці розміщується інформація про обладнання, будівлі, транспортні засоби та інші матеріальні активи. Структура аналогічна до таблиці нематеріальних активів. На рис 3.2 зображено відтворення таблиці у коді, позначення полів, їх визначення зображено на таблиці 3.2.

```
if (!query.exec("CREATE TABLE IF NOT EXISTS tangible_assets ( "
    "id INTEGER PRIMARY KEY AUTOINCREMENT, "
    "code TEXT, "
    "name TEXT, "
    "full_name TEXT, "
    "asset_type TEXT, "
    "account TEXT, "
    "initial_cost REAL, "
    "inventory_number TEXT, "
    "cipher TEXT, "
    "unit TEXT, "
    "entry_date DATE, "
    "activity_type TEXT)")) {
    QMessageBox::critical(this, "Error", "Failed to create table: " + query.lastError().text());
    qDebug() << "Failed to create table: " << query.lastError().text();
}
```

Рис. 3.2 – Код для створення таблиці «Основні засоби».

Поле	Тип даних	Визначення
id	INT PRIMARY KEY IDENTITY	Унікальний ідентифікатор, автоматично збільшується
code	NVARCHAR(50)	Текстовий код активу
name	NVARCHAR(255)	Назва активу
full_name	NVARCHAR(255)	Повна назва або опис активу (опціонально)
asset_type	NVARCHAR(50)	Тип активу
account	NVARCHAR(50)	Код рахунку для фінансових цілей
initial_cost	MONEY	Початкова вартість активу

inventory_number	NVARCHAR(50)	Інвентарний або відстежувальний номер (опціонально)
cipher	NVARCHAR(50)	Внутрішній код або ідентифікатор (опціонально)
unit	NVARCHAR(50)	Одиниця виміру
entry_date	DATE	Дата придбання активу
activity_type	NVARCHAR(50)	Тип діяльності, пов'язаної з активом

Таблиця 3.2 Таблиця основних засобів.

Інформація про амортизацію нематеріальних активів: У цій таблиці відстежуються розрахунки амортизації для кожного активу. Вона містить ідентифікатор активу, тип, який використовується метод амортизації, дати початку та закінчення періоду амортизації. Позначення полів, їх визначення зображено на таблиці 3.3.

```

if (!query.exec("CREATE TABLE IF NOT EXISTS intangible_assets_depreciation (
    "id INTEGER PRIMARY KEY AUTOINCREMENT, "
    "asset_id INTEGER, "
    "expense_type TEXT, "
    "depreciation_method TEXT, "
    "salvage_value REAL, "
    "depreciation_term DATE, "
    "FOREIGN KEY(asset_id) REFERENCES intangible_assets(id)")) {
    QMessageBox::critical(this, "Error", "Failed to create table 'intangible_assets_depreciation': " + query.lastError().text());
    // qDebug() << "Failed to create table 'intangible_assets_depreciation': " << query.lastError().text();
    ui->statusbar->showMessage("Failed to create table 'intangible_assets_depreciation': " + query.lastError().text());
}

```

Рис. 3.3– Код для створення таблиці «Амортизація нематеріальних активів»

Поле	Тип даних	Визначення
id	INT PRIMARY KEY IDENTITY	Первичный ключ с автоинкрементом для уникального идентификатора

asset_id	INT	Зовнішній ключ, що посиляється на intangible_assets.id
expense_type	NVARCHAR(50)	Тип витрат, понесених під час амортизації
depreciation_method	NVARCHAR(50)	Метод, що використовується для розрахунку амортизації
salvage_value	MONEY	(Необов'язково) Оціночна залишкова вартість наприкінці терміну корисного використання
depreciation_term	DATE	Дата закінчення періоду амортизації

Таблиця 3.3 Таблиця амортизації нематеріальних активів.

Інформація про амортизацію основних засобів: У цій таблиці відстежуються розрахунки амортизації для кожного активу. Структура аналогічна до таблиці амортизації нематеріальних активів. Код для створення таблиці зображений 3.4, визначення полів і визначення зображені на таблиці 3.4.

```

if (!query.exec("CREATE TABLE IF NOT EXISTS tangible_assets_depreciation ( "
    "id INTEGER PRIMARY KEY AUTOINCREMENT, "
    "asset_id INTEGER, "
    "expense_type TEXT, "
    "depreciation_method TEXT, "
    "salvage_value REAL, "
    "depreciation_term DATE, "
    "FOREIGN KEY(asset_id) REFERENCES tangible_assets(id)")) {
    QMessageBox::critical(this, "Error", "Failed to create table 'tangible_assets_depreciation': " + query.lastError().text());
    qDebug() << "Failed to create table 'tangible_assets_depreciation': " << query.lastError().text();
    ui->statusbar->showMessage("Failed to create table 'tangible_assets_depreciation': " + query.lastError().text());
}

```

Рис. 3.4 – Код для створення таблиці «Амортизація основних засобів»

Поле	Тип даних	Визначення
id	INT PRIMARY KEY IDENTITY	Первичний ключ с автоінкрементом для унікального ідентификатора
asset_id	INT	Зовнішній ключ, що посилається на tangible_assets.id
expense_type	NVARCHAR(50)	Тип витрат, понесених під час амортизації
depreciation_method	NVARCHAR(50)	Метод, що використовується для розрахунку амортизації
salvage_value	MONEY	(Необов'язково) Оціночна залишкова вартість наприкінці терміну корисного використання
depreciation_term	DATE	Дата закінчення періоду амортизації

Таблиця 3.4 Таблиця амортизації основних засобів.

У системі реалізовані таблиці для обліку руху активів та їх місцезнаходження, таблиці їх поля та визначення таб. 3.5-3.6.

Поле	Тип даних	Визначення
id	INT PRIMARY KEY IDENTITY	Первинний ключ з автоінкрементом для унікального ідентифікатор
location_name	NVARCHAR(255)	Назва місцезнаходження

description	NVARCHAR(255)	(Необов'язково) Опис місцезнаходження
-------------	---------------	---------------------------------------

Таблиця 3.5 Таблиця місцезнаходження активів.

Поле	Тип даних	Визначення
id	INT PRIMARY KEY IDENTITY	Первинний ключ з автоінкрементом для унікального ідентифікатора
asset_id	INT	Зовнішній ключ, що посиляється на tangible_assets.id або intangible_assets.id
asset_type	NVARCHAR(50)	Тип активу (матеріальний або нематеріальний)
from_location_id	INT	Зовнішній ключ, що посиляється на asset_locations.id (початкове місцезнаходження)
to_location_id	INT	Зовнішній ключ, що посиляється на asset_locations.id (кінцеве місцезнаходження)
movement_date	DATE	(Необов'язково) Опис переміщення

Таблиця 3.6 Таблиця руху активів.

Увесь код для створення бази даних та їх заповнення див. у додатку Б, сторінка 115.

Переваги системи управління активами

- Гнучкість і масштабованість: Система легко адаптується до потреб. Додавання нових типів активів або зміна існуючих не потребує зусиль;
- Швидкий пошук та обробка даних: Оптимізована структура бази даних забезпечує високу продуктивність, дозволяючи вам швидко знаходити необхідну інформацію;
- Аналіз та звітність: Глибокий аналіз стану активів та їх амортизації дозволяє приймати обґрунтовані управлінські рішення.

При створенні інформаційної системи використано SQLite, легка, вбудована та самодостатня система управління реляційними базами даних (СКБД), має ряд переваг перед традиційними серверними базами даних для програм управління активами:

1. Простота та зручність використання: Файлова архітектура SQLite виключає необхідність складної установки, налаштування та адміністрування, що робить її ідеальною для середовищ з обмеженими ресурсами;
2. Портативність: Однофайлова природа SQLite дозволяє легко розповсюджувати та розгортати її на різних платформах, включаючи настільні комп'ютери, сервери та мобільні пристрої;
3. Продуктивність та масштабованість: SQLite забезпечує виняткову продуктивність при виконанні загальних завдань управління активами, таких як вставка, пошук та оновлення даних, навіть при роботі з великими масивами даних. Його масштабованість забезпечує безперебійну роботу зі зростаючими портфелями активів;
4. Надійність та цілісність даних: Відповідність SQLite стандарту ACID (Atomicity, Consistency, Isolation, Durability) гарантує цілісність та несуперечність даних навіть у разі відключення електрики або збою системи;
5. Необхідність налаштування та обслуговування: SQLite вимагає мінімального налаштування та обслуговування, що знижує накладні витрати ІТ-відділу та дозволяє зосередитись на основних видах діяльності з управління активами;

6. Крос-платформна сумісність: Широка підтримка SQLite у різних операційних системах, включаючи Windows, macOS, Linux та Android, забезпечує сумісність з різними ІТ-середовищами;

7. Безпека: SQLite пропонує надійні засоби захисту, включаючи шифрування та механізми контролю доступу, для забезпечення збереження конфіденційних даних про активи;

8. Відкритий вихідний код та підтримка спільноти: Відкритий вихідний код SQLite сприяє формуванню активної спільноти розробників, надаючи велику документацію, форуми підтримки та інструменти сторонніх розробників.

Взаємодія з базами даних: QSqlDatabase і QSqlQuery забезпечують зручний і ефективний доступ до бази даних, що дозволяє легко виконувати запити і маніпулювати даними.

1. Модель обліку основних засобів та амортизації:

Бізнес-логіка: Моделі описують структуру даних і бізнес-правила, що застосовуються до активів та їхньої амортизації. Сюди входить розрахунок амортизації, облік зносу та інші важливі операції.

Гнучкість та масштабованість: Моделі дозволяють легко додавати нові типи активів і методи амортизації, забезпечуючи гнучкість і масштабованість системи.

2. Загальний принцип взаємодії:

Фронтенд взаємодіє з бекендом через асинхронні запити:

1. Qt надає зручний та інтуїтивно зрозумілий інтерфейс для користувачів, що дозволяє їм швидко отримувати необхідну інформацію та виконувати операції.

2. Асинхронна обробка запитів підвищує продуктивність та швидкість реакції системи, забезпечуючи безперебійну роботу навіть при великій кількості запитів.

3. Бекенд обробляє запити, взаємодіє з базою даних та виконує бізнес-логіку:

4. Використання C++ та Qt для логіки бекенду забезпечує високу продуктивність та надійність системи.

5. Взаємодія з базою даних через QSqlDatabase та QSqlQuery дозволяє легко виконувати складні запити та маніпулювати даними.

База даних забезпечує зберігання даних та доступ до них:

SQLite забезпечує надійне та безпечне зберігання даних, що є критично важливим для облікових систем.

Оптимізація запитів і використання індексів забезпечують швидкий доступ до даних і підвищують продуктивність системи.

Така архітектура забезпечує створення масштабованої, ефективної та надійної облікової інформаційної системи, що відповідає сучасним вимогам і стандартам. Вона забезпечує високий рівень продуктивності, безпеки та надійності, що є критично важливим для будь-якої облікової системи.

Таким чином, простота, переносимість, продуктивність, надійність, низькі вимоги до обслуговування, крос-платформна сумісність, безпека та відкритий вихідний код SQLite роблять його відмінним вибором для систем управління активами, особливо в умовах обмежених ресурсів або там, де переносимість та простота розгортання мають вирішальне значення.

Система управління активами базується на реляційній базі даних (БД), яка є центральним сховищем інформації про активи. БД складається з шести взаємопов'язаних таблиць:

1. Нематеріальні активи;
2. Основні засоби;
3. Інформація про амортизацію основних засобів;
4. Інформація про амортизацію нематеріальних активів;
5. Рух активів;
6. Місцезнаходження активів.

Ці таблиці пов'язані між собою первинними ключами і зовнішніми ключами, забезпечуючи цілісність і узгодженість даних. Даталогічна модель бази даних зображена на рис. 3.4.

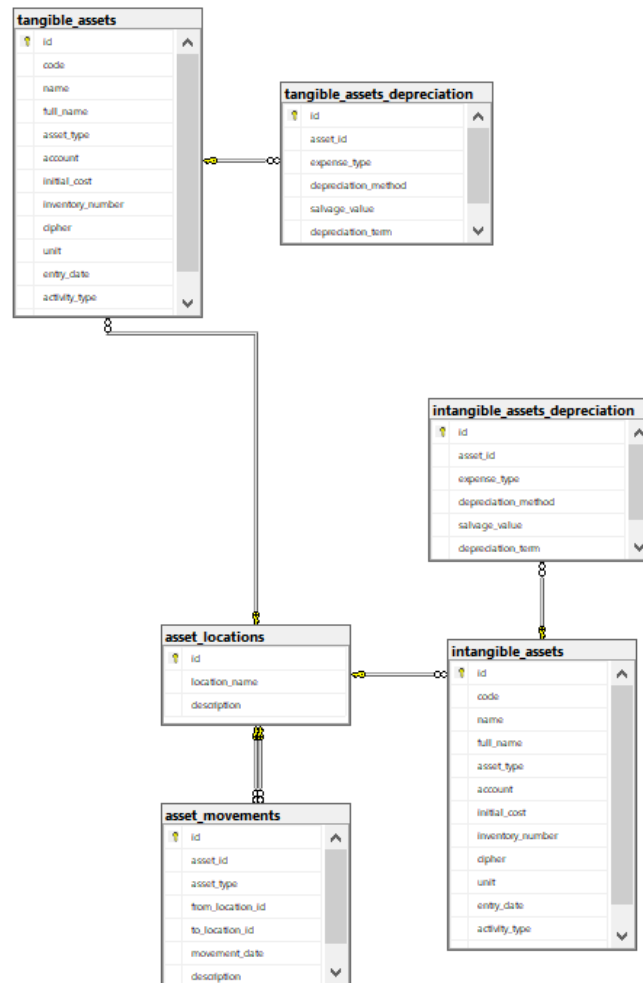


Рис. 3.4— Датологічна модель бази даних.

Код для створення таблиць бази даних інформаційної системи бухгалтерського обліку наведений у додатку Б, сторінка 115.

Інформація зберігається на сервері, який забезпечує надійне зберігання і швидкий доступ до даних. Також у системі реалізовано такі методи контролю інформації:

- Авторизація: Користувачам надаються різні рівні доступу до даних залежно від їхньої ролі в системі;
- Аудит: Усі дії користувачів у системі реєструються для відстеження та контролю.

Для забезпечення надійності та достовірності інформації у системі бухгалтерського обліку використовуються такі заходи:

1. Резервне копіювання: Регулярне створення резервних копій даних для уникнення втрати інформації у випадку непередбачуваних ситуацій або збоїв системи;
2. Швидкість та ефективність обробки: Забезпечення швидкості та ефективності обробки інформації, щоб користувачі могли отримувати доступ до неї вчасно та без затримок;
3. Доступність системи: Забезпечення неперервного доступу до інформації та функціонування системи навіть у випадку виникнення технічних або програмних проблем;
4. Відновлення: У разі збою системи можна відновити дані з резервної копії;
5. Перевірка даних: Застосування контрольних сум, хеш-функцій та інших методів для виявлення будь-яких змін у даних, які можуть бути викликані помилковими діями чи зовнішніми втручанням.

Інформаційне забезпечення забезпечує надійне, безпечне та ефективне зберігання, управління та доступ до інформації про активи. Це дає змогу організації ефективно управляти своїми активами, оптимізувати їх використання та підвищувати рентабельність.

Інформаційна система бухгалтерського обліку основних засобів та нематеріальних активів має наступну загальну схему:

1. Користувацький інтерфейс (UI): Взаємодія з користувачем здійснюється через інтуїтивно зрозумілий та ергономічний інтерфейс, розроблений з використанням технологій QT C++;
2. База даних: Для зберігання та організації даних використовується СКБД SQLite, яка забезпечує ефективний доступ до інформації;
3. Бізнес-логіка: Логіка програмного забезпечення, що відповідає за обробку даних та виконання різних операцій відповідно до вимог бізнес-процесів;
4. Модуль аналітики: Компонент, який відповідає за аналіз та виведення статистичної інформації на основі накопичених даних;

5. Серверні компоненти: Використовуються для забезпечення комунікації з іншими системами та сервісами.

Інформаційна модель зображена на рисунку 2.7 сторінка 28.

Інформаційне забезпечення відіграє важливу роль у забезпеченні надійності, ефективності та безпеки функціонування системи. Його структура, принципи організації та застосування відповідних методів контролю дозволяють забезпечити якісну роботу програмного забезпечення і задоволення потреб користувачів.

3.2 Технічне забезпечення

Інформаційна система базується на сучасних технологіях і має розподілену структуру, що забезпечує високу продуктивність, масштабованість і надійність. Нижче наведено детальний опис основних характеристик та компонентів цієї системи.

1. Фронтенд:

Qt Framework:

Кросплатформеність: Qt підтримує розробку програм, які можуть працювати на різних операційних системах, включаючи Windows, macOS та Linux.

Графічний інтерфейс користувача (GUI): Qt надає широкий спектр інструментів для створення сучасних та зручних інтерфейсів, включаючи можливості для створення складних графічних елементів, таких як таблиці, графіки та діаграми.

Сигнали та слоти: Механізм сигналів та слотів Qt дозволяє легко реалізувати асинхронну обробку подій, що підвищує продуктивність та чуйність інтерфейсу.

Модульність: Qt дозволяє легко розширювати функціональність програми шляхом додавання нових модулів та компонентів.

2. Бекенд:

C++ та Qt:

Висока продуктивність: Використання C++ забезпечує високу продуктивність та ефективну обробку даних, що є критично важливим для облікових систем.

Многопоточність: Qt надає інструменти для багатопотоковості, що дозволяє обробляти декілька запитів одночасно і забезпечує безперервну роботу системи.

Безпека: C++ дозволяє детально контролювати управління пам'яттю та ресурсами, забезпечуючи високий рівень безпеки додатків.

Для ефективної роботи інформаційної системи з обліку основних засобів та нематеріальних активів необхідно ретельно підібрати конфігурацію технічних засобів. Розглянемо рекомендаційну конфігурацію серверного обладнання та характеристик комп'ютерів користувачів системи в залежності від їх потреб.

Серверне обладнання

Сервер є центральним компонентом, що забезпечує зберігання та обробку даних, а також взаємодію між різними компонентами системи. Оскільки облікова система може мати значні навантаження, серверне обладнання повинно відповідати високим вимогам щодо продуктивності, надійності та масштабованості.

Приклад рекомендованої конфігурації сервера, із урахуванням на використання інформаційної системи на масштабному підприємстві:

- Процесор: Intel Xeon або аналогічний, мінімум 2 ядра, частота 2.2 ГГц і вище;
- Оперативна пам'ять: 8 ГБ DDR4 ECC (можливість розширення до 128 ГБ);
- Система зберігання даних:
- Основний диск: SSD, кількість пам'яті визначається необхідністю (для операційної системи та баз даних)
- Додатковий диск: HDD, кількість пам'яті визначається необхідністю (для архівування даних)

- Мережева карта: Gigabit Ethernet, 2 порти для забезпечення надійного підключення

- Операційна система: Windows Server 2019 або Linux (Ubuntu Server, CentOS)

- Резервне копіювання: Система резервного копіювання на базі NAS (Network Attached Storage) або спеціалізованого програмного забезпечення для резервування

- Джерело безперебійного живлення (UPS): Потужністю не менше 1500 ВА для забезпечення безперервної роботи сервера в разі перебоїв з електропостачанням

Ці рекомендації до серверної частини є необов'язковими, систему і компоненти серверної частини потрібно розширювати за необхідності підприємства та збільшенні навантаження на інформаційну систему.

Клієнтське обладнання

Комп'ютери користувачів системи повинні забезпечувати швидкий доступ до сервера, стабільну роботу додатків, а також підтримувати всі необхідні програмні компоненти.

Рекомендована конфігурація комп'ютерів користувачів:

- Процесор: Intel Core i3 або AMD Ryzen 3, частота 2.0 ГГц і вище
- Оперативна пам'ять: мінімум 4 ГБ DDR3 (можливість розширення до 16 ГБ)

- Система зберігання даних: HDD або SSD, мінімум 16 ГБ
- Відеокарта: Інтегрована або дискретна відеокарта з
- Мережева карта: Gigabit Ethernet, Wi-Fi (для бездротового підключення)

- Операційна система: Windows 10 ,11 або Linux (Ubuntu, Fedora)
- Переферія: Клавіатура, мишка, принтер (за необхідністю)
- Джерело безперебійного живлення (UPS): Потужністю не менше 600 ВА для забезпечення безперебійної роботи в разі перебоїв з електропостачанням

Приклад кількості обладнання на підприємстві, що дозволить у повній мірі використовувати інформаційну систему та даватиме можливість подальшого розвитку, за необхідності.

Сервери:

- 1 основний сервер для бази даних та обробки даних
- 1 резервний сервер (за необхідністю для забезпечення відмовостійкості та резервування)

Клієнтські комп'ютери:

Кількість комп'ютерів залежить від кількості користувачів системи. Наприклад, для відділу бухгалтерії, де працює 10 осіб, необхідно 10 клієнтських комп'ютерів.

Ретельно підібране технічне забезпечення є ключовим фактором для стабільної та ефективної роботи інформаційної системи з обліку основних засобів та нематеріальних активів. Сервер з високою продуктивністю, надійним зберіганням даних та резервним копіюванням забезпечує централізоване управління і обробку інформації. Водночас, клієнтські комп'ютери повинні мати достатню потужність для швидкої та стабільної роботи користувачів з системою. Забезпечення безперебійного живлення та захисту від збоїв гарантує безперервність роботи всіх компонентів системи.

3.3 Програмне забезпечення

При розробці системи була використана технологія QT. QT підтримує розробку програм, які можуть працювати на різних операційних системах, включаючи Windows, macOS, Linux, iOS та Android. Це забезпечує широке охоплення та легкість розгортання у різних середовищах. Така кроссплатформеність необхідна лише у окремих випадках.

Багатий графічний інтерфейс користувача (GUI): Qt надає широкі інструменти та бібліотеки для створення сучасних та зручних інтерфейсів. Це включає підтримку віджетів, макетів, фреймворків графічного вигляду та багато іншого. Це дозволяє розробляти складні графічні елементи, такі як таблиці, графіки, діаграми та інтерактивні форми.

Віджети Qt та Qt Quick: Qt пропонує як традиційну розробку інтерфейсу користувача на основі віджетів, так і сучасний декларативний підхід з використанням QML та Qt Quick, що дозволяє розробникам обирати найкращий підхід для своїх потреб.

- **Механізм сигналів та слотів:** Однією з найпотужніших можливостей Qt є механізм сигналів та слотів. Він забезпечує безпечний за типом спосіб зв'язку між об'єктами;

- **Сигнали:** Використовуються для сповіщення про виникнення події. Вони можуть бути видані будь-яким об'єктом;

- **Слоти:** Функції, які викликаються у відповідь на сигнал. Слоти можуть бути функціями-членами, звичайними функціями або лямбда-виразами;

- **Тип Безпека:** Механізм гарантує, що сигнали і слоти мають сумісні сигнатури, запобігаючи багатьом помилкам під час виконання;

- **Гнучкі з'єднання:** Сигнали можуть бути підключені до декількох слотів і навпаки, а з'єднання можуть бути встановлені через потоки, що забезпечує надійне програмування, кероване подіями;

- **Модульність та розширюваність:** Модульна структура Qt дозволяє розробникам використовувати лише ті компоненти, які їм потрібні, зберігаючи додатки легкими та ефективними. Крім того, архітектура плагінів Qt дозволяє легко розширювати функціональність;

- **Доступність:** Qt містить засоби для забезпечення доступності програм для користувачів з обмеженими можливостями, відповідно до таких стандартів, як WCAG та Section 508.

QSqlDatabase - це ключовий компонент Qt, який забезпечує абстракцію для роботи з базами даних. Цей клас спрощує доступ до даних, дозволяючи розробникам підключатися до різних типів баз даних, виконувати SQL-запити, обробляти результати та керувати з'єднаннями.

QSqlDatabase та QSqlQuery забезпечують зручний та ефективний доступ до бази даних, дозволяючи легко виконувати запити та маніпулювати даними. Qt підтримує різні системи керування базами даних, зокрема MySQL, PostgreSQL, SQLite та інші.

Підтримка різноманітних баз даних: QSqlDatabase може підключатися до широкого спектру баз даних, включаючи MySQL, PostgreSQL, SQLite, Oracle та Microsoft SQL Server. Це дає розробникам гнучкість у виборі відповідної бази даних для їх проєкту.

- Абстракція SQL-запитів: QSqlDatabase використовує зручний інтерфейс для виконання SQL-запитів, приховуючи деталі конкретної бази даних. Це робить код більш читабельним та портативним;

- Обробка результатів запитів: QSqlDatabase пропонує зручні методи для обробки результатів SQL-запитів, таких як витяг даних з таблиць, перехід до наступного запису та навігація по результатах;

- Управління з'єднаннями: QSqlDatabase автоматично керує з'єднаннями з базою даних, гарантуючи їх правильне відкриття, закриття та повторне використання.

QTabWidget - це елемент GUI Qt, який використовується для створення інтерфейсів з вкладками. Цей клас дозволяє організувати контент програми на логічні групи, розміщуючи кожен групу на окремій вкладці. Це робить інтерфейс більш зручним для користувача, дозволяючи легко перемикаватися між різними розділами або модулями програми.

Переваги використання QTabWidget:

- Зручна організація контенту: QTabWidget дозволяє розбивати складний контент на логічні групи, роблячи його більш доступним та зрозумілим для користувачів

- Економія простору: Qt

QPrinter - це клас Qt, який використовується для друку документів та зображень. Цей клас надає потужний та гнучкий інтерфейс для налаштування параметрів друку, форматування та друку даних з програми.

Основні можливості QPrinter:

- Вибір принтера: QPrinter дозволяє користувачам вибрати принтер, на якому вони хочуть друкувати, з списку доступних принтерів.

- **Налаштування параметрів друку:** Користувачі можуть налаштувати різні параметри друку, такі як орієнтація сторінки, розмір паперу, роздільна здатність друку та маржі.

- **Формування друкованих сторінок:** QPrinter дозволяє розробникам програмного забезпечення формувати друковані сторінки, додаючи заголовки, колонтитули, рисунки та інші елементи.

Друк даних з програми: QPrinter може друкувати текстовий контент, зображення, таблиці та інші дані з програми.

QPdfWriter - це клас Qt, який використовується для створення PDF-файлів з динамічним контентом. Цей клас надає потужний та гнучкий інтерфейс для генерації PDF-документів, які можуть включати текст, зображення, таблиці, графіки та інші елементи.

Переваги використання QPdfWriter:

- **Створення динамічних PDF-документів:** QPdfWriter дозволяє розробникам програмного забезпечення створювати PDF-документи з даними, які генеруються динамічно, наприклад, звіти, квитанції, рахунки-фактури та інші;

- **Контроль форматування PDF:** QPdfWriter дозволяє розробникам контролювати форматування PDF-документів, включаючи шрифти, поля, стилі та інші атрибути;

- **Вставка зображень та графіків:** QPdfWriter дозволяє вставляти зображення та графіки в PDF-документи, роблячи їх більш інформативними та візуально привабливими;

- **Додавання посилань та закладок:** QPdfWriter дозволяє додавати посилання та закладки до PDF-документів, полегшуючи навігацію по документу;

Окрім вищезгаданих технологій, програмне забезпечення для управління активами може використовувати й інші технології, такі як:

- **QStandardItemModel:** Цей клас Qt використовується для створення та управління моделями даних, які можуть використовуватися для відображення інформації про активи в таблицях та списках;

- **QChart:** Цей клас Qt використовується для створення графіків та діаграм, які можуть візуалізувати дані про активи, такі як розподіл за категоріями, амортизація та динаміка вартості;

- **QNetworkAccessManager:** Цей клас Qt використовується для здійснення HTTP-запитів, що може бути корисно для отримання даних про активи з зовнішніх джерел;

- **Qt XML:** Цей модуль Qt використовується для роботи з XML-даними, що може бути корисно для імпорту та експорту даних про активи.

Використання цих технологій Qt дозволило створити ефективне та зручне програмне забезпечення для управління активами, яке може значно допомогти організаціям у оптимізації управління своїми активами. Завдяки своїй кроссплатформеній природі, багатому функціоналу та простоті використання, Qt є ідеальним фреймворком для розробки програмного забезпечення для управління активами, яке відповідає потребам сучасних бізнесів.

Як вже вище було зазначено, програмне забезпечення для обліку активів та нарахування амортизації повинно бути добре структуроване та організоване за модульним принципом для забезпечення гнучкості, розширюваності та підтримки майбутніх змін. Нижче наведено основні модулі інформаційної системи та їх функціональне призначення:

1. Модуль управління активами

Цей модуль відповідає за зберігання та обробку інформації про основні та нематеріальні активи компанії.

Основні функції:

- Додавання нових активів до бази даних, рис 3.5.

```

void NoneMaterialActives::buttonBox_accepted_for_InTangibleAsset()
{
    QString code = ui->lineEdit_kod->text();
    QString name = ui->lineEdit_nazva->text();
    QString fullName = ui->lineEdit_povna_nazva->text();
    QString assetType = ui->comboBox_vid_aktiva->currentText();
    QString account = ui->comboBox_rakhunok_obliku->currentText().split(" - ").first();
    double initialCost = ui->doubleSpinBox_pochatkova_vartist->value();
    QString inventoryNumber = ui->lineEdit_inventarnyy_nomer->text();
    QString cipher = ui->lineEdit_shifr->text();
    QString unit = ui->comboBox_odnitsya_vymiru->currentText();
    QDate entryDate = ui->dateEdit_data_vvodu->date();
    QString activityType = ui->comboBox_vid_diyalnosti->currentText();

    if (code.isEmpty() || name.isEmpty() || fullName.isEmpty() || inventoryNumber.isEmpty() || cipher.isEmpty()) {
        QMessageBox::warning(this, "Input Error", "Please fill in all required fields.");
        return;
    }

    QSqlQuery query(db);
    query.prepare("INSERT INTO intangible_assets (code, name, full_name, asset_type, account, initial_cost, inventory_number, cipher, unit, entry_date, activity_type) "
        "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
    query.addBindValue(code);
    query.addBindValue(name);
    query.addBindValue(fullName);
    query.addBindValue(assetType);
    query.addBindValue(account);
    query.addBindValue(initialCost);
    query.addBindValue(inventoryNumber);
    query.addBindValue(cipher);
    query.addBindValue(unit);
    query.addBindValue(entryDate);
    query.addBindValue(activityType);

    if (!query.exec()) {
        QMessageBox::critical(this, "Database Error", "Failed to insert data into database.");
    }
    else {
        QMessageBox::information(this, "Success", "Data has been successfully added to the database.");
        this->accept(); // Закрити вікно при успішному додаванні даних
    }
}

void NoneMaterialActives::buttonBox_accepted_for_TangibleAsset()
{

```

Рис. 3.5 Фрагмент коду додання нових активів до інформаційної системи, повний код наведений у додатку В сторінка 117.

- Редагування існуючих записів про активи;
- Видалення активів, які вже відписані або продані;
- Перегляд активів ;
- Генерація звітів про активи (наприклад, звіт про залишок активів на певну дату) рис 3.6.

```

232 void generateReport::generateAssetsReport()
233 {
234     QString fileName = QFileDialog::getSaveFileName(this, tr("Зберегти звіт"), "", tr("PDF Files (*.pdf)"));
235     if (fileName.isEmpty()) {
236         return; // User canceled saving
237     }
238
239     // Prepare the SQL query
240     QSqlQuery query(db);
241     QString selectedCode = ui->comboBox_active_code->currentText();
242     QString selectedCategory = ui->comboBox_active_category->currentText();
243     QString startDate = ui->dateEdit_from->date().toString("yyyy-MM-dd");
244     QString endDate = ui->dateEdit_to->date().toString("yyyy-MM-dd");
245
246     QString queryStr = "SELECT * FROM intangible_assets WHERE code = ? AND account = ? AND entry_date BETWEEN ? AND ?";
247     query.prepare(queryStr);
248     query.addBindValue(selectedCode);
249     query.addBindValue(selectedCategory);
250     query.addBindValue(startDate);
251     query.addBindValue(endDate);
252
253     // Check for query execution error
254     if (!query.exec()) {
255         QMessageBox::critical(this, "Помилка", "Не вдалося виконати запит: " + query.lastError().text());
256         return;
257     }
258
259     // Create PDF writer
260     QPdfWriter writer(fileName);
261
262     // Set page size based on Qt version (adjust as needed)
263     writer.setPageSize(QPageSize(QPageSize::A4)); // Use setPageSize for Qt 5.6+
264
265     // Create painter object for drawing on the PDF
266     QPainter painter(&writer);
267     painter.setPen(Qt::black);
268     painter.setFont(QFont("Arial", 12));
269
270     // Define margins for better readability
271     const int marginX = 250;
272     const int marginY = 250;
273     int y = marginY; // Starting position with margin
274
275     // Draw report header with dynamic centering
276     int textWidth = painter.fontMetrics().horizontalAdvance("Звіт про активи\n");
277     painter.drawText((writer.width() - textWidth) / 2, y, "Звіт про активи\n");
278     y += 520;
279
280     // Draw table header (optional)
281     QStringList headers = {"Код", "Назва", "Тип актива", "Рахунок", "Початкова вартість", "Інвентарний номер", "Шифр", "Одиниця виміру", "Дата вводу", "Тип активності"};
282     QVector<int> columnWidths = {100, 180, 200, 200, 200, 150, 300, 300, 300, 300};
283     int tableX = marginX;
284     int tableY = y;
285     for (int i = 0; i < headers.size(); ++i) {
286         painter.drawText(tableX, tableY, headers[i]);
287         tableX += columnWidths[i];
288     }
289     tableY += 520;

```

Рис. 3.6 Фрагмент коду генерації звітів про активи.

2. Модуль розрахунку амортизації

Цей модуль відповідає за автоматичний розрахунок амортизації для кожного активу згідно з встановленими правилами та методами нарахування.

Основні функції:

- Вибір методу нарахування амортизації (прямолінійний, виробничий, зменшеного залишку тощо);
- Розрахунок сум амортизації для кожного активу на певний період часу;
- Внесення результатів розрахунків до бази даних.

Нижче наведена реалізація підрахунку амортизації від обраного методу розрахунку, методів розрахунку амортизації розроблено три найвідоміші, які цільно описують можливості інформаційної системи в рамках дипломного проєкту.

Усі формули для розрахунку амортизації вказана на сторінці 42.

- Прямолінійний метод;

Реалізація на C++ зображена на рис 3.6.

```

QSqlQuery straightLineQuery(db);
QString queryStr;
if (category == "нематеріальні активи" || category == "інші нематеріальні активи") {
    queryStr = "SELECT initial_cost, salvage_value, useful_life FROM intangible_assets WHERE id = ?";
} else if (category == "основні засоби") {
    queryStr = "SELECT initial_cost, salvage_value, useful_life FROM tangible_assets WHERE id = ?";
} else {
    QMessageBox::critical(this, "Error", "Invalid asset category selected.");
    return 0.0;
}

straightLineQuery.prepare(queryStr);
straightLineQuery.addBindValue(assetId);
if (!straightLineQuery.exec()) {
    QMessageBox::critical(this, "Error", "Failed to execute query: " + straightLineQuery.lastError().text());
    return 0.0;
}

if (straightLineQuery.next()) {
    double initialCost = straightLineQuery.value("initial_cost").toDouble();
    double salvageValue = straightLineQuery.value("salvage_value").toDouble();
    int usefulLife = straightLineQuery.value("useful_life").toInt();

    depreciationAmount = (initialCost - salvageValue) / usefulLife;
}

```

Рис. 3.6 Реалізація прямолінійного методу у коді інформаційної системи

- Метод зменшеного залишку.

Реалізація на C++ зображена на рис 3.7.

Повний код реалізації наведений у додатку В, сторінка 117.

```

 QSqlQuery diminishingBalanceQuery(db);
 QString queryStr;

 if (category == "нематеріальні активи" || category == "інші нематеріальні активи") {
     queryStr = "SELECT initial_cost, salvage_value, useful_life FROM intangible_assets WHERE id = ?";
 } else if (category == "основні засоби") {
     queryStr = "SELECT initial_cost, salvage_value, useful_life FROM tangible_assets WHERE id = ?";
 } else {
     QMessageBox::critical(this, "Error", "Invalid asset category selected.");
     return 0.0;
 }

 diminishingBalanceQuery.prepare(queryStr);
 diminishingBalanceQuery.addBindValue(assetId);
 if (!diminishingBalanceQuery.exec()) {
     QMessageBox::critical(this, "Error", "Failed to execute query: " + diminishingBalanceQuery.lastError().text());
     return 0.0;
 }

 if (diminishingBalanceQuery.next()) {
     double initialCost = diminishingBalanceQuery.value("initial_cost").toDouble();
     double salvageValue = diminishingBalanceQuery.value("salvage_value").toDouble();
     int usefulLife = diminishingBalanceQuery.value("useful_life").toInt();

     // Рассчитываем амортизацию на основе метода уменьшенного залишка
     double remainingValue = initialCost;
     for (int i = 0; i < usefulLife; ++i) {
         // Уменьшаем остаточную стоимость актива каждый год
         remainingValue -= (initialCost - salvageValue) / usefulLife;
     }
     // Амортизация - это разница между начальной стоимостью и оставшейся стоимостью
     depreciationAmount = initialCost - remainingValue;
 } else {
     QMessageBox::critical(this, "Error", "Invalid depreciation method selected.");
     return 0.0;
 }

```

Рис. 3.7 Реалізація методу зменшеного залишку у коді інформаційної системи.

- Виробничий метод.

Реалізація на C++ зображена на рис 3.8

```

 QSqlQuery unitsOfProductionQuery(db);
 QString queryStr;

 if (category == "нематеріальні активи" || category == "інші нематеріальні активи") {
     queryStr = "SELECT initial_cost, salvage_value, useful_life FROM intangible_assets WHERE id = ?";
 } else if (category == "основні засоби") {
     queryStr = "SELECT initial_cost, salvage_value, useful_life FROM tangible_assets WHERE id = ?";
 } else {
     QMessageBox::critical(this, "Error", "Invalid asset category selected.");
     return 0.0;
 }

 unitsOfProductionQuery.prepare(queryStr);
 unitsOfProductionQuery.addBindValue(assetId);
 if (!unitsOfProductionQuery.exec()) {
     QMessageBox::critical(this, "Error", "Failed to execute query: " + unitsOfProductionQuery.lastError().text());
     return 0.0;
 }

 if (unitsOfProductionQuery.next()) {
     double initialCost = unitsOfProductionQuery.value("initial_cost").toDouble();
     double salvageValue = unitsOfProductionQuery.value("salvage_value").toDouble();
     int usefulLife = unitsOfProductionQuery.value("useful_life").toInt();

     // Предположим, что амортизация рассчитывается на основе времени использования актива
     // и что стоимость актива уменьшается пропорционально с увеличением времени
     if (usefulLife > 0) {
         depreciationAmount = (initialCost - salvageValue) / usefulLife;
     } else {
         depreciationAmount = 0.0;
     }
 } else if (depreciationMethod == "Зменшений залишок") {

```

Рис. 3.8 Реалізація виробничого методу у коді інформаційної системи.

3. Модуль звітності

Цей модуль відповідає за генерацію різноманітних звітів та аналітичних даних щодо обліку активів та амортизації.

Основні функції:

- Генерація звітів про рух активів та інші за певний період (наприклад, за місяць, квартал, рік);
- Аналіз динаміки амортизаційних витрат;
- Визначення ефективності використання активів за певний період

Нижче наведений фрагмент коду генерації звітів рис. 3.9, повну реалізацію див. додаток В, сторінка 117.

```

25 ~ if (reportType == "assets_report")
26 ~ {
27 ~ ui->lineEdit_chosed_report->setText("Звіт з активів");
28 ~ setFieldsForAssetsReport();
29 ~ connect(ui->buttonBox, &QDialogButtonBox::accepted, this, &generateReport::generateAssetsReport);
30 ~ }
31 ~
32 ~ else if (reportType == "depreciation_report")
33 ~ {
34 ~ ui->lineEdit_chosed_report->setText("Звіт з амортизації");
35 ~ setFieldsForDepreciationReport();
36 ~ connect(ui->buttonBox, &QDialogButtonBox::accepted, this, &generateReport::generateDepreciationReport);
37 ~ }
38 ~
39 ~ else if (reportType == "move_report")
40 ~ {
41 ~ ui->lineEdit_chosed_report->setText("Звіт про рух активів");
42 ~ setFieldsForMoveReport();
43 ~ connect(ui->buttonBox, &QDialogButtonBox::accepted, this, &generateReport::generateMoveReport);
44 ~ }
45 ~
46 ~ else if (reportType == "inventory_report")
47 ~ {
48 ~ ui->lineEdit_chosed_report->setText("Звіт про інвентаризацію");
49 ~ setFieldsForInventoryReport();
50 ~ connect(ui->buttonBox, &QDialogButtonBox::accepted, this, &generateReport::generateInventoryReport);
51 ~ }
52 ~
53 ~ else if (reportType == "category_report")
54 ~ {
55 ~ ui->lineEdit_chosed_report->setText("Звіт за категорією активів");
56 ~ setFieldsForCategoryReport();
57 ~ connect(ui->buttonBox, &QDialogButtonBox::accepted, this, &generateReport::generateCategoryReport);
58 ~ }
59 ~
60 ~ ui->comboBox_active_type->addItem("Основні засоби", "Інші необоротні активи", "Нематеріальні активи", "Інші нематеріальні активи");
61 ~ ui->comboBox_operation_type->addItem("придбання", "визбуття", "випиш амортизації", "амортизація", "ремонт/обслуговування", "перекласифікація", "оренда", "повернення в оренду", "випиш відшкодування", "випиш відшкодування", "випиш", "подарунок");
62 ~ ui->comboBox_depreciation_method->addItem("Прямолінійний метод", "Зменшений вартості", "Виробничий метод");
63 ~
64 ~ connect(ui->comboBox_active_type, &QComboBox::currentTextChanged, this, &generateReport::activeTypeChanged);
65 ~ connect(ui->comboBox_active_code, &QComboBox::currentTextChanged, this, &generateReport::activeCodeChanged);
66 ~
67 ~ generateReport::generateReport();
68 ~ delete ui;
69 ~
70 ~
71 ~ void generateReport::activeTypeChanged(const QString &type)
72 ~ {
73 ~ populateActiveCodes(type);
74 ~ }
75 ~
76 ~ void generateReport::activeCodeChanged(const QString &code)
77 ~ {
78 ~ QString type = ui->comboBox_active_type->currentText();
79 ~ populateActiveCategory(code, type);
80 ~ populateActiveCode(code, type);
81 ~ }
82 ~
83 ~ void generateReport::populateActiveCodes(const QString &type)
84 ~ {
85 ~ ui->comboBox_active_code->clear();
86 ~
87 ~ QString queryStr;
88 ~ if (type == "Нематеріальні активи" || type == "Інші нематеріальні активи")
89 ~ {
90 ~ queryStr = "SELECT code FROM intangible_assets";
91 ~ }
92 ~ else if (type == "Основні засоби")
93 ~ {
94 ~ queryStr = "SELECT code FROM tangible_assets";
95 ~ }
96 ~ else
97 ~ {
98 ~ return; // Для інших типів активів не обновляємо список кодів
99 ~ }
100 ~
101 ~ QSqlQuery query(db);
102 ~ if (!query.exec(queryStr))
103 ~ {
104 ~ QMessageBox::critical(this, "Database Error", "Failed to retrieve asset codes: " + query.lastError().text());
105 ~ return;
106 ~ }
107 ~

```

Рис. 3.9 Фрагмент коду генерації звітів інформаційної системи.

4. Модуль інтерфейсу користувача

Цей модуль забезпечує взаємодію користувача з програмою через графічний інтерфейс.

Основні функції:

- Відображення списку активів та їх стану;
- Можливість внесення змін до даних про активи та налаштувань амортизації;
- Генерація та відображення звітів та аналітичної інформації.

Нижче приведений фрагмент код головного меню користувацького інтерфейсу рис. 3.10.

```

220 <addaction name="action_wiki"/>
221 </widget>
222 <widget class="QMenu" name="menu_2">
223 <property name="title">
224 <string>Про нас</string>
225 </property>
226 <addaction name="action_aboutQt"/>
227 <addaction name="separator"/>
228 <addaction name="action_aboutAuth"/>
229 </widget>
230 <widget class="QMenu" name="menu_3">
231 <property name="title">
232 <string>Редагування</string>
233 </property>
234 <addaction name="action_copy"/>
235 <addaction name="action_cut"/>
236 <addaction name="action_paste"/>
237 <addaction name="action_undo"/>
238 <addaction name="action_redo"/>
239 <addaction name="separator"/>
240 </widget>
241 <widget class="QMenu" name="menu_4">
242 <property name="title">
243 <string>Звіт</string>
244 </property>
245 <addaction name="action_generateAssetsReport"/>
246 <addaction name="action_generateDepreciationReport"/>
247 <addaction name="action_generateMoveReport"/>
248 <addaction name="action_generateInventoryReport"/>
249 <addaction name="action_generateCategoryReport"/>
250 </widget>
251 <addaction name="menuFile"/>
252 <addaction name="menu_3"/>
253 <addaction name="menu"/>
254 <addaction name="menu_4"/>
255 <addaction name="menu_2"/>
256 </widget>
257 <widget class="QStatusBar" name="statusbar"/>
258 <widget class="QToolBar" name="toolBar">
259 <property name="windowTitle">
260 <string>toolBar</string>
261 </property>
262 <attribute name="toolBarArea">
263 <enum>TopToolBarArea</enum>
264 </attribute>
265 <attribute name="toolBarBreak">
266 <bool>false</bool>
267 </attribute>
268 <addaction name="action_copy"/>
269 <addaction name="separator"/>
270 <addaction name="action_paste"/>
271 <addaction name="separator"/>
272 <addaction name="action_cut"/>
273 <addaction name="separator"/>
274 <addaction name="action_undo"/>
275 <addaction name="separator"/>
276 <addaction name="action_redo"/>
277 </widget>

```

Рис. 3.10 Фрагмент коду користувацького інтерфейсу інформаційної системи.

Увесь код реалізації повного функціоналу, що відповідає рамкам бакалаврського проєкту наведений у додатку В, сторінка 117.

Модульна структура програмного забезпечення для обліку активів та нарахування амортизації забезпечує його ефективне функціонування, спрощує процес розробки та підтримки, а також робить систему більш гнучкою та розширюваною. Кожен модуль виконує певний набір функцій, що дозволяє розділити завдання та забезпечити їх ефективну роботу в межах інформаційної системи.

3.4 Результати реалізації інформаційної системи

В рамках виконання кваліфікаційного проєкту було розроблено та впроваджено інформаційну систему для управління активами саме нематеріальними та основними засобами. Система має потенціал на розширення свого функціоналу, так як тема важко-осяжна в рамках бакалаврського проєкту. Ця система ґрунтується на наступних оригінальних пропозиціях та наукових розробках:

1. Розробка моделі даних для управління активами в рамках бакалаврського проєкту: Була розроблена модель даних, яка описує всі сутності, атрибути та зв'язки, пов'язані з активами організації. Ця модель даних забезпечує єдине джерело інформації про активи та полегшує доступ до даних та їх аналіз;

2. Застосування алгоритмів розрахунку амортизації: Були реалізовані алгоритми розрахунку амортизації для різних типів активів, таких як основні засоби, нематеріальні активи. Ці алгоритми дозволяють точно розраховувати амортизацію активів протягом їх терміну служби. Алгоритми повноцінно описані у додатку В, сторінка 117;

3. Розробка модуля звітності: Був розроблений модуль звітності, який дозволяє користувачам генерувати різні звіти про активи, такі як інвентарний звіт, звіт про рух активів та звіт про амортизацію. Ці звіти надають користувачам цінну інформацію про стан активів організації. Усі звіти див. додаток Г, сторінка 112.

Інформаційна система була пілотно впроваджена в одному з підрозділів організації. Пілотне впровадження показало, що система:

1. Знижує витрати на управління активами: Завдяки автоматизації багатьох завдань з управління активами, система дозволяє організації економити час та ресурси;

2. Покращує точність даних про активи: Завдяки централізованому сховищу даних про активи, система забезпечує більшу точність та надійність даних;

3. Підвищує прозорість управління активами: Завдяки модулю звітності, система забезпечує користувачів цінною інформацією про стан активів організації, що робить управління активами більш прозорим та підзвітним.

Розроблена та впроваджена інформаційна система для управління активами є ефективним інструментом(зображення інформаційної системи див). у додатках Г, сторінка 288, який може допомогти організаціям оптимізувати управління своїми активами. Система ґрунтується на оригінальних пропозиціях х, які забезпечують її гнучкість, масштабованість та надійність. Пілотне впровадження системи показало, що вона може значно знизити витрати, покращити точність даних та підвищити прозорість управління активами.

ВИСНОВКИ

Під час проєктування кваліфікаційного дипломного проєкту було розглянуто та реалізовано інформаційну систему, яка призначена для обліку основних засобів та нематеріальних активів із розрахунком їх амортизації та можливістю вести їх повний облік та формувати звіти, інформаційна система має потенціал на розвиток та виконує поставлені перед нею вимоги, в рамках бакалаврського проєкту. Проєкт базується на сучасних технологіях, таких як QT C++ та СКБД SQLite, що дозволяє забезпечити високу продуктивність, надійність та зручність використання системи.

Під час розробки інформаційної системи було досягнуто наступних результатів:

1. Проаналізовано предметну область: Визначено основні вимоги до системи, зокрема щодо обліку основних засобів та нематеріальних активів, їх амортизації та формування звітів;
2. Розроблено архітектуру системи: Створено деталізовану архітектуру, що включає користувацький інтерфейс, бізнес-логіку та базу даних. Забезпечено інтеграцію між цими компонентами для ефективної роботи системи;
3. Реалізовано користувацький інтерфейс: Створено інтуїтивно зрозумілий інтерфейс для введення, редагування та перегляду даних, що полегшує роботу користувачів із системою;
4. Розроблено базу даних: Впроваджено СКБД SQLite для зберігання та обробки даних, що забезпечує швидкий доступ та високу надійність зберігання інформації;
5. Забезпечено надійність та безпеку даних: Реалізовано механізми резервного копіювання, а також контролю цілісності даних;
6. Впроваджено методи контролю інформації: Введено механізми валідації та верифікації даних, моніторинг системи для своєчасного виявлення та виправлення помилок;
7. Проведено тестування: Здійснено всебічне тестування системи для забезпечення її стабільної роботи та відповідності поставленим вимогам.

За результатами виконання дипломної роботи можна зробити наступні висновки:

- Розробка та впровадження ефективної інформаційної системи для управління активами є актуальною задачею для сучасних організацій. Така система дозволяє оптимізувати управління активами, знижувати витрати, покращувати точність даних та підвищувати прозорість бізнес-процесів.

- Було розроблено інформаційну систему для управління активами організації, що відповідає цілям та завданням, поставленим перед нею на початку дослідження, в рамках дипломного проектування, з урахуванням глибини тематики інформаційної системи. Система ґрунтується на оригінальних пропозиціях та наукових розробках, які забезпечують її ефективність.

- Розроблена інформаційна система може бути впроваджена в різних організаціях, незалежно від їх розміру та галузі діяльності. Система дозволить покращити управління активами та підвищити ефективність бізнес-процесів.

- Система, побудована на базі Qt, C++ та SQLite, забезпечує високу продуктивність, надійність та безпеку, необхідні для ефективного управління бухгалтерським обліком. Її архітектура дозволяє легко адаптуватися до змін та вимог бізнесу, забезпечуючи при цьому високу якість та стабільність роботи.

- Результати дослідження можуть бути використані для подальших наукових розробок в галузі управління активами. Зокрема, перспективними напрямками досліджень є інтеграція інформаційної системи з іншими бухгалтерського обліку для збільшення потенціалу та функціоналу системи, а також розробка методів аналізу даних для прогнозування терміну служби активів та оптимізації їх використання.

Такий підхід до розробки інформаційної системи дозволяє створити потужний інструмент для управління активами та амортизацією, який сприяє підвищенню ефективності бізнес-процесів та прийняттю обґрунтованих рішень на основі даних.

Загалом, виконана дипломна робота підтверджує доцільність та ефективність застосування сучасних технологій для вирішення завдань обліку

та управління основними засобами, а також підкреслює важливість комплексного підходу до розробки інформаційних систем.

Таким чином, розроблена інформаційна система відповідає сучасним стандартам та вимогам до обліку основних засобів та нематеріальних активів. Вона забезпечує високу ефективність, надійність та зручність у використанні, що дозволяє значно спростити процеси обліку та управління активами в організації. Отримані результати можуть бути використані як база для подальшого розвитку та вдосконалення інформаційної системи, розширення її функціональності та інтеграції з іншими системами організації.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Цивільний кодекс України від 23 січня 2003 року № 450-IV.
2. Податковий кодекс України від 02 грудня 2003 року № 2121-IV.
3. Національні стандарти бухгалтерського обліку (НСБО), затверджені наказом Міністерства фінансів України.
4. Бабінський В.В., Горохольський В.О., Ковальчук В.В. Інформаційні системи управління активами. Київ: МАУП, 2018.
5. Лузан О.М., Олійник А.А., Пашкевич О.Б. Інформаційні системи та технології в управлінні активами. Київ: МАУП, 2020.
6. Симоненко В.В., Ставнійчук О.В., Тарасенко О.В. Інформаційні системи та технології в економіці. Київ: КНЕУ, 2017.
7. Веб-сайт Міністерства фінансів України: <https://minfin.gov.ua/>
8. Веб-сайт Національної академії наук України: <https://nas.gov.ua/>
9. QT Documentation. Офіційна документація по бібліотеці QT, включаючи розділи по роботі з графічним інтерфейсом та базами даних. Доступно за адресою: <https://doc.qt.io/>
10. C++ Programming Language. Bjarne Stroustrup. The definitive guide to the C++ programming language, covering language features, standard libraries, and best practices.
11. SQLite Documentation. Офіційна документація по СКБД SQLite, включаючи опис синтаксису SQL, функціональних можливостей та особливостей використання. Доступно за адресою: <https://www.sqlite.org/docs.html>
12. Unified Modeling Language (UML). Офіційний стандарт UML, що описує методи моделювання інформаційних систем. Доступно за адресою: <https://www.omg.org/spec/UML/2.5.1/PDF>
13. РД 50-34.698-90 "Вимоги до змісту документів з математичного забезпечення". Стандарт, що визначає структуру та зміст документів з математичного забезпечення інформаційних систем.
14. РД 50-34.698-90 "Вимоги до змісту документів з інформаційного забезпечення". Стандарт, що визначає вимоги до змісту документів з

інформаційного забезпечення, включаючи класифікацію та кодування даних, проєктування форм первинних документів та структуру інформаційних масивів.

15. ISO/IEC 19501:2005 (OMG-UML VER 1.3) "Information technology – Open Distributed Processing – Unified Modeling Language (UML)". Міжнародний стандарт, що визначає специфікацію мови UML для моделювання інформаційних систем.

16. Stack Overflow. Спільнота розробників програмного забезпечення, де можна знайти рішення різноманітних проблем, пов'язаних з розробкою на C++ та використанням бібліотеки QT. Доступно за адресою: <https://stackoverflow.com/>

ДОДАТКИ

Графічні матеріали

Додаток Г

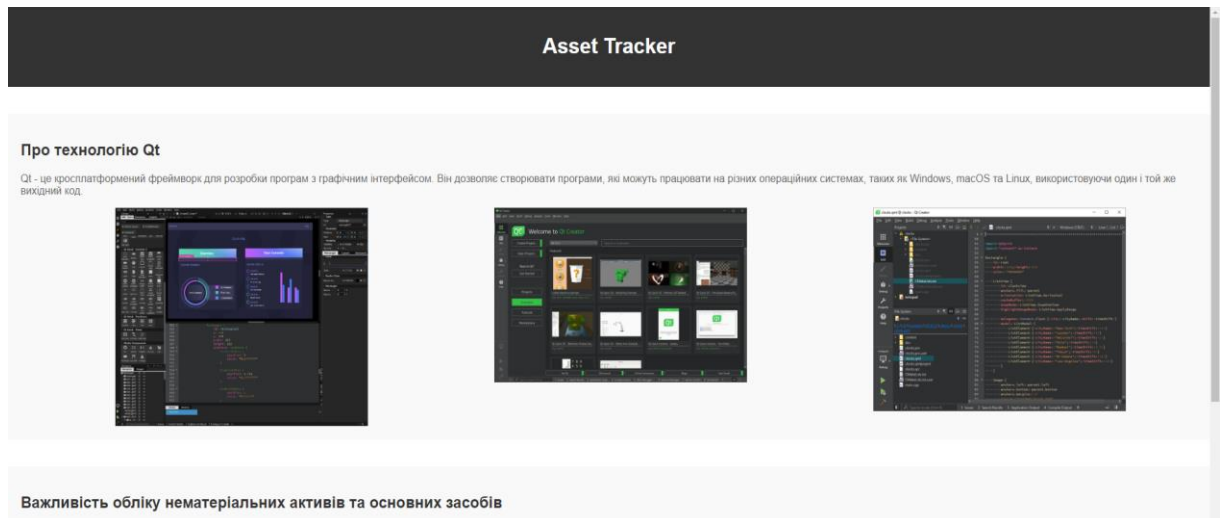


Рис. 4.1 Сайт для інсталяції програми

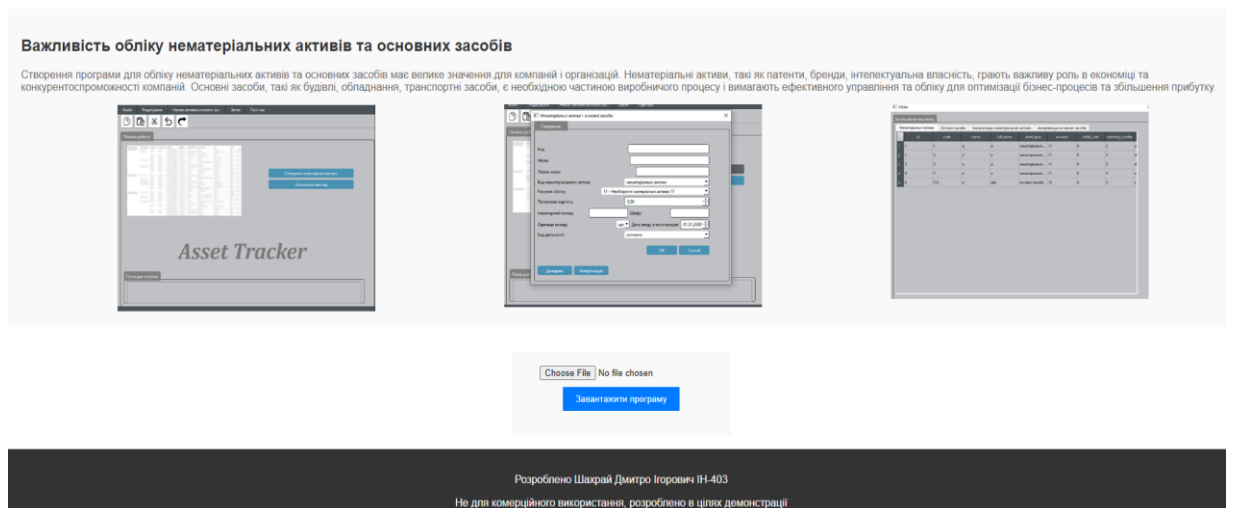


Рис. 4.2 Сайт для інсталяції програми

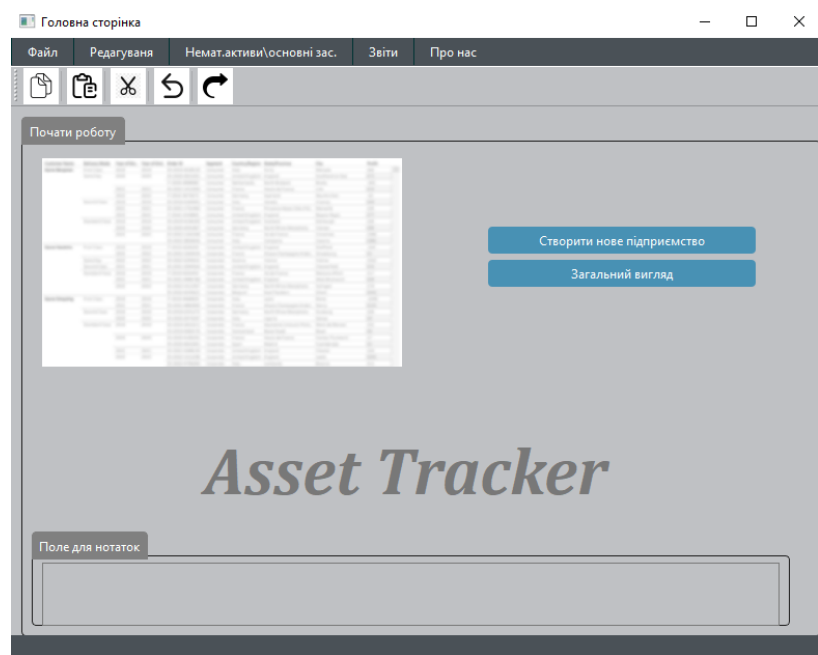


Рис. 4.3 Головна сторінка інформаційної системи

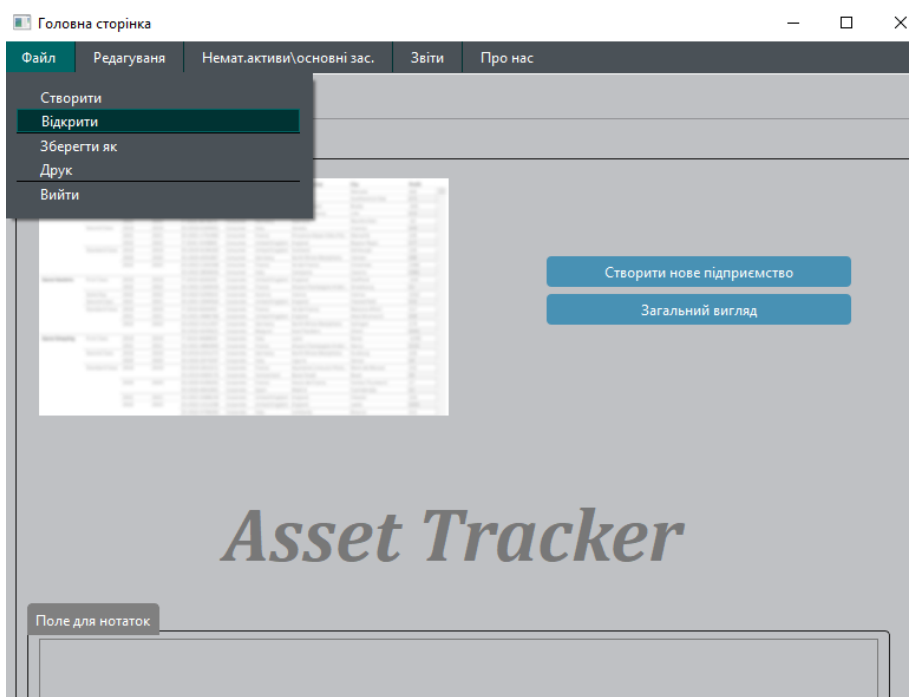


Рис. 4.4 Головна сторінка із зображенням «Файл панелі» інформаційної системи

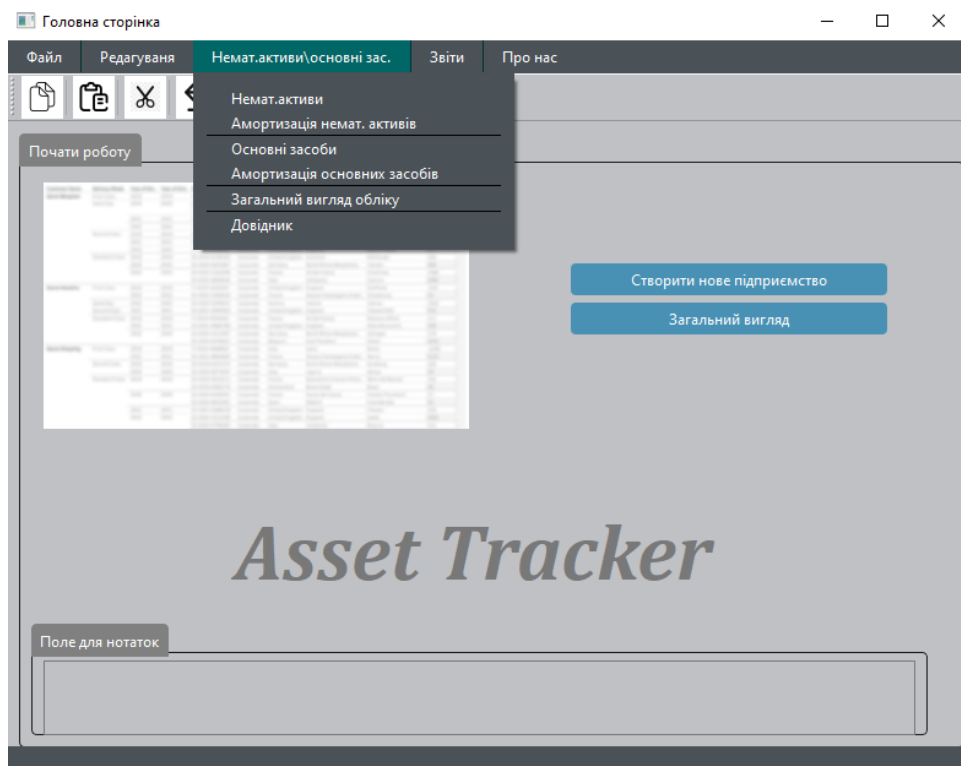


Рис. 4.5 Головна сторінка із зображенням «Нематеріальні активи\основні засоби панелі» інформаційної системи

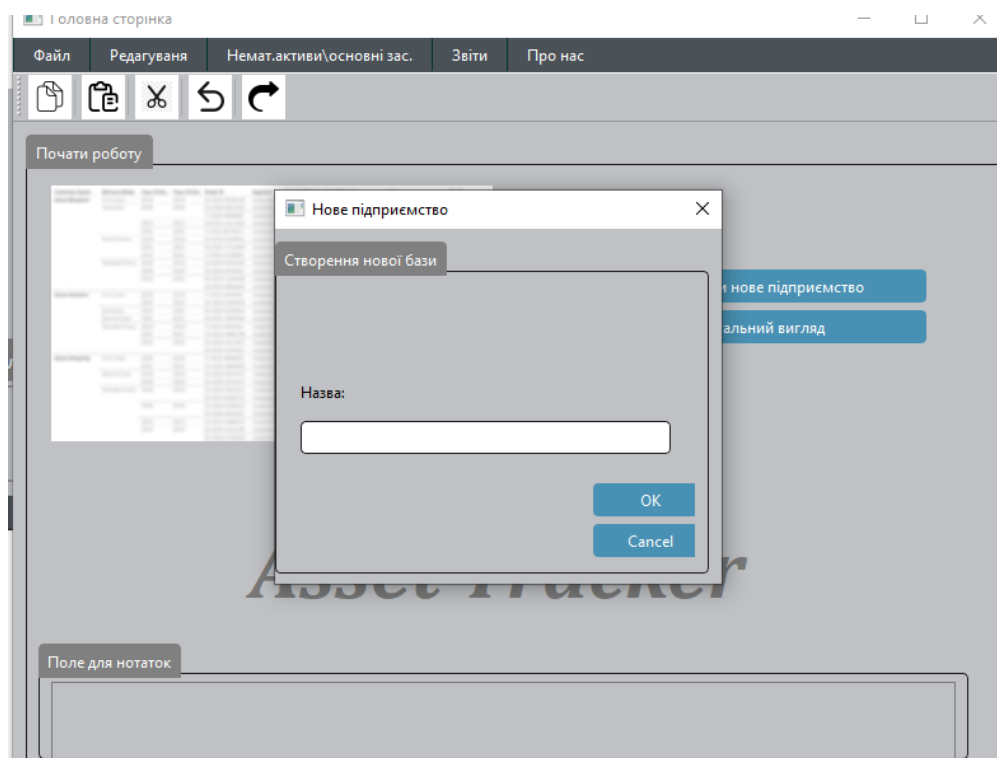


Рис. 4.6 Вікно створення бази підприємства інформаційної системи

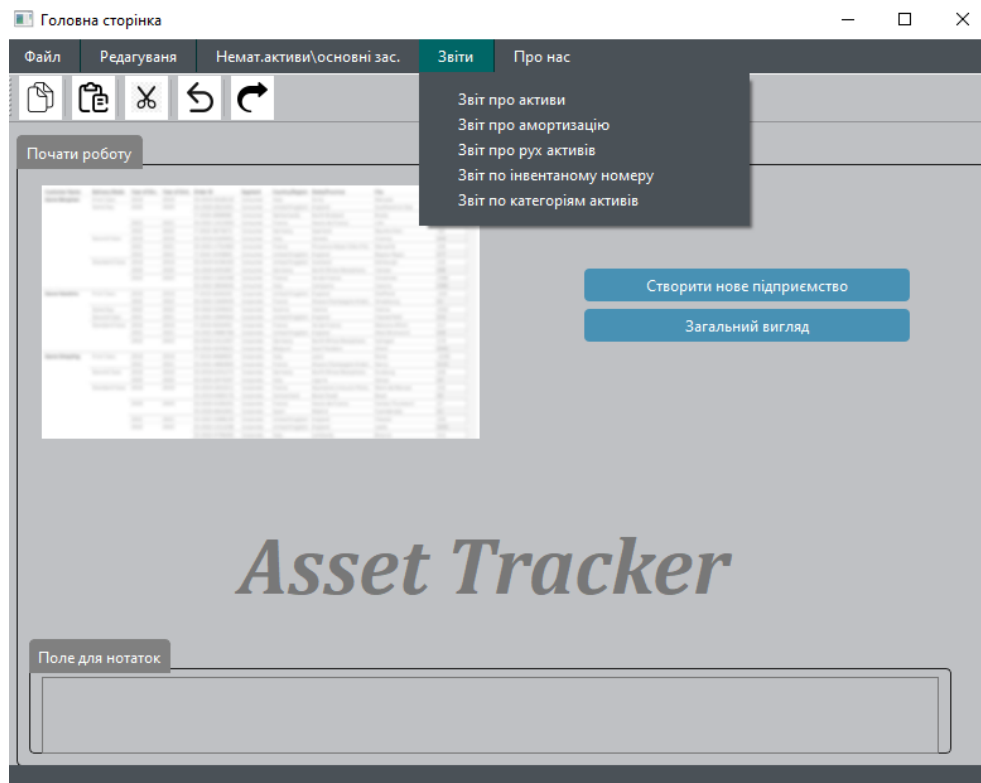


Рис. 4.7 Головна сторінка із зображенням «Звіт панелі» інформаційної системи

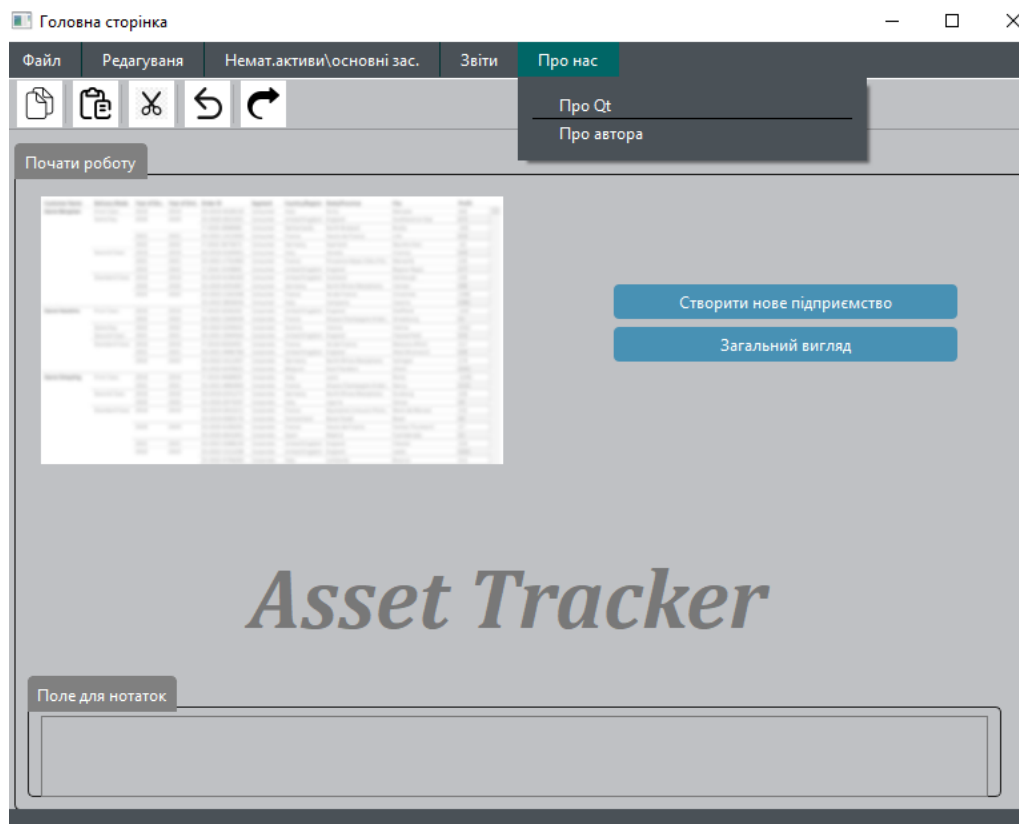


Рис. 4.8 Головна сторінка із зображенням «Про нас панелі» інформаційної системи

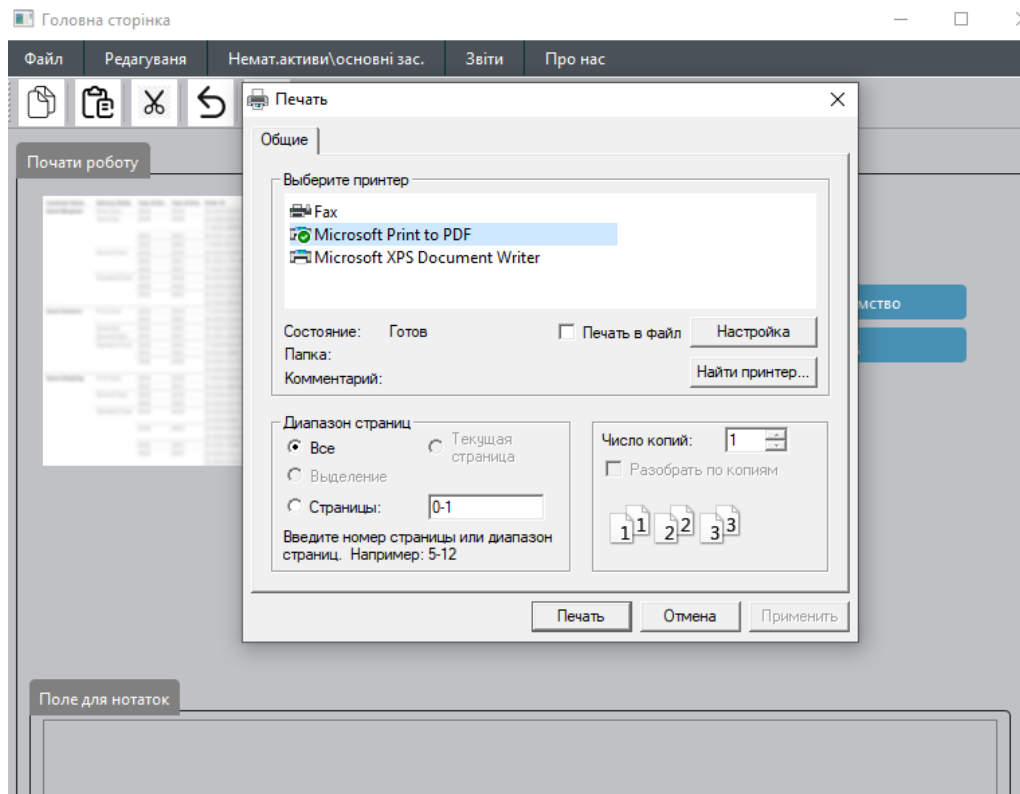


Рис. 4.9 Вікно друку бази підприємства інформаційної системи

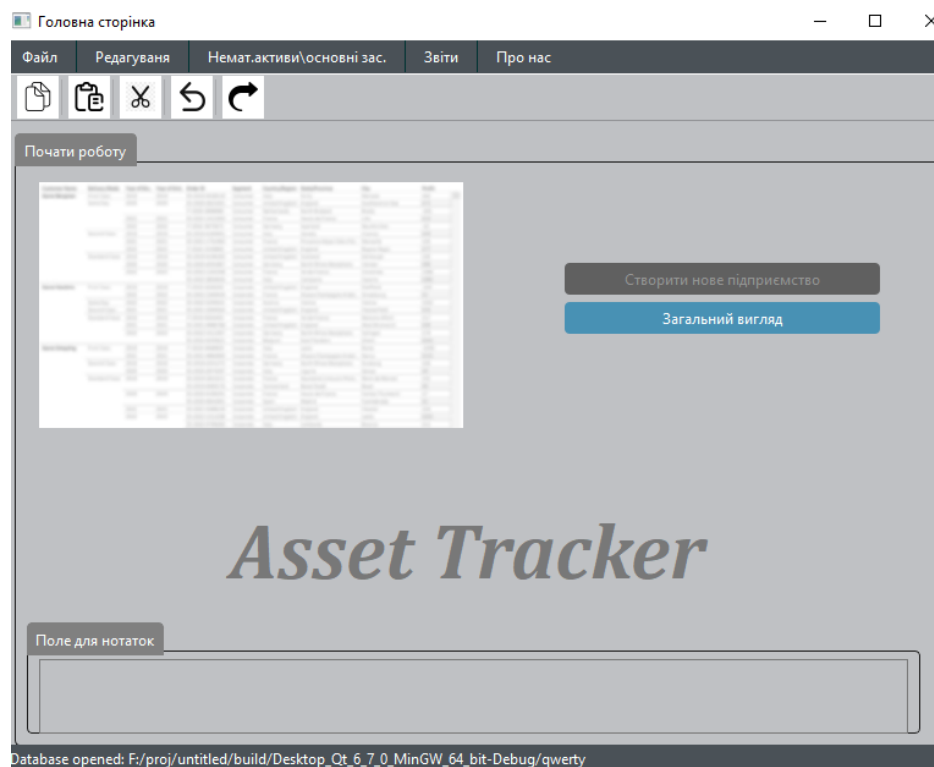


Рис. 4.10 Головне вікно після відкриття бази інформаційної системи

Облік

Загальний вигляд обліку

	Нематеріальні активи		Основні засоби		Амортизація нематеріальних активів		Амортизація основних засобів	
	id	code	name	full_name	asset_type	account	initial_cost	inventory_number
1	1	1	q	a	нематеріальні ...	11	0	2
2	2	3	e	a	нематеріальні ...	11	0	3
3	3	3	e	a	нематеріальні ...	11	0	3
4	4	3	e	a	нематеріальні ...	11	0	3
5	6	213	a	ads	основні засоби	10	0	3

Рис. 4.11 Загальний вигляд таблиці «Нематеріальні активи» підприємства інформаційної системи

Облік

Загальний вигляд обліку

Нематеріальні активи

Основні засоби

Амортизація нематеріальних активів

Амортизація основних засобів

	id	asset_id	expense_type	preciation_methc	salvage_value	depreciation_term
1	6	1	Основні засоби	Виробничий ...	123	2000-01-01

Рис. 4.12 Загальний вигляд таблиці «Амортизація нематеріальні активи» підприємства інформаційної системи

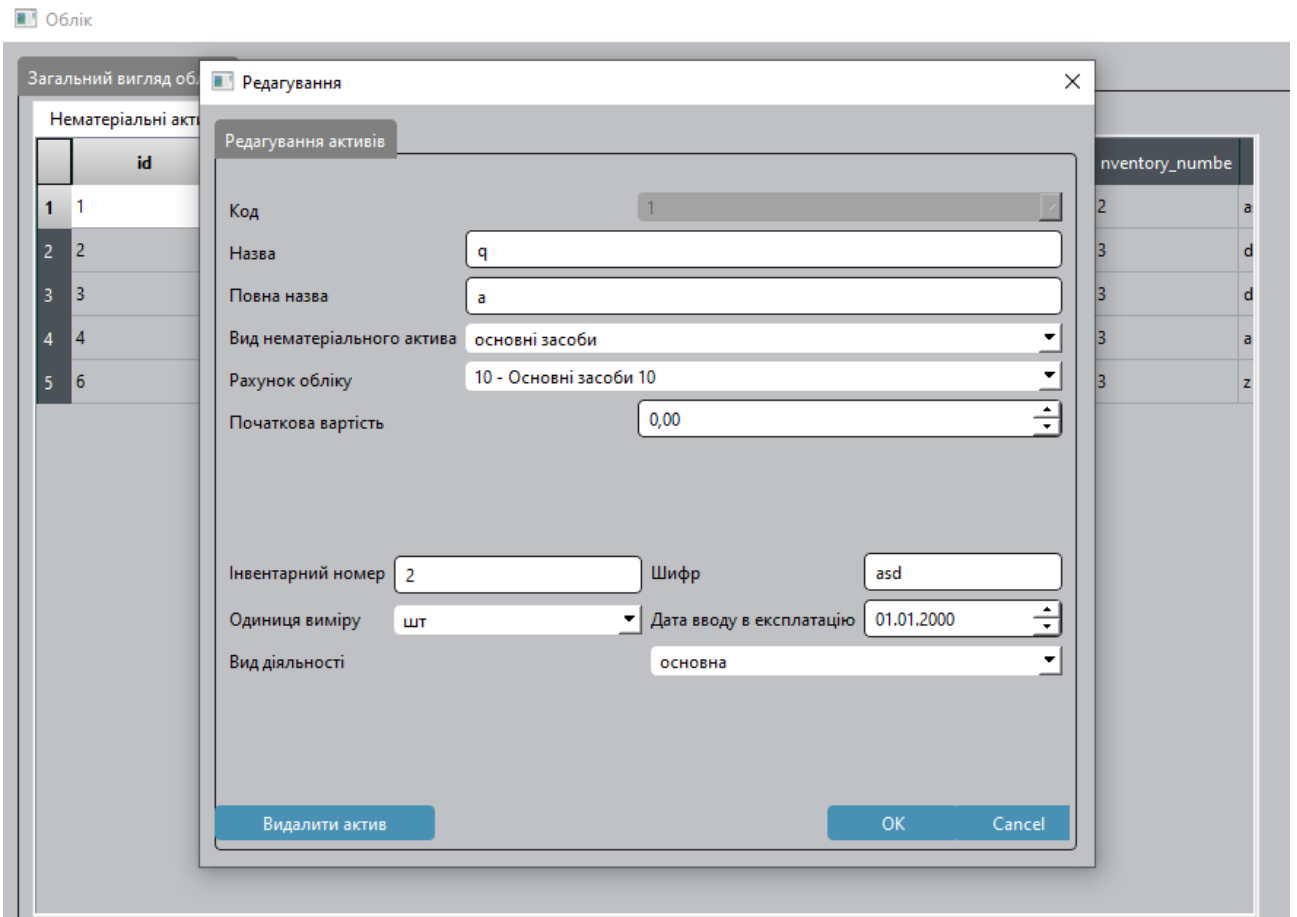


Рис. 4.13 Вікно редагування активів підприємства інформаційної системи

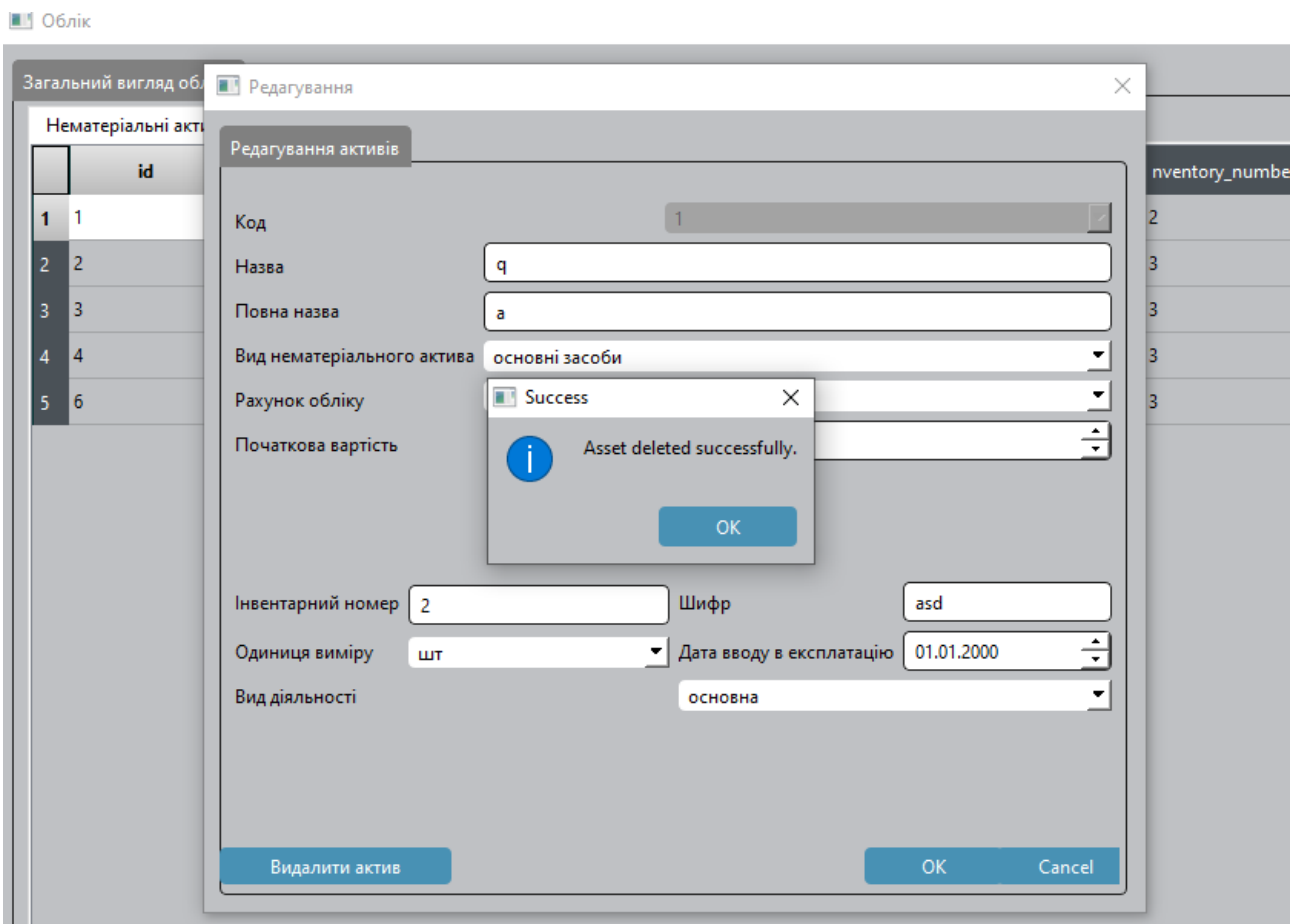


Рис. 4.14 Вікно видалення активу підприємства інформаційної системи

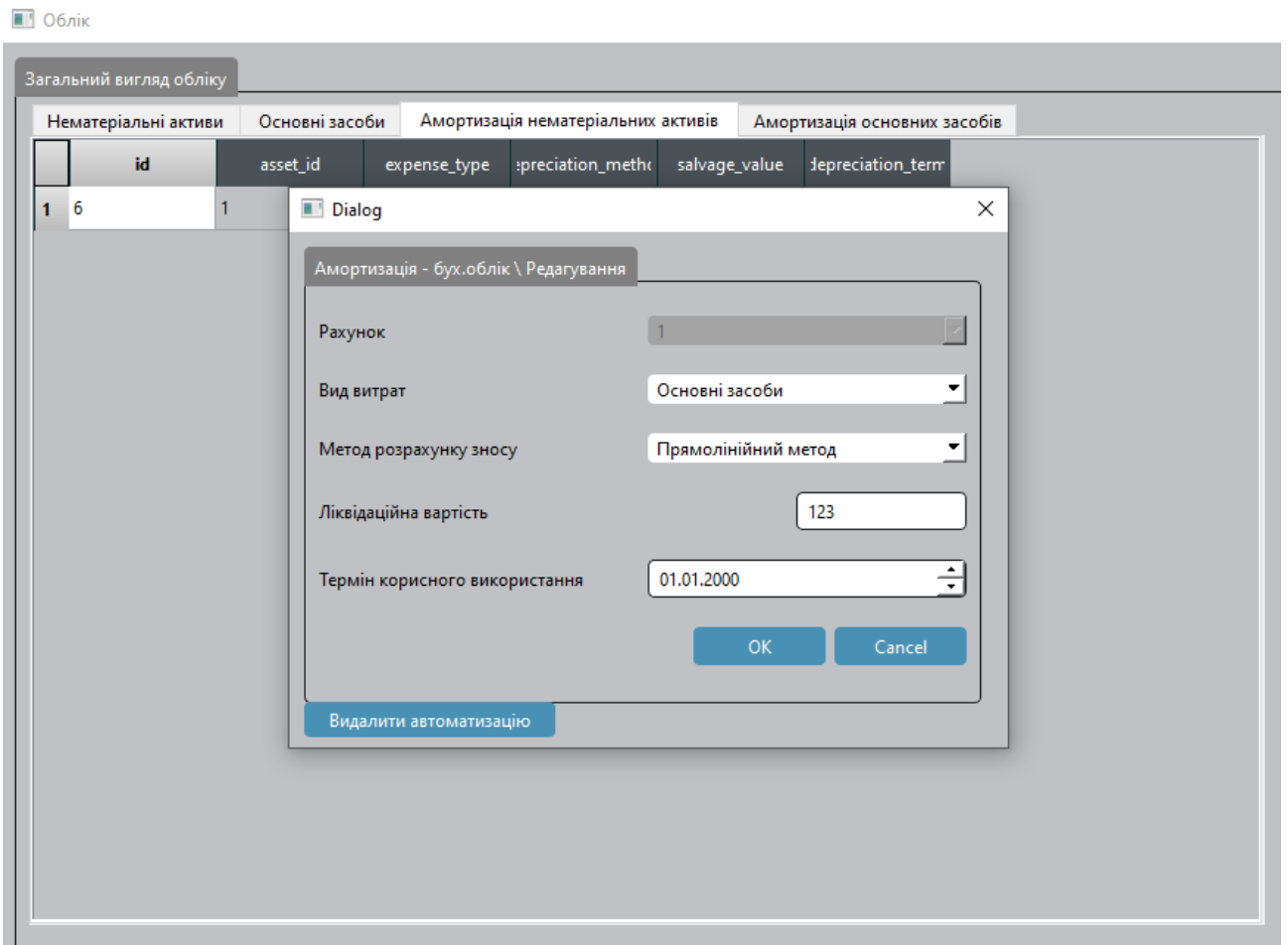


Рис. 4.15 Вікно редагування амортизації підприємства інформаційної системи

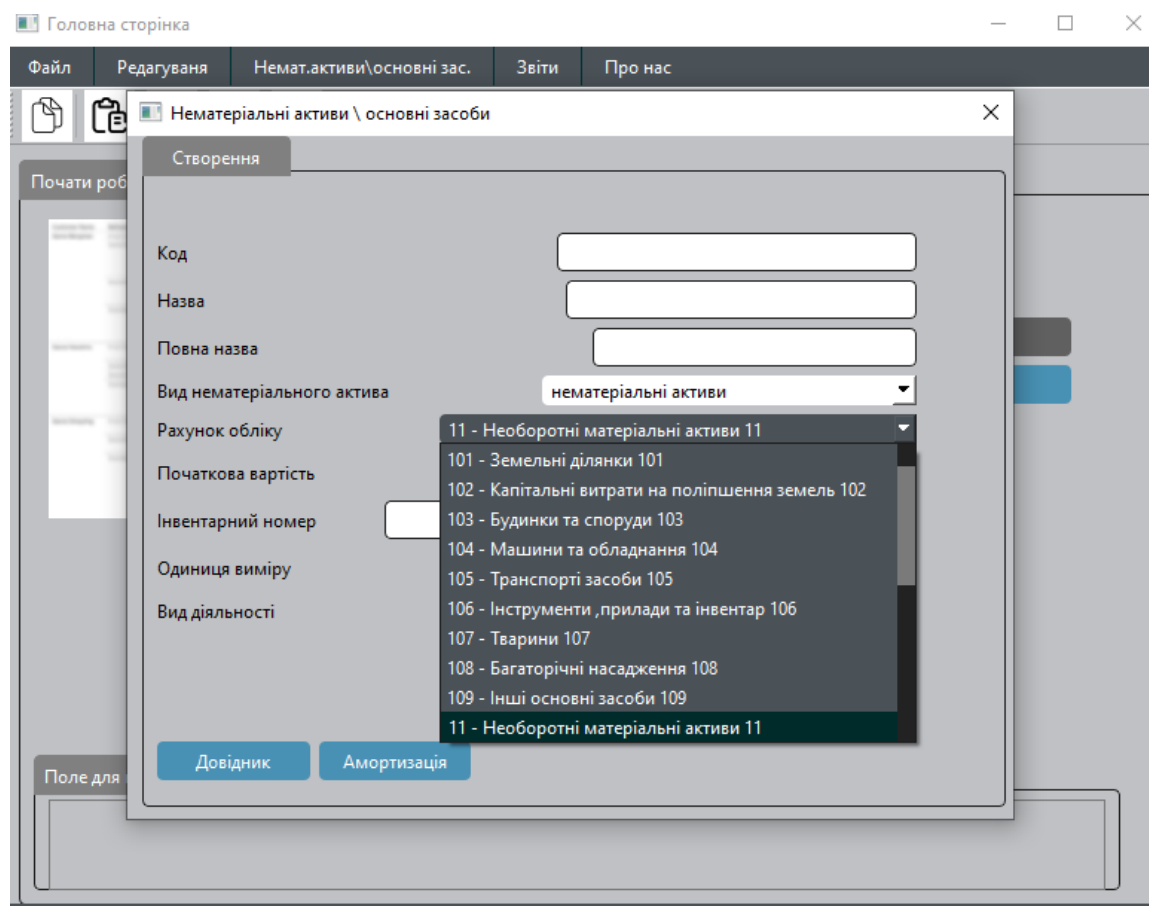


Рис. 4.16 Вікно створення активів підприємства інформаційної системи

Нематеріальні активи

Нематеріальні активи - це активи, які не мають фізичної форми, але мають цінність для підприємства. Вони включають інтелектуальну власність та інші активи, які можуть приносити дохід або забезпечувати конкурентні переваги. Приклади нематеріальних активів:

Патенти — права на винаходи, які забезпечують ексклюзивне використання та захист від копіювання іншими.

Авторські права - права на літературні, музичні та інші художні твори, що забезпечують захист від несанкціонованого використання.

Торгові марки — це символи, назви або логотипи, які відрізняють товари або послуги однієї компанії від інших.

Франшизи - права на використання бізнес-моделі або бренду іншої компанії.

Ділова репутація (гудвілл) — нематеріальна цінність компанії, пов'язана з її репутацією, клієнтською базою та іншими нематеріальними факторами, які можуть підвищити її ринкову вартість.

Програмне забезпечення — ліцензії на використання певних програмних продуктів.

Основні засоби

Основні кошти - це матеріальні активи, які використовуються у виробничій діяльності компанії та мають тривалий термін служби. Вони включають фізичні об'єкти, які необхідні для функціонування підприємства.

Приклади основних засобів:

Земля — ділянки землі, що використовуються для розміщення будівель, сільськогосподарських потреб чи інших цілей.

Будівлі та споруди — офісні будівлі, фабрики, склади та інші будівлі, що використовуються для ведення бізнесу.

Машини та обладнання — виробничі машини, інструменти, транспортні засоби та інші пристрої, що використовуються у виробничому процесі.

Комп'ютери та офісна техніка — комп'ютери, принтери, сканери та інші пристрої, необхідні для адміністративної діяльності.

Меблі та офісне обладнання - столи, стільці, шафи та інші предмети, що використовуються в офісах та інших робочих місцях.

Відмінність нематеріальних активів від основних засобів:

Фізична форма:

Нематеріальні активи: Не мають фізичної форми.

Основні засоби: Мають фізичну форму.

Природа використання:

Нематеріальні активи: В основному пов'язані з правами, інтелектуальною власністю та іншими нематеріальними перевагами.

Основні засоби: Використовуються у виробничій та адміністративній діяльності підприємства, забезпечуючи фізичну інфраструктуру.

Спосіб амортизації:

Нематеріальні активи: Амортизація зазвичай розраховується на основі терміну корисного використання активів, який може бути обмежений або необмежений.

Основні засоби: Амортизація розраховується на основі фізичного зносу та терміну служби активів.

OK

Рис. 4.17 Вікно довідка інформаційної системи

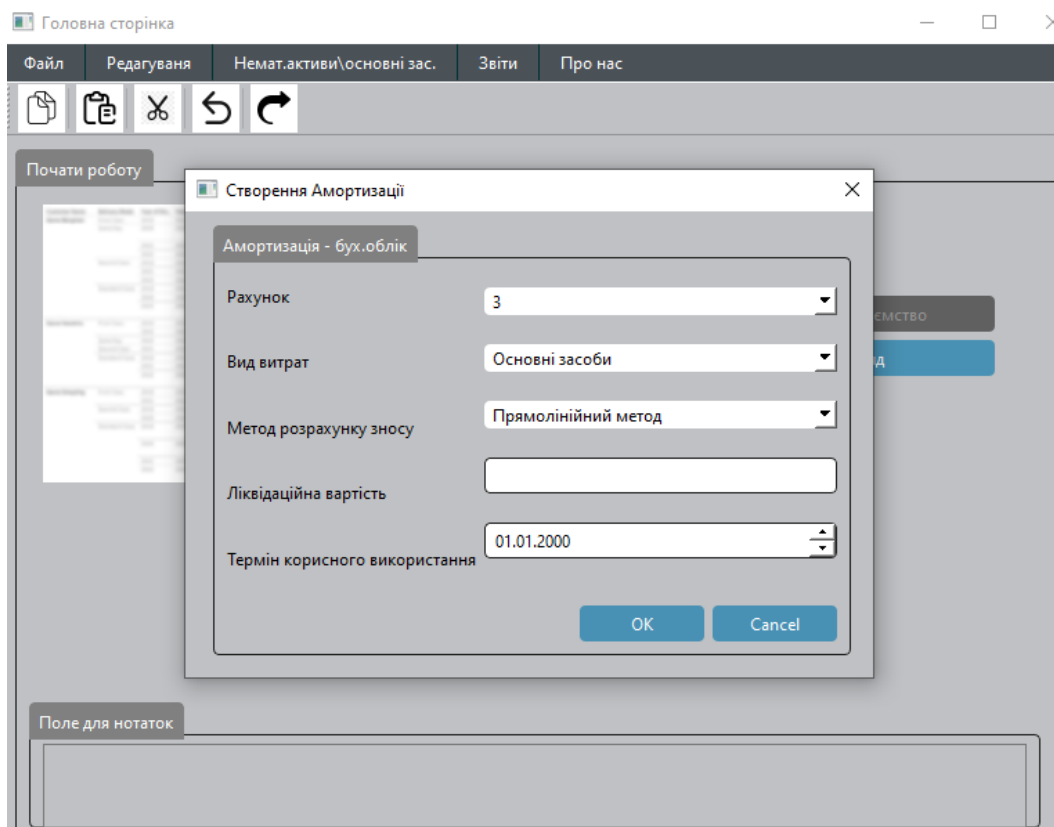


Рис. 4.18 Вікно створення амортизації інформаційної системи

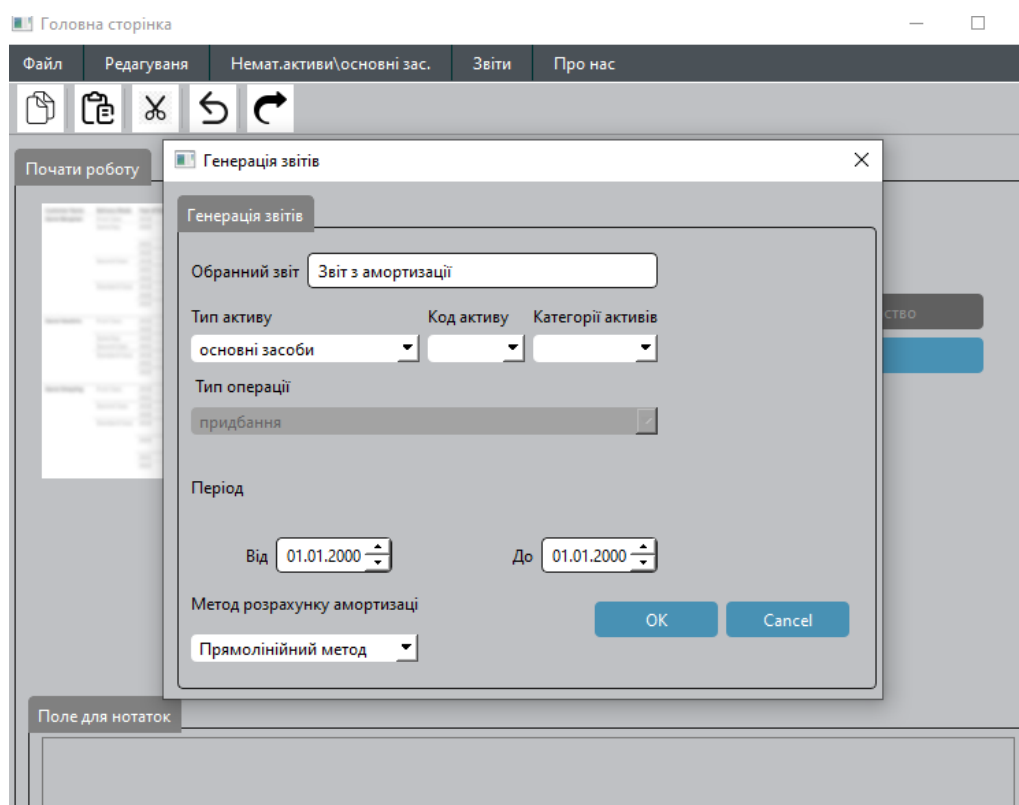


Рис. 4.19 Вікно генерації звітів «Звіт амортизації» підприємства інформаційної системи

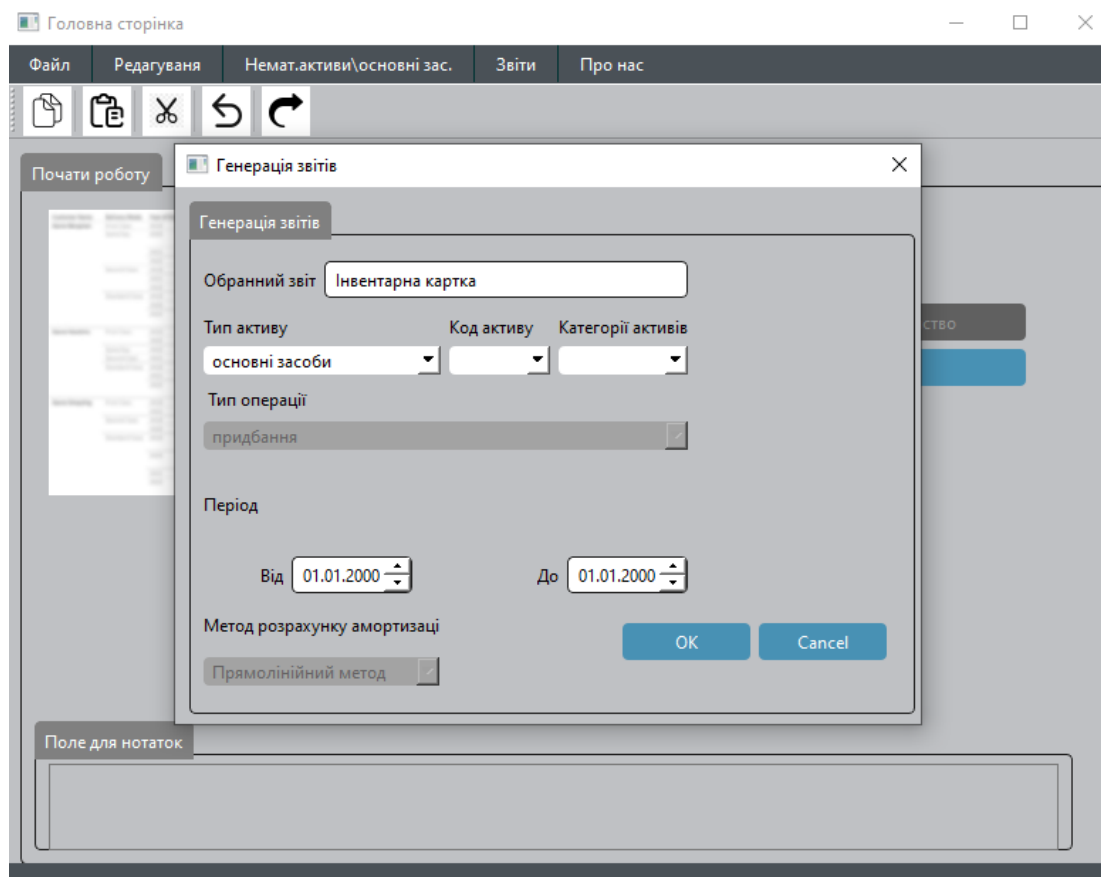


Рис. 4.20 Вікно генерації звітів «Звіт інвентарна картка» підприємства інформаційної системи

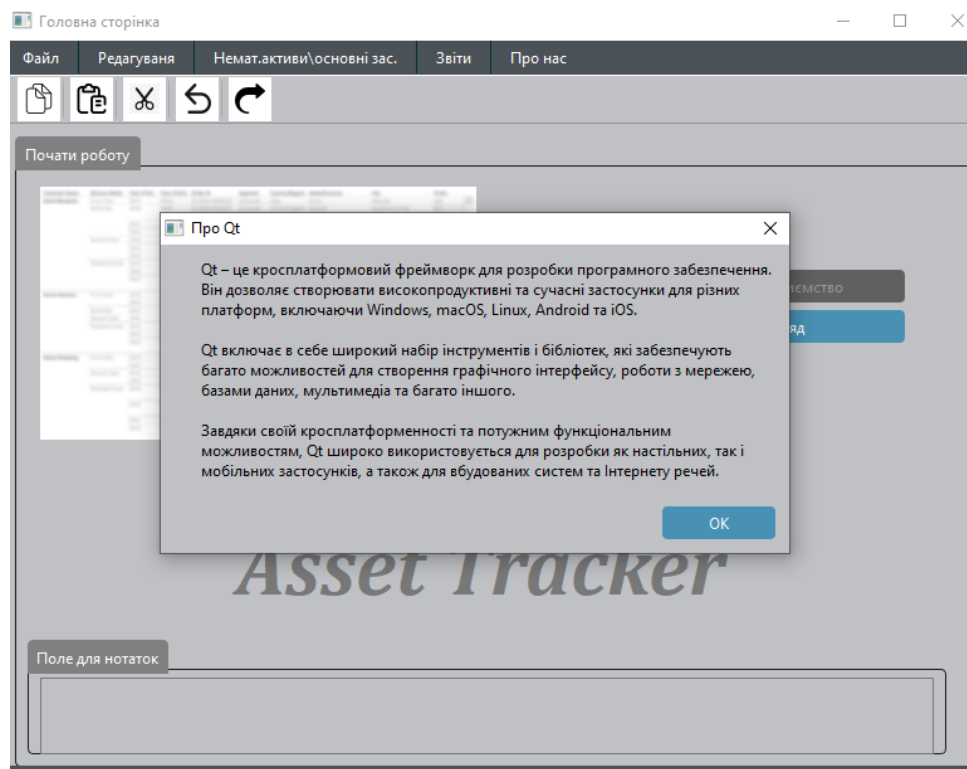


Рис. 4.21 Вікно «Про QT» інформаційної системи

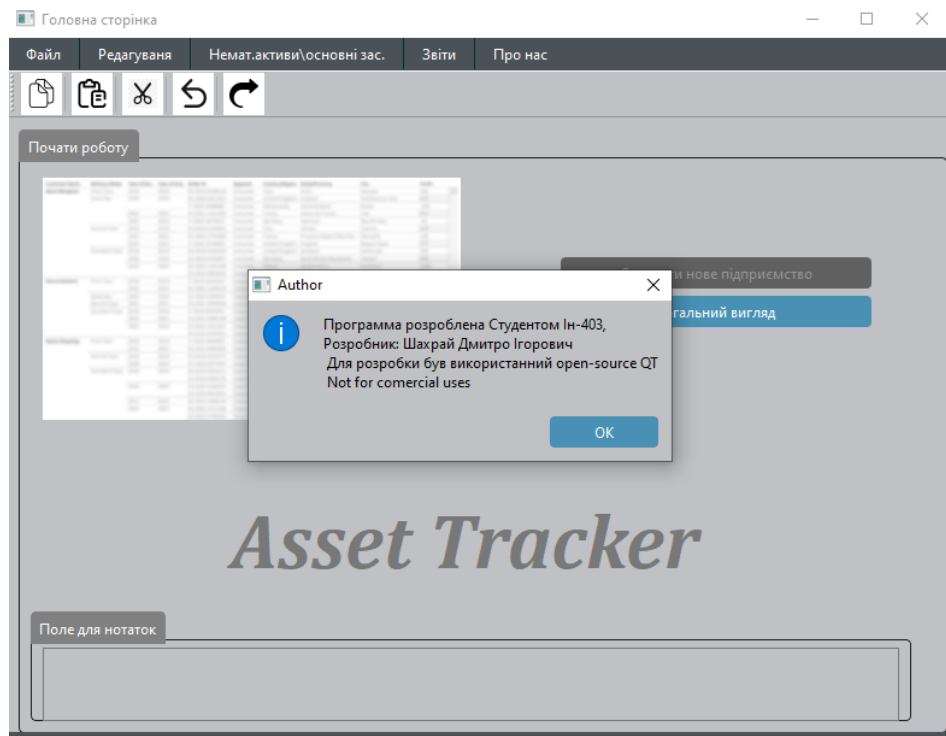


Рис. 4.22 Вікно «Про Автора» інформаційної системи

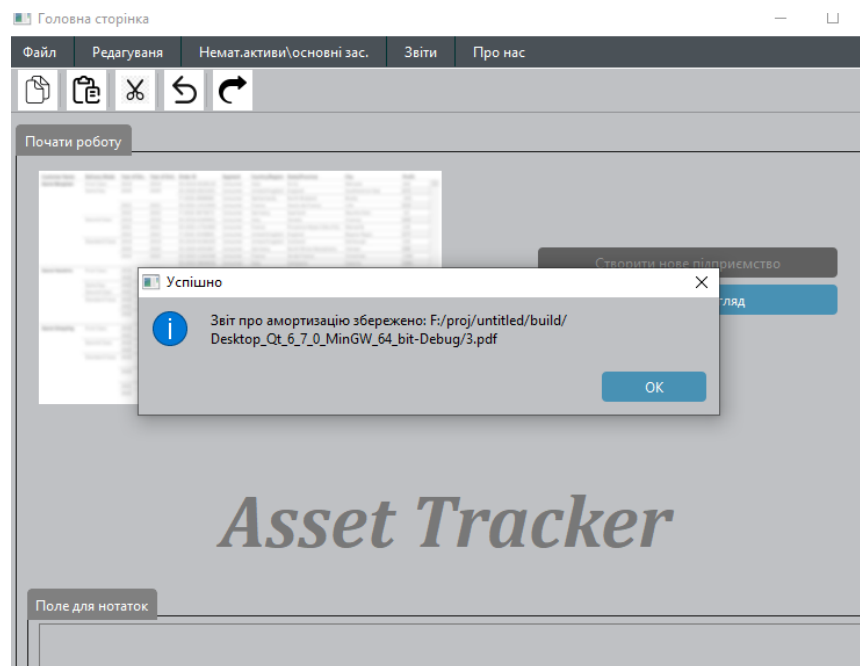


Рис. 4.23 Вікно збереження звіту бази підприємства інформаційної системи

Інвентарна картка актива		
Назва актива: е		
Код актива: 3		
Тип актива: нематеріальні активи		
Дата вступу: 01.01.2000		
Початкова вартість: 0		
Інвентарний номер: 3		
Рахунок: 11		
Одиниця міри: шт		
Додаткова інформація об активі:		
Дата	Тип операції	Опис
01.01.2023	Покупка	Куплений комп'ютер для відділу бухгалтерії
		Дата: 09.06.2024
		Підпис: _____

Рис. 4.24 Звіт «Інвентарна картка» підприємства інформаційної системи

Звіт по категорії активів

Обраний категорій активу основні засоби

ID	Код	Назва	Повна назва	Тип актива	Рахунок	Поч. вар	Інв.номер	Шифр	Один.вим	Дата вводу	Тип акту
2	3	е	а	основні засоби	11 0	3	d	шт	2000-01-01	основна	

Дата:

10.06.2024

Підпис:

Рис. 4.25 Звіт «Звіт по категоріям активів» підприємства інформаційної системи

Звіт про активи

ID	Код	Назва	Повна назва	Тип актива	Рахунок	Поч. вар	Інв.номер	Шифр	Один.вим	Дата вводу	Тип акт.
2	3	е	а	нематеріальні активи	11	0	3	d	шт	2000-01-01	основна
3	3	е	а	нематеріальні активи	11	0	3	d	шт	2000-01-01	основна
4	3	е	а	нематеріальні активи	11	0	3	a	шт	2000-01-01	основна

Дата:

11.06.2024

Підпис:

Рис. 4.26 Звіт «Звіт про активи» підприємства інформаційної системи

Звіт про переміщення активів

Назва активу	Код активу	Тип операції	
е	3	придбання	основна
е	3	вибуття	основна
е	3	придбання	додаткова

Місце знаходження об'єкта після переміщення _____

Передається актив код активу _____

Одиниця міри _____

Дата: 10.06.2024

Підпис: _____

Рис. 4.27 Вікно Звіту «Звіт про рух активу» підприємства інформаційної системи

Звіт про амортизацію

Назва	Код	Тип активу	Дата вступу	Початкова вартість	Остаточна вартість	Строк служби	Метод амортизації
е	3	нематеріальні актив	01.01.1998	100000	40000	5	прямоліній метод

Величина амортизації

30000

Дата: 10.06.2024

Підпис: _____

Рис. 4.28 Звіт «Звіт про амортизацію прямолінійним методом» підприємства інформаційної системи

Звіт про амортизацію							
Назва	Код	Тип активу	Дата вступу	Початкова вартість	Остаточна вартість	Строк служби	Метод амортизації
е	3	нематеріальні актив	01.01.1998	100000	40000	5	зменшеного залишку
Величина амортизації		Результат амортизації					
3000		7000					

Рис. 4.29 Звіт «Звіт про амортизацію методом зменшеного залишку» підприємства інформаційної системи

Звіт про амортизацію							
Назва	Код	Тип активу	Дата вступу	Початкова вартість	Остаточна вартість	Строк служби	Метод амортизації
е	3	нематеріальні актив	01.01.1998	100000	40000	5	виробничий метод
Величина амортизації		Кількість одиниць		Результат амортизації			
3000		7		700			
Дата: 10.06.2024							
Підпис: _____							

Рис. 4.30 Звіт «Звіт про амортизацію виробничим методом» підприємства інформаційної системи

Додаток А

Main.html

```
<!DOCTYPE html>
```

```
<html lang="uk">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>Asset Tracker</title>
```

```
<link rel="stylesheet" href="styles.css">
```

```
</head>
```

```
<body>
```

```
<header>
```

```
<div class="header-inner">
```

```
<h1>Asset Tracker</h1>
```

```
</div>
```

```
</header>
```

```
<main>
```

```
<div class="panel">
```

```
<h2>Про технологію Qt</h2>
```

<p>Qt - це кросплатформений фреймворк для розробки програм з графічним інтерфейсом. Він дозволяє створювати програми, які можуть працювати на різних операційних системах, таких як Windows, macOS та Linux, використовуючи один і той же вихідний код.</p>

```
<div class="img_div">
```

```

```

```

```

```

```

```
</div>
```

```
</div>
```

```
<div class="panel">
```

```
<h2>Важливість обліку нематеріальних активів та основних засобів</h2>
```

<p>Створення програми для обліку нематеріальних активів та основних засобів має велике значення для компаній і організацій. Нематеріальні активи, такі як патенти, бренди, інтелектуальна власність, грають важливу роль в економіці та конкурентоспроможності компаній. Основні засоби, такі як будівлі, обладнання, транспортні засоби, є необхідною частиною виробничого процесу і вимагають ефективного управління та обліку для оптимізації бізнес-процесів та збільшення прибутку.</p>

```
<div class="img_div">
```


</div>

</div>

<div class="panel">

<form id="programForm" enctype="multipart/form-data">

<input type="file" id="programInput" accept=".exe, .zip" required>

<button type="submit">Завантажити програму</button>

</form>

<p id="status"></p>

</div>

</main>

<footer>

<div class="footer-inner">

<p>Розроблено Шахрай Дмитро Ігорович ІН-403</p>

<p>Не для комерційного використання, розроблено в цілях демонстрації</p>

</div>

</footer>

<script src="script.js"></script>

</body>

</html>

Styles.css

/ Остальные стили остаются без изменений /

```
body {  
    font-family: Arial, sans-serif;  
    margin: 0;  
    padding: 0;  
}
```

```
header {  
    background-color: #333;  
    color: #fff;  
    text-align: center;  
    padding: 20px 0;  
}
```

```
main {  
    display: flex;  
    flex-direction: column;  
    align-items: center;  
    margin-top: 20px;  
}
```

```
form {  
    width: 300px;  
    display: flex;  
    flex-direction: column;  
    align-items: center;  
}
```

```
input[type="file"] {  
    margin-bottom: 10px;  
}
```

```
button {  
    padding: 10px 20px;  
    background-color: #007bff;  
    color: #fff;  
    border: none;  
    cursor: pointer;
```

```
}
```

```
button:hover {  
    background-color: #0056b3;  
}
```

```
header {  
    background-color: #333;  
    color: #fff;  
    text-align: center;  
    padding: 20px 0;  
}
```

```
.header-inner {  
    width: 80%;  
    margin: 0 auto;  
}
```

```
footer {  
    background-color: #333;  
    color: #fff;  
    padding: 20px 0;  
    text-align: center;  
}
```

```
.footer-inner {  
    width: 80%;  
    margin: 0 auto;  
}
```

```
.panel {  
    background-color: #f9f9f9;  
    padding: 20px;  
    padding: 20px;  
    margin: 20px 0;  
}
```

```
.panel img_div{
```

```
        width:100%;
        float:left;
    }
    .panel h2 {
        color: #333;
    }

    .panel p {
        color: #666;
    }
    .panel img {

        width: 400px;
        height: 100%;
        display: block;
        margin: 0 auto;
        margin-left:150px;
        margin-right:50px;
        float : left;
    }
```

Scripts.js

```
document.getElementById('programForm').addEventListener('submit', function(event) {  
    event.preventDefault();
```

```
    var form = event.target;
```

```
    var formData = new FormData(form);
```

```
    var status = document.getElementById('status');
```

```
    var xhr = new XMLHttpRequest();
```

```
    xhr.open('POST', 'upload.php', true);
```

```
    xhr.onload = function() {
```

```
        if (xhr.status === 200) {
```

```
            status.textContent = 'Программа успішно завантажена!';
```

```
        } else {
```

```
            status.textContent = 'Помилка завантаження програми.';
```

```
        }
```

```
    };
```

```
    xhr.send(formData);
```

```
});
```

Додаток Б

SQLite код:

```
CREATE DATABASE assets;  
GO  
USE assets;  
GO
```

-- Table for intangible assets

```
CREATE TABLE intangible_assets (  
  id INT PRIMARY KEY IDENTITY, -- Auto-incrementing integer for unique identifier  
  code NVARCHAR(50) NOT NULL, -- Textual code for the asset  
  name NVARCHAR(255) NOT NULL, -- Name of the intangible asset  
  full_name NVARCHAR(255), -- Optional: Full name or description of the asset  
  asset_type NVARCHAR(50), -- Type of intangible asset (e.g., Software, Patent)  
  account NVARCHAR(50), -- Account code for financial purposes  
  initial_cost MONEY NOT NULL, -- Initial cost of the asset  
  inventory_number NVARCHAR(50), -- Optional: Inventory or tracking number  
  cipher NVARCHAR(50), -- Optional: Internal code or identifier  
  unit NVARCHAR(50), -- Unit of measurement (e.g., License)  
  entry_date DATE NOT NULL, -- Date the asset was acquired  
  activity_type NVARCHAR(50) -- Type of activity related to the asset (e.g., Acquisition, Disposal)  
);
```

-- Table for tangible assets

```
CREATE TABLE tangible_assets (  
  id INT PRIMARY KEY IDENTITY, -- Auto-incrementing integer for unique identifier  
  code NVARCHAR(50) NOT NULL, -- Textual code for the asset  
  name NVARCHAR(255) NOT NULL, -- Name of the tangible asset  
  full_name NVARCHAR(255), -- Optional: Full name or description of the asset  
  asset_type NVARCHAR(50), -- Type of tangible asset (e.g., Equipment, Building)  
  account NVARCHAR(50), -- Account code for financial purposes  
  initial_cost MONEY NOT NULL, -- Initial cost of the asset  
  inventory_number NVARCHAR(50), -- Optional: Inventory or tracking number  
  cipher NVARCHAR(50), -- Optional: Internal code or identifier  
  unit NVARCHAR(50), -- Unit of measurement (e.g., Unit, Set)  
  entry_date DATE NOT NULL, -- Date the asset was acquired  
  activity_type NVARCHAR(50) -- Type of activity related to the asset (e.g., Acquisition, Disposal)  
);
```

-- Table for intangible asset depreciation

```
CREATE TABLE intangible_assets_depreciation (  
  id INT PRIMARY KEY IDENTITY, -- Auto-incrementing integer for unique identifier  
  asset_id INT NOT NULL, -- Foreign key referencing intangible_assets.id  
  expense_type NVARCHAR(50), -- Type of expense incurred during depreciation  
  depreciation_method NVARCHAR(50), -- Method used for calculating depreciation  
  salvage_value MONEY, -- Optional: Estimated salvage value at the end of useful life  
  depreciation_term DATE, -- Date the depreciation period ends  
  FOREIGN KEY (asset_id) REFERENCES intangible_assets(id)  
);
```

-- Table for tangible asset depreciation

```
CREATE TABLE tangible_assets_depreciation (  
  id INT PRIMARY KEY IDENTITY, -- Auto-incrementing integer for unique identifier  
  asset_id INT NOT NULL, -- Foreign key referencing tangible_assets.id  
  expense_type NVARCHAR(50), -- Type of expense incurred during depreciation  
  depreciation_method NVARCHAR(50), -- Method used for calculating depreciation  
  salvage_value MONEY, -- Optional: Estimated salvage value at the end of useful life  
  depreciation_term DATE, -- Date the depreciation period ends  
  FOREIGN KEY (asset_id) REFERENCES tangible_assets(id)  
);
```

-- Table for asset locations

```
CREATE TABLE asset_locations (  
  id INT PRIMARY KEY IDENTITY, -- Auto-incrementing integer for unique identifier
```

```

location_name NVARCHAR(255) NOT NULL, -- Name of the location
description NVARCHAR(255) -- Optional: Description of the location
);

-- Table for asset movements
CREATE TABLE asset_movements (
id INT PRIMARY KEY IDENTITY, -- Auto-incrementing integer for unique identifier
asset_id INT NOT NULL, -- Foreign key referencing either tangible_assets.id or intangible_assets.id
asset_type NVARCHAR(50) NOT NULL, -- Type of asset (either 'tangible' or 'intangible')
from_location_id INT NOT NULL, -- Foreign key referencing asset_locations.id (initial location)
to_location_id INT NOT NULL, -- Foreign key referencing asset_locations.id (destination location)
movement_date DATE NOT NULL, -- Date of the movement
description NVARCHAR(255), -- Optional: Description of the movement
FOREIGN KEY (from_location_id) REFERENCES asset_locations(id),
FOREIGN KEY (to_location_id) REFERENCES asset_locations(id)
);

-- Alter table tangible_assets to add a location_id column
ALTER TABLE tangible_assets
ADD location_id INT NULL;

-- Add foreign key constraint to tangible_assets
ALTER TABLE tangible_assets
ADD FOREIGN KEY (location_id) REFERENCES asset_locations(id);

-- Alter table intangible_assets to add a location_id column
ALTER TABLE intangible_assets
ADD location_id INT NULL;

-- Add foreign key constraint to intangible_assets
ALTER TABLE intangible_assets
ADD FOREIGN KEY (location_id) REFERENCES asset_locations(id);

```

Додаток В

Файл код Untitled.pro

QT += core gui widgets sql printsupport

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++17

You can make your code fail to compile if it uses deprecated APIs.

In order to do so, uncomment the following line.

#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000 # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \

authomation.cpp \

dbcreate.cpp \

editactive.cpp \

editauthomation.cpp \

generatereport.cpp \

main.cpp \

mainwindow.cpp \

nonematerialactives.cpp \

tablepage.cpp

HEADERS += \

authomation.h \

dbcreate.h \

editactive.h \

editauthomation.h \

generatereport.h \

mainwindow.h \

nonematerialactives.h \

tablepage.h

FORMS += \

authomation.ui \

dbcreate.ui \

editactive.ui \

editauthomation.ui \

```
generatereport.ui \  
mainwindow.ui \  
nonematerialactives.ui \  
tablepage.ui
```

```
# Default rules for deployment.
```

```
qnx: target.path = /tmp/${TARGET}/bin
```

```
else: unix:!android: target.path = /opt/${TARGET}/bin
```

```
!isEmpty(target.path): INSTALLS += target
```

```
RESOURCES += \  
resources.qrc
```

Automation.cpp

```
#include "authomation.h"
```

```
#include "ui_authomation.h"
```

```
#include <QMessageBox>
```

```
#include <QSqlQuery>
```

```
#include <QSqlError>
```

```
#include <QDebug> // для отладки
```

```
Authomation::Authomation(bool isIntangible, QSqlDatabase database, QWidget parent)
```

```
    : QDialog(parent)
```

```
    , ui(new Ui::Authomation)
```

```
    , db(database)
```

```
{
```

```
    ui->setUpUi(this);
```

```
    if (!db.isOpen()) {
```

```
        QMessageBox::critical(this, "Databaspe Error", "Failed to open the database: " + db.lastError().text());
```

```
        return;
```

```
    }
```

```
    QSqlQuery query;
```

```
    if (isIntangible) {
```

```
        query.exec("SELECT code, name FROM intangible_assets");
```

```
        while (query.next()) {
```

```
            QString code = query.value(0).toString();
```

```
            QString name = query.value(1).toString();
```

```
            qDebug() << "Intangible Asset - Code:" << code << ", Name:" << name; // отладочное сообщение
```

```
            ui->comboBox_assetId->addItem(code, code);
```

```
        }
```

```
        connect(ui->buttonBox, &QDialogButtonBox::accepted, this, &Authomation::addIntangibleAssetDepreciation);
```

```
    } else {
```

```
        query.exec("SELECT code, name FROM tangible_assets");
```

```
        while (query.next()) {
```

```
            QString code = query.value(0).toString();
```

```
            QString name = query.value(1).toString();
```

```
            qDebug() << "Tangible Asset - Code:" << code << ", Name:" << name; // отладочное сообщение
```

```
            ui->comboBox_assetId->addItem(code, code);
```

```
        }
```

```
        connect(ui->buttonBox, &QDialogButtonBox::accepted, this, &Authomation::addTangibleAssetDepreciation);
```

```
    }
```

```
ui->comboBox_expenseType->addItem({"Основні засоби", "Інші необоротні активи", "Нематеріальні активи", "Гудвіл", "Інші нематеріальні активи"});
```

```
ui->comboBox_depreciationMethod->addItem({"Прямолінійний метод", "Зменшений залишок", "Виробничий метод"});
```

```
}
```

```
Authomation::~Authomation()
```

```
{
```

```
    delete ui;
```

```
}
```

```
void Authomation::addIntangibleAssetDepreciation() {
```

```
    int assetId = ui->comboBox_assetId->currentData().toInt();
```

```
    QString expenseType = ui->comboBox_expenseType->currentText();
```

```
    QString depreciationMethod = ui->comboBox_depreciationMethod->currentText();
```

```
    double salvageValue = ui->lineEdit_salvageValue->text().toDouble();
```

```
    QDate depreciationTerm = ui->dateEdit_depreciationTerm->date();
```

```
    QSqlQuery query;
```

```
    query.prepare("INSERT INTO intangible_assets_depreciation (asset_id, expense_type, depreciation_method, salvage_value, depreciation_term) "
```

```
        "VALUES (:asset_id, :expense_type, :depreciation_method, :salvage_value, :depreciation_term)");
```

```
    query.bindValue(":asset_id", assetId);
```

```
    query.bindValue(":expense_type", expenseType);
```

```
    query.bindValue(":depreciation_method", depreciationMethod);
```

```
    query.bindValue(":salvage_value", salvageValue);
```

```
    query.bindValue(":depreciation_term", depreciationTerm);
```

```
    if (!query.exec()) {
```

```
        QMessageBox::critical(this, "Error", "Failed to insert depreciation record: " + query.lastError().text());
```

```
        qDebug() << "Failed to insert depreciation record: " << query.lastError().text();
```

```
        return;
```

```
    }
```

```
    QMessageBox::information(this, "Success", "Intangible asset depreciation record added successfully");
```

```
    accept(); // Закреть диалоговое окно после успешного добавления
```

```
}
```

```

void Authomation::addTangibleAssetDepreciation() {
    int assetId = ui->comboBox_assetId->currentData().toInt();
    QString expenseType = ui->comboBox_expenseType->currentText();
    QString depreciationMethod = ui->comboBox_depreciationMethod->currentText();
    double salvageValue = ui->lineEdit_salvageValue->text().toDouble();
    QDate depreciationTerm = ui->dateEdit_depreciationTerm->date();

    QSqlQuery query;

    query.prepare("INSERT INTO tangible_assets_depreciation (asset_id, expense_type, depreciation_method,
salvage_value, depreciation_term) "
        "VALUES (:asset_id, :expense_type, :depreciation_method, :salvage_value, :depreciation_term)");
    query.bindValue(":asset_id", assetId);
    query.bindValue(":expense_type", expenseType);
    query.bindValue(":depreciation_method", depreciationMethod);
    query.bindValue(":salvage_value", salvageValue);
    query.bindValue(":depreciation_term", depreciationTerm);

    if (!query.exec()) {
        QMessageBox::critical(this, "Error", "Failed to insert depreciation record: " + query.lastError().text());
        qDebug() << "Failed to insert depreciation record: " << query.lastError().text();
        return;
    }

    QMessageBox::information(this, "Success", "Tangible asset depreciation record added successfully");
    accept(); // Закрывать диалоговое окно после успешного добавления
}

```

```

Dbcreate.cpp

#include "dbcreate.h"
#include "ui_dbcreate.h"
#include "tablepage.h"
#include <QMessageBox>

dbcreate::dbcreate(QWidget parent)
    : QDialog(parent)
    , ui(new Ui::dbcreate)
{
    ui->setupUi(this);

}

dbcreate::~dbcreate()
{
    delete ui;
}

QString dbcreate::getDatabaseName() const
{
    return ui->lineEdit->text();
}

void dbcreate::on_buttonBox_accepted()
{
    if (ui->lineEdit->text().isEmpty()) {
        QMessageBox::warning(this, "Input Error", "The text field cannot be empty.");
        return;
    }

    accept(); // Закрыць дыялогавае акно с прыняціем
}

```

Editactive.cpp

```
#include "dbcreate.h"
```

```
#include "ui_dbcreate.h"
```

```
#include "tablepage.h"
```

```
#include <QMessageBox>
```

```
dbcreate::dbcreate(QWidget parent)
```

```
    : QDialog(parent)
```

```
    , ui(new Ui::dbcreate)
```

```
{
```

```
    ui->setUpUi(this);
```

```
}
```

```
dbcreate::~dbcreate()
```

```
{
```

```
    delete ui;
```

```
}
```

```
QString dbcreate::getDatabaseName() const
```

```
{
```

```
    return ui->lineEdit->text();
```

```
}
```

```
void dbcreate::on_buttonBox_accepted()
```

```
{
```

```
    if (ui->lineEdit->text().isEmpty()) {
```

```
        QMessageBox::warning(this, "Input Error", "The text field cannot be empty.");
```

```
        return;
```

```
    }
```

```
    accept(); // Закрыць дыялогавае акно с прыняццям
```

```
}
```

Editauthomation.cpp

```
#include "editauthomation.h"
```

```
#include "ui_editauthomation.h"
```

```
editAuthomation::editAuthomation(QSqlDatabase database, int assetId, bool tangibleStatus, QWidget parent)
```

```
    : QDialog(parent)
```

```
    , ui(new Ui::editAuthomation)
```

```
    , db(database)
```

```
    , currentAssetId(assetId)
```

```
{
```

```
    ui->setupUi(this);
```

```
    if (!db.isOpen()) {
```

```
        QMessageBox::critical(this, "Database Error", "Failed to open the database: " + db.lastError().text());
```

```
        return;
```

```
    }
```

```
    if (!tangibleStatus) {
```

```
        loadDepreciationCodesForIntangible();
```

```
        if (currentAssetId != -1) {
```

```
            loadDepreciationDataForIntangible(assetId);
```

```
        connect(ui->buttonBox, &QDialogButtonBox::accepted, this, &editAuthomation::saveDepreciationDataForIntangible);
```

```
        connect(ui->buttonBox, &QDialogButtonBox::rejected, this, &editAuthomation::cancelButton_clicked);
```

```
        connect(ui->pushButton_delete, &QPushButton::clicked, this, &editAuthomation::deleteDepreciationForIntangible);
```

```
    } else {
```

```
        loadDepreciationCodesForTangible();
```

```
        if (currentAssetId != -1) {
```

```
            loadDepreciationDataForTangible(assetId);
```

```
        connect(ui->buttonBox, &QDialogButtonBox::accepted, this, &editAuthomation::saveDepreciationDataForTangible);
```

```
        connect(ui->buttonBox, &QDialogButtonBox::rejected, this, &editAuthomation::cancelButton_clicked);
```

```
        connect(ui->pushButton_delete, &QPushButton::clicked, this, &editAuthomation::deleteDepreciationForTangible);
```

```
    }
```

```
    ui->comboBox_expenseType->addItem("Основні засоби", "Інші необоротні активи", "Нематеріальні активи", "Гудвіл", "Інші нематеріальні активи");
```

```
    ui->comboBox_depreciationMethod->addItem("Прямолінійний метод", "Зменшений залишок", "Виробничий метод");
```

```
}
```

```
editAuthomation::~~editAuthomation()
```

```
{
```

```
    delete ui;
```

```
}
```

```
void editAuthomation::loadDepreciationCodesForIntangible()
```

```
{
```

```
    QSqlQuery query(db);
```

```
    query.prepare("SELECT id, asset_id FROM intangible_assets_depreciation");
```

```
    if (!query.exec()) {
```

```
        QMessageBox::critical(this, "Error", "Failed to load asset codes: " + query.lastError().text());
```

```
        return;
```

```
    }
```

```
    while (query.next()) {
```

```
        int id = query.value("id").toInt();
```

```
        QString code = query.value("asset_id").toString();
```

```
        ui->comboBox_assetId->addItem(code, id);
```

```
    }
```

```
    ui->comboBox_assetId->setEnabled(false); // Блокируем возможность изменения кода актива
```

```
}
```

```
void editAuthomation::loadDepreciationDataForIntangible(int assetId)
```

```
{
```

```
    QSqlQuery query(db);
```

```
    query.prepare("SELECT FROM intangible_assets_depreciation WHERE asset_id = ?");
```

```
    query.addBindValue(assetId);
```

```
    if (!query.exec()) {
```

```
        qDebug() << "Error executing query:" << query.lastError().text();
```

```
        QMessageBox::critical(this, "Error", "Failed to load depreciation data: " + query.lastError().text());
```

```
        return;
```

```
    }
```

```
    if (!query.next()) {
```

```

qDebug() << "No data found for asset ID:" << assetId;
QMessageBox::critical(this, "Error", "No data found for asset ID: " + QString::number(assetId));
return;
}

ui->comboBox_expenseType->setCurrentText(query.value("expense_type").toString());
ui->comboBox_depreciationMethod->setCurrentText(query.value("depreciation_method").toString());
ui->lineEdit_salvageValue->setText(query.value("salvage_value").toString());
ui->dateEdit_depreciationTerm->setDate(query.value("depreciation_term").toDate());
int index = ui->comboBox_assetId->findData(assetId);
if (index != -1) {
    ui->comboBox_assetId->setCurrentIndex(index);
}
}

void editAuthomation::saveDepreciationDataForIntangible()
{

    QSqlQuery query(db);

    query.prepare("UPDATE intangible_assets_depreciation SET expense_type = ?, depreciation_method = ?,
salvage_value = ?, depreciation_term = ? WHERE asset_id = ?");

    query.addBindValue(ui->comboBox_expenseType->currentText());
    query.addBindValue(ui->comboBox_depreciationMethod->currentText());
    query.addBindValue(ui->lineEdit_salvageValue->text().toDouble());
    query.addBindValue(ui->dateEdit_depreciationTerm->date());
    query.addBindValue(currentAssetId);

    if (!query.exec()) {
        QMessageBox::critical(this, "Error", "Failed to save depreciation data: " + query.lastError().text());
    } else {
        QMessageBox::information(this, "Success", "Depreciation data saved successfully.");
        accept(); // Закрыть диалоговое окно после сохранения
    }

}

void editAuthomation::deleteDepreciationForIntangible()
{
    QSqlQuery query(db);
    query.prepare("DELETE FROM intangible_assets_depreciation WHERE asset_id = ?");

```

```

query.addBindValue(currentAssetId);

if (!query.exec()) {
    QMessageBox::critical(this, "Error", "Failed to delete depreciation data: " + query.lastError().text());
} else {
    QMessageBox::information(this, "Success", "Depreciation data deleted successfully.");
    // Обновить список данных об амортизации для нематериальных активов
    loadDepreciationCodesForIntangible();
    ui->comboBox_assetId->setCurrentIndex(-1); // Сбросить выбор
}
}

void editAuthomation::loadDepreciationCodesForTangible()
{
    QSqlQuery query(db);
    query.prepare("SELECT id, asset_id FROM tangible_assets_depreciation");

    if (!query.exec()) {
        QMessageBox::critical(this, "Error", "Failed to load asset codes: " + query.lastError().text());
        return;
    }

    while (query.next()) {
        int id = query.value("id").toInt();
        QString code = query.value("asset_id").toString();
        ui->comboBox_assetId->addItem(code, id);
    }

    ui->comboBox_assetId->setEnabled(false); // Блокируем возможность изменения кода актива
}

void editAuthomation::loadDepreciationDataForTangible(int assetId)
{
    QSqlQuery query(db);
    query.prepare("SELECT FROM tangible_assets_depreciation WHERE asset_id = ?");
    query.addBindValue(assetId);

    if (!query.exec()) {

```

```

qDebug() << "Error executing query:" << query.lastError().text();
QMessageBox::critical(this, "Error", "Failed to load depreciation data: " + query.lastError().text());
return;
}

if (!query.next()) {
    qDebug() << "No data found for asset ID:" << assetId;
    QMessageBox::critical(this, "Error", "No data found for asset ID: " + QString::number(assetId));
    return;
}

ui->comboBox_expenseType->setCurrentText(query.value("expense_type").toString());
ui->comboBox_depreciationMethod->setCurrentText(query.value("depreciation_method").toString());
ui->lineEdit_salvageValue->setText(query.value("salvage_value").toString());
ui->dateEdit_depreciationTerm->setDate(query.value("depreciation_term").toDate());

int index = ui->comboBox_assetId->findData(assetId);
if (index != -1) {
    ui->comboBox_assetId->setCurrentIndex(index);
}
}

void editAuthomation::saveDepreciationDataForTangible()
{

    QSqlQuery query(db);

    query.prepare("UPDATE tangible_assets_depreciation SET expense_type = ?, depreciation_method = ?,
    salvage_value = ?, depreciation_term = ? WHERE asset_id = ?");

    query.addBindValue(ui->comboBox_expenseType->currentText());
    query.addBindValue(ui->comboBox_depreciationMethod->currentText());
    query.addBindValue(ui->lineEdit_salvageValue->text().toDouble());
    query.addBindValue(ui->dateEdit_depreciationTerm->date());
    query.addBindValue(currentAssetId);

    if (!query.exec()) {
        QMessageBox::critical(this, "Error", "Failed to save depreciation data: " + query.lastError().text());
    } else {
        QMessageBox::information(this, "Success", "Depreciation data saved successfully.");
        accept(); // Закрывать диалоговое окно после сохранения
    }
}

```

```

    }
}

void editAuthomation::deleteDepreciationForTangible()
{
    QSqlQuery query(db);
    query.prepare("DELETE FROM tangible_assets_depreciation WHERE asset_id = ?");
    query.addBindValue(currentAssetId);

    if (!query.exec()) {
        QMessageBox::critical(this, "Error", "Failed to delete depreciation data: " + query.lastError().text());
    } else {
        QMessageBox::information(this, "Success", "Depreciation data deleted successfully.");
        // Обновить список данных об амортизации для материальных активов
        loadDepreciationCodesForTangible();
        ui->comboBox_assetId->setCurrentIndex(-1); // Сбросить выбор
    }
}

void editAuthomation::saveButton_clicked()
{
    if (!tangibleStatus) {
        saveDepreciationDataForIntangible();
    } else {
        saveDepreciationDataForTangible();
    }
}

void editAuthomation::deleteButton_clicked()
{
    if (!tangibleStatus) {
        deleteDepreciationForIntangible();
    } else {
        deleteDepreciationForTangible();
    }
}

void editAuthomation::comboBox_code_currentIndexChanged(int index)
{
    int assetId = ui->comboBox_assetId->itemData(index).toInt();
    if (assetId != -1 && assetId != currentAssetId) {

```

```
if (!tangibleStatus) {  
    loadDepreciationDataForIntangible(assetId);  
} else {  
    loadDepreciationDataForTangible(assetId);  
}  
}  
}  
  
void editAuthomation::cancelButton_clicked()  
{  
    reject(); // Закрыть диалоговое окно без сохранения  
}
```

Generatereport.cpp

```
#include "generatereport.h"
```

```
#include "ui_generatereport.h"
```

```
#include <QMessageBox>
```

```
#include <QSqlQuery>
```

```
#include <QSqlError>
```

```
#include <QPdfWriter>
```

```
#include <QPainter>
```

```
#include <QFileDialog>
```

```
#include <QDate>
```

```
#include <QFontMetrics>
```

```
generateReport::generateReport(QString reportType, QSqlDatabase database, QWidget parent)
```

```
    : QDialog(parent)
```

```
    , ui(new Ui::generateReport)
```

```
    , db(database)
```

```
{
```

```
    ui->setupUi(this);
```

```
    if (reportType == " ")
```

```
    {
```

```
        QMessageBox::critical(this, "Report Error", "Failed to open the report: ");
```

```
        return;
```

```
    }
```

```
    else
```

```
    {
```

```
        if (reportType == "assets_report")
```

```
        {
```

```
            ui->lineEdit_chosed_report->setText("Звіт з активів");
```

```
            setFieldsForAssetsReport();
```

```
            connect(ui->buttonBox, &QDialogButtonBox::accepted, this, &generateReport::generateAssetsReport);
```

```
        }
```

```
        else if (reportType == "depreciation_report")
```

```
        {
```

```
            ui->lineEdit_chosed_report->setText("Звіт з амортизації");
```

```
            setFieldsForDepreciationReport();
```

```
            connect(ui->buttonBox, &QDialogButtonBox::accepted, this, &generateReport::generateDepreciationReport);
```

```
        }
```

```
        else if (reportType == "move_report")
```

```
        {
```

```

        ui->lineEdit_chosed_report->setText("Звіт руху активів");
        setFieldsForMoveReport();
        connect(ui->buttonBox, &QDialogButtonBox::accepted, this, &generateReport::generateMoveReport);
    }
    else if (reportType == "inventory_report")
    {
        ui->lineEdit_chosed_report->setText("Інвентарна картка");
        setFieldsForInventoryReport();
        connect(ui->buttonBox, &QDialogButtonBox::accepted, this, &generateReport::generateInventoryReport);
    }
    else if (reportType == "category_report")
    {
        ui->lineEdit_chosed_report->setText("Звіт з категорій активів");
        setFieldsForCategoryReport();
        connect(ui->buttonBox, &QDialogButtonBox::accepted, this, &generateReport::generateCategoryReport);
    }

    ui->comboBox_active_type->addItem({"основні засоби", "інші необоротні активи", "нематеріальні активи",
    "інші нематеріальні активи"});

    ui->comboBox_operation_type->addItem({"придбання", "вибуття", "зміна вартості", "амортизація",
    "ремонт/обслуговування", "модернізація", "оренда", "повернення з оренди", "зміна місцезнаходження",
    "зміна відповідальної особи", "злиття", "поділ"});

    ui->comboBox_depreciation_method->addItem({"Прямолінійний метод", "Зменшений залишок",
    "Виробничий метод"});

    connect(ui->comboBox_active_type, &QComboBox::currentTextChanged, this,
    &generateReport::on_activeTypeChanged);

    connect(ui->comboBox_active_code, &QComboBox::currentTextChanged, this,
    &generateReport::on_activeCodeChanged);
    }
    }

generateReport::~generateReport()
{
    delete ui;
}

void generateReport::on_activeTypeChanged(const QString &type)
{
    populateActiveCodes(type);
}

```

```
}
```

```
void generateReport::on_activeCodeChanged(const QString &code)
```

```
{
```

```
    QString type = ui->comboBox_active_type->currentText();
```

```
    populateActiveCategory(code, type);
```

```
    populateDateRange(code, type);
```

```
}
```

```
void generateReport::populateActiveCodes(const QString &type)
```

```
{
```

```
    ui->comboBox_active_code->clear();
```

```
    QString queryStr;
```

```
    if (type == "нематеріальні активи" || type == "інші нематеріальні активи")
```

```
    {
```

```
        queryStr = "SELECT code FROM intangible_assets";
```

```
    }
```

```
    else if (type == "основні засоби")
```

```
    {
```

```
        queryStr = "SELECT code FROM tangible_assets";
```

```
    }
```

```
    else
```

```
    {
```

```
        return; // Для других типов активов не обновляем список кодов
```

```
    }
```

```
    QSqlQuery query(db);
```

```
    if (!query.exec(queryStr))
```

```
    {
```

```
        QMessageBox::critical(this, "Database Error", "Failed to retrieve asset codes: " + query.lastError().text());
```

```
        return;
```

```
    }
```

```
    while (query.next())
```

```
    {
```

```
        ui->comboBox_active_code->addItem(query.value(0).toString());
```

```
    }
```

```
}
```

```

void generateReport::populateActiveCategory(const QString &code, const QString &type)
{
    QString queryStr;
    if (type == "нематеріальні активи" || type == "інші нематеріальні активи")
    {
        queryStr = "SELECT account FROM intangible_assets WHERE code = ?";
    }
    else if (type == "основні засоби")
    {
        queryStr = "SELECT account FROM tangible_assets WHERE code = ?";
    }
    else
    {
        return; // Для других типів активів не обновляем категорию
    }

    QSqlQuery query(db);
    query.prepare(queryStr);
    query.addBindValue(code);

    if (!query.exec())
    {
        QMessageBox::critical(this, "Database Error", "Failed to retrieve asset account: " + query.lastError().text());
        return;
    }

    if (query.next())
    {
        ui->comboBox_active_category->clear();
        ui->comboBox_active_category->addItem(query.value(0).toString());
    }
}

void generateReport::populateDateRange(const QString &code, const QString &type)
{
    QString queryStr;
    if (type == "нематеріальні активи" || type == "інші нематеріальні активи")
    {

```

```

        queryStr = "SELECT MIN(entry_date) AS start_date, MAX(entry_date) AS end_date FROM intangible_assets
WHERE code = ?";

    }

    else if (type == "основні засоби")
    {
        queryStr = "SELECT MIN(entry_date) AS start_date, MAX(entry_date) AS end_date FROM tangible_assets
WHERE code = ?";

    }

    else
    {
        return; // Для других типов активов не обновляем диапазон дат
    }

    QSqlQuery query(db);
    query.prepare(queryStr);
    query.addBindValue(code);

    if (!query.exec())
    {
        QMessageBox::critical(this, "Database Error", "Failed to retrieve date range: " + query.lastError().text());
        return;
    }

    if (query.next())
    {
        QDate startDate = query.value("start_date").toDate();
        QDate endDate = query.value("end_date").toDate();
        ui->dateEdit_from->setDate(startDate);
        ui->dateEdit_to->setDate(endDate);
    }
}

void generateReport::setFieldsForAssetsReport()
{
    // Активируем только те поля, которые нужны для отчета об активах
    ui->comboBox_operation_type->setEnabled(false);
    ui->dateEdit_from->setEnabled(true);
    ui->dateEdit_to->setEnabled(true);
    ui->comboBox_depreciation_method->setEnabled(false);
    // и т.д.

```

```
}
```

```
void generateReport::setFieldsForDepreciationReport()
```

```
{
```

```
// Активируем только те поля, которые нужны для отчета об амортизации
```

```
ui->comboBox_operation_type->setEnabled(false);
```

```
ui->dateEdit_from->setEnabled(true);
```

```
ui->dateEdit_to->setEnabled(true);
```

```
ui->comboBox_depreciation_method->setEnabled(true);
```

```
// и т.д.
```

```
}
```

```
void generateReport::setFieldsForMoveReport()
```

```
{
```

```
// Активируем только те поля, которые нужны для отчета о движении активов
```

```
ui->comboBox_operation_type->setEnabled(true);
```

```
ui->dateEdit_from->setEnabled(true);
```

```
ui->dateEdit_to->setEnabled(true);
```

```
ui->comboBox_depreciation_method->setEnabled(false);
```

```
// и т.д.
```

```
}
```

```
void generateReport::setFieldsForInventoryReport()
```

```
{
```

```
// Активируем только те поля, которые нужны для отчета по инвентарному номеру
```

```
ui->comboBox_operation_type->setEnabled(false);
```

```
ui->dateEdit_from->setEnabled(false);
```

```
ui->dateEdit_to->setEnabled(false);
```

```
ui->comboBox_depreciation_method->setEnabled(false);
```

```
// и т.д.
```

```
}
```

```
void generateReport::setFieldsForCategoryReport()
```

```
{
```

```
// Активируем только те поля, которые нужны для отчета по категориям активов
```

```
ui->comboBox_operation_type->setEnabled(false);
```

```
ui->dateEdit_from->setEnabled(false);
```

```
ui->dateEdit_to->setEnabled(false);
```

```
ui->comboBox_depreciation_method->setEnabled(false);
```

```

// и т.д.
}

void generateReport::generateAssetsReport()
{
    QString fileName = QFileDialog::getSaveFileName(this, tr("Зберегти звіт"), "", tr("PDF Files (.pdf)"));
    if (fileName.isEmpty()) {
        return; // User canceled saving
    }

    // Prepare the SQL query
    QSqlQuery query(db);
    QString selectedCode = ui->comboBox_active_code->currentText();
    QString selectedCategory = ui->comboBox_active_category->currentText();
    QString startDate = ui->dateEdit_from->date().toString("yyyy-MM-dd");
    QString endDate = ui->dateEdit_to->date().toString("yyyy-MM-dd");

    QString queryStr = "SELECT FROM intangible_assets WHERE code = ? AND account = ? AND entry_date
    BETWEEN ? AND ?";

    query.prepare(queryStr);
    query.addBindValue(selectedCode);
    query.addBindValue(selectedCategory);
    query.addBindValue(startDate);
    query.addBindValue(endDate);

    // Check for query execution error
    if (!query.exec()) {
        QMessageBox::critical(this, "Помилка", "Не вдалося виконати запит: " + query.lastError().text());
        return;
    }

    // Create PDF writer
    QPdfWriter writer(fileName);

    // Set page size based on Qt version (adjust as needed)
    writer.setPageSize(QPageSize(QPageSize::A4)); // Use setPageSize for Qt 5.6+

    // Create painter object for drawing on the PDF
    QPainter painter(&writer);

```

```

painter.setPen(Qt::black);
painter.setFont(QFont("Arial", 12));

// Define margins for better readability
const int marginX = 250;
const int marginY = 250;
int y = marginY; // Starting position with margin

// Draw report header with dynamic centering
int textWidth = painter.fontMetrics().horizontalAdvance("Звіт про активи\n");
painter.drawText((writer.width() - textWidth) / 2, y, "Звіт про активи\n");
y += 520;


// Draw table header (optional)
QStringList headers = {"Код", "Назва", "Повна назва", "Тип актива", "Рахунок", "Початкова вартість",
"Інвентарний номер", "Шифр", "Одиниця виміру", "Дата вводу", "Тип активності"};
QVector<int> columnWidths = {400, 1050, 2000, 2000, 2000, 1050, 3050, 2000, 3000, 3400, 3000};
int tableX = marginX;
int tableY = y;
for (int i = 0; i < headers.size(); ++i) {
    painter.drawText(tableX, tableY, headers[i]);
    tableX += columnWidths[i];
}
tableY += 520;


// Loop through query results and draw asset information
while (query.next()) {
    tableX = marginX;
    for (int i = 0; i < headers.size(); ++i) {
        QString data = query.value(i).toString();
        painter.drawText(tableX, tableY, data);
        tableX += columnWidths[i];
    }
    tableY += 520;
}

```

```

        textWidth = painter.fontMetrics().horizontalAdvance("Дата: " +
QDate::currentDate().toString("dd.MM.yyyy") + "\n");

        painter.drawText(((writer.width() - textWidth) / 2)+3500, y+4000, "Дата: " +
QDate::currentDate().toString("dd.MM.yyyy") + "\n");

        y += 620;

// Draw signature line (optional)
textWidth = painter.fontMetrics().horizontalAdvance("Підпис: _____\n");
painter.drawText(((writer.width() - textWidth) / 2)+3500, y+4000, "Підпис: _____\n");
y += 620; // Move down for table data

// Show success message after saving the report
QMessageBox::information(this, "Успішно", "Звіт про активи збережено у файлі " + fileName);
}

void generateReport::generateMoveReport()
{
    // Get filename for saving the report
    QString fileName = QFileDialog::getSaveFileName(this, tr("Зберегти звіт"), "", tr("PDF Files (.pdf)"));
    if (fileName.isEmpty()) {
        return; // User canceled saving
    }

    // Prepare the SQL query
    QSqlQuery query(db);
    QString selectedCode = ui->comboBox_active_code->currentText();
    QString selectedCategory = ui->comboBox_active_category->currentText();
    QString startDate = ui->dateEdit_from->date().toString("yyyy-MM-dd");
    QString endDate = ui->dateEdit_to->date().toString("yyyy-MM-dd");

    QString tableName;
    if (selectedCategory == "нематеріальні активи" || selectedCategory == "інші нематеріальні активи") {
        tableName = "intangible_assets";
    } else if (selectedCategory == "основні засоби") {
        tableName = "tangible_assets";
    } else {
        QMessageBox::critical(this, "Error", "Invalid asset category selected.");
        return;
    }
}

QString queryStr = "SELECT FROM " + tableName + " WHERE code = ? AND account = ? AND entry_date

```

```

BETWEEN ? AND ?";

query.prepare(queryStr);
query.addBindValue(selectedCode);
query.addBindValue(selectedCategory);
query.addBindValue(startDate);
query.addBindValue(endDate);


// Check for query execution error
if (!query.exec()) {
    QMessageBox::critical(this, "Помилка", "Не вдалося виконати запит: " + query.lastError().text());
    return;
}


// Create PDF writer
QPdfWriter writer(fileName);


// Set page size based on Qt version (adjust as needed)
writer.setPageSize(QPageSize(QPageSize::A4)); // Use setPageSize for Qt 5.6+


// Create painter object for drawing on the PDF
QPainter painter(&writer);
painter.setPen(Qt::black);
painter.setFont(QFont("Arial", 12));


// Define margins for better readability
const int marginX = 50;
const int marginY = 50;
int y = marginY; // Starting position with margin


// Draw report header with dynamic centering
int textWidth = painter.fontMetrics().horizontalAdvance("Звіт про переміщення активів\n");
painter.drawText((writer.width() - textWidth) / 2, y, "Звіт про переміщення активів\n");
y += 20;


// Draw table header
painter.drawText(marginX, y, "Назва активу");
painter.drawText(marginX + 200, y, "Код активу");
// Add more headers as needed

```

```

// Move to next line
y += 20;

// Loop through query results and draw asset information
while (query.next()) {

    QString assetName = query.value("name").toString();
    QString assetCode = query.value("code").toString();
    // Retrieve more information as needed from the query result

    // Draw asset information
    painter.drawText(marginX, y, assetName);
    painter.drawText(marginX + 200, y, assetCode);
    // Draw more information as needed

    // Move to next line
    y += 20;
    textWidth = painter.fontMetrics().horizontalAdvance("Місце знаходження об'єкта після переміщення
    _____\n");
    painter.drawText(((writer.width() - textWidth) / 2) + 3500, y + 4000, "Місце знаходження об'єкта після
    переміщення _____\n");
    textWidth = painter.fontMetrics().horizontalAdvance("Передається актив код активу
    _____\n");
    painter.drawText(((writer.width() - textWidth) / 2) + 3500, y + 4000, "Передається актив код активу
    _____\n");
    textWidth = painter.fontMetrics().horizontalAdvance("Одиниця міри _____\n");
    painter.drawText(((writer.width() - textWidth) / 2) + 3500, y + 4000, "Одиниця міри
    _____\n");

    // Add footer with date and signature
    textWidth = painter.fontMetrics().horizontalAdvance("Дата: " + QDate::currentDate().toString("dd.MM.yyyy") +
    "\n");
    painter.drawText(((writer.width() - textWidth) / 2) + 3500, y + 4000, "Дата: " +
    QDate::currentDate().toString("dd.MM.yyyy") + "\n");
    y += 20;
    textWidth = painter.fontMetrics().horizontalAdvance("Підпис: _____\n");
    painter.drawText(((writer.width() - textWidth) / 2) + 3500, y + 4000, "Підпис: _____\n");

    // Finalize the report
    painter.end();
}

// Show success message after saving the report

```

```
    QMessageBox::information(this, "Успішно", "Звіт про переміщення активів збережено у файлі " + fileName);  
}
```

```
void generateReport::generateInventoryReport()  
{
```

```
    QString fileName = QFileDialog::getSaveFileName(this, tr("Зберегти звіт"), "", tr("PDF Files (.pdf)"));  
    if (fileName.isEmpty()) {  
        return; // Пользователь отменил сохранение  
    }
```

```
    QSqlQuery query(db);  
    QString assetID = ui->comboBox_active_code->currentText(); // Получаем ID актива из интерфейса  
    QString queryStr = "SELECT FROM intangible_assets WHERE id = ?";  
    query.prepare(queryStr);  
    query.addBindValue(assetID);
```

```
    if (!query.exec()) {  
        QMessageBox::critical(this, "Помилка", "Не вдалось виконати запит: " + query.lastError().text());  
        return;  
    }
```

```
    // Создаем PDF-писателя  
    QPdfWriter writer(fileName);
```

```
    // Устанавливаем размер страницы (можно настроить по желанию)  
    writer.setPageSize(QPageSize(QPageSize::A4));
```

```
    // Создаем рисовальщика для работы с PDF  
    QPainter painter(&writer);  
    painter.setPen(Qt::black);  
    painter.setFont(QFont("Arial", 12));
```

```
    const int marginX = 50;  
    const int marginY = 50;  
    int y = marginY; // Начальная позиция с отступом
```

```
    // Рисуем заголовок отчета
```

```

int textWidth = painter.fontMetrics().horizontalAdvance("Інвентарна карточка актива\n");
painter.drawText((writer.width() - textWidth) / 2, y, "Інвентарна карточка актива\n");
y += 620;

// Получаем данные об активе из запроса
if (query.next()) {
    QString assetName = query.value("name").toString();
    QString assetCode = query.value("code").toString();
    QString assetType = query.value("asset_type").toString();
    QString entryDate = query.value("entry_date").toDate().toString("dd.MM.yyyy");
    QString initialCost = query.value("initial_cost").toString();
    QString inventoryNumber = query.value("inventory_number").toString();
    QString account = query.value("account").toString();
    QString unit = query.value("unit").toString();

    // Выводим информацию об активе на карточку
    painter.drawText(marginX, y, "Назва актива: " + assetName);
    y += 620;
    painter.drawText(marginX, y, "Код актива: " + assetCode);
    y += 620;
    painter.drawText(marginX, y, "Тип актива: " + assetType);
    y += 620;
    painter.drawText(marginX, y, "Дата вступу: " + entryDate);
    y += 620;
    painter.drawText(marginX, y, "Початкова вартість: " + initialCost);
    y += 620;
    painter.drawText(marginX, y, "Інвентарний номер: " + inventoryNumber);
    y += 620;
    painter.drawText(marginX, y, "Рахунок: " + account);
    y += 620;
    painter.drawText(marginX, y, "Одиниця міри: " + unit);
    y += 620;

} else {
    QMessageBox::warning(this, "Попередження", "Актив с ID " + assetID + " не знайдено.");
    return;
}

```

```

// Заголовок таблицы
int tableHeaderY = y + 620;

painter.drawText(marginX, tableHeaderY, "Додаткова інформація об активі:");
y += 640;

// Заголовки столбцов таблицы
painter.drawText(marginX, y + 1000, "Дата");
painter.drawText(marginX + 500 + 3000, y + 1000, "Тип операції");
painter.drawText(marginX + 750 + 5000, y + 1000, "Опис");

y += 620;

painter.drawText(marginX, y + 1000, "01.01.2023");
painter.drawText(marginX + 500 + 3000, y + 1000, "Покупка");
painter.drawText(marginX + 750 + 5000, y + 1000, "Куплений комп'ютер для відділу бухгалтерії");
// Добавьте данные для других строк, если это необходимо
// Добавьте данные для других строк, если это необходимо

textWidth = painter.fontMetrics().horizontalAdvance("Дата: " + QDate::currentDate().toString("dd.MM.yyyy") +
"\n");

painter.drawText(((writer.width() - textWidth) / 2) + 3500, y + 4000, "Дата: " +
QDate::currentDate().toString("dd.MM.yyyy") + "\n");

y += 620;

// Draw signature line (optional)
textWidth = painter.fontMetrics().horizontalAdvance("Підпис: _____ \n");
painter.drawText(((writer.width() - textWidth) / 2) + 3500, y + 4000, "Підпис: _____ \n");
y += 620; // Move down for table data

// Завершаем отчет
painter.end();

// Показываем сообщение об успешном сохранении отчета
QMessageBox::information(this, "Успішно", "Інвентарна карточка актива збережена в файлі: " + fileName);

}

void generateReport::generateCategoryReport()

```

```

{
    // Get filename for saving the report
    QString fileName = QFileDialog::getSaveFileName(this, tr("Зберегти звіт"), "", tr("PDF Files (.pdf)"));
    if (fileName.isEmpty()) {
        return; // User canceled saving
    }

    // Prepare the SQL query
    QSqlQuery query(db);
    QString selectedCategory = ui->comboBox_active_category->currentText();
    QString startDate = ui->dateEdit_from->date().toString("yyyy-MM-dd");
    QString endDate = ui->dateEdit_to->date().toString("yyyy-MM-dd");

    QString tableName;
    if (selectedCategory == "нематеріальні активи" || selectedCategory == "інші нематеріальні активи") {
        tableName = "intangible_assets";
    } else if (selectedCategory == "основні засоби") {
        tableName = "tangible_assets";
    } else {
        QMessageBox::critical(this, "Error", "Invalid asset category selected.");
        return;
    }

    QString queryStr = "SELECT FROM " + tableName + " WHERE entry_date BETWEEN ? AND ?";
    query.prepare(queryStr);
    query.addBindValue(startDate);
    query.addBindValue(endDate);

    // Check for query execution error
    if (!query.exec()) {
        QMessageBox::critical(this, "Помилка", "Не вдалося виконати запит: " + query.lastError().text());
        return;
    }

    // Create PDF writer
    QPdfWriter writer(fileName);

    // Set page size based on Qt version (adjust as needed)
    writer.setPageSize(QPageSize(QPageSize::A4)); // Use setPageSize for Qt 5.6+

```

```

// Create painter object for drawing on the PDF
QPainter painter(&writer);
painter.setPen(Qt::black);
painter.setFont(QFont("Arial", 12));

// Define margins for better readability
const int marginX = 50;
const int marginY = 50;
int y = marginY; // Starting position with margin
int textWidth = painter.fontMetrics().horizontalAdvance("Обраний категорій активу\n");
    painter.drawText((writer.width() - textWidth) / 2, y, "Обраний категорій активу\n");
    y += 20;

// Draw report header with dynamic centering
int textWidth = painter.fontMetrics().horizontalAdvance("Звіт по категорії активів\n");
painter.drawText((writer.width() - textWidth) / 2, y, "Звіт по категорії активів\n");
y += 20;

// Draw table header
QStringList headers = {"Назва активу", "Код активу", "Тип актива", "Рахунок", "Початкова вартість",
"Інвентарний номер", "Шифр", "Одиниця виміру", "Дата вводу", "Тип активності"};
 QVector<int> columnWidths = {300, 150, 200, 150, 150, 150, 150, 150, 150, 150};
int tableX = marginX;
int tableY = y;
for (int i = 0; i < headers.size(); ++i) {
    painter.drawText(tableX, tableY, headers[i]);
    tableX += columnWidths[i];
}
tableY += 20;

// Loop through query results and draw asset information
while (query.next()) {
    tableX = marginX;
    for (int i = 0; i < headers.size(); ++i) {
        QString data = query.value(i).toString();
        painter.drawText(tableX, tableY, data);
        tableX += columnWidths[i];
    }
}

```

```

        tableY += 20;
    }

    // Add footer with date and signature
    textWidth = painter.fontMetrics().horizontalAdvance("Дата: " + QDate::currentDate().toString("dd.MM.yyyy") +
"\n");
    painter.drawText(((writer.width() - textWidth) / 2) + 3500, tableY + 4000, "Дата: " +
QDate::currentDate().toString("dd.MM.yyyy") + "\n");
    tableY += 20;
    textWidth = painter.fontMetrics().horizontalAdvance("Підпис: _____\n");
    painter.drawText(((writer.width() - textWidth) / 2) + 3500, tableY + 4000, "Підпис: _____\n");

    // Finalize the report
    painter.end();

    // Show success message after saving the report
    QMessageBox::information(this, "Успішно", "Звіт по категорії активів збережено у файлі " + fileName);
}

void generateReport::generateDepreciationReport()
{

    QString fileName = QFileDialog::getSaveFileName(this, tr("Зберегти звіт"), "", tr("PDF Files (.pdf)"));
    if (fileName.isEmpty()) {
        return; // Пользователь отменил сохранение
    }

    // Підготавлюємо SQL-запрос для получения информации об амортизации
    QSqlQuery query(db);
    QString selectedCode = ui->comboBox_active_code->currentText();
    QString selectedCategory = ui->comboBox_active_type->currentText();
    QString depreciationMethod = ui->comboBox_depreciation_method->currentText();
    QString startDate = ui->dateEdit_from->date().toString("yyyy-MM-dd");
    QString endDate = ui->dateEdit_to->date().toString("yyyy-MM-dd");

    QString queryStr;
    if (selectedCategory == "нематеріальні активи" || selectedCategory == "інші нематеріальні активи") {
        queryStr = "SELECT FROM intangible_assets WHERE code = ? AND account = ? AND entry_date BETWEEN
? AND ?";
    }

```

```

    } else if (selectedCategory == "основні засоби") {
        queryStr = "SELECT FROM tangible_assets WHERE code = ? AND account = ? AND entry_date BETWEEN ?
AND ?";
    } else {
        QMessageBox::critical(this, "Error", "Invalid asset category selected.");
        return;
    }

    query.prepare(queryStr);
    query.addBindValue(selectedCode);
    query.addBindValue(selectedCategory);
    query.addBindValue(startDate);
    query.addBindValue(endDate);

    // Проверяем ошибки выполнения запроса
    if (!query.exec()) {
        QMessageBox::critical(this, "Помилка", "Не вдалося виконати запит: " + query.lastError().text());
        return;
    }

    // Создаем объект для записи PDF
    QPdfWriter writer(fileName);

    // Устанавливаем размер страницы
    writer.setPageSize(QPageSize(QPageSize::A4));

    // Создаем объект рисования на PDF
    QPainter painter(&writer);
    painter.setPen(Qt::black);
    painter.setFont(QFont("Arial", 12));

    // Определяем отступы
    const int marginX = 1050;
    const int marginY = 1050;
    int y = marginY;

    // Рисуем заголовок отчета
    int textWidth = painter.fontMetrics().horizontalAdvance("Звіт про амортизацію\n");
    painter.drawText((writer.width() - textWidth) / 2, y, "Звіт про амортизацію\n");

```

```
y += 820;
```

```
// Рисуем заголовок таблицы
```

```
QStringList headers = {"Назва", "Код", "Тип активу", "Дата вступу", "Початкова вартість", "Остаточна вартість", "Строк служби", "Метод амортизації", "Величина амортизації"};
```

```
QVector<int> columnWidths = {500, 500, 500, 500, 500, 800, 900, 600, 800};
```

```
int tableX = marginX;
```

```
int tableY = y;
```

```
for (int i = 0; i < headers.size(); ++i) {
```

```
    painter.drawText(tableX, tableY, headers[i]);
```

```
    tableX += columnWidths[i];
```

```
}
```

```
tableY += 720;
```

```
// Цикл по результатам запроса и рисование информации об амортизации
```

```
while (query.next()) {
```

```
    QString assetName = query.value("name").toString();
```

```
    QString assetCode = query.value("code").toString();
```

```
    QString assetType = query.value("asset_type").toString();
```

```
    QString entryDate = query.value("entry_date").toDate().toString("dd.MM.yyyy");
```

```
    QString initialCost = query.value("initial_cost").toString();
```

```
    QString salvageValue = query.value("salvage_value").toString();
```

```
    QString usefulLife = query.value("useful_life").toString();
```

```
    QString depreciationMethod = ui->comboBox_depreciation_method->currentText();
```

```
// Рассчитываем сумму амортизации в зависимости от выбранного метода
```

```
double depreciationAmount = calculateDepreciation(query.value("id").toInt(), selectedCategory, depreciationMethod);
```

```
// Рисуем информацию об активе и амортизации
```

```
tableX = marginX;
```

```
painter.drawText(tableX, tableY, assetName);
```

```
tableX += columnWidths[0];
```

```
painter.drawText(tableX, tableY, assetCode);
```

```
tableX += columnWidths[1];
```

```
painter.drawText(tableX, tableY, assetType);
```

```
tableX += columnWidths[2];
```

```
painter.drawText(tableX, tableY, entryDate);
```

```
tableX += columnWidths[3];
```

```
painter.drawText(tableX, tableY, initialCost);
```

```

tableX += columnWidths[4];
painter.drawText(tableX, tableY, salvageValue);
tableX += columnWidths[5];
painter.drawText(tableX, tableY, usefulLife);
tableX += columnWidths[6];
painter.drawText(tableX, tableY, depreciationMethod);
tableX += columnWidths[7];
painter.drawText(tableX, tableY, QString::number(depreciationAmount));
tableY += 720;
}

// Завершаем рисование
painter.end();

// Сообщаем пользователю, что отчет сгенерирован
QMessageBox::information(this, "Успішно", "Звіт про амортизацію збережено: " + fileName);

}

```

```

double generateReport::calculateDepreciation(int assetId, const QString& category, const QString&
depreciationMethod)
{
    // Инициализация суммы амортизации
    double depreciationAmount = 0.0;

    // Определение расчета амортизации на основе выбранного метода
    if (depreciationMethod == "Прямолінійний метод") {
        // Расчет амортизации с использованием прямолинейного метода
        QSqlQuery straightLineQuery(db);
        QString queryStr;
        if (category == "нематеріальні активи" || category == "інші нематеріальні активи") {
            queryStr = "SELECT initial_cost, salvage_value, useful_life FROM intangible_assets WHERE id = ?";
        } else if (category == "основні засоби") {
            queryStr = "SELECT initial_cost, salvage_value, useful_life FROM tangible_assets WHERE id = ?";
        } else {
            QMessageBox::critical(this, "Error", "Invalid asset category selected.");
            return 0.0;
        }
    }
}

```

```

straightLineQuery.prepare(queryStr);
straightLineQuery.addBindValue(assetId);
if (!straightLineQuery.exec()) {
    QMessageBox::critical(this, "Error", "Failed to execute query: " + straightLineQuery.lastError().text());
    return 0.0;
}

if (straightLineQuery.next()) {
    double initialCost = straightLineQuery.value("initial_cost").toDouble();
    double salvageValue = straightLineQuery.value("salvage_value").toDouble();
    int usefulLife = straightLineQuery.value("useful_life").toInt();

    depreciationAmount = (initialCost - salvageValue) / usefulLife;
}
} else if (depreciationMethod == "Виробничий метод") {
    // Расчет амортизации с использованием производного метода
    QSqlQuery unitsOfProductionQuery(db);
    QString queryStr;

    if (category == "нематеріальні активи" || category == "інші нематеріальні активи") {
        queryStr = "SELECT initial_cost, salvage_value, useful_life FROM intangible_assets WHERE id = ?";
    } else if (category == "основні засоби") {
        queryStr = "SELECT initial_cost, salvage_value, useful_life FROM tangible_assets WHERE id = ?";
    } else {
        QMessageBox::critical(this, "Error", "Invalid asset category selected.");
        return 0.0;
    }

    unitsOfProductionQuery.prepare(queryStr);
    unitsOfProductionQuery.addBindValue(assetId);
    if (!unitsOfProductionQuery.exec()) {
        QMessageBox::critical(this, "Error", "Failed to execute query: " + unitsOfProductionQuery.lastError().text());
        return 0.0;
    }

    if (unitsOfProductionQuery.next()) {
        double initialCost = unitsOfProductionQuery.value("initial_cost").toDouble();
        double salvageValue = unitsOfProductionQuery.value("salvage_value").toDouble();

```

```

int usefulLife = unitsOfProductionQuery.value("useful_life").toInt();

// Предположим, что амортизация рассчитывается на основе времени использования актива
// и что стоимость актива уменьшается пропорционально с увеличением времени
if (usefulLife > 0) {
    depreciationAmount = (initialCost - salvageValue) / usefulLife;
} else {
    depreciationAmount = 0.0;
}
}

} else if (depreciationMethod == "Зменшений залишок") {
    // Расчет амортизации с использованием метода зменшеного залишку
    QSqlQuery diminishingBalanceQuery(db);
    QString queryStr;

    if (category == "нематеріальні активи" || category == "інші нематеріальні активи") {
        queryStr = "SELECT initial_cost, salvage_value, useful_life FROM intangible_assets WHERE id = ?";
    } else if (category == "основні засоби") {
        queryStr = "SELECT initial_cost, salvage_value, useful_life FROM tangible_assets WHERE id = ?";
    } else {
        QMessageBox::critical(this, "Error", "Invalid asset category selected.");
        return 0.0;
    }
}

diminishingBalanceQuery.prepare(queryStr);
diminishingBalanceQuery.addBindValue(assetId);
if (!diminishingBalanceQuery.exec()) {
    QMessageBox::critical(this, "Error", "Failed to execute query: " + diminishingBalanceQuery.lastError().text());
    return 0.0;
}

if (diminishingBalanceQuery.next()) {
    double initialCost = diminishingBalanceQuery.value("initial_cost").toDouble();
    double salvageValue = diminishingBalanceQuery.value("salvage_value").toDouble();
    int usefulLife = diminishingBalanceQuery.value("useful_life").toInt();

    // Рассчитываем амортизацию на основе метода зменшеного залишку
    double remainingValue = initialCost;
    for (int i = 0; i < usefulLife; ++i) {

```

```
// Уменьшаем остаточную стоимость актива каждый год
remainingValue -= (initialCost - salvageValue) / usefulLife;
}

// Амортизация - это разница между начальной стоимостью и оставшейся стоимостью
depreciationAmount = initialCost - remainingValue;
}

} else {
    QMessageBox::critical(this, "Error", "Invalid depreciation method selected.");
    return 0.0;
}

return depreciationAmount;
```

Main.cpp

```
#include "mainwindow.h"
```

```
#include <QApplication>
```

```
#include <QFile>
```

```
int main(int argc, char argv[])
```

```
{
```

```
    QApplication a(argc, argv);
```

```
    QFile stylesheet(":/Diffnes.qss");
```

```
    stylesheet.open(QFile::ReadOnly);
```

```
    QString styleSheet = QLatin1String(stylesheet.readAll());
```

```
    a.setStyleSheet(styleSheet);
```

```
    MainWindow w;
```

```
    w.show();
```

```
    return a.exec();
```

```
}
```

```

MainWindow.cpp

#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "nonematerialactives.h"
#include "generatereport.h"
#include "dbcreate.h"
#include "tablepage.h"
#include "editactive.h"
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlError>
#include <QMessageBox>
#include <QPixmap>
#include <QFileDialog>
#include <QDir>
#include <QDebug>
#include <QStandardItemModel>
#include <QTextDocument>
#include <QPrinter>
#include <QInputDialog>
#include <QDateEdit>
#include <QProcess>
#include <QPrintDialog>

MainWindow::MainWindow(QWidget parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    QPixmap pix(":/resources/img/image_blr_backgr.jpg");
    int w = ui->label->width();
    int h = ui->label->height();
    ui->label->setPixmap(pix.scaled(w, h, Qt::KeepAspectRatio));

    db = QSqlDatabase::addDatabase("SQLITE", "main_connection");

    // Load the stylesheet

```

```
}
```

```
MainWindow::~MainWindow()
```

```
{
```

```
    delete ui;
```

```
}
```

```
void MainWindow::createTables()
```

```
{
```

```
    if (!db.isOpen()) {
```

```
        ui->statusbar->showMessage("Database is not open!");
```

```
        return;
```

```
    }
```

```
    QSqlQuery query(db);
```

```
    if (!query.exec("CREATE TABLE IF NOT EXISTS intangible_assets ("
```

```
        "id INTEGER PRIMARY KEY AUTOINCREMENT, "
```

```
        "code TEXT, "
```

```
        "name TEXT, "
```

```
        "full_name TEXT, "
```

```
        "asset_type TEXT, "
```

```
        "account TEXT, "
```

```
        "initial_cost REAL, "
```

```
        "inventory_number TEXT, "
```

```
        "cipher TEXT, "
```

```
        "unit TEXT, "
```

```
        "entry_date DATE, "
```

```
        "activity_type TEXT))) {
```

```
        QMessageBox::critical(this, "Error", "Failed to create table: " + query.lastError().text());
```

```
        qDebug() << "Failed to create table: " << query.lastError().text();
```

```
    }
```

```
    if (!query.exec("CREATE TABLE IF NOT EXISTS tangible_assets ("
```

```
        "id INTEGER PRIMARY KEY AUTOINCREMENT, "
```

```
        "code TEXT, "
```

```
        "name TEXT, "
```

```
        "full_name TEXT, "
```

```
        "asset_type TEXT, "
```

```

        "account TEXT, "
        "initial_cost REAL, "
        "inventory_number TEXT, "
        "cipher TEXT, "
        "unit TEXT, "
        "entry_date DATE, "
        "activity_type TEXT))) {
    QMessageBox::critical(this, "Error", "Failed to create table: " + query.lastError().text());
    qDebug() << "Failed to create table: " << query.lastError().text();
}

if (!query.exec("CREATE TABLE IF NOT EXISTS intangible_assets_depreciation (\"
    \"id INTEGER PRIMARY KEY AUTOINCREMENT, \"
    \"asset_id INTEGER, \"
    \"expense_type TEXT, \"
    \"depreciation_method TEXT, \"
    \"salvage_value REAL, \"
    \"depreciation_term DATE, \"
    \"FOREIGN KEY(asset_id) REFERENCES intangible_assets(id)\")) {
    QMessageBox::critical(this, "Error", "Failed to create table 'intangible_assets_depreciation': " +
query.lastError().text());
    // qDebug() << "Failed to create table 'intangible_assets_depreciation': " << query.lastError().text();
    ui->statusbar->showMessage("Failed to create table 'intangible_assets_depreciation': " + query.lastError().text());
}

if (!query.exec("CREATE TABLE IF NOT EXISTS tangible_assets_depreciation (\"
    \"id INTEGER PRIMARY KEY AUTOINCREMENT, \"
    \"asset_id INTEGER, \"
    \"expense_type TEXT, \"
    \"depreciation_method TEXT, \"
    \"salvage_value REAL, \"
    \"depreciation_term DATE, \"
    \"FOREIGN KEY(asset_id) REFERENCES tangible_assets(id)\")) {
    QMessageBox::critical(this, "Error", "Failed to create table 'tangible_assets_depreciation': " +
query.lastError().text());
    // qDebug() << "Failed to create table 'tangible_assets_depreciation': " << query.lastError().text();
    ui->statusbar->showMessage("Failed to create table 'tangible_assets_depreciation': " + query.lastError().text());
}
}

```

```
void MainWindow::on_action_cut_triggered()
{
    ui->commentSpace->cut();
}
```

```
void MainWindow::on_action_copy_triggered()
{
    ui->commentSpace->copy();
}
```

```
void MainWindow::on_action_paste_triggered()
{
    ui->commentSpace->paste();
}
```

```
void MainWindow::on_action_redo_triggered()
{
    ui->commentSpace->redo();
}
```

```
void MainWindow::on_action_undo_triggered()
{
    ui->commentSpace->undo();
}
```

```
void MainWindow::on_pushButton_createNewDb_clicked()
{
    dbcreate dialog(this);
    if (dialog.exec() == QDialog::Accepted)
    {
        dbName = dialog.getDatabaseName();
        db.setDatabaseName(dbName);

        if (!db.open()) {
            QMessageBox::critical(this, "Error", "Failed to open database: " + db.lastError().text());
            //QDebug() << "Failed to open database: " << db.lastError().text();
            ui->statusbar->showMessage("Failed to open database: " + db.lastError().text());
            return;
        }
    }
}
```

```

createTables();

tablepage tbl_p = new tablepage(dbName, this);
tbl_p->show();
}
}

void MainWindow::on_action_new_triggered()
{
    dbcreate dialog(this);
    if (dialog.exec() == QDialog::Accepted)
    {
        dbName = dialog.getDatabaseName();
        db.setDatabaseName(dbName);

        if (!db.open()) {
            QMessageBox::critical(this, "Error", "Failed to open database: " + db.lastError().text());
            //qDebug() << "Failed to open database: " << db.lastError().text();
            ui->statusbar->showMessage("Failed to open database: " + db.lastError().text());
            return;
        }

        createTables();

        tablepage tbl_p = new tablepage(dbName, this);
        tbl_p->show();
    }
}

void MainWindow::on_pushButton_generalView_clicked()
{
    tablepage tbl_p = new tablepage(dbName, this);
    tbl_p->setAttribute(Qt::WA_DeleteOnClose);
    tbl_p->show();
}

void MainWindow::on_action_quit_triggered()
{

```

```

        close();
    }

void MainWindow::on_action_open_triggered()
{
    QString fileName = QFileDialog::getOpenFileName(this, tr("Open Database"), "", tr("Database Files (*.db *.sqlite *.sqlite3);;All Files (*)"));

    if (fileName.isEmpty()) {
        return;
    }

    db.setDatabaseName(fileName);

    if (!db.open()) {
        QMessageBox::critical(this, tr("Error"), tr("Failed to open database"));
        return;
    }

    dbName = fileName;
    //qDebug() << "Database opened: " << dbName;
    ui->statusbar->showMessage( "Database opened: " + dbName);
    ui->pushButton_generalView->setEnabled(true);
    ui->pushButton_createNewDb->setEnabled(false);
}

void MainWindow::on_action_nmActives_triggered()
{
    NoneMaterialActives nmActives = new NoneMaterialActives(true, db, this);
    nmActives->exec();
}

void MainWindow::on_action_saveAs_triggered()
{
    QString fileName = QFileDialog::getSaveFileName(this, tr("Save Database"), "", tr("Database Files (*.db *.sqlite *.sqlite3)"));

    if (fileName.isEmpty()) {
        return;
    }

    db.setDatabaseName(fileName);

```

```

if (!db.open()) {
    QMessageBox::critical(this, tr("Error"), tr("Failed to create database"));
    return;
}

createTables();

QMessageBox::information(this, tr("Success"), tr("Database created and saved successfully"));
}

void MainWindow::on_action_fixedAss_triggered()
{
    NoneMaterialActives nmActives = new NoneMaterialActives(false, db, this);
    nmActives->exec();
}

void MainWindow::on_action_amActive_triggered()
{
    bool isIntangible = true;
    Authomation form = new Authomation(isIntangible, db, this);
    form->exec();
}

void MainWindow::on_action_amFixedAss_triggered()
{
    bool isIntangible = false;
    Authomation form = new Authomation(isIntangible, db, this);
    form->exec();
}

void MainWindow::on_action_generalView_triggered()
{
    on_pushButton_generalView_clicked();
}

void MainWindow::on_action_generateAssetsReport_triggered()
{
    generateReport generateRep = new generateReport("assets_report", db, this);

```

```

generateRep->exec();
}

void MainWindow::on_action_generateDepreciationReport_triggered()
{
    generateReport generateRep = new generateReport("depreciation_report", db, this);
    generateRep->exec();
}

void MainWindow::on_action_generateMoveReport_triggered()
{
    generateReport generateRep = new generateReport("move_report", db, this);
    generateRep->exec();
}

void MainWindow::on_action_generateInventoryReport_triggered()
{
    generateReport generateRep = new generateReport("inventory_report", db, this);
    generateRep->exec();
}

void MainWindow::on_action_generateCategoryReport_triggered()
{
    generateReport generateRep = new generateReport("category_report", db, this);
    generateRep->exec();
}

void MainWindow::on_action_aboutQt_triggered()
{
    QString aboutText = "Qt – це кросплатформовий фреймворк для розробки програмного забезпечення. "
        "Він дозволяє створювати високопродуктивні та сучасні застосунки для різних "
        "платформ, включаючи Windows, macOS, Linux, Android та iOS.\n\n"
        "Qt включає в себе широкий набір інструментів і бібліотек, які забезпечують багато "
        "можливостей для створення графічного інтерфейсу, роботи з мережею, базами даних, "
        "мультимедіа та багато іншого.\n\n"
        "Завдяки своїй кросплатформенності та потужним функціональним можливостям, Qt "
        "широко використовується для розробки як настільних, так і мобільних застосунків, а також "
        "для вбудованих систем та Інтернету речей.";
}

```

```
    QMessageBox::about(this, "Про Qt", aboutText);  
}
```

```
void MainWindow::on_action_aboutAuth_triggered()
```

```
{  
    QMessageBox::information(this, "Author", "Програма розроблена Студентом Ін-403, \nРозробник: Шахрай  
Дмитро Ігорович\n Для розробки був використаний open-source QT\n Not for comercial uses");  
}
```

```
void MainWindow::on_action_print_triggered()
```

```
{  
    QString fileName = QFileDialog::getOpenFileName(this, "Виберіть PDF файл", "", "PDF Files (.pdf)");
```

```
    if (!fileName.isEmpty())
```

```
{  
    QPrinter printer(QPrinter::HighResolution);  
    printer.setPrinterName("Printer Name");
```

```
    QPrintDialog dialog(&printer, this);
```

```
    if (dialog.exec() == QDialog::Accepted)
```

```
{  
    QFile file(fileName);  
    if (!file.open(QIODevice::ReadOnly))  
    {  
        QMessageBox::critical(this, "Помилка", "Не вдалося відкрити файл для друку");  
        return;  
    }  
}
```

```
    QTextDocument document;
```

```
    document.setHtml(file.readAll());
```

```
    document.print(&printer);
```

```
    file.close();
```

```
}
```

```
}
```

```
}
```

```
void MainWindow::on_action_Wiki_triggered()
```

{

QString aboutText = "Нематеріальні активи\n"

"Нематеріальні активи - це активи, які не мають фізичної форми, але мають цінність для підприємства. Вони включають інтелектуальну власність та інші активи, які можуть приносити дохід або забезпечувати конкурентні переваги.\n"

"Приклади нематеріальних активів:\n"

"\n"

"Патенти — права на винаходи, які забезпечують ексклюзивне використання та захист від копіювання іншими.\n"

"Авторські права - права на літературні, музичні та інші художні твори, що забезпечують захист від несанкціонованого використання.\n"

"Торгові марки — це символи, назви або логотипи, які відрізняють товари або послуги однієї компанії від інших.\n"

"Франшизи - права на використання бізнес-моделі або бренду іншої компанії.\n"

"Ділова репутація (гудвілл) — нематеріальна цінність компанії, пов'язана з її репутацією, клієнтською базою та іншими нематеріальними факторами, які можуть підвищити її ринкову вартість.\n"

"Програмне забезпечення — ліцензії на використання певних програмних продуктів.\n"

"\n"

"Основні засоби\n"

"Основні кошти - це матеріальні активи, які використовуються у виробничій діяльності компанії та мають тривалий термін служби. Вони включають фізичні об'єкти, які необхідні для функціонування підприємства.\n"

"Приклади основних засобів:\n"

"\n"

"Земля — ділянки землі, що використовуються для розміщення будівель, сільськогосподарських потреб чи інших цілей.\n"

"Будівлі та споруди — офісні будівлі, фабрики, склади та інші будівлі, що використовуються для ведення бізнесу.\n"

"Машини та обладнання — виробничі машини, інструменти, транспортні засоби та інші пристрої, що використовуються у виробничому процесі.\n"

"Комп'ютери та офісна техніка — комп'ютери, принтери, сканери та інші пристрої, необхідні для адміністративної діяльності.\n"

"Меблі та офісне обладнання - столи, стільці, шафи та інші предмети, що використовуються в офісах та інших робочих місцях.\n"

"\n"

"Відмінність нематеріальних активів від основних засобів:\n"

"Фізична форма:\n"

"Нематеріальні активи: Не мають фізичної форми.\n"

"Основні засоби: Мають фізичну форму.\n"

"\n"

"Природа використання:\n"

"Нематеріальні активи: В основному пов'язані з правами, інтелектуальною власністю та іншими нематеріальними перевагами.\n"

"Основні засоби: Використовуються у виробничій та адміністративній діяльності підприємства, забезпечуючи фізичну інфраструктуру.\n"

"\n"

"Спосіб амортизації:\n"

"Нематеріальні активи: Амортизація зазвичай розраховується на основі терміну корисного використання активів, який може бути обмежений або необмежений.\n"

"Основні засоби: Амортизація розраховується на основі фізичного зносу та терміну служби активів.\n";

QMessageBox::about(this, "Про Нематеріальні активи і основні засоби", aboutText);

}

Nonematerialactives.cpp

```
#include "nonematerialactives.h"
```

```
#include "ui_nonematerialactives.h"
```

```
#include <authomation.h>
```

```
#include <QMessageBox>
```

```
#include <QSqlQuery>
```

```
#include <QSqlError>
```

NoneMaterialActives::NoneMaterialActives(const bool isNMA, QSqlDatabase database, QWidget parent)

```
: QDialog(parent)
```

```
, ui(new Ui::NoneMaterialActives)
```

```
, db(database)
```

```
{
```

```
ui->setUi(this);
```

```
if (!db.isOpen()) {
```

```
    QMessageBox::critical(this, "Database Error", "Failed to open the database: " + db.lastError().text());
```

```
    return;
```

```
}
```

```
// Заполнение виджетов списка (ComboBox)
```

```
ui->comboBox_vid_aktiva->addItem({"основні засоби", "інші необоротні активи", "нематеріальні активи",  
"гудвіл", "інші нематеріальні активи"});
```

```
ui->comboBox_rakhunok_obliku->addItem({"10 - Основні засоби 10", "100 - Інвестиційна нерухомість 100",  
"101 - Земельні ділянки 101", "102 - Капітальні витрати на поліпшення земель 102", "103 - Будинки та  
споруди 103", "104 - Машини та обладнання 104", "105 - Транспортні засоби 105", "106 - Інструменти,  
прилади та інвентар 106", "107 - Тварини 107", "108 - Багаторічні насадження 108", "109 - Інші основні  
засоби 109", "11 - Необоротні матеріальні активи 11", "111 - Бібліотечні фонди 111", "112 - Малоцінні  
необоротні матеріальні активи 112", "113 - Тимчасові (нетитульні) споруди 113", "114 - Природні ресурси  
114", "115 - Інвентарна тара 115", "116 - Предмет прокату 116", "117 - Інші необоротні активи 117", "12 -  
Нематеріальні активи 12", "121 - Права користування природними ресурсами 121", "122 - Права  
користування майном 122", "123 - Права на комерційні позначення 123", "124 - Права на об'єкти  
промислової власності 124", "125 - Авторське право та суміжні з ним права 125", "127 - Інші нематеріальні  
активи 127"});
```

```
ui->comboBox_odunitsya_vymiru->addItem({"шт", "кг", "л", "м", "м²", "м³", "г", "т"});
```

```
ui->comboBox_vid_diyalnosti->addItem({"основна", "додаткова"});
```

```
if (isNMA)
```

```
{
```

```
ui->comboBox_vid_aktiva->setCurrentIndex(2);
```

```
ui->comboBox_rakhunok_obliku->setCurrentIndex(11);
```

```
connect(ui->pushButton_2, &QPushButton::clicked, this, &NoneMaterialActives::IntangibleAssetDepreciation);
```

```
connect(ui->buttonBox, &QDialogButtonBox::accepted, this,  
&NoneMaterialActives::buttonBox_accepted_for_IntangibleAsset);
```

```
}
```

```

else
{
    ui->comboBox_vid_aktiva->setCurrentIndex(0);
    ui->comboBox_rakhunok_obliku->setCurrentIndex(0);
    connect(ui->pushButton_2, &QPushButton::clicked, this, &NoneMaterialActives::TangibleAssetDepreciation);
    connect(ui->buttonBox, &QDialogButtonBox::accepted, this,
    &NoneMaterialActives::buttonBox_accepted_for_TangibleAsset);
}
}

```

```

NoneMaterialActives::~NoneMaterialActives()

```

```

{
    delete ui;
}

```

```

void NoneMaterialActives::TangibleAssetDepreciation()

```

```

{
    // Определите, для какого типа активов вы хотите добавить амортизацию
    bool isIntangible = false; // или false, в зависимости от вашего выбора

    Authomation form = new Authomation(isIntangible, db, this);
    form->exec(); // Покажите форму как модальное диалоговое окно

}

```

```

void NoneMaterialActives::IntangibleAssetDepreciation()

```

```

{
    // Определите, для какого типа активов вы хотите добавить амортизацию
    bool isIntangible = true; // или false, в зависимости от вашего выбора

    Authomation form = new Authomation(isIntangible, db, this);
    form->exec(); // Покажите форму как модальное диалоговое окно

}

```

```

void NoneMaterialActives::buttonBox_accepted_for_InTangibleAsset()

```

```

{

    QString code = ui->lineEdit_kod->text();
    QString name = ui->lineEdit_nazva->text();
    QString fullName = ui->lineEdit_povna_nazva->text();

```

```

QString assetType = ui->comboBox_vid_aktiva->currentText();
QString account = ui->comboBox_rakhunok_obliku->currentText().split(" - ").first();
double initialCost = ui->doubleSpinBox_pochatkova_vartist->value();
QString inventoryNumber = ui->lineEdit_inventarnyy_nomer->text();
QString cipher = ui->lineEdit_shifr->text();
QString unit = ui->comboBox_odinitsya_vymiru->currentText();
QDate entryDate = ui->dateEdit_data_vvodu->date();
QString activityType = ui->comboBox_vid_diyalnosti->currentText();

if (code.isEmpty() || name.isEmpty() || fullName.isEmpty() || inventoryNumber.isEmpty() || cipher.isEmpty()) {
    QMessageBox::warning(this, "Input Error", "Please fill in all required fields.");
    return;
}

}

 QSqlQuery query(db);

    query.prepare("INSERT INTO intangible_assets (code, name, full_name, asset_type, account, initial_cost,
inventory_number, cipher, unit, entry_date, activity_type) "
        "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");

    query.addBindValue(code);
    query.addBindValue(name);
    query.addBindValue(fullName);
    query.addBindValue(assetType);
    query.addBindValue(account);
    query.addBindValue(initialCost);
    query.addBindValue(inventoryNumber);
    query.addBindValue(cipher);
    query.addBindValue(unit);
    query.addBindValue(entryDate);
    query.addBindValue(activityType);

    if (!query.exec()) {
        QMessageBox::critical(this, "Database Error", "Failed to insert data into database.");
    } else {
        QMessageBox::information(this, "Success", "Data has been successfully added to the database.");
        this->accept(); // Закрыть окно при успешном добавлении данных
    }

}

void NoneMaterialActives::buttonBox_accepted_for_TangibleAsset()

```

```

{

QString code = ui->lineEdit_kod->text();
QString name = ui->lineEdit_nazva->text();
QString fullName = ui->lineEdit_povna_nazva->text();
QString assetType = ui->comboBox_vid_aktiva->currentText();
QString account = ui->comboBox_rakhunok_obliku->currentText().split(" - ").first();
double initialCost = ui->doubleSpinBox_pochatkova_vartist->value();
QString inventoryNumber = ui->lineEdit_inventarnyy_nomer->text();
QString cipher = ui->lineEdit_shifr->text();
QString unit = ui->comboBox_odinitsya_vymiru->currentText();
QDate entryDate = ui->dateEdit_data_vvodu->date();
QString activityType = ui->comboBox_vid_diyalnosti->currentText();

if (code.isEmpty() || name.isEmpty() || fullName.isEmpty() || inventoryNumber.isEmpty() || cipher.isEmpty()) {
    QMessageBox::warning(this, "Input Error", "Please fill in all required fields.");
    return;
}

QString query(db);

query.prepare("INSERT INTO tangible_assets (code, name, full_name, asset_type, account, initial_cost,
inventory_number, cipher, unit, entry_date, activity_type) "
    "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");

query.addBindValue(code);
query.addBindValue(name);
query.addBindValue(fullName);
query.addBindValue(assetType);
query.addBindValue(account);
query.addBindValue(initialCost);
query.addBindValue(inventoryNumber);
query.addBindValue(cipher);
query.addBindValue(unit);
query.addBindValue(entryDate);
query.addBindValue(activityType);

if (!query.exec()) {
    QMessageBox::critical(this, "Database Error", "Failed to insert data into database.");
} else {
    QMessageBox::information(this, "Success", "Data has been successfully added to the database.");
}
}

```

```
this->accept(); // Закреть окно при успешном добавлении данных
}

}
```

```
void NoneMaterialActives::on_pushButton_clicked()
```

```
{
    QString aboutText = "Нематеріальні активи\n"

        "Нематеріальні активи - це активи, які не мають фізичної форми, але мають цінність для підприємства. Вони включають інтелектуальну власність та інші активи, які можуть приносити дохід або забезпечувати конкурентні переваги.\n"

        "Приклади нематеріальних активів:\n"

        "\n"

        "Патенти — права на винаходи, які забезпечують ексклюзивне використання та захист від копіювання іншими.\n"

        "Авторські права - права на літературні, музичні та інші художні твори, що забезпечують захист від несанкціонованого використання.\n"

        "Торгові марки — це символи, назви або логотипи, які відрізняють товари або послуги однієї компанії від інших.\n"

        "Франшизи - права на використання бізнес-моделі або бренду іншої компанії.\n"

        "Ділова репутація (гудвілл) — нематеріальна цінність компанії, пов'язана з її репутацією, клієнтською базою та іншими нематеріальними факторами, які можуть підвищити її ринкову вартість.\n"

        "Програмне забезпечення — ліцензії на використання певних програмних продуктів.\n"

        "\n"

        "Основні засоби\n"

        "Основні кошти - це матеріальні активи, які використовуються у виробничій діяльності компанії та мають тривалий термін служби. Вони включають фізичні об'єкти, які необхідні для функціонування підприємства.\n"

        "Приклади основних засобів:\n"

        "\n"

        "Земля — ділянки землі, що використовуються для розміщення будівель, сільськогосподарських потреб чи інших цілей.\n"

        "Будівлі та споруди — офісні будівлі, фабрики, склади та інші будівлі, що використовуються для ведення бізнесу.\n"

        "Машини та обладнання — виробничі машини, інструменти, транспортні засоби та інші пристрої, що використовуються у виробничому процесі.\n"

        "Комп'ютери та офісна техніка — комп'ютери, принтери, сканери та інші пристрої, необхідні для адміністративної діяльності.\n"

        "Меблі та офісне обладнання - столи, стільці, шафи та інші предмети, що використовуються в офісах та інших робочих місцях.\n"

        "\n"

        "Відмінність нематеріальних активів від основних засобів:\n"
```

"Фізична форма:\n"

"Нематеріальні активи: Не мають фізичної форми.\n"

"Основні засоби: Мають фізичну форму.\n"

"\n"

"Природа використання:\n"

"Нематеріальні активи: В основному пов'язані з правами, інтелектуальною власністю та іншими нематеріальними перевагами.\n"

"Основні засоби: Використовуються у виробничій та адміністративній діяльності підприємства, забезпечуючи фізичну інфраструктуру.\n"

"\n"

"Спосіб амортизації:\n"

"Нематеріальні активи: Амортизація зазвичай розраховується на основі терміну корисного використання активів, який може бути обмежений або необмежений.\n"

"Основні засоби: Амортизація розраховується на основі фізичного зносу та терміну служби активів.\n";

QMessageBox::about(this, "Про Нематеріальні активи і основні засоби", aboutText);

}

Tablepage.cpp

```
#include "tablepage.h"
#include "ui_tablepage.h"
#include "editactive.h"
#include "editauthomation.h"
#include "dbcreate.cpp"
#include <QSqlQuery>
#include <QSqlRecord>
#include <QMessageBox>
#include <QInputDialog>
#include <QSqlError>

tablepage::tablepage(const QString &dbName, QWidget parent)
    : QDialog(parent)
    , ui(new Ui::tablepage)

{
    ui->setupUi(this);
    openDatabase(dbName);
    setupModels();
    if (!db.isOpen()) {
        QMessageBox::critical(this, "Database Error", "Failed to open the database: " + db.lastError().text());
        return;
    }
}

tablepage::~tablepage()
{
    delete ui;
}

void tablepage::openDatabase(const QString &dbName)
{
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(dbName);

    if (!db.open()) {
        QMessageBox::critical(this, "Error", "Failed to open database.");
    }
}
```

```

void tablepage::setupModels()
{

    intangibleModel = new QSqlTableModel(this, db);
    intangibleModel->setTable("intangible_assets");
    intangibleModel->select();
    ui->tableView->setModel(intangibleModel);

    tangibleModel = new QSqlTableModel(this, db);
    tangibleModel->setTable("tangible_assets");
    tangibleModel->select();
    ui->tableView_2->setModel(tangibleModel);

    // Исправляем ошибку здесь
    intangible_assets_depreciation = new QSqlTableModel(this, db);
    intangible_assets_depreciation->setTable("intangible_assets_depreciation");
    intangible_assets_depreciation->select();
    ui->tableView_3->setModel(intangible_assets_depreciation);

    tangible_assets_depreciation = new QSqlTableModel(this, db);
    tangible_assets_depreciation->setTable("tangible_assets_depreciation");
    tangible_assets_depreciation->select();
    ui->tableView_4->setModel(tangible_assets_depreciation);

}

void tablepage::on_tableView_doubleClicked(const QModelIndex &index)
{

    if (!index.isValid()) {
        return;
    }

    int assetId = intangibleModel->data(intangibleModel->index(index.row(), 0)).toInt(); // Получаем id актива из модели
    if (assetId != -1) {

```

```

        QSqlDatabase db = QSqlDatabase::database(); // Получаем подключение к БД

        editActive editDialog(db, assetId, false, this);
        if (editDialog.exec() == QDialog::Accepted) {
            // Обновить таблицу после успешного редактирования
            intangibleModel->select();
        }
    }
}

void tablepage::on_tableView_2_doubleClicked(const QModelIndex &index)
{
    if (!index.isValid()) {
        return;
    }

    int assetId = tangibleModel->data(tangibleModel->index(index.row(), 0)).toInt(); // Получаем id актива из модели
    if (assetId != -1) {
        QSqlDatabase db = QSqlDatabase::database(); // Получаем подключение к БД

        editActive editDialog(db, assetId, true, this);
        if (editDialog.exec() == QDialog::Accepted) {
            // Обновить таблицу после успешного редактирования
            tangibleModel->select();
        }
    }
}

void tablepage::on_tableView_3_doubleClicked(const QModelIndex &index)
{
    if (!index.isValid()) {
        return;
    }

    int assetId = intangible_assets_depreciation->data(intangible_assets_depreciation->index(index.row(), 1)).toInt();
    // Получаем id актива из модели

```

```

if (assetId != -1) {
    QSqlDatabase db = QSqlDatabase::database(); // Получаем подключение к БД

    editAuthomation editDialogAuth(db, assetId, false, this);
    if (editDialogAuth.exec() == QDialog::Accepted) {
        // Обновить таблицу после успешного редактирования
        intangible_assets_depreciation->select();
    }
}

}

void tablepage::on_tableView_4_doubleClicked(const QModelIndex &index)
{
    if (!index.isValid()) {
        return;
    }

    int assetId = tangible_assets_depreciation->data(tangible_assets_depreciation->index(index.row(), 1)).toInt(); //
    Получаем id актива из модели
    if (assetId != -1) {
        QSqlDatabase db = QSqlDatabase::database(); // Получаем подключение к БД

        editAuthomation editDialogAuth(db, assetId, true, this);
        if (editDialogAuth.exec() == QDialog::Accepted) {
            // Обновить таблицу после успешного редактирования
            tangible_assets_depreciation->select();
        }
    }
}

/

void tablepage::loadAssets()
{
    // Реализация загрузки активов в QTableView
    QSqlQueryModel model = new QSqlQueryModel();

    model->setQuery("SELECT id, code, name, full_name, asset_type, account, initial_cost, inventory_number,
    cipher, unit, entry_date, activity_type FROM intangible_assets", QSqlDatabase::database());

    ui->tableView->setModel(model);
}

```

```
// Настройка представления для отображения id актива в роли Qt::UserRole
for (int row = 0; row < model->rowCount(); ++row) {
    QModelIndex index = model->index(row, 0); // 0 - колонка id
    int assetId = model->data(index).toInt();
    model->setData(index, assetId, Qt::UserRole);
}
}/
```

Mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
  <class>MainWindow</class>
  <widget class="QMainWindow" name="MainWindow">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>783</width>
        <height>597</height>
      </rect>
    </property>
    <property name="windowTitle">
      <string>Головна сторінка</string>
    </property>
    <property name="styleSheet">
      <string notr="true"/>
    </property>
    <widget class="QWidget" name="centralwidget">
      <widget class="QWidget" name="layoutWidget">
        <property name="geometry">
          <rect>
            <x>10</x>
            <y>10</y>
            <width>771</width>
            <height>501</height>
          </rect>
        </property>
        <layout class="QVBoxLayout" name="verticalLayout">
          <item>
            <widget class="QGroupBox" name="groupBox">
              <property name="title">
                <string>Почати роботу</string>
              </property>
              <widget class="QLabel" name="label">
                <property name="geometry">
                  <rect>
                    <x>20</x>
```

```
<y>40</y>
<width>351</width>
<height>201</height>
</rect>
</property>
<property name="maximumSize">
  <size>
    <width>351</width>
    <height>201</height>
  </size>
</property>
<property name="text">
  <string/>
</property>
</widget>
<widget class="QWidget" name="layoutWidget">
  <property name="geometry">
    <rect>
      <x>450</x>
      <y>40</y>
      <width>258</width>
      <height>191</height>
    </rect>
  </property>
  <layout class="QVBoxLayout" name="verticalLayout_3">
    <item>
      <layout class="QVBoxLayout" name="verticalLayout_2">
        <item>
          <widget class="QPushButton" name="pushButton_createNewDb">
            <property name="styleSheet">
              <string notr="true"/>
            </property>
            <property name="text">
              <string>Створити нове підприємство</string>
            </property>
          </widget>
        </item>
        <item>
          <widget class="QPushButton" name="pushButton_generalView">
```

```
<property name="text">
<string>Загальний вигляд</string>
</property>
</widget>
</item>
</layout>
</item>
</layout>
</widget>
<widget class="QGroupBox" name="groupBox_2">
<property name="geometry">
<rect>
<x>10</x>
<y>400</y>
<width>731</width>
<height>91</height>
</rect>
</property>
<property name="maximumSize">
<size>
<width>731</width>
<height>16777215</height>
</size>
</property>
<property name="styleSheet">
<string notr="true"/>
</property>
<property name="title">
<string>Поле для нотаток</string>
</property>
<widget class="QTextEdit" name="commentSpace">
<property name="geometry">
<rect>
<x>10</x>
<y>30</y>
<width>711</width>
<height>61</height>
</rect>
</property>
```

```

<property name="maximumSize">
<size>
<width>731</width>
<height>61</height>
</size>
</property>
</widget>
</widget>
<widget class="QLabel" name="label_2">
<property name="enabled">
<bool>>false</bool>
</property>
<property name="geometry">
<rect>
<x>10</x>
<y>290</y>
<width>721</width>
<height>101</height>
</rect>
</property>
<property name="font">
<font>
<family>Cambria</family>
<pointsize>48</pointsize>
<italic>>true</italic>
<bold>>true</bold>
</font>
</property>
<property name="mouseTracking">
<bool>>false</bool>
</property>
<property name="styleSheet">
<string notr="true">color: rgb(120, 120, 120);
qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:0, stop:0 rgba(0, 0, 0, 255), stop:1 rgba(255, 255, 255, 255));
</string>
</property>
<property name="frameShape">
<enum>QFrame::NoFrame</enum>
</property>

```

```
<property name="frameShadow">
  <enum>QFrame::Sunken</enum>
</property>
<property name="lineWidth">
  <number>4</number>
</property>
<property name="midLineWidth">
  <number>1</number>
</property>
<property name="text">
  <string>Asset Tracker</string>
</property>
<property name="textFormat">
  <enum>Qt::AutoText</enum>
</property>
<property name="alignment">
  <set>Qt::AlignCenter</set>
</property>
<property name="wordWrap">
  <bool>true</bool>
</property>
</widget>
</widget>
</item>
</layout>
</widget>
</widget>
<widget class="QMenuBar" name="menubar">
  <property name="geometry">
    <rect>
      <x>0</x>
      <y>0</y>
      <width>783</width>
      <height>21</height>
    </rect>
  </property>
  <property name="styleSheet">
    <string notr="true"/>
  </property>
```

```
<widget class="QMenu" name="menuFile">
<property name="title">
<string>Файл</string>
</property>
<addaction name="action_new"/>
<addaction name="action_open"/>
<addaction name="separator"/>
<addaction name="action_saveAs"/>
<addaction name="action_print"/>
<addaction name="separator"/>
<addaction name="separator"/>
<addaction name="action_quit"/>
</widget>

<widget class="QMenu" name="menu">
<property name="title">
<string>Немат.активи\основні зас.</string>
</property>
<addaction name="action_nmActives"/>
<addaction name="action_amActive"/>
<addaction name="separator"/>
<addaction name="action_fixedAss"/>
<addaction name="action_amFixedAss"/>
<addaction name="separator"/>
<addaction name="action_generalView"/>
<addaction name="separator"/>
<addaction name="action_Wiki"/>
</widget>

<widget class="QMenu" name="menu_2">
<property name="title">
<string>Про нас</string>
</property>
<addaction name="action_aboutQt"/>
<addaction name="separator"/>
<addaction name="action_aboutAuth"/>
</widget>

<widget class="QMenu" name="menu_3">
<property name="title">
<string>Редагування</string>
</property>
```

```
<addaction name="action_copy"/>
<addaction name="action_cut"/>
<addaction name="action_paste"/>
<addaction name="action_undo"/>
<addaction name="action_redo"/>
<addaction name="separator"/>
</widget>
<widget class="QMenu" name="menu_4">
<property name="title">
<string>Звіт</string>
</property>
<addaction name="action_generateAssetsReport"/>
<addaction name="action_generateDepreciationReport"/>
<addaction name="action_generateMoveReport"/>
<addaction name="action_generateInventoryReport"/>
<addaction name="action_generateCategoryReport"/>
</widget>
<addaction name="menuFile"/>
<addaction name="menu_3"/>
<addaction name="menu"/>
<addaction name="menu_4"/>
<addaction name="menu_2"/>
</widget>
<widget class="QStatusBar" name="statusbar"/>
<widget class="QToolBar" name="toolBar">
<property name="windowTitle">
<string>toolBar</string>
</property>
<attribute name="toolBarArea">
<enum>TopToolBarArea</enum>
</attribute>
<attribute name="toolBarBreak">
<bool>>false</bool>
</attribute>
<addaction name="action_copy"/>
<addaction name="separator"/>
<addaction name="action_paste"/>
<addaction name="separator"/>
<addaction name="action_cut"/>
```

```
<addaction name="separator"/>
<addaction name="action_undo"/>
<addaction name="separator"/>
<addaction name="action_redo"/>
</widget>

<action name="action_new">
  <property name="text">
    <string>Створити</string>
  </property>
</action>

<action name="action_open">
  <property name="text">
    <string>Відкрити</string>
  </property>
</action>

<action name="action_print">
  <property name="text">
    <string>Друк</string>
  </property>
</action>

<action name="action_quit">
  <property name="text">
    <string>Вийти</string>
  </property>
</action>

<action name="action_nmActives">
  <property name="text">
    <string>Немат.активи</string>
  </property>
</action>

<action name="action_amActive">
  <property name="text">
    <string>Амортизація немат. активів</string>
  </property>
</action>

<action name="action_fixedAss">
  <property name="text">
    <string>Основні засоби</string>
  </property>
```

```
</action>

<action name="action_amFixedAss">
  <property name="text">
    <string>Амортизація основних засобів</string>
  </property>
</action>

<action name="action_generalView">
  <property name="text">
    <string>Загальний вигляд обліку</string>
  </property>
</action>

<action name="action_Wiki">
  <property name="text">
    <string>Довідник</string>
  </property>
</action>

<action name="action_aboutQt">
  <property name="text">
    <string>Про Qt</string>
  </property>
</action>

<action name="action_aboutAuth">
  <property name="text">
    <string>Про автора</string>
  </property>
</action>

<action name="action_copy">
  <property name="icon">
    <iconset resource="resources.qrc">
      <normaloff>:/img/copy_img.png</normaloff>:/img/copy_img.png</iconset>
    </property>
    <property name="text">
      <string>Копіювати</string>
    </property>
  </action>

  <action name="action_cut">
    <property name="icon">
      <iconset resource="resources.qrc">
        <normaloff>:/img/cut_img.jpg</normaloff>:/img/cut_img.jpg</iconset>
```

```
</property>
<property name="text">
<string>Випізнати</string>
</property>
</action>
<action name="action_undo">
<property name="icon">
<iconset resource="resources.qrc">
<normaloff>:/img/undo_icon.png</normaloff>:/img/undo_icon.png</iconset>
</property>
<property name="text">
<string>Undo</string>
</property>
</action>
<action name="action_redo">
<property name="icon">
<iconset resource="resources.qrc">
<normaloff>:/img/redo_icon.png</normaloff>:/img/redo_icon.png</iconset>
</property>
<property name="text">
<string>Redo</string>
</property>
</action>
<action name="action_paste">
<property name="icon">
<iconset resource="resources.qrc">
<normaloff>:/img/paste_img.png</normaloff>:/img/paste_img.png</iconset>
</property>
<property name="text">
<string>Вставити</string>
</property>
</action>
<action name="action_saveAs">
<property name="text">
<string>Зберегти як</string>
</property>
</action>
<action name="action_generateAssetsReport">
<property name="text">
```

```
<string>Звіт про активи</string>
</property>
</action>
<action name="action_generateDepreciationReport">
  <property name="text">
    <string>Звіт про амортизацію</string>
  </property>
</action>
<action name="action_generateMoveReport">
  <property name="text">
    <string>Звіт про рух активів</string>
  </property>
</action>
<action name="action_generateInventoryReport">
  <property name="text">
    <string>Звіт по інвентаному номеру</string>
  </property>
</action>
<action name="action_generateCategoryReport">
  <property name="text">
    <string>Звіт по категоріям активів</string>
  </property>
</action>
</widget>
<resources>
  <include location="resources.qrc"/>
</resources>
<connections/>
</ui>
```

Diffness.qss

/-----QWidget-----/

QWidget

```
{
    background-color: #bfc1c4;
    color: black;
    selection-background-color: #fff;
    selection-color: #000;
}
```

/-----QLabel-----/

QLabel

```
{
    background-color: transparent;
    color: black;
}
```

/-----QMenuBar-----/

QMenuBar

```
{
    background-color: #4a5157;
    color: #fff;
}
```

QMenuBar::item

```
{
    background-color: transparent;
    border-left: 1px solid #003333;
    padding: 5px;
    padding-left: 15px;
    padding-right: 15px;
}
```

```
}
```

```
QMenuBar::item:selected
```

```
{  
    background-color: #003333;  
    border: 1px solid #006666;  
    color: #fff;  
}
```

```
QMenuBar::item:pressed
```

```
{  
    background-color: #006666;  
    border: 1px solid #006666;  
    color: #fff;  
}
```

```
/-----QMenu-----/
```

```
QMenu
```

```
{  
    background-color: #4a5157;  
    border: 1px solid #4a5157;  
    padding: 10px;  
    color: #fff;  
}
```

```
QMenu::item
```

```
{  
    background-color: transparent;  
    padding: 2px 20px 2px 20px;  
    min-width: 200px;
```

```
}
```

QMenu::separator

```
{  
    background-color: black;  
    height: 1px;  
  
}
```

QMenu::item:disabled

```
{  
    color: #555;  
    background-color: transparent;  
    padding: 2px 20px 2px 20px;  
  
}
```

QMenu::item:selected

```
{  
    background-color: #003333;  
    border: 1px solid #006666;  
    color: #fff;  
  
}
```

Имя пользователя: Інформаційних систем в економіці Шкуратовська Те...	ID проверки: 1016350065
Дата проверки: 12.06.2024 05:42:32 EEST	Тип проверки: Doc vs Internet + Library
Дата отчета: 12.06.2024 07:12:32 EEST	ID пользователя: 100005745

Название файла: Шахрай_Д

Количество страниц: 106 Количество слов: 14342 Количество символов: 116423 Размер файла: 4.48 MB ID файла: 1016151823

13% Совпадения

Наибольшее совпадение: 4.23% с источником из Библиотеки (ID файла: 1016112527)

4.61% Источники из Интернета 236 Страница 108

12.4% Источники из Библиотеки 568 Страница 112

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

Форма навчання: денна

КВАЛІФІКАЦІЙНИЙ БАКАЛАВРСЬКИЙ ПРОЄКТ

на тему

ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ВЕДЕННЯ
ОБЛІКУ ОСНОВНИХ ЗАСОБІВ І НЕМАТЕРІАЛЬНИХ АКТИВІВ У
БУХГАЛТЕРСЬКОМУ ОБЛІКУ

здобувача Шахрай Дмитра Ігоровича

Науковий керівник:

к.е.н., доцент

Ситник Н.В.

Проект допущений до захисту перед екзаменаційною комісією з
атестації здобувачів вищої освіти

завідувач кафедри:

к.е.н., доцент

Тішков Б.О.

Київ 2024