

**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

«Інформаційні управляючі системи і технології»

галузь знань 12 Інформаційні технології

спеціальність 122 Комп'ютерні науки

Форма навчання: денна

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему: «Розроблення інформаційної управляючої системи для моніторингу та
обліку робочого часу працівників»

здобувачки **Новак Катерини Миколаївни**
_____ (підпис)

Науковий керівник: д.е.н., проф. Ріппа С.П.
(науковий ступінь, учене звання, ПІБ)
_____ (підпис)

**Робота допущена до захисту перед екзаменаційною
комісією з атестації здобувачів вищої освіти (ЕК)**

Завідувач кафедри: к.е.н., доцент Тішков Б.О.

(підпис)

**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
«Інформаційні управляючі системи і технології»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

ПОГОДЖЕНО:
Керівник проєктної групи (гарант)
освітньо-професійної програми

ЗАТВЕРДЖУЮ:
Завідувач кафедри інформаційних
систем в економіці

_____ Устенко С.В.
« ____ » _____ 2025 р.

_____ Тішков Б.О.
« ____ » _____ 2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
здобувача вищої освіти **Новак Катерини Миколаївни**
очної (денної) форми навчання

на підготовку кваліфікаційної магістерської роботи
на тему: «Розроблення ІУС для моніторингу та обліку робочого часу працівників»

Тему затверджено наказом ректора Університету від «20» лютого 2025 р. № **332-ст**
Кваліфікаційна магістерська робота виконується на матеріалах
умовного підприємства та Інтернет-ресурсів

План кваліфікаційної магістерської роботи

Розділ I ТЕОРЕТИЧНИЙ РОЗДІЛ. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА АНАЛІЗ ІСНУЮЧИХ СИСТЕМ

Розділ II АНАЛІТИЧНИЙ РОЗДІЛ. СТВОРЕННЯ МОДЕЛЕЙ, РОЗГЛЯД ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ УПРАВЛІННЯ ПРОЕКТАМИ

Розділ III КОНСТРУКТИВНИЙ РОЗДІЛ. РОЗРОБЛЕННЯ ПРОЕКТНИХ РІШЕНЬ ТА ЇХ РЕАЛІЗАЦІЯ

Об'єкт дослідження: системи управління робочим часом на підприємствах.

Предмет дослідження: інформаційна управляюча система для автоматизованого моніторингу та обліку робочого часу працівників.

Мета кваліфікаційної магістерської роботи: розробка та впровадження інформаційної управляючої системи (ІУС) для автоматизованого моніторингу та обліку робочого часу працівників підприємства. Ця система повинна фіксувати час початку та завершення роботи, обробляти дані про виконані завдання, а також надавати аналітичні звіти для керівництва.

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

У розділі I: Дослідити предметну область (моніторинг робочого часу), провести аналіз існуючих інформаційних систем з моніторингу часу та зробити обґрунтування обраних технологій.

У розділі II Побудувати діаграми UML, що відображатимуть роботу користувачів в системі, описати постановку задач, моделі і методи моніторингу робочого часу, діаграми взаємодії і послідовності ІУС обліку та аналізу робочого часу. Розглянути основні методи візуалізації в системі моніторингу робочого часу.

У розділі III Представити види забезпечення інформаційної системи: інформаційного, технічного, програмного; продемонструвати результати реалізації ІУС обліку та моніторингу робочого часу.

—

**Завдання підготував
науковий керівник**

(підпис)

Ріппа Сергій Петрович

(ініціали, прізвище)

« ____ » _____ 202_ р.

**Завдання одержав
здобувач**

(підпис)

Новак Катерина Миколаївна

(ініціали, прізвище)

« ____ » _____ 202_ р.

РЕЦЕНЗІЯ

на кваліфікаційну магістерську роботу
здобувачки Новак Катерини Миколаївни
на тему «РОЗРОБЛЕННЯ ІУС ДЛЯ МОНІТОРИНГУ ТА ОБЛІКУ РОБОЧОГО ЧАСУ
ПРАЦІВНИКІВ»
за спеціальністю 122 «Комп'ютерні науки»

Актуальність теми: Представлена на рецензування кваліфікаційна магістерська робота за своєю тематикою та актуальністю відповідає сьогодишнім викликам і загрозам, перед якими опинилась Україна у сфері управління трудовими ресурсами, оскільки в тяжких умовах тривалої повномасштабної війни з росією, коли триває воєнний стан, мільйони громадян України знаходяться в евакуації і є тимчасово-переміщеними особами.

Характеристика роботи. Проведене здобувачем дослідження складається традиційним чином із трьох розділів і 8-ми додатків. Перший розділ присвячений характеристиці предметної області та об'єкта дослідження, а також аналізу профільної літератури, де здобувач компактно та інформативно представляє актуальну ситуацію і динаміку розвитку сфери систем і технологій моніторингу робочого часу та контролю продуктивності праці. В другому розділі поданий детальний аналіз і формулювання специфікацій, UseCase діаграми основних процесів щодо обліку робочого часу і моніторингу персоналу та представлена діаграма класів, які можна віднести до авторської новизни. Третій розділ зосереджений на проєктних рішеннях щодо інфологічної моделі БД, обґрунтуванні вибору СКБД, проєктуванні бази знань та засобах інтелектуального аналізу даних, розглядається архітектура та інструментарій програмного забезпечення, а також тестування і технічне забезпечення проєкту. Змістовно і структурно виконаний проєкт відповідає обраній темі та сформульованим завданням, її оформлення також вкладається в основні нормативи.

Виконана робота має окремі несуттєві недоліки, які не впливають на остаточну оцінку: зокрема, участь здобувачки в конференції і оприлюднені тези можна було б відобразити у списку використаних джерел, є несуттєві помилки і опіски в оформленні роботи.

Загальний висновок. У цілому представлений на рецензування кваліфікаційна магістерська робота на тему «РОЗРОБЛЕННЯ ІУС ДЛЯ МОНІТОРИНГУ ТА ОБЛІКУ РОБОЧОГО ЧАСУ ПРАЦІВНИКІВ» на здобуття ступеня магістра свідчить про належну кваліфікацію автора щодо обраної тематики, проєкт виконаний на якісному рівні, за змістом та практичною цінністю відповідає вимогам МОН і на високу оцінку, а її авторка, Новак Катерина Миколаївна заслуговує на присвоєння їй освітньо-кваліфікаційного рівня «магістр» за спеціальністю «Комп'ютерні науки».

Рецензент:



Погореловська І.Д., к.е.н., доцент,
доцент кафедри комп'ютерних та інформаційних
технологій і систем факультету фінансів та
цифрових технологій Державного
податкового університету

«19» травня 2025 р.

Відгук
на кваліфікаційну магістерську роботу
здобувачки вищої освіти КНЕУ
Новак Катерини Миколаївни на тему:
**«РОЗРОБЛЕННЯ ІУС ДЛЯ МОНІТОРИНГУ ТА ОБЛІКУ РОБОЧОГО
ЧАСУ ПРАЦІВНИКІВ»**

Актуальність теми дослідження. Тематика кваліфікаційної магістерської роботи є беззаперечно актуальною, оскільки в неї розглядаються сучасні технології інформатизації систем управління персоналом, моніторингу та обліку робочого часу, що є важливою складовою реформування ринку управління трудовими ресурсами в Україні, в т.ч. з врахуванням складних умов сьогоденного воєнного стану і великої кількості переміщених та евакуйованих осіб.

Наукова новизна. Представлені в проєкті матеріали мають певні елементи наукової новизни, зокрема: здобувач досить деталізовано вибудовує концепцію організації сучасних ІТ-сервісів управління персоналом, які представлені на ринку, досліджує їх переваги та недоліки і пропонує свій, персоніфікований для організації моніторингових і облікових процесів, формування відповідних БД.

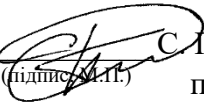
Якість проведеного аналізу проблеми, текстового та табличного матеріалу. Підготовлена до захисту кваліфікаційна магістерська робота вміщує результати опрацювання автором як теоретичних аспектів розробки ІУС моніторингу і обліку робочого часу, так й практичних надбань запровадження різноманітних веб-сервісів і ІТ-послуг в цієї сфері, в т.ч. з врахуванням певних обмежень і рекомендацій профільних фахівців. Текст роботи досить належним чином доповнено табличними матеріалами, графікою і прикладами веб-документів.

Обґрунтованість пропозицій і висновків та їх практична цінність. Наведені автором пропозиції щодо застосування інструментальних засобів розробки та впровадження ІУС моніторингу та обліку робочого часу можна вважати у достатній мірі обґрунтованими, а завершальні висновки за результатами виконаного проєкту належним чином узагальнюють його основний зміст. Результати виконання кваліфікаційної роботи можуть бути корисні для застосування в процесах проєктування і розробки веб-ресурсів установ та організацій в сфері управління персоналом. В роботі представлений працюючий прототип розробленої системи.

Наявність недоліків. Суттєвих зауважень до проєкту не має, є незначні відхилення від норм форматування, які не знижують на загальну оцінку.

Загальний висновок і оцінка роботи. Кваліфікаційна магістерська робота на тему «РОЗРОБЛЕННЯ ІУС ДЛЯ МОНІТОРИНГУ ТА ОБЛІКУ РОБОЧОГО ЧАСУ ПРАЦІВНИКІВ» у цілому відповідає встановленим вимогам та рекомендується до захисту з високою оцінкою, а її авторки *Новак Катерині Миколаївни* може бути присвоєний освітньо-кваліфікаційний рівень «магістр» за спеціальністю 122 «Комп'ютерні науки» освітньо-професійної програми «Інформаційні управляючі системи і технології».

Науковий керівник _____


(підпис М.П.)

С.П. Ріппа, д.е.н., професор,
професор кафедри інформаційних систем в економіці
Навчально-наукового інституту "Інститут
інформаційних технологій в економіці" КНЕУ

«17» травня 2025 р.

Реферат

Кваліфікаційна магістерська робота містить 88 сторінок, 4 таблиці, 24 рисунки, список використаних джерел з 50 найменувань, 8 додатків.

«Розроблення ІУС для моніторингу та обліку робочого часу працівників»

Об'єктом дослідження кваліфікаційної магістерської роботи є системи управління робочим часом на підприємствах.

Предметом дослідження є інформаційна управляюча система для автоматизованого моніторингу та обліку робочого часу працівників.

Мета і завдання дослідження. Основною метою кваліфікаційної магістерської роботи є розробка та впровадження інформаційної управляючої системи (ІУС) для автоматизованого моніторингу та обліку робочого часу працівників підприємства. Ця система повинна фіксувати час початку та завершення роботи, обробляти дані про виконані завдання, а також надавати аналітичні звіти для керівництва.

Відповідно до поставленої мети визначені такі *завдання*:

- дослідити предметну область, виконати аналіз існуючих систем обліку робочого часу працівників;
- представити обґрунтування вибору підходів і технологій для створення системи;
- визначити структуру та представити характеристику системи для автоматизованого моніторингу робочого часу;
- описати методи та моделі в системі;
- спроектувати базу даних для системи;
- спроектувати базу знань для системи;
- описати програмне забезпечення для системи;
- описати технічне забезпечення для системи;
- представити результати реалізації системи.

Теоретична, методична та практична значущість отриманих результатів. У процесі дослідження було проаналізовано предметну область обліку робочого часу, розглянуто підходи до фіксації присутності працівників та збору даних про їхню активність за допомогою спеціалізованого програмного забезпечення. Здійснено порівняльний аналіз зручності, точності та масштабованості таких рішень. У результаті було обґрунтовано доцільність створення інформаційно-управлінської системи для автоматизованого моніторингу робочого часу, яка забезпечує фіксацію годин роботи, формування звітів та аналітики, а також підвищує прозорість та контроль у трудовому процесі. Практична значущість полягає у підвищенні продуктивності, оптимізації розрахунків заробітної плати та покращенні взаємодії між працівниками і керівництвом.

Рік виконання кваліфікаційної магістерської роботи – 2025.

Рік захисту роботи – 2025.

Ключові слова: облік робочого часу, моніторинг, працівник, звітність, автоматизація, база даних, продуктивність, контроль.

АНОТАЦІЯ

Магістерської роботи студентки 6 курсу ННІ «Інститут інформаційних технологій в економіці» Новак Катерини Миколаївни

виконану на тему:

«Розроблення ІУС для моніторингу та обліку робочого часу працівників»

Київ: Кафедра інформаційних систем в економіці, 2025 р.

Магістерська робота присвячена актуальній проблемі моніторингу та обліку часу працівників на підприємстві, яку пропонується вирішувати за допомогою інтелектуальних систем та сучасних засобів обчислювальної техніки. Робота складається з трьох розділів, які послідовно розкривають теоретичні, проектні та практичні аспекти створення інформаційної управляючої системи.

Перший розділ присвячений дослідженню та аналізу підходів до створення інформаційних управляючих систем у контексті обліку робочого часу. У цьому розділі розглядаються сучасні інформаційні системи, які застосовуються в управлінні персоналом та автоматизації обліку робочого часу. Проводиться детальний аналіз предметної області, в рамках якого досліджується, які технології і системи вже використовуються для вирішення подібних завдань. Крім того, обґрунтовується вибір найбільш оптимальних підходів і технологій для побудови запропонованої інформаційної системи, враховуючи специфіку підприємства та потреби управління персоналом.

Другий розділ містить характеристику інформаційних управляючих систем, методів та моделей, які будуть використані для вирішення завдань моніторингу і обліку робочого часу. У цьому розділі представлена структура розробленої системи, її основні елементи та функціональні можливості. Також докладно описані методи інтелектуального аналізу даних, які допоможуть забезпечити ефективну обробку великих обсягів інформації та наповнення бази знань системи. Значна увага приділяється моделям прогнозування та оптимізації управлінських процесів, які можуть бути впроваджені в системі для підвищення її ефективності.

Третій розділ носить конструктивний характер і присвячений розробці та реалізації проектних рішень. У ньому детально описується проектування бази даних та сховища даних, яке стане основою для збереження та обробки інформації про робочий час працівників. Також розглядається створення бази знань, яка включає засоби інтелектуального аналізу даних для виявлення трендів і аномалій у використанні робочого часу. У цьому розділі описується процес розробки програмного забезпечення для інформаційної управляючої системи, зокрема вибір архітектури, інструментальних засобів, тестування та впровадження програмного забезпечення. Крім того, обґрунтовується вибір технічного забезпечення, яке дозволить забезпечити стабільну і швидку роботу системи, враховуючи ресурсні вимоги.

У висновках підсумовуються результати дослідження та розробки системи, надаються рекомендації щодо впровадження інформаційної управляючої системи для моніторингу та обліку робочого часу на підприємстві. Також визначаються перспективи подальшого розвитку системи, зокрема в контексті інтеграції з іншими підсистемами управління персоналом, розширення функціональних можливостей та адаптації до нових умов і вимог підприємства.

ЗМІСТ

ВСТУП.....	3
РОЗДІЛ 1	6
ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПІДХОДІВ ДО СТВОРЕННЯ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ ПРЕДМЕТНОЇ ОБЛАСТІ	6
1.1 Аналіз існуючих інформаційних управляючих систем (предметної області).....	6
1.1.1 Дослідження предметної області	7
1.1.2 Дослідження інформаційних управляючих систем	9
1.2 Обґрунтування вибору підходів і технологій для створення інформаційної управляючої системи	11
РОЗДІЛ 2	15
ХАРАКТЕРИСТИКА ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ ТА МЕТОДИ І МОДЕЛІ	15
2.1. Структура і характеристика системи	15
2.2. Методи та моделі в інформаційних управляючих системах і технологіях.....	23
РОЗДІЛ 3	25
РОЗРОБЛЕННЯ ПРОЄКТНИХ РІШЕНЬ ТА ЇХ РЕАЛІЗАЦІЯ	25
3.1. Проєктування бази даних та/або сховища даних для інформаційної управляючої системи.....	25
3.1.1 Проєктування інфологічної (логічної) моделі БД	25
3.1.2 Обґрунтування вибору СКБД та проєктування фізичної моделі БД в середовищі вибраної СКБД.....	29
3.1.3 Контрольний приклад та обрахунок прогнозних обсягів БД	30
3.2. Проєктування бази знань інформаційної управляючої системи та засоби інтелектуального аналізу даних	32
3.2.1 Реалізація інтелектуального аналізу великих даних з метою наповнення бази знань	33
3.3 Програмне забезпечення.....	35
3.3.1 Проєктування програмного забезпечення.....	35
3.3.2 Вибір архітектури програмного забезпечення.....	37
3.3.3 Інструментальні засоби розробки для створення прикладного програмного забезпечення інформаційних управляючих систем.....	38
3.3.4 Тестування програмного забезпечення.....	41
3.3.5 Керівництво (інструкція) користувача	42
3.4 Технічне забезпечення	44

3.4.1 Обґрунтування вибору технічного забезпечення	45
3.4.2 Ресурсні вимоги до технічного забезпечення	46
3.5 «Реалізація інформаційної управляючої системи»	48
ВИСНОВКИ	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	59
ДОДАТКИ.....	63
Додаток А	63
Додаток Б	64
Додаток В	71
Додаток Г	75
Додаток Д	79
Додаток Е	80
Додаток Є	81
Додаток Ж	84

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IUC – інформаційна управляюча система;

AI – штучний інтелект;

ML – машинне навчання;

IoT – Інтернет речей;

AR, VR – доповнена та віртуальна реальність;

UML – Unified Modeling Language;

ETL – Extract, Transform, Load;

ELT – Extract, Load, Transform;

СКБД – систему керування базами даних;

SQL –Structured Query Language;

PCA – Principal Component Analysis;

JWT – JSON Web Token.

ВСТУП

Актуальність теми дослідження. З початком пандемії COVID-19 та зміною умов праці багатьох компаній, значно зріс попит на віддалену роботу. Це змусило роботодавців переглянути традиційні методи управління персоналом, особливо щодо моніторингу робочого часу та контролю продуктивності. В Україні ситуація загострилася через війну, що призвело до масового переходу на гібридні або повністю віддалені моделі роботи. У таких умовах традиційні системи обліку робочого часу в офісному середовищі виявилися неефективними для нових реалій. Тому виникла необхідність створення інформаційної управляючої системи (ІУС) для автоматизованого моніторингу та обліку робочого часу працівників, що працюють у віддаленому або гібридному режимі. Ця система дозволить не тільки підвищити продуктивність, а й ефективно керувати розподілом ресурсів, зменшуючи можливі витрати часу і контроль за виконанням завдань.

Аналіз останніх досліджень і публікацій. Різні дослідники в своїх роботах звертали увагу на проблему моніторингу робочого часу та оптимізації управлінських процесів у нових умовах. У дослідженні Yash Rathі (2021) під назвою «Remote Work and Employee Performance: A Study of the Challenges and Benefits» [1] аналізується вплив віддаленої роботи на продуктивність працівників, зокрема в контексті моніторингу робочого часу та управління персоналом. Автор підкреслює, що дистанційна робота вимагає використання сучасних технологій для обліку часу та продуктивності, що дозволить краще контролювати процеси та покращувати продуктивність персоналу. Однак, проблема адаптації цих технологій до специфіки віддаленої роботи досі залишається актуальним викликом.

Мета та завдання дослідження. Метою дослідження є розробка і впровадження інформаційної управляючої системи для автоматизованого моніторингу та обліку робочого часу працівників підприємства. Для досягнення цієї мети необхідно вирішити такі завдання:

- вивчити та проаналізувати сучасні методи та інструменти моніторингу робочого часу;
- розробити архітектуру інформаційної управляючої системи, що буде адаптована під специфіку підприємства;
- створити базу даних для зберігання та обробки інформації про робочий час працівників;
- впровадити автоматизовані механізми обліку робочого часу та оцінки продуктивності працівників;
- провести тестування системи на реальних прикладах підприємств;
- надати рекомендації щодо подальшого вдосконалення системи та можливостей її масштабування.

Об'єкт і предмет дослідження. Об'єктом дослідження є системи управління робочим часом на підприємствах, що працюють у віддаленому чи гібридному форматі. Предметом дослідження є інформаційна управляюча система для автоматизованого моніторингу та обліку робочого часу працівників.

Методи дослідження. У роботі застосовувалися такі методи:

- аналіз літератури та огляд існуючих рішень для виявлення недоліків поточних систем моніторингу робочого часу;
- математичне моделювання для оптимізації облікових процесів і розподілу робочих навантажень;
- статистичний аналіз для оцінки продуктивності працівників;
- методи системного аналізу для виявлення взаємозв'язків між елементами системи з метою підвищення її ефективності.

Інформаційна база дослідження. Дослідження базується на наукових публікаціях вітчизняних та зарубіжних авторів, статистичних даних офіційних сайтів і звітах компаній, які впроваджують подібні системи. Аналіз сучасних методів вирішення проблеми здійснювався на основі технічної документації та оглядів інноваційних систем.

Наукова новизна і практична значущість. Наукова новизна дослідження полягає в розробці нової інформаційної управляючої системи, яка дозволяє ефективно контролювати робочий час у віддаленому та гібридному режимах. Практична значущість полягає у можливості впровадження системи в реальні підприємства, що дозволить покращити управління робочими процесами і підвищити загальну продуктивність.

РОЗДІЛ 1

ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПІДХОДІВ ДО СТВОРЕННЯ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз існуючих інформаційних управляючих систем (предметної області)

Існує велика кількість програмних рішень, які дозволяють проводити моніторинг та облік робочого часу працівників. Одним із найпопулярніших застосунків для обліку робочого часу є UpWork [2]. Платформа пропонує надійний інструмент для відстеження часу роботи роблячи регулярні знімки екрану та дозволяє клієнтам стежити за прогресом виконання завдань. Перевагою є можливість виставляти точні рахунки клієнтам на основі погодинної ставки, узгодженої між замовником і виконавцем. Крім того, система може використовувати веб-камеру для більшого контролю за роботою працівників. Однак одним з основних недоліків UpWork є висока комісія платформи, що зменшується тільки після досягнення певного обсягу роботи, що може зайняти значний час.

У даній роботі був проведений порівняльний аналіз найбільш популярних рішень для обліку робочого часу, а саме:

- UpWork;
- Toggl Track [3];
- Tick [4];
- Timesheels [5];
- Office time [6];
- Timely [7];
- Hours [8];
- Kickidler [9];
- Croco Time [10];
- Monitask [11].

Результат аналізу продемонстровано в таблиці 1.1. На основі цього аналізу було виявлено недоліки, які будуть враховані при розробці власного

продукту, а саме потреба в простому та зрозумілому інтерфейсі, який не вимагатиме багато часу на налаштування, а також гнучка цінова політика, орієнтована на різні типи організацій.

Таблиця 1.1 – Порівняльний аналіз існуючих додатків

Сервіси	Безкоштовний тариф	Звітність та аналітика	Моніторинг сайтів та програм	Скріншоти робочого екрану	Звіт заробітної плати	Виставлення рахунків
UpWork	-	+	+	+	+	+
Toggl Track	+	+	-	-	-	+
Tick	+	+	-	-	+	+
Timesheets	+	+	-	+	+	-
Office Time	-	+	-	-	+	-
Timely	-	+	-	-	+	-
Hours	+	+	-	-	+	+
Kickidler	+	+	+	+	-	-
Croco Time	-	+	+	+	-	-
Monitask	+	+	+	+	-	-

Джерело даних: авторська розробка.

1.1.1 Дослідження предметної області

Розробка ІУС для моніторингу та обліку робочого часу працівників залишається актуальним питанням для сучасних організацій та підприємств. Системи такого типу дозволяють автоматизувати процеси контролю, підвищити ефективність управління персоналом, а також забезпечити прозорість та точність обліку робочого часу.

Передусім варто визначити, що *моніторинг робочого часу* — це процес фіксації, спостереження та аналізу діяльності працівників упродовж робочого дня з метою забезпечення прозорості, підвищення дисципліни та ефективності роботи. На відміну від звичайного табелювання, моніторинг включає не лише час входу/виходу з системи, а й дії працівника в процесі виконання завдань (наприклад, активність у певних програмах, знімки екрана тощо) [12].

Облік робочого часу — це систематизація та збереження даних про фактичну присутність і зайнятість працівників, на основі яких здійснюється

нарахування заробітної плати, оцінка продуктивності, планування навантаження та інші управлінські рішення. У контексті автоматизованої ІУС, цей облік реалізується за допомогою інтеграції з базою даних, де зберігається інформація про працівників, їхні робочі сесії, виконані завдання та звіти [13].

Характеристика предметної області включає аналіз основних завдань, що виникають при управлінні робочим часом та охоплює планування робочих змін, реєстрацію часу приходу та виходу працівників, контроль виконання завдань протягом робочого дня, облік понаднормових годин, а також надання звітності для керівництва. Такий підхід є важливим як для великих компаній, де потрібен детальний контроль за графіками багатьох співробітників, так і для малих підприємств, де точність обліку впливає на оптимізацію витрат.

Опис рішень у предметній області полягає в оперативному управлінні процесами, пов'язаними з моніторингом робочого часу. Сюди входить стратегічне планування робочих змін і відпусток, адміністративне управління трудовими ресурсами та оперативний контроль над відхиленнями від запланованих графіків. Також можливі альтернативні підходи, наприклад, впровадження гнучких робочих графіків або дистанційної роботи, що змінює методи відстеження часу. Періодичність прийняття таких рішень варіюється від щоденного контролю до планування на місяці вперед.

Особи, що приймають рішення в цій сфері включають менеджерів середньої та вищої ланки, відділи кадрів, а також ІТ-спеціалістів, що відповідають за впровадження і підтримку системи. У разі групового прийняття рішень важливо узгоджувати інтереси різних відділів, щоб система відповідала потребам як адміністрації, так і працівників.

Фактори, що впливають на середовище прийняття рішень, включають законодавчі вимоги щодо обліку робочого часу, корпоративні правила, економічні умови та розвиток технологій. Наприклад, зміни в трудовому законодавстві можуть вимагати адаптації ІУС до нових вимог.

Взаємозв'язки рішень виникають на етапах розробки та впровадження системи щодо методів обліку робочого часу, таких як використання

біометричних систем або RFID-карт, впливають на вибір технологій і майбутні модифікації системи.

Інформація, необхідна для прийняття рішень, включає дані про робочі графіки, відпустки, лікарняні, понаднормові години тощо. Важливим є наявність точних даних з усіх джерел – систем доступу, внутрішніх звітів тощо.

Знання, необхідні для прийняття рішень, можуть бути як декларативними (наприклад, знання законодавчих норм), так і процедурними (надання доступу до даних про працівників, автоматичне формування звітів).

Висновки щодо сутності ситуацій прийняття рішень можуть бути визначеними (детермінованими) або мати елементи невизначеності (за умов зміни законодавства або непередбачених відсутностей працівників).

1.1.2 Дослідження інформаційних управляючих систем

Інформаційні управляючі системи відіграють важливу роль у різних сферах економіки та техніки, забезпечуючи ефективний збір, обробку і використання даних для прийняття управлінських рішень. Ці системи постійно розвиваються разом із впровадженням нових технологій, таких як штучний інтелект (AI) [14], машинне навчання (ML) [15], великі дані (Big Data) [16], Інтернет речей (IoT) [17], доповнена та віртуальна реальність (AR, VR [18]), що забезпечує нові можливості для їхньої інтеграції у різні процеси управління та виробництва.

Одним із основних аспектів, що потребує аналізу під час дослідження існуючих ІУС, є повнота функціональних можливостей. Сучасні системи охоплюють широкий спектр функцій, включаючи моніторинг, аналіз даних, автоматизацію робочих процесів, управління ресурсами, контроль за виконанням завдань та забезпечення безпеки інформації. У різних галузях, наприклад, промисловість, логістика, охорона здоров'я, освіта та фінанси, ІУС здатні адаптуватися під специфічні потреби, надаючи інструменти для вирішення завдань різної складності.

Концептуальна схема управління ІУС передбачає розподіл системи на підсистеми, кожна з яких виконує конкретні функції, необхідні для забезпечення ефективного управління. Ця схема часто включає підсистеми збору, зберігання, обробки та аналізу даних, а також підсистеми для взаємодії з користувачами та зовнішніми системами. Розподіл завдань між підсистемами дозволяє ІУС працювати більш ефективно та гнучко адаптуватися до змін у зовнішньому середовищі чи вимогах користувачів.

ІУС орієнтовані на підтримку процесів прийняття рішень у ситуаціях, які потребують аналізу великих обсягів даних або врахування різних факторів і критеріїв. Ці системи дозволяють знизити рівень невизначеності та допомагають керівникам або операторам приймати зважені рішення на основі актуальних даних. Програмні компоненти ІУС можуть автоматизувати рутинні операції, забезпечуючи швидке прийняття рішень у таких сферах, як контроль виробничих процесів, управління запасами, оптимізація логістичних операцій тощо.

Сучасні ІУС передбачають використання потужних інструментів для управління базами даних, знань та програмним забезпеченням. Зазвичай у таких системах використовується архітектура клієнт-сервер, яка дозволяє забезпечити централізоване управління даними та доступ до них з різних пристроїв. ІУС також можуть включати сховища даних, що дозволяють зберігати великі обсяги структурованих та неструктурованих даних, а також використовувати системи управління знаннями для аналізу та інтерпретації інформації з різних джерел.

Сучасні тенденції в розробці ІУС спрямовані на впровадження інноваційних інформаційних технологій, таких як AI, ML, Big Data, IoT, AR, VR. Ці проривні технології дозволяють автоматизувати процеси, зменшити вплив людського фактора на прийняття рішень і забезпечити нові можливості для аналізу даних та прогнозування результатів. Впровадження таких технологій сприяє підвищенню продуктивності, зменшенню витрат та підвищенню точності в управлінських процесах.

Спеціалізовані ІУС часто реалізуються на базі платформ Arduino [19], ESP8266 [20], Orange Pi [21] та Raspberry Pi [22], що дозволяє створювати

високоєфективні системи для вирішення вузькоспеціалізованих завдань. Такі системи використовуються для моніторингу параметрів довкілля, управління обладнанням та іншими процесами, які потребують автоматизації. Перевага цих платформ полягає у їхній доступності, гнучкості налаштувань та можливості інтеграції з іншими системами.

Розробка ІУС часто включає створення інтелектуальних компонентів, таких як модулі штучного інтелекту для прогнозування або прийняття рішень на основі аналізу даних. Ці компоненти здатні адаптуватися до змін у зовнішньому середовищі та підвищувати ефективність роботи системи за рахунок гнучкого підходу до вирішення поставлених завдань.

ІУС також відіграють важливу роль у наукових дослідженнях, забезпечуючи ефективні інструменти для збору, обробки та аналізу наукових даних. Використання таких систем дозволяє отримувати наукові результати у вигляді звітів, графіків, статистичних даних або інших візуалізацій, що полегшує їхнє інтерпретування та використання.

Користувачі ІУС відзначають численні переваги від застосування цих систем, зокрема, підвищення продуктивності, автоматизацію рутинних процесів, зниження витрат та зменшення кількості помилок, пов'язаних з людським фактором. ІУС дозволяють швидше реагувати на зміни у середовищі та приймати обґрунтовані управлінські рішення.

1.2 Обґрунтування вибору підходів і технологій для створення інформаційної управляючої системи

У процесі створення ІУС для моніторингу та обліку робочого часу критично важливо визначити технологічні рішення, які дозволять реалізувати систему з високим рівнем надійності та зручності в користуванні. Такий вибір базується не лише на технічних можливостях інструментів, а й на вимогах, що сформовані на основі аналізу потреб замовника, користувачів та середовища функціонування системи.

На початковому етапі розробки були сформульовані ключові вимоги до майбутньої ІУС:

- забезпечення збору, фіксації та обробки інформації про робочий час працівників у режимі, наближеному до реального часу;
- можливість ручної корекції даних (за потреби адміністратора або менеджера);
- формування звітів про ефективність, відпрацьовані години, порушення робочого режиму тощо;
- простий, інтуїтивно зрозумілий веб-інтерфейс, доступний з різних пристроїв;
- захист персональних даних та обмеження доступу відповідно до ролі користувача (робітник, керівник, адміністратор);
- потенціал до інтеграції з іншими інформаційними системами підприємства, зокрема обліковими або ERP-рішеннями.

Проектування ІУС базується на створенні концептуальної моделі, яка відображає основні сутності (наприклад: Користувач, Робочий день, Проект, Завдання, Звіт), їх взаємозв'язки та поведінку в межах системи.

Для формалізації архітектури застосовано об'єктно-орієнтований підхід, де кожна ключова сутність відображається як окремий об'єкт з властивостями (атрибутами) та методами. Це дозволяє моделювати взаємодію користувачів із системою, відстежувати зміни в даних, будувати логіку обробки та взаємозв'язків.

Для проектування системи обрано об'єктно-орієнтовану методологію, що дозволяє:

- структуровано подати всі компоненти системи через UML-діаграми (класи, об'єкти, послідовності);
- створити модульну архітектуру, в якій легко реалізовувати та тестувати окремі функції;
- спростити підтримку, масштабування та впровадження нових функцій у майбутньому;
- забезпечити повторне використання коду та підвищити гнучкість системи.

Для візуалізації архітектури та логіки системи використовувалися наступні типи UML-діаграм:

- діаграма варіантів використання (use case) — для відображення сценаріїв взаємодії користувачів із системою;
- діаграма класів — для моделювання логічної структури сутностей;
- діаграма активностей — для опису бізнес-процесів (наприклад, створення звіту або реєстрація часу);
- діаграма послідовностей — для відображення порядку виконання дій між об'єктами.

Використано онлайн-редактор draw.i) для побудови UML-діаграм. Його переваги — безкоштовність, простота у використанні, інтеграція з Google Drive та підтримка експорту у PDF/SVG.

Для формалізації логіки процесів та їх оптимізації розглядалися інструменти ARIS Express і BPWin. Вони дозволяють наочно змодельовати потоки даних, управлінські цикли, прийняття рішень — що особливо важливо при інтеграції з внутрішніми регламентами підприємства.

В якості СУБД обрано Microsoft SQL Server, оскільки він забезпечує високу продуктивність обробки запитів, підтримку складних реляційних зв'язків між таблицями, інструменти для створення резервних копій, реплікації, контролю доступу, зручну роботу з представленнями (views), тригерами, збереженими процедурами, можливість масштабування при зростанні обсягів даних [23].

Для проєктування бази даних та управління нею використовувались SQL Server Management Studio (SSMS) [24] і ER-діаграми [25].

Для реалізації клієнтської логіки та побудови веб-інтерфейсу було обрано мову JavaScript як основну мову програмування, що забезпечує високу інтерактивність, динамічну взаємодію з елементами сторінки та широкі можливості для розширення [26].

Інтерфейс системи реалізовано за допомогою HTML5 [27], CSS3 [28] та чистого JavaScript, без залучення важких фреймворків, що дозволило досягти

гнучкості та легкості застосунку. Для побудови графіків і візуалізації звітів використовувалась бібліотека Chart.js [29].

Систему стилізовано з використанням Bootstrap 5, що забезпечило адаптивність, модульність та привабливе оформлення інтерфейсу. Уся логіка взаємодії користувача із системою (авторизація, перемикання між сторінками, обробка форм, побудова діаграм) реалізована засобами JavaScript на стороні клієнта [30].

РОЗДІЛ 2

ХАРАКТЕРИСТИКА ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ ТА МЕТОДИ І МОДЕЛІ

2.1. Структура і характеристика системи

Структуру розробленої системи подано на рисунку 2.1. Структура містить дві основні частини: клієнтську та серверну.



Рисунок 2.1 – Схема архітектури інформаційної системи

Серверна частина системи одержує дані і запити від клієнта, обробляє ці запити і відправляє результат обробки у вигляді відповіді на клієнтську частину системи. Однією з важливих функцій серверної частини даного застосунку це взаємодія з базою даних. Клієнтська частина системи призначена для взаємодії з користувачем, забезпечуючи зручний інтерфейс для огляду, зміни та модифікації даних в базі даних.

Функціональна структура інформаційних управляючих систем подавана в одній з нотації UML. Моделювання інформаційної підсистеми є важливим етапом розробки програмного забезпечення, оскільки воно дозволяє зрозуміти взаємодію між різними компонентами системи та залежностями між ними. Мета моделювання полягає в створенні зрозумілого та корисного опису системи, який згодом можна використати для подальшої розробки.

Для того, щоб точно передати всі можливості системи, було створено діаграми прецедентів, на яких описано можливі функції застосунку.

Суть діаграми прецедентів в SysML полягає в тому, що проєктована система подається у вигляді множини блоків (сутностей або компонентів), які взаємодіють між собою за допомогою варіантів використання. Варіант

використання (use case) використовується для опису послуг, які система надає користувачам або іншим блокам. Кожен варіант використання визначає деякий набір дій, які виконує система під час взаємодії з користувачами або іншими блоками.

Приклад Use Case діаграми наведено на рисунку 2.2. Дана діаграма відображає можливості користувача та замовника.

З неї можна побачити, що користувач (працівник) має наступні можливості:

- реєстрація в системі;
- авторизація (вхід до системи);
- створення нового завдання або проєкту;
- додавання коментарів до завдань;
- перегляд статусів завдань;
- завантаження або перегляд скріншотів, пов'язаних із завданнями.

Замовник:

- перегляд створених замовником завдань;
- перегляд статусу виконання завдань;
- ознайомлення зі скріншотами, що підтверджують виконання робіт;
- перегляд залишених коментарів до завдань;
- отримання звітів про витрачений час на виконання завдань.

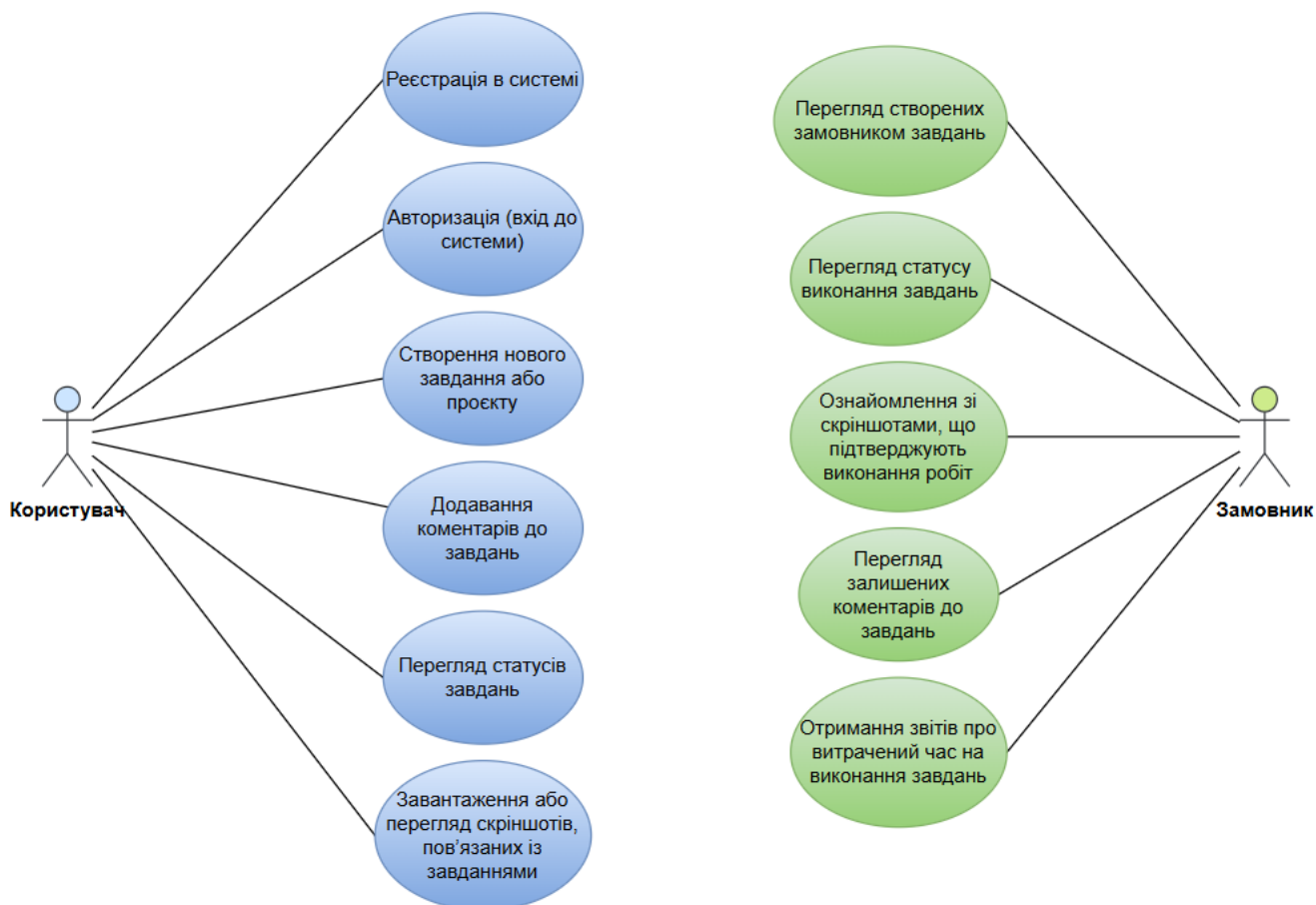


Рисунок 2.2 – Use Case діаграма

Джерело даних: авторська розробка.

На наведених нижче Sequence Diagrams відображається послідовність дій для кожного Use Case:

- 1) Реєстрація / авторизація користувача в системі (рис. 2.3).

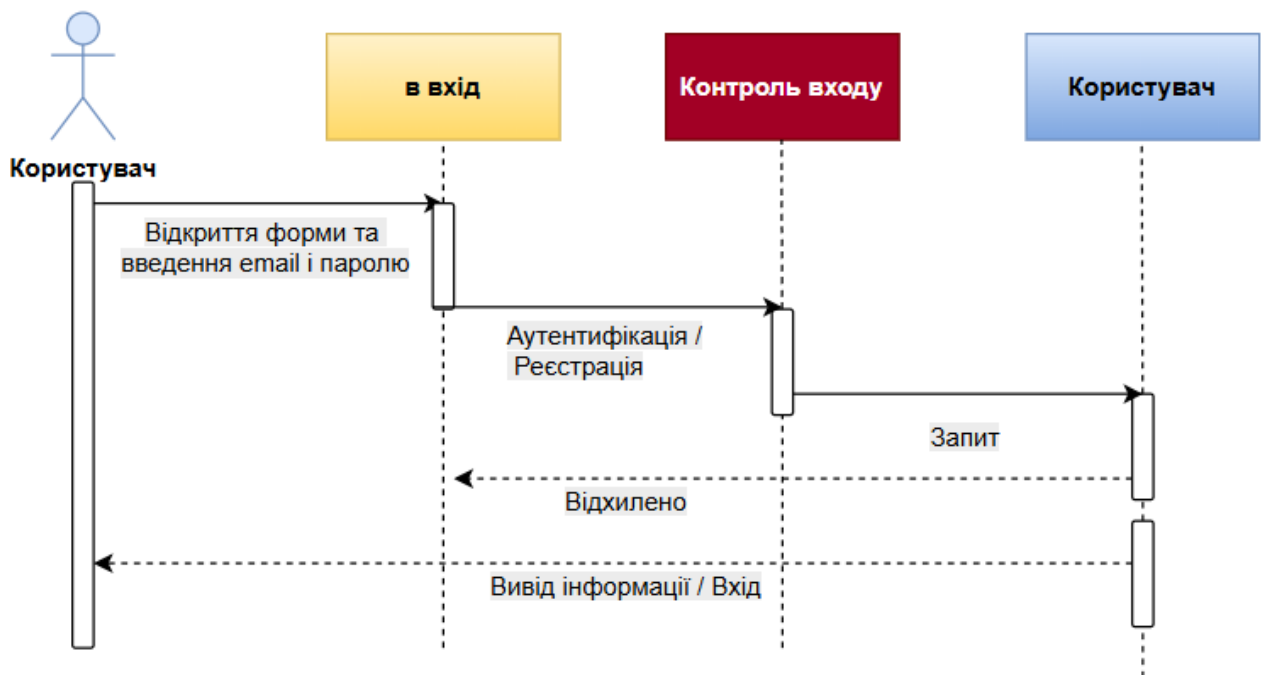


Рисунок 2.3 – Реєстрація / авторизація користувача в системі

Джерело даних: авторська розробка.

Користувач відкриває застосунок, на головній сторінці знаходить форму, де він може ввести необхідну інформацію, таку як логін, пароль. Якщо користувач ще не має облікового запису, він має можливість зареєструватися як новий користувач. На веб-сайті надається форма реєстрації, де користувач повинен ввести свої особисті дані, такі як ім'я, прізвище, електронна пошта, номер телефону тощо. Після введення цих даних користувача додають до бази даних системи як нового користувача. Після введення інформації в форму користувач надсилає дані на сервер, де вони перевіряються для аутентифікації. Якщо введені дані вірні, користувачу надається доступ до особистого облікового запису в системі.

Якщо користувач вже зареєстрований, то він вводить свій логін та пароль. Після введення логіну та пароля користувача, система перевіряє ці дані з інформацією, збереженою у базі даних системи. Вона перевіряє, чи наявні такі облікові дані у базі даних. Якщо облікові дані користувача збігаються з даними у базі даних, система дозволяє вхід в обліковий запис користувача. Користувач перенаправляється на особисту сторінку. Якщо введені облікові дані неправильні

або не знайдені у базі даних, система виводить відповідне повідомлення про помилку. Користувач може отримати інформацію про неправильний логін або пароль та можливість повторного введення правильних облікових даних.

2) Облік робочого часу користувачем (працівником) (рис. 2.4).



Рисунок 2.4 – Облік робочого часу користувачем (працівником)

Джерело даних: авторська розробка.

Користувач заходить в систему і вводить опис завдання, тривалість виконання, дату, час та статус (з випадаючого списку). Сервер отримує запит користувача і виконує запит до бази даних щодо прав доступу. Відбувається обробка запиту, створюється запис в таблиці «Завдання» та пов'язується зі статусом через таблицю «Статус». Після відбувається виведення інформації про успішне збереження даних.

3) Завантаження скріншотів (рис. 2.5).

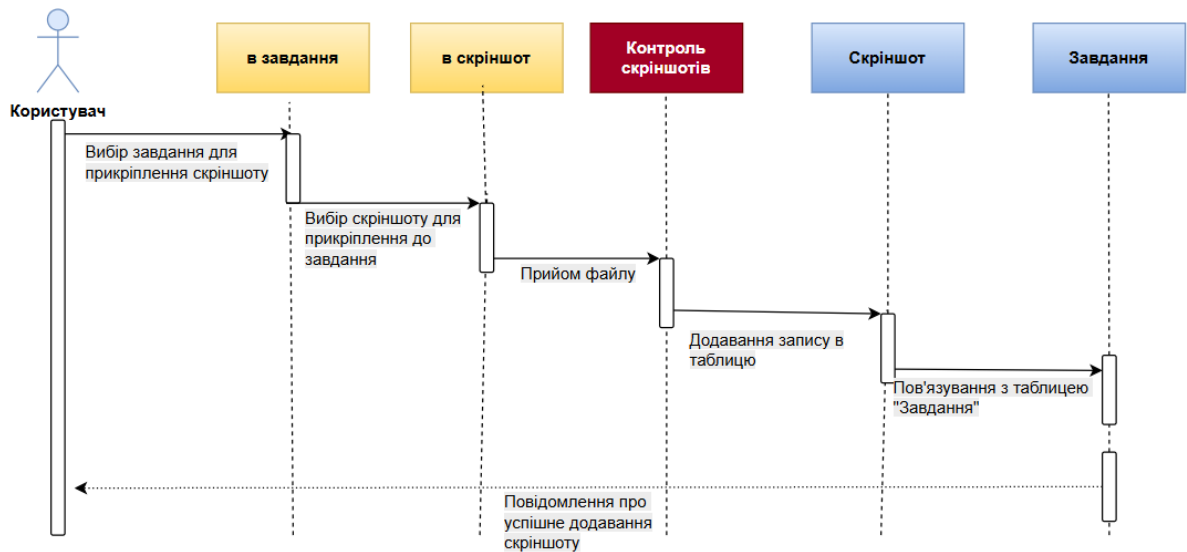


Рисунок 2.5 – Завантаження скріншотів

Джерело даних: авторська розробка.

Користувач обирає завдання з переліку завдань, які він виконує, щоб прикріпити скріншот з відображенням прогресу виконання задачі. Скріншот додається до таблиці, щоб далі його міг переглянути замовник (роботодавець). Користувачеві приходить повідомлення про успішне додавання скріншоту.

4) Створення завдання замовником (рис. 2.6).



Рисунок 2.6 – Створення завдання замовником

Джерело даних: авторська розробка.

Замовник надсилає запит до системи, щоб додати завдання для працівника (користувача). Сервер отримує запит від замовника і надсилає запит до бази даних, щоб обробити запит та створити запис в таблиці «Завдання». База даних містить дані про опис завдання, дату, час та статус.

5) Формування звіту замовником

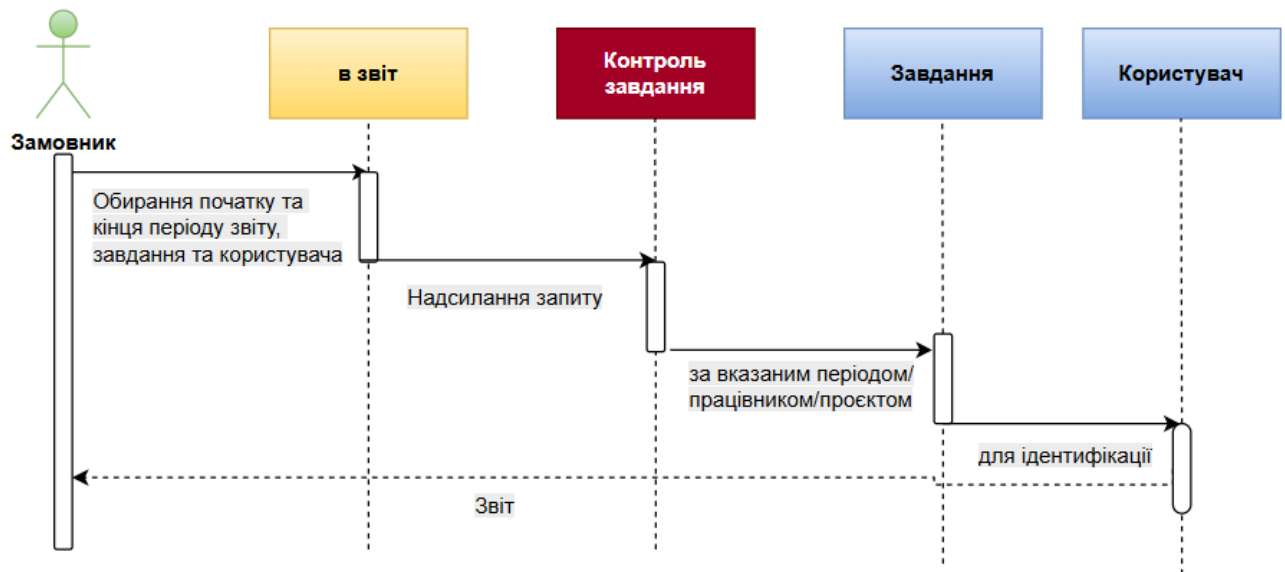


Рисунок 2.7 – Формування звіту замовником

Замовник обирає період початку та кінця звіту, завдання та користувача. Далі надсилається запит на отримання даних з таблиці завдання та користувач для ідентифікації. Формується звіт.

Для зображення структури системи було розроблено діаграму класів. Діаграма класів дозволяє формалізувати логічну модель програми на рівні структурних елементів системи, тобто класів. Вона надає статичне представлення про структуру програми – з яких класів вона складається, які зв'язки між цими класами та з чого складається кожен клас [30].

Нижче на рисунку 2.9 наведена діаграма класів.

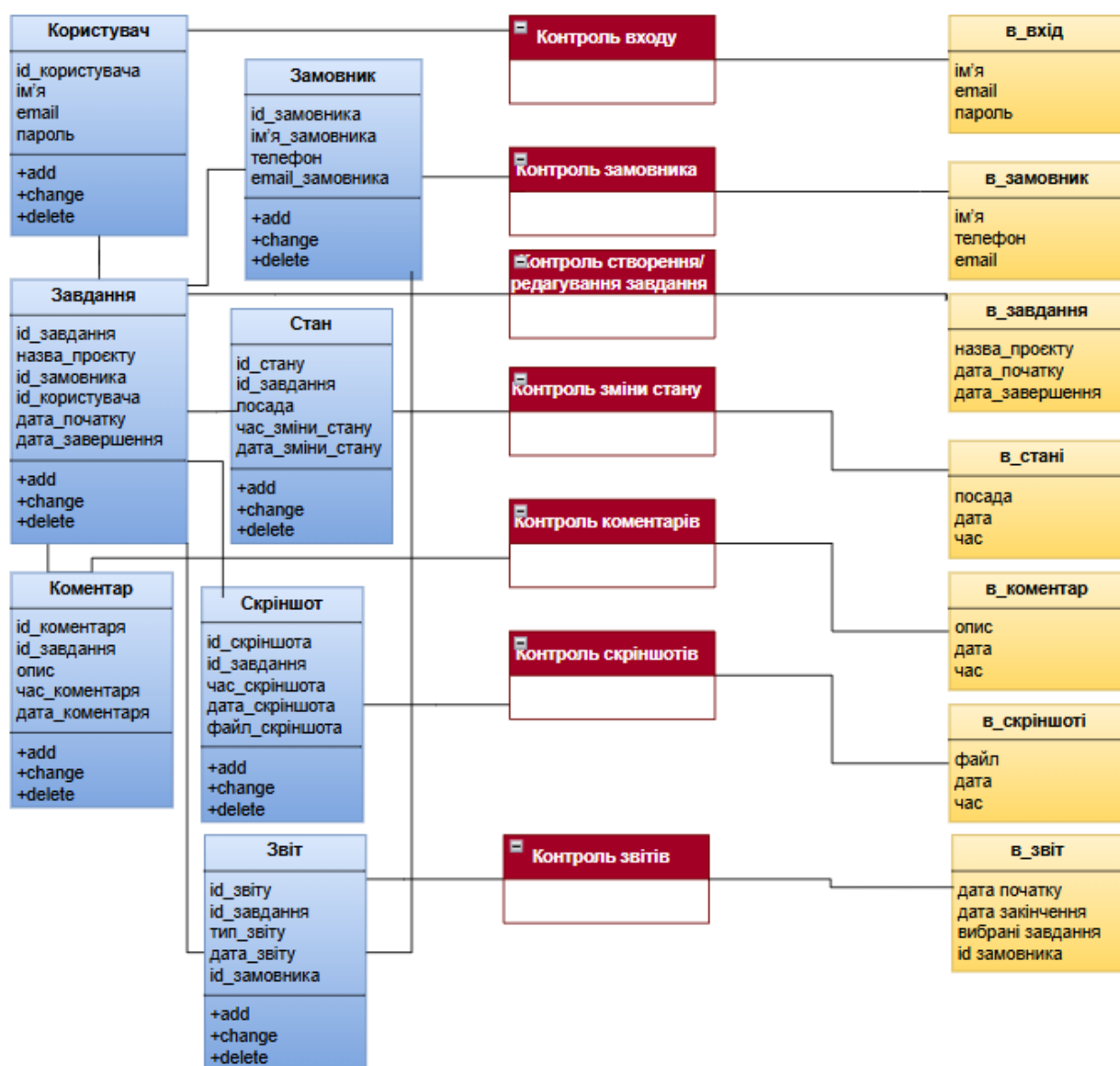


Рисунок 2.9 – Діаграма класів

Джерело даних: авторська розробка.

Блакитні об'єкти – це класи пов'язані безпосередньо з процесом моніторингу та обліку робочого часу працівників.

Червоні – клас перевірки інформації.

Жовті – вхідні дані, що вносяться користувачем.

2.2. Методи та моделі в інформаційних управляючих системах і технологіях

Методи та моделі в інформаційних управляючих системах є основою для проектування, реалізації та обслуговування таких систем, що включають обробку, зберігання та аналіз даних, а також забезпечують правильність і ефективність роботи системи. Методи в ІУС представляють собою підходи та техніки, що використовуються для вирішення конкретних завдань, які виникають під час обробки та передачі інформації, управління даними та забезпечення безпеки.

Одним із основних методів обробки даних є ETL (Extract, Transform, Load), який передбачає екстракцію, перетворення та завантаження даних із різних джерел у бази даних або сховища даних. Цей метод є важливим для інтеграції даних з різних джерел і підготовки їх до аналізу. Аналогічним є метод ELT (Extract, Load, Transform), коли дані спочатку завантажуються в базу даних, а потім відбувається їх трансформація в необхідний формат. Важливим аспектом є також регуляризація або нормалізація даних, що дозволяє привести їх до єдиного стандарту для подальшої обробки та аналізу [31].

Для обробки великих обсягів даних використовуються методи машинного навчання та аналізу даних. Вони дозволяють прогнозувати майбутні події, виявляти закономірності та залежності в масиві даних, що забезпечує більш точне управління ресурсами та прийняття рішень. Водночас важливою складовою є методи забезпечення безпеки інформаційної системи, серед яких шифрування даних для їх захисту від несанкціонованого доступу, а також методи аутентифікації та авторизації для перевірки особи користувача та надання доступу до певних ресурсів.

Моделі в ІУС допомагають описати структури, зв'язки та поведінку елементів системи. Однією з основних є реляційна модель даних, де дані організовуються у таблиці з чітко визначеними зв'язками між ними. Вона є основною для багатьох сучасних баз даних, оскільки дозволяє зберігати та обробляти структуровану інформацію з високою ефективністю. Іншою

популярною моделлю є об'єктно-орієнтована модель, в якій дані представлені у вигляді об'єктів, що включають як властивості, так і методи для взаємодії з ними. Це дозволяє створювати більш гнучкі і масштабовані інформаційні системи.

Також важливим є використання моделей для проектування програмного забезпечення. Наприклад, архітектурний патерн MVC (Model-View-Controller) дозволяє розділити систему на три основні компоненти: модель, яка відповідає за зберігання та обробку даних; перегляд, який відображає дані користувачеві; та контролер, що обробляє запити користувача і координує роботу моделі та перегляду. Це забезпечує чистоту коду та зручність у роботі з великими проектами [32].

Моделі безпеки відіграють важливу роль у захисті даних і запобіганні несанкціонованого доступу до ресурсів системи. Наприклад, модель багаторівневого доступу визначає, які користувачі мають доступ до яких даних, і це може бути реалізовано через ролі користувачів. Моделі резервування і масштабування забезпечують високу доступність системи, дозволяючи їй справлятися з великими навантаженнями і забезпечувати безперервну роботу навіть у разі збою однієї з частин системи.

Методи та моделі в ІУС дозволяють підвищити ефективність і безпеку системи, а також забезпечити її здатність до масштабування та адаптації до змінюваних умов. Вибір правильних методів і моделей забезпечує належну роботу інформаційних систем, що відповідають потребам користувачів і вимогам бізнесу.

РОЗДІЛ 3

РОЗРОБЛЕННЯ ПРОЄКТНИХ РІШЕНЬ ТА ЇХ РЕАЛІЗАЦІЯ

3.1. Проєктування бази даних та/або сховища даних для інформаційної управляючої системи

Основним завданням бази даних для інформаційної системи моніторингу і контролю робочого часу працівників є зберігання інформації про користувачів, проєкти, задачі, статуси, клієнтів і коментарі, що необхідні для правильного контролю часу, виставлення рахунків та генерації звітів [33].

3.1.1 Проєктування інфологічної (логічної) моделі БД

Проєктування інфологічної моделі бази даних починається з визначення основних сутностей та їх взаємозв'язків [34]. У системі обліку часу можна виділити наступні основні сутності:

Користувачі (User)

- Ідентифікатор користувача (UserID) — унікальний ідентифікатор кожного користувача.
- Ім'я користувача (Username) — ім'я або псевдонім, за яким користувач зареєстрований у системі.
- Електронна пошта (Email) — контактний email, що використовується для комунікації та входу в систему.
- Пароль (Password) — зашифрований пароль для входу в систему.
- Ідентифікатор задачі (TaskID) — кожен користувач може бути прив'язаний до конкретних задач, що він виконує.

Задачі (Task)

- Ідентифікатор задачі (TaskID) — унікальний ідентифікатор кожної задачі.
- Назва проєкту (ProjectName) — назва проєкту, до якого належить задача.
- Ідентифікатор замовника (CustomerID) — визначає, для якого клієнта виконується задача.

- Дата початку (Start) — дата початку виконання задачі.
- Дата завершення (End) — дата завершення задачі або прогнозована

дата її виконання.

Статуси (Status)

- Ідентифікатор статусу (StatusID) — унікальний ідентифікатор кожного статусу.
- Позиція (Position) — опис етапу або статусу задачі.
- Час зміни статусу (TimeStatus) — час, коли статус був оновлений.
- Дата статусу (DateStatus) — дата, коли статус задачі був змінений.
- Ідентифікатор коментаря (CommentID) — пов'язаний з коментарем для уточнення статусу задачі.

Коментарі (Comment)

- Ідентифікатор коментаря (CommentID) — унікальний ідентифікатор кожного коментаря.
- Опис (Description) — текстовий опис або коментар до задачі.
- Час коментаря (TimeComment) — час, коли був залишений коментар.
- Дата коментаря (DateComment) — дата коментаря.

Замовники (Customer)

- Ідентифікатор замовника (CustomerID) — унікальний ідентифікатор кожного замовника.
- Ім'я замовника (CustomerName) — ім'я компанії або фізичної особи, для якої виконується проект.
- Контактний телефон (Phone) — телефон замовника.
- Електронна пошта замовника (CustomerEmail) — email для зв'язку з замовником.

Скріншоти (Screenshots)

- Ідентифікатор скріншота (ScreenshotID) — унікальний ідентифікатор кожного скріншота.
- Час скріншота (TimeScreenshot) — час, коли був зроблений скріншот під час виконання задачі.

- Дата скріншота (DateScreenshot) — дата створення скріншота.
- Файл скріншота (FileScreenshot) — шлях до файлу або бінарні дані скріншота, що зберігаються в системі.

Зв'язки між сутностями

1. Користувачі (User) → Задачі (Task):

- Зв'язок: один користувач може бути прив'язаний до багатьох задач, але кожна задача належить лише одному користувачу.
- Тип зв'язку: один-до-багатьох
- Опис: один користувач може працювати над кількома задачами, але кожна задача призначена конкретному користувачу.

2. Задачі (Task) → Статуси (Status):

- Зв'язок: одна задача може мати багато статусів, але кожен статус належить до однієї конкретної задачі.
- Тип зв'язку: один-до-багатьох
- Опис: протягом виконання задачі її статус може змінюватися, тому кожна задача може мати кілька записів про статуси, що відображають її прогрес.

3. Задачі (Task) → Коментарі (Comment):

- Зв'язок: одна задача може мати багато коментарів, але кожен коментар прив'язаний до конкретної задачі.
- Тип зв'язку: один-до-багатьох
- Опис: до кожної задачі можуть бути додані коментарі для уточнення чи надання додаткової інформації.

4. Замовники (Customer) → Задачі (Task):

- Зв'язок: один замовник може мати багато задач, але кожна задача прив'язана лише до одного замовника.
- Тип зв'язку: один-до-багатьох
- Опис: один замовник може замовляти кілька задач або проєктів, але кожен проєкт належить до конкретного замовника.

5. Задачі (Task) → Скріншоти (Screenshots):

- Зв'язок: одна задача може мати багато скріншотів, але кожен скріншот пов'язаний з конкретною задачею.
- Тип зв'язку: один-до-багатьох
- Опис: під час виконання задачі може бути зроблено кілька скріншотів, що допомагають моніторити прогрес і активність працівника.

На рисунку 3.1 зображено інфологічну модель бази даних.

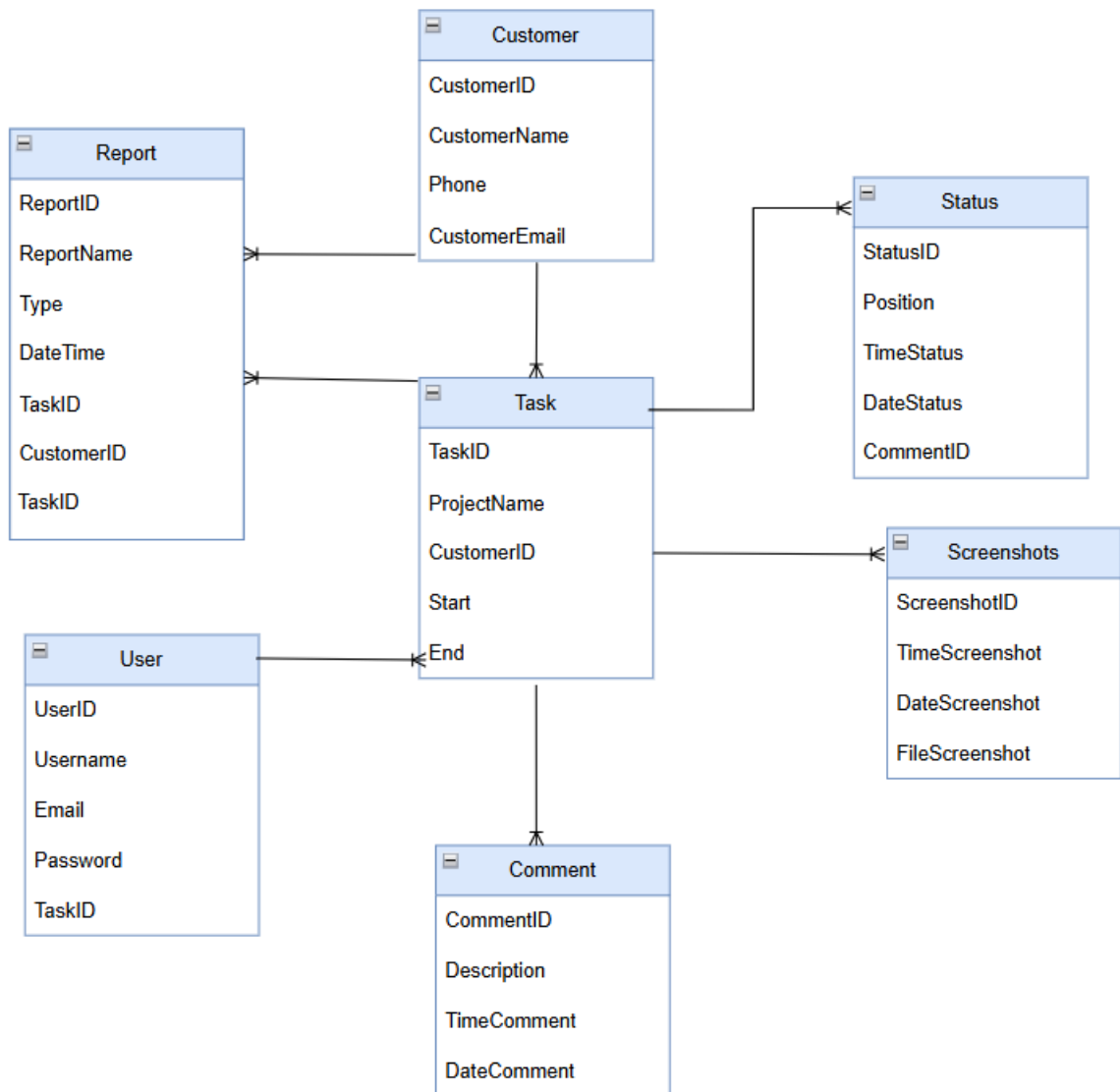


Рис.3.1. Інфологічна модель бази даних

3.1.2 Обґрунтування вибору СКБД та проєктування фізичної моделі БД в середовищі вибраної СКБД

Для реалізації бази даних було обрано систему керування базами даних (СКБД) SQL Server.

Вибір SQL Server як системи керування базами даних обґрунтовується кількома важливими аспектами, які забезпечують ефективну та надійну роботу інформаційних систем.

По-перше, SQL Server відзначається високою продуктивністю та масштабованістю. Завдяки оптимізації запитів та підтримці паралельної обробки даних, система здатна обробляти великі обсяги інформації. Крім того, підтримка розподілу даних дозволяє масштабувати систему відповідно до зростання бізнесу без значних втрат у продуктивності [35].

По-друге, SQL Server гарантує високий рівень безпеки даних. Він підтримує механізми аутентифікації, шифрування даних та аудиту доступу. Також SQL Server забезпечує дотримання принципів ACID (атомарність, консистентність, ізоляція, стійкість), що важливо для надійного зберігання та обробки даних у критичних умовах.

Також важливою перевагою є інтеграція SQL Server з іншими технологіями, такими як .NET, Microsoft Azure та хмарні сервіси. Це дозволяє створювати гнучкі рішення, які легко адаптуються до різноманітних бізнес-вимог та технологічних платформ [36].

Ще однією важливою перевагою є гнучкість у роботі з різними типами даних. SQL Server підтримує як OLTP (онлайн-обробка транзакційних даних), так і OLAP (онлайн-аналітична обробка даних), що дозволяє працювати з великими наборами транзакційних та аналітичних даних в межах однієї системи.

Також варто відзначити, що SQL Server використовує T-SQL, яке є розширенням стандарту SQL і дозволяє створювати складні запити, зберігання процедур, тригери і функції. Це дозволяє програмістам створювати складні логічні операції та автоматизувати численні процеси без необхідності використовувати зовнішні скрипти [37].

Крім того, SQL Server має вбудовані механізми для забезпечення високої доступності та відмовостійкості, такі як Always On Availability Groups і кластеризація, що забезпечує безперервну роботу системи навіть у разі апаратних збоїв або мережових проблем. На рисунку 3.2 продемонстрована фізична модель бази даних [38].

У додатку А надано структуру бази даних.

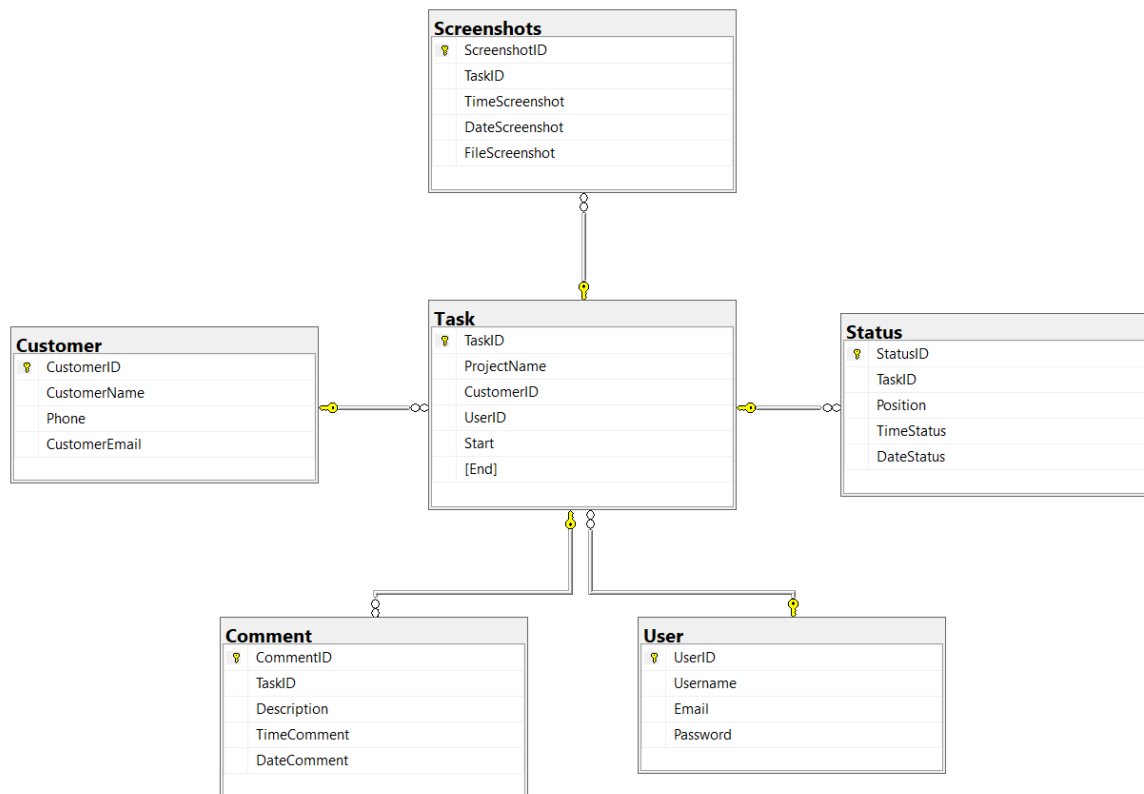


Рис.3.2 Фізична модель бази даних

3.1.3 Контрольний приклад та обрахунок прогнозних обсягів БД

Для відпрацювання контрольного прикладу заповнимо базу даних тестовими записами. У додатку Б надано усі запити для заповнення бази даних.

Контрольний приклад 1: Обчислення загального часу виконання задачі

Цей запит обчислює загальний час виконання задачі, враховуючи різницю між датою завершення (End) і датою початку (Start).

```
USE WorkTimeMonitoring; -- Вибір бази даних
```

```
GO
```

```
-- Обчислення загального часу виконання задачі у днях
```

```
SELECT
```

```
    TaskID,
```

```
    ProjectName,
```

```
    DATEDIFF(DAY, Start, [End]) AS TotalDurationDays
```

```
FROM Task
```

```
WHERE TaskID = 1;
```

```
GO
```

	TaskID	ProjectName	TotalDurationDays
1	1	Website Development	60

Рис.3.3 Обчислення загального часу виконання задачі у днях

Контрольний приклад 2: Підрахунок кількості змін статусу для конкретної задачі

```
USE WorkTimeMonitoring; -- Вибір бази даних
```

```
GO
```

```
-- Підрахунок кількості змін статусу для конкретної задачі
```

```
SELECT
```

```
    TaskID,
```

```
    COUNT(StatusID) AS StatusChanges
```

```
FROM Status
```

```
WHERE TaskID = 2
```

```
GROUP BY TaskID;
```

```
GO
```

	TaskID	StatusChanges
1	2	1

Рис. 3.4 Підрахунок кількості змін статусу для конкретної задачі

Для оцінки прогнозних обсягів бази даних розглянемо сценарій для середнього проєкту.

Припустимо, що:

- **Середній розмір одного запису:**
 - User: ~200 байт.
 - Customer: ~200 байт.
 - Task: ~300 байт.
 - Status: ~200 байт.
 - Comment: ~400 байт.
 - Screenshots: ~300 байт (з урахуванням посилання на файл).
- **Кількість записів у кожній таблиці для середнього проєкту:**
 - User: 10.
 - Customer: 10.
 - Task: 50.
 - Status: 200.
 - Comment: 150.
 - Screenshots: 500.

Обсяг (байт) = $(10 \times 200) + (10 \times 200) + (50 \times 300) + (200 \times 200) + (150 \times 400) + (500 \times 300) = 209,000$ байт (~203 КБ).

3.2. Проєктування бази знань інформаційної управляючої системи та засоби інтелектуального аналізу даних

У даному розділі розглядається процес проєктування бази знань (БЗ) для інформаційної управляючої системи (ІУС), а також реалізація інтелектуального аналізу даних (ІАД) з метою наповнення цієї бази знань. Важливість БЗ полягає у її здатності зберігати, обробляти та аналізувати великий обсяг даних, що є необхідним для прийняття рішень і автоматизації багатьох процесів, пов'язаних з управлінням та моніторингом робочих годин співробітників [39].

3.2.1 Реалізація інтелектуального аналізу великих даних з метою наповнення бази знань

Для реалізації інтелектуального аналізу великих даних в рамках розробки ІУС для моніторингу та відстеження робочих годин працівників використано набір методів та алгоритмів для обробки та аналізу даних. Основною метою є створення ефективної бази знань, яка автоматично відстежує і аналізує дані про робочі години співробітників.

Одним з перших етапів є попередня обробка даних. Для цього використовуються методи ETL (Extract, Transform, Load) та ELT (Extract, Load, Transform), які дозволяють зібрати дані з різних джерел (наприклад, електронні таблиці, системи контролю доступу), привести їх до єдиного формату та завантажити в базу даних [40]. Наприклад, інформація про час початку та закінчення робочого дня, перерви, запізнення буде приведена до стандартизованого формату і завантажена в базу.

Далі виконано нормалізацію та скорочення розмірності даних, оскільки вони можуть мати різні одиниці вимірювання. Для цього застосовано нормалізацію в діапазон від 0 до 1. Водночас, для зменшення складності використано методи скорочення розмірності, зокрема PCA (Principal Component Analysis), що дозволяє зберегти важливу інформацію при зниженні кількості характеристик [41].

Важливим етапом є глибинний аналіз даних, для якого застосовуються різноманітні алгоритми. Наприклад, для класифікації робочих годин, а також для виявлення аномалій або порушень робочого часу використовується алгоритм Naive Bayes [42]. Для сегментації співробітників за звичками щодо часу роботи застосовується k-means [43], що дозволяє групувати працівників, виявляючи типи графіків роботи. Також, для виявлення залежностей між робочими годинами і іншими факторами, використовується Apriori [44].

На етапі прогнозування використовується shallow artificial neural networks (ANN) для моделювання залежностей між різними факторами, що дозволяє передбачити майбутні зміни в робочому часі і коригувати графіки [45]. Для

прогнозування майбутніх робочих годин застосовується також аналіз часових рядів, що допомагає спрогнозувати навантаження на співробітників.

Результати обробки даних включають збереження відомостей про робочі години, перерви, запізнення та інші показники. Це дає можливість автоматично генерувати звіти для моніторингу робочого часу.

3.2.2 Проектування та реалізація бази знань інформаційної управляючої системи

На етапі ідентифікації бази знань визначаються задачі, що підлягають вирішенню в ІУС. Це включає автоматичний розрахунок робочих годин, виявлення запізнень, перевірка відповідності фактичного часу роботи заданому графіку. Основні поняття, які формалізуються, включають: робочі години, запізнення, перерва, продуктивність.

Формалізація цих понять полягає в тому, що робочі години визначаються як різниця між часом початку та закінчення робочого дня, за винятком часу на перерви. Запізнення обчислюється як різниця між фактичним часом початку робочого дня і встановленим графіком. Продуктивність може бути визначена як співвідношення фактично витрачених годин до запланованих [46].

На етапі концептуалізації бази знань визначається, як повинні зберігатися дані в системі для забезпечення їх ефективної обробки. Наприклад, для деяких працівників робочі години можуть бути гнучкими, а для інших — строго фіксованими. У базі знань будуть зберігатися дані про план робочого часу, фактично виконаний час, а також фактори, які впливають на відхилення від стандартного графіку [47].

Формалізація знань передбачає створення чітких моделей, які визначають правила для автоматичного обчислення робочих годин, запізнень і перерв, а також їхню взаємодію з іншими параметрами, такими як типи завдань або проекти. Це допоможе створити алгоритми для автоматичного коригування робочого часу.

Етап виконання (реалізація) полягає в інтеграції знань в інформаційну систему, що дозволяє автоматично обчислювати робочі години, виявляти порушення та коригувати робочі графіки. На цьому етапі система стає повністю функціональною і здатною працювати в реальному часі [48].

Нарешті, етап тестування є необхідним для перевірки правильності роботи бази знань. Тестуються різні сценарії, що включають реальні дані про робочі години, запізнення та інші змінні. Це дозволяє оцінити ефективність роботи системи і внести необхідні коригування.

У додатку Г наведено фрагменти первинних вхідних та вихідних даних.

3.3 Програмне забезпечення

У цьому розділі описується програмне забезпечення, яке було використано або розроблено в процесі реалізації інформаційно-управлінської системи (ІУС) для моніторингу та обліку робочого часу працівників. До програмного забезпечення належать: середовище розробки, мова програмування, програмні засоби, бібліотеки, фреймворки, система управління базами даних, а також створені компоненти ІУС.

3.3.1 Проектування програмного забезпечення

Проектування програмного забезпечення для інформаційно-управлінської системи (ІУС) для моніторингу та обліку робочого часу було здійснено на основі вимог технічного завдання та включало розробку бізнес-логіки та вибір технічних засобів для забезпечення належної функціональності системи.

Для організації програмних компонентів було обрано патерн проектування MVC (Model-View-Controller), який забезпечує ефективне розділення відповідальності між моделлю, відображенням та контролером. Модель обробляє і зберігає дані, відображення відповідає за взаємодію з користувачем, а контролер обробляє запити та передає їх між моделлю і відображенням [49].

Для веб-застосунку використано патерн MVVM (Model-View-ViewModel), що дозволяє відокремити логіку відображення від бізнес-логіки. Це дозволяє зручно керувати взаємодією користувача з системою та забезпечує масштабованість для майбутніх розширень.

Для розроблення програмного забезпечення було обрано мову JavaScript як основну мову програмування. Для створення клієнтської частини застосовувалися сучасні вебтехнології: HTML5, CSS3, JavaScript з використанням бібліотеки Chart.js для побудови діаграм та графіків. Стилзацію інтерфейсу виконано з використанням Bootstrap 5 [50], що забезпечило адаптивність та привабливий вигляд елементів керування.

Для організації структури сторінок застосовувалась компонентна розмітка та власна логіка маршрутизації засобами JavaScript. При необхідності обробки локальних даних використовувалось збереження у localStorage браузера.

Аутентифікація та авторизація користувачів реалізовані на стороні клієнта із використанням збереження сесій у пам'яті браузера (localStorage). Цей підхід є прийнятним для прототипів і внутрішніх систем, дозволяє швидко відновити доступ після перезавантаження сторінки та не потребує складної серверної інфраструктури.

Вибір архітектури був здійснений на основі вимог до безпеки, надійності та масштабованості системи. Оскільки система повинна бути здатною до швидкого розширення, було обрано мікросервісну архітектуру, що дає змогу легко адаптувати систему до змін і підвищити її гнучкість.

Прототипи структурних шаблонів програмного забезпечення були створені для полегшення розділення відповідальності між компонентами системи. Це дозволяє зменшити складність коду, спростити тестування та подальшу підтримку програмного продукту.

На етапі проєктування було успішно обрано й реалізовано необхідні технології та архітектурні рішення для забезпечення ефективної, безпечної та

масштабованої інформаційно-управлінської системи для моніторингу та обліку робочого часу.

3.3.2 Вибір архітектури програмного забезпечення

У процесі створення інформаційно-управлінської системи для моніторингу та обліку робочого часу працівників було прийнято рішення реалізувати її на основі монолітної архітектури. Такий підхід є доцільним на етапі активної розробки, оскільки забезпечує централізовану логіку обробки запитів, спрощує управління структурою проєкту та знижує інфраструктурні витрати.

Уся система реалізована у вигляді єдиного вебзастосунку з чітко структурованою клієнтською та серверною частинами. Користувацький інтерфейс побудовано з використанням HTML5, CSS3 та JavaScript із фреймворком Bootstrap 5. Динамічна поведінка, обробка форм, взаємодія з API та оновлення інтерфейсу без перезавантаження сторінки реалізовані засобами JavaScript із використанням асинхронних функцій (async/await).

Серверна частина реалізована на базі Node.js з використанням стандартних бібліотек та REST-архітектури для взаємодії між клієнтом і сервером. Усі функції авторизації, обробки даних, генерації звітів, управління задачами та обліку робочого часу обробляються централізовано в одному процесі. Взаємодія з базою даних (Microsoft SQL Server) реалізована через драйвери mssql, що дозволяє виконувати запити до бази в асинхронному режимі та ефективно працювати з великим обсягом інформації.

Для автентифікації користувачів застосовано модель сесійного збереження через cookie або механізм JSON Web Token (JWT), що дозволяє ідентифікувати користувача при кожному запиті без необхідності повторного логіну. У рамках монолітної структури це забезпечує надійну та просту у реалізації авторизацію, а також мінімізує накладні витрати на координацію між підсистемами.

У системі передбачено розмежування прав доступу відповідно до ролей користувачів. Основними типами є: працівник, керівник та адміністратор. Роль закріплюється за користувачем у момент реєстрації та визначає доступ до функціоналу: працівники можуть переглядати та звітувати про виконання завдань, залишати коментарі; керівники мають доступ до перегляду активності підлеглих, аналітики та управління завданнями; адміністратор має повний контроль над усією системою, зокрема над користувачами, контентом і налаштуваннями. Перевірка прав доступу реалізована як у серверній логіці, так і у клієнтських шаблонах — інтерфейс динамічно адаптується залежно від ролі користувача.

Передача даних між клієнтом і сервером здійснюється через захищене HTTPS-з'єднання, що гарантує конфіденційність інформації завдяки шифруванню на рівні транспортного протоколу. Усі запити, що містять облікові дані, інформацію про роботу чи скріншоти, не можуть бути перехоплені або змінені під час передачі.

Увесь вихідний код системи розміщується в одному репозиторії, що включає клієнтську частину, серверні маршрути, обробники запитів, шаблони та конфігураційні файли. Це дозволяє ефективно підтримувати систему, здійснювати контроль версій, вносити зміни та масштабувати її без потреби у складному розділенні модулів або мікросервісів.

Монолітна архітектура повністю відповідає вимогам до функціональності, безпеки, керованості та продуктивності. Обраний підхід забезпечує централізовану логіку, високу узгодженість компонентів та надійність функціонування системи в умовах реального використання.

3.3.3 Інструментальні засоби розробки для створення прикладного програмного забезпечення інформаційних управляючих систем

У процесі розробки програмного забезпечення інформаційно-управлінської системи для моніторингу та відстеження робочих годин співробітників було обрано низку інструментів та технологій, що дозволили

ефективно реалізувати бізнес-логіку, забезпечити стабільну роботу системи та зручну взаємодію між її компонентами.

Основою серверної частини виступає мова програмування JavaScript. Вона була обрана завдяки своїй гнучкості, зрозумілому синтаксису, потужній екосистемі бібліотек, а також широкому застосуванню у сфері веб-розробки. JavaScript дозволяє ефективно реалізовувати як структуровану логіку обробки даних, так і асинхронні процеси, що особливо важливо при побудові системи з високими вимогами до швидкодії.

Для реалізації вебінтерфейсу та клієнтської логіки було обрано JavaScript як основну мову програмування. Візуальна частина системи реалізована з використанням HTML5, CSS3 та фреймворку Bootstrap 5, що дозволило створити адаптивний, зручний для користувача інтерфейс. Функціональна логіка реалізована за допомогою JavaScript, без використання важких фронтенд-фреймворків, що надало системі гнучкості та простоти в обслуговуванні.

У якості системи керування базами даних було обрано Microsoft SQL Server. Такий вибір обумовлений технічними вимогами та наявною інфраструктурою замовника. У базі даних зберігається ключова інформація про користувачів, завдання, коментарі, скріншоти, а також журнали активності. Структура БД побудована за класичною реляційною моделлю з чітко визначеними зв'язками між таблицями.

Взаємодія з базою даних реалізується через серверну частину застосунку, що забезпечує виконання SQL-запитів, обробку запитів клієнтів, а також безпечну авторизацію користувачів. У системі реалізовані операції створення, читання, оновлення та видалення даних (CRUD), які відповідають на REST-запити з клієнтської частини.

Для реалізації асинхронної логіки в системі — зокрема, під час завантаження скріншотів, побудови графіків звітів або обробки великої кількості взаємодій на сторінці — було використано можливості JavaScript, включаючи роботу з асинхронними функціями (async/await) та обробниками подій. Це дозволило забезпечити швидке оновлення даних без перезавантаження сторінки,

а також забезпечити плавну та стабільну роботу інтерфейсу навіть у моменти пікового навантаження.

Фронтенд-секція системи побудована з використанням HTML5, CSS3, Bootstrap 5 та чистого JavaScript, що забезпечує гнучкість, адаптивність і простоту розгортання. Для візуалізації звітів та аналітики застосовується бібліотека Chart.js, яка дозволяє створювати інтерактивні графіки й діаграми (лінійні, кругові, гістограми тощо) на основі даних із бази.

Інтерфейс системи реалізовано таким чином, щоб він був зручним як на комп'ютерах, так і на мобільних пристроях. Адаптивна верстка дозволяє коректно відображати вміст у різних браузерах та на різних екранах, враховуючи потреби користувачів з різним рівнем технічної підготовки.

Для обміну даними між клієнтською частиною та базою даних використовується власна API-логіка, реалізована на стороні сервера, яка обробляє запити до Microsoft SQL Server. Передача даних відбувається у форматі JSON, що забезпечує просту інтеграцію та підтримку багатьох клієнтів. У разі потреби, можлива реалізація авторизації через токени, локальні сесії або localStorage.

З метою зменшення навантаження на сервер та підвищення швидкодії інтерфейсу, в системі застосовуються клієнтські методи кешування — наприклад, тимчасове збереження проміжних даних у localStorage браузера або повторне використання вже отриманих звітів без повторного запиту до сервера.

Серверна логіка реалізована у вигляді легкої монолітної архітектури, що об'єднує основні компоненти взаємодії з базою даних, генерації відповіді та обробки запитів. Такий підхід забезпечив швидкість розробки, легкість обслуговування і достатню масштабованість на етапі впровадження.

Обрані інструменти — JavaScript, Bootstrap, Chart.js, SQL Server — дозволили реалізувати стабільну та ефективну систему для моніторингу й обліку робочого часу, що відповідає сучасним вимогам до зручності, безпеки та гнучкості використання.

3.3.4 Тестування програмного забезпечення

У ході створення інформаційної системи для моніторингу та обліку робочого часу працівників було проведено повноцінне тестування, яке мало на меті перевірити функціональність, надійність і зручність користування програмним забезпеченням. З огляду на те, що реалізація системи здійснювалась із використанням мови JavaScript та вебтехнологій (HTML, CSS, JS), для тестування було використано відповідні інструменти, орієнтовані на клієнтське середовище.

Тестування виконувалось на кількох рівнях. Під час модульного тестування перевірялась робота окремих логічних компонентів, зокрема функцій обробки форм, обробників подій, відображення елементів інтерфейсу, побудови діаграм та перемикання між сторінками. Для написання юніт-тестів застосовувалась бібліотека Jest, що дозволила в ізольованому середовищі перевірити правильність роботи функцій, зокрема логіки генерації звітів, перемикання інтерфейсних блоків та обробки введених користувачем даних.

Інтеграційне тестування було спрямоване на перевірку взаємодії між компонентами. У системі, де немає серверної частини, особливу увагу було приділено тестуванню зв'язку між формами вводу, відображенням інтерфейсу та збереженням інформації у локальному контексті (наприклад, у пам'яті браузера). Важливо було переконатись, що при заповненні форм дані правильно відображаються у звітах, а дії користувача, як-от створення завдань чи перегляд скріншотів, коректно обробляються.

Системне тестування охоплювало перевірку працездатності всієї системи в цілому. Воно включало відкриття проєкту у різних браузерах (Google Chrome, Mozilla Firefox, Microsoft Edge), тестування адаптивності інтерфейсу на різних розмірах екрану, зокрема мобільних пристроях, а також перевірку відповідності зовнішнього вигляду елементів дизайну. Для цього активно використовувалися інструменти браузера DevTools та ручне тестування за попередньо складеними сценаріями. Було визначено кілька некритичних візуальних недоліків, зокрема

неправильне вирівнювання елементів на сторінці звітів та відсутність відступів у формі входу, які було оперативно виправлено.

З метою автоматизації функціонального тестування окремих елементів інтерфейсу застосовувалась бібліотека Playwright, яка дозволила змодельовати поведінку користувача — навігацію по сторінках, натискання кнопок, заповнення форм і перевірку відображення даних. Автоматизовані тести виявились корисними для регресійного тестування при повторних змінах інтерфейсу та стилів.

За підсумками тестування було підготовлено набір тест-кейсів, які охоплюють основні сценарії використання: реєстрацію та авторизацію користувача, створення та перегляд завдань, додавання скріншотів, формування звітів. Кожен тест-кейс містив опис вхідних даних, очікуваний результат, реальний результат виконання та висновок щодо успішності проходження тесту.

Результати тестування підтвердили, що система функціонує відповідно до поставлених вимог. Усі критичні помилки були усунуті, а розроблене програмне забезпечення готове до використання. Звіт про тестування знаходиться у додатку Г.

Також в додатку Е знаходяться тези з Всеукраїнської наукової конференції на тему «Аналіз напрямів автоматизації тестування програмного забезпечення.»

3.3.5 Керівництво (інструкція) користувача

Для роботи із сайтом потрібно відкрити вікно браузера та ввести url: <https://worktime.ua>.

При відкритті сайту користувач одразу ж потрапляє на головну сторінку (Рис. 3.7). На панелі зверху (хедері) відображені назви сторінок «Головна», «Звіти», «Сформований звіт», «Профіль», «Вийти». Також по центру розміщена кнопка «Увійти в систему». Нижче зображені три інформаційні таблиці з основними перевагами й можливостями системи, а саме — відстеження часу, автоматичні звіти, безпечний доступ.

Після натискання на кнопку «Увійти в систему» відображається форма для входу/ реєстрації (Рис. 3.9 – 3.10).

Якщо натиснути на панелі зверху на «Звіти», то відобразяться 9 видів можливих звітів (Рис. 3.12). Запропоновані звіти:

1. *Звіт за часом* (аналіз кількості відпрацьованих годин за день, тиждень або місяць). Цей звіт дає змогу керівнику чи замовнику побачити точну кількість годин, які кожен працівник фактично провів у системі протягом заданого періоду. Дані можуть бути згруповані по днях, тижнях або місяцях. Візуалізація реалізується у вигляді стовпчикових діаграм або таблиць. Розрахунок ведеться на основі часу активності на сайті та зафіксованих сесій (Рис. 3.13-3.14).

2. *Звіт по діяльності* (перегляд виконаних завдань та активності працівників). Дає змогу проаналізувати, які саме завдання були виконані за вибраний період, хто був відповідальний за їх виконання, скільки часу було витрачено та наскільки ефективно використовувався робочий час. Включає список задач із зазначенням статусів, дати створення, завершення та коментарів (Рис. 3.15).

3. *Оперативний підсумок* (зведена інформація про ефективність за короткий період). Використовується для швидкого огляду продуктивності команди за останні 24 години або останній робочий день. Включає загальну кількість активних годин, кількість завершених завдань, кількість порушень або простоїв.

4. *Табель обліку* (щомісячний табель з обліком робочого часу та відсутностей). Формується у вигляді таблиці, подібної до класичного табеля. Містить відомості про кожного працівника: присутність/відсутність, запізнення, лікарняні, відпустки. Може експортуватись у формат Excel або PDF для бухгалтерії або HR-відділу.

5. *Рейтинг* (порівняння ефективності співробітників за різними критеріями). Цей звіт генерує рейтинг працівників за показниками: кількість виконаних завдань, загальний відпрацьований час, середня тривалість виконання

задач, частка активного часу тощо. Дані можуть відображатися у вигляді топ-10 списку або діаграми (Рис. 3.16).

6. *Порушення* (випадки запізень, пропусків або несанкціонованих дій). Фіксуються усі відхилення від встановленого графіка чи регламенту: пізній початок роботи, ранній вихід, відсутність активності понад допустимий поріг, спроби обходу контролю. Звіт допомагає виявити системні порушення та реагувати на них.

7. *Діяльність* (докладна інформація про час, витрачений на задачі та проекти). Відображає часову структуру роботи співробітника або команди: які задачі виконувались, скільки часу було витрачено на кожну, яка частка задач завершена. Дає змогу порівняти планові показники з фактичними.

8. *Знімки* (перегляд автоматичних скріншотів з робочих місць). Система автоматично робить скріншоти з екранів працівників через задані інтервали. Цей звіт дозволяє переглянути послідовність знімків по кожному співробітнику, що дозволяє проконтролювати реальну зайнятість, а також перевірити, над чим працював користувач у конкретний момент часу.

9. *Динаміка* (візуалізація змін у продуктивності та завантаженні з часом). Цей аналітичний звіт демонструє, як змінювалась активність користувачів упродовж певного періоду. Наприклад, графік показує зростання або спад ефективності, інтенсивність роботи, частоту перерв. Це дозволяє виявити тренди, пікові навантаження або ознаки вигорання.

Сформовані звіти можна побачити через сторінку «Сформовані звіти», або ж натиснувши на відповідну назву звіту на сторінці «Звіти».

Також натиснувши на «Профіль» можна перейти в особистий профіль працівника, де вказані його ПІБ, e-mail, посада, стан (активний/ неактивний) та фото (за наявності) (Рис. 3.11).

3.4 Технічне забезпечення

Технічне забезпечення інформаційної управляючої системи (ІУС) для моніторингу та обліку робочого часу працівників включає апаратні та програмні

компоненти, необхідні для забезпечення всіх функцій системи. Комплекс технічних засобів (КТЗ) охоплює серверне обладнання, автоматизовані робочі місця (АРМ) користувачів, мережеву апаратуру передачі даних та периферійні пристрої.

3.4.1 Обґрунтування вибору технічного забезпечення

Комплекс технічних засобів ІУС включає наступні компоненти:

- центральний сервер, що виконує основні обчислювальні задачі і забезпечує збереження та обробку даних;
- автоматизовані робочі місця для користувачів системи;
- мережева апаратура, що забезпечує передачу даних між компонентами системи.

Для центрального сервера було обрано процесор Intel Xeon E5-2680 або його аналог завдяки високій багатоядерності та підтримці великої кількості потоків, що є критично важливим для обробки одночасних запитів від багатьох користувачів системи. Завдяки такій конфігурації забезпечується висока продуктивність серверної частини навіть при збільшенні навантаження. Оперативна пам'ять об'ємом 64 ГБ DDR4 гарантує стабільну роботу системи та ефективну обробку великої кількості одночасних процесів, включно з кешуванням даних для пришвидшення доступу до них. Накопичувач SSD об'ємом 1 ТБ обрано для забезпечення високої швидкості читання та запису даних, що безпосередньо впливає на швидкість обробки запитів користувачів та загальну продуктивність системи.

Як операційну систему на сервері передбачено встановлення Linux або Windows Server в залежності від потреб інтеграції, з урахуванням стабільності, безпеки та можливостей адміністрування.

Для надійного збереження даних передбачено окремий NAS-сервер, що дозволяє централізовано зберігати резервні копії, забезпечує гнучкість у розширенні обсягів пам'яті та гарантує безперервний доступ до архівних даних у разі збоїв основної системи.

Автоматизовані робочі місця (АРМ) для кінцевих користувачів комплектуються комп'ютерами з процесорами Intel Core i5, 8 ГБ оперативної пам'яті та SSD-диском на 256 ГБ. Така конфігурація дозволяє забезпечити комфортну роботу з веб-інтерфейсом системи, швидке завантаження сторінок, обробку даних у реальному часі та одночасне виконання декількох завдань без затримок. Операційна система Windows 10 Pro обрана через її розширені можливості з безпеки, підтримку корпоративних політик та сумісність із сучасним прикладним ПЗ.

Для побудови мережевої інфраструктури обрано комутатор Cisco Catalyst 9200 та маршрутизатор Cisco ISR 4321. Cisco Catalyst 9200 забезпечує високу швидкість обробки мережевого трафіку, підтримку сучасних стандартів безпеки та можливість масштабування мережі в майбутньому. Cisco ISR 4321 забезпечує надійний маршрутизаційний рівень із можливістю реалізації VPN, шифрування трафіку та підтримки якісного інтернет-з'єднання для віддаленої роботи користувачів.

Обране технічне забезпечення повністю відповідає вимогам щодо стабільності, безпеки, продуктивності та масштабованості інформаційно-управлінської системи, забезпечуючи її ефективну роботу як на етапі впровадження, так і при подальшому розвитку.

3.4.2 Ресурсні вимоги до технічного забезпечення

Для забезпечення стабільної роботи системи, яка виконує функції моніторингу та обліку робочого часу співробітників, визначено мінімальні технічні характеристики для кожного з ключових компонентів інфраструктури — від центрального сервера до кінцевих користувацьких пристроїв. Ці вимоги сформовані з урахуванням запланованої кількості одночасних користувачів, обсягів оброблюваних даних, а також особливостей збереження та резервування інформації.

В основі системи лежить центральний сервер, який виконує функції обробки запитів, запуску веб-застосунку, взаємодії з базою даних, а також

формування звітності. Для цієї ролі необхідно використовувати серверне обладнання з багатоядерним процесором, таким як Intel Xeon E5-2680 або аналогічним за продуктивністю. Обсяг оперативної пам'яті не повинен бути меншим за 64 ГБ DDR4, що дозволить безперешкодно обробляти десятки одночасних підключень та підтримувати стабільну роботу кешованих служб. Система зберігання даних на сервері має ґрунтуватися на твердотільному накопичувачі ємністю не менше 1 ТБ, що забезпечує високу швидкість читання та запису, особливо важливу при формуванні звітів або обробці великих обсягів журналів подій. У якості операційної системи рекомендовано використовувати Linux (наприклад, Ubuntu Server 22.04) або Windows Server 2019/2022 — залежно від інфраструктурних політик організації.

Оскільки система передбачає збереження зображень екранів працівників, логів активності та статистичних звітів, питання довготривалого та надійного зберігання даних є критично важливим. Для цього доцільно використання окремого NAS-сервера (Network Attached Storage) з можливістю створення RAID-масивів, що забезпечить не лише централізоване зберігання, але й резервування та швидкий доступ до архівної інформації. Такий підхід дозволяє розвантажити центральний сервер і підвищити надійність інфраструктури загалом.

Для роботи користувачів передбачено організацію автоматизованих робочих місць (АРМ). Типовий АРМ передбачає використання персонального комп'ютера з процесором Intel Core i5 (8-го покоління або новішим), 8 ГБ оперативної пам'яті та твердотільного накопичувача на 256 ГБ. Такої конфігурації достатньо для комфортної роботи в браузері, обробки задач, перегляду звітів та використання інтерактивного інтерфейсу системи. Операційна система на клієнтських пристроях — Windows 10 Pro, яка забезпечує повну сумісність із браузерами, підтримує групові політики та засоби корпоративної безпеки.

Мережева інфраструктура проєкту також повинна відповідати сучасним вимогам до продуктивності та безпеки. Для забезпечення стабільного

внутрішнього з'єднання рекомендовано використання керованого комутатора рівня 2/3, наприклад Cisco Catalyst 9200, що підтримує VLAN, QoS та інші механізми оптимізації трафіку. Для маршрутизації міжмережевого трафіку, підключення до Інтернету та реалізації політик безпеки — маршрутизатор корпоративного рівня, наприклад Cisco ISR 4321. Така конфігурація гарантує належний рівень доступності, резервування та захисту мережевого сегменту, де розташовано серверну частину ІУС.

На рисунку 3.6 зображена структурна схема комплексу технічних засобів.

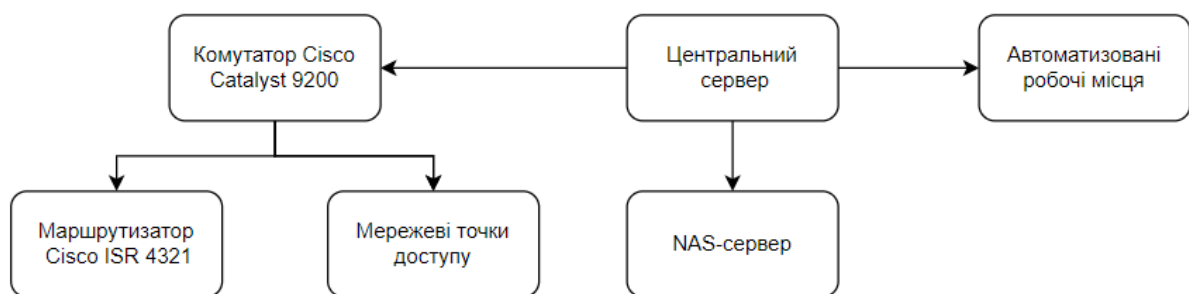


Рис.3.6 Структурна схема КТЗ ІУС для моніторингу та обліку робочого часу працівників

3.5 «Реалізація інформаційної управляючої системи»

Після запуску вебзастосунку користувач потрапляє на головну сторінку системи (рис. 3.7), яка містить вітальний блок, кнопку входу до системи та короткий опис основних функцій: автоматизований облік робочого часу, формування звітності та забезпечення безпечного доступу. Такий підхід дозволяє новим користувачам одразу зорієнтуватися у можливостях системи, а наявність кнопки «Увійти в систему» — швидко перейти до авторизації.

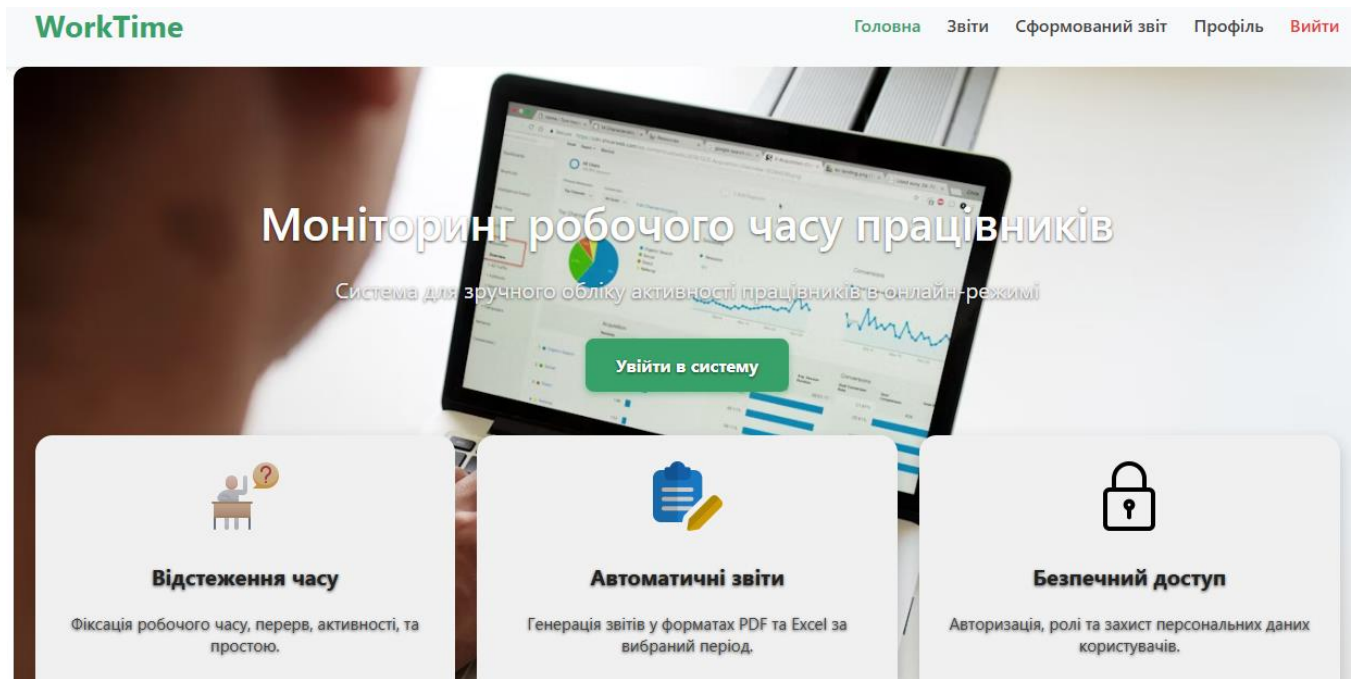


Рис.3.7 Головна сторінка системи для обліку робочого часу WorkTime

У разі наведення курсора на блоки із функціональними модулями, система реагує візуальним підсвічуванням елементів (рис. 3.8), що реалізовано через CSS-анімації та додає зручності при навігації, підкреслюючи інтерактивність інтерфейсу. Це особливо актуально в умовах високої щільності елементів на екрані, оскільки візуальний фідбек покращує UX-контекст взаємодії.

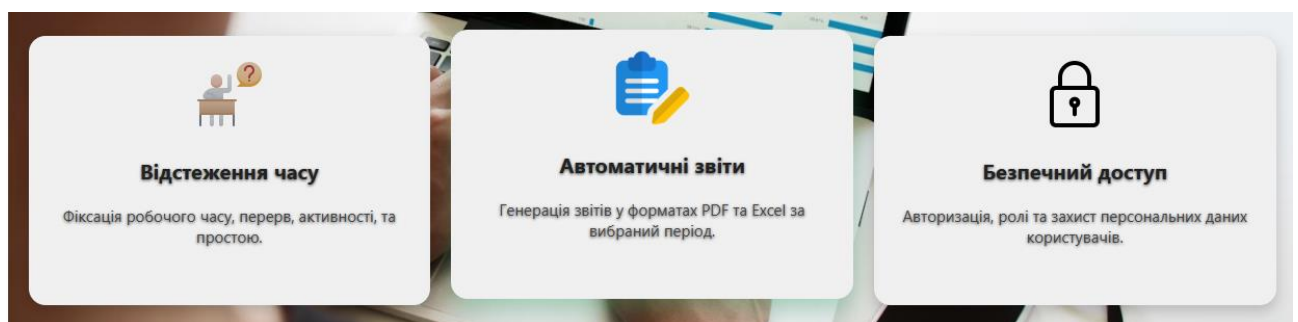
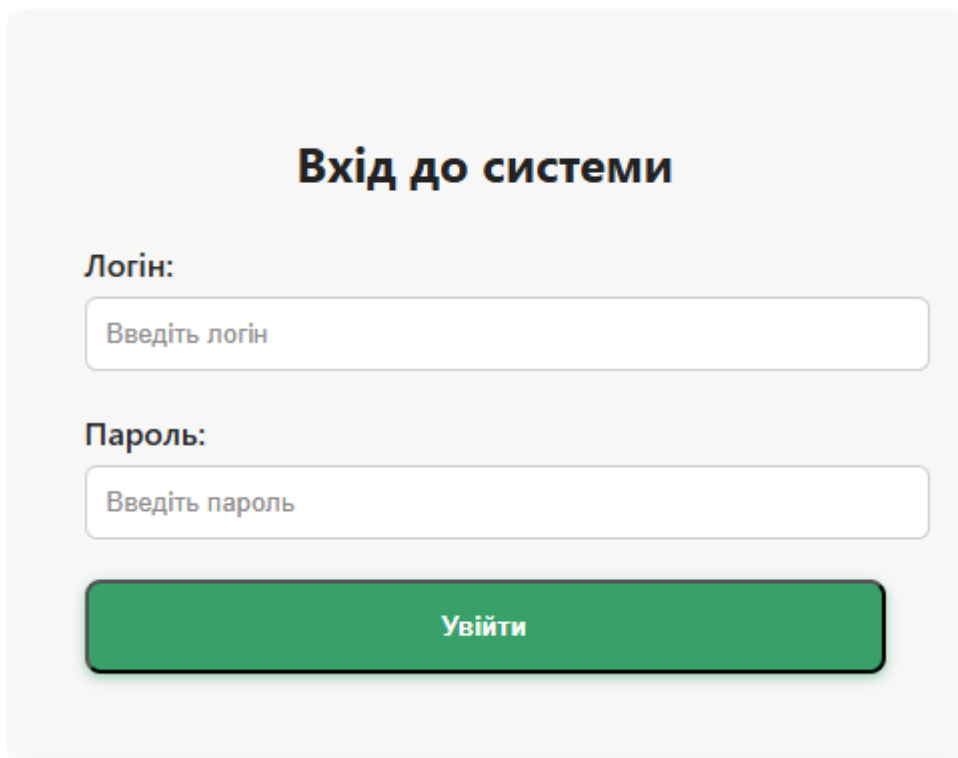


Рис.3.8 Підсвічування елементів на який наводиться курсор

На рис. 3.9 показано форму входу до системи, яка включає поля для логіну та пароля. Сторінка оформлена у мінімалістичному стилі, де збережено фокус на функціональності. Додаткові валідації забезпечують правильність введених даних. При введенні значень, як зображено на рис. 3.10, форма логіну

автоматично активує кнопку «Увійти», після чого відбувається перенаправлення до особистого кабінету.



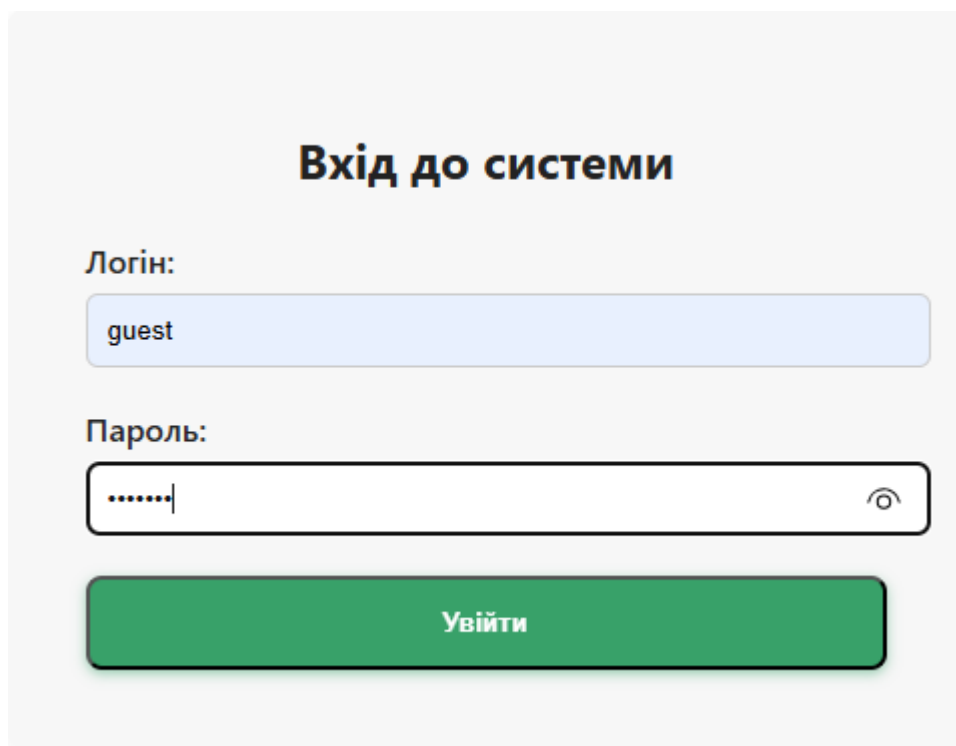
Вхід до системи

Логін:

Пароль:

Увійти

Рис.3.9 Вхід до системи



Вхід до системи

Логін:

Пароль:

Увійти

Рис.3.10 Заповнення полів логіна й паролю

Профіль користувача представлено на рис. 3.11. Він містить аватар (може бути змінений), ім'я, email, посаду користувача та його поточний статус у системі (наприклад, «Активний» або «Неактивний»). Також передбачено кнопку редагування профілю, яка дає змогу оновити персональні дані.

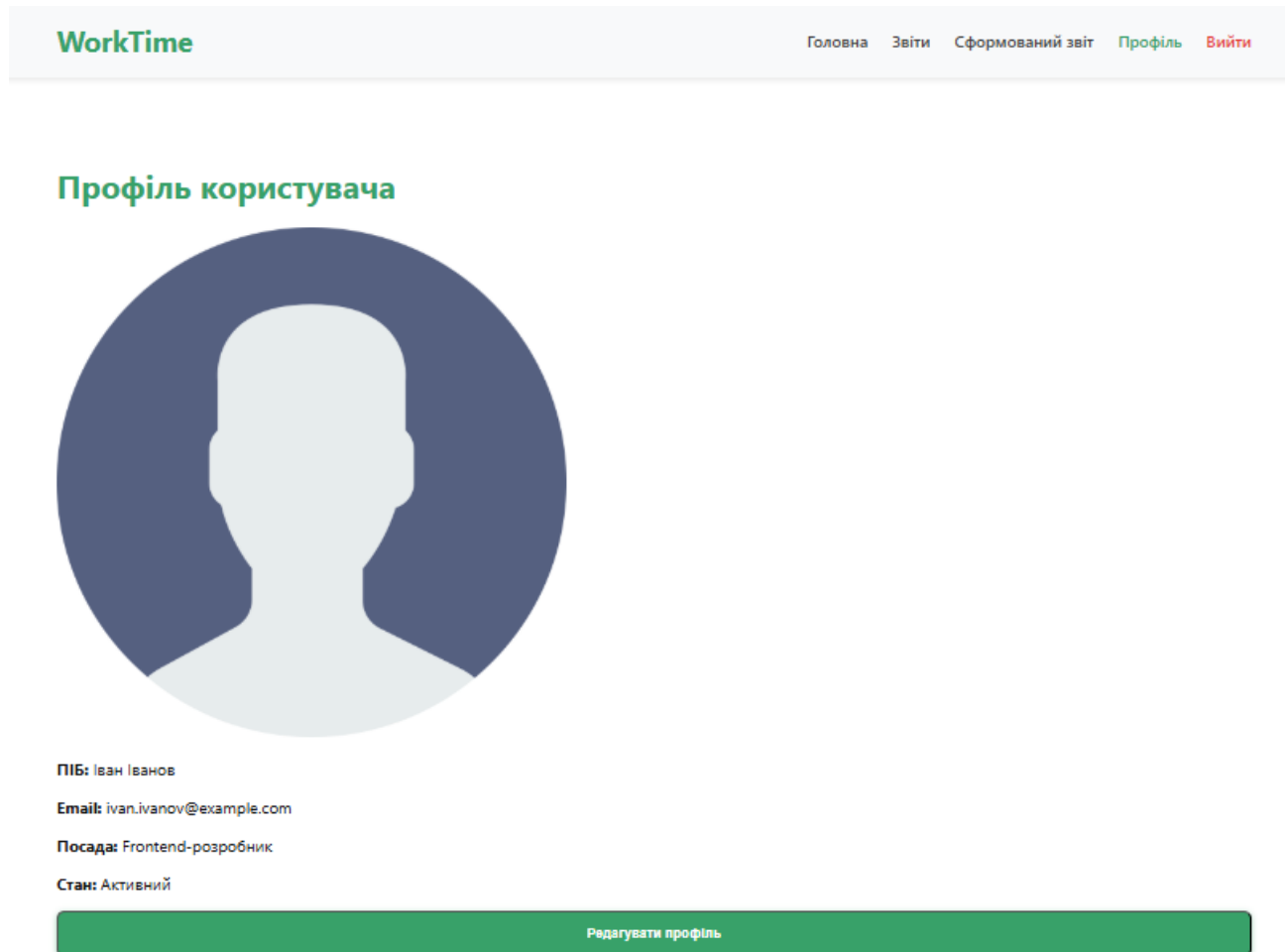


Рис.3.11 Профіль користувача

При переході до вкладки «Звіти», користувач отримує доступ до вибору типу звіту — усього передбачено дев'ять видів (рис. 3.12): звіт за часом, звіт по діяльності, оперативний підсумок, табель обліку, рейтинг, порушення, детальна діяльність, скріншоти та динаміка. Для кожного передбачено інтерактивну кнопку «Переглянути», яка відкриває відповідний модуль аналітики.

Звіти

Оберіть один із доступних типів звітів:

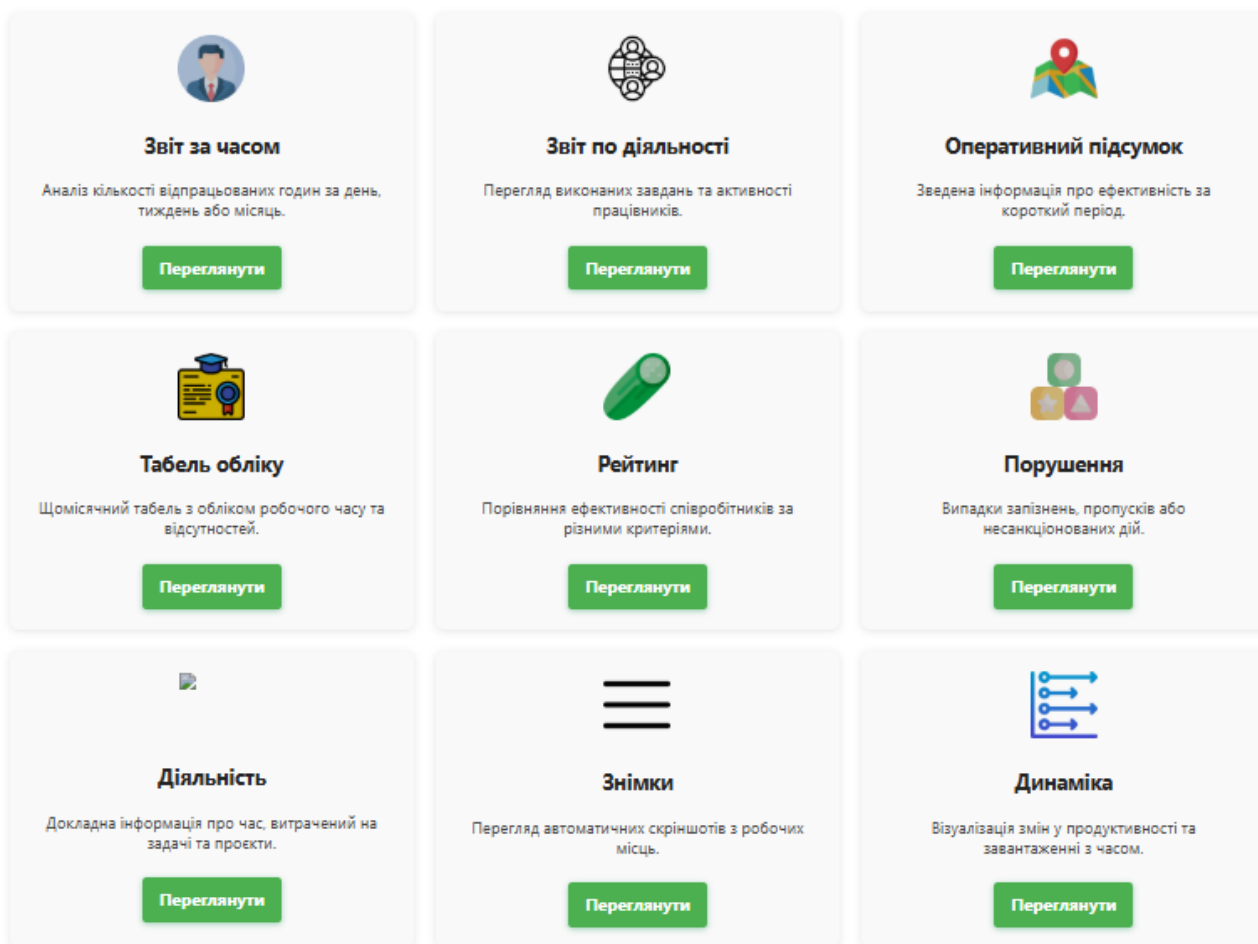


Рис.3.12 Доступні види звітів

Звіт за часом (рис. 3.13) представлений у вигляді стовпчикової діаграми, що ілюструє кількість годин, відпрацьованих кожним працівником за день. Зеленим кольором позначено продуктивний час, червоним — непродуктивний, а сірим — нейтральний або невизначений. Такий підхід дозволяє миттєво оцінити загальну залученість команди.

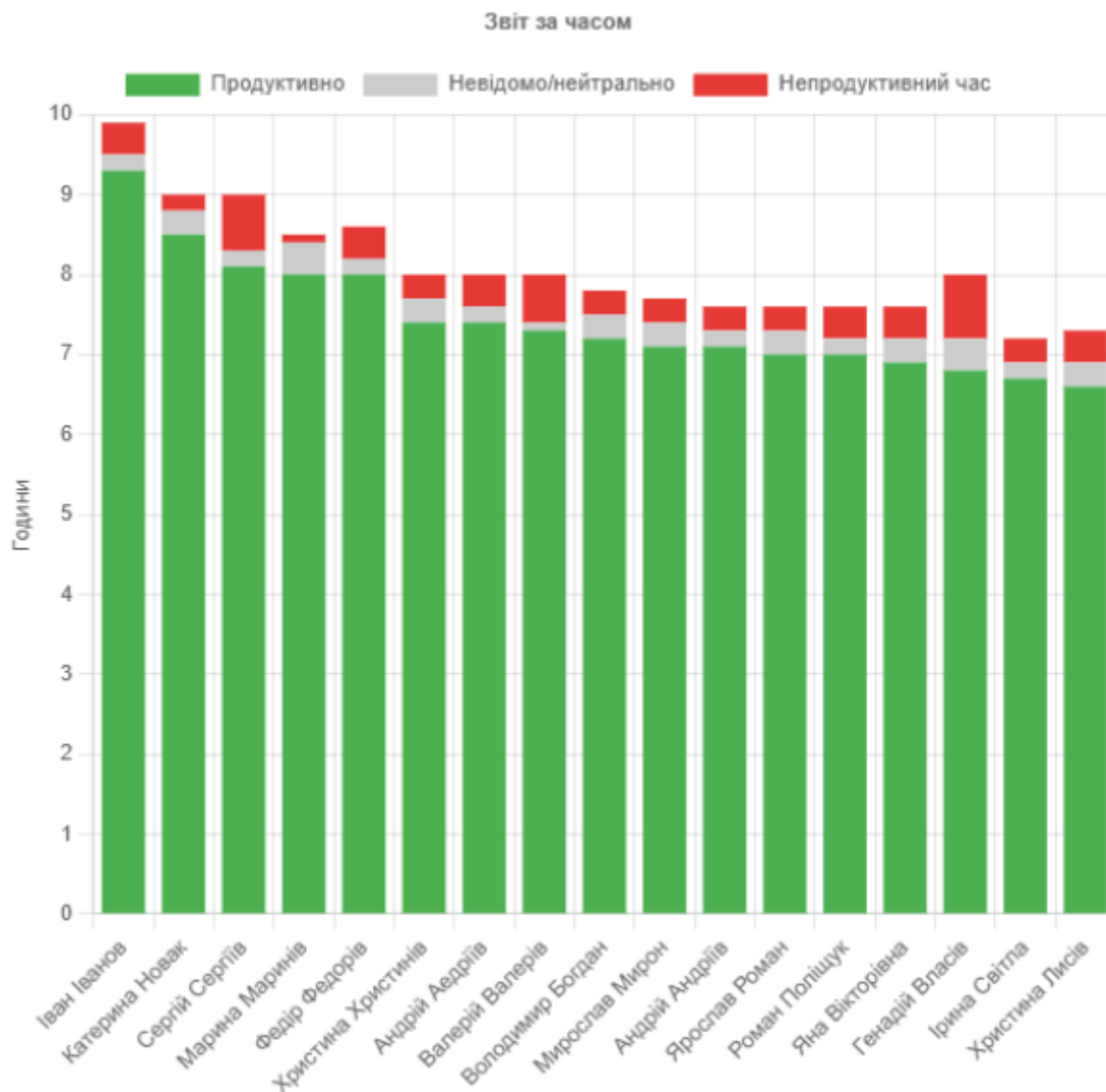


Рис.3.13 Звіт за часом

На рис. 3.14 подано деталізацію за конкретним працівником — зокрема, видно точні значення часу за всіма категоріями (наприклад, у Катерини Новак: 8.5 год продуктивно, 0.3 год нейтрально, 0.2 год непродуктивно).

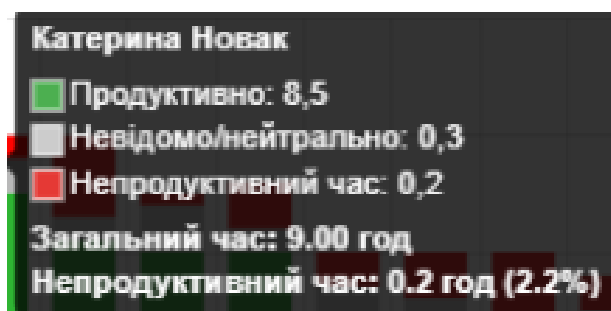


Рис.3.14 Непродуктивний час

У модулі звіту по діяльності (рис. 3.15) користувач має змогу обрати конкретного працівника та задати період спостереження. Якщо в зазначений період активність не була зафіксована, система інформує про це повідомленням. Інакше відображаються відповідні графіки.

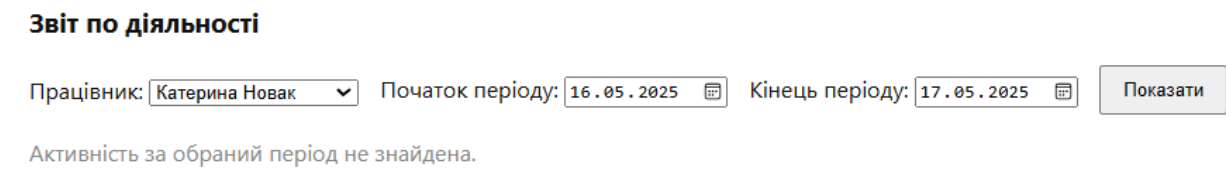


Рис.3.15 Звіт по діяльності

На рис. 3.16 показано рейтинг продуктивних працівників. Він сформований у вигляді кольорової стовпчикової діаграми, де кожен стовпчик відповідає конкретному працівнику. Висота стовпчика відображає відсоток продуктивності. Дана візуалізація дозволяє менеджерам швидко виявляти лідерів та потенційні зони ризику у команді.

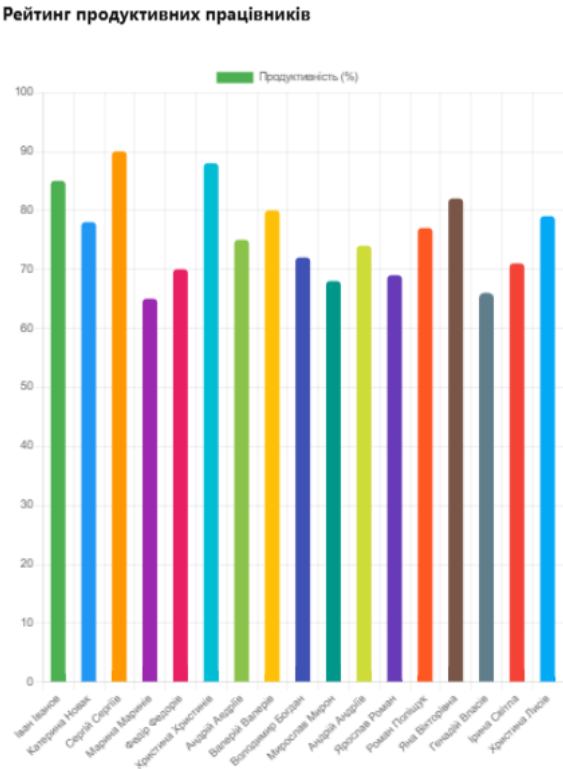


Рис.3.16 Рейтинг продуктивних працівників

ВИСНОВКИ

У кваліфікаційній магістерській роботі було розроблено інформаційно-управлінську систему для моніторингу та обліку робочого часу працівників під назвою WorkTime. Актуальність тематики обумовлена потребою підвищення прозорості трудової діяльності, забезпечення об'єктивної оцінки продуктивності персоналу, автоматизації процесів контролю робочого часу, а також запровадженням гнучких та дистанційних режимів роботи у сучасних організаціях, що особливо важливо в умовах поточної соціально-економічної нестабільності. Багато компаній досі використовують ручний або частково автоматизований облік, що унеможливорює отримання точної аналітики, виявлення порушень трудової дисципліни та об'єктивне оцінювання ефективності праці. Розробка комплексної автоматизованої системи, здатної збирати, обробляти, зберігати та візуалізувати дані про активність працівників, дозволяє вирішити ці завдання комплексно, забезпечуючи інструменти як для працівників, так і для керівників.

У теоретичній частині було виконано аналіз предметної області, визначено основні функціональні потреби системи, обґрунтовано вимоги до архітектури, безпеки та інтерфейсної складової майбутнього програмного забезпечення. Для обґрунтування вибору архітектурного рішення та ключового функціоналу було також розглянуто окремі функціональні особливості систем обліку часу, представлених у звіті, таких як ManicTime, TimeCamp, Clockify та Toggl Track, які продемонстрували різні підходи до реалізації збору даних, візуалізації активності та управління ролями користувачів. Аналіз дозволив сформулювати основні вимоги до функціоналу майбутньої системи, зокрема: мультирольову структуру, підтримку створення та ведення завдань, коментування, завантаження скріншотів, формування різних типів звітів, інтерактивність інтерфейсу та візуалізацію аналітики. Особливу увагу приділено питанню інформаційної безпеки, захисту даних користувачів та авторизації з використанням сучасних технологій.

У розділі про архітектуру розглянуто доцільність використання монолітного підходу як базового на етапі розробки. В системі реалізовано чітке розділення логіки за модулями: модуль обліку робочого часу, модуль керування завданнями, модуль генерації звітів, модуль скріншотів та зберігання файлів, модуль автентифікації та модуль адміністрування. Використано Node.js, Express, Microsoft SQL Server, при цьому взаємодія з базою реалізована без використання ORM — напямуч через SQL-запити, що дозволило мати повний контроль над структурою запитів і логікою обробки даних. З боку клієнта інтерфейс створено на HTML, CSS із використанням Bootstrap 5, JavaScript і бібліотеки Chart.js для побудови графіків.

Було створено повноцінну багатосторінкову вебсистему, яка включає:

- сторінки реєстрації та авторизації;
- панель керування користувачами (для адміністратора); створення завдань та призначення їх виконавцям;
- генерація звітів різних типів;
- профілі користувачів із зазначенням посади, активності (онлайн/офлайн), кількості виконаних завдань тощо;
- інтерфейс керівника для перегляду статусів та аналітики щодо підлеглих.

Інтерфейс адаптовано до мобільних пристроїв, реалізовано плавну навігацію між розділами, систему доступу на основі ролей, ітеративну побудову списків та таблиць за допомогою AJAX. Дані між фронтендом і бекендом передаються через REST API у форматі JSON, реалізована система захисту від CSRF, XSS та інших загроз.

Особлива увага була приділена реалізації звітів. Система дозволяє отримати:

- звіт за часом (з графіком відпрацьованих годин);
- звіт по діяльності (перелік завдань, статус, час виконання);
- оперативний підсумок (за обраний день чи тиждень);
- табель обліку (для HR та бухгалтерії);
- рейтинг працівників (за продуктивністю);

- аналіз порушень дисципліни (запізнення, відсутність);
- динаміку навантаження по відділах;
- перегляд автоматичних скріншотів;
- діаграми активності працівника з посиланням на сторінки.

Тестування системи відбувалося в кілька етапів: модульне, інтеграційне, ручне, системне. Було розроблено та виконано тест-кейси на всі критичні функції. Зокрема, перевірялися: реєстрація з валідацією даних, правильна фільтрація завдань, відповідність даних у звітах, успішність передачі скріншотів на сервер, контроль доступу до інтерфейсних елементів залежно від ролі, обробка помилок у REST API. Тестування показало стабільну роботу системи в типовому та піковому режимі навантаження. Виправлено понад 20 помилок у процесі інтеграції.

Досягнуті результати можна охарактеризувати такими кількісними показниками:

- 18 вебсторінок користувацького інтерфейсу;
- понад 1500 рядків коду JavaScript;
- 25 ендпоінтів REST API;
- 9 типів звітів із побудовою діаграм;
- 3 ролі користувачів з перевіркою доступу;
- понад 100 тестових сценаріїв;
- повна адаптивність під десктоп і мобільні екрани.

У результаті всіх виконаних етапів поставлена мета — створення зручної, функціональної та безпечної інформаційної системи моніторингу робочого часу — повністю реалізована. Вебзастосунок WorkTime є ефективним інструментом для відстеження активності працівників, оцінювання ефективності, автоматизації звітності та покращення управління трудовими ресурсами.

У перспективі можливе розширення функціоналу, зокрема:

- інтеграція з популярними месенджерами (Telegram, Slack);
- підтримка мобільного застосунку на React Native;
- введення таймера з паузами;

- автоматичне виявлення неактивності;
- оцінка настрою працівника за поведінковими метриками;
- автоматизовані аналітичні звіти керівнику;
- підключення до ERP/CRM систем.

Розроблена система готова до впровадження в невеликі та середні організації та може бути основою для подальшої розробки масштабованих рішень в корпоративному середовищі. З метою подальшого вдосконалення, систему можна доповнити розширеною адміністративною панеллю з аналітикою у реальному часі, підтримкою багатомовності, автоматичним резервним копіюванням даних, а також можливістю інтеграції з біометричними пристроями для більш точного обліку часу. Доцільним також є впровадження механізмів машинного навчання для прогнозування продуктивності та раннього виявлення порушень трудової дисципліни. Це розширить сферу застосування системи та дозволить іще точніше адаптувати її під потреби різних організацій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rath, Y. (2021). Remote Work and Employee Performance: A Study of the Challenges and Benefits. – Режим доступу: https://www.researchgate.net/publication/385256573_A_Study_on_Challenges_and_Benefits_of_Remote_and_Hybrid_Work_Model_from_Employees_Perspective
2. Upwork [Електронний ресурс]. – Режим доступу: <https://www.upwork.com/> – Дата доступу: 06.01.2025. – Назва з екрана
3. Toggl Track. Toggl Plan Time Tracking [Електронний ресурс]. – Режим доступу: <https://toggl.com/track/toggl-plan-time-tracking/> – Дата доступу: 06.01.2025. – Назва з екрана.
4. Tick [Електронний ресурс]. – Режим доступу: <https://www.tickspot.com/> – Дата доступу: 06.01.2025. – Назва з екрана.
5. Timesheets [Електронний ресурс]. – Режим доступу: <https://www.timesheets.com/> – Дата доступу: 06.01.2025. – Назва з екрана.
6. OfficeTime [Електронний ресурс]. – Режим доступу: <https://officetime.net/dev/> – Дата доступу: 06.01.2025. – Назва з екрана.
7. Timely [Електронний ресурс]. – Режим доступу: <https://timelyapp.com/home-v2-q4-24> – Дата доступу: 06.01.2025. – Назва з екрана.
8. HoursTracker [Електронний ресурс]. – Режим доступу: <http://hourstrackerapp.com/> – Дата доступу: 06.01.2025. – Назва з екрана.
9. Kickidler [Електронний ресурс]. – Режим доступу: <https://www.kickidler.com/> – Дата доступу: 06.01.2025. – Назва з екрана.
10. CrocoTime [Електронний ресурс]. – Режим доступу: <https://crocotime.com/ru/> – Дата доступу: 06.01.2025. – Назва з екрана.
11. Monitask [Електронний ресурс]. – Режим доступу: <https://www.monitask.com/en/> – Дата доступу: 06.01.2025. – Назва з екрана.
12. Моніторинг робочого часу [Електронний ресурс]. – Режим доступу: <://hurma.work/blog/oblik-robochogo-chasy//> – Дата доступу: 20.01.2025. – Назва з екрана.

13. Облік робочого часу [Електронний ресурс]. – Режим доступу: [Облік робочого часу – що це | ЦБА](#) – Дата доступу: 21.01.2025. – Назва з екрана.
14. Штучний інтелект [Електронний ресурс]. – Режим доступу: [Штучний інтелект — Вікіпедія](#) – Дата доступу: 05.02.2025. – Назва з екрана.
15. Машинне навчання [Електронний ресурс]. – Режим доступу: [Машинне навчання — Вікіпедія](#) – Дата доступу: 05.02.2025. – Назва з екрана.
16. Великі дані [Електронний ресурс]. – Режим доступу: [Великі дані — Вікіпедія](#) – Дата доступу: 05.02.2025. – Назва з екрана.
17. Інтернет речей [Електронний ресурс]. – Режим доступу: [Інтернет речей — Вікіпедія](#) – Дата доступу: 05.02.2025. – Назва з екрана.
18. Доповнена та віртуальна реальність [Електронний ресурс]. – Режим доступу: [Віртуальна реальність \(VR\) та доповнена реальність: що це таке? | Lviv Science Museum](#) – Дата доступу: 15.02.2025. – Назва з екрана.
19. Arduino [Електронний ресурс]. – Режим доступу: [What is Arduino? | Arduino Documentation](#) Дата доступу: 15.02.2025. – Назва з екрана.
20. ESP8266 [Електронний ресурс]. – Режим доступу: [ESP8266 - Wikipedia](#) Дата доступу: 15.02.2025. – Назва з екрана.
21. Orange Pi [Електронний ресурс]. – Режим доступу: [Orange Pi - Wikipedia](#) Дата доступу: 15.03.2025. – Назва з екрана.
22. Raspberry Pi [Електронний ресурс]. – Режим доступу: [Raspberry Pi - Wikipedia](#) Дата доступу: 15.03.2025. – Назва з екрана.
23. Microsoft SQL Server [Електронний ресурс]. – Режим доступу: [Microsoft SQL Server - Wikipedia](#) Дата доступу: 16.03.2025. – Назва з екрана.
24. Microsoft SQL Server [Електронний ресурс]. – Режим доступу: [Microsoft SQL Server - Wikipedia](#) Дата доступу: 16.03.2025. – Назва з екрана.
25. ER Diagrams [Електронний ресурс]. – Режим доступу: [Introduction of ER Model | GeeksforGeeks](#) Дата доступу: 17.03.2025. – Назва з екрана.
26. Java Script [Електронний ресурс]. – Режим доступу: [What is JavaScript? | Codecademy](#) Дата доступу: 18.03.2025. – Назва з екрана.

27. HTML [Електронний ресурс]. – Режим доступу: [HTML — Вікіпедія](#)
Дата доступу: 18.03.2025. – Назва з екрана.
28. CSS [Електронний ресурс]. – Режим доступу: [What is CSS3? What's new and different from traditional CSS?](#) Дата доступу: 25.03.2025. – Назва з екрана.
29. Chart.js [Електронний ресурс]. – Режим доступу: [Chart.js | Chart.js](#)
Дата доступу: 25.03.2025. – Назва з екрана.
30. Діаграма класів [Електронний ресурс]. – Режим доступу: [Діаграма класів — Вікіпедія](#) Дата доступу: 26.03.2025. – Назва з екрана.
31. Extract, load, transform (ELT) [Електронний ресурс]. – Режим доступу: [What Is Extract, Load, Transform \(ELT\)? | IBM](#) Дата доступу: 01.04.2025.
– Назва з екрана.
32. MVC (Model-View-Controller) [Електронний ресурс]. – Режим доступу: [Model-view-controller - Wikipedia](#) Дата доступу: 01.04.2025. – Назва з екрана.
33. Гайна Г.А. Основи проектування баз даних: Навчальний посібник. – К.: КНУБА, 2005. – 204 с.
34. Берко А.Ю., Верес О.М., Пасічник В.В. Системи управління базами даних та знань: Навчальний посібник. – Львів: Львівська політехніка, 2013. – 328 с.
35. Коваль В.С., Струбицький П.Р. Алгоритми і структури даних. – Навчальний посібник – Тернопіль: ФОП Шпак В.Б. – 2017. – 74 с.
36. Пасічник В.В., Резніченко В.А. Організація баз даних та знань. – К: Видавнича група ВНУ, 2006.- 384 с.
37. Мікула М.П., Коцюк Ю.А., Мікула О.М. Організація баз даних та знань: Навчальний посібник. – Острог: Острозька академія, 2021. – 384 с.
38. Основи інформаційних систем: Навч.посібник. – Вид. 2-ге, перероб. і доп. / В.Ф. Ситник, Т.А. Писаревська, Н.В. Єрєміна, О.С. Краєва; За ред. В.Ф. Ситника. – К.: КНЕУ, 2001. – 420 с.
39. Інформаційні системи в економіці: монографія за заг. ред. д-ра екон. наук, проф. С.В.Устенка. - К.:КНЕУ, 2012. – 425 с.

40. Доценко С.І. Організація та системи керування базами даних: Навчальний посібник. – Харків: УкрДУЗТ, 2023. – 117 с.
41. PCA (Principal Component Analysis) [Електронний ресурс]. – Режим доступу: [What Is Principal Component Analysis \(PCA\)? | IBM](#) Дата доступу: 01.04.2025.– Назва з екрана.
42. Naive Bayes [Електронний ресурс]. – Режим доступу: [What Are Naïve Bayes Classifiers? | IBM](#) Дата доступу: 02.04.2025. – Назва з екрана.
43. K-means Bayes [Електронний ресурс]. – Режим доступу: [What is K-Means algorithm and how it works – TowardsMachineLearning](#) Дата доступу: 02.04.2025. – Назва з екрана.
44. Apriori [Електронний ресурс]. – Режим доступу: [Apriori Algorithm | GeeksforGeeks](#) Дата доступу: 10.04.2025. – Назва з екрана.
45. Artificial Neural Networks [Електронний ресурс]. – Режим доступу: [Artificial Neural Networks and its Applications | GeeksforGeeks](#) Дата доступу: 10.04.2025. – Назва з екрана.
46. Інтелектуальний аналіз даних та машинне навчання. Частина 1. Базові методи та засоби аналізу даних / Я.В. Іванчук, В.І. Месюра, А.А. Яровий, О.Д. Манжілевський – Вінниця: ВНТУ, 2021. – 69 с.
47. Литвин В.В. Бази знань інтелектуальних систем підтримки прийняття рішень: Монографія. – Львів: Львівська політехніка, 2011. – 128с.
48. Гавриленко В.В., Струневич Л.М. Система управління базами даних MySQL. Навчальний посібник. – Київ: НТУ, 2013. – 61с.
49. React [Електронний ресурс]. – Режим доступу: <https://react.dev> Дата доступу: 11.04.2025. – Назва з екрана.
50. Bootstrap 5 [Електронний ресурс]. – Режим доступу: [Introduction · Bootstrap v5.0](#) Дата доступу: 12.04.2025. – Назва з екрана.

ДОДАТКИ

Додаток А

Структура бази даних:

- [-]  WorkTimeMonitoring
 - [-]  Database Diagrams
 -  dbo.Фізична модель
 - [-]  Tables
 - [+]  System Tables
 - [+]  FileTables
 - [+]  External Tables
 - [+]  Graph Tables
 - [+]  dbo.Comment
 - [+]  dbo.Customer
 - [+]  dbo.Screenshots
 - [+]  dbo.Status
 - [+]  dbo.Task
 - [+]  dbo.User
 - [+]  Dropped Ledger Tables
 - [+]  Views
 - [+]  External Resources
 - [+]  Synonyms
 - [+]  Programmability
 - [+]  Query Store
 - [+]  Service Broker
 - [+]  Storage
 - [+]  Security

Додаток Б

SQL запит для створення проєктованої бази даних:

-- Створення бази даних

```
CREATE DATABASE WorkTimeMonitoring;  
GO
```

-- Використання створеної бази даних

```
USE WorkTimeMonitoring;  
GO
```

-- Створення таблиці "Користувачі"

```
CREATE TABLE [User] (  
    UserID INT PRIMARY KEY IDENTITY(1,1),  
    Username NVARCHAR(100) NOT NULL,  
    Email NVARCHAR(255) NOT NULL UNIQUE,  
    Password NVARCHAR(255) NOT NULL  
);  
GO
```

-- Створення таблиці "Задачі"

```
CREATE TABLE Task (  
    TaskID INT PRIMARY KEY IDENTITY(1,1),  
    ProjectName NVARCHAR(255) NOT NULL,  
    CustomerID INT NOT NULL, -- Зовнішній ключ до таблиці Customer  
    UserID INT NOT NULL, -- Зовнішній ключ до таблиці User  
    Start DATE NOT NULL,  
    [End] DATE NULL  
);
```

GO

-- Створення таблиці "Статуси"

```
CREATE TABLE Status (  
    StatusID INT PRIMARY KEY IDENTITY(1,1),  
    TaskID INT NOT NULL, -- Зовнішній ключ до таблиці Task  
    Position NVARCHAR(255) NOT NULL,  
    TimeStatus DATETIME NOT NULL,  
    DateStatus DATE NOT NULL  
);  
GO
```

-- Створення таблиці "Коментарі"

```
CREATE TABLE Comment (  
    CommentID INT PRIMARY KEY IDENTITY(1,1),  
    TaskID INT NOT NULL, -- Зовнішній ключ до таблиці Task  
    Description NVARCHAR(MAX) NOT NULL,  
    TimeComment DATETIME NOT NULL,  
    DateComment DATE NOT NULL  
);  
GO
```

-- Створення таблиці "Замовники"

```
CREATE TABLE Customer (  
    CustomerID INT PRIMARY KEY IDENTITY(1,1),  
    CustomerName NVARCHAR(255) NOT NULL,  
    Phone NVARCHAR(20) NULL,  
    CustomerEmail NVARCHAR(255) NOT NULL
```

```
);  
GO
```

```
-- Створення таблиці "Скріншоти"
```

```
CREATE TABLE Screenshots (  
    ScreenshotID INT PRIMARY KEY IDENTITY(1,1),  
    TaskID INT NOT NULL, -- Зовнішній ключ до таблиці Task  
    TimeScreenshot DATETIME NOT NULL,  
    DateScreenshot DATE NOT NULL,  
    FileScreenshot NVARCHAR(255) NOT NULL -- Шлях до файлу  
);  
GO
```

```
-- Додавання зв'язків між таблицями
```

```
-- Зв'язок Task з User
```

```
ALTER TABLE Task  
ADD CONSTRAINT FK_Task_User FOREIGN KEY (UserID) REFERENCES  
[User](UserID);
```

```
-- Зв'язок Task з Customer
```

```
ALTER TABLE Task  
ADD CONSTRAINT FK_Task_Customer FOREIGN KEY (CustomerID)  
REFERENCES Customer(CustomerID);
```

```
-- Зв'язок Status з Task
```

```
ALTER TABLE Status  
ADD CONSTRAINT FK_Status_Task FOREIGN KEY (TaskID) REFERENCES  
Task(TaskID);
```

-- Зв'язок Comment з Task

ALTER TABLE Comment

ADD CONSTRAINT FK_Comment_Task FOREIGN KEY (TaskID)
REFERENCES Task(TaskID);

-- Зв'язок Screenshots з Task

ALTER TABLE Screenshots

ADD CONSTRAINT FK_Screenshots_Task FOREIGN KEY (TaskID)
REFERENCES Task(TaskID);

GO

USE WorkTimeMonitoring;

GO

-- Заповнення таблиці "Користувачі"

INSERT INTO [User] (Username, Email, Password)

VALUES

('JohnDoe', 'john.doe@example.com', 'hashed_password_123'),

('JaneSmith', 'jane.smith@example.com', 'hashed_password_456'),

('AlexBrown', 'alex.brown@example.com', 'hashed_password_789'),

('EmmaDavis', 'emma.davis@example.com', 'hashed_password_321'),

('LiamWilson', 'liam.wilson@example.com', 'hashed_password_654'),

('SophiaTaylor', 'sophia.taylor@example.com', 'hashed_password_987'),

('JamesClark', 'james.clark@example.com', 'hashed_password_111'),

('MiaWhite', 'mia.white@example.com', 'hashed_password_222'),

('LucasMoore', 'lucas.moore@example.com', 'hashed_password_333'),

('OliviaHall', 'olivia.hall@example.com', 'hashed_password_444');

GO

-- Заповнення таблиці "Замовники"

```
INSERT INTO Customer (CustomerName, Phone, CustomerEmail)
VALUES
('ACME Corp', '+123456789', 'contact@acme.com'),
('Tech Solutions', '+987654321', 'support@techsol.com'),
('Green Ventures', '+456123789', 'info@greenventures.com'),
('Skyline Inc.', '+741852963', 'hello@skyline.com'),
('Bright Future Ltd', '+852741963', 'team@brightfuture.com'),
('Starry Night', '+369258147', 'admin@starrynight.com'),
('CodeCraft', '+147852369', 'info@codecraft.com'),
('NextGen Systems', '+159357456', 'nextgen@systems.com'),
('Global Tech', '+753951456', 'global@tech.com'),
('EcoWorld', '+951753852', 'support@ecoworld.com');
GO
```

-- Заповнення таблиці "Задачі"

```
INSERT INTO Task (ProjectName, CustomerID, UserID, Start, [End])
VALUES
('Website Development', 1, 1, '2024-01-01', '2024-03-01'),
('Mobile App Design', 2, 2, '2024-02-01', NULL),
('Marketing Campaign', 3, 3, '2024-01-15', '2024-02-15'),
('Database Optimization', 4, 4, '2024-01-20', '2024-02-20'),
('UI/UX Redesign', 5, 5, '2024-02-10', NULL),
('Cloud Migration', 6, 6, '2024-01-05', '2024-01-30'),
('E-commerce Platform', 7, 7, '2024-02-01', '2024-03-01'),
('CRM Integration', 8, 8, '2024-01-25', NULL),
('Data Analytics Setup', 9, 9, '2024-01-15', '2024-02-15'),
```

```
('Security Audit', 10, 10, '2024-01-10', '2024-01-25');
```

```
GO
```

```
-- Заповнення таблиці "Статуси"
```

```
INSERT INTO Status (TaskID, Position, TimeStatus, DateStatus)
```

```
VALUES
```

```
(1, 'In Progress', GETDATE(), CAST(GETDATE() AS DATE)),
```

```
(2, 'Completed', GETDATE(), CAST(GETDATE() AS DATE)),
```

```
(3, 'Pending Approval', GETDATE(), CAST(GETDATE() AS DATE)),
```

```
(4, 'In Progress', GETDATE(), CAST(GETDATE() AS DATE)),
```

```
(5, 'On Hold', GETDATE(), CAST(GETDATE() AS DATE)),
```

```
(6, 'In Progress', GETDATE(), CAST(GETDATE() AS DATE)),
```

```
(7, 'Completed', GETDATE(), CAST(GETDATE() AS DATE)),
```

```
(8, 'In Progress', GETDATE(), CAST(GETDATE() AS DATE)),
```

```
(9, 'Pending Review', GETDATE(), CAST(GETDATE() AS DATE)),
```

```
(10, 'Completed', GETDATE(), CAST(GETDATE() AS DATE));
```

```
GO
```

```
-- Заповнення таблиці "Коментарі"
```

```
INSERT INTO Comment (TaskID, Description, TimeComment, DateComment)
```

```
VALUES
```

```
(1, 'Initial setup completed.', GETDATE(), CAST(GETDATE() AS DATE)),
```

```
(2, 'Design approved by client.', GETDATE(), CAST(GETDATE() AS DATE)),
```

```
(3, 'Database schema updated.', GETDATE(), CAST(GETDATE() AS DATE)),
```

```
(4, 'Bug fix applied.', GETDATE(), CAST(GETDATE() AS DATE)),
```

```
(5, 'Requirements clarified.', GETDATE(), CAST(GETDATE() AS DATE)),
```

```
(6, 'Testing in progress.', GETDATE(), CAST(GETDATE() AS DATE)),
```

```
(7, 'Code review completed.', GETDATE(), CAST(GETDATE() AS DATE)),
```

```
(8, 'Performance optimized.', GETDATE(), CAST(GETDATE() AS DATE)),  
(9, 'New feature added.', GETDATE(), CAST(GETDATE() AS DATE)),  
(10, 'Documentation updated.', GETDATE(), CAST(GETDATE() AS DATE));  
GO
```

```
-- Заповнення таблиці "Скріншоти"
```

```
INSERT INTO Screenshots (TaskID, TimeScreenshot, DateScreenshot,  
FileScreenshot)
```

```
VALUES
```

```
(1, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task1/screenshot1.png'),  
(2, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task2/screenshot2.png'),  
(3, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task3/screenshot3.png'),  
(4, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task4/screenshot4.png'),  
(5, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task5/screenshot5.png'),  
(6, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task6/screenshot6.png'),  
(7, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task7/screenshot7.png'),  
(8, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task8/screenshot8.png'),  
(9, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task9/screenshot9.png'),  
(10, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task10/screenshot10.png');
```

```
GO
```

Додаток В

SQL-запит для заповнення бази даних. Заповнюємо кожну таблицю 10 записами:

USE WorkTimeMonitoring;

GO

-- Заповнення таблиці "Користувачі"

INSERT INTO [User] (Username, Email, Password)

VALUES

('JohnDoe', 'john.doe@example.com', 'hashed_password_123'),

('JaneSmith', 'jane.smith@example.com', 'hashed_password_456'),

('AlexBrown', 'alex.brown@example.com', 'hashed_password_789'),

('EmmaDavis', 'emma.davis@example.com', 'hashed_password_321'),

('LiamWilson', 'liam.wilson@example.com', 'hashed_password_654'),

('SophiaTaylor', 'sophia.taylor@example.com', 'hashed_password_987'),

('JamesClark', 'james.clark@example.com', 'hashed_password_111'),

('MiaWhite', 'mia.white@example.com', 'hashed_password_222'),

('LucasMoore', 'lucas.moore@example.com', 'hashed_password_333'),

('OliviaHall', 'olivia.hall@example.com', 'hashed_password_444');

GO

-- Заповнення таблиці "Замовники"

INSERT INTO Customer (CustomerName, Phone, CustomerEmail)

VALUES

```
( 'ACME Corp', '+123456789', 'contact@acme.com'),
( 'Tech Solutions', '+987654321', 'support@techsol.com'),
( 'Green Ventures', '+456123789', 'info@greenventures.com'),
( 'Skyline Inc.', '+741852963', 'hello@skyline.com'),
( 'Bright Future Ltd', '+852741963', 'team@brightfuture.com'),
( 'Starry Night', '+369258147', 'admin@starrynight.com'),
( 'CodeCraft', '+147852369', 'info@codecraft.com'),
( 'NextGen Systems', '+159357456', 'nextgen@systems.com'),
( 'Global Tech', '+753951456', 'global@tech.com'),
( 'EcoWorld', '+951753852', 'support@ecoworld.com');
GO
```

```
-- Заповнення таблиці "Задачі"
```

```
INSERT INTO Task (ProjectName, CustomerID, UserID, Start, [End])
VALUES
( 'Website Development', 1, 1, '2024-01-01', '2024-03-01'),
( 'Mobile App Design', 2, 2, '2024-02-01', NULL),
( 'Marketing Campaign', 3, 3, '2024-01-15', '2024-02-15'),
( 'Database Optimization', 4, 4, '2024-01-20', '2024-02-20'),
( 'UI/UX Redesign', 5, 5, '2024-02-10', NULL),
( 'Cloud Migration', 6, 6, '2024-01-05', '2024-01-30'),
( 'E-commerce Platform', 7, 7, '2024-02-01', '2024-03-01'),
( 'CRM Integration', 8, 8, '2024-01-25', NULL),
( 'Data Analytics Setup', 9, 9, '2024-01-15', '2024-02-15'),
( 'Security Audit', 10, 10, '2024-01-10', '2024-01-25');
GO
```

```
-- Заповнення таблиці "Статуси"
```

```

INSERT INTO Status (TaskID, Position, TimeStatus, DateStatus)
VALUES
(1, 'In Progress', GETDATE(), CAST(GETDATE() AS DATE)),
(2, 'Completed', GETDATE(), CAST(GETDATE() AS DATE)),
(3, 'Pending Approval', GETDATE(), CAST(GETDATE() AS DATE)),
(4, 'In Progress', GETDATE(), CAST(GETDATE() AS DATE)),
(5, 'On Hold', GETDATE(), CAST(GETDATE() AS DATE)),
(6, 'In Progress', GETDATE(), CAST(GETDATE() AS DATE)),
(7, 'Completed', GETDATE(), CAST(GETDATE() AS DATE)),
(8, 'In Progress', GETDATE(), CAST(GETDATE() AS DATE)),
(9, 'Pending Review', GETDATE(), CAST(GETDATE() AS DATE)),
(10, 'Completed', GETDATE(), CAST(GETDATE() AS DATE));
GO

```

-- Заповнення таблиці "Коментарі"

```

INSERT INTO Comment (TaskID, Description, TimeComment, DateComment)
VALUES
(1, 'Initial setup completed.', GETDATE(), CAST(GETDATE() AS DATE)),
(2, 'Design approved by client.', GETDATE(), CAST(GETDATE() AS DATE)),
(3, 'Database schema updated.', GETDATE(), CAST(GETDATE() AS DATE)),
(4, 'Bug fix applied.', GETDATE(), CAST(GETDATE() AS DATE)),
(5, 'Requirements clarified.', GETDATE(), CAST(GETDATE() AS DATE)),
(6, 'Testing in progress.', GETDATE(), CAST(GETDATE() AS DATE)),
(7, 'Code review completed.', GETDATE(), CAST(GETDATE() AS DATE)),
(8, 'Performance optimized.', GETDATE(), CAST(GETDATE() AS DATE)),
(9, 'New feature added.', GETDATE(), CAST(GETDATE() AS DATE)),
(10, 'Documentation updated.', GETDATE(), CAST(GETDATE() AS DATE));
GO

```

-- Заповнення таблиці "Скріншоти"

```
INSERT INTO Screenshots (TaskID, TimeScreenshot, DateScreenshot,  
FileScreenshot)
```

```
VALUES
```

```
(1, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task1/screenshot1.png'),
```

```
(2, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task2/screenshot2.png'),
```

```
(3, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task3/screenshot3.png'),
```

```
(4, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task4/screenshot4.png'),
```

```
(5, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task5/screenshot5.png'),
```

```
(6, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task6/screenshot6.png'),
```

```
(7, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task7/screenshot7.png'),
```

```
(8, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task8/screenshot8.png'),
```

```
(9, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task9/screenshot9.png'),
```

```
(10, GETDATE(), CAST(GETDATE() AS DATE),  
'/screenshots/task10/screenshot10.png');
```

```
GO
```

Додаток Г

Вихідна інформація. Вихідною інформацією системи є результати обробки вхідних даних: сформовані звіти, оновлені статуси завдань, надіслані коментарі та скріншоти, що слугують для контролю за процесом виконання роботи. Ці дані використовуються адміністраторами, керівниками та замовниками.

Опис вихідних повідомлень представлено у табл. Г.1.

Таблиця Г.1– Перелік та опис вихідних повідомлень

Назва вихідного повідомлення	Ідентифікатор	Форма подання і вимоги до неї	Періодичність видання	Термін видання і допустимий час затримки	Користувач інформації
1	2	3	4	5	6
Статистичний звіт	OUT1	Таблиця у форматі PDF/XLSX з аналізом відпрацьованого часу, продуктивності, запізень	Щотижня / Щомісяця	Протягом 1 доби після звітного періоду	Керівник, HR-відділ
Коментар	OUT2	Текстовий коментар до завдання (зберігається в БД, відображається в UI); довжина до 500 символів	У момент збереження	В режимі реального часу	Учасники проєкту
Скріншоти завдань	OUT3	Зображення у форматі PNG/JPEG, архів ZIP; фіксується з інтервалом 10 хвилин	Під час виконання завдання	У режимі реального часу або щотижневий архів	Керівник, служба безпеки
Зміна стану завдання	OUT4	Повідомлення у вигляді запису в БД; змінюється статус (наприклад: "У роботі" → "Завершено")	У момент зміни статусу	У режимі реального часу	Керівник, система звітності

Джерело даних: авторська розробка.

Структурними одиницями вихідного повідомлення «Статистичний звіт», ідентифікатор якого – OUT1 є:

- Ім'я працівника
- Дата / період звіту
- Загальна кількість відпрацьованих годин
- Кількість запізнень / пропущених днів
- Середня продуктивність
- Нарахована зарплата
- Проект, до якого належить звіт

Структурними одиницями вихідного повідомлення «Коментар», ідентифікатор якого – OUT2 є:

- Ім'я автора
- ID користувач
- ID завдання
- Текст коментаря
- Дата та час публікації

Структурними одиницями вихідного масиву «Скріншоти завдань», ідентифікатор якого – OUT3 є:

- ID працівника
- ID завдання
- Дата і час зняття скріншоту
- Посилання / шлях до зображення
- Формат зображення

Структурними одиницями вихідного масиву «Зміна стану завдання», ідентифікатор якого – OUT4 є:

- ID завдання
- Назва завдання
- Попередній статус

- V_стані: посада, дата, час.
- V_коментар: опис, дата, час.
- V_скріншот: файл, дата, час.
- V_звіт: дата початку, дата завершення, вибрані завдання, id замовника.

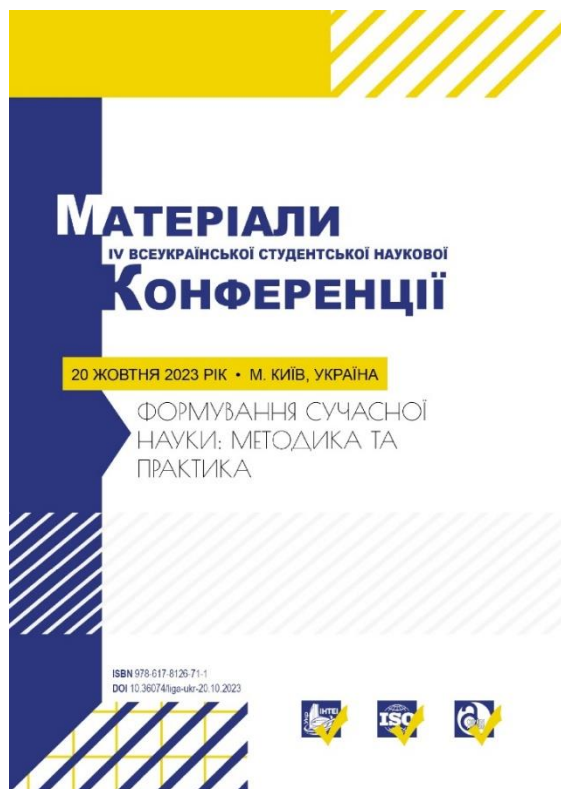
Додаток Д

Таблиця Д.1-Тест-кейси

№	Назва тест-кейсу	Вхідні дані	Очікуваний результат	Реальний результат	Висновок
1	Реєстрація нового користувача	Логін: admin Пароль: "12345678"	Користувача зареєстровано, відкрито головну сторінку	Реєстрація виконана, перенаправлення відбулося	Пройдено
2	Повторна реєстрація з тим самим email	Логін: admin нове ім'я і пароль	Повідомлення про помилку — email вже зареєстровано	Повідомлення з'явилося	Пройдено
3	Успішна авторизація користувача	Логін: admin Пароль: "12345678"	Вхід до системи, відкрито панель управління	Авторизація відбулася	Пройдено
4	Авторизація з помилковим паролем	Логін: admin Пароль: "wrongpass"	Повідомлення про неправильні дані	Помилка відображається	Пройдено
5	Створення нового завдання	Назва: "Звіт", Опис: "Підготувати до 12:00", Термін: 1 день	Завдання збережене, з'явилося у списку	Завдання з'явилося у таблиці	Пройдено
6	Спроба створити завдання без назви	Назва: "", інші поля заповнені	З'являється повідомлення про помилку	Повідомлення є, кнопка неактивна	Пройдено
7	Перегляд завдань користувачем-замовником	Користувач із правами замовника	Відображено список його завдань	Список видно	Пройдено
8	Додавання скріншоту до завдання	Вибране зображення + відповідне завдання	Зображення прикріплено, доступне для перегляду	Зображення з'явилося під завданням	Пройдено
9	Генерація звіту	Користувач, вибрано звіт	Побудовано діаграму активності та перерв	Діаграма з'явилась	Пройдено
10	Перегляд профілю користувача	Користувач авторизований	Виведено ім'я, логін, посаду, стан	Уся інформація відображається	Пройдено

Джерело: сформовано автором на основі виконаного дослідження

Тези, опубліковані в збірнику IV ВСЕУКРАЇНСЬКОЇ СТУДЕНТСЬКОЇ НАУКОВОЇ КОНФЕРЕНЦІЇ



Формування сучасної науки: методика та практика

СЕКЦІЯ 14.

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ

Мовак Катерина Миколаївна, здобувачка вищої освіти Навчально-науковий інститут «Інститут інформаційних технологій в економіці» Київський національний економічний університет ім. Вадима Гетьмана, Україна

Науковий керівник: Устенко Станіслав Веніамінович, д-р. екон. наук, професор, професор кафедри інформаційних систем в економіці Київський національний економічний університет ім. Вадима Гетьмана, Україна

АНАЛІЗ НАПРЯМІВ АВТОМАТИЗАЦІЇ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Актуальність. В сучасному світі розробки програмного забезпечення, автоматизація тестування визнається однією з найважливіших складових процесу розробки. Організація системи автоматичного тестування дозволяє ефективно перевірити функціональність та надійність програмного продукту, зменшує витрати, під час нього використовуються програмні засоби для виконання тестів і перевірки результатів виконання, що допомагає скоротити час тестування і спростити його процес [1].

Аналіз існуючих джерел. Існує декілька напрямів автоматизації тестування програмного забезпечення, такі як: модульне тестування, функціональне тестування, інтеграційне тестування та UI-тестування [1-3].

Виклад дослідження.

Проведемо порівняльний аналіз напрямів автоматизації тестування програмного забезпечення, таких як: модульне тестування, функціональне тестування, інтеграційне тестування та UI-тестування їх аналіз:

1. Модульне тестування передбачає автоматизацію тестування окремих модулів або компонентів програми. Воно дозволяє перевірити коректність її роботи кожного модуля в ізоляції, що полегшує виявлення та виправлення помилок.
2. Функціональне тестування зосереджене на перевірці функціональності програми. Автоматизовані тести перевіряють, чи виконують програмні компоненти встановлені функціональні вимоги.
3. Для перевірки взаємодії між різними компонентами програми застосовують інтеграційне тестування. Автоматизація цього процесу допомагає виявити проблеми взаємодії різних частин програми.
4. UI-тестування спрямоване на перевірку коректності роботи користувацького інтерфейсу програми. Автоматизовані тести перевіряють відповідність вигляду та поведінки інтерфейсу заданим вимогам.

Відзначимо переваги автоматизації тестування:

- Зниження витрат часу і ресурсів, адже ручне (мануальне) тестування вимагає

122

20 жовтня 2023 рік | Київ, Україна | Молодіжна наукова ліга

більше часу. Ручне тестування є простим типом тестування, що не вимагає великої кількості додаткових знань[3]. Мануальне тестування вимагає більших зусиль, але без нього ми не зможемо переконатися в тому, чи можлива автоматизація взагалі;

- Під час автоматизованого тестування можна уникнути помилок пов'язаних із втому або неправильними діями тестувальників (людський фактор);
- Автоматизація розширює обсяг тестування, що дозволяє проводити більшу кількість тестів та аналізувати різноманітні характеристики програмного продукту;
- Автоматизоване тестування сприяє виявленню помилок на початкових стадіях розробки, що дозволяє виправити їх до випуску програмного продукту в експлуатацію;
- Також необхідно визначити недоліки автоматизації програмного тестування:
 - Розробка автоматизованих тестів може бути дорогою та вимагати багато часу на початковому етапі;
 - Не всі види тестів можуть бути автоматизовані, наприклад, тестування на візуальний ефект, може вимагати ручної перевірки;
 - Програмний продукт постійно змінюється і автоматизовані тести потребують постійного оновлення.

Враховуючи вище зазначені напрями тестування та плюси і мінуси автоматизації тестування пропонується створити інформаційну систему для управління та моніторингу процесу тестування[4]. Дана система буде спиратись на потужну базу знань [5], яка узагальнить, систематизує та розширить знання щодо автоматизації тестування. Дослідження різних напрямків автоматизованого тестування допомагає розробникам та тестувальникам створити ефективну базу знань, де кожен напрямок вимагає відповідного підходу та методології для автоматизації. Наявність такої бази знань сприятиме підвищенню продуктивності, точності та швидкості тестування, а також покращить якість програмного продукту.

Система для управління та моніторингу процесу тестування може бути корисною для різних сфер застосування, де вимагається тестування програмного забезпечення, наприклад, таких як: інформаційні технології, фінансовий сектор, медична сфера, автомобільна промисловість, логістика та транспорт [6].

Висновки: Проведено порівняльний аналіз напрямів автоматизації тестування програмного забезпечення, таких як: модульне тестування, функціональне тестування, інтеграційне тестування та UI-тестування. Відзначено, що автоматизація тестування програмного забезпечення є важливим напрямом, який допомагає покращити якість та швидкість розробки програм. Вибір різних підходів до автоматизації тестування залежить від конкретних потреб проєкту та ресурсів, а також вимагає постійного удосконалення та підтримки. Розуміння цих використовуваних підходів автоматизації тестування допомагає зробити процес розробки програмного забезпечення більш ефективним та надійним. Запропоновано створити систему для керування та моніторингу процесу тестування, яка враховує напрями тестування та спирається на потужну базу знань. Вважаю, що дана робота допоможе мені в розробленні кваліфікаційної магістерської роботи з проєктування інформаційної управляючої системи для керування та моніторингу процесу тестування.

Список використаних джерел:

1. Автоматизоване тестування: матеріал з Вікіпедії [Електронний ресурс]. Режим доступу:

Формування сучасної науки: методика та практика

1. Автоматизоване тестування — Вікіпедія (wikipedia.org).
2. Стратегії для автоматизації тестування програмного забезпечення: матеріал з ZapTest [Електронний ресурс]. Режим доступу: Автоматизація тестування програмного забезпечення – історія, переваги, труднощі та інше! (zaptest.com).
3. Ручне тестування: матеріал з QALight [Електронний ресурс]. Режим доступу: Ручне та автоматизоване тестування - QALight.
4. Системне тестування: матеріал з ZapTest [Електронний ресурс]. Режим доступу: Системне тестування - типи, процес, інструменти та багато іншого! (zaptest.com).
5. База знань: матеріал з Вікіпедії [Електронний ресурс]. Режим доступу: База знань — Вікіпедія (wikipedia.org).
6. Моніторинг тестування: матеріал з QATestLab [Електронний ресурс]. Режим доступу: Види моніторингу, які допомагають у тестуванні | Онлайн-курси від компанії QATestLab.

123

124

Додаток Є

Тези, опубліковані в збірнику V Міжнародної науково-практичної конференції молодих вчених, аспірантів і студентів (Інтернет-конференція):

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
 Київський національний економічний університет імені Вадима Гетьмана
 Ярославська Державна Вища Технічно-Економічна Школа
 ім. о. Броніслава Маркевича (Польща)
 Академія фінансів та бізнесу Vistula (Польща)
 Європейський університет у Варшаві (Польща)
 Компанія Pro Insight sp. z o.o. (Польща)
 Вроцлавський університет економіки та бізнесу (Польща)
 Інститут кібернетики імені В.М. Глушкова НАН України
 Київський національний університет імені Тараса Шевченка
 Національний технічний університет України «КПІ ім. Ігоря Сікорського»
 Харківський національний економічний університет ім.Семена Кузнеця
 Національний технічний університет «ХПІ»
 Харківський національний університет імені В. Н. Каразіна
 Харківський національний університет радіоелектроніки
 Харківський національний університет міського господарства ім. О.М. Бекетова
 Херсонський національний технічний університет
 Запорізький національний університет
 Хмельницький національний університет
 Криворізький державний педагогічний університет
 Східноукраїнський національний університет ім. Володимира Даля
 Львівський національний університет імені Івана Франка
 Львівський національний технічний університет «Львівська політехніка»
 Кременчуцький національний університет ім. М. Остроградського

СУЧАСНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ В УПРАВЛІННІ

Збірник матеріалів

V Міжнародної науково-практичної конференції
 молодих вчених, аспірантів і студентів
 (Інтернет-конференція)

18–19 квітня 2024 р.

Новак К.М.

студентка

Київський національний економічний
університет імені Вадима Гетьмана

ekaterinanikolaevna01@ukr.net

ІННОВАЦІЙНІ ПІДХОДИ ДО ЗАХИСТУ ДАНИХ У BIG DATA

У сучасному цифровому світі, де обсяги даних швидко зростають, захист особистої та конфіденційної інформації стає пріоритетною задачею для багатьох організацій та користувачів. Зокрема, в сфері Big Data, де обробка та аналіз великих обсягів даних забезпечується за допомогою різних інноваційних технологій, виникають нові виклики у забезпеченні безпеки і конфіденційності.

Термін "великі дані" (Big Data) використовується для опису дуже великих обсягів даних, що перевищують можливості традиційних методів обробки та аналізу. Великі дані характеризуються трьома основними аспектами, відомими як "3V": об'єм (volume), швидкість (velocity) та різноманітність (variety) [1].

Приклади джерел Big Data є безліч і вони демонструють потенціал для застосування даних у різних галузях та сферах діяльності, наприклад:

- Медичні дані пацієнтів (дані про стан здоров'я пацієнтів, результати обстежень, лікування та знімки з апаратів MRI/CT)
- Транспортні системи міст (дані від GPS-сигналів та камер спостереження можуть бути використані для аналізу трафіку, руху транспортних засобів та оптимізації міської інфраструктури)
- Дані соціальних мереж (соціальні мережі надають великий потік даних про активність користувачів, їхні взаємодії та інтереси, що може бути використано для аналізу трендів та розуміння поведінки аудиторії)
- Геологічні дослідження (дані з сейсмічних датчиків, геологічних обстежень та інших джерел використовуються для дослідження геологічних процесів, виявлення нафтових і газових родовищ, прогнозування землетрусів) [2].

Проблеми безпеки даних виникають на всіх етапах їх обробки – у процесі генерації, передачі, зберігання, аналізу та візуалізації. Найбільший внесок у розвиток технологій безпеки даних Big Data роблять: International Cloud Security Альянс (CSA), Національний інститут стандартів і технологій США (NIST) та Агентство Європейського Союзу з безпеки мережі та інформації (ENISA).

Відомі компанії на ринку IT приділяють значну увагу розробці спеціалізованих рішень щодо захисту даних Big Data, найвідоміші з них перераховані в таблиці 1.

Таблиця 1 – Рішення щодо забезпечення захисту BIG DATA

Розробник	Рішення
IBM	Top tips for Big Data Security
Oracle	Enterprise Security for Big Data Environments
Forrester	Big Data Security Strategies For Hadoop Enterprise Data Lakes
Cloudera	Cloudera Security
Securosis	Securing Hadoop: Security Recommendations for Hadoop Environment
CSA	The Big Data Security and Privacy Handbook
NIST	Big Data Interoperability Framework

Джерело даних: авторська розробка

Традиційні методи захисту даних не завжди ефективні для роботи з Big Data. Масштабність, складність та гнучкість цих систем роблять їх більш вразливими до кібератак. Тому потрібні інноваційні підходи, які зможуть гарантувати безпеку Big Data.

До таких підходів належать:

- Аналіз аномалій: виявлення незвичайних патернів у поведінці користувачів або даних для раннього реагування на загрози.
- Машинне навчання: прогнозування та запобігання кібератакам за допомогою алгоритмів машинного навчання.
- Blockchain: забезпечення прозорості та цілісності даних за допомогою технології blockchain (децентралізованої бази даних, що складається з ланцюжка блоків, де кожен блок містить інформацію про попередній блок. Ця структура робить дані на блокчейні незмінними та прозорими для всіх учасників мережі).

Окрім інноваційних методів, важливо також використовувати традиційні методи захисту даних [3]:

- Шифрування даних (AES, RSA, ABE, IBE): захист даних від несанкціонованого доступу з допомогою шифрування, тобто перетворення даних з їхнього звичайного читабельного формату в код, який може бути прочитаний лише тими, хто має ключ дешифрування.
- Контроль доступу на основі ролей: надання користувачам доступу до даних лише на основі їх ролей та потреб.
- Токенізація даних: заміна конфіденційних даних токенами для збереження їх конфіденційності.
- Псевдонімізація: заміна особистих даних псевдонімами для захисту приватності.

Впровадження інноваційних підходів до захисту даних потребує комплексного підходу, тобто поєднання традиційних методів захисту з інноваційними технологіями, обирати методи захисту потрібно з урахуванням специфіки Big Data та потреб організації, проводити моніторинг нових загроз та оновлювати методи захисту.

Отже, після дослідження питання безпеки, проведення аналізу рішень світових розробників щодо забезпечення захисту даних, було встановлено, що інноваційні підходи до захисту даних є ключовими для забезпечення безпеки Big Data. Використання нових методів та технологій, а також комбінування їх з традиційними методами може допомогти організаціям захистити свої дані від кібератак, витоків інформації та інших загроз.

Список використаних джерел

1. BIG DATA: матеріал з Вікіпедії [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%BB%D0%B8%D0%BA%D1%96_%D0%B4%D0%B0%D0%BD%D1%96
2. Приклади застосування BIG DATA: матеріал з MitSloan [Електронний ресурс]. Режим доступу: <https://sloanreview.mit.edu/article/how-big-data-is-empowering-ai-and-machine-learning-at-scale/>
3. Big Data Security and Privacy Handbook: 100 Best Practices in Big Data Security and Privacy. Cloud Security Alliance. [Електронний ресурс]. Режим доступу: https://downloads.cloudsecurityalliance.org/assets/research/big-data/BigData_Security_and_Privacy_Handbook.pdf

Науковий керівник: Денісова О.О. – кандидат економічних наук, доцент

Фрагменти програмного коду

```

<!-- Головна сторінка з перевагами системи -->
<section id="home" class="hero active">
  <h2>Моніторинг робочого часу працівників</h2>
  <p>Система для зручного обліку активності працівників в онлайн-
режимі</p>
  <!-- Кнопка входу до системи -->
  <a href="#" class="btn" data-section="login">Увійти в систему</a>

<!-- Переваги системи у вигляді карток -->
<div class="features">
  <div class="card">
    <!-- Іконка переваги -->
    
    <h3>Відстеження часу</h3>
    <p>Фіксація робочого часу, перерв, активності, та простою.</p>
  </div>
  <div class="card">
    
    <h3>Автоматичні звіти</h3>
    <p>Генерація звітів у форматах PDF та Excel за вибраний період.</p>
  </div>
  <div class="card">
    
    <h3>Безпечний доступ</h3>

```

```

    <p>Авторизація, ролі та захист персональних даних
користувачів.</p>

```

```

    </div>

```

```

  </div>

```

```

</section>

```

```

<!-- Сторінка звітів з інтерактивними кнопками -->

```

```

<section id="reports" class="content">

```

```

  <h2>Звіти</h2>

```

```

  <p>Оберіть один із доступних типів звітів:</p>

```

```

  <div class="features">

```

```

    <!-- Картка звіту за часом -->

```

```

    <div class="card">

```

```

```

```

      <h3>Звіт за часом</h3>

```

```

      <p>Аналіз кількості відпрацьованих годин за день, тиждень або
місяць.</p>

```

```

      <!-- Кнопка перегляду -->

```

```

      <a href="#" data-section="report" class="btn">Переглянути</a>

```

```

    </div>

```

```

    <!-- Інші типи звітів оформлені аналогічно -->

```

```

  </div>

```

```

</section>

```

```

<!-- JS: Побудова графіку звіту за часом за допомогою Chart.js -->

```

```

<script>

```

```

  const ctx1 = document.getElementById('chartHours').getContext('2d');

```

```

  new Chart(ctx1, {

```

```

    type: 'bar', // Тип діаграми

```

```

data: {
  labels: ['Іван Іванов', 'Катерина Новак', 'Сергій Сергійов', 'Марина
Маринів'],
  datasets: [
    {
      label: 'Продуктивно',
      data: [9.3, 8.5, 8.1, 8],
      backgroundColor: '#4caf50',
      stack: 'Stack 0'
    },
    {
      label: 'Невідомо/нейтрально',
      data: [0.2, 0.3, 0.2, 0.4],
      backgroundColor: '#ccc',
      stack: 'Stack 0'
    },
    {
      label: 'Непродуктивний час',
      data: [0.4, 0.2, 0.7, 0.1],
      backgroundColor: '#e53935',
      stack: 'Stack 0'
    }
  ]
},
options: {
  responsive: true,
  plugins: {
    title: {
      display: true,
      text: 'Звіт за часом'
    }
  }
}

```

```

    },
    tooltip: {
      callbacks: {
        footer: function(context) {
          const total = context.reduce((sum, el) => sum + el.raw, 0);
          const waste = context.find(el => el.dataset.label ===
'Непродуктивный час');
          const wasteTime = waste ? waste.raw : 0;
          const percent = ((wasteTime / total) * 100).toFixed(1);
          return `Загальний час: ${total.toFixed(2)} год\nНепродуктивный
час: ${wasteTime} год (${percent}%)`;
        }
      }
    },
    interaction: {
      mode: 'index',
      intersect: false
    },
    scales: {
      x: { stacked: true },
      y: {
        stacked: true,
        title: {
          display: true,
          text: 'Години'
        }
      }
    }
  }
}

```

```
});  
</script>
```