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

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

Форма навчання: денна

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

на тему:

РОЗРОБКА НАВЧАЛЬНОЇ ПЛАТФОРМИ З ВИКОРИСТАННЯМ НОВІТНІХ ТЕХНОЛОГІЙ МАШИННОГО НАВЧАННЯ ДЛЯ ПЕРЕВІРКИ ЗНАНЬ СТУДЕНТІВ

здобувача Уманського Дмитра Борисовича _____
(ПІБ) *(Підпис)*

Науковий керівник:
к.е.н., доцент
Лозовик Ю. М.

Робота допущена до захисту перед екзаменаційною комісією з атестації здобувачів вищої освіти
завідувач кафедри:
к.е.н., доцент
_____Тішков Б.О.

Київ 2025

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

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

ПОГОДЖЕНО:
Керівник проектної групи (гарант) освітньо-професійної програми
_____Помазун О.М.
"_____" _____2025 р.

ЗАТВЕРДЖЕНО:
Завідувач кафедри інформаційних систем в економіці
_____Тішков Б.О.
"___" _____2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувача вищої освіти
Уманського Дмитра Борисовича
очної (денної) форми навчання

на підготовку кваліфікаційної бакалаврської роботи
на тему: ***«Розробка навчальної платформи з використанням новітніх технологій машинного навчання для перевірки знань студентів»***

Тему затверджено наказом ректора Університету від «7» березня 2025 р. № 529-ст.

Кваліфікаційна бакалаврська робота виконується на матеріалах отриманих під час навчання, у процесі проведення студентських досліджень, а також шляхом аналізу наукових джерел з обраної тематики

План кваліфікаційної бакалаврської роботи

Розділ 1. Характеристика та аналіз предметної галузі
Розділ 2. Структура інформаційної підсистеми
Розділ 3. Проектування та реалізація компонентів підсистеми
Об'єкт дослідження: новітні веб-технології та технології машинного навчання для

розробки навчальних платформ (на прикладі навчальної онлайн-платформи).

Предмет дослідження: методи та технології розробки навчальної онлайн-платформи з використанням сучасних веб-рішень (Django, React.js тощо), а також впровадження існуючих моделей машинного навчання для перевірки академічної доброчесності студентів.

Мета випускного бакалаврського проекту: створення прототипу навчальної онлайн-платформи зі зручним інтерфесом, можливістю створення нових курсів, виконання та здачі робіт, а також їх перевірки на ШІ-згенерований контент. Примітка: У роботі реалізовано частковий візуальний прототип, а також використано існуючу моделі для класифікації тексту. Повністю функціональної платформи не передбачено.

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

У розділі 1 Характеристика та аналіз предметної галузі виконати наступні завдання:

Дослідити ринок онлайн-навчання, провести аналіз різних платформ. Визначити основні функціональні та нефункціональні вимоги до власної платформи. Провести аналіз існуючих онлайн-платформ для навчання, виокремити їхні основні переваги та недоліки. Дослідити наукові публікації, статті та дослідження відносно розробки навчальних або подібних платформ. Провести аналіз сучаних технологій для розробки такого виду веб-застосунків.

У розділі 2 Структура інформаційної підсистеми виконати наступні завдання:

Сформолювати функціональні та нефункціональні вимоги до системи. Сформулювати технічні та бізнес-вимоги до функціоналу. Чітко сформулювати завдання розробки, виходячи з проведеного аналізу. Визначити обсяг робіт та етапи впровадження прототипу. Розробити візуальні представлення системної архітектури з деталізацією складових компонентів і їх взаємозв'язків. Розробити потрібні UML-діаграми.

У розділі 3 Проєктування та реалізація компонентів підсистеми виконати наступні завдання:

Визначити оптимальну модель зберігання інформації (реляційна чи документоорієнтована база даних). Спроєктувати схему бази даних з основними сутностями. Визначити стек технологій. Обґрунтувати вибір хостингу та API.

Детально зобразити аргументований вибір інструментів розробки. Розкрити особливості організації API. Описати процес розробки повноцінної інформаційної системи. Створення моделі машинного навчання.

Завдання підготував науковий керівник _____ **Лозовик Юрій Миколайович**

«10» березня 2025 р.

Завдання одержав студент **Уманський Дмитро Борисович**

_____

«10» березня 2025 р.

Відгук
на кваліфікаційну бакалаврську роботу
здобувачки навчально-наукового інституту
«Інститут інформаційних технологій в економіці»
освітньо-професійної програми
«Комп'ютерні науки»

Уманського Дмитра Борисовича
на тему ***«*Розроблення онлайн-платформи для перевірки знань студентів з використанням новітніх технологій машинного навчання*»***

1. Актуальність теми дипломної роботи зумовлена тим, що більшість існуючих платформ онлайн-навчання зосереджені лише на поданні навчального матеріалу та простому тестуванні, не враховуючи можливості автоматизованої перевірки відкритих текстових відповідей. Застосування технологій машинного навчання, зокрема трансформерних моделей, відкриває нові можливості у сфері автоматизованої оцінки знань. Інтеграція таких моделей із сучасними вебтехнологіями, як Django і React, дозволяє створити функціональні, зручні у використанні платформи нового покоління. Це дозволяє значно знизити навантаження на викладачів, підвищити об’єктивність оцінювання та забезпечити індивідуальний підхід до навчання. Дослідження присвячене проєктуванню підсистеми автоматизованої перевірки знань з використанням інструментів ШІ та сучасних вебфреймворків, є надзвичайно актуальним у контексті розвитку цифрової освіти та інноваційних освітніх технологій.

2. Позитивні риси кваліфікаційної магістерської роботи:
Автором проведено ґрунтовне теоретико-практичне дослідження та аналіз підходів щодо створення інформаційної системи онлайн-платформи для перевірки знань студентів з використанням новітніх технологій машинного навчання, проаналізовані аналоги програмних рішень, визначені їх переваги та недоліки та здійснено проектування власної системи. Автор роботи чітко визначив функціональні та нефункціональні вимоги до системи, розробив основні UML-діаграми (use-case, діяльності, послідовностей, класів, компонентів, трасування, прецедентів), даталогічну модель БД, необхідні для реалізації власної інформаційної системи. У роботі застосовано популярні й перевірені технології веб-розробки — Django (бекенд) та React.js (фронтенд), що дозволяє створити масштабовану, гнучку та зручну для користувача систему.

3. Наявність самостійних розробок автора
Робота виконана самостійно. У роботі чітко розмежовано текст автора та запозичення, наявні посилання на використані джерела. Автор роботи, **Уманського Дмитра Борисовича,** проаналізовані наявні аналоги, спроєктовано власну онлайн-платформи для перевірки знань студентів з використанням новітніх технологій машинного навчання. Значну увагу приділено етапу проєктуванню системи. Використання попередньо навченої трансформерної

моделі RoBERTa для класифікації текстів додає роботі інноваційності, а також демонструє можливість автоматизованої перевірки відповідей на основі ШІ.

4. Цінність теоретичних висновків та практичних рекомендацій:
Автором опрацьовано значний обсяг інформації, здійснено систематизацію та узагальнення відомостей щодо проєктування і розробки прототипу онлайн-платформи для перевірки знань студентів з використанням новітніх технологій машинного навчання.

5. Наявність недоліків:

- відсутність економічного обґрунтування створення нової інформаційної системи, а також порівняння розробленої системи з наявними аналогами, проаналізованими в теоретичному розділі роботи;
- відсутній опис серверної частини застосунку, оскільки основний акцент зроблений на проєктування;
- відсутнє модульне, системне та інтеграційне тестування серверної частини додатку;
- у роботі наведені лише візуальні тест-кейси.

6. Загальна оцінка кваліфікаційної магістерської роботи та її допущення до захисту перед ЕК:

Зазначене вище дозволяє констатувати, що дипломна робота **Уманського Дмитра Борисовича** є самостійно завершеним дослідженням. Автор роботи вміє працювати з технологіями проєктування та програмувати інформаційну систему онлайн-платформи для перевірки знань студентів з використанням новітніх технологій машинного навчання. Дипломна робота **Уманського Дмитра Борисовича** відповідає вимогам щодо написання та оформлення бакалаврських робіт і може бути рекомендована до захисту в ЕК. Робота може бути оцінена на відміно (95 б).

Науковий керівник,
доцент кафедри інформаційних систем
в економіці, к. е. н. Лозовик Ю.М.

_____ _____
(підпис) *(прізвище, ініціали)*

“13” червня 2025 р.

РЕЦЕНЗІЯ

на кваліфікаційну бакалаврську роботу
здобувача вищої освіти
Уманського Дмитра Борисовича
Тема:
«**Розроблення онлайн-платформи для перевірки знань студентів з використанням новітніх технологій машинного навчання**»

Актуальність теми кваліфікаційної бакалаврської роботи і доцільність її розроблення. Тема роботи є надзвичайно актуальною в умовах розвитку цифрової освіти. Більшість сучасних платформ навчання обмежуються традиційним тестуванням, не підтримуючи можливості інтелектуального аналізу відкритих відповідей. Застосування технологій машинного навчання, зокрема трансформерних моделей, дозволяє реалізувати автоматизовану перевірку знань, що значно підвищує об'єктивність оцінювання. Доцільність розроблення подібної системи не викликає сумнівів, адже вона спрямована на вирішення конкретних проблем сучасної освіти та оптимізацію роботи викладачів.

Якість проведеного дослідження. Автором проведено глибокий теоретичний аналіз наукових джерел та наявних аналогів онлайн-платформ. Визначено їхні переваги та недоліки, обґрунтовано вибір інструментів для реалізації власного рішення. Робота включає чітке формування функціональних та нефункціональних вимог, розробку UML-діаграм, створення бази даних та реалізацію прототипу системи. Використання сучасного стеку технологій (Django, React, RoBERTa) засвідчує якість проведеної розробки та здатність автора працювати з актуальними інструментами.

Позитивні риси кваліфікаційної бакалаврської роботи:

- високий рівень самостійності виконання проєкту;
- інноваційний підхід до реалізації задачі перевірки знань;
- обґрунтоване застосування технологій машинного навчання;
- наявність детального проєктування (uml-діаграми, даталогічна модель БД);
- створення функціонального прототипу системи із застосуванням сучасних вебтехнологій;
- практична спрямованість і можливість подальшого використання результатів у навчальному процесі.

Зауваження:

- у роботі відсутній економічний розділ із обґрунтуванням витрат на розробку;
- не подано детального опису серверної частини системи;
- відсутнє модульне та системне тестування функціоналу;
- обмежено використання інструментів порівняльного аналізу розробленої системи з аналогами;

- тестування представлене лише у вигляді візуальних кейсів без технічного обґрунтування.

Практична значимість висновків і рекомендацій. Отримані результати можуть бути використані для розробки й впровадження подібних навчальних систем у вищих навчальних закладах. Реалізований прототип демонструє приклад ефективної інтеграції машинного навчання у сферу освіти. Запропоновані підходи можуть бути адаптовані для різних предметних галузей та рівнів підготовки. Практична значущість також полягає у потенціалі масштабування платформи для масового використання.

Зав. відділення «Програмування»
Відокремленого структурного підрозділу
«Фаховий коледж інформаційних
систем і технологій» _____ Світлана Миколаївна ЯНЧУК

“13” червня 2025 р.

АНОТАЦІЯ

Кваліфікаційної бакалаврської роботи студента 4 курсу
Навчально-наукового інституту «Інституту інформаційних технологій в економіці»
Уманського Дмитра Борисовича, виконаної на тему:
«Розробка навчальної платформи з використанням новітніх технологій машинного навчання для перервіки знань студентів»
Київ: кафедра інформаційних систем в економіці, 2025р.

Ця робота присвячена дослідженню процесу проєктування та теоретичному обґрунтуванню вибору архітектури для підсистеми автоматизованої перевірки знань. Увагу було зосереджено на інтеграції найсучасніших технологій веб-розробки (таких як Django для бекенду та React.js для фронтенду) з потужними інструментами машинного навчання для класифікації тексту. Зокрема, було використано попередньо навчену трансформерну модель RoBERTa від OpenAI для детекції ШІ-генерації у відповідях.

Розділ 1 дає загальну характеристику предметної області. У ньому аналізується сучасний ринок онлайн-навчання, формулюються ключові вимоги до освітніх платформ та розглядаються існуючі практичні рішення. Також досліджего переваги та недоліки обраних технологій для реалізації нашої підсистеми.

Розділ 2 детально описує процес проєктування структури інформаційної підсистеми. Тут визначено основні функціональні та нефункціональні вимоги до системи, сформульовано постановку задачі та розроблено архітектуру її ключових компонентів: користувацького інтерфейсу, серверної частини та бази даних.

Розділ 3 містить теоретичну частину проєктування та початкову реалізацію компонентів підсистеми. Детально розглянуто інформаційне, технічне та програмне забезпечення. Особлива увага приділяється процесу створення та

апробації моделі машинного навчання для класифікації тексту (зокрема, використання попередньо навченої моделі) та реалізації прототипу інтерфейсу, який демонструє ключові функціональні можливості майбутньої платформи.

За своєю суттю, ця робота має переважно теоретичний характер із практичними елементами прототипування, і вона є міцною основою для подальшої повноцінної реалізації навчальної онлайн-платформи.

РЕФЕРАТ

Кваліфікаційна бакалаврська робота містить 87 сторінок, 8 таблиць, 21 Рисунок , список літератури з 30 найменувань, 3 додатки.

Розробка навчальної платформи з використанням новітніх технологій машинного навчання для перевірки знань студентів

Перелік ключових слів: навчальна платформа, ШІ-контент, Django, React.js, трансформери, RoBERTa, BERT, веб-розробка, PostgreSQL, API.

Предметом дослідження є: методи та технології розробки навчальної онлайн-платформи з використанням сучасних веб-рішень (Django, React.js тощо), а також [методи та технології] впровадження існуючих моделей машинного навчання для перевірки академічної доброчесності студентів.

Об’єктом дослідження виступають: новітні веб-технології та технології машинного навчання для розробки навчальних платформ (на прикладі навчальної онлайн-платформи).

Мета випускного бакалаврського проекту полягає в: створенні прототипу навчальної онлайн-платформи зі зручним інтерфесом, можливістю створення нових курсів, виконання та здачі робіт, а також їх перевірки на ШІ-згенерований контент. Примітка: У роботі реалізовано частковий візуальний прототип, а також використано існуючу моделі для класифікації тексту. Повністю функціональної платформи не передбачено.

Завданнями випускного бакалаврського проекту є:

- Дослідження сучасних тенденцій у розробці навчальних онлайн-платформ та аналіз існуючих рішень.
- Обґрунтування вибору технологічного стеку (Django, React.js, REST API, PostgreSQL).
- Проєктування архітектури системи, включаючи схему бази даних, API та інтерфейсів.
- Реалізація ключових модулів: авторизація, створення курсу, приєднання до курсу, система перевірки зданої роботи на ШІ-контейнт,коментування.

Апаратні та програмні засоби, що використовувались на при проектуванні: ноутбук з ОС Windows 11 Pro (процесор Ryzen 5 5600, 16 ГБ

ОЗУ, SSD 500GB), програмне забезпечення: Google Colab, Visual Studio Code, PostgreSQL, Draw.io, Miro, Enterprise Architect.

Результати досягнуті в процесі роботи: розроблено модель для перевірки тексту на ШІ-контент, створено частковий візуальний прототи системи.

Одержані результати можуть бути використані у навчальних цілях як приклад сучасної FullStack-розробки з використанням актуальних технологій, з подальшим адаптуванням прототипу в повноцінну інформаціну систему.

Рік виконання випускного бакалаврського проекту: 2025 рік

Рік захисту випускного бакалаврського проекту: 2025 рік

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, ВЕЛИЧИН І ТЕРМІНІВ

ІЗ – інформаційне забезпечення

ІС – інформаційна система

UI – User Interface (інтерфейс користувача)

UX – User Experience (досвід користувача)

СКБД – система керування базами даних

ПЗ – програмне забезпечення

ОС – операційна система

API – Application Programming Interface

REST – Representational State Transfer (тип API)

JSON – JavaScript Object Notation

CRUD – Create, Read, Update, Delete (операції над даними)

UML – Unified Modeling Language

IDE – Integrated Development Environment

WCAG – Web Content Accessibility Guidelines

RAD – Rapid Application Development

ARM / APM – автоматизоване робоче місце

vCPU, RAM, SSD – віртуальний процесор, оперативна пам’ять, твердотільний накопичувач

LOGS, AN_DATA, SYS_ACT – внутрішні ідентифікатори повідомлень/подій системи

Django, React.js, Redux, PostgreSQL – назви технологій/бібліотек

GitHub – репозиторії для коду

Visual Studio Code – середовище розробки

ВСТУП

Актуальність теми зумовлена необхідністю модернізації освітнього процесу та зростаючим попитом на ефективні інструменти для перевірки знань студентів. Сучасні технології веб-розробки, такі як Django та React.js, у поєднанні з машинним навчанням, відкривають нові можливості для створення інтерактивних та адаптивних навчальних платформ. Це дозволяє не лише автоматизувати процес оцінювання, а й забезпечити персоналізований підхід до навчання, що є особливо важливим в умовах дистанційної освіти. Однак, питання інтеграції складних алгоритмів машинного навчання для аналізу відповідей студентів та оптимізації архітектури таких платформ залишаються недостатньо дослідженими. Саме це й зумовлює необхідність проведення даної роботи.

Аналіз літературних джерел свідчить, що питання розробки навчальних платформ активно досліджуються (Mohammed Ouadout et al., 2021). Зокрема, у працях багатьох авторів висвітлюються переваги React.js для створення інтерактивних інтерфейсів (Songtao Chen et al., 2019), Django для побудови надійного серверного функціоналу (Songtao Chen et al., 2020), а також використання Python для розробки моделей машинного навчання (Sebastian Raschka et al., 2020). Деякі джерела також аналізують архітектурні підходи для освітніх систем (Роман Пановик et al., 2025). Проте, специфіка застосування цих технологій, доповнених машинним навчанням, для створення навчальних платформ, орієнтованих саме на перевірку знань студентів, потребує додаткового, глибшого вивчення.

Об'єктом дослідження виступає процес проєктування навчальних платформ для перевірки знань студентів.

Предметом дослідження є методика створення інтерактивного прототипу такої платформи з використанням Django та React.js, а також застосуванням машинного навчання для перевірки відповідей студентів.

Метою роботи є розробка концептуального прототипу навчальної платформи, що демонструє можливості інтеграції сучасних веб-технологій та алгоритмів машинного навчання для ефективної перевірки знань.

Для досягнення поставленої мети визначено низку завдань: дослідження вимог до функціоналу навчальних платформ з перевірки знань, обґрунтування архітектурних рішень із застосуванням Django та React.js, а також створення інтерактивного прототипу інтерфейсу та розробка підсистем машинного навчання для аналізу відповідей. У процесі дослідження використовувалися такі методи як аналіз літературних джерел, порівняльна оцінка технологій, вербально-описовий аналіз UI/UX рішень та прототипування.

Теоретична значущість роботи полягає в систематизації підходів до інтеграції машинного навчання з сучасними веб-технологіями для створення освітніх платформ. Практична цінність проявляється у створенні прототипу, готового до впровадження, який демонструє ефективність автоматизованої перевірки знань. Робота має чітку структуру, що включає три розділи, присвячені аналізу предметної галузі, проєктуванню архітектури та опису прототипу.

Важливо зазначити, що через обмежені часові рамки дослідження зосереджено переважно на UI/UX аспектах та інтеграції машинного навчання, без повної реалізації всієї Back-end частини. Проведений огляд літератури підтверджує актуальність теми, але водночас вказує на необхідність подальших досліджень у сфері оптимізації інтерфейсів для навчальних платформ та удосконалення алгоритмів перевірки знань. Запропонований прототип є важливим першим кроком у створенні повноцінного рішення для автоматизованої перевірки знань студентів.

РОЗДІЛ 1
ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1. Характеристика предметної галузі та об’єкта дослідження

Цей проєкт (та відповідна предметна область) зосереджений на сфері освітніх технологій, а саме на розробці навчальної платформи для автоматизованої перевірки знань студентів. У роботі детально розглядається, як така система може приймати та аналізувати відповіді, керувати різноманітними навчальними матеріалами та забезпечувати зворотний зв'язок для користувачів. Основна ідея полягає в автоматизації процесу оцінювання, що дозволить значно скоротити час на перевірку робіт і покращити якість навчання.

Основними бізнес-процесами такої системи є: формування навчальних завдань, збір відповідей, обробка даних за допомогою машинного навчання, а також відображення результатів. Для успішної роботи критично важливим є безперебійне узгодження таких підсистем, як генерація тестів, аналіз відповідей студентів, керування обліковими записами користувачів, зберігання навчальних даних та інтеграція зворотного зв'язку. Завдяки сучасним технологіям Django та React.js, доповненим можливостями Python для моделей машинного навчання, ми можемо автоматизувати значну частину цих етапів, що призведе до швидшої та точнішої перевірки знань і, як наслідок, до підвищення ефективності освітнього процесу.

Організаційна структура навчальної платформи з перевірки знань передбачає чітке розмежування відповідальності між адміністраторами, викладачами та студентами.

Категорія адміністраторів поділяється на кілька спеціалізованих ролей:

- Системний адміністратор: Відповідає за технічне функціонування платформи. Це включає підтримку серверів, забезпечення

безперебійної роботи Django (для бекенду) та React.js (для фронтенду), а також інтеграцію та моніторинг моделей машинного навчання на Python, що використовуються для перевірки знань. Вони стежать за продуктивністю та розв'язують будь-які технічні несправності.

- Адміністратор контенту: Займається управлінням усіма навчальними матеріалами. Ця роль включає допомогу викладачам у завантаженні та структуруванні курсів, тестів та інших освітніх ресурсів, а також перевірку правильності форматування завдань для ефективної роботи систем машинного навчання.
- Адміністратор користувачів: Керує обліковими записами всіх користувачів платформи – студентів та викладачів. Це передбачає реєстрацію, редагування даних, управління правами доступу до курсів та функцій платформи.

Роль викладачів полягає у створенні та керуванні навчальним контентом. Вони завантажують матеріали, формують тестові завдання, налаштовують параметри оцінювання та взаємодіють з моделями машинного навчання для аналізу відповідей студентів. Викладачі також можуть переглядати результати та надавати додатковий зворотний зв'язок.

Студенти – це безпосередні користувачі платформи. Вони взаємодіють з інтерфейсом для доступу до навчальних матеріалів, виконання завдань та отримання результатів автоматизованої перевірки своїх знань.

Функціональна структура організації чітко розмежовує завдання для кожної посадової особи. Системний адміністратор відповідає за технічну підтримку платформи. Адміністратор контенту оновлює інформацію щодо доступних навчальних матеріалів. Адміністратор користувачів веде облік студентів та викладачів. Викладач консультує студентів, відповідає на питання щодо процесу навчання тощо. Для прийняття рішень використовується інформація про попит на різні навчальні курси, складність завдань, успішність студентів та інші освітні метрики. Ці дані є важливими для коригування навчальної програми, визначення оптимального рівня складності та оптимізації

ресурсів платформи. Додатково використовується інформація для формування звітів про прогрес навчання, аналізу ефективності методик викладання та розробки освітніх стратегій. Інформація для прийняття рішень складається не лише з фактичних результатів, а й з прогнозів на основі попередніх успіхів студентів та аналізу їхніх потреб. Документообіг включає збереження та обробку даних про успішність студентів, результати тестів, навчальні матеріали та запити користувачів, що зберігаються в базі даних та можуть бути відсортовані за різними критеріями. Множина інформаційних сутностей включає студентів, викладачів, навчальні курси, завдання, результати тестів та інші елементи. Відносини між цими сутностями включають залежність між успішністю студентів, пройденими курсами та їхніми оцінками. Проблеми існуючих навчальних систем часто пов'язані з відсутністю інтеграції між процесами подачі матеріалів, перевірки знань та надання зворотного зв'язку. Це може призводити до затримок у оцінюванні, неточностей у відстеженні прогресу та зниження задоволеності студентів від навчального процесу.

Потреба вдосконалення інформаційної системи навчальної платформи обумовлена прагненням оптимізувати процеси, зменшити кількість помилок та підвищити якість перевірки знань. Впровадження нової системи дозволить автоматизувати ключові етапи оцінювання, забезпечити доступ до актуальних навчальних матеріалів та підвищити загальну ефективність освітнього процесу.

У сучасному освітньому просторі навчальна платформа має бути не просто інструментом для надання завдань, а комплексною системою, яка охоплює всі аспекти взаємодії зі студентами та їхнього прогресу. Від ефективності цієї платформи безпосередньо залежать такі ключові показники, як рівень засвоєння знань, мотивація студентів, швидкість отримання зворотного зв'язку та загальна якість освітніх послуг. Тому розробка та вдосконалення інформаційних систем для цієї галузі, особливо з інтеграцією Django, React.js та машинного навчання на Python, є особливо актуальним та перспективним напрямком.

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

У сфері онлайн-навчання існує декілька популярних систем, що фокусуються на перевірці знань студентів та використовуються для автоматизації процесу тестування, управління навчальнимим матеріалами та забезпечення ефективного зворотного зв’язку. Серед них можна виділити низку найпопулярнігих: Moodle, Google Classroom та Coursera. Вони демонструють різні підходи до автоматизації та керування навчальним процесом і оцінюванням знань.

Moodle:

Moodle – це провідна відкрита система управління навчанням (LMS), спеціально розроблена для створення динамічних онлайн-курсів та інтерактивного освітнього середовища. Вона надає викладачам потужні інструменти для організації навчального процесу, а студентам – зручний доступ до матеріалів та можливість активної взаємодії. Це дозволяє легко управляти контентом, адаптувати його під різні потреби та забезпечувати персоналізований навчальний досвід.

Основними релевантними до даної роботи функціями Moodle є:

- Автоматизація завдань та тестів: Moodle дозволяє викладачам створювати різноманітні типи завдань, включаючи тести з автоматичною перевіркою (множинний вибір, відповідність, "вірна/хибна" відповідь). Це значно спрощує процес оцінювання та дозволяє студентам отримувати миттєвий зворотний зв'язок.
- Управління навчальним контентом: Система забезпечує централізоване сховище для всіх навчальних матеріалів – лекцій, презентацій, відео, статей. Викладачі можуть легко завантажувати, організовувати та оновлювати контент, а студенти мають до нього постійний доступ.
- Інструменти для оцінювання та зворотного зв'язку: Окрім автоматичних

тестів, Moodle підтримує ручну перевірку завдань, оцінювання за критеріями та надає широкі можливості для надання детального зворотного зв'язку.

- Збір даних для аналітики: Moodle збирає детальну статистику про активність студентів, їхній прогрес, результати тестів та взаємодію з курсовими матеріалами. Ці дані є безцінними для аналізу ефективності навчання, виявлення проблемних зон та адаптації навчальних програм.

Проте, варто зазначити, що хоча Moodle є дуже гнучкою та масштабованою платформою, її відкритий характер та широкий функціонал можуть вимагати значних зусиль для налаштування та інтеграції спеціалізованих рішень, таких як наші моделі машинного навчання на Python для розширеної перевірки знань. Це може бути викликом для команд без досвіду роботи з її внутрішньою архітектурою.

Google Classroom – це хмарна платформа, розроблена Google для спрощення процесу навчання, викладання та оцінювання, як у класі, так і дистанційно. Вона інтегрована з іншими сервісами Google Workspace (Docs, Drive, Calendar, Meet), що робить її надзвичайно зручним та доступним інструментом для викладачів та студентів. Її простота використання та інтуїтивно зрозумілий інтерфейс дозволяють швидко розгорнути навчальний процес і забезпечити централізоване управління завданнями та комунікацією.

Основні функції включають:

- Централізоване управління завданнями: Викладачі можуть легко створювати, розподіляти та збирати завдання, а також встановлювати терміни здачі. Студенти отримують сповіщення про нові завдання та можуть подавати свої роботи безпосередньо через платформу.
- Оцінювання та зворотний зв'язок: Google Classroom дозволяє викладачам швидко перевіряти роботи студентів, додавати коментарі та виставляти оцінки.
- Організація навчального контенту: Платформа дозволяє організовувати навчальні матеріали за темами, завантажувати різноманітні файли

(документи, презентації, відео) з Google Drive та ділитися ними зі студентами. Це спрощує структурування інформації, яка може бути використана для формування навчальних завдань.

- Інтеграція та доступність: Будучи хмарним рішенням, Google Classroom доступний з будь-якого пристрою, що має підключення до інтернету. Його інтеграція з іншими сервісами Google робить обмін файлами та спільну роботу безшовною.

Незважаючи на свою зручність та широке поширення, Google Classroom є більш орієнтованим на спрощення базових організаційних процесів в освіті. Він не надає вбудованих інструментів для просунутої автоматизованої перевірки знань на основі машинного навчання.

Coursera – це провідна глобальна платформа для онлайн-освіти, яка співпрацює з університетами та компаніями по всьому світу, пропонуючи широкий спектр курсів, спеціалізацій та ступенів. Вона фокусується на високоякісному контенті та надає можливості для самостійного навчання та розвитку професійних навичок.

До основних функцій належать:

- Масштабована перевірка знань: Coursera активно використовує різні методи оцінювання, включаючи тести з множинним вибором, програмовані тести (для IT-курсів) та оцінювання за допомогою колег (peer-grading) для більш складних завдань. Це демонструє наявність механізмів для автоматизованої або напівавтоматизованої перевірки.
- Інтеграція з професійним контентом: Платформа співпрацює з відомими освітніми та індустріальними лідерами, що забезпечує високу якість та актуальність навчальних матеріалів.
- Персоналізований шлях навчання: Хоча це не завжди є явною функцією, Coursera пропонує певною мірою адаптивний досвід, дозволяючи студентам обирати свій темп навчання.
- Збір даних для аналізу успішності: Coursera збирає значний обсяг даних про те, як студенти взаємодіють з курсами, їхню успішність у тестах та

завданнях. Ці дані використовуються для вдосконалення курсів та виявлення закономірностей у навчанні.

Однак, незважаючи на свою потужність та інноваційність, Coursera є закритою комерційною платформою. Це означає, що можливості для глибокої кастомізації та інтеграції алгоритмів машинного навчання (розроблених на Python) є обмеженими.

Аналіз існуючих навчальних платформ, таких як Moodle, Google Classroom та Coursera, демонструє значний прогрес у сфері автоматизації ключових освітніх процесів. Це включає управління навчальним контентом, розподіл завдань та базове оцінювання. Усі ці платформи успішно оптимізують взаємодію між викладачами та студентами, забезпечуючи зручний та інтуїтивно зрозумілий інтерфейс для доступу до матеріалів та виконання завдань. Проте, кожна з цих систем має свої переваги та недоліки, особливо у контексті глибинної автоматизованої перевірки знань з використанням машинного навчання. Це відкриває значні можливості для вдосконалення та адаптації під специфічні потреби.

Moodle демонструє високу ефективність у створенні та керуванні навчальним контентом, надаючи викладачам широкі можливості для організації курсів та завдань. Його інтуїтивно зрозумілий інтерфейс для завантаження матеріалів та базового тестування дозволяє легко здійснювати оцінювання та забезпечувати зворотний зв'язок. Однак, незважаючи на його гнучкість, Moodle потребує вдосконалення в плані глибинної аналітики та персоналізації процесу перевірки знань. Наприклад, інтеграція з моделями машинного навчання на Python дозволить не тільки збирати дані про успішність студентів, але й ефективніше прогнозувати їхні прогалини в знаннях, а також створювати персоналізовані навчальні траєкторії чи додаткові рекомендації.

У системі Google Classroom важливим досягненням є простота використання та безшовна інтеграція з екосистемою Google Workspace, що дозволяє викладачам швидко створювати класи, розподіляти завдання та збирати роботи в режимі реального часу. Це значно підвищує рівень зручності для

користувачів. Однак для подальшого вдосконалення цієї платформи, особливо в контексті нашої теми, слід зосередитися на інтеграції з більш потужними інструментами для автоматизованої перевірки знань. Це дозволить знизити ризики, пов'язані з необхідністю ручної перевірки великих обсягів робіт та суб'єктивністю оцінювання.

Coursera застосовує сучасні підходи до організації навчання та взаємодії з курсовим контентом, зокрема через автоматичні нагадування та рекомендації курсів. Однак є простір для покращення персоналізації перевірки знань та адаптивного навчання. Для цього можна інтегрувати систему штучного інтелекту, яка б прогнозувала сильні та слабкі сторони студентів на основі їхніх попередніх відповідей і пропонувала спеціалізовані рекомендації щодо додаткового матеріалу або завдань. Також доцільно вдосконалити процеси моніторингу успішності студентів та автоматизації деталізованого звітування, щоб зменшити людський фактор при оцінюванні складних завдань та уникнути помилок.

Навіть попри значні успіхи в автоматизації освітнього процесу, існуючі навчальні платформи мають багато шляхів для вдосконалення, особливо у контексті глибинної перевірки знань та персоналізації навчання. Одним із основних напрямків розвитку є інтеграція більш потужних аналітичних інструментів для збору та обробки даних про навчальну поведінку студентів. Це дозволить платформам краще адаптувати свої освітні стратегії, пропонувати персоналізовані навчальні траєкторії та оптимізувати складність завдань.

Великою перевагою також стане і покращення інтеграції з іншими системами, як от платформи управління навчальним контентом та інструменти для автоматизації зворотного зв'язку. Це допоможе зменшити час на перевірку завдань, знизити ймовірність помилок в оцінюванні, зменшити навантаження на викладачів та підвищити якість взаємодії. Даний проєкт, за допомогою використання Django, React.js та Python для моделей машинного навчання, прагне реалізувати саме ці удосконалення.

Отже, попри свою функціональність, сучасні навчальні платформи мають

значний простір для вдосконалення, особливо у сфері автоматизованої перевірки знань та персоналізації навчання. Покращення аналітичних можливостей, глибша інтеграція з іншими освітніми системами та автоматизація внутрішніх процесів можуть значно підвищити ефективність навчання, знизити навантаження на викладачів та посилити залученість студентів. Автоматизація аналізу складних відповідей за допомогою машинного навчання та вдосконалення механізмів зворотного зв'язку стануть ключовими кроками до підвищення ефективності сучасних освітніх платформ.

Одним із важливих аспектів, якому приділяються значну увагу в науковій літературі є UX/UI-дизайн (Miya, Thamsanqa Keith et al., 2022). Інтуїтивний інтерфейс та зручніть взаємодії сильно сприяє успіху подібних платформ, адже це впливає на мотивацію та ефективність навчання студентів. Попереднє прототипування також значно допомагає у подальшій розробці інтерфейсів.

Також згадується використання систем аналітики та прийняття рішень, особливо з урахуванням нефінансових показників, (Norhaiza Khairudin et al., 2015). Сучасні дослідження демонструють ефективність використання даних про навчальну поведінку користувачів та результати їхньої діяльності для оптимізації освітніх процесів.

РОЗДІЛ 2
СТРУКТУРА ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ

2.1. Формування вимог до інформаційної підсистеми

Бізнес-вимоги до навчальної платформи охоплюють потреби, що визначають основні цілі її функціонування. Така система має охоплювати наступні освітні операції: управління навчальним контентом, створення та обробку завдань, спілкування з користувачами, автоматизовану перевірку знань (за допомогою машинного навчання на Python), а також адміністрування та аналіз успішності.

Інформаційна система автоматизує весь цикл надання освітніх послуг, охоплюючи обробку навчальних запитів, реалізацію проєктів (курсів/модулів) та контроль якості результатів перевірки знань. У результаті дослідження предметної області були визначені ключові групи зацікавлених сторін (внутрішні та зовнішні стейкхолдери), їхні цілі та вимоги до функціональних можливостей системи.

Зовнішні стейкхолдери:

- Студенти (користувачі, що прагнуть отримати знання або перевірити свої вміння) – основні користувачі, які взаємодіють з навчальною платформою.
- Викладачі (методисти, розробники навчальних програм) – надають необхідний контент та критерії для оцінювання.
- Освітні заклади (університети, коледжі, компанії, що проводять внутрішнє навчання) – організації, які впроваджують платформу для своїх освітніх потреб.

Бізнес-цілі зовнішніх стейкхолдерів:

Студенти:

- Отримати зручний доступ до різноманітного навчального контенту та

інтерактивних завдань.

- Мати можливість швидкого проходження завдань та отримання автоматизованого зворотного зв'язку щодо своїх знань.
- Отримати високу якість навчання, можливість відстежувати власний прогрес та персоналізовані рекомендації.

Викладачі:

- Забезпечити стабільний канал для розміщення своїх навчальних матеріалів та завдань.
- Отримати прозорі умови співпраці, зручний інтерфейс для завантаження контенту та налаштування параметрів перевірки.
- Забезпечити ефективне управління навчальними програмами та мінімізувати ризики невідповідності контенту.

Освітні заклади:

- Оптимізувати процеси перевірки знань, покращити точність оцінювання та прогнозування успішності студентів.
- Знизити кількість помилок у процесі оцінювання та отримати зручні інструменти для управління навчальними групами.
- Забезпечити інтеграцію з існуючими студентськими інформаційними системами для автоматичного отримання даних про студентів та оновлення їхнього статусу.

Внутрішні стейкхолдери:

Внутрішні стейкхолдери включають ключових учасників, відповідальних за розвиток і функціонування навчальної платформи:

- Керівництво проєкту (менеджери проєкту, інвестори) – вони приймають стратегічні рішення щодо розвитку платформи, планують витрати та оцінюють загальну успішність.
- Команда розробки (Python-розробники моделей машинного навчання, Django-розробники бекенду, React.js-розробники фронтенду, UX/UI дизайнери, тестувальники) – ця команда відповідає за розробку, підтримку та постійне вдосконалення навчальної платформи.

- Адміністратори платформи (системні адміністратори, адміністратори контенту, адміністратори користувачів) – вони керують щоденною роботою системи, забезпечують її функціональність, актуальність даних та взаємодію з користувачами.

Бізнес-цілі внутрішніх стейкхолдерів:

Керівництво проєкту:

- Забезпечити конкурентоспроможність навчальної платформи на ринку освітніх технологій.
- Збільшити кількість активних користувачів (студентів та викладачів) завдяки зручній та ефективній платформі.
- Знизити витрати на перевірку знань та управління навчальним процесом через автоматизацію за допомогою машинного навчання.

Команда розробки:

- Забезпечити стабільну та безпечну роботу навчальної платформи, використовуючи Django для бекенду та React.js для фронтенду.
- Створити інтуїтивно зрозумілий та ефективний інтерфейс для всіх типів користувачів.
- Інтегрувати передові моделі машинного навчання на Python для точної та автоматизованої перевірки знань.

Адміністратори платформи:

- Оперативно керувати навчальним контентом та обліковими записами користувачів.
- Контролювати актуальність навчальних матеріалів та завдань.
- Задовольняти вимоги користувачів (викладачів та студентів) щодо швидкості, якості та надійності роботи платформи.

Діаграма вимог (рис. 1) візуалізує взаємозв'язки між різними типами вимог, необхідних для розробки навчальної платформи. Вона показує, як бізнес-правила, функціональні вимоги та припущення пов'язані з головною вимогою до платформи – "ІС навчальної платформи", і як ці вимоги сприяють досягненню загальної мети проєкту. Загальна мета проєкту – "Надати якісний продукт, який

оцінюватиме академічну доброчесність студентів" – є головною метою, яка досягається через виконання всіх цих вимог. Це підкреслює, як усі елементи системи, розробленої на Django та React.js, працюють разом для досягнення освітніх цілей.

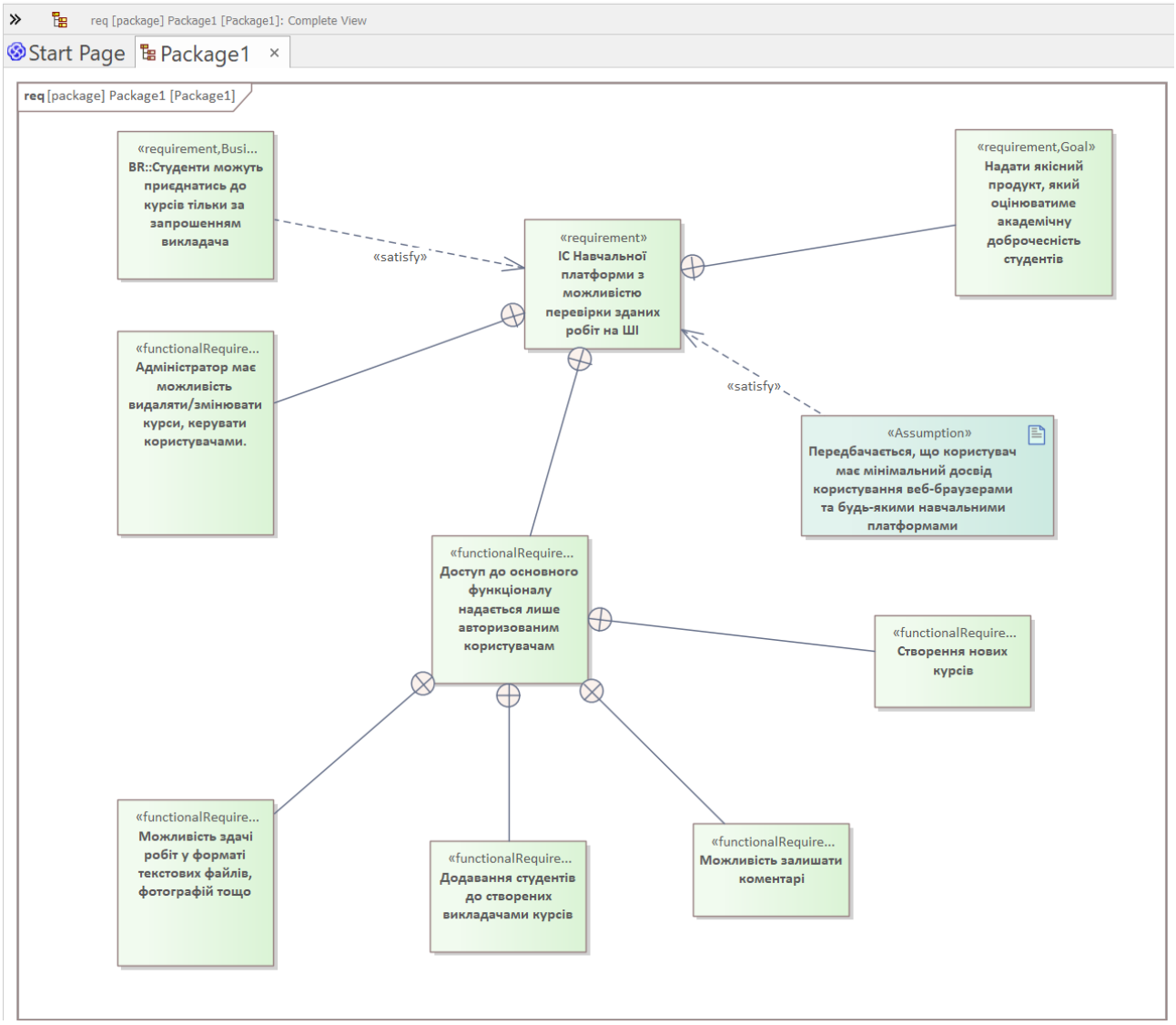


Рисунок 2.1 – Діаграма вимог

(Джерело – робота автора)

Нижче представлена таблична специфікація вимог (рис. 2), яка деталізує кожну вимогу з урахуванням її типу, статусу та пріоритету. Ця таблиця є структурованим представленням вимог до нашої навчальної платформи, що дозволяє легко відстежувати їхні характеристики. Кожна вимога, незалежно від того, чи це бізнес-правило, функціональна вимога, мета чи припущення, має свій

ідентифікатор, статус (Proposed – запропоновано) та пріоритет (високий, середній). Така деталізація допомагає у плануванні розробки, розподілі ресурсів та забезпеченні відповідності платформи всім необхідним критеріям.

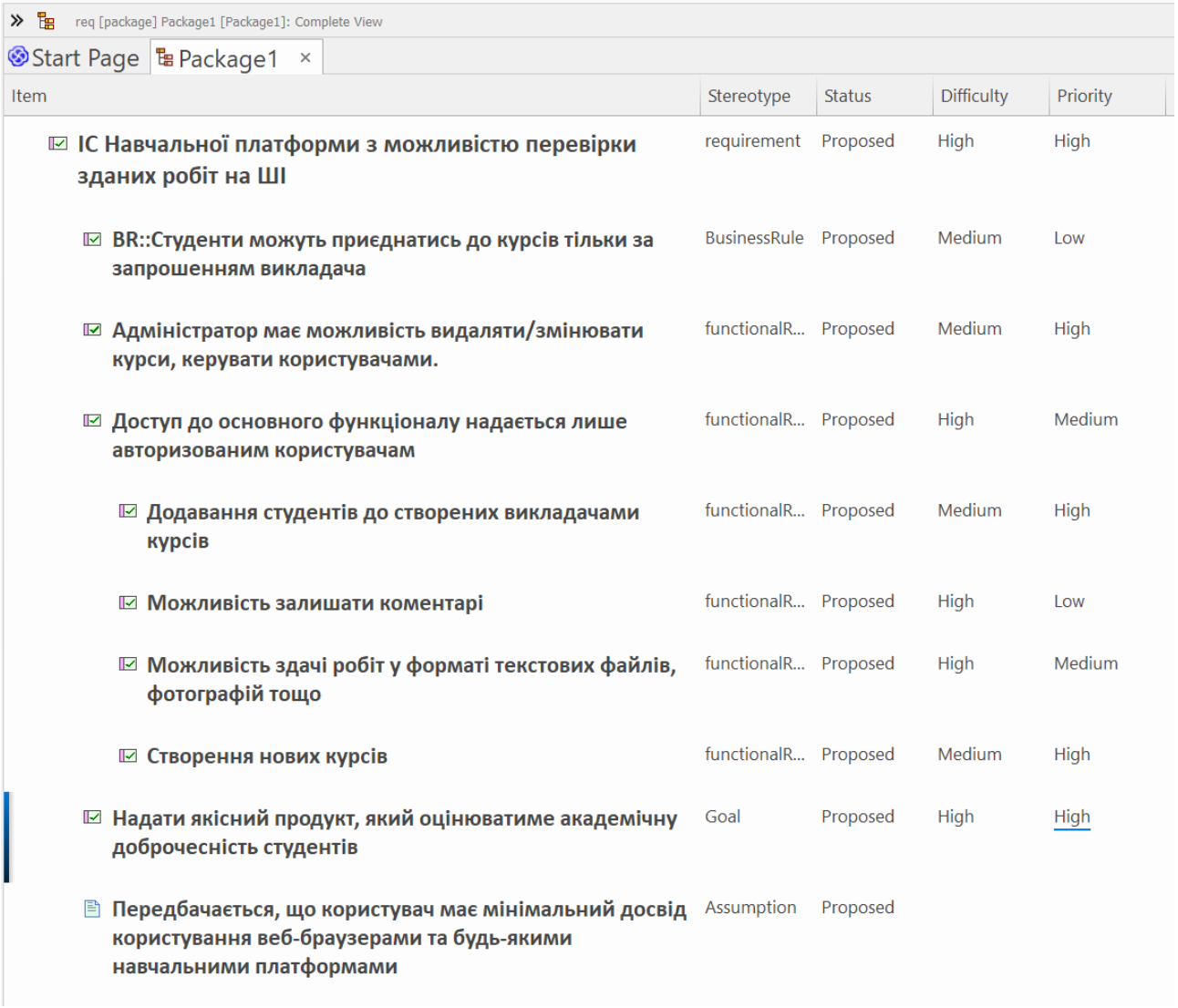


Рисунок 2.2 – Специфікація вимог

(Джерело – робота автора)

2.2. Постановка та алгоритм розв’язання задачі

2.2.1. Постановка задачі

Характеристика задачі. Призначенням задачі є створення автоматизованої навчальної платформи, яка забезпечить ефективну та

об'єктивну перевірку знань студентів з використанням передових алгоритмів машинного навчання (МН). Техніко-економічна сутність полягає в оптимізації та підвищенні якості освітнього процесу за рахунок автоматизації рутинних операцій з оцінювання, зменшення суб'єктивного фактора та надання швидкого персоналізованого зворотного зв'язку. Розв'язання цієї задачі за допомогою ЕОМ обґрунтоване високим обсягом даних (відповіді студентів, навчальні матеріали), складністю їх аналізу, потребою в масштабованості для великої кількості користувачів та неможливістю ручного опрацювання з необхідною швидкістю та точністю, особливо для завдань, що вимагають розуміння контексту (наприклад, відкриті відповіді).

Об’єкти, за управління якими розв’язується задача:

- Студенти: Облікові записи, прогрес навчання, результати тестів, відповіді на завдання.
- Викладачі: Облікові записи, створені курси, навчальні матеріали, тестові завдання, критерії оцінювання.
- Навчальні курси / Модулі: Структура, контент, доступність, терміни.
- Тестові завдання: Типи питань, варіанти відповідей, правильні відповіді, складність.
- Моделі машинного навчання: Параметри, результати аналізу відповідей, точність.
- Система оповіщень: Повідомлення для користувачів.
- Аналітичні дані: Статистика успішності, звіти про проходження курсів.

Вихідна інформація призначена для інформування студентів про їхній прогрес та результати, викладачів – про успішність груп та окремих студентів, а також для адміністраторів – про загальний стан системи та її ефективність. Вона використовується для прийняття рішень щодо корекції навчальних програм, персоналізації навчання, виявлення проблемних тем та оцінки ефективності методів викладання.

Періодичність розв’язання і термін видачі вихідної інформації:

- Перевірка завдань: Негайно після подачі студентом (для тих, що

перевіряються автоматично), або протягом кількох хвилин/годин для тих, що вимагають МН-аналізу, залежно від складності моделі та навантаження системи.

- Результати тестів та зворотний зв'язок для студентів: Миттєво або протягом декількох секунд після завершення тесту/аналізу відповіді.
- Звіти для викладачів: На вимогу (за запитом) або щоденно/щотижнево (автоматично сформовані зведені звіти).
- Аналітичні дані для адміністраторів: На вимогу (за запитом) або щомісячно/квартально (агреговані системні звіти).

Умови, за яких припиняється автоматизоване розв’язання задачі:

- Вичерпання ліміту спроб для завдання/тесту.
- Блокування облікового запису студента або викладача.
- Видалення курсу або завдання з системи.
- Технічний збій системи або несправність компонентів (наприклад, модуля МН).
- Перехід завдання в режим ручної перевірки викладачем.

Зв’язки даної задачі з іншими задачами:

- Задача управління користувачами: Інтеграція з механізмами реєстрації, авторизації, управління ролями (Django Auth System).
- Задача управління навчальним контентом: Отримання матеріалів та завдань для перевірки (Django Models).
- Задача формування звітів: Надання агрегованих даних для аналітичних звітів.
- Задача комунікації: Інтеграція з системою сповіщень (наприклад, для push-сповіщень через React.js-інтерфейс).

Розподіл дій між персоналом:

- Створення/редагування курсу/завдання: Викладач (персонал) через React.js-інтерфейс; дані передаються через Django-бекенд в/з бази даних.
- Подача відповіді студентом: Студент (персонал) через React.js-

інтерфейс; дані передаються через Django-бекенд в/з бази даних.

- Автоматизована перевірка відповіді: Модулі машинного навчання на Python (технічний засіб) отримують дані від Django-бекенду, обробляють їх та повертають результат.
- Перегляд результатів/зворотного зв'язку: Студент або викладач (персонал) через React.js-інтерфейс; дані надаються через Django-бекенд в/з бази даних.
- Ручна дооцінка/корекція результатів МН: Викладач (персонал) через React.js-інтерфейс; оновлені дані зберігаються в базі даних.
- Моніторинг системи та користувачів: Системний адміністратор (персонал) через адміністративну панель Django або кастомні інтерфейси на React.js.

Інформаційна модель задачі.

Інформаційна модель задачі автоматизованої перевірки знань студентів (на рис. 3) навчальної платформи визначає основні сутності та їхні взаємозв'язки. Вона є фундаментом для проектування бази даних та логіки взаємодії компонентів системи (Django бекенду, React.js фронтенду та Python-моделей машинного навчання).

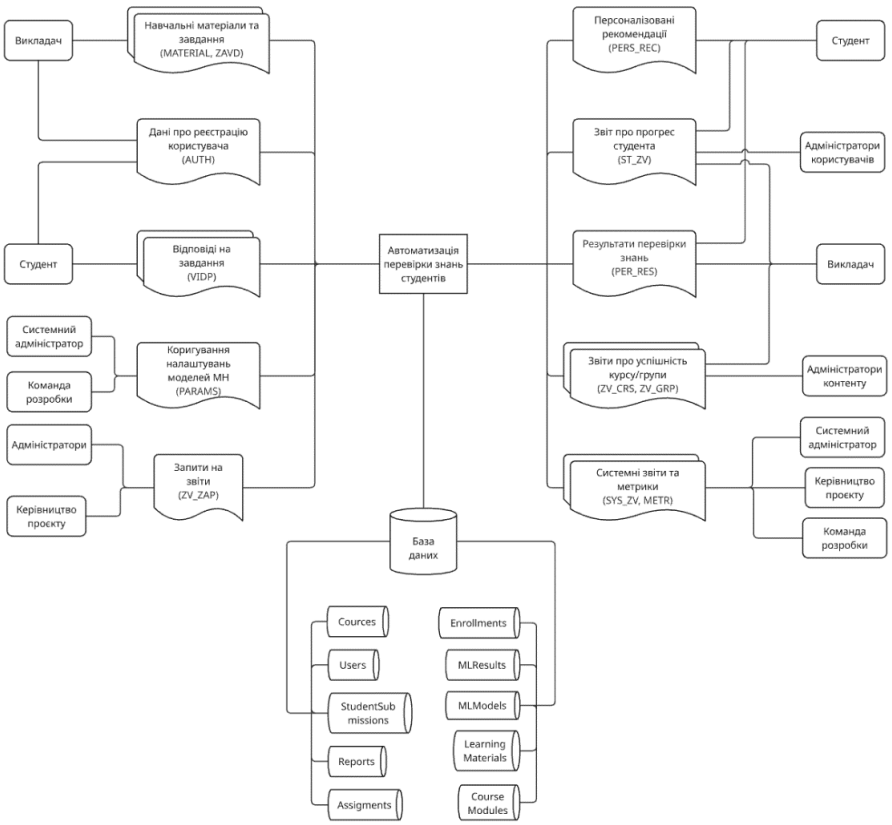


Рисунок 2.3 – Інформаційна модель

(Джерело – робота автора)

Вихідна інформація, що генерується навчальною платформою, призначена для надання актуальних даних про прогрес навчання, результати перевірки знань та ефективність системи різним категоріям користувачів. Вона використовується для прийняття обґрунтованих рішень щодо коригування навчальних програм, персоналізації освітнього процесу, виявлення проблемних зон у знаннях студентів та оцінки загальної ефективності роботи платформи. Вихідні дані формуються як у вигляді інтерактивних відеограм (відображення в React.js інтерфейсі), так і у вигляді масивів даних, що зберігаються в базі даних PostgreSQL для подальшого аналізу.

У таблиці 2.1 надано перелік і опис вихідних повідомлень.

Таблиця 2.1. Вихідні повідомлення

№ з/п	Назва вихідного повідомлення	Ідентифікатор	Форма подання і вимоги до неї	Періодичність видання	Термін видання і допустимий час затримки	Користувачі інформації
1	Результати	PER_RES	Відеограма (веб-	Після кожної	Миттєво (для	Студент,

	перевірки завдання		інтерфейс); чітке відображення балів, відсотків, статусу "здано/не здано", коментарів МН та викладача.	перевірки завдання	автоматичних); до 5 хв (для МН-аналізу); до 24 год (для ручної дооцінки викладачем)	Викладач
2	Звіт про прогрес студента	ST_ZV	Відеограма (веб-інтерфейс); таблиця з курсами, модулями, завданнями, балами, часом виконання, історією спроб. Можливість експорту в PDF/CSV.	На вимогу студента/викладача/адміністратора	До 10 секунд	Студент, Викладач, Адміністратор користувачів
3	Звіт про успішність курсу/групи	ZV_CRS, ZV_GRP	Відеограма (веб-інтерфейс); зведені дані про середній бал групи, відсоток успішності, найбільш проблемні завдання/теми. Діаграми та графіки.	На вимогу викладача/адміністратора; щотижнево (автоматично)	До 30 секунд (за запитом); до 1 години (автоматично)	Викладач, Адміністратор контенту
4	Системні метрики та статистика МН	SYS_ZV, METR	Відеограма (адмін-панель); дані про завантаження сервера, кількість активних користувачів, точність моделей МН, час обробки.	На вимогу системного адміністратора; щоденно/щомісячно (автоматично)	До 1 хвилини (за запитом); до 6 годин (автоматично)	Системний адміністратор, Команда розробки, Керівництво проєкту
5	Персоналізовані рекомендації	PERS_REC	Відеограма (веб-інтерфейс, push-сповіщення); пропозиції додаткових матеріалів, завдань або курсів.	Після завершення модуля/курсу, або при виявленні прогалин у знаннях	До 1 хвилини	Студент

(Джерело – робота автора)

Далі наводиться перелік та опис структурних одиниць вихідних повідомлень, які мають самостійне змістовне значення. Ці показники

формуються на основі даних з PostgreSQL та обробки логікою Django і Python-моделей машинного навчання.

1. Результати перевірки завдання (PER_RES):

- Бал_за_завдання: Ціле число або число з одним знаком після коми (наприклад, 0-100 або 0.0-10.0). Обчислюється з точністю до 0.1, якщо це бал від моделі МН, або до 1, якщо це цілісний бал. Надійність обчислення забезпечується валідацією вхідних даних та внутрішньою логікою Python-моделі МН.
- Коментар_МН: Текстове значення, що містить аналіз або рекомендації від моделі машинного навчання. Генерується на основі алгоритмів МН; його точність залежить від якості навчених моделей.
- Коментар_викладача: Текстове значення, введене викладачем.

2. Детальний звіт про прогрес студента (ST_ZV):

- Загальний_бал_по_курсу: исло з двома знаками після коми (відсоткове значення). Обчислюється як середньозважений бал за всіма завданнями курсу.
- Середній_час_виконання_завдання: Час у хвилинах або годинах, обчислюється з точністю до однієї хвилини.
- Кількість_спроб_виконання: Ціле число.

3. Звіт про успішність групи/курсу (ZV_CRS, ZV_GRP):

- Середній_бал_групи_по_курсу: Число з двома знаками після коми (відсоткове значення).
- Відсоток_студентів_із_проблемними_темами: Число з двома знаками після коми. Визначається на основі аналізу МН-моделями прогалин у знаннях студентів.
- Кількість активних студентів курсу: Ціле число.

4. Системні метрики та статистика МН (SYS_ZV, METR):

- Завантаження_процесора_сервера: Відсоткове значення, ціле число.
- Точність_МН_моделі: Число з двома знаками після коми (коефіцієнт точності, наприклад, 0.95). Обчислюється в Python-моделях МН на

основі тестових даних.

- Час_обробки_МН_запиту: Час у мілісекундах, ціле число.

5. Персоналізовані рекомендації (PERS_REC):

- Рекомендований_матеріал: Текстове посилання на навчальний матеріал або завдання. Генерується на основі аналізу МН-моделями профілю знань студента.
- Причина_рекомендації: Текстове пояснення, чому був рекомендований той чи інший матеріал.

У додатку А наведено ескіз результатів перевірки завдання на ШІ-контент.

Вхідна інформація для нашої навчальної платформи – це всі дані, які надходять до системи і необхідні для її функціонування, обробки, зберігання та прийняття рішень. Вона є джерелом для формування оперативних даних у базі даних PostgreSQL, а також для навчання та роботи моделей машинного навчання на Python. Ці дані надходять з різних джерел і мають різну періодичність.

У таблиці 2.2. наведено перелік і опис вхідних повідомлень.

Таблиця 2.2 – Вхідні повідомлення

№з/п	Назва вхідного повідомлення	Ідентифікатор	Форма подання	Термін і частота надходження	Джерело
1	Дані реєстрації/авторизації користувача	AUTH	Веб-форма (React.js), JSON-об'єкт (для API Django)	При реєстрації, кожному вході в систему	Студент, Викладач, Адміністратори
2	Навчальні матеріали	MATERIAL	Текстові файли (PDF, DOCX), відеофайли (MP4), зображення (JPG, PNG), Markdown-текст	За потребою, при створенні/оновленні курсу	Викладач
3	Завдання	ZAVD	Веб-форма (React.js), JSON-об'єкт	При створенні/редагуванні завдання	Викладач
4	Відповіді студента на завдання	VIDP	Текстові поля (вільна відповідь), вибір варіантів, завантажені файли	Після виконання завдання	Студент
5	Дані для навчання/оновлення МН-моделей	PARAMS	CSV/JSON файли, структуровані набори даних	За потребою, при донавчанні моделей	Команда розробки, Системний адміністратор
6	Запити на генерацію звітів	ZV_ZAP	Веб-форма (React.js), API-запит	За потребою	Користувачі

(Джерело – робота автора)

Перелік та опис структурних одиниць вхідних повідомлень.

Далі наводиться перелік та опис структурних одиниць вхідних повідомлень, які мають самостійне змістовне значення. Ці дані є критично важливими для функціонування платформи, її бази даних (PostgreSQL) та роботи моделей машинного навчання на Python.

1. Дані реєстрації/авторизацій користувача (AUTH):
- Email: Має відповідати стандартному формату email.
- Пароль: Мінімальна довжина, наявність спецсимволів (валідація на фронтенді React.js та бекенді Django).
- Ім’я: Текстове значення.
2. Навчальні матеріали (MATERIAL):
- Зміст_матеріалу: Текстовий формат, підтримка Markdown.
- Посилання_на_матеріал: Валідний URL.
- Тип_матеріалу: Перелік (наприклад, "Текст", "Відео", "Посилання", "Документ").
3. Завдання (ZAVD):
- Текст_завдання: Текстове значення.
- Тип_завдання: Перелік (наприклад, "Один_вибір", "Множинний_вибір", "Вільна_відповідь", "Код").
- Критерії_оцінювання: Текстове значення, JSON-структура.
4. Відповіді студента (VIDP):
- Текст_відповіді: Текстове значення, необмежена довжина.
- Завантажений_файл: Бінарні дані файлу (PDF, DOCX, PY, JS тощо).
5. Дані для навчання/оновлення МН-моделей (PARAMS):
- Дані_для_тренування: Структурований формат (CSV, JSON), відповідність вимогам Python-моделі МН.
- Оновлені_параметри: Числове, текстове, логічне значення (залежить від параметра).
6. Запит на генерацію звітів (ZV_ZAP):

- Тип_звіту: Перелік (наприклад, "Прогрес_студента", "Успішність_курсу").
- ID_об’єкта_для_звіту: Ідентифікатор студента, курсу, групи.

2.2. Алгоритм розв’язання задачі

Використовувана інформація.

Для розв'язання задачі "Підсистема автоматизованої перевірки знань студентів" використовується інформація, що надходить від користувачів, з інших підсистем, а також дані, які зберігаються в базі даних PostgreSQL. Ця інформація є основою для роботи Django-бекенду, логіки JavaScript на фронтенді та, що найважливіше, для навчання та виконання моделей машинного навчання на Python. Вона дозволяє ідентифікувати користувачів, надавати навчальний контент, фіксувати відповіді студентів, а також забезпечувати функціонування алгоритмів перевірки.

Наведені масиви сформовані з вхідних повідомлень та масивів, що генеруються іншими компонентами системи, для реалізації даного алгоритму.

Таблиця 2.3 – Перелік масивів використовуваної інформації

Масив	Ідентифікатор	Максимальна кількість записів
Користувачі	USERS_DATA	1 мільйон
Курси	COURSES_DATA	10 тисяч
Модулі курсів	MODULES_DATA	50 тисяч
Навчальні матеріали	MATERIALS_DATA	100 тисяч
Завдання	ASSIGNMENTS_DATA	200 тисяч
Відповіді студентів	SUBMISSIONS_DATA	5 мільйонів
Моделі МН	ML_MODELS_DATA	100
Налаштування моделей МН	ML_CONFIG_DATA	100
Дані для навчання МН-моделей	ML_TRAIN_DATA	1 мільйон
Записи на курс	ENROLLMENTS_DATA	10 мільйонів

(Джерело – робота автора)

Результати розв'язання задачі "Підсистема автоматизованої перевірки знань студентів" призначені для надання об'єктивної та своєчасної оцінки знань студентів, інформування викладачів про прогрес їхніх груп та окремих учнів, а також для забезпечення адміністраторів даними про ефективність та функціонування системи. Ці результати використовуються для зворотного зв'язку, прийняття освітніх та управлінських рішень.

Наведені масиви сформовані для видачі вихідних повідомлень (відеограм, звітів) та масивів, що зберігаються в базі даних PostgreSQL для подальшого аналізу та розв'язання цієї або інших задач.

Таблиця 2.4. Перелік масивів результатної інформації

Масив	Ідентифікатор	Максимальна кількість записів
Результат перевірки МН	ML_RESULTS_DATA	5 мільйонів
Звіти про прогрес Студента	STUD_PROG_REPORTS	1 мільйон
Звіти про успішність Курсу/Групи	COURSE_PERF_REPORTS	100 тисяч
Системні метрики та Статистика МН	SYS_METRICS_DATA	10 тисяч
Персоналізовані рекомендації	PERSON_REC_DATA	5 мільйонів

(Джерело – робота автора)

Математичний опис підсистеми автоматизованої перевірки знань, з урахуванням використання машинного навчання, стосується, перш за все, моделей та формул, що використовуються для оцінювання відповідей студентів, прогнозування їхнього прогресу, генерування персоналізованих рекомендацій, а також виявлення відповідей, згенерованих штучним інтелектом. Враховуючи, що всі моделі розроблятимуться на Python, далі описано концепції, які лежать в основі цих алгоритмів.

Опис процесу та об’єктів.

Основні процеси, що потребують математичного опису, включають оцінювання відповіді студента на завдання та ідентифікацію можливої генерації

ШІ. Об'єктами цих процесів є:

- Відповідь студента (S_i): Вхідні дані (текст, код, вибір варіантів) для аналізу.
- Правильна відповідь / Еталон (R_j): Набір еталонних даних, наданих викладачем.
- Модель оцінювання знань (M_{score}): Алгоритм, навчений для визначення якості знань на основі відповіді.
- Модель виявлення ШІ (M_{AI_detect}): Алгоритм, навчений для класифікації відповіді як людської або згенерованої ШІ.
- Оцінка знань ($Score_i$): Числове значення, що відображає якість відповіді S_i.
- Ймовірність генерації ШІ (P_{AI_gen}): Значення від 0 до 1, що вказує на ймовірність того, що відповідь була згенерована ШІ.

Прийнятні допущення.

Достатній обсяг та якість навчальних даних: Для ефективного функціонування обох типів моделей МН (оцінювання та виявлення ШІ) передбачається наявність великих, різноманітних та якісно розмічених даних. Це включає приклади як людських, так і ШІ-згенерованих відповідей для тренування M_{AI_detect}.

Варіативність ШІ-моделей: Припускається, що M_{AI_detect} буде достатньо узагальненою, щоб виявляти відповіді від різних генеративних моделей ШІ, а не лише тих, на яких вона була навчена.

Стабільність та адаптивність моделей МН: Обидві моделі (оцінювання та виявлення ШІ) повинні зберігати свою ефективність протягом певного періоду та мати можливість регулярного донавчання (перенавчання), оскільки мовні моделі ШІ постійно розвиваються.

Визначеність критеріїв: Для оцінювання знань існують чіткі або навчені критерії. Для виявлення ШІ, критеріями є статистичні та лінгвістичні ознаки, що відрізняють людський текст від згенерованого.

Математичні формули розрахунку основних показників.

Математичний опис, включає формули, релевантні для виявлення ШІ-генерованого тексту.

1. Формула оцінки відповіді на завдання з вільною формою (семантична схожість) – M_{score}: Для завдань, що передбачають вільну текстову відповідь, оцінка знань може ґрунтуватися на семантичній схожості відповіді студента (S_i) з еталонною відповіддю (R_j) або набором ключових понять. Це реалізується за допомогою векторних представлень тексту (embedding) та метрик схожості.

$$Score_i = f_{similarity}\left(Embedding(S_i), Embedding(R_j)\right) \times Max\_Score_{Assignment} \quad (2.1)$$

Де:

- $f_{similarity}(\cdot, \cdot)$ – функція схожості, що повертає значення від 0 до 1.
- $Embedding(X)$ – векторне представлення тексту *X,* отримане за допомогою попередньо навчених моделей обробки природної мови (NLP) на Python (наприклад, Word2Vec, BERT embeddings).
- $Max\_Score_{Assignment}$ – максимальний бал, який можна отримати за дане завдання.

Ця формула дозволяє моделі МН оцінити, наскільки змістовно відповідь студента відповідає правильній відповіді або ключовим концепціям, навіть якщо формулювання не є ідентичним.

2. Формула для розрахунку ймовірності генерації відповіді ШІ (M_{AI_detect}): Для виявлення того, чи була відповідь S_i генерована штучним інтелектом, використовується модель класифікації (наприклад, бінарний класифікатор). Модель аналізує лінгвістичні, стилістичні та статистичні ознаки тексту.

$$P_{AI\_gen}(S_i) = \sigma(LinearLayer(Features_{AI}(S_i))) \quad (2.2)$$

Де:

- $P_{AI\_gen}(S_i)$ – ймовірність того, що відповідь S_i була згенерована ШІ

(значення від 0 до 1).

- $Features_{AI}(S_i)$ – набір ознак, вилучених з відповіді S_i, які є індикаторами генерації ШІ (наприклад, перплексія, burstiness, наявність типових фраз, синтаксичні патерни, використання специфічного словника). Це можуть бути як лінгвістичні особливості, так і ознаки, вилучені за допомогою більш складних нейронних мереж.
- $LinearLayer$ – лінійний шар або частина нейронної мережі, що обробляє ознаки.
- σ – сигмоїдна функція активації, яка перетворює вихідне значення на ймовірність.

Ця формула відображає роботу класифікатора, який на основі вилучених ознак оцінює, наскільки відповідь S_i схожа на текст, згенерований ШІ. Важливо зазначити, що точність цієї моделі буде залежати від різноманітності навчальних даних (як людських, так і згенерованих різними моделями ШІ) та здатності моделі адаптуватися до нових генеративних моделей.

3. Формула для розрахунку загального балу студента за курс: Загальний бал студента ($TotalScore_{Student}$) за курс може бути розрахований як середньозважений бал за всіма виконаними завданнями курсу, з можливим коригуванням на ймовірність генерації ШІ. Якщо $P_{AI\_gen}(S_i)$ перевищує певний поріг θ_{AI}, бал за завдання може бути анульований або зменшений, що потребує додаткової перевірки викладачем.

$$TotalScore_{Student} = \frac{\sum_{j=1}^{N}(Score_j \times Weight_j \times \mathbb{I}(P_{AI\_gen}(S_j) < \theta_{AI}))}{\sum_{j=1}^{N} Weight_j} \quad (2.3)$$

Де:

- $Score_j$ – бал студента за j-те завдання, отриманий від M_{score}.
- $Weight_j$ – вага j-того завдання у загальній оцінці курсу.
- N – загальна кількість оцінених завдань у курсу.
- $\mathbb{I}(\cdot)$ – індикаторна функція, яка дорівнює 1, якщо умова істинна, і 0, якщо хибна.
- θ_{AI} – порогове значення ймовірності генерації ШІ, вище якого відповідь

вважається підозрілою.

Ця формула забезпечує агреговану оцінку успішності студента, враховуючи при цьому виявлення випадків, де відповідь могла бути згенерована ШІ. Це дозволяє системі автоматично реагувати на такі ситуації.

Оцінка відповідності розроблених моделей реальному процесу в різних умовах роботи.

Розроблені математичні моделі та алгоритми на Python прагнуть максимально відповідати реальному процесу оцінювання та забезпечувати доброчесність.

Для об'єктивних завдань (множинний вибір): Модель M_{score} забезпечує високу точність. M_{AI_detect} для таких завдань менш релевантна, але може бути використана для виявлення підозрілих патернів у виборі відповідей.

Для суб'єктивних завдань (вільні відповіді): Точність M_{score} залежить від якості навчальних даних та складності моделі. Точність M_{AI_detect} є ключовою, але вона також залежить від постійного оновлення моделей та даних для навчання, оскільки генеративні моделі ШІ швидко еволюціонують. При виявленні підозри на генерацію ШІ (коли P_{AI_gen} перевищує θ_{AI}), система буде виділяти таку відповідь для обов'язкової ручної перевірки викладачем. Це забезпечує гнучкість та достовірність, не покладаючись виключно на автоматизоване рішення.

При прогнозуванні прогресу та генерації рекомендацій: Моделі можуть надавати достовірні прогнози та рекомендації, що допомагає викладачам та студентам вчасно реагувати на потенційні проблеми у навчанні.

Загалом, математичні моделі є основою для автоматизації та виявлення ШІ-генерованих відповідей, але їх ефективність підтримується та контролюється через моніторинг (Django-бекенд) та можливість втручання людини (викладача через React.js-інтерфейс) для забезпечення високої якості освітнього процесу та академічної доброчесності.

Алгоритм розв’язання задачі на ЕОМ.

В цьому підрозділі описано логіку роботи підсистеми автоматизованої

перевірки знань студентів, включно з виявленням відповідей, згенерованих ШІ. Тут детально описано послідовність етапів розрахунку та взаємодію між компонентами системи: React.js (фронтенд), Django (бекенд) та Python-моделями машинного навчання (МН), що працюють з базою даних PostgreSQL.

Загальна логіка алгоритму:

1. Подача відповіді студентом: Студент заповнює форму відповіді на завдання у React.js-інтерфейсі. Дані відповіді, разом з ID_Завдання та ID_Студента, надсилаються на Django-бекенд. Django-бекенд зберігає відповідь у таблиці StudentSubmissions в PostgreSQL зі статусом "В_очікуванні_перевірки".
2. Запуск процесу автоматизованої перевірки: Django-бекенд, отримавши нову відповідь або за розкладом, ініціює процес перевірки. Визначається Тип_завдання з таблиці Assignments та асоційована ID_МН_моделі_для_перевірки (з MLModels).
3. Виявлення генерації ШІ (для вільних відповідей): Якщо Тип_завдання передбачає вільну текстову відповідь, Django-бекенд викликає Python-модель для виявлення ШІ, передаючи їй текст відповіді студента. Модель аналізує текст і повертає ймовірність генерації ШІ. Django-бекенд порівнює дану ймовірність з порогом.
4. Формування та відображення результатів: Django-бекенд агрегує дані з StudentSubmissions та MLResults. Результати перевірки надсилаються на React.js-фронтенд для відображення студенту та викладачу. Якщо відповідь була "Підозрілою_на_ШІ_генерацію", це чітко позначається в інтерфейсі, і студенту може бути запропоновано перевиконати завдання або чекати на ручну перевірку.

Схемою алгоритму (рис. 4) передбачені всі ситуації, які можуть виникнути в процесі перевірки завдання, від його подачі до отримання результату та можливого втручання викладача.

.

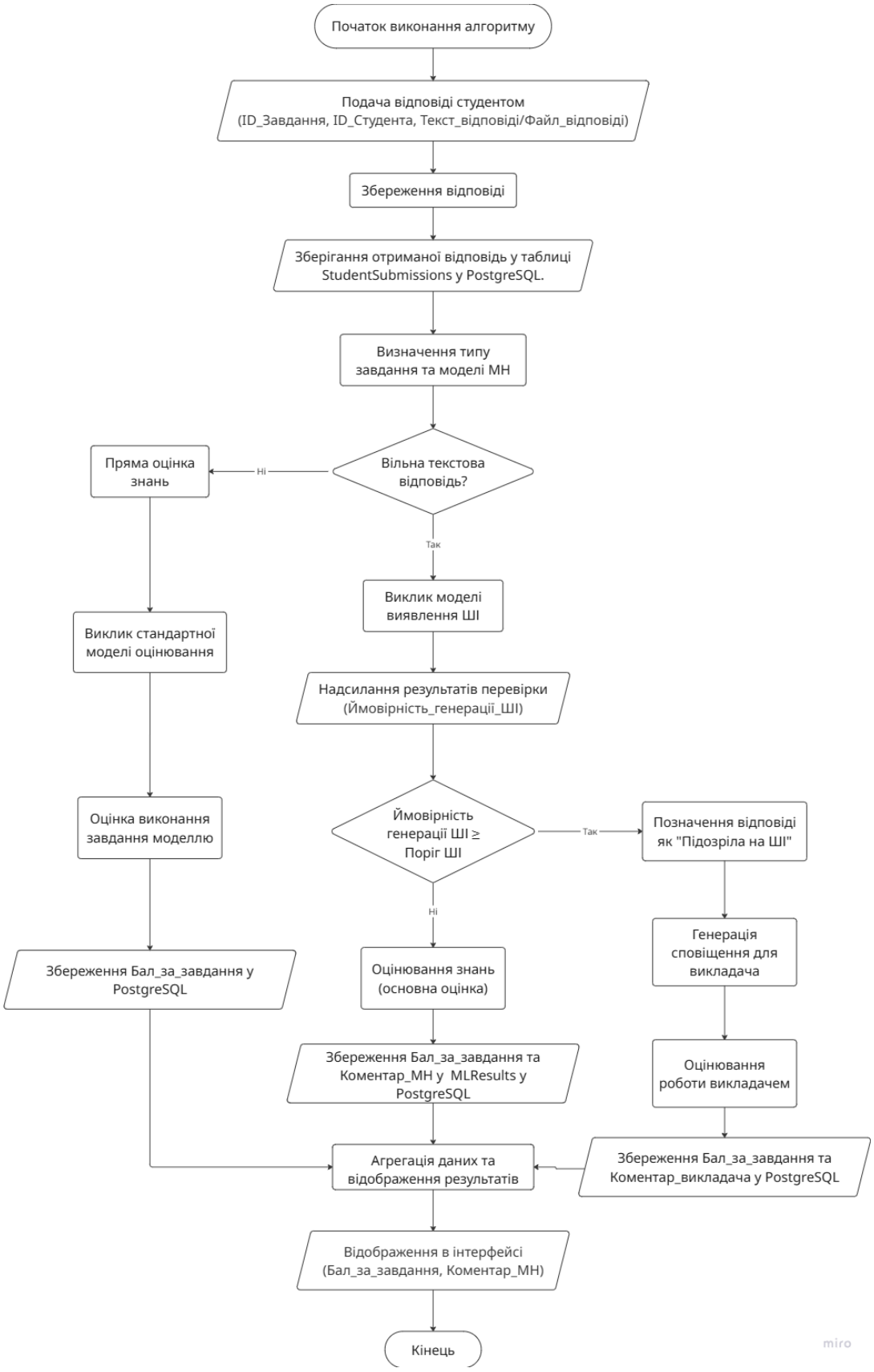


Рисунок 2.4 – Алгоритм розв’язання задачі

(Джерело – робота автора)

2.3. Моделювання структури ІС

Моделювання є універсальним засобом для вивчення складних систем за допомогою їх спрощення шляхом абстрагування. Моделе-орієнтований інжиніринг є сучасним перспективним підходом до розробки складних систем, зокрема інформаційних, і передбачає, що основними розроблюваними артефактами є моделі, з яких генеруються інші системні артефакт.

2.3.1. Моделювання поведінки системи

1) Діаграма прецендентів.

Діаграма прецедентів документує функціональні вимоги у вигляді варіантів використання (use cases) як типової взаємодії системи з акторами. Склад та обсяг системи, що подає діаграма прецедентів, відповідає функціональним вимогам до "Підсистеми автоматизованої перевірки знань студентів" та її постановці.

Серед основних акторів я виділив Студента, Викладача та Адміністратора Системи. Студент може приєднуватись до курсу, виконувати та коментувати завдання. Після авторизації він додається до списку користувачів. Викладач створює курси, додає до них завдання, до яких потім також може лишати коментарі. Після авторизації також додається до списку користувачів. Адміністратор, в даному випадку, може авторизуватись та керувати списком користувачів після цього. На рисунку 5 зображена діаграма прецендентів, яка описує дані процеси.

Примітка: під час створення діаграми я трішки помилився, та додав акторів зверху та знизу діаграми. Вже на етапі написання роботи згадав, що всі актори мають бути з боків.

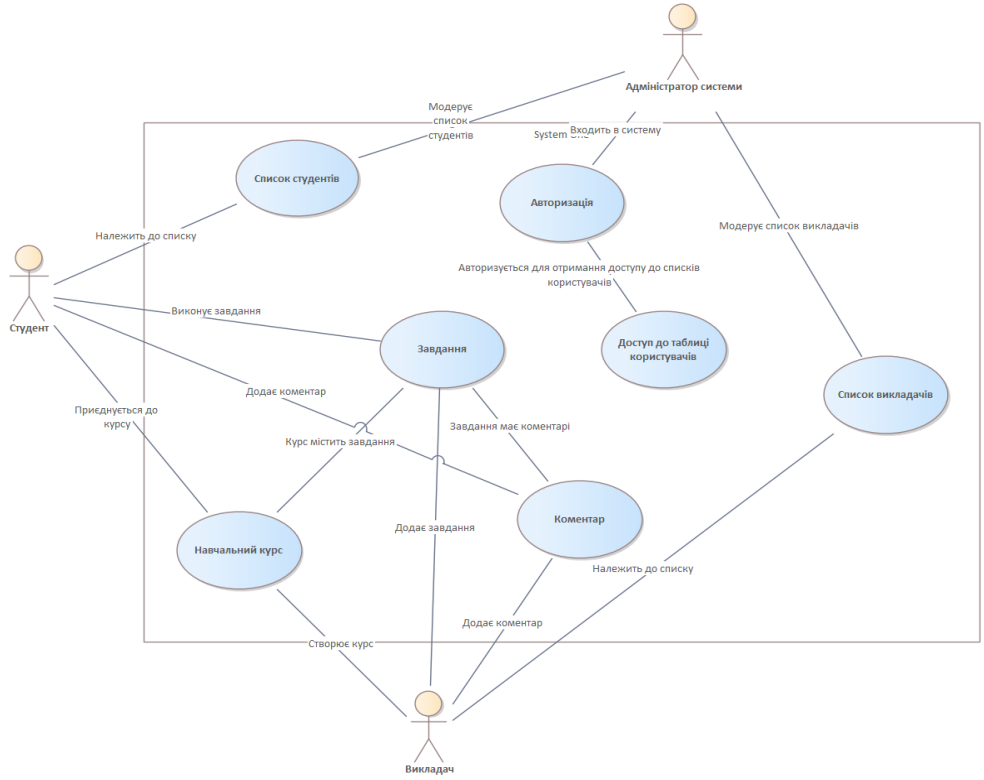


Рисунок 2.5 – Діаграма прецендентів

(Джерело – робота автора)

2) Діаграми послідовності.

Діаграми послідовності (Sequence Diagrams) деталізують типову взаємодію між об'єктами системи та акторами в часовій послідовності. Я описав сценарії для двох прецендентів: створення курсу та вхід адміністратора в систему.Це дозволить виокремити основні об'єкти (компоненти) проектованої системи та їхні взаємодії.

Наведена на рисунку 6 діаграма відображає процес входу адміністратора в систему. Спершу дані, які він використав для входу, передаються у спеціальний сервіс автентифікації. Дані перевіряються та звіряються з базою даних. Далі назад в сервіс надсилається повідомлення про те, були дані правильними чи ні. І вже в залежності від цього сервіс вирішує, чи авторизувати адміністратора.

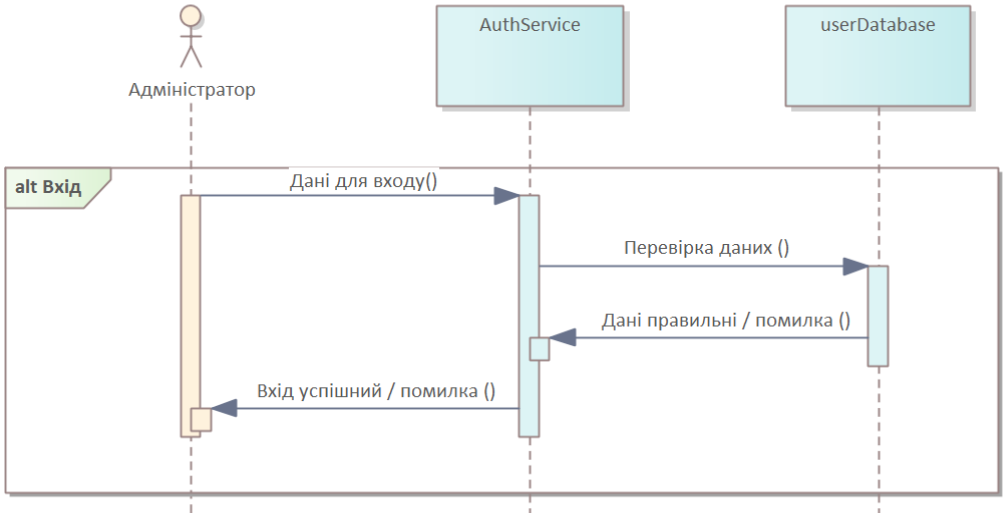


Рисунок 2.6 – Діаграма послідовності «Вхід адміністратора в систему»

(Джерело – робота автора)

На рисунку 2.7 зображено діаграму послідовності для процесу створення курсу. Спершу викладач заповнює спеціальну форму, надаючи всю потрібну інформацію. Ця інформація перевіряється на сервері й заноситься в базу даних. Користувача (викладача) потім повідомляє, було курс створено успішно, чи є помилки.

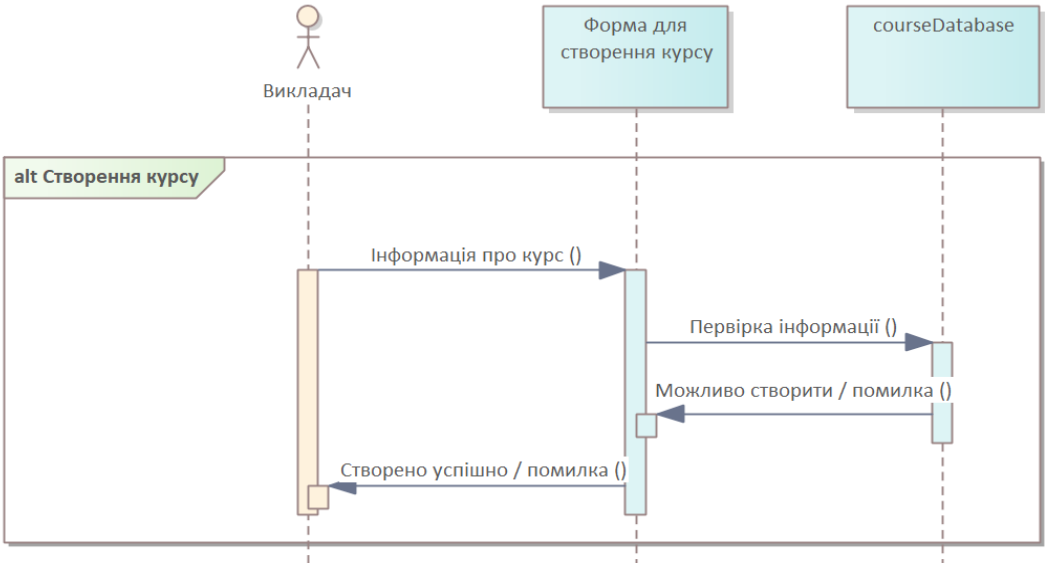


Рисунок 2.7 – Діаграма послідовності «Створення курсу»

(Джерело – робота автора)

2.3.2. Моделювання структури системи

Моделювання структури системи здійснюється за допомогою діаграм класів UML, які відображають статичну структуру класів системи та зв'язків між ними. Це дозволяє представити, як дані та логіка системи організовані в об'єктах, що є основою для розробки Django-моделей та компонентів на React.js.

Наведена діаграма класів (рис. 8) представляє структуру класів, що відповідають за процес входу адміністратора в систему. Вона містить в собі наступні класи:

- loginForm з атрибутами email (string) та password (string), а також функцією submitLogin(), яка повертає булеанове значення, яке залежить від того, чи було все заповнено правильно.
- AuthService з атрибутом logTime (string) та функцією isAdmin(), яка перевіряє, чи є користувач адміністратором й повертає відповідне булеанове значення. Зв’язок 1 до 1 з user.
- userDatabase з методами saveUser та User.findOne. Зв’язок 1 до 1 з AuthService та 1 до багатьох з user.
- User з атрибутами email (string), name (string), password (string) та role (string).

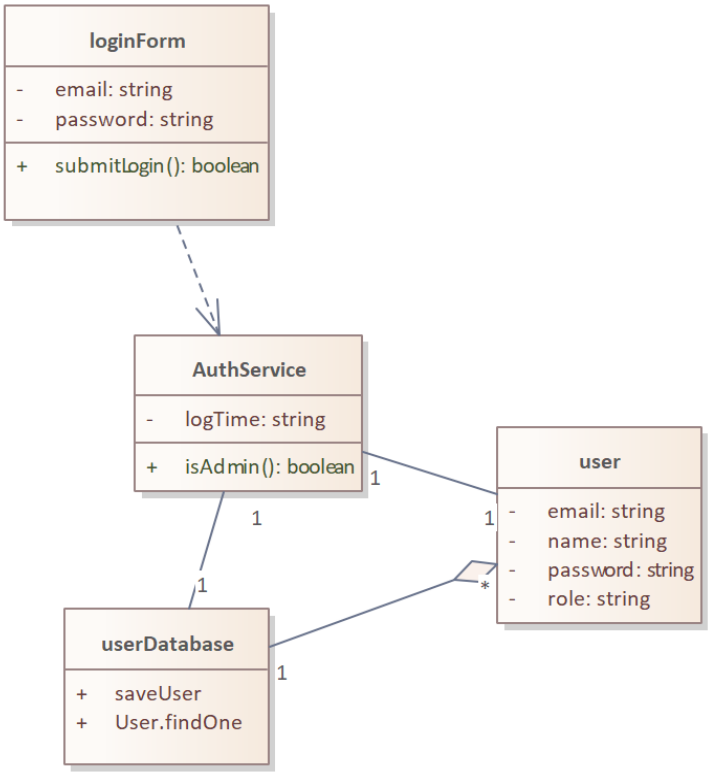


Рисунок 2.8 – Діаграма класів «Авторизація»

(Джерело – робота автора)

Діаграма на рис. 2.9 відображає структуру класів, що відповідають за процес створення нового курсу. Вона містить в собі наступні класи:

- courseForm з атрибутами blockName (string), courseName (string), duration (string), teacherName (string) та функцією submitForm(), яка повертає сутність класу Course.
- courseService з атрибутом course (Course), який відповідає сутності з форми, а також функції validateForm(), яка перевіряє, чи було правильно заповнено форму й повертає відповідне булеанове значення, а також addCourse(), яка додає курс й повертає True, якщо курс було додано успішно. Має зв’язок 1 до 1 з courseDatabase та 1 до 1 з course.
- Course з атрибутами blockName (string), courseName (string), duration (string) та teacherName (string), що відповідають даним з форми. Має зв’язок багато до 1 з courseDatabase.
- courseDatabase з функціями saveCourse та Course.findOne.

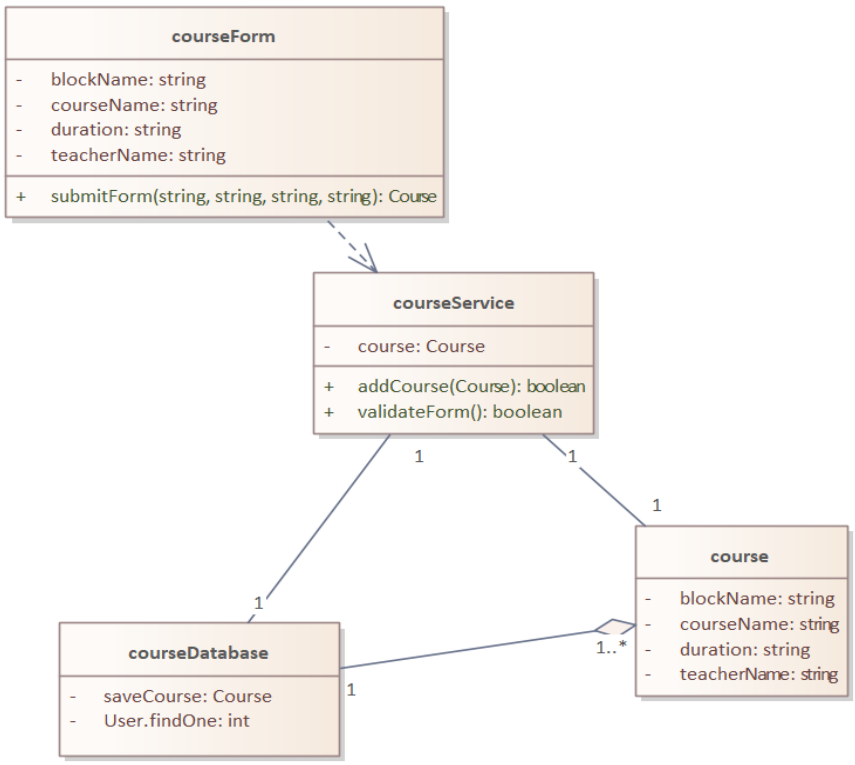


Рисунок 2.9 – Діаграма класів «Створення курсу»

(Джерело – робота автора)

На додаток до діаграм класів UML, які зосереджуються на об'єктно-орієнтованих аспектах системи, для складніших систем з різними типами компонентів (апаратне, програмне забезпечення) часто використовуються структурні діаграми SysML. Ці діаграми пропонують більш системний погляд на архітектуру, дозволяючи моделювати систему як сукупність взаємопов'язаних блоків.

Діаграма визначення блоків (BDD) є фундаментальним інструментом у SysML для подання внутрішньої структури системи у вигляді ієрархії блоків різних типів. Блоки в SysML є узагальненим поняттям, яке може представляти як апаратні компоненти (наприклад, серверний кластер для обробки МН), так і програмні компоненти (наприклад, модуль Django-бекенду, Python-модель ШІ-детекції), а також дані чи енергію. На рисунку 10 зображено таку діаграму.

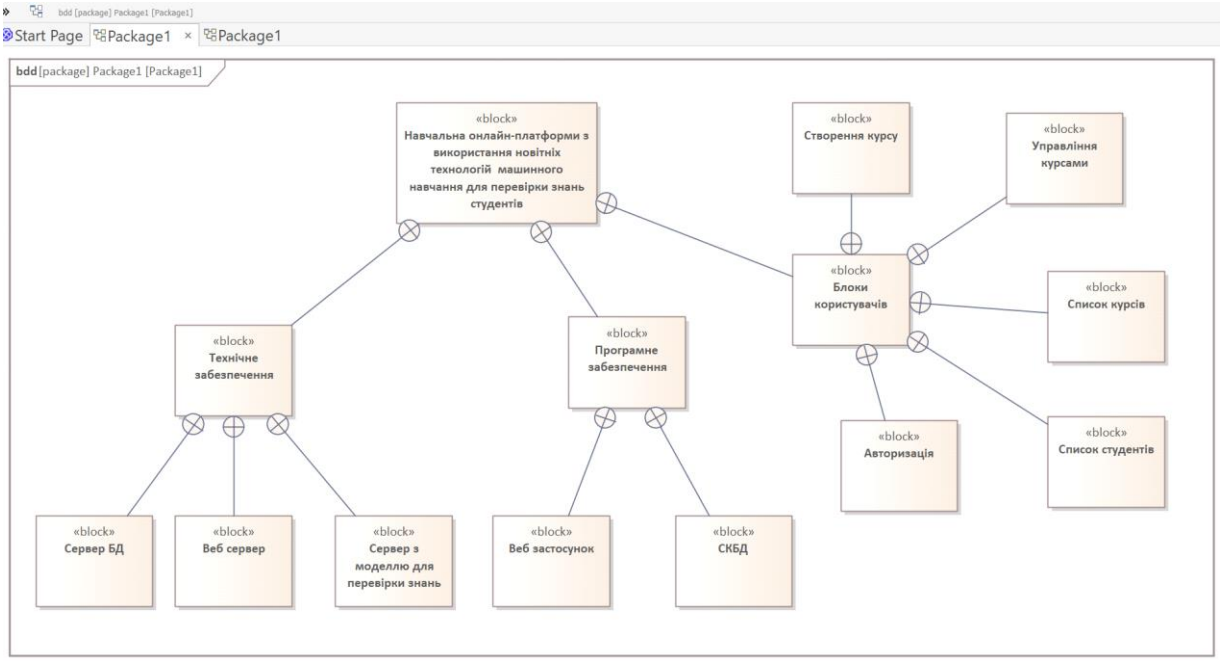


Рисунок 2.10 – Діаграма визначення блоків

(Джерело – робота автора)

BDD допомагає ієрархічно розкласти систему на її складові частини, даючи чітке уявлення про те, з чого складається система і як ці складові організовані, без заглиблення у деталі їх внутрішньої взаємодії.

Діаграма внутрішніх блоків (IBD), на відміну від BDD, яка визначає "що є", фокусується на "як це працює всередині". Вона використовується для подання внутрішньої будови конкретного блоку, демонструючи його як сукупність частин (властивостей блоку), їх з'єднання між собою безпосередньо чи через порти, а також потоків даних, матеріалів чи енергії, що існують між частинами і портами. На рисунку 11 зображено таку діаграму.

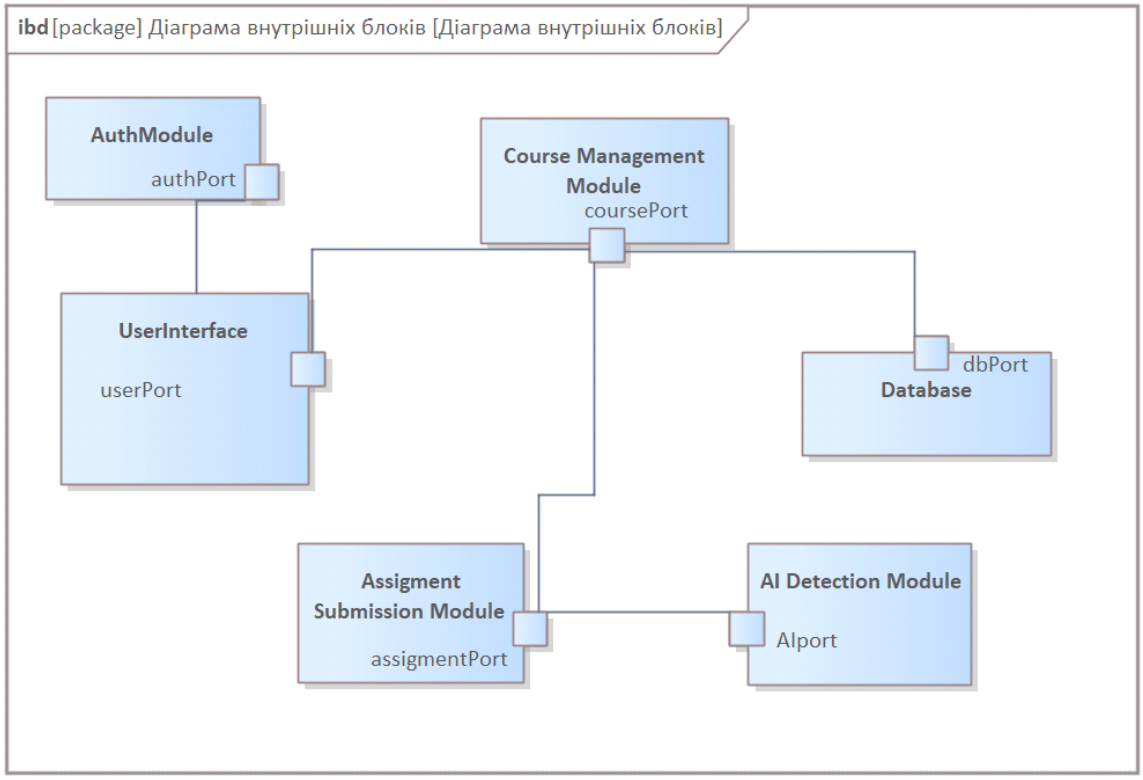


Рисунок 2.11 – Діаграма внутрішніх блоків

(Джерело – робота автора)

IBD є надзвичайно корисною для візуалізації архітектури програмного забезпечення, показуючи, як різні програмні модулі (частини) обмінюються даними та взаємодіють для виконання функціоналу системи. Це допомагає краще зрозуміти логіку та залежності всередині складних компонентів системи.

2.3.3. Розподіл вимог за компонентами

У розробці системи управління навчальним процесом трасування вимог є критично важливим. Цей процес створює прозорий зв'язок між початковими вимогами та реалізованими функціями, дозволяючи контролювати їхнє виконання та запобігаючи втраті будь-яких ключових вимог на етапах проєктування, розробки чи тестування.

Завдяки трасуванню вимог у рамках цього проєкту ми переконалися, що кожна функціональна та нефункціональна вимога до платформи онлайн-курсів

була належним чином інтегрована в її архітектуру, інтерфейс користувача та основні модулі. Такий підхід забезпечив повну відповідність готового продукту очікуванням викладачів і студентів, тим самим значно покращивши загальну якість та ефективність навчального середовища.

Представлена діаграма (рис. 12) деталізує процес трасування вимог для модуля авторизації в системі онлайн-курсів. Вона наочно відображає, як функціональні вимоги пов'язані з реалізованими компонентами та відповідними тест-кейсами. Діаграма чітко ілюструє, що всі аспекти автентифікації — від реєстрації та входу до валідації облікових даних та обробки помилок — були ретельно перевірені та підтверджені за допомогою тестування.

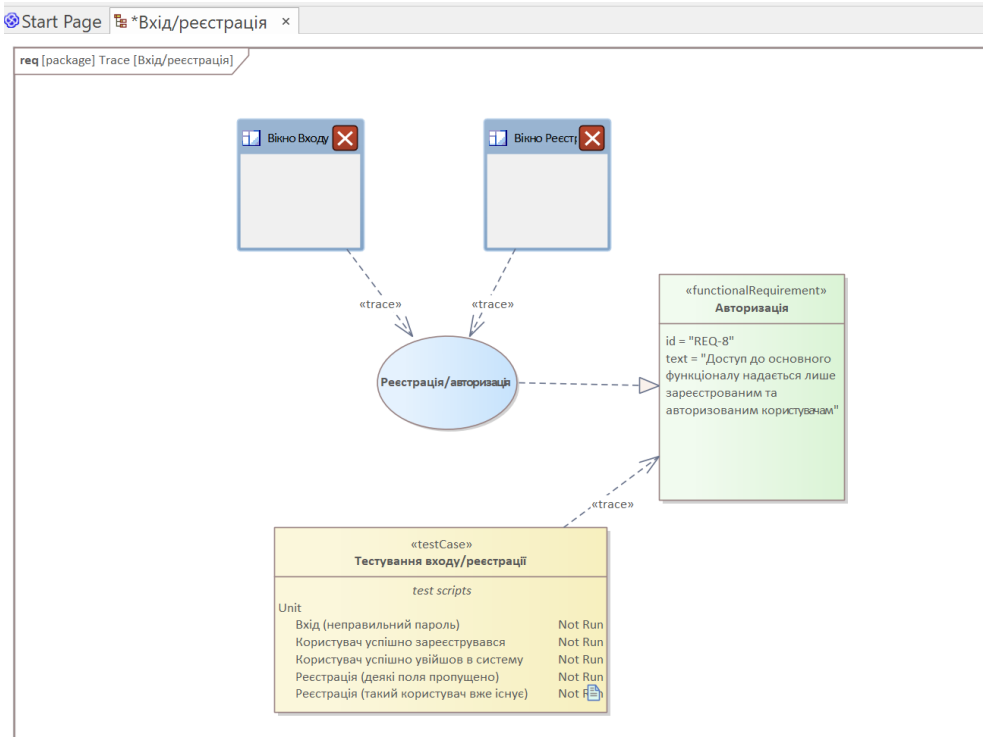


Рисунок 2.12 – Діаграма трасування Вхід/реєстрація

(Джерело – робота автора)

Матриця (рис. 13) відображає, як вихідні вимоги до системи авторизації знайшли своє повне втілення в реалізованих функціях. Вона чітко показує, що кожен запланований сценарій використання коректно працює в кінцевому продукті.

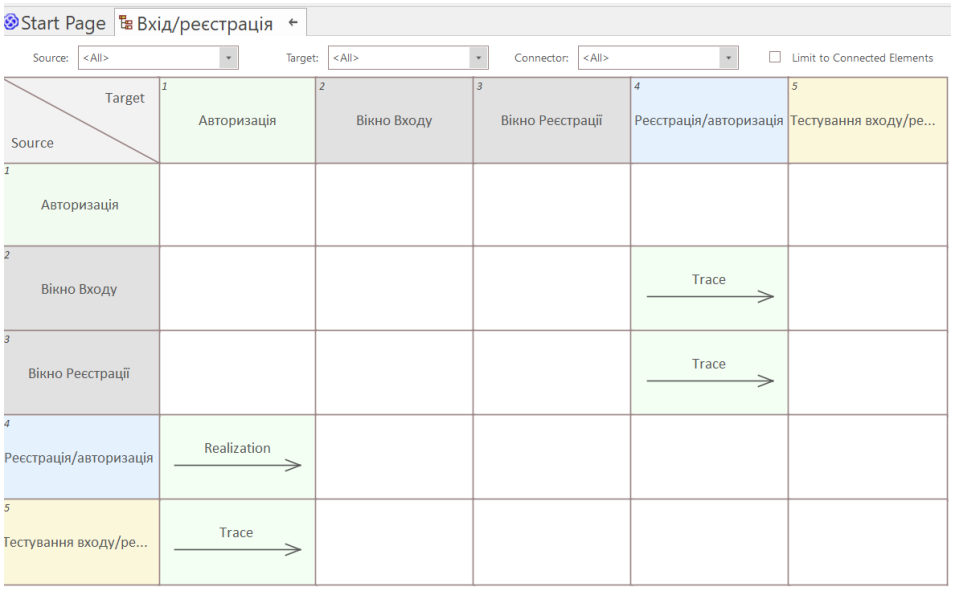


Рисунок 2.13 – Матриця взаємозв’язків Вхід/реєстрація

(Джерело – робота автора)

Ця схема трасування (рис 2.14) ілюструє процес проходження курсу та коментарі, показуючи взаємозв’язок між вимогами, сценаріями використання та тестовими кейсами.

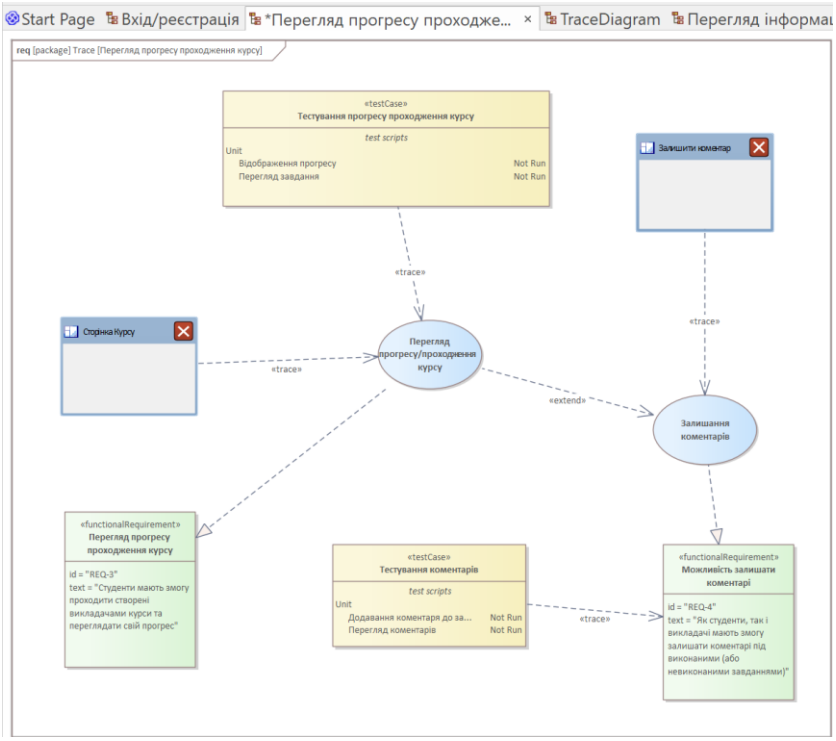


Рисунок 2.14 – Діаграма трасування Прогрес проходження курсу/коментарі

(Джерело – робота автора)

Ця матриця (рис.15) відображає взаємозв'язок між ключовими компонентами системи проходженні курсу та коментарями.

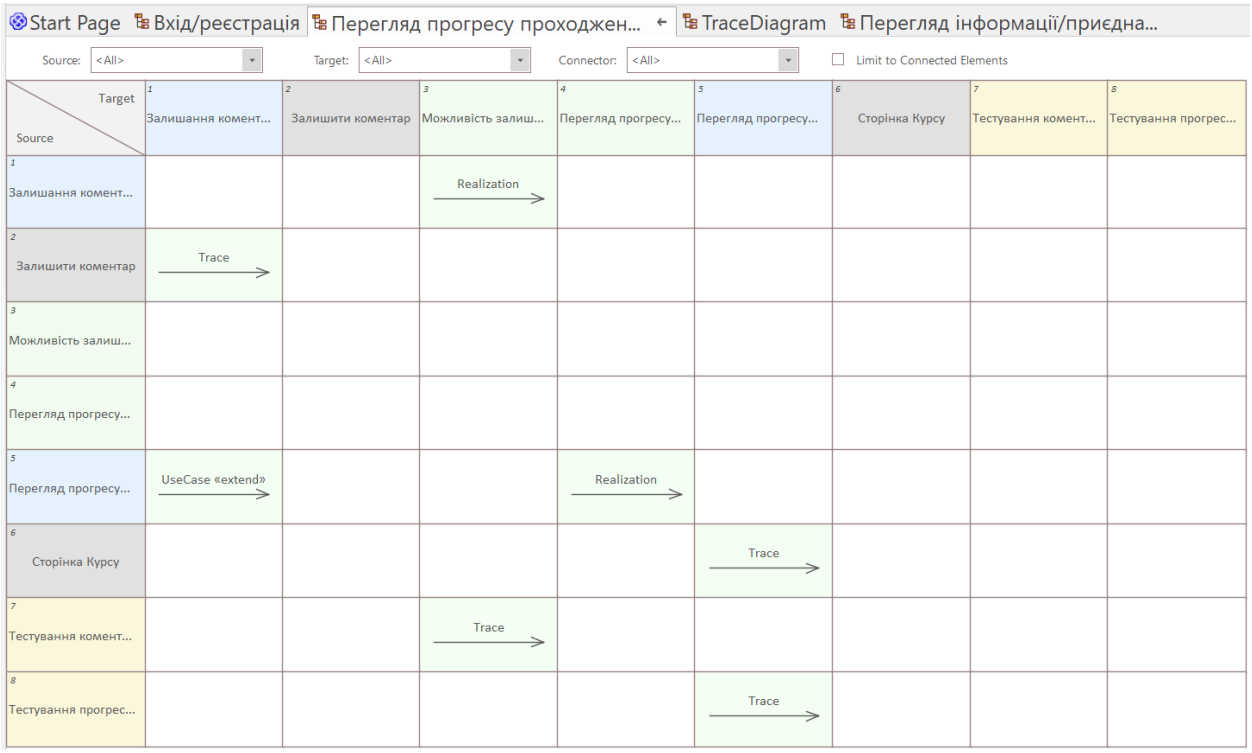


Рисунок 2.15 – Матриця взаємозв’язків Прогрес проходження курсу/коментарі

(Джерело – робота автора)

Тест-кейси та діаграми трасування, розроблені в рамках проєкту, формують надійну систему контролю якості для онлайн-платформи. Вони охоплюють весь спектр взаємодій користувачів — від базової авторизації до складних процесів створення курсів, управління завданнями та моніторингу прогресу. Ключова перевага полягає в можливості точного відстеження того, як кожна вимога відображена в реалізації, що значно полегшує процес прийняття рішень. Цей підхід забезпечує прозорість та керованість на всіх етапах життєвого циклу ПЗ, гарантуючи відповідність очікуванням користувачів та функціональним вимогам

РОЗДІЛ 3
ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ ПІДСИСТЕМ

3.1. Інформаційне забезпечення

Інформаційне забезпечення (ІЗ) підсистеми автоматизованої перевірки знань є комплексною сукупністю елементів, що утворюють єдину систему класифікації та кодування інформації, уніфіковані системи документації, а також масиви даних. Усе це разом забезпечує ефективне формування, зберігання, обробку та видачу всієї інформації, необхідної для функціонування системи. Його ключове призначення полягає у створенні надійної та продуктивної інформаційної інфраструктури, здатної підтримувати всі оголошені функціональні можливості підсистеми.

3.1.1. Загальна характеристика інформаційного забезпечення

Структура ІЗ логічно поділяється на дві основні складові. Позамашинна інформаційна база охоплює такі елементи, як первинні документи (наприклад, завдання, що створюються викладачами), системи класифікаторів та кодів (що включають типи користувачів, формати завдань), а також регламентуючу документацію, яка описує інформаційні процеси та стандарти. Внутрішньомашинна інформаційна база, у свою чергу, представлена головним чином базою даних PostgreSQL, яка слугує центральним сховищем для всіх оперативних даних — від профілів користувачів та навчальних курсів до деталей завдань, відповідей студентів та результатів перевірок, включно з даними від моделей машинного навчання. Крім того, до внутрішньомашинної бази належать і масиви даних, що спеціально підготовлені для навчання та ефективного функціонування Python-моделей машинного навчання.

Принципи, що лежать в основі організації інформаційного забезпечення, є фундаментальними для його ефективності. Принцип єдності інформаційного простору гарантується стандартизацією форматів даних, а також використанням єдиних класифікаторів та унікальних ідентифікаторів у межах усієї платформи. Достовірність та повнота інформації досягаються завдяки багатоетапній валідації вхідних даних, яка відбувається як на рівні React.js-фронтенду, так і на Django-бекенді, доповнюючись застосуванням ретельних методів контролю. Актуальність даних підтримується через їх оперативне оновлення та швидку обробку інформаційних потоків. Гнучкість та розширюваність ІЗ забезпечені архітектурою, що дозволяє легко інтегрувати нові сутності та атрибути, а також адаптуватися до зміни взаємозв'язків без необхідності значного перепроектування системи. Нарешті, безпека гарантується суворим розмежуванням прав доступу, застосуванням шифрування для чутливих даних та регулярним резервним копіюванням бази даних.

Основним носієм даних для постійного зберігання в системі є дискова пам'ять, зокрема SSD-накопичувачі серверів, на яких розміщена база даних. Всі оперативні дані, що активно використовуються під час обробки, тимчасово зберігаються в оперативній пам'яті (RAM) серверів, що виконують Django-бекенд та Python-процеси моделей машинного навчання.

Для управління та зберігання всієї інформації обрано реляційну систему керування базами даних (СКБД) PostgreSQL. Цей вибір обґрунтований кількома ключовими характеристиками: надійність та відмовостійкість PostgreSQL є запорукою цілісності даних завдяки її стабільній роботі та повній підтримці транзакцій. Будучи СКБД з відкритим вихідним кодом, вона пропонує гнучкість, активну спільноту підтримки та відсутність ліцензійних витрат. Розширюваність системи забезпечується її здатністю підтримувати різноманітні типи даних, включаючи JSONB для гнучкого зберігання, а також широкий спектр розширень. PostgreSQL також демонструє високу продуктивність при виконанні складних SQL-запитів, що критично для генерації аналітичних звітів. Її відмінна сумісність з Django ORM суттєво спрощує процес розробки та ефективне

управління базою даних на рівні Python-коду. Нарешті, масштабованість PostgreSQL дозволяє їй адаптуватися до зростаючих обсягів даних та навантажень, як шляхом вертикального, так і горизонтального масштабування.

Система реалізує кілька ключових методів контролю інформації. Контроль достовірності виконується на кількох рівнях: валідація даних під час введення (на клієнтському React.js та серверному Django рівнях), забезпечення цілісності посилань за допомогою зовнішніх ключів у PostgreSQL, а також перехресний контроль відповідності даних з різних джерел. Контроль повноти забезпечується обов'язковістю заповнення ключових полів у формах введення даних та перевіркою наявності всіх необхідних даних перед їх обробкою. Контроль надійності досягається за рахунок використання транзакцій для всіх операцій зміни даних, детального журналювання для можливості відновлення та регулярного резервного копіювання бази даних. Контроль своєчасності включає моніторинг часу обробки завдань та генерації звітів, а також систему сповіщень про можливі затримки.

Вимоги до надійності інформації передбачають безперебійний доступ до даних та їх цілісність, при цьому середній час відновлення після збою не повинен перевищувати 30 хвилин. Вимоги до достовірності інформації диктують, що дані в системі мають точно відображати реальний стан справ, а допустимий рівень помилок при введенні та обробці не повинен перевищувати 0.1%. Особлива увага приділяється високій достовірності результатів МН-перевірок та виявлення ШІ-генерації, що постійно підтверджується метриками точності самих моделей.

Загальна схема інформаційного забезпечення підсистеми (рис. 3.1) є наступною. Інформація надходить від джерел інформації (студенти, викладачі, адміністратори) та вводиться до системи через React.js-інтерфейс. Потім дані проходять обробку на Django-бекенді, де відбувається їх валідація, застосування бізнес-логіки та інтеграція з модулями машинного навчання. Всі структуровані дані зберігаються у PostgreSQL. Python-моделі машинного навчання виконують ключові операції з обробки та аналізу даних, включаючи оцінку знань та виявлення ШІ-генерованого контенту. Результати цієї обробки, а також інші

дані, видаються користувачам через React.js-інтерфейс у вигляді відеограм та звітів, а також у формі сповіщень (електронна пошта, push-повідомлення).

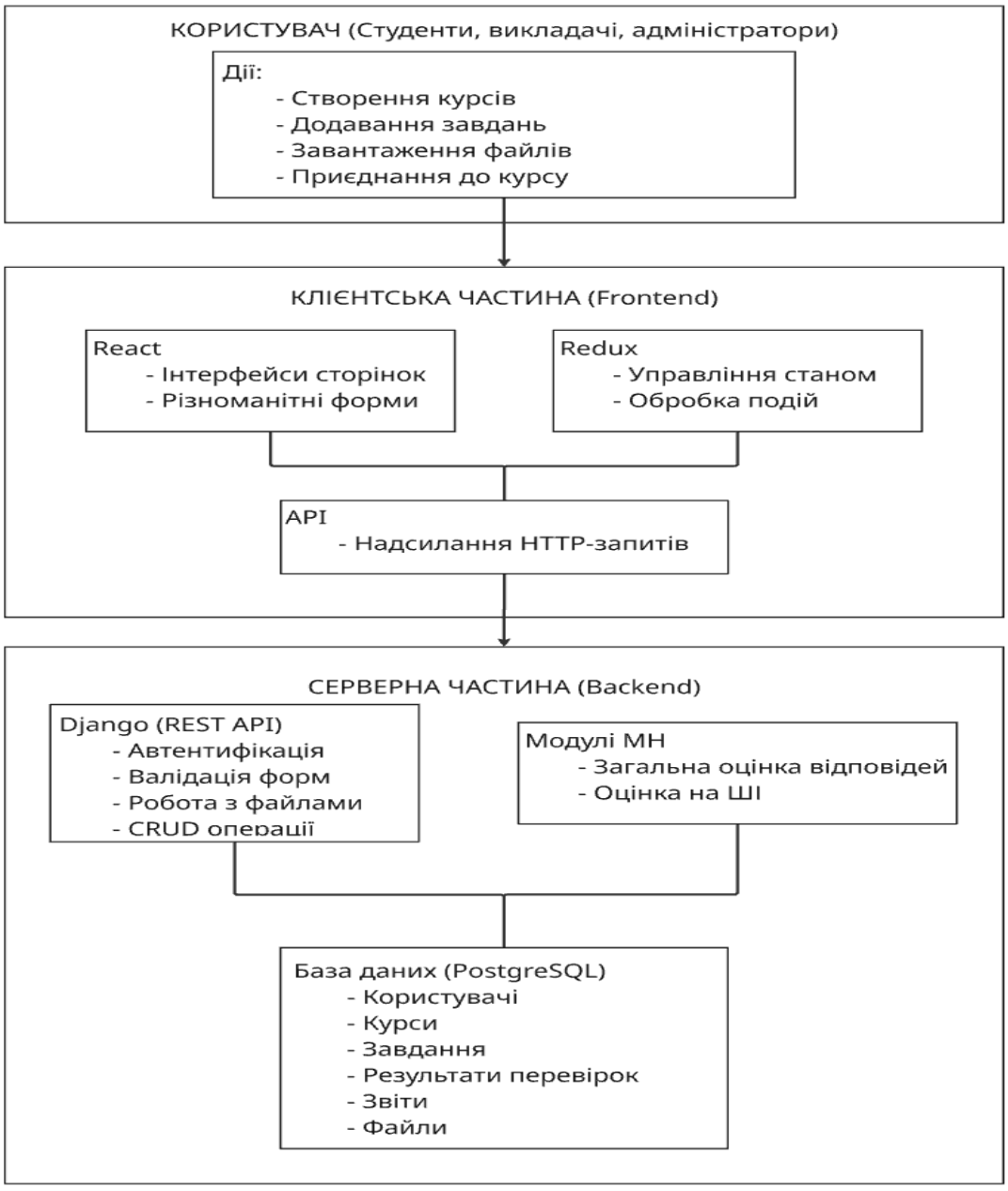


Рисунок 3.1 – Загальна схема інформаційного забезпечення

(Джерело – робота автора)

Конкретні елементи, які будуть використані для функціонування інформаційної системи, включають: PostgreSQL як СКБД; Python (з фреймворком Django та бібліотеками для МН) та JavaScript (з React.js) як основні мови програмування; Django REST Framework, React, Redux, Pandas, NumPy, Scikit-learn, TensorFlow/PyTorch, NLTK/SpaCy як ключові фреймворки та

бібліотеки. Також будуть використані Git та Docker як інструменти розробки та розгортання.

3.1.2. Організація збору і передавання первинної інформації

Організація збору та передачі первинної інформації є критично важливою для забезпечення життєдіяльності підсистеми автоматизованої перевірки знань. Цей процес охоплює всі вхідні дані, що надходять до системи від різних джерел інформації та з різною періодичністю, формуючи основу для її функціонування.

Основним джерелом первинної інформації є користувачі системи: студенти, викладачі та адміністратори. Студенти виступають джерелом своїх відповідей на завдання, які вони формують безпосередньо в React.js-інтерфейсі. Ці дані, що включають текстові відповіді, обрані варіанти або завантажені файли, передаються на Django-бекенд методом HTTP POST відразу після завершення роботи над завданням. Носієм інформації тут виступає веб-форма та дані HTTP-запиту.

Викладачі, у свою чергу, є джерелом навчального контенту, параметрів завдань, критеріїв оцінювання та ручних корекцій балів. Вони взаємодіють із системою через React.js-інтерфейс, використовуючи веб-форми для введення текстових описів, завантаження файлів матеріалів (PDF, DOCX) та налаштування параметрів тестів. Ця інформація передається на Django-бекенд також за допомогою HTTP POST/PUT запитів, як правило, за потребою — при створенні нового курсу, завдання або при необхідності коригування існуючих даних. Носієм є веб-інтерфейс та файлові завантаження.

Адміністратори системи, а також команда розробки, є джерелом даних для навчання та конфігурації моделей машинного навчання на Python. Ці дані можуть надходити у вигляді CSV/JSON файлів, завантажених через спеціальні інтерфейси адміністративної панелі Django, або навіть безпосередньо з бази даних PostgreSQL, де зберігаються вже існуючі відповіді студентів, що

використовуються для донавчання моделей. Періодичність надходження такої інформації нерегулярна – за потребою, коли виникає необхідність оновити моделі або змінити їх параметри.

Загальні вимоги до організації збору та передачі інформації включають своєчасність подання, повноту введених даних, достовірність інформації та її відповідність визначеним форматам. Кожен вхідний потік даних контролюється на відповідних етапах: на клієнтській стороні (React.js) для базової валідації введення, на серверній стороні (Django) для більш глибокої перевірки бізнес-правил та цілісності, а також на рівні PostgreSQL для забезпечення цілісності даних через схему бази даних. Це гарантує, що лише коректна та валідна інформація потрапляє до пам'яті ЕОМ та постійного сховища.

3.1.3. Побудова системи класифікації та кодування

Для ефективного функціонування підсистеми автоматизованої перевірки знань критично важливою є чітка та послідовна система класифікації та кодування об'єктів. Вона забезпечує однозначну ідентифікацію, легкий пошук та уніфіковане зберігання даних, що є основою для роботи як Django-бекенду, так і Python-моделей машинного навчання.

В роботі було використано різні класифікатори для ключових об'єктів системи. Наприклад, для користувачів застосовується простий, але ефективний класифікатор ролей, що визначає Студент, Викладач та Адміністратор. Для кожної ролі присвоюється унікальний числовий код або текстовий ідентифікатор, який фіксується у базі даних PostgreSQL. Метод кодування тут – це пряме кодування або кодування за порядком, де кожній ролі відповідає певний дискретний код, що спрощує авторизацію та розмежування доступу.

Щодо типів завдань, існує класифікатор, який включає `Тест_з_вибором_відповіді`, `Вільна_відповідь` та `Завдання_на_написання_коду`. Кожен тип отримує унікальний

ідентифікатор. Метод кодування – також пряме кодування, що дозволяє системі швидко визначати необхідний алгоритм перевірки та асоційовану МН-модель. Довжина коду є фіксованою, зазвичай, це короткий текстовий рядок або ціле число, що відображає назву типу.

Для статусів перевірки відповідей студентів впроваджено класифікатор, який може включати такі стани, як `В_очікуванні_перевірки`, `Перевірено_МН`, `Підозріла_на_ШІ_генерацію` та `Перевірено_Викладачем`. Ці статуси кодуються текстовими ідентифікаторами або числовими значеннями для зручності зберігання в PostgreSQL та маніпуляцій на Django-бекенді. Метод кодування є кодуванням за порядком, де кожному статусу відповідає певний стан життєвого циклу відповіді.

Всі ідентифікатори об'єктів системи, такі як `ID_Студента`, `ID_Завдання`, `ID_Курсу` та `ID_Відповіді`, кодуються за допомогою глобально унікальних ідентифікаторів (UUID). Цей метод кодування забезпечує унікальність кожного запису в PostgreSQL без необхідності централізованої координації, що є ідеальним для розподілених систем та масштабування. Довжина UUID фіксована (32 шістнадцяткові цифри, розділені дефісами), що гарантує надзвичайно низьку ймовірність колізій.

Використання цих систем класифікації та кодування забезпечує високу узгодженість даних по всій системі, спрощує розробку запитів до бази даних, підвищує швидкість обробки інформації та є фундаментом для ефективної взаємодії між різними компонентами підсистеми, включно з Python-моделями машинного навчання, яким потрібні чітко визначені вхідні та вихідні категорії.

3.1.4. Проектування форм первинних документів, машинограм та відеокадрів

Проектування форм первинних документів, машинограм та відеокадрів є

невід'ємною частиною інформаційного забезпечення, оскільки саме вони є тими інтерфейсами, через які користувачі взаємодіють з інформацією, а система надає їм результати своєї роботи. Це не лише візуальне представлення даних, а й обґрунтування їхнього складу, змісту та форми, що відповідає вимогам до зручності використання та ефективності.

Основними відеокадрами (екранними формами) у даній підсистемі є інтерфейси веб-платформи, реалізовані на React.js. Ці форми відображають як вхідні дані для користувача, так і результати обробки. Кожна така форма має свій унікальний ідентифікатор (наприклад, `/student/assignment/:id`, `/teacher/submission-review/:id`) та чітко визначених користувачів – студенти, викладачі або адміністратори.

Для студентів ключовими є відеокадри для подачі відповідей на завдання. Вони містять поле для введення тексту (для вільних відповідей), можливість завантаження файлів (для коду або документів), або інтерфейс для вибору варіантів (для тестів). Після обробки, студент отримує відеокадр з результатами перевірки. Ця машинограма містить `Отриманий_бал_за_завдання`, `Коментар_МН` від автоматизованої системи, а також `Коментар_викладача`, якщо була ручна дооцінка. Важливо, що у випадку підозри на генерацію ШІ, цей відеокадр чітко відображає відповідний індикатор, пропонуючи студенту перевиконати завдання або очікувати на рішення викладача. Ці дані доступні студенту періодично, за його запитом, одразу після завершення автоматизованої або ручної перевірки.

Для викладачів розроблені відеокадри для створення та редагування завдань, що містять поля для назви, опису, типу завдання, максимального балу та терміну здачі. Також критично важливими є відеокадри для перегляду відповідей студентів та ручної дооцінки. Ці форми відображають повний текст відповіді, бал, наданий системою, ймовірність генерації ШІ, а також надають можливість для введення нового балу та розширеного коментаря. Всі ці дані доступні викладачеві за потребою, коли йому необхідно перевірити роботи або проаналізувати підозрілі випадки.

Вимоги до цих форм включають інтуїтивно зрозумілий дизайн, мінімалістичність (не перевантаженість елементами), адаптивність для різних пристроїв та чітке розмежування інформації для різних ролей користувачів. Це забезпечує ефективну комунікацію та високу зручність використання підсистеми.

3.1.5. Структура інформаційних масивів

У цьому параграфі детально описується структура кожного інформаційного масиву, що використовується в підсистемі, згідно з встановленим форматом. Це дозволяє чітко визначити склад даних, їх тип, бізнес-правила та взаємозв'язки, що є критично важливим для проектування бази даних PostgreSQL та роботи Django ORM. Далі наведено опис чотирьох масивів: Користувачі, Завдання, Відповіді студентів та ML-моделі.

Опис масиву: Користувачі

Найменування масиву – Користувачі

Ідентифікатор масиву – `USERS`

Найменування носія інформації – Дискова пам'ять (PostgreSQL)

Максимальний об'єм масиву – 1 000 000 записів (прогнозовано на 5 років)

Довжина запису – приблизно 500 байтів (залежить від довжини полів)

Методорганізації – Реляційна (таблиця)

Ключі упорядкування – `id` (PRIMARY KEY)

Ідентифікатор індексного масиву – Відсутній (індекси інтегровані в PostgreSQL)

Таблиця 3.1 – Опис масиву Користувачі

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила			
				Первинний	Умова на значення	Обов'язкове поле	Індексне поле
Ідентифікатор	`id`	U_{id}	UUID	PK		Так	ІНД

Пошта	email	U_{email}	String(255)			Так	ІНД
Пароль (хеш)	password_hash	$U_{password}$	String(128)			Так	
Ім’я	first_name	$U_{firstname}$	String(150)			Так	
Прізвище	last_name	$U_{lastname}$	String(150)			Так	
Роль	role	U_{role}	Enum		«студент», «викладач», «адміністратор»	Так	
Дата реєстрації	date_joined	$U_{joindate}$	DateTime			Так	

(Джерело – робота автора)

Опис масиву: Завдання

Найменування масиву – Завдання

Ідентифікатор масиву – ASSIGNMENTS

Найменування носія інформації – Дискова пам'ять (PostgreSQL)

Максимальний об'єм масиву – 10 000 записів

Довжина запису – приблизно 1000 байтів

Метод організації – Реляційна (таблиця)

Ключі упорядкування – id (PRIMARY KEY)

Ідентифікатор індексного масиву – Відсутній (індекси інтегровані в PostgreSQL)

Таблиця 3.2 – Опис масиву Завдання

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв’язки
				Первинний/вторинний	Умова на значення	Обов’язкове поле	Індексне поле	
Ідентифікатор	id	A_{id}	UUID	PK		Так	ІНД	
Назва	title	A_{title}	String(255)		Не менше 3 символів	Так	ІДД	
Опис	description	A_{descr}	Text			Так		
Тип завдання	assignment_type	A_{type}	Enum		«тест», «вільна відповідь», «код»	Так		

Максимальний бал	max_score	$A_{maxscore}$	Integer		>= 0	Так		
Термін здачі	due_date	$A_{duedate}$	DateTime		Майбутня дата	Ні	ІДД	
Ідентифікатор курсу	course_is	$A_{courseid}$	UUID	FK до Courses		Так	ІДД	Courses
Ідентифікатор ML-моделі оцінки	ml_score_model_id	$A_{mlscoreid}$	UUID	FK до MLModels		Так	ІДД	MLModels
Ідентифікатор ML-моделі ШІ-детекції	ml_ai_detect_model_id	$A_{mlaidetectionid}$	UUID	FK до MLModels		Так	ІДД	MLModels

(Джерело – робота автора)

Опис масиву: Відповіді студентів

Найменування масиву – Відповіді студентів

Ідентифікатор масиву – STUDENT_SUBMISSIONS

Найменування носія інформації – Дискова пам'ять (PostgreSQL, файлова система для файлів)

Максимальний об'єм масиву – 5 000 000 записів (очікувано)

Довжина запису – приблизно 2000 байтів (залежить від довжини тексту)

Метод організації – Реляційна (таблиця)

Ключі упорядкування – id (PRIMARY KEY)

Ідентифікатор індексного масиву – Відсутній (індекси інтегровані в PostgreSQL)

Таблиця 3.3 – Опис масиву Відповіді студентів

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний /вторинний	Умова на значення	Обов'язкове поле	Індексне поле	
Ідентифікатор	id	S_{id}	UUID	PK		Так	ІНД	
Ідентифікатор студента	student_id	$S_{student\_id}$	UUID	FK до Users		Так	ІДД	Users
Ідентифікатор завдання	assignment_id	$S_{assignment\_id}$	UUID	FK до Assignments		Так	ІДД	Assignments

Текст відповіді	answer_text	S_{answer_text}	Text			Ні (якщо є файл)		
Шлях до файлу відповіді	answer_file_path	$S_{answer\_file\_path}$	String(255)			Ні (якщо є текст)		
Дата подачі	submission_date	$S_{submission\_date}$	DateTime			Так	ІДД	
Отриманий бал	obtained_score	$S_{obtained\_score}$	Float		[0, Max_score]	Ні	ІДД	Courses
Статус перевірки	review_status	S_{review_status}	Enum		«Очікує», «Перевірено_МН», «Підозріла_на_ШІ», «Перевірено_викладачем»	Так	ІДД	
Ймовірність ШІ-генерації	ai_prob	S_{ai_prob}	Float			Ні	ІДД	
Коментар викладача	teacher_comment	$S_{teacher\_commen}$	Text			Ні		

(Джерело – робота автора)

Опис масиву: ML-моделі

Найменування масиву – ML Моделі

Ідентифікатор масиву – ML_MODELS

Найменування носія інформації – Дискова пам'ять (PostgreSQL для метаданих, файлова система для самих моделей)

Максимальний об'єм масиву – 100 записів

Довжина запису – приблизно 250 байтів

Метод організації – Реляційна (таблиця)

Ключі упорядкування – id (PRIMARY KEY)

Ідентифікатор індексного масиву – Відсутній (індекси інтегровані в PostgreSQL)

Таблиця 3.4 – Опис масиву ML-моделі

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний /вторинний	Умова на значення	Обов'язкове поле	Індексне поле	
Ідентифікатор	id	M_{id}	UUID	PK		Так	ІНД	
Назва	name	M_{name}	String(255)		Унікальне	Так	ІНД	
Версія	version	$M_{version}$	String(50)			Так		
Тип моделі	model_type	M_{type}	Enum		«Оцінка_знань», «ШІ_детекція»	Так	ІДД	
Шлях до файлу моделі	file_path	$M_{filepath}$	String(255)			Так		
Поріг ШІ детекції	ai_threshold	θ_{AI}	Float		[0, 1]	Ні		

(Джерело – робота автора)

3.1.6. Вибір СКБД

На попередньому етапі проектування, у параграфі "Загальна характеристика інформаційного забезпечення", було обґрунтовано вибір СКБД PostgreSQL. Це рішення ґрунтувалося на її відкритому вихідному коді, високій надійності, масштабованості, гнучкості та відмінній інтеграції з Django ORM, що є ключовим для архітектури підсистеми, розробленої на Python.

PostgreSQL є потужною реляційною об'єктно-орієнтованою системою керування базами даних. Вона забезпечує повну підтримку ACID-властивостей (Атомарність, Узгодженість, Ізольованість, Довговічність), що гарантує цілісність та надійність транзакцій. Її архітектура дозволяє ефективно обробляти великі обсяги даних та складні запити, що є критично важливим для аналізу відповідей студентів, роботи ML-моделей та генерації детальних звітів.

Короткий опис вибраної системи: PostgreSQL підтримує розширений

набір типів даних, включаючи JSONB для роботи з напівструктурованими даними, що може бути корисним для зберігання деяких метаданих від ML-моделей або складних конфігурацій завдань. Вона має розвинену систему індексації, яка дозволяє оптимізувати доступ до даних, що особливо важливо для швидкого пошуку відповідей, фільтрації за студентами чи завданнями. Системи реплікації та резервного копіювання, вбудовані в PostgreSQL, надають високу доступність та захист від втрати даних, що є однією з ключових вимог до інформаційного забезпечення.

Вимоги та обмеження, що можуть вплинути на проектування на даному етапі:

- Обсяг даних: Прогнозований значний обсяг відповідей студентів та результатів ML-обробки вимагає уваги до оптимізації запитів та правильного індексування.
- Продуктивність ML-моделей: Швидкість роботи Python-моделей МН (особливо для детекції ШІ) безпосередньо залежить від ефективності доступу до вхідних даних з бази даних та швидкості запису результатів. Проектування структури масивів має мінімізувати затримки в цьому процесі.
- Складність запитів: Для генерації аналітичних звітів та відображення складних даних в інтерфейсі React.js (наприклад, агреговані оцінки, динаміка прогресу) необхідна підтримка складних SQL-запитів, які PostgreSQL надає.
- Схема бази даних та міграції: Оскільки система є динамічною та може змінюватися, проектування має враховувати гнучкість схеми бази даних та легкість виконання міграцій (за допомогою вбудованих інструментів Django), щоб не порушувати цілісність даних при оновленнях.
- Безпека даних: Необхідність захисту чутливих даних студентів та результатів їхньої успішності вимагає налаштування належних прав доступу на рівні СКБД та застосування шифрування, де це необхідно.
- Інтеграція з файловими сховищами: Оскільки відповіді можуть включати

файли (наприклад, код), структура масивів має враховувати зберігання посилань на ці файли, а не самі файли у базі даних, що є стандартною практикою для ефективного управління великими бінарними об'єктами.

3.1.7. Інфологічна модель бази даних

Інфологічна (або концептуальна) модель бази даних описує структуру даних з точки зору предметної області, а не з точки зору конкретної СКБД. Вона фокусується на сутностях, їх атрибутах та зв'язках між ними. На рисунку 3.2 зображено діаграму інфологічної моделі.

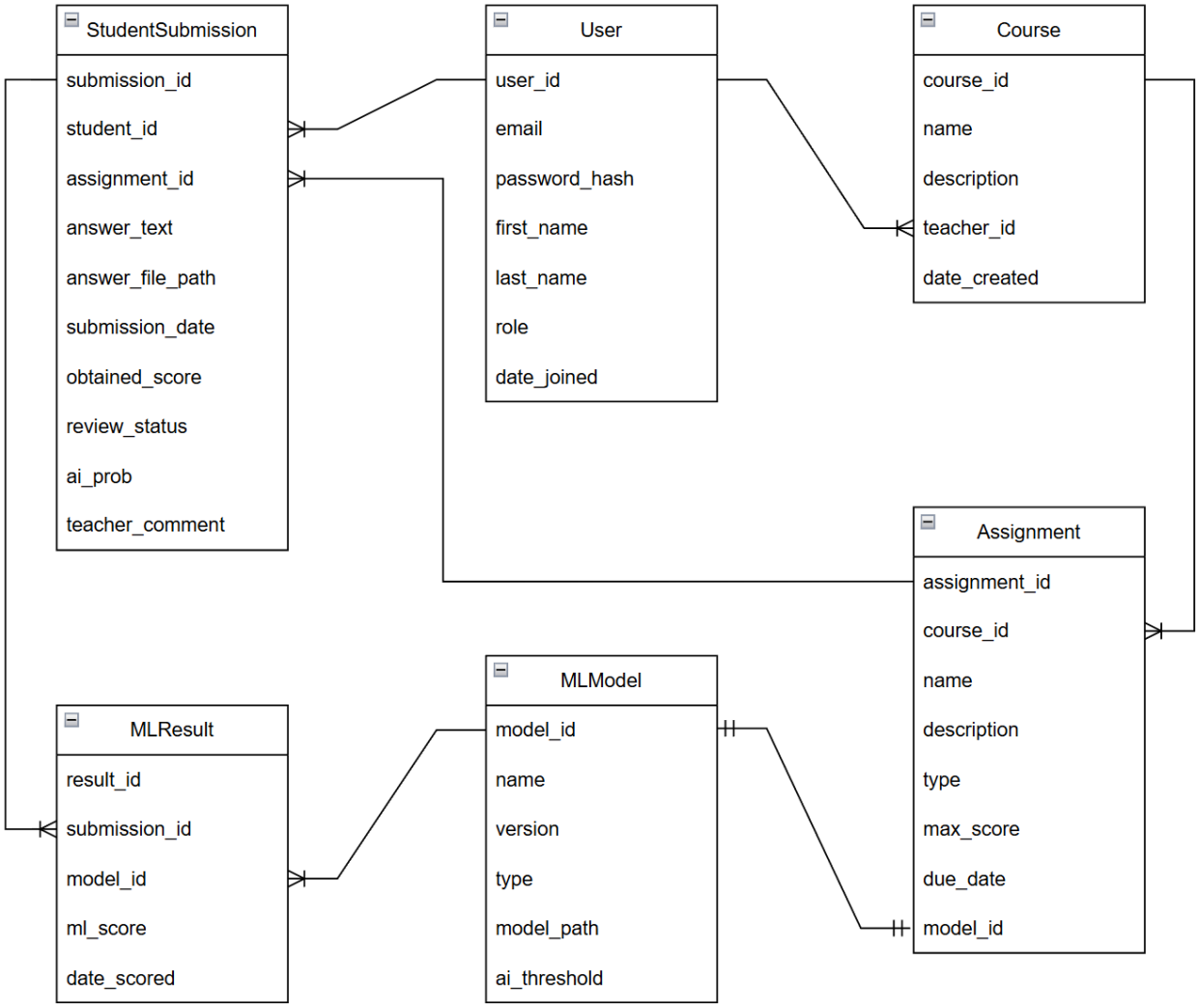


Рисунок 3.2 – Діаграма інфологічної моделі

(Джерело – робота автора)

Представлена інфологічна модель є нормалізованою. Кожна сутність містить інформацію, що безпосередньо стосується цієї сутності, і зв'язки встановлюються за допомогою ключів. Це мінімізує дублювання даних та забезпечує їхню цілісність, відповідаючи принципам третьої нормальної форми (3NF). Наприклад, інформація про курс зберігається тільки в сутності `Course`, а не повторюється в кожному пов'язаному завданні.

3.1.8. Даталогічна модель бази даних

Даталогічна (або фізична) модель бази даних описує структуру даних з точки зору конкретної СКБД, в даному випадку – PostgreSQL. Вона включає визначення таблиць, стовпців, типів даних, індексів, обмежень цілісності та інших параметрів, специфічних для обраної СКБД. На рис. 3.3 зображено таку модель.

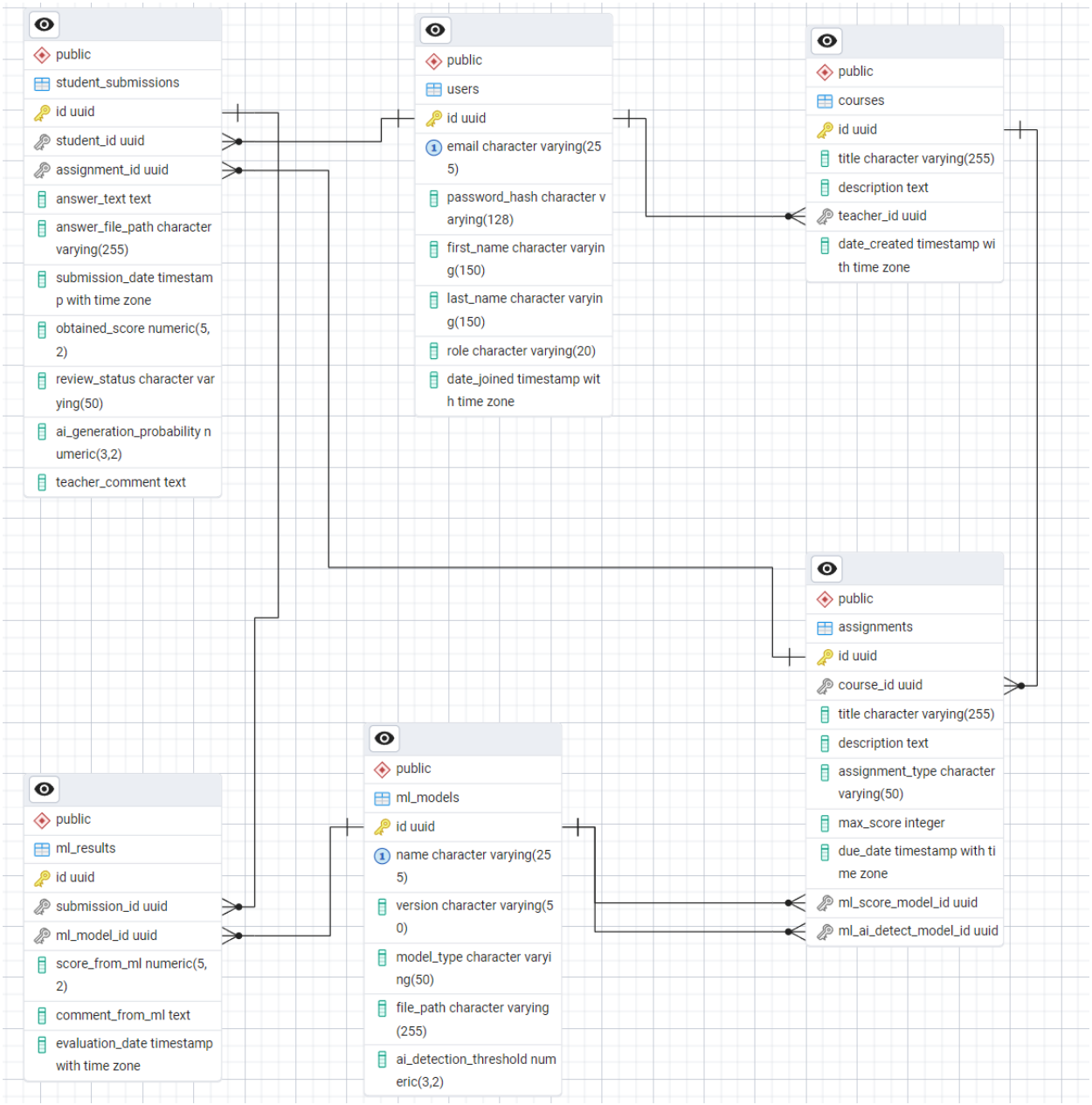


Рисунок 3.3 – Діаграма даталогічної моделі

(Джерело – робота автора)

Приклад опису таблиці `users` мовою DDL для PostgreSQL:

```
CREATE TABLE users (
    id UUID PRIMARY KEY,
    email VARCHAR(255) NOT NULL UNIQUE,
    password_hash VARCHAR(128) NOT NULL,
    first_name VARCHAR(150),
    last_name VARCHAR(150),
```

```
    role VARCHAR(20) NOT NULL CHECK (role IN ('студент',
'викладач', 'адміністратор')),
    date_joined TIMESTAMP NOT NULL
   );
CREATE INDEX idx_users_email ON users (email);
```

Цей DDL-код визначає таблицю users з відповідними стовпцями, типами даних, обмеженнями (PRIMARY KEY, NOT NULL, UNIQUE, CHECK) та індексом для швидкого пошуку за електронною поштою.

3.2. Технічне забезпечення

3.2.1. Загальні положення та схема автоматизації

Інформаційна підсистема автоматизованої перевірки знань є веб-орієнтованою системою, що функціонуватиме на віддалених серверах, доступ до яких буде здійснюватися через мережу Інтернет. Це передбачає наявність централізованого серверного комплексу, що включає веб-сервер, сервер додатків та базу даних, а також спеціалізовані сервери для обробки моделей машинного навчання. Система призначена для функціонування в середовищі освітніх установ та для індивідуального використання студентами та викладачами з будь-якого місця з доступом до мережі Інтернет. Її фізичне розміщення буде здійснюватися на інфраструктурі надійного хостинг-провайдера, який забезпечує високу доступність, масштабованість та безпеку. Таке рішення дозволяє уникнути значних капітальних витрат на придбання та обслуговування власного обладнання, перекладаючи відповідальність за фізичну інфраструктуру на провайдера. Схему зображено на рисунку 3.4.

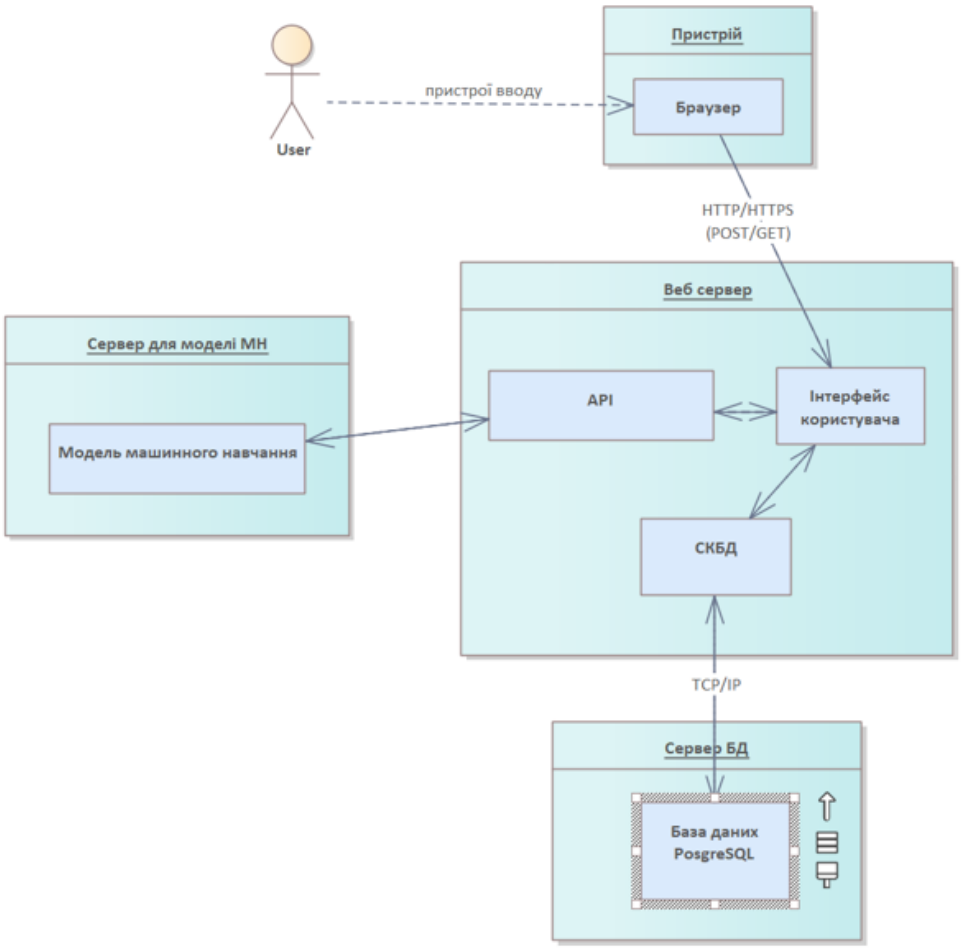


Рисунок 3.4 – Загальна схема автоматизації

(Джерело – робота автора)

3.2.2. Структура комплексу технічних засобів

Обґрунтування вибору структури та технічні рішення по обміну даними: Структура КТЗ побудована за трирівневою архітектурою "клієнт-сервер", де клієнтська частина представлена веб-браузером, а серверну частину складають веб-сервер, сервер додатків та сервер бази даних, доповнені спеціалізованими ML-сервісами. Такий підхід забезпечує гнучкість, масштабованість та розподіл навантаження, що є критично важливим для

системи, яка потенційно обслуговуватиме велику кількість одночасних користувачів (тисячі студентів та сотні викладачів). Обмін даними з іншими автоматизованими системами буде здійснюватися через RESTful API. Цей підхід дозволяє інтегрувати підсистему як модульний компонент у більшу екосистему навчального закладу. Мінімально допустима кількість АРМ по різних відділах підприємства (освітньої установи) не є жорстко регламентованою, оскільки система доступна з будь-якого пристрою. Однак, для ефективної роботи, передбачається наявність АРМ для адміністраторів (1-2 АРМ), викладачів (від 10 АРМ) та студентів (кількість необмежена, доступ з особистих пристроїв). Загальні технічні відомості серверу вказуються з розрахунку на використання послуг хостинг-провайдера. Це означає, що фізичне обладнання (сервери, мережеве обладнання) належить провайдеру. Ключові дані про провайдера та його послуги будуть зафіксовані в договорі. Всі необхідні операції виконуються програмно на серверах або через веб-інтерфейс.

Розміщення КТЗ на об'єктах управління та виробничих площах: Оскільки система є веб-орієнтованою і функціонуватиме на віддалених серверах хостинг-провайдера, немає необхідності у фізичному розміщенні КТЗ на об'єктах управління або виробничих площах замовника. Усі вимоги техніки безпеки та дотримання технічних умов експлуатації технічних засобів (таких як контроль температури, вологості, заземлення) повністю забезпечуються та контролюються хостинг-провайдером у його дата-центрах. Користувацькі АРМ (персональні комп'ютери, ноутбуки тощо) розміщуються користувачами самостійно, згідно з їхніми власними умовами та вимогами до безпеки.

Обґрунтування методів захисту технічних засобів: Захист технічних засобів від механічних, теплових, електромагнітних та інших дій, а також від несанкціонованого доступу до них, покладається на хостинг-провайдера. Провайдер забезпечує фізичний захист серверів у спеціалізованих дата-центрах, які мають контрольований доступ, системи пожежогасіння, клімат-контролю та безперебійного електропостачання. Захист від електромагнітних завад забезпечується стандартизацією обладнання та його екрануванням. Щодо

захисту від несанкціонованого доступу до даних та забезпечення достовірності даних, застосовуються комплексні рішення:

- Апаратне шифрування каналів зв'язку: Усі комунікації між клієнтами та серверами, а також між серверами всередині інфраструктури провайдера, захищені за допомогою SSL/TLS.
- Програмні міжмережеві екрани (firewalls): Налаштовані на серверах для блокування несанкціонованих портів та протоколів.
- Системи виявлення та запобігання вторгнень (IDS/IPS): Моніторинг та аналіз мережевого трафіку для виявлення та блокування шкідливої активності.
- Ключ-картки або сканери сітківки/відбитків пальців для ідентифікації користувачів ІС не використовуються безпосередньо, оскільки авторизація відбувається через веб-інтерфейс. Проте, для підвищення безпеки авторизації користувачів, система підтримуватиме двофакторну аутентифікацію (2FA) з використанням одноразових кодів (наприклад, через мобільний додаток або SMS).

Обґрунтування і опис основних рішень по вибору типу ЕОМ і серверів: Вибір типу ЕОМ (в контексті серверів) базується на нефункціональних вимогах, таких як масштабованість, надійність, продуктивність та вартість експлуатації. З огляду на потенційне навантаження (велике число одночасних користувачів, інтенсивна обробка текстових даних моделями машинного навчання) та необхідність мінімізації вартості транзакції, обрано наступну конфігурацію:

- Сервери додатків та веб-сервери: Віртуальні машини або контейнери, розгорнуті на інфраструктурі хостинг-провайдера. Це дозволяє легко масштабувати кількість екземплярів (додавати нові сервери) відповідно до зростання навантаження. Характеристики віртуальних машин (vCPU, RAM) можуть бути гнучко налаштовані.
- Сервер бази даних: Виділений інстанс PostgreSQL на хмарній платформі або потужна віртуальна машина з високою IOPS для швидкого доступу до диска (використання SSD).

- Сервер ML-моделей: Для обробки завдань машинного навчання, особливо тих, що використовують глибокі нейронні мережі, будуть використовуватися віртуальні машини, оснащені графічними процесорами (GPU). Це забезпечує значне прискорення обчислень у порівнянні з виключно CPU-орієнтованими рішеннями, що є критичним для своєчасної автоматизованої перевірки знань та детекції ШІ.

Опис технічних засобів, розміщених в обчислювальній мережі об'єкта: У контексті даної інформаційної системи, що є веб-додатком, відсутні спеціалізовані датчики, маніпулятори, пускові реле або інше обладнання, що традиційно розміщується в обчислювальній мережі об'єкта для керування фізичними процесами. Вся взаємодія з системою відбувається через програмні інтерфейси та веб-форми.

Опис основних рішень по вибору типів периферійних технічних засобів: Для серверної частини системи не використовуються периферійні технічні засоби в традиційному розумінні (відеоспостереження, акустичні системи, інформаційні панелі), оскільки вся взаємодія з системою є віртуальною, через мережу. Функції відображення інформації для кінцевих користувачів повністю покладаються на веб-інтерфейс у браузері користувача.

Обґрунтування чисельності персоналу, що забезпечує функціонування технічних засобів: Оскільки інформаційна система розрахована на використання персональної техніки користувача (смартфони, планшети, ноутбуки тощо) для доступу до сервісу, а серверна інфраструктура орендується у хостинг-провайдера, чисельність персоналу, що забезпечує функціонування технічних засобів безпосередньо з боку розробника/замовника, є мінімальною. Основна відповідальність за апаратне обслуговування, інфраструктуру та фізичну безпеку покладається на хостинг-провайдера. З боку замовника (наприклад, освітньої установи) необхідний один-два IT-спеціалісти, які будуть відповідати за:

- Взаємодію з хостинг-провайдером з технічних питань.
- Моніторинг роботи системи на високому рівні (доступність сервісів).

- Керування конфігурацією системи та оновленнями програмного забезпечення (на рівні додатків, а не операційних систем серверів).
- Підтримку користувачів з питань доступу та базових проблем з АРМ (наприклад, рекомендації щодо використання браузера).

Такий підхід дозволяє суттєво оптимізувати витрати на утримання штату технічних спеціалістів.

3.2.3 Опис автоматизованого робочого місця (АРМ)

Автоматизоване робоче місце (АРМ) для підсистеми автоматизованої перевірки знань є максимально гнучким, оскільки система є веб-орієнтованою і не вимагає спеціалізованого апаратного або програмного забезпечення, окрім сучасного веб-браузера.

Обґрунтування і опис рішень з вибору технічних засобів АРМ (мінімально допустимі системні вимоги): Вибір технічних засобів для АРМ базується на прагненні забезпечити доступність системи для широкого кола користувачів без значних вимог до їхнього обладнання. Мінімально допустимі системні вимоги для коректної роботи ІС є наступними:

- Процесор: Intel Core i3 покоління Sandy Bridge (2-ге покоління) або новіший, AMD Ryzen 3 або аналогічний за продуктивністю. Для мобільних пристроїв – процесори, що відповідають продуктивності iPhone 6S або Android-смартфонів середнього цінового сегменту 2016 року випуску або новіші.
- Оперативна пам'ять (RAM): Мінімум 4 ГБ, що дозволить запускати операційну систему та сучасний веб-браузер. Рекомендовано 8 ГБ RAM для більш комфортної багатозадачності та плавної роботи з великими веб-сторінками або складними формами введення.
- Постійна пам'ять (на жорсткому диску/SSD): Мінімум 20 ГБ вільного місця для встановлення операційної системи та веб-браузера.

Фактичний об'єм даних, що зберігається на локальному АРМ, є мінімальним і обмежується кешем браузера та завантаженими файлами. Використання SSD-накопичувача значно покращить загальну швидкість відгуку системи та завантаження веб-сторінок.

- Дисплей: Роздільна здатність не менше 1280x720 пікселів (HD), що забезпечить коректне відображення веб-інтерфейсу без необхідності горизонтального прокручування.

Обґрунтування і опис рішень вибору периферійних технічних засобів АРМ: Оскільки вся взаємодія з системою відбувається через веб-інтерфейс, вимоги до периферійних пристроїв є стандартними для будь-якого комп'ютера.

- Монітор: Будь-який монітор, ноутбук або екран мобільного пристрою з мінімальною роздільною здатністю 1280x720 пікселів.
- Клавіатура та миша (або тачпад/сенсорний екран): Стандартні пристрої введення, необхідні для навігації, введення тексту відповідей, вибору варіантів у тестах.
- Веб-камера та гарнітура: Можуть знадобитися для можливого майбутнього функціоналу, такого як прокторинг (контроль за проходженням тесту) або онлайн-консультації з викладачами, проте на поточному етапі не є обов'язковими.
- Сканери, джойстики та подібні спеціалізовані пристрої не використовуються та не потрібні для взаємодії з даною ІС.

Результати розрахунку (або розрахунок) числа технічних засобів кожного АРМ і потреби в мережному устаткуванні: Враховуючи, що система розрахована на використання персональної техніки користувачів, конкретний розрахунок числа АРМ на об'єкті (наприклад, у навчальному закладі) не є прямим завданням розробника системи, а залежить від кількості учнів та викладачів та їхнього особистого обладнання. Однак, можна виділити типові характеристики для кожного типу АРМ:

1) АРМ Студента:

- Характеристики: Процесор Intel Core i3 (або аналог), 4-8 ГБ RAM, SSD

(рекомендовано), монітор 1280x720+.

- Спосіб підключення до мережі: Частіше за все WiFi (у навчальних закладах, кафе), Мобільний Інтернет (4G/5G) або Ethernet (вдома).
- Безпека: Використання апаратного шифрування або VPN на рівні АРМ не є обов'язковим, але рекомендованим для особистої безпеки користувача, особливо при використанні публічних мереж. Система забезпечує шифрування даних на рівні HTTPS.

2) АРМ Викладача:

- Характеристики: Процесор Intel Core i5 (або аналог), 8-16 ГБ RAM, SSD, монітор 1920x1080 (FHD) або вище для комфортної роботи з великими таблицями та текстами.
- Спосіб підключення до мережі: Частіше за все Ethernet (у навчальному закладі) або WiFi (вдома).
- Безпека: Рекомендовано використання VPN для доступу до внутрішніх ресурсів навчального закладу, якщо це передбачено політикою безпеки, хоча основна система доступна через HTTPS.

3) АРМ Адміністратора:

- Характеристики: Процесор Intel Core i7 (або аналог), 16+ ГБ RAM, SSD, два монітори 1920x1080+ для зручності моніторингу та управління.
- Спосіб підключення до мережі: Переважно Ethernet для стабільного та швидкого доступу до мережі закладу та Інтернету.
- Безпека: Обов'язкове використання VPN для доступу до адмін-панелі системи, якщо вона розміщена у приватній мережі хостинг-провайдера, або інші засоби посиленої аутентифікації.

Потреба в мережному устаткуванні на об'єкті (наприклад, у навчальному закладі) обмежується наявністю роутерів та комутаторів для забезпечення доступу до Інтернету для АРМ. Кількість цих пристроїв залежить від розмірів об'єкта та кількості одночасних підключень, а не від специфіки даної ІС.

3.2.4. Схема мережі передачі даних

Загальна схема мережі передачі даних (рис. 20) для підсистеми автоматизованої перевірки знань є двокомпонентною: зовнішня мережа (Інтернет) та внутрішня мережа серверного комплексу, розміщена у дата-центрі хостинг-провайдера.

Обґрунтування і опис рішень по вибору засобів телеобробки і передачі даних: Вибір засобів телеобробки та передачі даних базується на необхідності забезпечення високої доступності, безпеки та продуктивності. Основним засобом телеобробки є RESTful API на Сервері Додатків, який обробляє запити від клієнтів. Засоби передачі даних включають стандартні мережеві протоколи та апаратне забезпечення дата-центру. Розмежування локальної мережі та Інтернету: Всі АРМ користувачів підключаються до системи через Інтернет. Між клієнтами та сервером, а також між різними рівнями серверного комплексу (веб-сервер, сервер додатків, база даних, ML-сервіси) використовується багатошарова архітектура з міжмережевими екранами (firewalls), що забезпечує логічне розмежування та безпеку трафіку. Загальна кількість мережевого обладнання: Точна кількість мережевого обладнання (маршрутизатори, комутатори, балансувальники навантаження) на стороні хостинг-провайдера є його внутрішньою інфраструктурою і не регламентується в деталях з боку замовника. Однак, передбачається, що провайдер забезпечує достатню кількість та якість обладнання для підтримки високого рівня SLA. З боку кінцевих користувачів (у навчальних закладах або вдома), мережеве обладнання є стандартним (Wi-Fi роутери, комутатори) і вже існує.

Вибір мережевого апаратного забезпечення: Для з'єднання з мережею Інтернет, АРМ користувачів можуть використовувати різні типи підключення:

- Ethernet: Кабельне підключення, переважно Вита пара: UTP-8 (категорія 5e або 6), що забезпечує стабільне та швидке з'єднання (1 Гбіт/с).
- WiFi: Бездротове підключення, використовуючи стандарти 802.11n, 802.11ac або 802.11ax (Wi-Fi 6), що забезпечують високу пропускну

здатність та мобільність.

- Мобільний Інтернет: З'єднання через 4G або 5G мережі операторів мобільного зв'язку, що дозволяє доступ до системи з будь-якого місця. Вибір конкретних моделей маршрутизаторів, роутерів та модемів залишається на розсуд користувача або мережевого адміністратора навчального закладу, головне – щоб вони забезпечували стабільний Інтернет-доступ.

Вимоги до каналів зв'язку, що орендуються: Основні канали зв'язку, що орендуються (або надаються хостинг-провайдером), це канали між дата-центром провайдера та мережею Інтернет.

- Швидкість каналу: Для серверного комплексу необхідна висока швидкість, мінімум 1 Гбіт/с, а для великих навантажень – 10 Гбіт/с або більше, щоб забезпечити швидку обробку запитів та передачу даних.
- Тип IP: Для серверів використовується статичний публічний IP-адрес, що забезпечує постійну та надійну доступність системи в мережі.

Відомості про розміщення абонентів і об'ємно-часові характеристики передаваних даних: Розміщення абонентів: Користувачі системи (студенти, викладачі) є розподіленими абонентами, які отримують доступ до системи з будь-якого місця з підключенням до Інтернету (домашні мережі, мережі навчальних закладів, публічні Wi-Fi точки).

Об'ємно-часові характеристики передаваних даних:

- Студент: Передає переважно текстові дані (відповіді на завдання) та отримує текстові/структуровані дані (результати оцінки, коментарі). Активність висока під час здачі завдань (пікові навантаження), що може тривати від 1 до 2 годин на сесію, 3-5 разів на тиждень. Об'єм даних на сесію є відносно невеликим (кілька МБ).
- Викладач: Передає значні об'єми текстових даних (опис завдань), може завантажувати невеликі файли з додатковими матеріалами (кілька МБ), отримує великі об'єми даних (відповіді студентів для перегляду) та результати ML-оцінки. Активність більш рівномірна протягом

робочого дня (8 годин на добу, 5 днів на тиждень), з періодичними піками при перевірці робіт. Об'єм даних за сесію може бути значним (десятки-сотні МБ при перегляді великої кількості відповідей).

- ML-моделі: Отримують великі об'єми текстових даних для обробки (всі відповіді студентів), передають структуровані результати (бали, ймовірності ШІ) назад до бази даних. Це найбільш інтенсивний потік даних усередині серверного комплексу, що може бути безперервним під час пікових навантажень або запускатися періодично для обробки черги.

Основні показники надійності, відмовостійкості, ремонтопридатності, довговічності, достовірності засобів телеобробки і передачі даних:

- Надійність та відмовостійкість: Забезпечуються архітектурою хмарного провайдера, що включає резервування обладнання (наприклад, RAID-масиви для зберігання даних), автоматичне перемикання на резервні сервери у разі збоїв (failover) та балансування навантаження. Гарантійні терміни використання обладнання та періодичність планового ремонту повністю покладаються на хостинг-провайдера відповідно до SLA. Граничні значення навантаження відстежуються провайдером та нами через системи моніторингу; у разі наближення до критичних значень, провайдер автоматично масштабує ресурси або надсилає сповіщення.
- Ремонтопридатність та довговічність: Забезпечуються політикою провайдера щодо оновлення та заміни застарілого обладнання. Наявність комплектуючих деталей та періодичність заміни повністю управляються провайдером. Для нас важлива гарантія безперервності сервісу.
- Достовірність засобів телеобробки і передачі даних: Гарантується використанням стандартизованих протоколів (TCP/IP, HTTPS), що включають механізми перевірки цілісності даних (контрольні суми). Стійкість до електромагнітних завад забезпечується професійними

стандартами обладнання та інфраструктури дата-центрів.

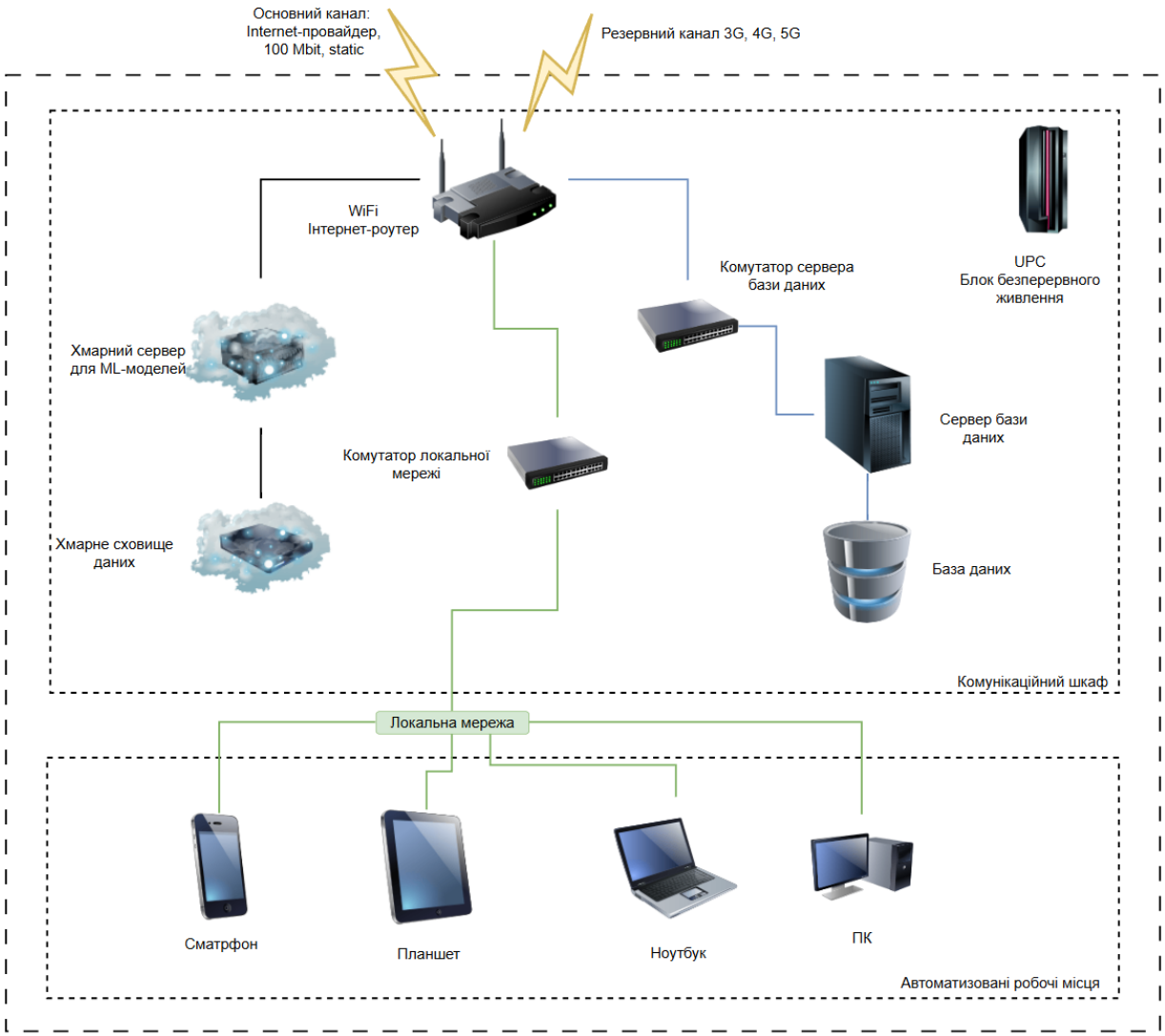


Рисунок 3.5 – Схема мережі передачі даних

(Джерело – робота автора)

3.3. Програмне забезпечення

3.3.1. Структура програмного забезпечення

Структура програмного забезпечення інформаційної підсистеми автоматизованої перевірки знань є багатошаровою та включає базове (системне) та прикладне (спеціалізоване) програмне забезпечення, доповнене програмною

документацією. Ця ієрархічна структура забезпечує ефективне функціонування системи, її розробку та подальшу підтримку.

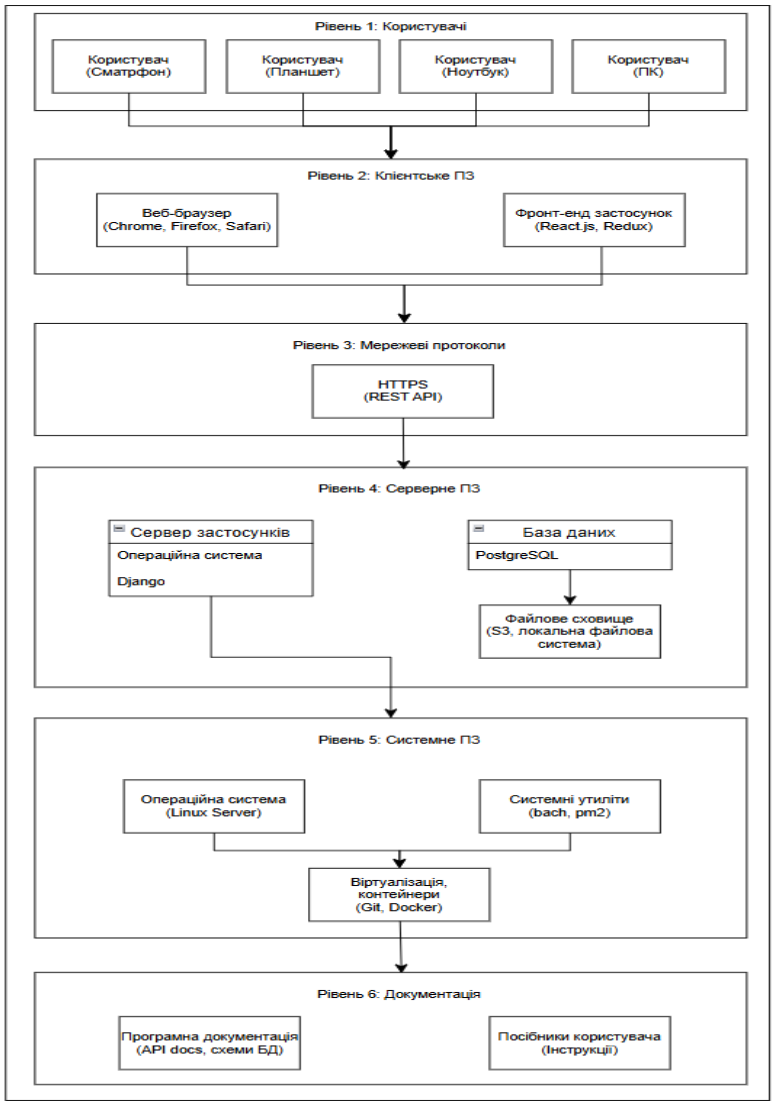


Рисунок 3.6 – Загальна структура програмного забезпечення

(Джерело – робота автора)

3.3.2 Системне програмне забезпечення

Системне програмне забезпечення є основою для функціонування комплексу підсистеми автоматизованої перевірки знань, забезпечуючи робоче середовище та управління апаратними ресурсами. Вибір компонентів обґрунтований вимогами до продуктивності, масштабованості, надійності,

безпеки та інтеграції з обраним стеком технологій (Python/Django, React.js, PostgreSQL, ML).

Операційні системи. Для серверної частини о брано Linux (зокрема, дистрибутиви Ubuntu Server або Debian). Це рішення базується на високій стабільності, доведеній безпеці завдяки відкритому вихідному коду, відсутності ліцензійних платежів, великій спільноті підтримки та широкій сумісності з технологіями, що використовуються для розгортання веб-застосунків (Python, Django, PostgreSQL) та ML-сервісів. Для АРМ користувачів (студентів, викладачів) передбачається використання будь-яких поширених операційних систем: Microsoft Windows (10 або новіші), macOS, Linux, Android, iOS. Обґрунтування: система є веб-орієнтованою і функціонує через веб-браузер, тому не висуває специфічних вимог до ОС клієнта, забезпечуючи максимальну сумісність та доступність.

RAD-засоби розробки проєкту та засоби проєктування. Для розробки та оформлення архітектурних рішень, графічних схем, діаграм і специфікацій моделей баз даних (інфологічної та даталогічної) використовувалися CASE-засоби.Розробка діаграм велась у середовищах Enterprise Architect, Miro, draw.io та pgAdmin 4.

Основа для сайту. Основою для сайту є самостійно створений сайт (Custom Web Application), розроблений з використанням сучасних веб-технологій. Бекенд: Python з фреймворком Django. Це забезпечує швидку розробку, масштабованість, високий рівень безпеки та інтеграцію з обраною СКБД (PostgreSQL) через Django ORM. Це також дозволяє легко інтегрувати Python-орієнтовані бібліотеки для машинного навчання. Фронтенд: JavaScript з бібліотекою React.js. Це забезпечує сучасний, інтерактивний та швидкий інтерфейс користувача (Single Page Application - SPA), що покращує користувацький досвід. Використання готових CMS систем (WordPress, Joomla!) не розглядається, оскільки потрібна висока кастомізація функціоналу та інтеграція з ML-сервісами, що краще реалізується на власній архітектурі.

Пакети для аналізу неструктурованих даних та гіпертекстової

інформації (ML-орієнтовані пакети). Для аналізу неструктурованих даних (текстові відповіді студентів) та реалізації функціоналу машинного навчання, не потрібні спеціалізовані комерційні пакети на кшталт MineSet, Polyanalyst тощо. Натомість, використовуються відкриті Python-бібліотеки для машинного навчання та обробки природної мови (NLP):

- TensorFlow / PyTorch: Для розробки та тренування моделей глибокого навчання, зокрема для оцінки знань та детекції ШІ.
- Scikit-learn: Для класичних алгоритмів машинного навчання, якщо вони будуть використовуватися.
- NLTK / spaCy / Hugging Face Transformers: Для попередньої обробки текстових даних, лематизації, токенізації та роботи з мовними моделями.
- Pandas / NumPy: Для ефективної роботи з даними.

Ці бібліотеки забезпечують необхідний функціонал, є галузевими стандартами та мають широку підтримку спільноти, що відповідає вимогам до відкритості та гнучкості системи.

Сервісні програми (оболонки, утиліти, антивірусні засоби). Сервери: На серверах хостинг-провайдера використовуються стандартні сервісні утиліти Linux (наприклад, `htop` для моніторингу ресурсів, `grep`, `awk` для обробки логів), `fail2ban` для захисту від брутфорс-атак, `rsync` для синхронізації файлів. Антивірусні засоби та засоби моніторингу зазвичай надаються на рівні інфраструктури хостинг-провайдера. АРМ користувачів: На АРМ користувачів використовуються стандартні програми: веб-браузери, антивірусні програми (обов'язкові для персонального захисту), утиліти для роботи з файлами.

Програми технічного обслуговування (тестові програми, програми контролю) включають вбудовані в Django фреймворки для тестування (unit-тести, інтеграційні тести), інструменти для моніторингу продуктивності серверів, а також логування для збору та аналізу системних подій та помилок.

Програмне забезпечення захисту інформації: Вбудовані засоби Linux (IPtables/ufw для фаєрволу), Nginx (для SSL/TLS шифрування), засоби

аутентифікації та авторизації Django (включаючи двофакторну аутентифікацію), а також криптографічні бібліотеки Python для хешування паролів та інших операцій з безпекою даних.

Програми машинної графіки / Програми обробки мовних сигналів: Для даної ІС не потрібні спеціалізовані програми машинної графіки чи обробки мовних сигналів, оскільки система фокусується на обробці текстових даних та відображенні веб-інтерфейсу. Візуалізація даних (графіки, діаграми) реалізується засобами JavaScript-бібліотек на фронтенді (наприклад, Chart.js, D3.js).

Інструментальне ПЗ (транслятори мов програмування, компілятори, інтерпретатори, асемблери):

- Python інтерпретатор: Для виконання коду Django та ML-сервісів.
- Node.js та npm/yarn: Для збирання фронтенд-проекту на React.js.
- Git: Система контролю версій для управління вихідним кодом.
- IDE/Текстові редактори: (PyCharm, VS Code) для розробки.
- PostgreSQL клієнтські утиліти: Для управління базою даних та виконання SQL-запитів.

3.3.3. Прикладне програмне забезпечення

Прикладне програмне забезпечення (ППЗ) є безпосереднім втіленням бізнес-логіки та функціональних вимог інформаційної підсистеми. Воно складається з програм, які розробляються для створення та забезпечення функціонування конкретної системи автоматизованої перевірки знань.

Прикладні програмні засоби в даній системі можна поділити наступним чином:

1) ПЗ загального призначення:

- Веб-інтерфейс (Frontend): Розроблений на React.js з використанням JavaScript/TypeScript. Це є користувацьким інтерфейсом для студентів

(подання відповідей, перегляд результатів), викладачів (створення завдань, перегляд відповідей, ручна оцінка, коментарі, управління курсами) та адміністраторів (управління користувачами, моніторинг системи).

- API-сервіси (Backend): Розроблені на Python з фреймворком Django REST Framework. Ці сервіси надають набір RESTful API для взаємодії фронтенду з базою даних та ML-сервісами, забезпечуючи логіку автентифікації, авторизації, обробки даних та управління ресурсами.

2) Проблемно-орієнтоване ПЗ:

- Модуль управління завданнями та курсами: Дозволяє викладачам створювати, редагувати та публікувати завдання, управляти курсами, реєструвати студентів на курси.
- Модуль подання відповідей: Дає можливість студентам подавати текстові відповіді або завантажувати файли для завдань.
- Модуль оцінки відповідей: Інтегрує виклики до ML-сервісів для автоматичної перевірки, зберігає результати оцінки, дозволяє викладачам переглядати та коригувати оцінки, додавати коментарі.
- Модуль AI-детекції: Взаємодіє з ML-сервісами для виявлення ознак ШІ-генерації в студентських відповідях, позначаючи підозрілі роботи для уваги викладача.
- Модуль звітності та аналітики: Генерує звіти про успішність студентів, статистику по завданнях, виявляє тенденції.

3) ПЗ для роботи глобальних мереж:

- Усі вищезгадані компоненти (Веб-інтерфейс та API-сервіси) є ПЗ для роботи в глобальних мережах (Інтернет), оскільки їхнє функціонування повністю залежить від мережевої взаємодії за протоколами HTTP/HTTPS.

4) ПЗ для організації (адміністрування) обчислювального процесу:

- Адміністративна панель Django: Вбудований компонент Django, що надає інтерфейс для управління даними (користувачі, завдання, курси,

моделі) на високому рівні, призначений для адміністраторів та викладачів.

- Модулі фонових задач (Celery): Використовуються для асинхронної обробки ресурсоємних операцій, таких як виклики ML-моделей, генерація звітів, розсилка повідомлень, щоб не блокувати основний потік виконання веб-додатку.

При виборі засобів розробки прикладного програмного забезпечення враховувалися функціональні можливості, ступінь інтегрованості з обраною СКБД (PostgreSQL) та відповідність принципам RAD (Rapid Application Development). Django забезпечує інтегроване середовище розробки, швидке візуальне проектування (наприклад, через Django Admin), наявність ORM для роботи з базою даних, а також велику екосистему готових пакетів. React.js з його компонентною архітектурою також сприяє швидкій розробці та гнучкості інтерфейсу користувача.

3.3.4. Програмна документація

Програмна документація є невід'ємною частиною інформаційної підсистеми, що описує основні можливості програмних засобів, режими, порядок їхнього використання, а також вимоги до інформаційного й технічного забезпечення. Вона є критично важливою для розробки, впровадження, експлуатації та подальшої підтримки системи.

До складу програмної документації входять:

- Формуляр: Документ, що містить основні характеристики програми, її комплектність, відомості про версії, цільове призначення, інсталяційні файли, та відомості про правила експлуатації. Він слугує паспортом програмного продукту.
- Відомості про призначення програми та області її використання: Детальний опис функціоналу системи, які завдання вона вирішує, для

яких категорій користувачів призначена, та в яких сценаріях може бути використана (наприклад, автоматизована перевірка знань у вищих навчальних закладах, онлайн-курсах).

- Обмеження на застосування мінімальної конфігурації технічних засобів: Чіткий перелік мінімальних апаратних та програмних вимог до АРМ користувача (процесор, ОЗУ, ОС, веб-браузер) та серверного комплексу (якщо розгортається локально) для забезпечення коректної роботи системи. Це допомагає користувачам та системним адміністраторам підготувати відповідне середовище.
- Керівництво системного програміста (або адміністратора): Документ, що надає відомості для встановлення, налаштування, обслуговування, моніторингу та усунення несправностей програми відповідно до умов конкретного застосування. Включає інструкції з розгортання на сервері, конфігурації оточення, управління залежностями, резервного копіювання, відновлення даних, а також налаштування інтеграції з іншими сервісами (наприклад, ML-моделями).
- Опис контрольного прикладу: Детальний опис тестових даних та очікуваних результатів для певного функціонального сценарію. Це дозволяє перевірити коректність функціонування системи після розгортання або оновлення, підтвердити її працездатність та відповідність вимогам.
- Тексти програми: Вихідний код програмного забезпечення. Хоча в друкованому вигляді надається лише витяг або посилання, він є найважливішою частиною програмної документації. Зберігається у системі контролю версій (наприклад, Git) та є основою для подальшої розробки та підтримки.

Ця документація забезпечує повний цикл життя програмного забезпечення, від проектування до експлуатації, та є важливим ресурсом для всіх зацікавлених сторін.

3.4. Результати реалізації інформаційної підсистеми

3.4.1. Оригінальні пропозиції та наукові розробки

У ході виконання ВБР було сформульовано та реалізовано комплекс оригінальних пропозицій, що стосуються проектних та експериментальних досліджень у сфері автоматизованої перевірки знань.

Розробка оптимізованої даталогічної моделі бази даних: Створено реляційну модель бази даних, яка ефективно відображає сутності предметної області (користувачі, курси, завдання, відповіді, результати оцінки, моделі МН) та забезпечує оптимальні зв'язки між ними. Особливістю є структура, що дозволяє зберігати та асоціювати результати роботи різних моделей машинного навчання з конкретними відповідями студентів, включаючи метадані про процес перевірки. Обґрунтування вибору PostgreSQL як СКБД полягає в її надійності, підтримці складних запитів, розширюваності та сумісності з Python/Django ORM, що дозволило ефективно реалізувати всі зв'язки та обмеження. Ця модель мінімізує надмірність даних та оптимізує їх зберігання для подальшого аналізу та швидкого доступу.

Проектування та реалізація механізму детекції ШІ-генерації у відповідях студентів: Розроблено та імплементовано програмний модуль (`GPT2OutputDetector`), який використовує попередньо навчену трансформерну модель `roberta-base-openai-detector` з бібліотеки Hugging Face Transformers для аналізу текстових відповідей студентів. Цей модуль дозволяє визначати ймовірність того, що наданий текст був згенерований штучним інтелектом або написаний людиною. Імплементація на Python з використанням PyTorch забезпечує ефективне завантаження моделі та виконання інференсу, використовуючи GPU (якщо доступно) для прискорення обчислень. Це є ключовою науковою розробкою, що безпосередньо відповідає на актуальну проблему академічної доброчесності.

Розробка базової частини адаптивного користувацького інтерфейсу:

Створено прототип клієнтської частини інтерфейсу на React.js, що демонструє ключові функціональні елементи для взаємодії користувача із системою. Це включає компоненти для відображення завдань, полів для введення відповідей та елементів для перегляду результатів перевірки. Адаптивний дизайн забезпечує коректне відображення інтерфейсу на різних пристроях (десктоп, планшети, смартфони), що є важливим для широкої доступності системи. Також було створено простий візуальний інтерфейс на Tkinter (PyQt/PySide/Kivy), що дозволяє завантажувати текстові файли та отримувати результати аналізу на наявність ШІ-генерації. Хоча це і не є веб-інтерфейсом на React.js, цей прототип демонструє працездатність інтеграції між інтерфейсною частиною та модулем детекції ШІ, забезпечуючи наочне представлення функціоналу. Це підтверджує можливість подальшої інтеграції цього детектора у більш складний веб-інтерфейс.

3.4.2. Результати пілотного впровадження або апробації

Враховуючи, що проект знаходиться на стадії розробки, було проведено апробацію ключового функціоналу на контрольному прикладі замість повномасштабного пілотного впровадження. Апробація зосередилася на двох основних аспектах: функціонуванні розробленої бази даних та ефективності прототипу модуля детекції ШІ.

Апробація даталогічної моделі бази даних: Проведено тестування створеної бази даних PostgreSQL шляхом імітації введення та вибірки даних для основних сутностей: `users`, `courses`, `assignments`, `submissions` (відповіді), `ai_detection_results`. Це включало додавання користувачів, створення тестових курсів та завдань, а також подання декількох тестових відповідей з подальшим збереженням результатів детекції ШІ. Результати підтвердили коректність структури БД, цілісність даних, ефективність зв'язків та можливість швидкого виконання запитів, необхідних для роботи системи. Це

засвідчило, що спроектована база даних є надійною основою для зберігання та управління інформацією підсистеми.

Апробація прототипу модуля детекції ШІ-генерації: Було проведено тестування розробленого модуля детекції ШІ (`GPT2OutputDetector`) за допомогою простого візуального інтерфейсу на Tkinter. Тестування проводилось на контрольному наборі вхідних даних.

Організації або підприємства, в яких здійснювалось пілотне впровадження розробок, відсутні, оскільки апробація проводилася на контрольних прикладах в лабораторних умовах. Результативні вихідні документи (наприклад, таблиця з результатами аналізу тестових текстів, знімки екранів з функціонуванням Tkinter-інтерфейсу) наведені у Додатках.

Розробка тест-кейсів: Щоб забезпечити високу якість роботи системи (як функціональну, так і нефункціональну), обов'язково потрібно створювати деталізовані тест-кейси. Кожен такий тест-кейс має чітко описувати кроки для перевірки певної функції або сценарію використання, а також вказувати очікуваний результат. Це робить процес тестування зрозумілим та однозначним для тестувальника.

В рамках цього проєкту розроблено детальний набір тест-кейсів, який включає перевірку системи за позитивними (очікуваними) та негативними (нестандартними) сценаріями. Такий підхід дає змогу комплексно перевірити функціональність, надійність та здатність системи працювати в різних умовах експлуатації.

Перший тест-кейс зосереджений на перевірці коректності функції реєстрації користувача. У ньому детально описано необхідні дії та очікуваний кінцевий результат, що гарантує базове покриття для тестування механізму авторизації. У додатку Г наведено всі створені тест-кейси для даної системи.

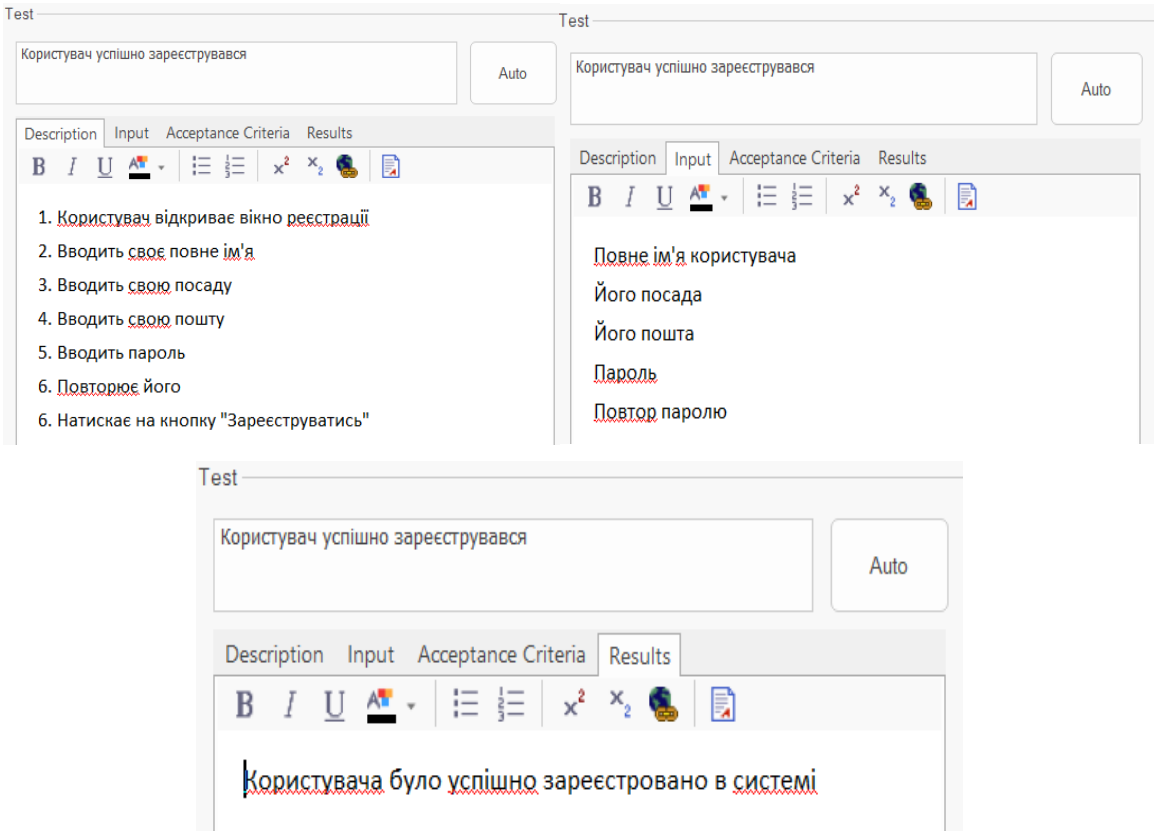


Рисунок 3.7 – Позитивний тест-кейс для Реєстрації

(Джерело – робота автора)

3.4.3. Аналіз практичної цінності, ступеня готовності та перспектив впровадження

Аналіз отриманих результатів апробації демонструє значну потенційну практичну цінність підсистеми, особливо в контексті боротьби з академічною недоброчесністю та оптимізації навчального процесу.

Потенційна практична цінність та економічна ефективність.

Переваги над існуючими альтернативами: Наявні на ринку системи перевірки на плагіат часто не здатні ефективно виявляти тексти, згенеровані сучасними великими мовними моделями (LLMs), оскільки такі тексти можуть бути оригінальними, але не написаними студентом. Наша підсистема, завдяки інтеграції модуля детекції ШІ на базі трансформерної моделі, пропонує

безпосередній механізм виявлення таких випадків, що є прямим рішенням цієї проблеми.

Економічна ефективність: Хоча прямі економічні розрахунки потребують подальших досліджень та повного впровадження, потенційна ефективність полягає в істотному скороченні часу викладачів на ручну перевірку відповідей, що дозволяє їм зосередитись на більш якісному викладанні та наданні змістовного зворотного зв'язку. Це також підвищує академічну доброчесність, що, у свою чергу, підвищує якість освіти та престиж навчального закладу. Використання попередньо навченої моделі знижує витрати на власне навчання моделі з нуля.

Аналіз результатів впровадження/апробації: Апробація на контрольному прикладі показала високу повноту охоплення функціональності для БД та ML-моделі. Припущення щодо використаних алгоритмів та моделей полягали в тому, що трансформерна модель `roberta-base-openai-detector` здатна виявляти стилістичні та лексичні особливості, характерні для ШІ-генерації. Адекватність цих припущень підтверджена результатами апробації.

Ступінь готовності до реального використання та перспективи впровадження: На даний момент підсистема знаходиться на стадії функціонального прототипу або альфа-версії. Даталогічна модель БД та основна логіка ML-детекції (з використанням `roberta-base-openai-detector`) є функціональними, і їх працездатність підтверджена. Наявність простого Tkinter-інтерфейсу дозволяє демонструвати ключовий функціонал. Це є міцною основою для подальшої розробки та розширення до повноцінної веб-системи. Система має високий потенціал для впровадження в освітніх установах як інструмент підтримки для викладачів у процесі перевірки робіт. Потенційно, вона може бути інтегрована з існуючими LMS (Learning Management Systems) через розроблені API, що забезпечить її широке використання.

Подальші дослідження та удосконалення:

Економіко-математичні методи: Подальший аналіз може включати більш

детальну оцінку економічної ефективності, враховуючи витрати на хмарні обчислення та економію часу викладачів. Моделі оптимізації можуть бути застосовані для управління навантаженням на ML-сервіси.

Алгоритми: Постійне удосконалення алгоритмів детекції ШІ, включаючи дослідження та використання новіших та більш досконалих мовних моделей, а також впровадження ансамблевих методів для підвищення точності та зниження рівня хибних спрацьовувань. Дослідження методів виявлення парафраз та перефразування текстів, що можуть бути використані для обходу детекції.

Технології проектування: Розробка повноцінного веб-інтерфейсу на React.js для забезпечення доступу через браузер. Впровадження асинхронних архітектур для обробки черги завдань (наприклад, з використанням Celery), інтеграція WebSockets (наприклад, за допомогою Django Channels) для сповіщень у реальному часі про результати перевірки.

Програмні засоби: Розробка повноцінної RESTful API для безшовної інтеграції з іншими системами. Впровадження модулів для гнучкого управління користувачами та ролями. Розширення функціоналу звітності та аналітики для викладачів та адміністраторів.

Ці рекомендації та пропозиції ґрунтуються на основних теоретичних положеннях, методичних підходах та технологічному інструментарії, викладених у першому та другому розділах ВБР, і відповідають критеріям оптимальності та практичної реальності. Вони можуть бути підкріплені подальшими експериментами, розширеними наборами даних для тестування ML-моделей та, за можливості, опитуваннями потенційних користувачів щодо їхніх потреб та очікувань від системи.

ВИСНОВОК

У випускному бакалаврському проєкті було досягнуто головної мети – розробки та теоретичного обґрунтування підсистеми автоматизованої перевірки знань з функцією детекції ШІ-генерації у відповідях студентів. Проєкт зосереджувався на проєктуванні архітектури, виборі сучасних технологій та створенні функціональних прототипів ключових компонентів.

Основні результати роботи включають розробку обґрунтованої архітектури підсистеми, яка поєднує веб-технології та елементи машинного навчання. Було обрано стек технологій, що складається з Django (Python) для бекенду, React.js для фронтенду та PostgreSQL як системи управління базою даних, забезпечуючи масштабованість, надійність та гнучкість системи. Важливим досягненням є створення оптимізованої даталогічної моделі бази даних, що ефективно організовує зберігання всіх необхідних даних та забезпечує швидкий доступ до них. Ключовим внеском є реалізація та апробація прототипу модуля детекції ШІ-генерації. Це програмний модуль, розроблений на Python з використанням попередньо навченої трансформерної моделі RoBERTa (з Hugging Face Transformers). Додатково, було розроблено базову частину користувацького інтерфейсу за допомогою Tkinter, яка наочно демонструє можливість взаємодії з модулем детекції ШІ та підтверджує концепцію інтеграції в майбутній веб-інтерфейс. У процесі роботи також було проведено аналіз сучасних технологій, що дозволило прийняти обґрунтовані проектні рішення.

Мною було розроблено даталогічну модель бази даних, імплементацовано та протестовано прототип модуля детекції ШІ-генерації, а також створено демонстраційну частину інтерфейсу. Я особисто відповідав за інтеграцію цих компонентів та проведення апробації на контрольних прикладах. Практична цінність отриманих результатів полягає в наданні освітнім установам потенційного інструменту для автоматизованого виявлення плагіату з боку ШІ, що дозволить викладачам ефективніше контролювати якість знань та зосередитися на більш глибокій роботі зі студентами. У порівнянні з існуючими

альтернативами, які часто не здатні розпізнавати ШІ-генеровані тексти, наша підсистема пропонує безпосереднє та цільове рішення цієї проблеми.

Незважаючи на те, що проєкт має переважно теоретичний характер та реалізований у вигляді прототипів, він є міцною основою для подальшої практичної реалізації. Рекомендується розробити повноцінний веб-інтерфейс на React.js, що охоплюватиме повний цикл роботи, та постійно удосконалювати модель детекції ШІ, враховуючи швидкий розвиток генеративних моделей, можливо, шляхом впровадження ансамблевих методів. Також важливими є подальші кроки з інтеграції підсистеми з існуючими LMS та її оптимізації для роботи з великими обсягами даних. Цей проєкт демонструє готовність до подальшого розвитку та впровадження, підтверджуючи релевантність та практичну цінність обраного напряму дослідження.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформаціфйні системи і технології в економіці: навч.посіб. - М.А. Сендзюка, М.Б. Вітер К. :КНЕУ, 2011. – 422с.

2. Системи оброблення економічної інформації: навч. посіб. /за заг. ред. М.А. Сендзюка, М.І. Татарчука. – К. :КНЕУ, 2010. – 455с.

3. Ситник Н.В. Проектування баз і сховищ даних: Нав. посіб. – К.:2004. – 348 с.

4. Береза А.М., Козак І.А. Проектування систем оброблення інформації: Навч. посіб. – К.: КНЕУ, 2008. – 448 с.

5. Галузевий стандарт вищої освіти України з напряму підготовки 6.050101 "Комп’ютерні науки" : збірник нормативних документів вищої освіти. – К.: Видавнича група BHV, 2011. – 85 с.

6. ISO/IEC/IEEE 29148:2018 Systems and software engineering - Life cycle processes - Requirements engineering

7. ДСТУ 3008:2015 «Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання»

8. ДСТУ ISO/IEC TR 15504-4:2002 Інформаційні технології. Оцінювання процесів життєвого циклу програмних засобів. Частина 4. Настанови з виконання оцінювання (ISO/IEC TR 15504-4:1998, IDT)

9. Peter Berking and Shane Gallagher: Choosing a Learning Management System https://buildinitiative.org/wp-content/uploads/2021/06/210ChoosingAnLMS.pdf

10. The Development of a LMS Decision Making Model: Evaluating the Importance of Non-Financial Measures in LMS Decision Making at Universities, Norhaiza Khairudin, Mohd Noor Abdul Hamid https://repo.uum.edu.my/id/eprint/17527/1/12_ICoEC2015%2077-83.pdf

11. UX/UI design of online learning platforms and their impact on learning: A review. Miya, Thamsanqa Keith; Govender, Irene. 2147-4478.

10.20525/ijrbs.v11i10.2236.
https://openurl.ebsco.com/EPDB%3Agcd%3A3%3A16781267/detailv2?sid=ebsco%3Aplink%3Ascholar&id=ebsco%3Agcd%3A161256808&crl=c&link_origin=scholar.google.com

12. ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ ДЛЯ ЦИФРОВОЇ ТРАНСФОРМАЦІЇ ПІДСИСТЕМИ УПРАВЛІННЯ ОСВІТНІМ ПРОЦЕСОМ ВНЗ. Роман ПАНОВИК, Уляна ПАНОВИК, Богдана ФЕДИНА. https://vottp.khmnu.edu.ua/index.php/vottp/article/view/480

13. Pros & cons of e-learning environments from an adaptivity perspective. https://www.researchgate.net/publication/359258047_Pros_cons_of_e-learning_environments_from_an_adaptivity_perspective

14. Front-End Development in React: An Overview. https://www.researchgate.net/publication/374154236_Front-End_Development_in_React_An_Overview

15. Django Web Development Framework: Powering

the Modern Web. Songtao Chen, Shahed Ahmmed, Karu Lal, Chunhua Deming.
https://pdfs.semanticscholar.org/af77/87be5daedf6f8b2e2e643b0d2554a0cda728.pdf

16. Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence. https://www.mdpi.com/2078-2489/11/4/193?utm_source=rss&utm_medium=rss

17. Overview of E-Learning Platforms for Teaching and Learning. https://www.researchgate.net/publication/350328024_Overview_of_E-Learning_Platforms_for_Teaching_and_Learning

18. Jurafsky, D., & Martin, J. H. (2021). Speech and Language Processing (3rd ed.). Draft. Stanford University.

19. Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is All You Need. Advances in Neural Information Processing Systems (NeurIPS).

20. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT.

21. Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. ACM Computing Surveys (CSUR), 34(1), 1–47.

22. Liu, Y., Ott, M., Goyal, N., et al. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint arXiv:1907.11692.

23. Doshi-Velez, F., & Kim, B. (2017). Towards A Rigorous Science of Interpretable Machine Learning. arXiv preprint arXiv:1702.08608.

24. React.js Documentation. URL: https://react.dev/

25. Django Documentation. URL: https://docs.djangoproject.com/

26. Python Documentation. URL: https://docs.python.org/

27. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/

28. Hugging Face Transformers. RoBERTa. URL: https://huggingface.co/docs/transformers/model_doc/roberta

29. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. URL: https://www.deeplearningbook.org/

30. Russell, S. J., & Norvig, P. (2010). Artificial Intelligence: A Modern Approach (3rd ed.). Prentice Hall.

ДОДАТОК А

```python
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch
import torch.nn.functional as F

class GPT2OutputDetector:
    def __init__(self):
        self.tokenizer = AutoTokenizer.from_pretrained("roberta-base-openai-detector")
        self.model = AutoModelForSequenceClassification.from_pretrained("roberta-base-openai-detector")
        self.device = 'cuda' if torch.cuda.is_available() else 'cpu'
        self.model.to(self.device)

    def predict(self, text):
        inputs = self.tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length=512).to(self.device)
        with torch.no_grad():
            outputs = self.model(**inputs)
            logits = outputs.logits
            probs = F.softmax(logits, dim=1).squeeze().tolist()
        return {
            "human_prob": round(probs[0]*100, 2),
            "ai_prob": round(probs[1]*100, 2),
        }
```

```python
import tkinter as tk
from tkinter import filedialog
from detector import GPT2OutputDetector

import sys
import os
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))

detector = GPT2OutputDetector()

def analyze_file():
    filepath = filedialog.askopenfilename(filetypes=[("Text files", "*.txt")])
    if not filepath:
        return

    with open(filepath, 'r', encoding='utf-8') as file:
        text = file.read()

    result = detector.predict(text)

    result_str = f"Human Probability: {result['human_prob']}%\nAI Probability: {result['ai_prob']}%"

    output_text.delete("1.0", tk.END)
    output_text.insert(tk.END, result_str)

    with open("result.txt", "w", encoding='utf-8') as result_file:
        result_file.write(result_str)

window = tk.Tk()
window.title("AI Output Detector")
window.geometry("400x300")

frame = tk.Frame(window)
frame.pack(pady=20)

choose_button = tk.Button(frame, text="Choose File", command=analyze_file)
choose_button.pack()

output_text = tk.Text(window, height=10, wrap="word")
output_text.pack(padx=10, pady=10)

window.mainloop()
```

Рисунок А.1 – Код детектора та простого інтерфейсу

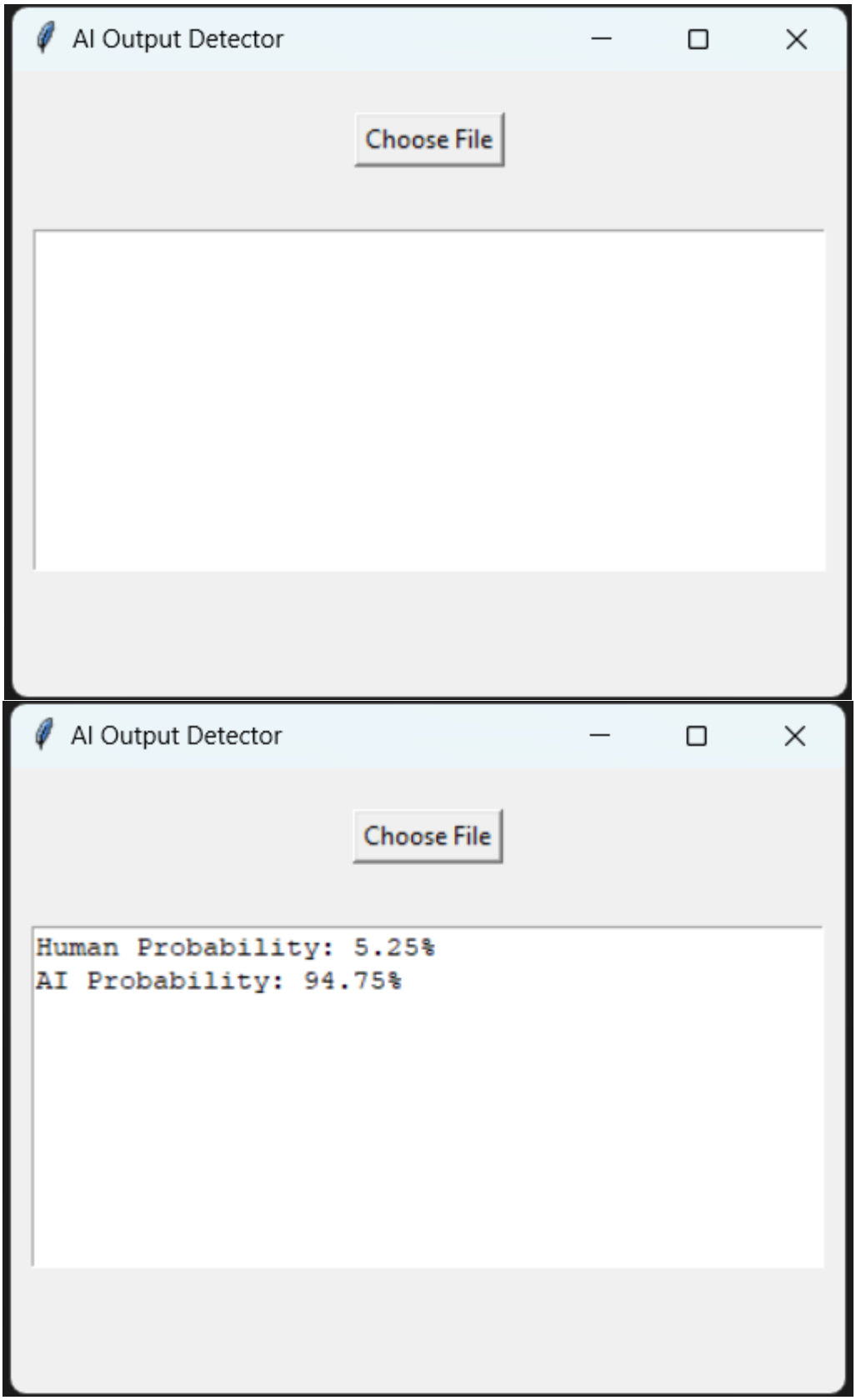


Рисунок А.2 – Вікно завантаженню файлу та результат оцінки

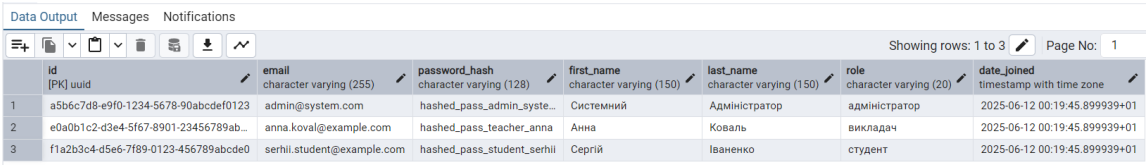
ДОДАТОК Б

```sql
INSERT INTO users (id, email, password_hash, first_name, last_name, role)
VALUES (
    'e0a0b1c2-d3e4-5f67-8901-23456789abcd'::uuid,
    'anna.koval@example.com',
    'hashed_pass_teacher_anna',
    'Анна',
    'Коваль',
    'викладач'
);

-- Додаємо студента
INSERT INTO users (id, email, password_hash, first_name, last_name, role)
VALUES (
    'f1a2b3c4-d5e6-7f89-0123-456789abcde0'::uuid,
    'serhii.student@example.com',
    'hashed_pass_student_serhii',
    'Сергій',
    'Іваненко',
    'студент'
);

-- Додаємо адміністратора
INSERT INTO users (id, email, password_hash, first_name, last_name, role)
VALUES (
    'a5b6c7d8-e9f0-1234-5678-90abcdef0123'::uuid,
    'admin@system.com',
    'hashed_pass_admin_system',
    'Системний',
    'Адміністратор',
    'адміністратор'
);
```

Рисунок Б.1 – Додавання користувачів в БД



Data Output | Messages | Notifications

Showing rows: 1 to 3 | Page No: 1

	id [PK] uuid	email character varying (255)	password_hash character varying (128)	first_name character varying (150)	last_name character varying (150)	role character varying (20)	date_joined timestamp with time zone
1	a5b6c7d8-e9f0-1234-5678-90abcdef0123	admin@system.com	hashed_pass_admin_syste...	Системний	Адміністратор	адміністратор	2025-06-12 00:19:45.899939+01
2	e0a0b1c2-d3e4-5f67-8901-23456789ab...	anna.koval@example.com	hashed_pass_teacher_anna	Анна	Коваль	викладач	2025-06-12 00:19:45.899939+01
3	f1a2b3c4-d5e6-7f89-0123-456789abcde0	serhii.student@example.com	hashed_pass_student_serhii	Сергій	Іваненко	студент	2025-06-12 00:19:45.899939+01

Рисунок Б.2 – Результат додавання користувачів в БД

ДОДАТОК В

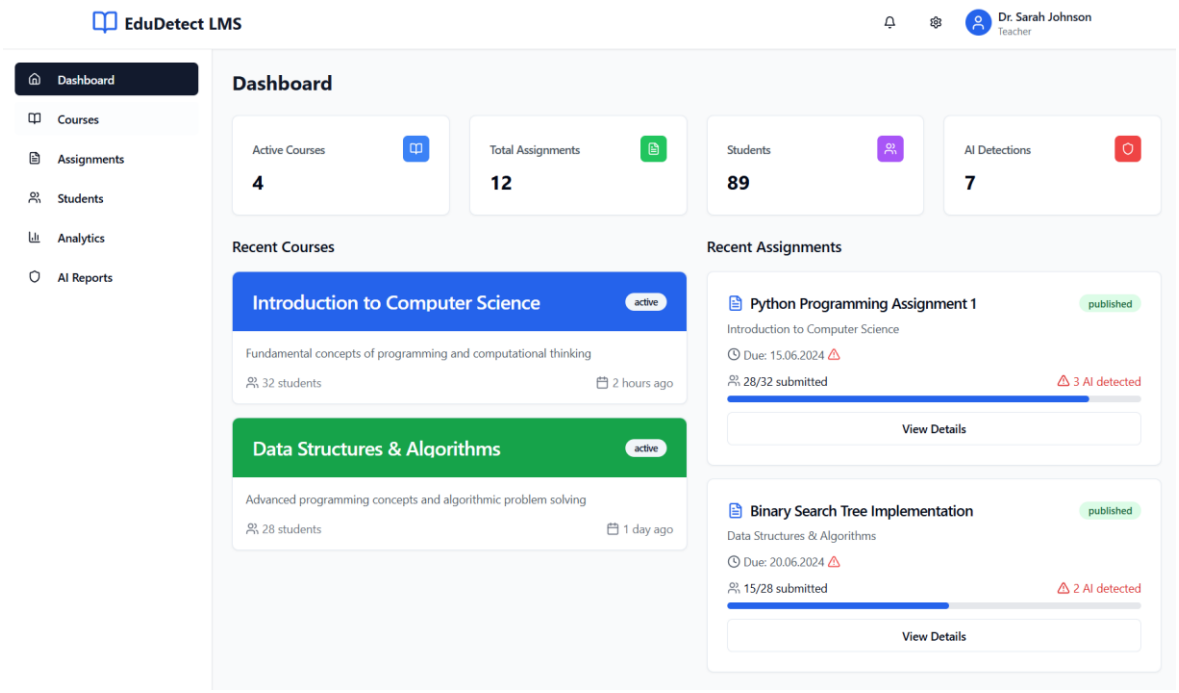


Рисунок В.1 – Головна сторінка для викладача

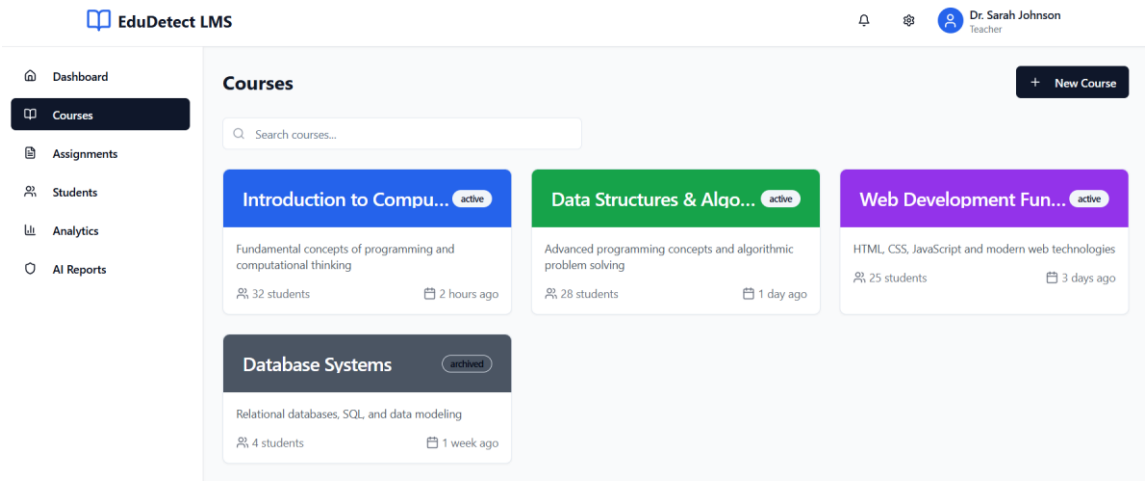


Рисунок В.2 – Сторінка курсів

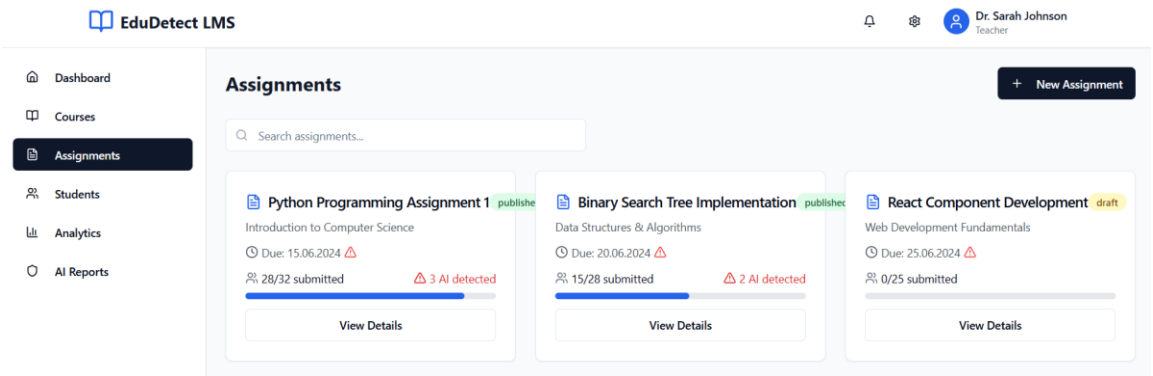


Рисунок В.3 – Сторінка завдань

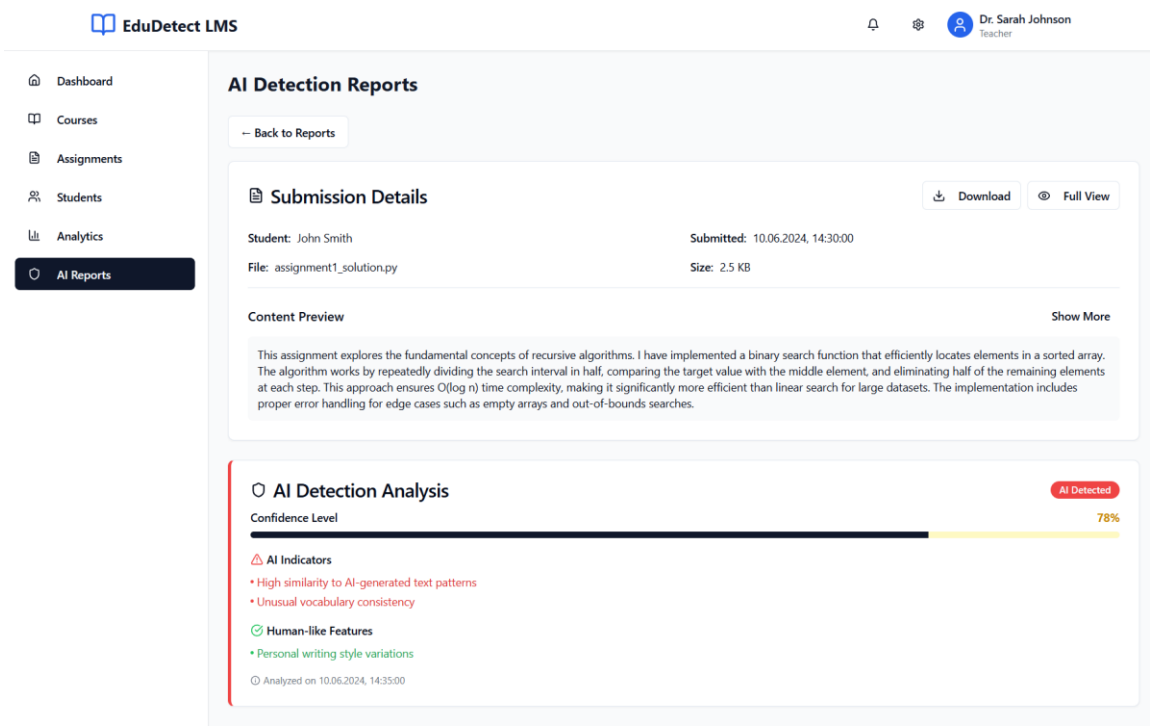


Рисунок В.4 – Сторінка звіту по детекції ШІ до одного із завдань

ДОДАТОК Г

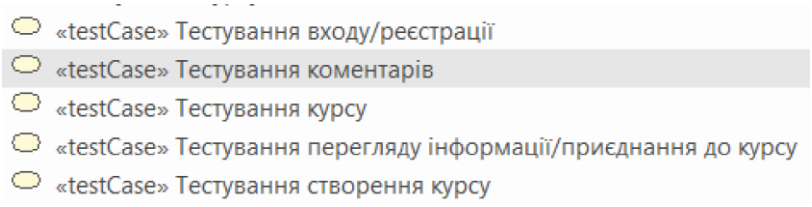


Рисунок Г.1 – Список усіх створених тест-кейсів

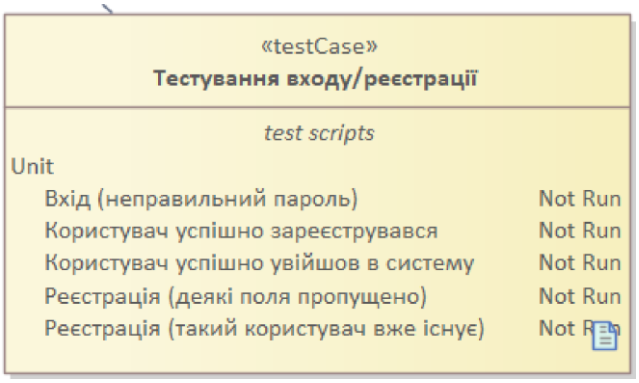
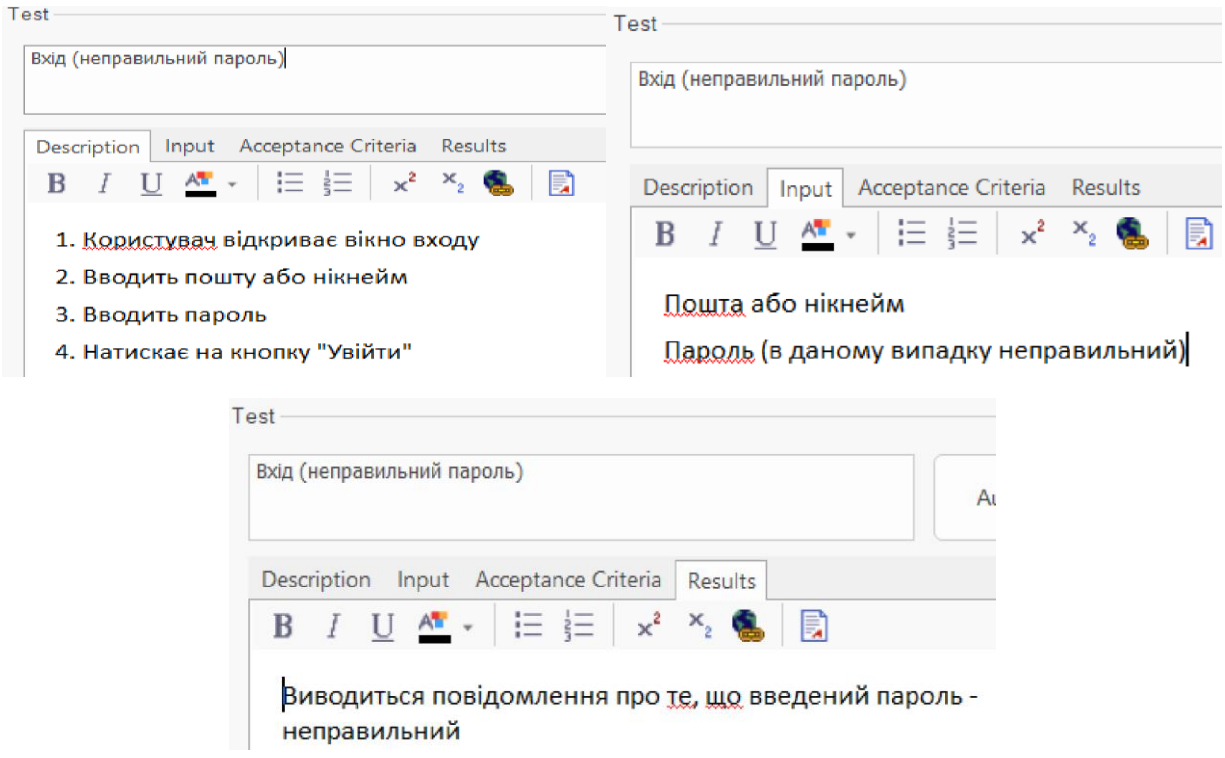
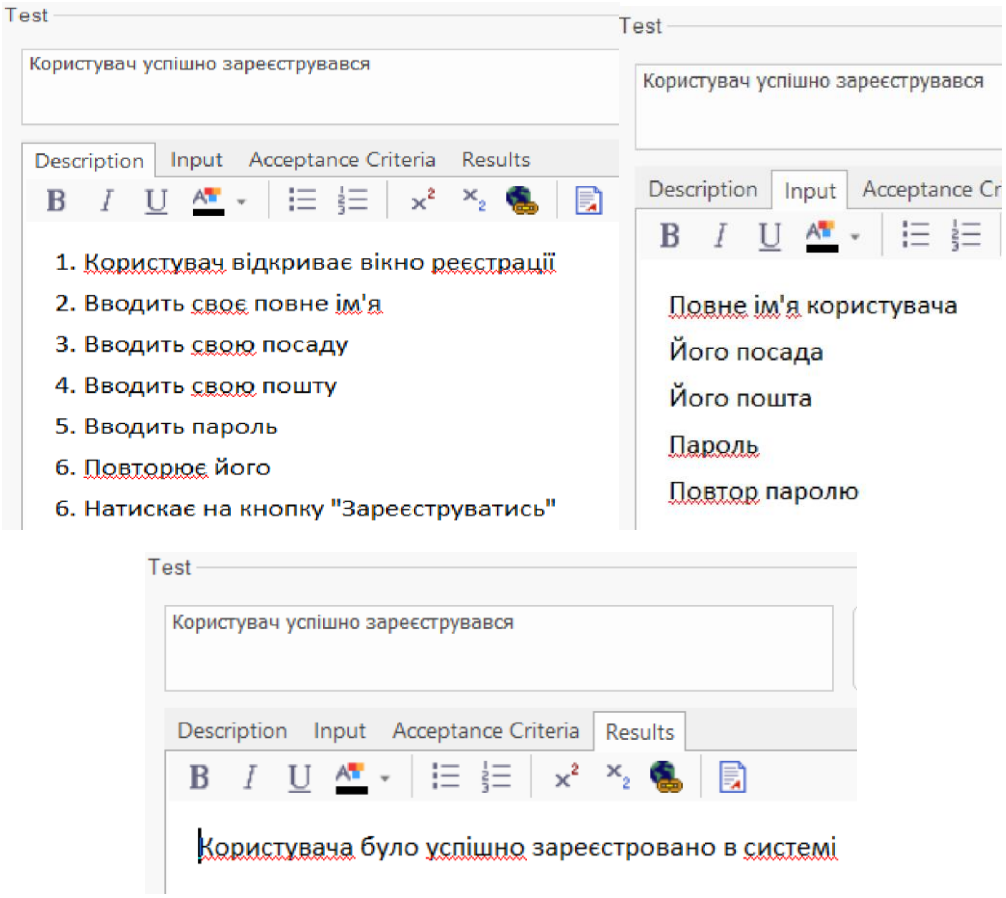


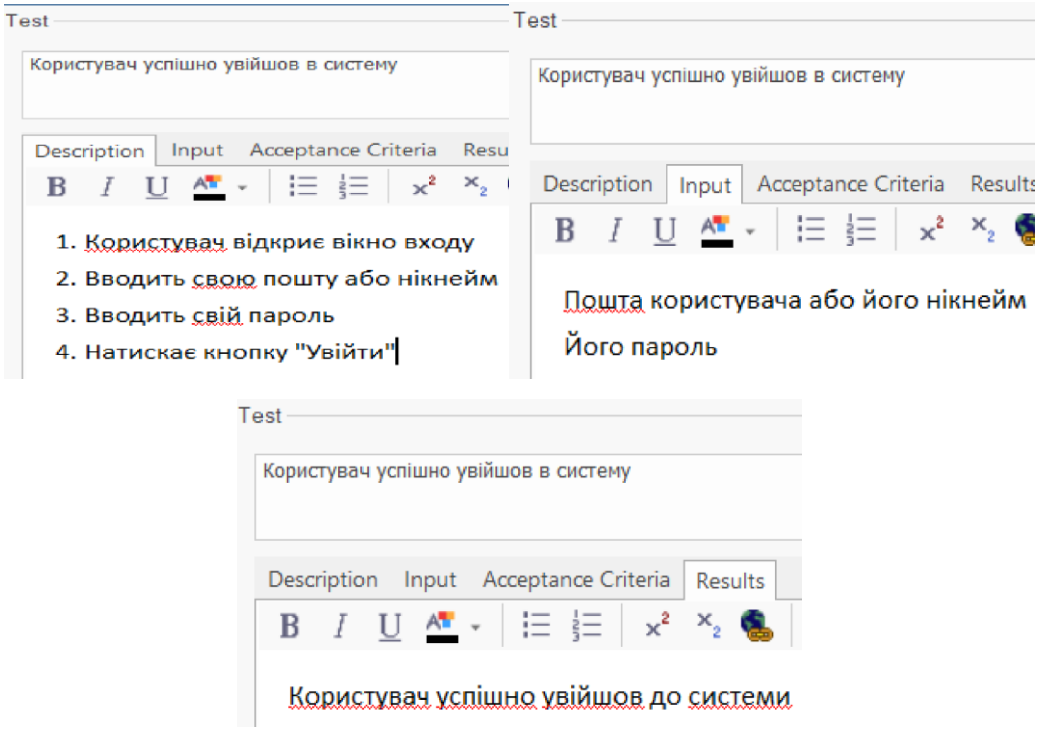
Рисунок Г.2 – Тест кейс «Тестування авторизації»



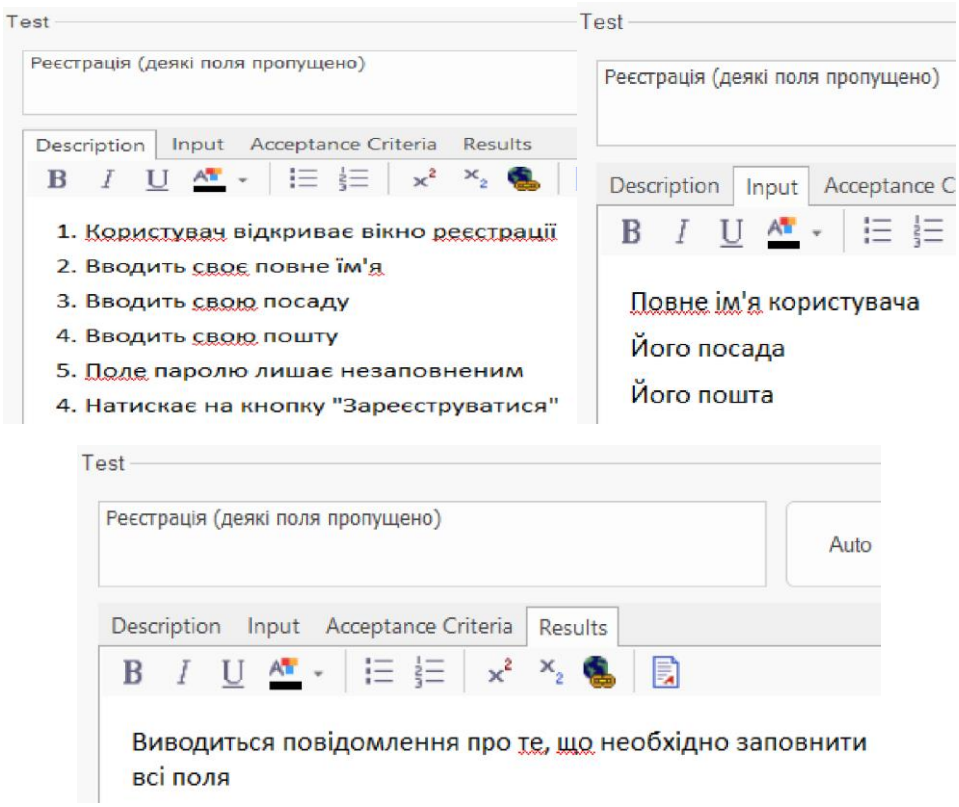
а)


Test
Користувач успішно зареєструвався
Description
Input
Acceptance Criteria
Results
1. Користувач відкриває вікно реєстрації
2. Вводить своє повне ім'я
3. Вводить свою посаду
4. Вводить свою пошту
5. Вводить пароль
6. Повторює його
6. Натискає на кнопку "Зареєструватись"
Повне ім'я користувача
Його посада
Його пошта
Пароль
Повтор паролю
Користувача було успішно зареєстровано в системі

б)


Test
Користувач успішно увійшов в систему
Description
Input
Acceptance Criteria
Results
1. Користувач відкриє вікно входу
2. Вводить свою пошту або нікнейм
3. Вводить свій пароль
4. Натискає кнопку "Увійти"
Пошта користувача або його нікнейм
Його пароль
Користувач успішно увійшов до системи

в)



г)

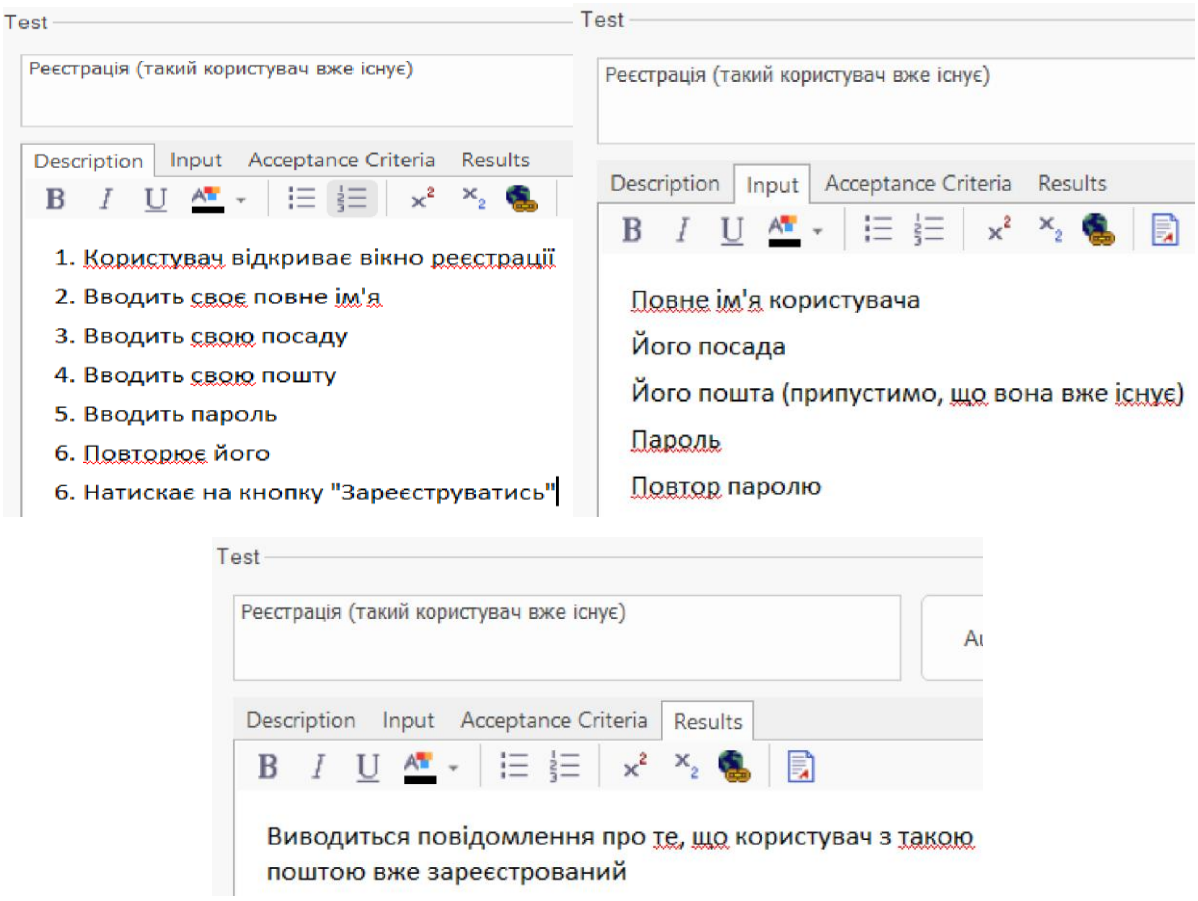


Рисунок Г.3 – Тестові сценарії тест кейсу «Тестування входу/реєстрації»

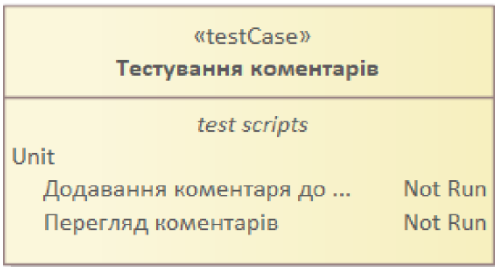
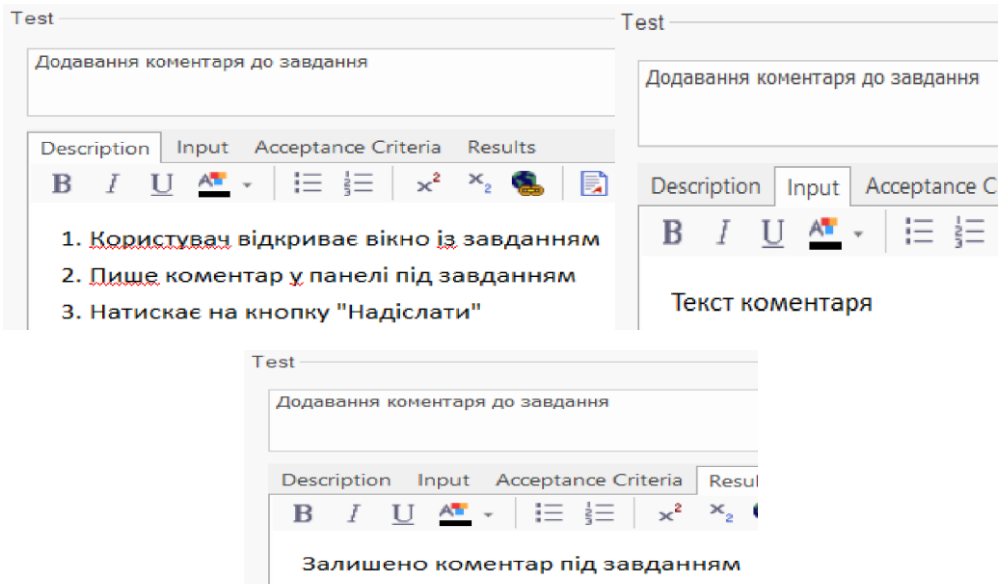
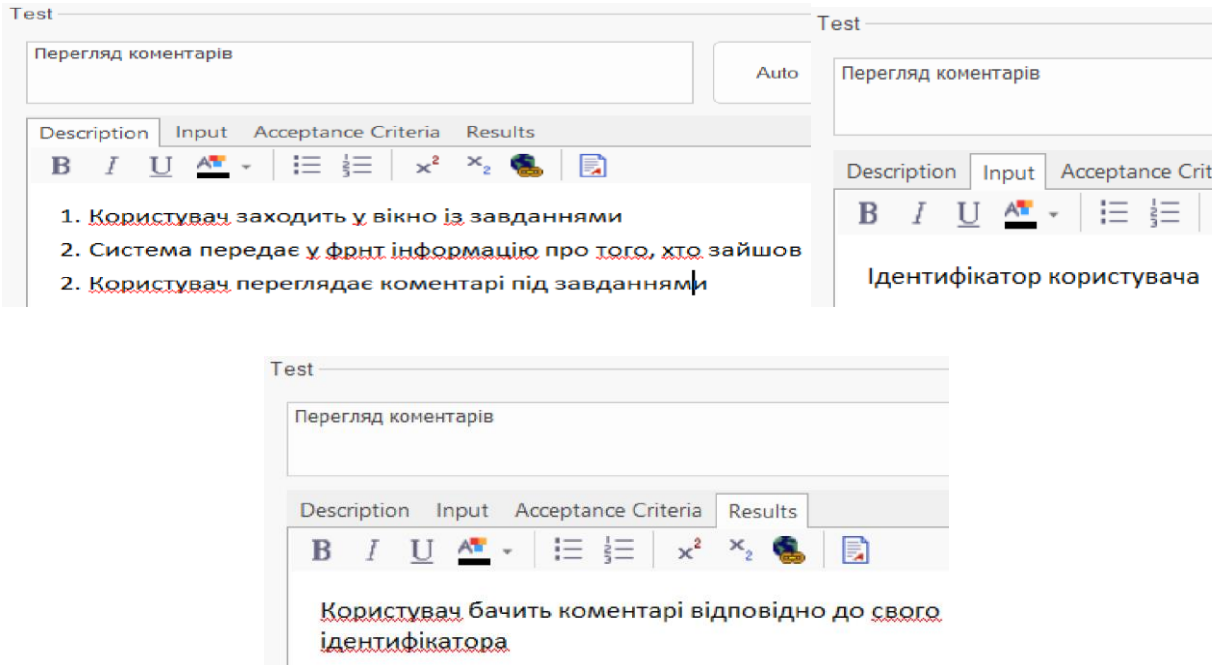


Рисунок Г.4 – Тест кейс «Тестування коментарів»



а)



б)

Рисунок Г.5 – Тестові сценарії тест кейсу «Тестування коментарів»

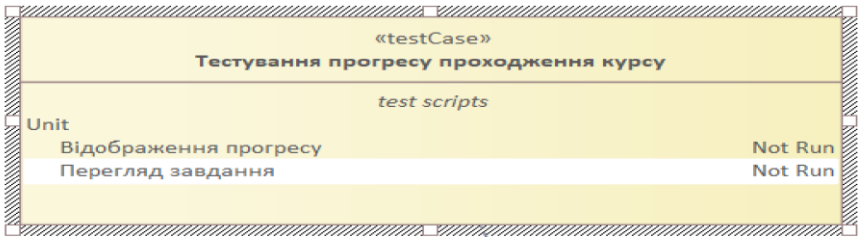
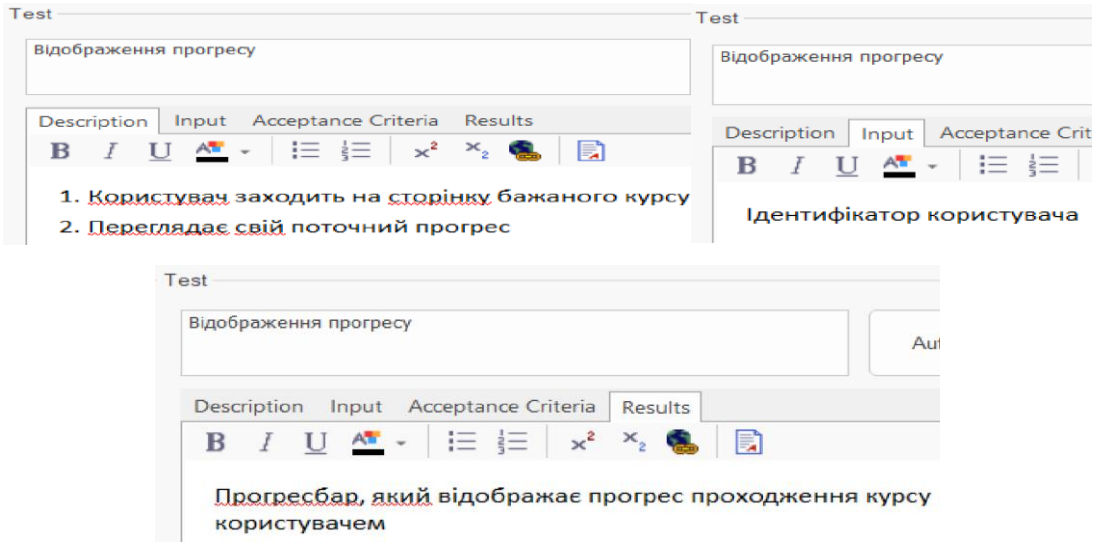
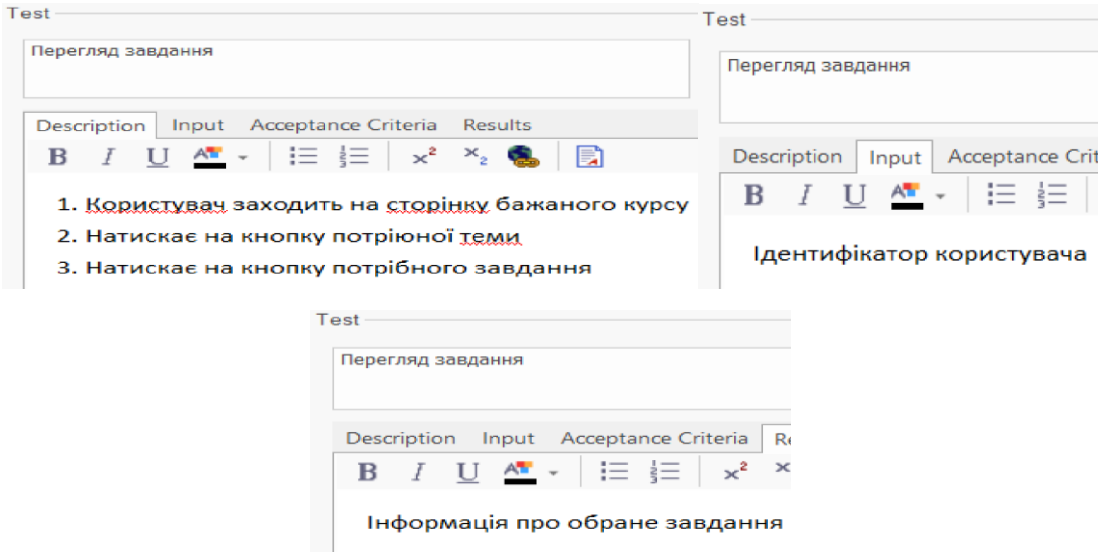


Рисунок Г.6 – Тест кейс «Тестування прогресу проходження курсу»



а)



б)

Рисунок Г.7 – Тестові сценарії «Тестування прогресу проходження курсу»

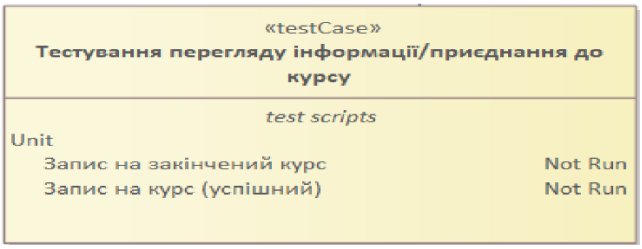
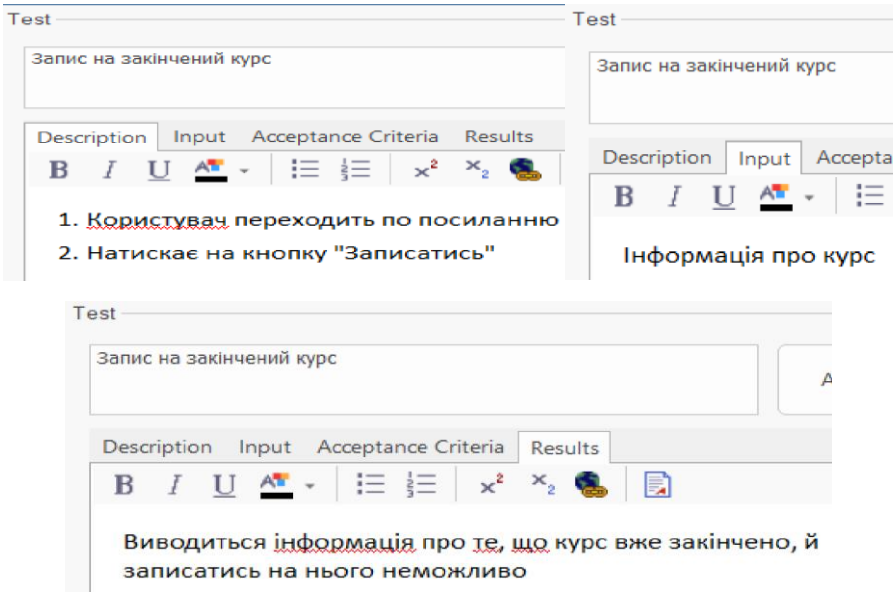
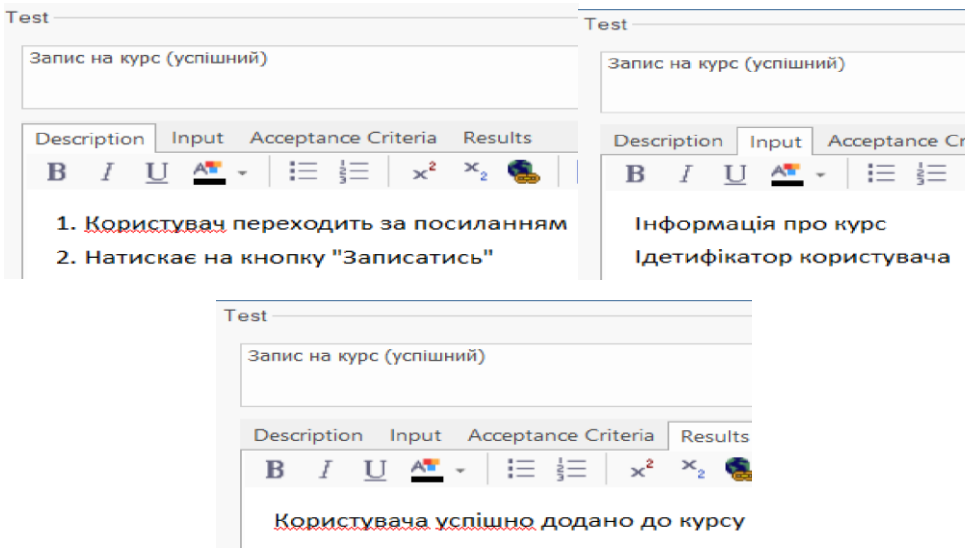


Рисунок Г.8 – Тест «Тестування перегляду інформації/приєднання до курсу»



а)



б)

Рисунок Г.9 – «Тестування перегляду інформації/приєднання до курсу»

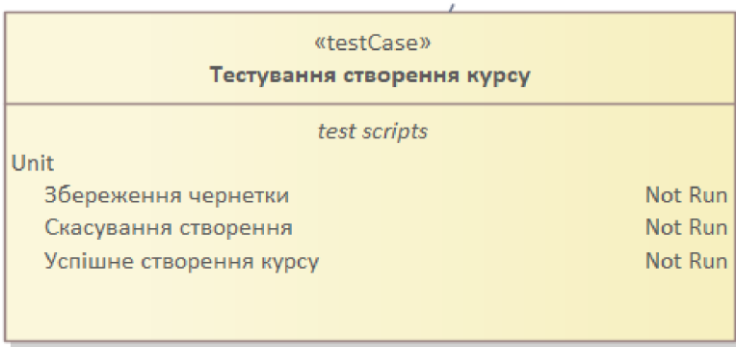
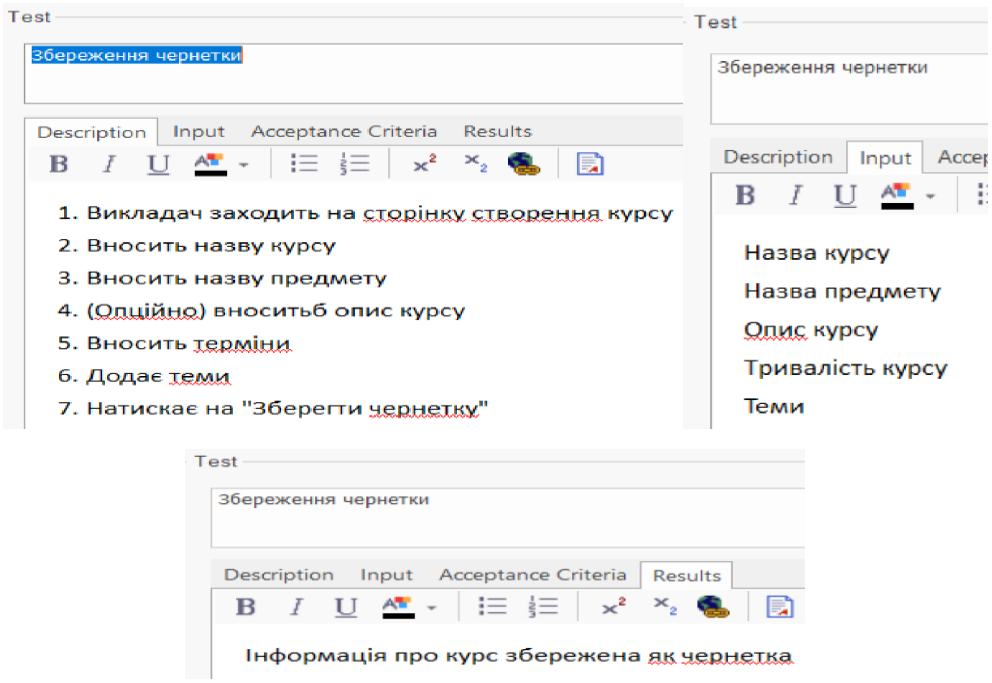
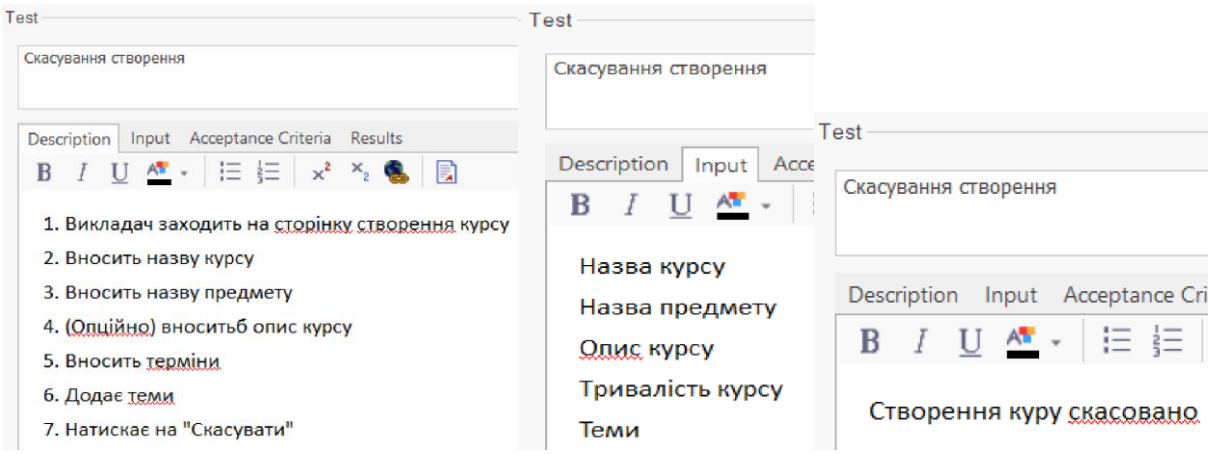


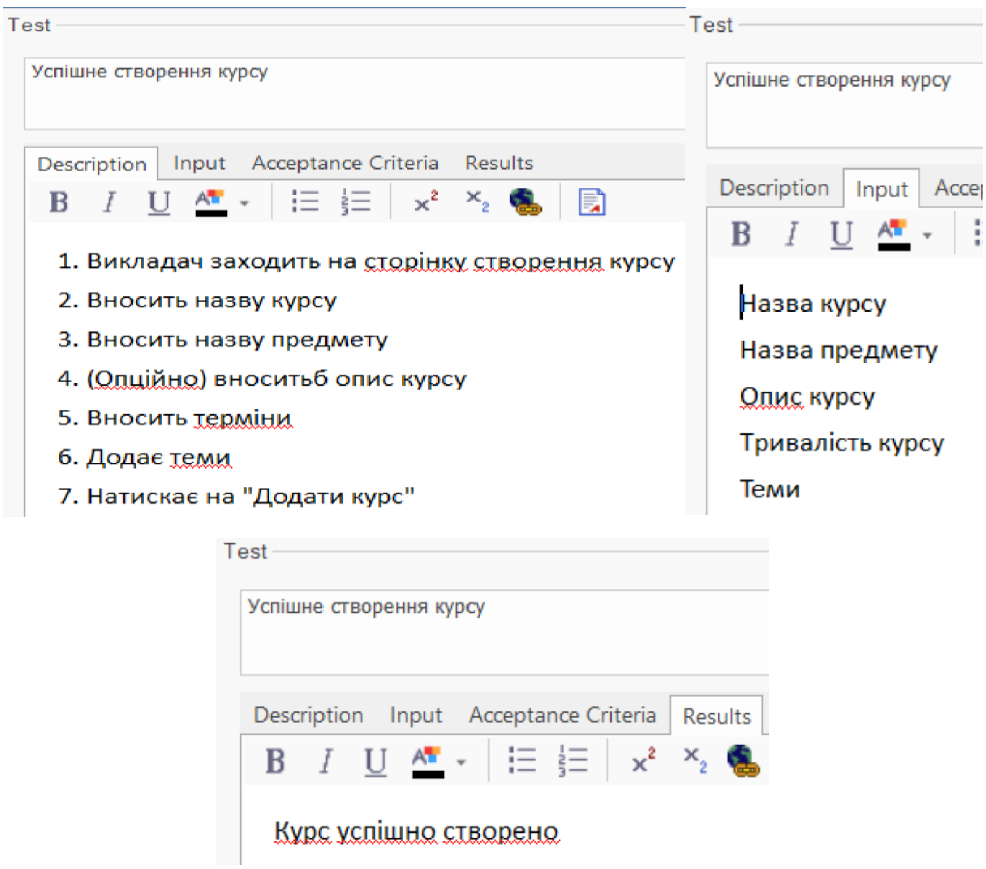
Рисунок Г.10 – Тест кейс «Тестування створення курсу»



а)



б)



в)

Рисунок Г.11 – Тестові сценарії тест кейсу «Тестування створення курсу»

ДОДАТОК Д – КОПІЯ ТЕЗ ДОПОВІДІ

Уманський Д.Б., студент 4-го курсу
Київський національний економічний університет імені Вадима Гетьмана
udmitro87@gmail.com

ТРАНСФОРМЕРИ ЯК ІНСТРУМЕНТ КЛАСИФІКАЦІЇ ТЕКСТУ В NLP

Однією з ключових задач у сфері обробки природної мови (Natural Language Processing, NLP) є класифікація тексту — процес віднесення текстових фрагментів до певних категорій. Зі стрімким розвитком штучного інтелекту та появою великих мовних моделей особливу увагу привертають трансформерні архітектури, які забезпечують нову якість обробки текстових даних.

Машинне навчання — підгалузь штучного інтелекту, яка базується на побудові алгоритмів, здатних навчатися на основі даних. У контексті NLP це дає змогу автоматизувати такі задачі, як класифікація тексту, аналіз тональності, машинний переклад, пошук інформації та інше (Jurafsky & Martin, 2021) [1].

Архітектура Transformer, запропонована у 2017 році дослідниками Google, замінила традиційні рекурентні нейронні мережі (RNN) завдяки здатності обробляти довгі залежності у тексті за допомогою механізму self-attention (Vaswani et al., 2017) [2]. Це дозволило досягти кращої якості розуміння тексту при меншому обчислювальному навантаженні у порівнянні з попередніми моделями.

BERT (Bidirectional Encoder Representations from Transformers) — це передтренована трансформерна модель, яка читає текст одночасно зліва направо та справа наліво. Такий підхід (двонаправлене навчання) забезпечує глибше розуміння контексту кожного слова у реченні. На відміну від попередніх моделей, BERT навчається не просто передбачати наступне слово, а реконструювати пропущені токени у реченнях та визначати, чи пов’язані речення між собою логічно (Devlin et al., 2019) [3].

У задачі класифікації тексту трансформери дозволяють ефективно кодувати всю текстову інформацію у векторні представлення. До виходу трансформерів поширеними були методи з використанням TF-IDF (term frequency–inverse document frequency), Naive Bayes, SVM (support vector machines), однак вони не враховували семантичні зв’язки на глибокому рівні (Sebastiani, 2002) [4].

Трансформери працюють шляхом перетворення вхідного тексту у векторне представлення (embedding), яке містить інформацію про значення слів та їхні контекстні зв’язки. Ці вектори обробляються через кілька шарів механізмів самоуваги (self-attention) та нормалізації. У підсумку модель формує контекстуалізоване подання кожного токена, а згодом і всього тексту загалом (Vaswani et al., 2017) [2].

У випадку класифікації тексту, до трансформера додається голова класифікації (classification head) — зазвичай це один або кілька повнозв’язних (fully connected) шарів, які приймають на вхід вектор — спеціальний токен, що представляє увесь текст. На цьому етапі застосовується softmax, що дає

ймовірність належності тексту до кожної з можливих категорій.

1. Токенізація — вхідний текст перетворюється у послідовність токенів за допомогою спеціального токенізатора (наприклад, WordPiece).
2. Перетворення в тензори — токени переводяться у числові ідентифікатори та доповнюються масками (attention masks).
3. Подача до трансформера — модель обробляє вхід і формує вектор [CLS]. Класифікаційна голова — на вектор [CLS] подається класифікаційна нейромережа.
4. Функція втрат і оптимізація — для навчання використовується функція втрат, наприклад CrossEntropyLoss.

Оскільки моделі типу BERT уже попередньо навчені на великих корпусах (Wikipedia, BooksCorpus), для конкретної задачі (наприклад, класифікації відгуків, новин або електронних листів) потрібно провести лише до-навчання (fine-tuning). Це дає змогу значно скоротити час і обчислювальні ресурси порівняно з повним навчанням моделі з нуля (Devlin et al., 2019) [3].

Fine-tuning виконується таким чином:

1. Ініціалізується попередньо навчена модель.
2. До неї додається класифікаційна голова.
3. Вся модель (або лише верхні шари) донавчається на цільовому датасеті протягом кількох епох.

Переваги трансформерів:

- глобальна увага до контексту;
- висока масштабованість;
- можливість донавчання під специфічні задачі (fine-tuning).

Моделі типу BERT, RoBERTa, ALBERT та DistilBERT показали себе як надзвичайно ефективні інструменти у вирішенні широкого спектра NLP-задач (Liu et al., 2019) [5].

Наразі активно досліджуються напрямки:

- удосконалення інтерпретованості моделей трансформерного типу (Doshi-Velez & Kim, 2017) [6];
- оптимізація моделей для мобільних та вбудованих пристроїв;
- комбінування трансформерів з іншими видами штучного інтелекту, зокрема когнітивними та нейро-нечіткими технологіями.

Трансформерні архітектури, зокрема BERT, є одним із найперспективніших інструментів для реалізації задач класифікації тексту. Їх використання дозволяє досягати високих результатів навіть у складних і неоднозначних випадках. Подальший розвиток цієї галузі сприятиме зростанню якості автоматизованої обробки текстової інформації у науці, освіті, економіці та інших сферах.

Список використаних джерел

1. Jurafsky, D., & Martin, J. H. (2021). Speech and Language Processing (3rd ed.). Draft. Stanford University.

2. Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is All You Need. Advances in Neural Information Processing Systems (NeurIPS).

3. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-

training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT.

4. Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. ACM Computing Surveys (CSUR), 34(1), 1–47.

5. Liu, Y., Ott, M., Goyal, N., et al. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint arXiv:1907.11692.

6. Doshi-Velez, F., & Kim, B. (2017). Towards A Rigorous Science of Interpretable Machine Learning. arXiv preprint arXiv:1702.08608.

Науковий керівник: Помазун О.М., к.е.н., доцент.

ДОДАТОК Е – ЗВІТ ПОДІБНОСТІ

StrikePlagiarism.com

Дата звіту 6/13/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kyiv National Economic University named after Vadym Hetman KNEU

Заголовок
РОЗРОБКА НАВЧАЛЬНОЇ ПЛАТФОРМИ З ВИКОРИСТАННЯМ НОВІТНІХ ТЕХНОЛОГІЙ МАШИННОГО НАВЧАННЯ ДЛЯ ПЕРЕВІРКИ ЗНАНЬ СТУДЕНТІВ

Автор **Уманський Дмитро Борисович**
Науковий керівник / Експерт **Лозовик Ю.М.**

підрозділ
кафедра інформаційних систем в економіці

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

5.21% 5.21% КП 1

1.06% 1.06% КП 2

0.66% 0.66% КЦ

25
Довжина фрази для коефіцієнта подібності 2

15971
Кількість слів

126548
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	1
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	¶	0
Парафрази (SmartMarks)	a	48

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://ir.kneu.edu.ua/bitstreams/a5c7419f-c089-4b0c-8e70-3923cf3566a2/download	43 0.27 %
2	https://ir.kneu.edu.ua/bitstreams/22acc4f6-0461-4d18-a9a4-ff2fbd8c10ff/download	37 0.23 %
3	https://ir.kneu.edu.ua/bitstreams/22acc4f6-0461-4d18-a9a4-ff2fbd8c10ff/download	34 0.21 %
4	https://ir.kneu.edu.ua/bitstreams/b20c497c-320d-46ce-8d1d-a56f3c2eec3f/download	31 0.19 %

Рисунок Е.1 – Короткий звіт подібності