Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Програмно-апаратний комплекс для вивчення української мови з використанням штучного інтелекту

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав: студент 2-го курсу, групи 1КІ-24м
спеціальності 123 — Комп'ютерна інженерія
_____ Тетерев В. І.

Керівник: к.т.н., проф. каф. ОТ
_____ Азарова А. О.

«12» грудня 2025 р.

Опонент: к.т.н., доцент каф. МБІС
_____ Карпінець В. В.

«12» грудня 2025 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.

«18» грудня 2025 р.

ВНТУ 2025

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Освітній рівень — магістр

Напрям підготовки — 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри
обчислювальної техніки
_____проф., д.т.н. О. Д. Азаров
«_25_» __вересня__ 2025 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту **Тетереву Віталію Ігоровичу**

1 Тема роботи: «Програмно-апаратний комплекс для вивчення української мови з використанням штучного інтелекту» керівник роботи: Азарова Анжеліка Олексіївна к.т.н., проф., затверджено наказом вищого навчального закладу від 24.09.2025 р. № 313.

2 Строк подання студентом роботи — 4 грудня 2025 року.

3 Вихідні дані до роботи: методи розробки програмно-апаратних комплексів, навчальні матеріали на тему створення програмно апаратних-комплексів, технічна документація мови програмування Python, аналоги.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналітичний огляд програмних та апаратних засобів, сучасних мовних моделей і технологій штучного інтелекту, моделювання та проєктування програмно-апаратного комплексу, практична реалізація програмно-апаратного комплексу для вивчення української мови, тестування програмно-апаратного комплексу, економічна частина.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): структурна схема програмно-апаратного комплексу.

6 Консультанти розділів роботи наведені у таблиці 1.

Таблиця 1 — Консультанти розділів МКР

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Азарова А. О. к.т.н., проф. каф. ОТ		
5	Ратушняк О. Г. к.т.н., доц. каф. ЕПВМ		
Нормоконтроль	Швець С.І., асист. каф. ОТ		

7 Дата видачі завдання 25.09.2025

8 Календарний план розділів роботи наведено у таблиці 2.

Таблиця 2 — Календарний план МКР

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи	24.09.2025	Виконано
2	Аналіз методів оброблення мовлення та мовних технологій ШІ	25.09.2025	Виконано
3	Проєктування архітектури комплексу та його функціональних модулів	6.10.2025	Виконано
4	Вибір технологій розроблення та підготовка технічних рішень	19.10.2025	Виконано
5	Реалізація модулів комплексу та інтеграція системи	28.10.2025	Виконано
6	Підготовка матеріалів для опису розробленого комплексу	9.11.2025	Виконано
7	Оформлення пояснювальної записки	23.11.2025	Виконано
8	Перевірка якості виконання магістерської роботи	11.12.2025	Виконано

Студент _____ Тетерев Віталій Ігорович

Керівник _____ к.т.н., проф. каф. ОТ Азарова Анжеліка Олексіївна

АНОТАЦІЯ

УДК 004.896

Тетерев В. І. Програмно-апаратний комплекс для вивчення української мови з використанням штучного інтелекту. Магістерська кваліфікаційна робота зі спеціальності 123 — комп'ютерна інженерія, освітня програма — комп'ютерна інженерія. Вінниця: ВНТУ, 2025. 130 с.

На укр. мові. Бібліогр.: 38 назв; рис.: 29; табл.: 9.

У магістерській кваліфікаційній роботі розглянуто процес розроблення програмно-апаратного комплексу для вивчення української мови із застосуванням технологій штучного інтелекту. Проаналізовано сучасні підходи до створення інтерактивних освітніх систем та існуючі програмно-апаратні рішення, виявлено їх основні недоліки. Розроблено архітектуру програмно-апаратного комплексу, що поєднує апаратну платформу та інтелектуальні програмні модулі, орієнтовані на підвищення адаптивності й ефективності навчального процесу.

Основним напрямом дослідження є проєктування та реалізація модулів розпізнавання мовлення, граматичного аналізу, генерації навчальних завдань і синтезу мовлення, що забезпечують персоналізоване та адаптивне навчання української мови. Для реалізації використано бібліотеки Vosk API, pyttsx3, Coqui TTS та мовну модель Gemini 2.0 Flash.

Запропоновано гнучку та масштабовану структуру комплексу, проведено його тестування, а також оцінено економічну ефективність. Комплекс орієнтований на використання у мовних курсах, закладах освіти та для самостійного навчання, з можливістю адаптації до інших мов.

Ключові слова: штучний інтелект, українська мова, програмно-апаратний комплекс, розпізнавання мовлення, генерація завдань, Vosk API, Gemini 2.0 Flash.

ABSTRACT

Teterev V. I. Software and Hardware Complex for Learning the Ukrainian Language Using Artificial Intelligence. Comprehensive Master's Qualification Thesis in specialty 123 — Computer Engineering, educational program — Computer Engineering. Vinnytsia: VNTU, 2025. 130 p.

In Ukrainian. Bibliogr.: 38 titles; ill.: 29; tables: 9.

The master’s qualification thesis examines the development of a software and hardware complex for learning the Ukrainian language using artificial intelligence technologies. Modern approaches to the design of interactive educational systems are analyzed, existing solutions and their limitations are identified, and a custom architecture integrating hardware platforms and intelligent software modules is developed.

The main focus of the study is the design and implementation of modules for speech recognition, grammatical analysis, task generation, and speech synthesis, enabling personalized and adaptive learning of the Ukrainian language. The system employs tools such as Vosk API, pyttsx3, Coqui TTS, and the Gemini 2.0 Flash language model.

A flexible and scalable structure of the complex is proposed, tested for functionality, and assessed for economic feasibility. The system is intended for use in language courses, educational institutions, and self-study, with the possibility of adaptation to other languages.

Keywords: artificial intelligence, Ukrainian language, software and hardware complex, speech recognition, task generation, Vosk API, Gemini 2.0 Flash.

ЗМІСТ

ВСТУП

Актуальність теми полягає у тому, що у сучасному світі стрімкий розвиток штучного інтелекту значно впливає на всі сфери людської діяльності, зокрема освіту. Існуючі програмні навчальні комплекси дозволяють автоматизувати певні аспекти навчального процесу, проте більшість із них мають обмеження, пов’язані з відсутністю адаптивності до індивідуальних особливостей користувача, низьким рівнем інтерактивності та недостатнім використанням сучасних інтелектуальних технологій. Такі системи переважно орієнтовані на статичну подачу матеріалу, не аналізують мовлення користувача, не здійснюють зворотного зв’язку у режимі реального часу та не забезпечують гнучкої адаптації навчального процесу. Отже, дослідження в напрямку розроблення програмно-апаратного комплексу, який поєднує сучасні технології штучного інтелекту, засоби мовної аналітики та апаратні компоненти, є актуальним. Такий підхід дозволяє усунути недоліки наявних рішень, забезпечити персоналізоване навчання, підвищити ефективність засвоєння матеріалу та розширити можливості практичного використання української мови.

Зв’язок роботи з науковими програмами, планами, темами простежується у тому, що робота виконувалася відповідно до плану науково-дослідних робіт кафедри обчислювальної техніки у межах напряму «Інтелектуальні системи підтримки навчання та автоматизації освітніх процесів». Дослідження спрямоване на розвиток технологій, що поєднують програмні та апаратні компоненти для ефективного навчання мовам, та має тісний зв’язок із сучасними науковими програмами в галузі штучного інтелекту та освітніх технологій.

Метою магістерської кваліфікаційної роботи є удосконалення процесу вивчення української мови (з урахуванням специфіки іноземної аудиторії) шляхом розроблення програмно-апаратного комплексу, який забезпечує адаптивність, інтерактивність та персоналізацію навчання на основі

технологій штучного інтелекту.

Для досягнення мети необхідно розв’язати такі **завдання**:

— проаналізувати сучасні програмно-апаратні рішення, що використовуються в освітніх цілях;

— дослідити можливості застосування технологій штучного інтелекту під час вивчання мови;

— розробити архітектуру комплексу, що поєднує програмні та апаратні компоненти;

— реалізувати модулі розпізнавання мовлення, аналізу тексту, генерації навчальних матеріалів та синтезу мовлення;

— забезпечити інтеграцію мовних моделей у процес навчання;

— провести тестування та оцінити ефективність створеного комплексу;

— довести економічну доцільність розробленого програмно-апаратного комплексу.

Об’єкт дослідження — процес побудови програмно-апаратного комплексу з елементами штучного інтелекту для освітніх потреб.

Предмет дослідження — методи, моделі та технології створення інтерактивної навчальної системи з вивчення української мови з використанням AI.

Методи дослідження у процесі виконання магістерської кваліфікаційної роботи охоплювали застосування системного аналізу для дослідження предметної області та формування вимог до програмно-апаратного комплексу, методів структурного і функціонального моделювання для проєктування архітектури системи та взаємодії її модулів, методів проєктування програмних систем для реалізації модульної та масштабованої програмної складової, а також методів машинного навчання й оброблення природної мови для аналізу мовлення користувача, перевірки мовних конструкцій і генерації адаптивних навчальних завдань; додатково застосовувалися методи цифрового оброблення сигналів для роботи з

аудіоданими, методи інтеграції програмного та апаратного забезпечення для забезпечення взаємодії апаратних компонентів із програмними модулями та методи експериментального тестування й аналізу ефективності для оцінювання коректності роботи системи та результативності навчального процесу.

Новизною одержаних результатів є те, що було удосконалено підхід до побудови інтерактивного середовища для вивчення української мови на основі технологій штучного інтелекту, який, на відміну від існуючих підходів, забезпечує інтеграцію мовних моделей, модулів розпізнавання і синтезу мовлення, а також інтелектуальних механізмів адаптації навчального процесу до індивідуальних особливостей користувача, а також забезпечує інтелектуальне оброблення даних у реальному часі, що дозволяє підвищити ефективність засвоєння української мови.

Практична цінність роботи полягає у створенні дієздатного прототипу програмно-апаратного комплексу, який може бути використаний як основа для розроблення інтерактивних систем навчання української мови у закладах освіти з урахуванням іноземної аудиторії, мовних курсах або для самостійного навчання. Запропоновані підходи можуть бути адаптовані до інших мов та навчальних дисциплін. Реалізований комплекс сприяє підвищенню рівня цифрової грамотності користувачів, стимулює інтерес до вивчення української мови та підтримує концепцію «інтелектуального навчання».

Публікації відображають результати розроблення програмно-апаратного комплексу для вивчення української мови з використанням штучного інтелекту, за якими опубліковано тези доповіді [1], а також подано три заявки на реєстрацію свідоцтв на авторське право на створений програмно-апаратний комплекс.

Тетерев В.І., Азарова А.О. Програмно-апаратний комплекс для вивчення української мови з використанням штучного інтелекту. Науково-технічна конференція ВНТУ Молодь в науці : Електронне наукове видання матеріалів

конференції, м. Вінниця, 2025. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/26626

Апробацію результатів магістерської роботи здійснено на Міжнародній науково-практичній Інтернет-конференції ВНТУ «Молодь в науці: дослідження, проблеми, перспективи» (м. Вінниця, 2025 р.).

1 АНАЛІЗ СУЧАСНИХ ТЕХНОЛОГІЙ ТА АРХІТЕКТУРНИХ РІШЕНЬ ПРОГРАМНО-АПАРАТНИХ КОМПЛЕКСІВ У СФЕРІ ОСВІТИ

1.1 Сучасні тенденції навчальних програмно-апаратних систем

Програмно-апаратні комплекси навчального призначення на сучасному етапі розвитку інформаційних технологій представляють собою складні інтегровані системи, що поєднують програмні алгоритми і апаратні засоби для організації ефективного та персоналізованого навчання. Вони спрямовані на підвищення інтерактивності освітнього процесу та створення умов для адаптації навчального матеріалу під потреби конкретного користувача. Застосування таких систем у вивченні мов, зокрема української, дозволяє автоматизувати оцінку усних та письмових навичок учнів, відстежувати прогрес у реальному часі та формувати адаптивні навчальні траєкторії, що враховують індивідуальні особливості та темп навчання.

Сучасні тенденції розвитку програмно-апаратних систем включають активну інтеграцію технологій штучного інтелекту, таких як автоматичне розпізнавання голосу, синтез мовлення та обробка природної мови. У межах підготовки даного матеріалу було переглянуто та проаналізовано низку статей із журналів, що індексуються в базі Scopus, які описують актуальні підходи до застосування ASR, діалогових систем, чатботів і великих мовних моделей у мовній освіті.

Зокрема, Нго, Фунг і Чень у метааналізі демонструють, що автоматичне розпізнавання мовлення може суттєво підтримувати розвиток вимови та усного мовлення завдяки частому тренуванню і швидкому зворотному зв’язку. Перевагою підходу є масштабованість і можливість регулярної практики без постійної присутності викладача, однак недоліками залишаються залежність якості фідбеку від точності ASR, чутливість до шумів та акцентів, а також неоднорідність ефекту в різних умовах навчання [2].

Огляд Шадієва та Лю узагальнює практики використання ASR у

допоміжному вивченні мови та підкреслює, що найкращі результати дає поєднання розпізнавання з чіткою методикою подання помилок і рекомендацій. Перевага — систематизація типових сценаріїв застосування ASR у навчанні, недолік — те, що значна частина рішень оцінюється на обмежених вибірках, а критерії ефективності (метрики) між роботами часто несумісні, що ускладнює пряме порівняння підходів [3].

Окремим напрямом є діалогові системи для розвитку говоріння. Хоу та Мін у метааналізі діалогових CALL-систем показують, що розмовні агенти здатні підвищувати мовленнєву активність і зменшувати страх помилки через «безпечний» простір спілкування. Перевага — тренування навичок діалогу в наближених до реальності умовах, недоліки — обмеження природності діалогу, потреба в сценарному контролі та ризик формування неприродних мовних шаблонів у разі слабких моделей розуміння контексту [4].

Використання ASR у мобільних практиках підтверджує дослідження Цзяна, Чень і Хуана щодо мобільної «диктовки» з підтримкою розпізнавання: підхід може покращувати точність і плавність усного мовлення через багаторазові короткі вправи й миттєвий фідбек. Перевага — висока доступність (смартфон як платформа), недолік — залежність від якості мікрофона/середовища, а також ризик того, що учень «підлаштовується під розпізнавач», а не під реальну комунікативну зрозумілість [5].

Сун, Лю, Чжан та співавтори розглядають поєднання ASR із взаємоперевіркою (peer correction) і показують, що комбінація автоматичного та соціального фідбеку підсилює навчальний ефект. Перевага — більш глибока рефлексія помилок і мотиваційна підтримка, недоліки — нестабільна якість «людського» фідбеку, додаткове навантаження на організацію процесу та питання конфіденційності аудіоданих [6].

Еверс і Чень у дослідженні з peer feedback на базі ASR демонструють, що структурований зворотний зв’язок (автоматичний + від одногрупників) може підвищувати результати вимови дорослих учнів. Перевага —

комбінування автоматизації та навчання через взаємодію, недоліки — потреба в модерації якості взаємооцінювання, можливі упередження між учасниками та залежність від стабільності ASR у «живих» умовах [7].

У іншій роботі Еверс і Чень показують, що ефективність ASR-підтримки може відрізнятися залежно від навчальних стилів і способу подання підказок. Перевага — аргументація необхідності персоналізації інтерфейсу й формату фідбеку, недолік — ризик нерівномірних результатів для різних груп користувачів без адаптивних механізмів (наприклад, різних візуалізацій і пояснень) [8].

Тімпе-Лафлін, Сидоренко та Дауріо аналізують використання spoken dialogue technology (SDS) у практиці говоріння та ставлення викладачів до таких інструментів. Перевага — можливість тренувати комунікативні сценарії (питання–відповідь, уточнення, реакції), недоліки — складність інтеграції в навчальну програму, ризик «збоїв» діалогу та потреба в обмеженні відкритих відповідей для підтримки стабільності системи [9].

Паралельно розвиваються чатботи для навчання мов як «легка» форма діалогового інтерфейсу. Хуан, Г’ю та Фраєр у систематичному огляді показують, що чатботи можуть підвищувати мотивацію й частоту практики завдяки доступності та швидким відповідям. Перевага — низький поріг входу для користувача, недоліки — обмежена глибина педагогічної корекції, ризик шаблонних відповідей і зниження якості навчання без методично продуманих сценаріїв [10].

Чон досліджує «можливості» (affordances) AI-чатботів у класі EFL для молодших учнів: інструмент підтримує залученість і дає можливість частіше практикувати мову. Перевага — зростання активності та інтересу, недолік — потреба у вікових обмеженнях, контролі контенту та ризики помилкових/небажаних відповідей без модерації [11].

Аннамалай, Рашид, Хашмі та співавтори на прикладі вищої освіти показують, що чатботи можуть бути корисними як допоміжний тьютор для

вправ, уточнень і повторення матеріалу. Перевага — підтримка навчання поза аудиторією, недоліки — нерівномірна якість відповідей, обмеженість корекції «помилок мислення» та ризик надмірної залежності студента від підказок [12].

Аннамалай, Елтахір, Зйоуд та співавтори аналізують використання чатботів з позиції теорії самодетермінації й підкреслюють роль автономії, компетентності та залученості. Перевага — пояснення мотиваційних механізмів, недолік — те, що позитивні ефекти залежать від якості сценаріїв і дизайну взаємодії; без цього чатбот перетворюється на «довідник», а не навчальний інструмент [13].

Ду та Деніел у систематичному огляді AI-чатботів для практики говоріння EFL узагальнюють підходи до побудови розмовної практики та оцінювання результатів. Перевага — структуризація доказів щодо розвитку говоріння, недоліки — обмежена стандартизація метрик, різна тривалість експериментів і слабка відтворюваність результатів між контекстами навчання [14].

Лай та Лі у систематичному огляді розглядають інструменти conversational AI в ELT і підкреслюють, що ефективність найбільша там, де є інтеграція аналітики, адаптації складності та педагогічних підказок. Перевага — акцент на необхідності «зв’язки» ШІ з методикою, недолік — типова проблема безпеки/приватності даних та необхідність людського контролю якості відповідей і рекомендацій [15].

Окремо, з появою великих мовних моделей актуальним стає порівняння LLM із класичними інструментами. Ші, Чай, Чжоу та Обрі порівнюють ChatGPT і автоматизоване оцінювання письма (AWE) у навчанні письма: LLM здатна давати розгорнуті пояснення та приклади перефразування, тоді як AWE забезпечує більш стабільні формальні метрики. Перевага LLM — багатство зворотного зв’язку й індивідуалізація, недоліки — ризик фактологічних помилок/«галюцинацій», складність контролю академічної доброчесності та потреба в механізмах верифікації порад [16].

Важливою характеристикою сучасних програмно-апаратних комплексів є їхня інтеграція з апаратними платформами, включно з інтерактивними панелями, планшетами та мобільними пристроями. Це забезпечує користувачам можливість працювати у гібридному середовищі, де поєднуються фізична взаємодія з сенсорними пристроями і цифрові алгоритми обробки навчальних даних. Однією з ключових тенденцій є підвищення мобільності та універсальності систем, що дозволяє учням взаємодіяти з навчальним середовищем у будь-який час і в будь-якому місці, а також збирати навчальну телеметрію для подальшої адаптації навчального контенту.

На рис. 1.1 наведено типову архітектуру хмарної освітньої системи з елементами програмно-апаратної інтеграції, яка демонструє взаємодію між клієнтськими пристроями, сервісами штучного інтелекту та освітніми базами даних. Така архітектура відображає сучасний підхід до побудови навчальних систем, де ключову роль відіграє хмарна інфраструктура, що забезпечує масштабованість, розподілене оброблення даних і віддалений доступ користувачів до навчальних ресурсів.

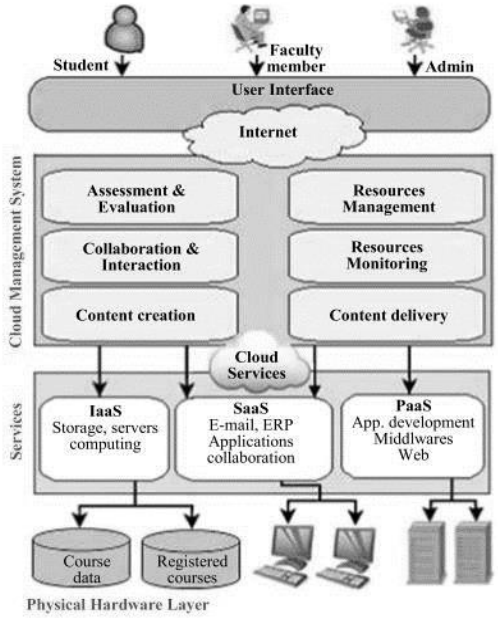

Рисунок 1.1 — Типова архітектура хмарної освітньої системи з елементами програмно-апаратної інтеграції

Додатковим напрямом розвитку є впровадження розподілених обчислювальних модулів із можливістю обробки даних на периферії (Edge Computing). Такий підхід дозволяє значно знизити затримки при обробці мовних сигналів і забезпечити швидкий відгук системи у реальному часі, що критично для інтерактивних мовних тренажерів, де якість користувацького досвіду напряму залежить від швидкості реакції на вимову або репліку.

Одночасно активно застосовуються контейнеризовані сервіси та мікросервісна архітектура, що дозволяє окремим компонентам ПАК (аналіз мови, синтез голосу, генерація завдань) працювати незалежно і масштабуватися за потреби. Наприклад, у великому класі або лабораторії кількість одночасних користувачів може різко зростати — у цьому випадку кожен сервіс може бути розгорнутий в окремому контейнері, що підвищує стабільність роботи системи та дає можливість гнучко керувати ресурсами для ASR, діалогового модуля та аналітики.

Ще одним перспективним напрямом є апаратне прискорення обробки нейромережевих моделей через FPGA або спеціалізовані NPU, що дозволяє виконувати складні алгоритми NLP без необхідності постійного підключення до хмарних серверів. Це особливо важливо для локальних навчальних центрів або мобільних лабораторій, де обмежена пропускна здатність мережі та критична вимога до приватності мовленнєвих даних.

В сучасних ПАК також зростає роль інструментів телеметрії та моніторингу стану системи: сенсори температури та навантаження процесорів, логування подій та контролери живлення. Вони дозволяють забезпечувати стабільну роботу апаратних вузлів, своєчасне виявлення відмов і автоматичне перенаправлення обчислень на резервні модулі. Застосування цих технологій створює технічну основу для високопродуктивного, масштабованого та надійного ПАК, який здатний одночасно обслуговувати велику кількість учнів і підтримувати складні адаптивні сценарії навчання української мови.

Сучасні програмно-апаратні системи активно розвивають аналітичні

функції, що дозволяють відстежувати поведінку користувачів, оцінювати ефективність виконання завдань і визначати слабкі місця у засвоєнні матеріалу. Завдяки цьому педагогам надається можливість своєчасно коригувати навчальний процес і приймати обґрунтовані рішення щодо розвитку навчальних програм. Одночасно з цим розвиваються технології взаємодії з користувачем, що включають інтелектуальні інтерфейси, адаптивні підказки та систему рекомендацій, які підсилюють залученість і підтримують персоналізацію.

Надзвичайно важливим напрямом розвитку є створення комплексних інтегрованих систем, які поєднують апаратну та програмну частини з інтелектуальними алгоритмами. Такі системи дозволяють реалізувати повноцінне середовище навчання, в якому автоматично генеруються навчальні матеріали, відслідковуються результати виконання завдань, проводиться оцінка знань та формуються рекомендації для подальшого навчання. Інтеграція великих мовних моделей і сучасних апаратних платформ відкриває нові перспективи для розвитку адаптивного навчання української мови, забезпечуючи високий рівень персоналізації та ефективності освітнього процесу.

Розвиток програмно-апаратних систем навчального призначення сприяє також підвищенню масштабованості навчальних рішень, що дозволяє забезпечити одночасне навчання великої кількості користувачів із збереженням індивідуального підходу до кожного. Це особливо важливо для дистанційної освіти та навчання у гібридному форматі, коли апаратно-програмні рішення забезпечують стабільність роботи, інтерактивність та безперервний доступ до навчальних ресурсів.

Аналіз статей із бази Scopus показав, що актуальні підходи (ASR-тренажери, діалогові системи, чатботи та LLM) забезпечують масштабованість, часту практику і персоналізований фідбек, але мають спільні обмеження: залежність від точності розпізнавання, чутливість до умов запису

й акцентів, нестабільність якості рекомендацій, складність педагогічної інтеграції та ризики приватності/контролю контенту. Тому, враховуючи недоліки наявних підходів, у роботі доцільно обрати гібридний підхід побудови програмно-апаратного комплексу: швидке локальне (edge) опрацювання мовлення для мінімальної затримки та підвищення приватності, поєднане з хмарними сервісами NLP/LLM для генерації вправ і пояснень, а також з модульною (мікросервісною) архітектурою та аналітикою прогресу, що забезпечує керованість, масштабованість і адаптивність навчання української мови.

1.2 Архітектура освітніх програмно-апаратних комплексів

Архітектура програмно-апаратних комплексів (ПАК) для освітніх систем визначає взаємодію між технічними й програмними компонентами, що забезпечують реалізацію навчальних процесів у цифровому середовищі. Такі комплекси поєднують обчислювальні ресурси, мережеві інтерфейси, модулі штучного інтелекту, бази даних та елементи користувацького інтерфейсу в єдину інтегровану систему. Основна мета архітектури полягає у створенні гнучкої, масштабованої та надійної структури, здатної адаптуватися до змін вимог користувачів та технологічного середовища .

На рис. 1.2 наведено узагальнену архітектуру апаратної системи штучного інтелекту для освітніх застосунків, що демонструє взаємозв’язок між апаратними модулями, сервісами обробки даних та користувацьким інтерфейсом. Така архітектура відображає базові принципи побудови більшості сучасних навчальних ПАК, зокрема систем типу Google Classroom AI, Duolingo AI Tutor та LingQ Smart Learning Environment, які поєднують аналітику користувацьких дій і адаптивні механізми навчання.

Разом із тим, аналіз архітектур подібних систем показує низку характерних обмежень. Наприклад, Duolingo AI Tutor використовує централізовану хмарну архітектуру, що ускладнює локальне розгортання та

обмежує можливості персоналізації навчального середовища. Google Classroom AI не передбачає інтеграції апаратних модулів для офлайн-взаємодії (наприклад, сенсорних пристроїв чи мікроконтролерів), що знижує ефективність роботи студентів у гібридному форматі. LingQ натомість має розвинену систему NLP-модулів, але її архітектура орієнтована на англомовні корпуси, що обмежує ефективність при вивченні української мови.

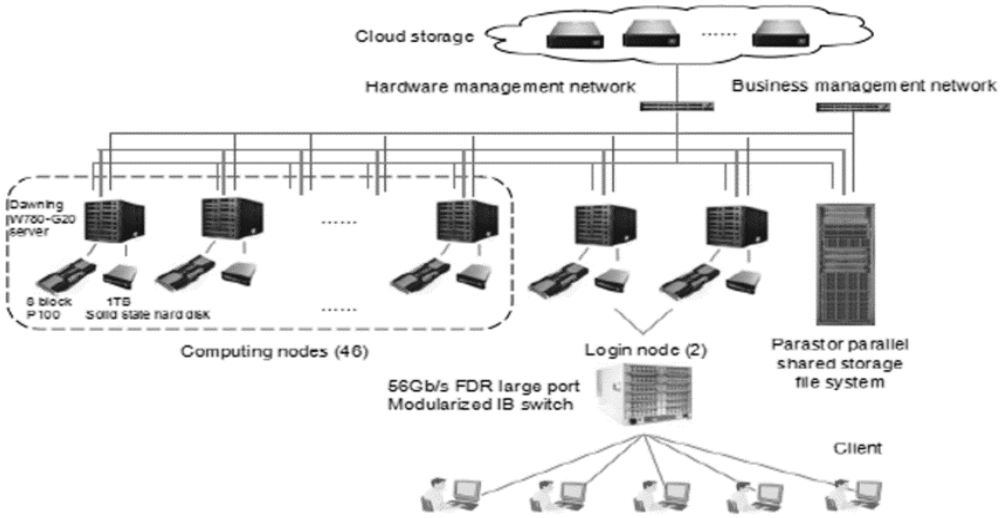

Рисунок 1.2 — Архітектура апаратної системи штучного інтелекту для освітніх застосунків

Типовий програмно-апаратний комплекс освітнього призначення складається з кількох взаємопов’язаних рівнів. На апаратному рівні використовується набір пристроїв введення й виведення даних — персональні комп’ютери, планшети, мікрофони, камери, інтерактивні панелі, серверне обладнання або мікроконтролери. Ці пристрої забезпечують фізичну взаємодію користувача із системою, збір мовних, текстових чи візуальних сигналів і передачу їх на обробку.

Програмний рівень охоплює операційні системи, прикладне програмне забезпечення, драйвери, бази даних і сервіси штучного інтелекту. Центральною частиною є програмна платформа, яка виконує обробку даних, аналіз результатів і взаємодію з користувачем через інтерфейс. У контексті

вивчення української мови важливу роль відіграють модулі обробки природної мови (NLP), системи розпізнавання мовлення, синтезу голосу та машинного перекладу, які інтегруються з апаратними компонентами через API або мережеві протоколи[17].

Важливою характеристикою архітектури є її модульність. Це дозволяє гнучко замінювати або вдосконалювати окремі частини системи без необхідності повного перероблення комплексу. Наприклад, у навчальному ПАК можна оновити модель штучного інтелекту для аналізу мовних помилок, не змінюючи інтерфейс користувача чи структуру бази даних. Такий підхід спрощує обслуговування, масштабування та впровадження нових функцій.

Для забезпечення ефективності роботи освітнього ПАК важливими є також мережеві технології. Хмарні сервіси дозволяють зберігати великі обсяги навчальних матеріалів, забезпечують доступ із будь-якої точки світу та синхронізацію користувацьких даних. Використання контейнеризації (наприклад, Docker) і мікросервісної архітектури підвищує надійність системи та дає змогу паралельно розгортати кілька компонентів[18].

На рисунку 1.3 показано хмарну модель реалізації освітнього програмно-апаратного комплексу, яка ілюструє розподіл компонентів між клієнтською частиною, серверною інфраструктурою та хмарними сервісами штучного інтелекту. Подібні підходи використовуються у таких освітніх системах, як Coursera AI Infrastructure та Khan Academy AI Mentor, що базуються на моделі мікросервісної взаємодії. Проте їх архітектури часто страждають від надмірної залежності від зовнішніх API, що може створювати проблеми з безпекою даних та стабільністю при обмеженому інтернет-з’єднанні.

Загалом архітектура програмно-апаратного комплексу для освітніх застосунків повинна поєднувати високу продуктивність апаратних засобів, інтелектуальні алгоритми обробки даних і зручний інтерфейс для користувача. Її якісне проєктування є основою для створення ефективних систем навчання,

що використовують штучний інтелект і відповідають сучасним вимогам цифрової освіти.

Проведений аналіз існуючих архітектур показує, що більшість наявних освітніх програмно-апаратних комплексів не забезпечують повної адаптивності. Основними недоліками таких систем є відсутність інтегрованих локальних апаратних модулів, обмежена підтримка офлайн-режиму, а також недостатня мовна адаптація NLP-компонентів.

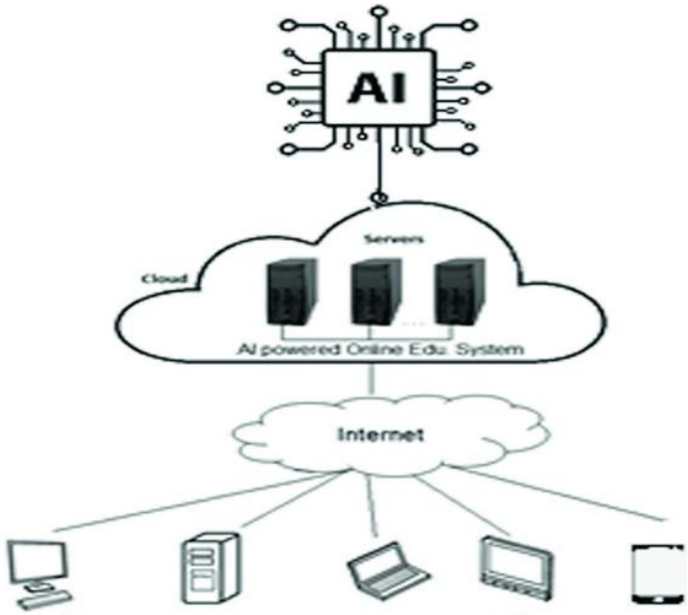

Рисунок 1.3 — Хмарна модель реалізації освітнього програмно-апаратного комплексу

Тому в подальших розділах даної роботи буде представлено власну архітектуру програмно-апаратного комплексу для вивчення української мови із застосуванням ШІ, яка покликана усунути виявлені недоліки, забезпечити гнучку модульну структуру та поєднати апаратну й програмну частини в єдину інтелектуальну екосистему навчання.

1.3 Використання штучного інтелекту у програмно-апаратних системах освіти

Штучний інтелект (ШІ) є ключовим елементом сучасних програмно-

апаратних комплексів, що призначені для автоматизації освітніх процесів. Його застосування дозволяє не лише оптимізувати подання навчального матеріалу, а й забезпечити індивідуалізацію освітнього підходу, адаптацію до рівня знань користувача, а також створення інтелектуальних інструментів аналізу результатів навчання. У межах таких систем ШІ виступає як аналітичний, діагностичний і рекомендаційний модуль, який взаємодіє з іншими компонентами комплексу через стандартизовані протоколи обміну даними.

Основні напрями використання ШІ в освітніх системах включають розпізнавання природної мови, аналіз текстів, автоматичну оцінку знань, генерацію навчального контенту, прогнозування результатів та моделювання поведінки користувачів. Для вивчення української мови особливе значення мають системи обробки природної мови (NLP), які дозволяють здійснювати морфологічний, синтаксичний і семантичний аналіз речень, а також автоматично визначати граматичні помилки, лексичні неточності чи порушення синтаксису. Такі технології лежать в основі адаптивних тренажерів, чат-ботів і голосових асистентів, здатних проводити інтерактивні уроки або діалогові вправи українською мовою[19].

Аналіз існуючих програмно-апаратних комплексів, які інтегрують технології штучного інтелекту в освітнє середовище, свідчить про різноманітність архітектурних підходів та рівень застосування нейронних мереж. Так, Duolingo Max базується на мовній моделі GPT-4, що дозволяє системі генерувати діалоги та здійснювати контекстне пояснення помилок. Однак така архітектура повністю залежить від хмарних обчислень і не передбачає локальної оптимізації моделей для конкретної мови, зокрема української[20].

Coursera AI Learning Companion використовує моделі типу BERT і T5 для семантичного аналізу текстів курсів і запитань студентів, що забезпечує автоматичну побудову навчальних рекомендацій. Проте така система

переважно орієнтована на англомовні корпуси та не має спеціалізованих модулів для адаптації до морфологічних особливостей української мови[21].

У системі Khan Academy AI Mentor застосовується гібридна архітектура з моделями Transformer у поєднанні з аналітичними модулями Learning Analytics. Це забезпечує високий рівень адаптивності, проте обмеження полягає у відсутності глибокої персоналізації для окремих користувачів і недостатньому урахуванні емоційно-комунікативного аспекту навчання.

У програмно-апаратних системах важливим аспектом є реалізація моделей машинного навчання безпосередньо на пристроях користувачів або в хмарному середовищі. Вбудовані нейромережеві процесори (NPU) чи графічні прискорювачі (GPU) дозволяють виконувати складні обчислення локально, що підвищує швидкодію і зменшує затримки при обробці мовних сигналів. У хмарних архітектурах, навпаки, основні обчислення виконуються на потужних серверах, що забезпечує масштабованість і можливість навчання великих моделей ШІ. Оптимальна архітектура визначається балансом між продуктивністю, безпекою та доступністю ресурсів.

Інтелектуальні освітні системи використовують різні підходи до реалізації навчальних моделей. Зокрема, нейронні мережі типу Transformer (наприклад, BERT, GPT, LLaMA, Mistral) дозволяють здійснювати високоточний аналіз контексту і генерацію природних текстів українською мовою. Їх інтеграція у ПАК відкриває можливості для створення систем автоматичного пояснення граматичних конструкцій, оцінювання мовлення учня в реальному часі або персоналізованої побудови навчальних маршрутів[22].

На основі порівняння підходів у зазначених системах можна відзначити, що найефективнішими для створення інтелектуальних освітніх ПАК є моделі архітектури Transformer, зокрема їх сучасні варіації — GPT-4-turbo, Claude 3, LLaMA 3, які мають здатність до багатомовної генерації та контекстного розуміння української мови. У той час як системи, побудовані на попередніх

архітектурах, таких як RNN чи Seq2Seq, демонструють нижчу точність у синтаксичному розборі та обмежену можливість контекстного навчання, що знижує ефективність інтерактивної взаємодії в навчальному процесі.

Впровадження таких моделей у програмно-апаратний комплекс дозволить забезпечити не лише високу якість аналізу текстів і мовлення, а й створити можливість контекстної адаптації навчальних сценаріїв під конкретного користувача.

Ще одним напрямом використання ШІ у програмно-апаратних системах є аналітика навчальних даних (Learning Analytics). Вона дозволяє визначати закономірності у поведінці студентів, прогнозувати їхній прогрес, своєчасно виявляти труднощі у засвоєнні матеріалу та надавати рекомендації викладачам. Завдяки цьому навчальний процес стає не лише ефективнішим, а й більш адаптивним до потреб користувачів.

Важливою складовою впровадження ШІ є етичні та безпекові аспекти. Програмно-апаратні системи мають забезпечувати захист персональних даних, уникати упередженості алгоритмів і гарантувати прозорість прийняття рішень. Використання ШІ в освітніх технологіях повинно бути спрямоване на підтримку викладача, а не на його заміну, зберігаючи при цьому гуманістичний характер навчання.

Отже, штучний інтелект у програмно-апаратних системах освіти виступає центральною ланкою, що поєднує аналітичні, мовні та адаптивні можливості сучасних технологій. Його впровадження дозволяє створювати нові моделі навчання, які орієнтовані на особистість користувача, забезпечують динамічну взаємодію з освітнім контентом і сприяють підвищенню якості вивчення української мови в умовах цифрового середовища.

Аналіз показує, що наявні освітні ПАКи, які використовують ШІ, мають низку архітектурних обмежень — зокрема, залежність від хмарної інфраструктури, відсутність локальної обробки мовних даних, обмежену

підтримку української морфології та недостатню інтеграцію із сенсорними апаратними модулями.

Враховуючи це, у розроблюваному в другому розділі програмно-апаратному комплексі доцільно застосувати моделі типу Transformer, оптимізовані для української мови, які забезпечують високу точність морфологічного аналізу та гнучкість у генерації навчального контенту. Такий підхід дозволить досягти більшої адаптивності системи, підвищити якість автоматизованої оцінки знань та створити персоналізоване навчальне середовище[23].

1.4 Технічні засоби реалізації програмно-апаратних комплексів

Програмно-апаратні комплекси (ПАК) освітнього призначення ґрунтуються на поєднанні сучасного апаратного забезпечення та інтелектуальних програмних компонентів. Технічна частина таких систем визначає їхню продуктивність, швидкодію, стабільність роботи та можливість інтеграції зі сторонніми сервісами. Для створення ефективного ПАК необхідно враховувати архітектуру обчислювальних пристроїв, канали передачі даних, сенсорні системи, периферійні модулі, а також засоби введення та виведення інформації[24].

До базових апаратних засобів належать центральні процесори (CPU), графічні процесори (GPU) та нейронні прискорювачі (NPU), які забезпечують виконання складних обчислень, пов’язаних із машинним навчанням, обробкою природної мови чи розпізнаванням зображень. Вибір між цими типами процесорів залежить від архітектури системи: локальні пристрої часто використовують енергоефективні CPU або інтегровані GPU, тоді як хмарні рішення базуються на потужних обчислювальних вузлах із паралельною обробкою даних.

Важливим компонентом ПАК є модулі введення мовної інформації — мікрофони, сенсорні панелі, камери та графічні планшети. Вони забезпечують

взаємодію користувача із системою у реальному часі, дозволяючи фіксувати мовлення, жести або текстові введення. Для вивчення української мови особливо цінними є високочутливі мікрофонні масиви з підтримкою шумозаглушення, які підвищують точність розпізнавання мовлення навіть у шумному середовищі.

До периферійних елементів належать пристрої візуалізації — інтерактивні панелі, проектори, дисплеї з тактильним зворотним зв’язком. Вони дають змогу реалізувати елементи гейміфікації, інтерактивного навчання та візуалізації результатів навчального процесу. Додатково можуть використовуватись сенсорні браслети або камери відстеження погляду для моніторингу залученості учня, що є важливою частиною адаптивного навчання.

Аналіз існуючих освітніх систем, що використовують апаратну інтеграцію, свідчить про наявність різних технічних підходів до реалізації. Так, у системі Google Classroom Hardware Integration Kit переважає хмарно-орієнтована архітектура, яка базується на звичайних вебкамерних модулях і стандартних ноутбуках, проте не передбачає локальної обробки мовних даних і роботи без мережевого підключення.

У проєктах NVIDIA Jetson Education Platform і Coral AI Edge TPU Learning Kit використовуються вбудовані нейронні прискорювачі, що дозволяють запускати моделі машинного навчання безпосередньо на пристрої, зменшуючи затримки при розпізнаванні мовлення. Проте такі рішення часто мають високу вартість і потребують додаткової оптимізації під конкретну мову [25].

У противагу цьому, рішення на базі Orange Pi 5 демонструють гарний баланс між вартістю, енергоефективністю та можливістю підключення зовнішніх сенсорів, однак обмежені у продуктивності при роботі з великими мовними моделями. Це обумовлює необхідність комбінованого підходу — частина ШІ-алгоритмів виконується локально, тоді як обчислювально складні

моделі функціонують у хмарі[26].

Мережеві модулі — ще один ключовий елемент апаратної структури. Використання Wi-Fi 6, Bluetooth 5.3 або 5G-з'єднань дозволяє створювати стабільні канали комунікації між компонентами системи, забезпечуючи обмін даними між локальними пристроями і хмарним сервером ШІ. У випадках, коли система використовується у навчальному класі або лабораторії, може застосовуватися гібридна мережа з локальним оброблення для зменшення затримок і центральною синхронізацією результатів у хмарі.

Для забезпечення відмовостійкості системи доцільно застосовувати резервування ключових компонентів: дубльовані сенсорні масиви, резервні контролери живлення та подвійні канали зв'язку. Це дозволяє підтримувати безперервність навчального процесу навіть при часткових апаратних збоях.

Серед апаратних платформ, що найчастіше застосовуються у розробці освітніх систем, варто виділити Raspberry Pi, NVIDIA Jetson Nano, Google Coral та Arduino. Raspberry Pi забезпечує достатню продуктивність для виконання базових мовних алгоритмів та взаємодії з периферією. Jetson Nano і Google Coral оснащені GPU або TPU-модулями, що дає можливість працювати з моделями машинного навчання без підключення до потужних серверів. Arduino, своєю чергою, є ефективною платформою для реалізації сенсорних модулів, контролю апаратних процесів і збору навчальних даних із зовнішнього середовища[27].

З боку програмного забезпечення важливу роль відіграють драйвери, операційні системи реального часу (RTOS), середовища для машинного навчання (TensorFlow, PyTorch, ONNX Runtime), а також мови програмування нижчого рівня (C++, Rust) для керування апаратними процесами. У поєднанні з високорівневими інструментами (Python, Node.js, Java) вони дозволяють створювати інтерактивні навчальні додатки з інтелектуальними функціями, здатними до аналізу мовних даних, оцінювання результатів та адаптації до користувача.

Енергоефективність та охолодження є ще двома критично важливими аспектами технічної реалізації. У навчальних закладах або мобільних середовищах система має працювати тривалий час без перегріву і з мінімальним енергоспоживанням. Для цього застосовуються системи активного охолодження (вентилятори, теплові трубки) або пасивного (алюмінієві радіатори, термопрокладки), а також технології керування енергоспоживанням на рівні процесора.

З урахуванням проведеного аналізу існуючих апаратних платформ, для реалізації програмно-апаратного комплексу, описаного у другому розділі, доцільно обрати гібридну архітектуру, яка поєднує локальну обробку на Raspberry Pi 5 або Jetson Nano з хмарним розгортанням мовних моделей. Такий підхід забезпечує автономність системи при збереженні високої обчислювальної потужності.

Крім того, передбачається використання модульних сенсорних систем (мікрофонні масиви ReSpeaker, камери Raspberry HQ), інтерактивних панелей із підтримкою multi-touch і Bluetooth-модулів нового покоління для стабільної комунікації з периферією. Це створить умови для максимальної інтеграції апаратної частини з програмними компонентами, що реалізують штучний інтелект.

Таким чином, технічна реалізація програмно-апаратного комплексу для вивчення української мови із використанням ШІ передбачає збалансоване поєднання продуктивних обчислювальних компонентів, надійних сенсорів, мережевих модулів та інтелектуальних інтерфейсів. Від правильного вибору апаратних засобів залежить не лише ефективність роботи системи, але й можливість інтеграції ШІ-алгоритмів, масштабування у реальних умовах.

1.5 Інтеграція програмного та апаратного забезпечення у навчальних системах

Інтеграція програмного та апаратного забезпечення є ключовим етапом

у створенні повноцінних програмно-апаратних комплексів (ПАК), оскільки саме вона забезпечує узгоджену взаємодію між усіма компонентами системи. У сфері освітніх технологій, де необхідно обробляти як текстову, так і аудіовізуальну інформацію, така інтеграція набуває особливого значення. Вона дозволяє реалізувати адаптивне навчання, персоналізований зворотний зв’язок і підтримку мовного середовища у режимі реального часу.

Основна задача інтеграції полягає в тому, щоб забезпечити безперервний обмін даними між апаратними сенсорами (мікрофонами, камерами, сенсорними панелями), обчислювальними модулями (CPU, GPU, NPU) та програмними компонентами, які аналізують і обробляють отримані дані. Для цього використовуються проміжні програмні шари — драйвери, SDK (Software Development Kits) та API (Application Programming Interfaces), які виступають “мостом” між фізичними пристроями та логічною частиною системи.

У типових освітніх комплексах взаємодія програмного й апаратного рівнів відбувається за клієнт-серверною моделлю. Локальні пристрої учня (планшет або комп’ютер) збирають дані про дії користувача — введення тексту, аудіо, відео чи натискання сенсорних елементів — і передають їх на серверну частину, де працює модуль штучного інтелекту. Сервер, у свою чергу, обробляє ці дані, визначає рівень виконання завдання, оцінює мовні помилки, а потім надсилає результати назад на клієнтський пристрій. Такий підхід дозволяє забезпечити масштабованість системи та централізоване оновлення моделей машинного навчання.

Для реалізації інтеграції часто застосовуються універсальні протоколи взаємодії: REST API, WebSocket, MQTT, або gRPC. REST API підходить для запитів, що не потребують постійного з’єднання, тоді як WebSocket забезпечує двосторонню комунікацію у режимі реального часу — наприклад, під час усного тестування чи тренування вимови. MQTT використовується у випадках, коли потрібно обмінюватися невеликими пакетами даних між

великою кількістю пристроїв, наприклад, у мережі класів із кількома інтерактивними панелями.

Важливою складовою інтеграції є синхронізація даних між апаратними модулями та програмним ядром. Наприклад, коли користувач промовляє слово, аудіопотік одночасно передається до модуля розпізнавання мовлення (ASR) і до інтерфейсу користувача, який візуалізує поточний рівень гучності та якість вимови. Ця синхронність можлива завдяки системам буферизації та часових міток (timestamps), що гарантують коректну обробку навіть при незначних мережевих затримках.

У рамках навчальних систем особливу роль відіграє AI-орієнтована інтеграція, яка полягає у поєднанні традиційних обчислювальних процесів із моделями машинного навчання. Для цього використовується контейнеризація (Docker) та оркестрація (Kubernetes), що дозволяють швидко розгортати модулі ШІ незалежно від конкретного апаратного середовища. Наприклад, мовна модель може бути розміщена у хмарі, тоді як локальний пристрій відповідає лише за запис голосу, передачу запиту та отримання відповіді. Такий підхід мінімізує вимоги до ресурсів клієнтського обладнання.

Інтеграція також передбачає захист даних і забезпечення безпеки користувачів. У випадку освітніх систем із використанням ШІ важливо дотримуватися принципів анонімності та шифрування. Для цього застосовуються протоколи TLS/SSL, токен-автентифікація (JWT) і контроль доступу на рівні користувацьких ролей. Це гарантує, що персональні дані учнів або викладачів не потраплять у сторонні руки, а система залишатиметься захищеною навіть при роботі через інтернет.

Ще одним аспектом є сумісність компонентів. Програмне забезпечення має бути кросплатформним — підтримувати різні операційні системи (Windows, Linux, Android), типи пристроїв та архітектури процесорів. Для цього застосовуються фреймворки на кшталт Electron, Flutter, Qt або React Native, які дозволяють створювати єдину базу коду для різних платформ. Це

знижує витрати на розробку і спрощує технічну підтримку комплексу.

Таким чином, інтеграція програмного та апаратного забезпечення у навчальних системах є фундаментом для створення ефективних і гнучких рішень у сфері цифрової освіти. Від якості цієї інтеграції залежить стабільність системи, точність роботи моделей штучного інтелекту, а також зручність взаємодії користувачів із навчальним середовищем. Оптимальне поєднання апаратної продуктивності та інтелектуального програмного забезпечення формує основу майбутніх AI-комплексів для вивчення української мови.

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ ДЛЯ ВИВЧЕННЯ УКРАЇНСЬКОЇ МОВИ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

2.1 Розроблення структурної моделі програмно-апаратного комплексу

Програмно-апаратний комплекс для вивчення української мови з використанням штучного інтелекту є інтегрованою системою, що поєднує апаратні та програмні компоненти з метою створення інтелектуального навчального середовища. Основним завданням комплексу є забезпечення можливості інтерактивного навчання української мови (з урахуванням іноземної аудиторії) шляхом голосової взаємодії користувача з комп’ютерною системою, автоматичного розпізнавання мовлення, аналізу тексту та генерації відповідей у зручній для користувача формі.

У структурному плані комплекс складається з трьох основних частин: апаратної, програмної та інформаційної.

До апаратної частини належать пристрої введення та виведення — мікрофон, динаміки та персональний комп’ютер (ноутбук), на якому розміщується програмне забезпечення. Апаратна складова забезпечує отримання голосового сигналу від користувача, його перетворення на цифрову форму, а також відтворення синтезованого мовлення системи.

Програмна частина реалізує інтелектуальне оброблення даних і включає такі модулі:

— модуль прийому мовлення, який здійснює захоплення звукового сигналу з мікрофона;

— модуль розпізнавання мовлення (ASR), що перетворює аудіоінформацію у текстову форму;

— модуль аналізу тексту, який перевіряє граматику, лексику та синтаксис із використанням методів обробки природної мови (NLP);

— модуль генерації навчальних завдань і відповідей, який формує інтерактивні вправи на основі помилок користувача та рівня його знань;

— модуль синтезу мовлення (TTS), який озвучує згенеровані відповіді;

— модуль користувацького інтерфейсу, що забезпечує взаємодію користувача з усіма компонентами системи.

Кожен модуль програмно-апаратного комплексу функціонує автономно, але взаємодіє з іншими через спільний обмін даними. Така модульна структура дає змогу легко оновлювати окремі компоненти без зміни всієї системи.

Зокрема, модуль розпізнавання мовлення побудовано на базі відкритої бібліотеки Vosk API, яка підтримує українську мову та забезпечує роботу в офлайн-режимі, що важливо для автономності комплексу.

Модуль синтезу мовлення використовує технології pyttsx3 або Coqui TTS, які генерують природне звучання української мови з урахуванням інтонаційних особливостей.

Для лінгвістичного аналізу використовується модель Gemini 2.0 Flash, яка виконує перевірку граматики та формує пояснення у форматі AI-відповіді. Модель аналізує текст цілісно, без застосування rule-based NLP або детального морфологічного розбору.

Взаємодія між модулями реалізується через внутрішню чергу подій (event queue), що забезпечує послідовну передачу результатів розпізнавання, аналізу та генерації. Це дозволяє підтримувати стабільну роботу системи навіть при тривалому навчальному сеансі.

Крім того, комплекс має можливість зберігати проміжні дані користувача (відповіді, статистику успішності, час виконання завдань), що забезпечує адаптивність навчального процесу.

Інформаційна частина комплексу містить навчальні матеріали, базу даних користувачів і моделі штучного інтелекту, необхідні для виконання мовного аналізу та генерації контенту. Зберігання даних реалізується у форматі локальної бази SQLite, що не потребує додаткових витрат і дозволяє швидко обробляти запити користувача.

Загальна структурна модель комплексу представлена у вигляді послідовної взаємодії модулів:

користувач → прийом мовлення → розпізнавання → аналіз тексту → генерація відповіді → синтез мовлення → зворотний зв’язок.

Структурну схему взаємодії основних модулів програмно-апаратного комплексу подано на рис. 2.1.

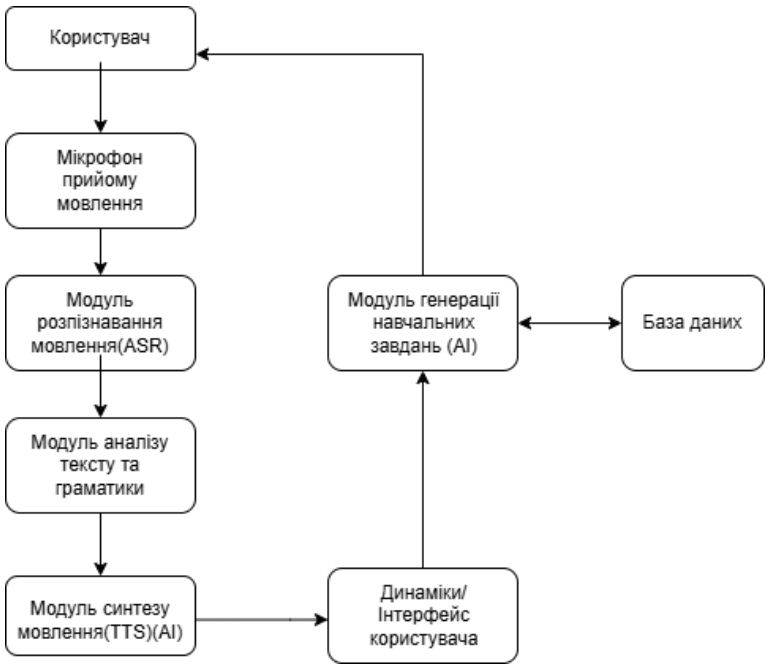

Рисунок 2.1 — Структурна схема програмно-апаратного комплексу для вивчення української мови з використанням штучного інтелекту

Такий підхід забезпечує циклічність навчального процесу, коли система не лише оцінює мовлення користувача, але й надає миттєвий зворотний зв’язок у голосовій або текстовій формі.

Розроблена структурна модель дозволяє забезпечити гнучкість та масштабованість системи, можливість інтеграції нових алгоритмів штучного інтелекту, а також розширення функціональності без необхідності зміни апаратної частини.

Комплекс може використовуватись як автономно, так і в складі навчальних платформ, що робить його універсальним інструментом для

вивчення української мови з урахуванням іноземної аудиторії, а також у різних умовах — від індивідуального навчання до використання у закладах освіти.

2.2 Проєктування архітектури основних модулів системи

Архітектура програмно-апаратного комплексу для вивчення української мови побудована за принципом модульності та взаємної незалежності компонентів. Це забезпечує гнучкість системи, можливість масштабування та подальшого розширення функціоналу без суттєвих змін у кодовій базі .

Основна мета архітектури — забезпечення послідовного потоку даних між модулями, від моменту введення голосу користувачем до формування зворотного голосового або текстового повідомлення. Кожен модуль виконує чітко визначену функцію і взаємодіє з іншими за допомогою внутрішнього API, що спрощує розробку та налагодження системи.

Модуль прийому мовлення. Цей модуль є початковою складовою архітектури програмно-апаратного комплексу і відповідає за захоплення голосового сигналу користувача з мікрофона та його подальше перетворення у цифрову форму. Від якості цього етапу залежить точність усього процесу навчання, тому система повинна забезпечувати стабільний рівень запису, мінімізацію шумів і чіткість мовлення.

Для реалізації використовується бібліотека PyAudio, що надає інструменти для запису та обробки звукових потоків у реальному часі. Модуль ініціює підключення до активного мікрофона, визначає параметри аудіопотоку (частоту дискретизації, кількість каналів, глибину бітності) та створює буфер для збереження поточних даних. Найчастіше використовується моноформат із частотою дискретизації 16 кГц, що є стандартом для систем розпізнавання української мови.

Після початку запису система виконує попередню фільтрацію сигналу. Реалізуються прості алгоритми нормалізації гучності, видалення шумів навколишнього середовища та усереднення енергії звуку, щоб зменшити

вплив перепадів тону або відстані до мікрофона. Отриманий звуковий потік зберігається у тимчасовій пам’яті у форматі MP3 і передається на наступний етап обробки.

Для зручності користувача передбачено графічні елементи керування: кнопки «Почати запис» і «Завершити запис», а також індикатор активності мікрофона. Крім того, модуль реалізує функцію автоматичного визначення тиші — якщо користувач перестає говорити протягом певного часу (наприклад, 2-3 секунди), запис припиняється автоматично. Це робить систему більш зручною та адаптованою для природної взаємодії.

У разі відсутності підключеного мікрофона або помилки вводу система виводить повідомлення про проблему та пропонує користувачу перевірити пристрій. Таким чином, модуль прийому мовлення виступає надійним інтерфейсом між користувачем і програмною частиною комплексу, забезпечуючи перетворення фізичного звуку в цифрові дані з необхідним рівнем якості для подальшого аналізу.

Цей модуль є ключовою ланкою між аудіосигналом і текстовим представленням мовлення користувача. Його основне завдання — виконати перетворення звукових хвиль у послідовність символів, зрозумілих для системи обробки природної мови.

Для реалізації розпізнавання використовується бібліотека Vosk API, яка підтримує українську мову, працює офлайн і не потребує підключення до мережі Інтернет. Це забезпечує автономність системи та можливість її використання в умовах обмеженого доступу до мережі. Модуль аналізує аудіопотік, поділяє його на короткі звукові сегменти, виконує спектральний аналіз і визначає послідовність фонем. Після цього за допомогою мовної моделі формується текстовий результат[28].

Отриманий текст проходить перевірку на коректність і відправляється до модуля аналізу граматики. Якщо рівень достовірності розпізнавання нижчий за встановлений поріг, система повідомляє користувача про

необхідність повторення фрази або уточнення вимови. Таким чином, користувач може миттєво скоригувати свій голос, що підвищує ефективність навчального процесу.

Модуль також підтримує збереження журналу розпізнаних висловлювань, що дає змогу відстежувати прогрес користувача й проводити подальший аналіз помилок. У разі потреби результати розпізнавання можуть бути передані до модуля оцінювання для формування статистики правильних і неправильних відповідей.

У результаті модуль розпізнавання мовлення забезпечує точне, швидке та стабільне перетворення аудіоінформації у текстову, створюючи базу для подальшого інтелектуального аналізу та генерації навчальних завдань[29].

Технологічну послідовність обробки мовлення, зокрема етапи запису, попередньої обробки та розпізнавання звуку за допомогою бібліотек PyAudio та Vosk API, подано на рисунку 2.2.

Після розпізнавання тексту система передає його у модуль обробки природної мови. Для граматичного аналізу використовується LanguageTool, який визначає орфографічні та синтаксичні помилки, а також надає рекомендації щодо їх виправлення. Результати аналізу зберігаються у тимчасовій структурі даних і передаються до модуля генерації навчальних завдань. Крім перевірки, модуль виконує лінгвістичну класифікацію введеного речення — визначає частини мови, тип речення (стверджувальне, запитальне тощо) і складність конструкції. Це дає змогу надалі формувати вправи, що відповідають рівню користувача.

Даний модуль є центральним елементом системи. На основі результатів аналізу тексту він формує навчальні завдання різного типу: вправи на виправлення помилок, підбір правильного варіанту слова або складання речень. Для цього застосовуються правила, закладені в алгоритм системи, а також інформація з локальної бази даних, де зберігається рівень користувача та історія його результатів. Модуль реалізує адаптивну логіку: якщо

користувач часто повторює ті самі помилки, система знижує складність завдань і надає додаткові пояснення; у разі успішного виконання — підвищує рівень складності.

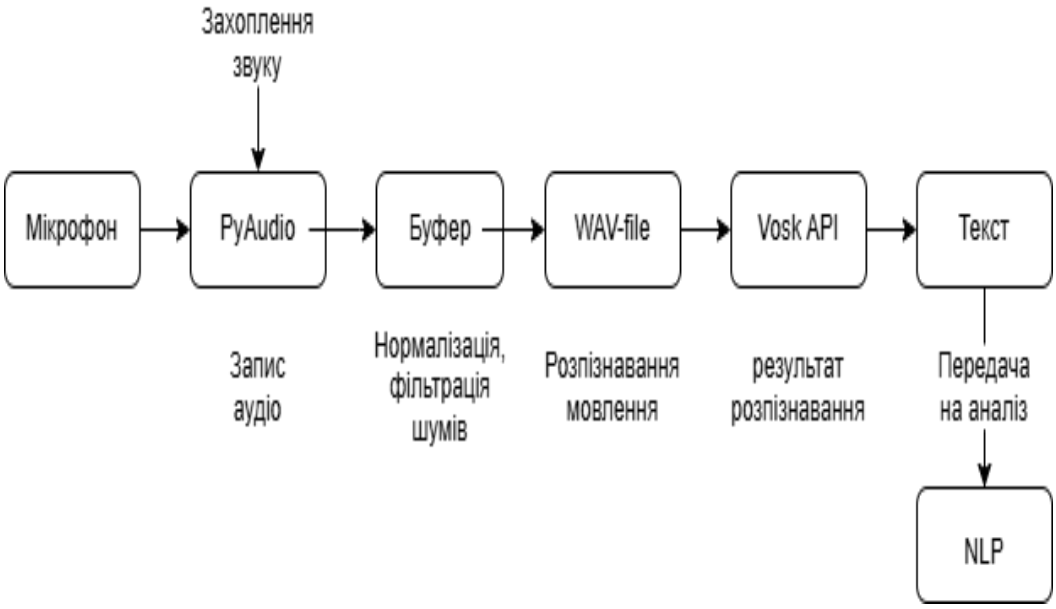

Рисунок 2.3 — Технологічна послідовність обробки мовлення у програмно-апаратному комплексі

Після формування навчальної відповіді або підказки система озвучує її користувачу через модуль синтезу мовлення. Використовується бібліотека pyttsx3, яка працює офлайн і підтримує українські голоси. Модуль перетворює текст відповіді в аудіофайл, який миттєво відтворюється через динаміки. При цьому користувач може обрати швидкість і тембр озвучення.

Інтерфейс користувача побудований у вигляді простого десктопного застосунку на основі бібліотеки Tkinter або PyQt6. Основні елементи — кнопки “Почати запис”, “Перевірити”, “Прослухати відповідь”, а також поле для виведення розпізнаного тексту та коментарів системи. Інтерфейс має можливість перемикання мови (українська / англійська), що робить його зручним для іноземних користувачів.

Архітектура системи має однорівневу структуру, де всі модулі взаємодіють через внутрішні методи обміну даними. Такий підхід спрощує

тестування і дозволяє виконувати налагодження кожного компонента окремо. Модульна побудова забезпечує стабільність, незалежність компонентів і можливість подальшого розширення — наприклад, інтеграції веб-версії або мобільного клієнта.

Таким чином, проєктована архітектура забезпечує повний цикл оброблення мовлення — від введення голосу до отримання навчального результату — і є оптимальним рішенням для створення автономного інтелектуального комплексу для вивчення української мови.

2.3 Моделювання процесу перетворення мовлення користувача на текстовий фрагмент

Процес перетворення мовлення користувача на текст є одним з ключових етапів роботи програмно-апаратного комплексу. Саме він забезпечує взаємодію між користувачем і системою, переводячи звукову інформацію у текстову форму, придатну для подальшого лінгвістичного аналізу.

Моделювання цього процесу дозволяє формалізувати послідовність дій, визначити оптимальні параметри обробки сигналу та оцінити ефективність алгоритмів розпізнавання українського мовлення.

Розроблена модель базується на послідовному проходженні шести основних етапів: захоплення, оцифрування, попередня обробка, збереження, розпізнавання та текстова інтерпретація. Така структура дає змогу забезпечити безперервний потік даних і стабільну роботу системи в реальному часі.

На першому етапі здійснюється захоплення голосового сигналу за допомогою мікрофона. Аналоговий звуковий потік перетворюється у цифровий сигнал із фіксованими параметрами (частота дискретизації 16 кГц, 16 біт, моно), що забезпечує оптимальне співвідношення між якістю та швидкодією. Для реалізації цього процесу використовується бібліотека PyAudio, яка дозволяє записувати звук у реальному часі, контролювати буфер

обробки та визначати активність голосу (Voice Activity Detection).

Далі відбувається попередня обробка сигналу, що передбачає фільтрацію шумів, нормалізацію амплітуди та видалення коротких пауз. Ці дії зменшують вплив зовнішніх перешкод і покращують розбірливість мовлення. Для моделювання цього етапу використовуються стандартні методи цифрової обробки сигналів — фільтрація ковзним середнім, спектральне згладжування та нормалізація енергії звукової хвилі. У результаті формується стабілізований сигнал, який зберігається у форматі MP3, сумісному з більшістю систем розпізнавання мовлення.

Третій етап — розпізнавання мовлення, що виконується модулем Vosk API. Його робота базується на використанні акустичної та мовної моделей, об’єднаних у статистичну схему. Акустична модель визначає відповідність між звуковими характеристиками сигналу та фонемами української мови, а мовна — перевіряє граматичну послідовність слів. Таким чином, на виході формується найбільш ймовірний варіант тексту, який відповідає вимовленому реченню.

Математична модель такого процесу описується формулою (2.1).

$$W = \arg\max\_W \left(P(X|W) \times P(W) \right) / P(X), \quad (2.1)$$

де X — набір акустичних ознак;

W — послідовність слів;

P(X|W) — ймовірність появи сигналу для даної послідовності;

P(W) — імовірність самої мовної конструкції.

Розпізнавання здійснюється у реальному часі, із середньою затримкою не більше 0,5 секунди між завершенням фрази та появою тексту.

На етапі сегментації сигналу мовлення ділиться на короткі часові інтервали (фрейми) тривалістю від 20 до 25 мс. Для кожного фрейму обчислюються мел-частотні кепстральні коефіцієнти (MFCC), які слугують

числовим представленням спектральних характеристик звуку. Ці ознаки передаються до нейронної мережі, яка визначає ймовірність належності кожного фрагмента до певної фонеми. Далі результати об’єднуються в суцільну послідовність, що утворює текстову фразу.

Після розпізнавання система переходить до етапу текстової інтерпретації, на якому виконується базова перевірка коректності результату. Слова, що не відповідають правилам української мови, відмічаються для подальшої перевірки у модулі NLP. У випадку низької впевненості (менше 85%) користувачеві може бути запропоновано повторити фразу. Це підвищує надійність системи без втрати природності взаємодії.

Для оцінювання точності моделі застосовується коефіцієнт Word Accuracy (A), що визначається за формулою (2.2):

$$A = \frac{N-(S+D+I)}{N} \times 100\%, \quad (2.2)$$

де N — загальна кількість слів;

S — кількість замінених;

D — пропущених;

I — вставлених.

У процесі тестування середній показник точності для українських текстів становить від 92 до 95%, що є прийнятним рівнем для навчальних систем.

Особливістю моделі є можливість адаптації до користувача. Після кількох сеансів взаємодії система автоматично оновлює параметри фільтрації шумів, нормалізації та гучності відповідно до індивідуальних характеристик голосу. Такий механізм дозволяє покращити якість розпізнавання для конкретного користувача, особливо в разі повторюваних вправ або тренувальних діалогів.

Отриманий текстовий результат передається до модуля аналізу (NLP),

де проводиться граматичний розбір і визначається рівень володіння мовою. Таким чином, змодельований процес охоплює повний цикл перетворення мовлення — від захоплення звуку до формування осмисленого тексту, придатного для подальшого навчального використання.

Проведене моделювання підтвердило, що поєднання PyAudio для збору сигналу та Vosk API для розпізнавання забезпечує необхідну точність, стабільність і автономність системи. Модель ефективно працює без підключення до Інтернету, що робить її придатною для навчальних аудиторій, де важлива локальна обробка даних та швидка реакція.

Завдяки оптимальній послідовності етапів та використанню сучасних алгоритмів обробки мовлення, розроблений комплекс здатний точно відтворювати усне мовлення користувача у текстовому форматі, забезпечуючи основу для інтелектуального аналізу й формування персоналізованих навчальних завдань.

2.4 Моделювання методу аналізу граматики та генерації завдань

Алгоритм аналізу граматики та генерації навчальних завдань є ключовою інтелектуальною складовою програмно-апаратного комплексу. Його завдання полягає у тому, щоб на основі розпізнаного тексту виконати автоматичну перевірку граматичної правильності, визначити рівень володіння мовою користувачем і сформувати відповідні навчальні вправи для подальшого вдосконалення мовних навичок.

Розроблений алгоритм функціонує у два послідовні етапи — аналіз граматики із застосуванням підходу NLP та генерація навчальних завдань за допомогою AI. Перший відповідає за обробку тексту, а другий — за створення адаптивних навчальних матеріалів на основі результатів аналізу.

На етапі аналізу граматики вхідним даним є текст, отриманий після розпізнавання мовлення модулем ASR. Для лінгвістичної обробки використовується бібліотека LanguageTool, адаптована під українську мову.

Алгоритм здійснює морфологічний розбір тексту, визначає частини мови, узгодження чисел, родів і відмінків, а також перевіряє синтаксичні зв’язки між словами.

Процес моделюється у вигляді ланцюга послідовних дій: токенізація → морфологічний аналіз → перевірка правил → формування звіту про помилки. Кожен етап реалізується окремим підмодулем, що забезпечує модульність і можливість розширення функціоналу.

Для оцінювання граматичної правильності використовується коефіцієнт G (2.3), який визначається як співвідношення правильно вжитих конструкцій n_p до загальної кількості перевірених фрагментів n:

$$G = (n_p / n) \times 100\% \quad (2.3)$$

Якщо показник падає нижче певного порогу (наприклад, 80%), система визначає, що користувачу необхідне додаткове тренування конкретної теми (відмінювання, узгодження, порядок слів тощо).

Результати граматичного аналізу передаються до модуля генерації навчальних завдань. На цьому етапі система використовує принцип адаптивного навчання: кожне завдання формується з урахуванням попередніх помилок користувача. Для цього створюється профіль користувача — набір параметрів, які зберігаються у базі даних і містять інформацію про типи помилок, час виконання, рівень складності та кількість повторень.

Модель генерації завдань використовує алгоритм класифікації помилок за типами:

— орфографічні (пропуски, зайві символи, неправильні літери);

— морфологічні (невірне відмінювання, рід, число);

— синтаксичні (порушення порядку слів, відсутність узгодження);

— пунктуаційні (помилки у вживанні розділових знаків).

На основі класифікації система обирає шаблон завдання з бази.

Наприклад, якщо виявлено помилки у вживанні відмінків, формується завдання типу «виберіть правильну форму слова»; якщо порушено порядок слів, генерується вправа «відновіть правильну послідовність».

Для побудови завдань застосовується комбінований метод: використання шаблонів речень і автоматична генерація варіантів відповіді за допомогою алгоритмів машинного навчання. Це дозволяє створювати нескінченну кількість унікальних завдань без ручного втручання викладача.

У процесі моделювання передбачено зворотний зв’язок: після виконання вправи користувач отримує оцінку та коротке пояснення. Дані про успішність зберігаються у базі, а система оновлює рівень користувача. Таким чином, навчальний процес набуває ознак адаптивності — чим вищий показник виконання, тим складніші завдання генерує система.

Для формального опису процесу генерації навчальних завдань введемо функцію (2.4):

$$T = f(G, L, P), \tag{2.4}$$

де G — рівень граматичної коректності;

L — поточний рівень володіння мовою;

P — набір попередніх помилок.

Функція f визначає тип завдання, кількість прикладів і рівень складності, що адаптуються до користувача.

Оцінка ефективності алгоритму визначається коефіцієнтом успішності (2.5).

$$S = (nc / n_t) \times 100\%, \tag{2.5}$$

де nc — кількість правильно виконаних завдань;

n_t — загальна кількість наданих завдань.

Якщо S перевищує 85%, система переходить на складніший рівень вправ, у протилежному випадку — генерує додаткові тренувальні завдання на ті ж теми.

Розроблена модель поєднує класичні методи обробки природної мови з елементами штучного інтелекту. Вона дозволяє не лише виявляти граматичні помилки, а й перетворювати їх на навчальний матеріал, формуючи індивідуальний шлях навчання. Це створює основу для повністю автономної системи, здатної не просто перевіряти, а активно навчати українській мові на основі персоналізованого підходу.

2.5 Проєктування підсистеми синтезу мовлення та інтерфейсу

Підсистема синтезу мовлення та користувацького інтерфейсу є завершальним етапом взаємодії користувача з програмно-апаратним комплексом. Її мета — забезпечити зворотний зв’язок системи у зручній для сприйняття формі: візуальній (текст, елементи інтерфейсу) та аудіальній (озвучення результату). Ці два компоненти працюють узгоджено, утворюючи цілісний навчальний простір.

Підсистема синтезу мовлення. Основне завдання підсистеми синтезу мовлення полягає в озвученні текстових результатів, згенерованих системою. Це може бути як правильна відповідь, так і пояснення до виконаного завдання або граматична підказка.

Для реалізації синтезу використовується бібліотека pyttsx3, що працює офлайн та підтримує українську мову. Її головною перевагою є автономність — синтез не потребує підключення до Інтернету, що відповідає вимогам дипломного проєкту.

Архітектура модуля передбачає такі основні компоненти:

— блок генерації тексту отримує вхідний текст від попередніх модулів (наприклад, результат аналізу чи навчального завдання);

— мовний процесор перетворює текст у фонетичну форму,

враховуючи правила наголосу та інтонації української мови;

— аудіогенератор (pyttsx3 Engine) створює звуковий потік на основі параметрів голосу (тон, швидкість, гучність);

— блок відтворення передає сформований звуковий сигнал на динаміки пристрою користувача.

Алгоритм роботи підсистеми можна представити як послідовність:

Отримання тексту → Фонетичний розбір → Генерація звуку → Відтворення мовлення.

Для підвищення природності синтезу передбачено налаштування параметрів голосу:

— rate (швидкість) від 150 до 200 слів за хвилину;

— volume (гучність) від 0.8 до 1.0;

— voice_id український чоловічий або жіночий голос залежно від налаштувань користувача.

Синтезатор також може озвучувати навчальні завдання або коментарі викладача, що робить процес навчання більш інтерактивним. У майбутньому передбачено можливість інтеграції з нейронним синтезом мовлення (наприклад, VITS або Coqui TTS) для покращення якості звуку та інтонаційної природності.

Підсистема користувацького інтерфейсу (UI) забезпечує взаємодію між користувачем і системою. Вона має інтуїтивно зрозумілу структуру та забезпечує швидкий доступ до всіх функцій комплексу.

Для реалізації обрано бібліотеку Tkinter, яка входить до стандартного пакету Python та дозволяє створити легкий, кросплатформний графічний інтерфейс без додаткових витрат.

Інтерфейс включає такі основні елементи:

— головне вікно програми, яке містить панель керування та навчальну область.

— кнопку активації мікрофона, що запускає процес запису голосу

користувача.

— текстове поле відображення, у якому показується розпізнаний текст та підказки системи.

— кнопку “Прослухати результат”, яка активує підсистему синтезу мовлення.

— меню налаштувань, де можна обрати рівень складності, мову інтерфейсу (українська або англійська) та параметри голосу.

Інтерфейс моделюється за принципом мінімалізму та інклюзивності, що особливо важливо для іноземних користувачів, які тільки починають вивчати українську мову. Всі повідомлення виводяться двома мовами — українською та мовою користувача, обраною при першому запуску системи. Це забезпечує ефект подвійного навчання: користувач бачить український текст і водночас розуміє його значення.

У структурі інтерфейсу передбачено також інформаційний індикатор стану системи, який сигналізує про активність модулів (“Запис триває…”, “Розпізнавання завершено”, “Результат озвучено”). Це дозволяє користувачу чітко розуміти, на якому етапі перебуває процес обробки мовлення.

Інтеграція двох підсистем. Синтез мовлення і користувацький інтерфейс функціонують як єдина підсистема зворотного зв’язку. Після розпізнавання тексту ASR-модулем, інтерфейс отримує результат, відображає його на екрані та за потреби передає до синтезатора для озвучення.

Ця взаємодія відбувається через проміжний модуль обміну даними, який забезпечує синхронність між текстовими та звуковими подіями. Такий підхід дозволяє уникнути затримок і забезпечити плавний користувацький досвід.

Функціональну схему роботи підсистеми наведено на рис. 2.4, де показано послідовність взаємодії між інтерфейсом та основними модулями.

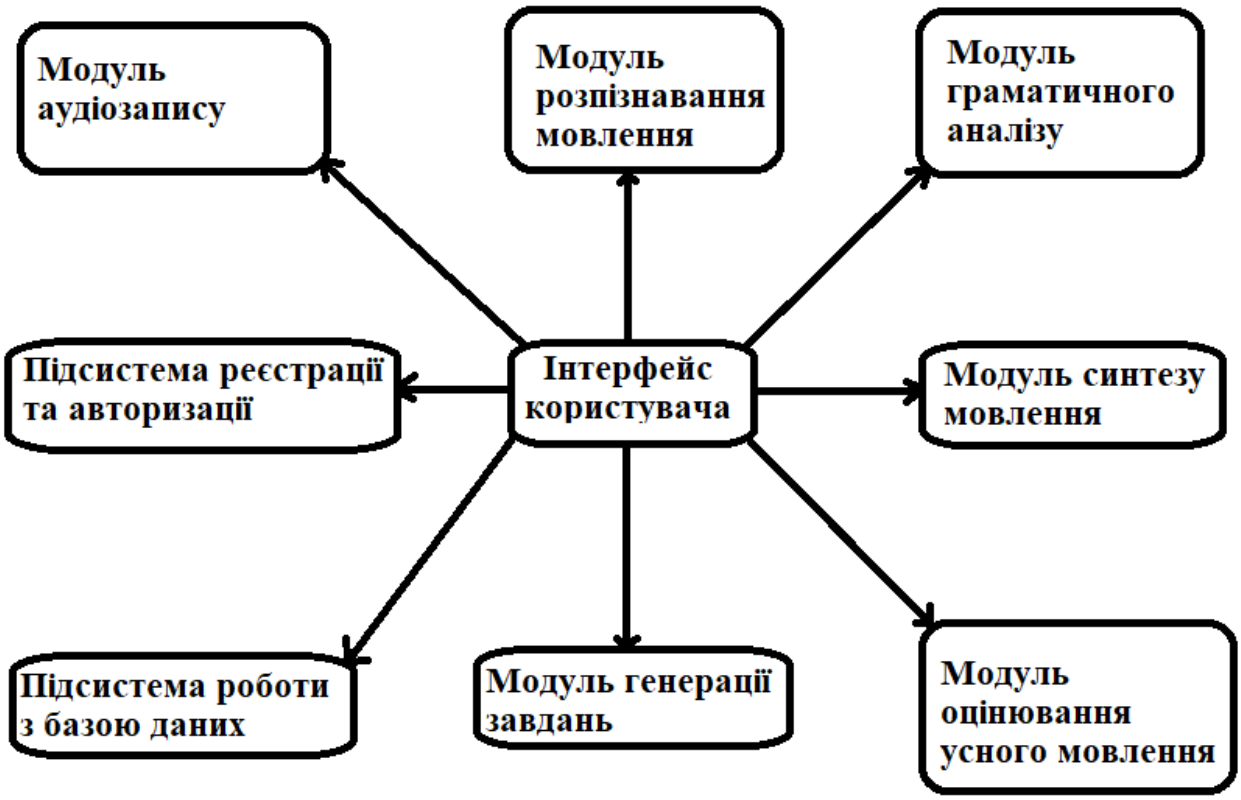

Рисунок 2.4 — Схема взаємодії користувацького інтерфейсу з модулями

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ

3.1 Обґрунтування вибору підходів, методів та засобів розроблення

Аналіз функціональних та експлуатаційних вимог визначає ключові параметри, яким має відповідати програмно-апаратний комплекс. Основним завданням системи є підтримка повного циклу роботи з усним мовленням користувача: запис, розпізнавання, аналіз та формування навчальних завдань. Тому до системи висувається вимога забезпечення стабільного аудіозапису, коректної обробки мовленнєвого сигналу та можливості подальшого аналізу отриманого тексту.

Важливою функціональною вимогою є точність розпізнавання української мови, оскільки від якості транскрипції залежить коректність граматичного аналізу та адаптивність навчального процесу. Система повинна реагувати на індивідуальні особливості вимови та забезпечувати однаково прийнятні результати в типових умовах використання.

До експлуатаційних вимог належить необхідність забезпечення стабільної роботи комплексу в умовах обмежених ресурсів персонального комп’ютера та можливих фонових навантажень операційної системи. Це вимагає оптимізації обчислювальних операцій, мінімізації затримок під час обробки мовлення та раціонального використання оперативної пам’яті. Інтерфейс комплексу повинен залишатися простим, інтуїтивним і достатньо швидким у роботі, щоб забезпечити комфортну взаємодію користувачів різного рівня підготовки та не створювати додаткових бар’єрів у процесі навчання.

Також важливо забезпечити можливість автономної роботи системи: основні функції, включаючи збереження даних, відтворення підказок, оброблення аудіо та взаємодію з користувачем, мають виконуватися локально. Це визначає подальший вибір архітектурних рішень і технологій, що використовуються під час розроблення комплексу.

Вибір модульної архітектури зумовлюється тим, що комплекс об’єднує різнорідні за своїм призначенням компоненти, які виконують окремі етапи обробки мовлення та організації навчального процесу. Робота з аудіосигналами, розпізнавання мовлення, граматичний аналіз, генерація навчальних завдань, синтез мовлення, формування статистики та забезпечення взаємодії з апаратною платформою — усе це функції, що мають різну природу й різний рівень обчислювальної складності. Їх інтеграція у монолітну систему ускладнила б розроблення, масштабування та супровід проєкту.

Модульна архітектура дозволяє чітко відокремити логіку кожного функціонального компонента, полегшуючи локалізацію змін та спрощуючи тестування. Окремі модулі можуть вдосконалюватися або замінюватися незалежно від інших, що особливо актуально у випадку з моделями штучного інтелекту, які регулярно оновлюються та потребують адаптації під нові вимоги. Це забезпечує гнучкість системи та можливість поступового розвитку без необхідності повної перебудови комплексу.

Крім того, модульна архітектура дає змогу ефективно розподіляти обчислювальне навантаження між різними частинами системи. Операції, пов’язані з обробкою аудіо або взаємодією з інтерфейсом, можуть виконуватися локально й не потребують значних ресурсів, тоді як складні обчислення, пов’язані з роботою мовних моделей, можуть передаватися на зовнішні сервіси. Такий підхід забезпечує оптимальне використання доступних ресурсів і підтримує стабільність роботи комплексу навіть за високої інтенсивності запитів.

Таким чином, модульна архітектура створює структуровану та передбачувану основу для подальшого проєктування, дозволяючи узгоджено інтегрувати всі компоненти комплексу та забезпечувати їх ефективну взаємодію у процесі роботи.

Вибір підходу до життєвого циклу розроблення визначається необхідністю послідовно створювати, перевіряти та вдосконалювати окремі

компоненти комплексу, які мають різний ступінь складності та різну залежність від апаратної і програмної частин. Ураховуючи модульну структуру системи та наявність компонентів, що потребують експериментальної перевірки (зокрема модулів розпізнавання мовлення, граматичного аналізу та генерації навчальних завдань), доцільним є застосування ітеративно-інкрементного підходу.

Такий підхід дозволяє реалізовувати систему поетапно, починаючи з базових можливостей — запису аудіо, відображення інтерфейсу та роботи локальної бази даних — і поступово розширюючи функціонал шляхом інтеграції більш складних компонентів. Кожна ітерація передбачає створення працездатної частини комплексу, що дає можливість оперативно виявляти недоліки та оцінювати взаємодію між модулями.

Ітеративна модель полегшує адаптацію до змін, що неминучі під час роботи з мовними технологіями, адже параметри моделей штучного інтелекту, їх точність або вимоги до ресурсів можуть змінюватися в процесі розроблення. Це дозволяє вносити корективи без порушення загальної структури комплексу.

Крім того, ітеративний процес є оптимальним для систем, у яких важливо оцінювати продуктивність і взаємодію кожного модуля окремо. Поетапне впровадження дає змогу своєчасно виявляти вузькі місця, коригувати обчислювальне навантаження та поступово оптимізувати реалізацію, зберігаючи стабільність роботи комплексу на всіх етапах його розвитку.

Розподіл функцій між програмною та апаратною частиною комплексу визначається необхідністю забезпечити стабільну, передбачувану роботу системи та водночас гарантувати високу якість обробки мовлення. Апаратна частина відповідає за фізичну взаємодію з користувачем: захоплення аудіосигналу, відтворення звуку, підтримання роботи мікрофона, динаміків та інших переферійних пристроїв, а також виконання базових обчислень,

необхідних для функціонування програми. Чітке розмежування між обов’язками апаратного та програмного середовищ дозволяє мінімізувати затримки під час запису й відтворення аудіо, що є критично важливим для коректного функціонування модулів розпізнавання та синтезу мовлення.

Програмна частина зосереджує основну логіку обробки даних: виконання граматичного аналізу, формування навчальних завдань, управління статистикою, роботу локальної бази даних і взаємодію з мовними моделями. Значна частина обчислювально складних операцій, таких як обробка мовлення моделями штучного інтелекту, може виконуватися із залученням зовнішніх сервісів, що дозволяє компенсувати обмеження продуктивності апаратної платформи.

Такий підхід забезпечує оптимальний баланс між локальною автономністю та можливістю використання сучасних мовних технологій. Апаратна частина виконує функції забезпечення стабільного середовища роботи та взаємодії з користувачем, тоді як програмна частина реалізує інтелектуальні можливості комплексу. У результаті досягається ефективне використання ресурсів, а структура системи залишається гнучкою для подальшого розширення та інтеграції нових модулів.

Вибір мови програмування є ключовим рішенням, оскільки від нього залежить зручність реалізації окремих модулів комплексу, можливість інтеграції з бібліотеками обробки мовлення та відповідність технічним вимогам системи. Python був обраний завдяки поєднанню простоти синтаксису, широкої екосистеми готових рішень і підтримки необхідних технологій для роботи з аудіо, мовними моделями та графічним інтерфейсом. Його модульність та наявність численних інструментів для роботи з даними, мережею та зовнішніми AI-сервісами дозволяють ефективно реалізувати як базові функції, так і складні алгоритмічні компоненти комплексу.

Однією з головних причин вибору Python є наявність стабільних бібліотек для запису та обробки звуку, а також підтримка сучасних моделей

розпізнавання та синтезу мовлення, що значно спрощує реалізацію ключових функцій комплексу. Важливо й те, що Python природно інтегрується з API мовних моделей та сервісами штучного інтелекту, що дозволяє реалізувати модулі граматичного аналізу та генерації завдань без необхідності розроблення власних алгоритмів низького рівня.

Python також є оптимальним вибором для локального розгортання системи, оскільки забезпечує достатню продуктивність для більшості операцій комплексу та підтримується різними операційними системами без додаткових налаштувань. Завдяки цьому вдається уникнути зайвих проміжних надбудов і спростити процес інсталяції програмного забезпечення. Поєднання стабільності, універсальності та широкої екосистеми робить Python збалансованим середовищем для швидкої розробки, тестування та подальшого масштабування можливостей навчального комплексу.

Вибір засобів для побудови інтерфейсу користувача є важливою складовою проєктування комплексу, адже саме інтерфейс визначає зручність взаємодії користувача з усіма функціональними модулями. Навчальна система повинна бути інтуїтивною, доступною та здатною наочно відображати результати роботи модулів розпізнавання, аналізу та генерації завдань. З огляду на ці вимоги було обрано PyQt6 — інструментарій, що надає широкі можливості для побудови сучасних графічних інтерфейсів.

PyQt6 вирізняється гнучкістю у компонуванні елементів, підтримує різні стилі оформлення, дозволяє створювати багатовкладкові та багатовіконні структури, що є важливим для організації окремих робочих областей комплексу: вікна запису мовлення, області аналізу, перегляду статистики та налаштувань. Використання сигналів і слотів забезпечує зручне та передбачуване керування подіями, що спрощує інтеграцію інтерфейсу з модулями обробки аудіо, розпізнавання мовлення та генерації навчальних матеріалів.

Окремою перевагою PyQt6 є стабільна робота в середовищі Windows, що

дозволяє використовувати його без необхідності додаткових налаштувань або встановлення сторонніх залежностей. Завдяки цьому застосунок залишається продуктивним та передбачувано працює навіть під час активної взаємодії з іншими модулями системи. Важливим аспектом також є наявність розвиненої спільноти та якісної офіційної документації, що значно спрощує процес розроблення, налагодження та подальшої підтримки програмного забезпечення.

Таким чином, PyQt6 є оптимальним вибором для створення інтерфейсу навчального комплексу, оскільки поєднує функціональність, стабільність, продуктивність та зручність у реалізації, забезпечуючи повноцінну взаємодію користувача з усіма компонентами системи.

Вибір технологій для роботи з аудіо та мовлення зумовлений потребою забезпечити стабільний запис голосу користувача, точне розпізнавання українського мовлення та якісний синтез голосових повідомлень. Для реалізації модуля аудіозапису використано бібліотеки sounddevice та wave, які коректно працюють із більшістю стандартних мікрофонів і забезпечують достатню якість сигналу для подальшої мовної обробки. Їхньою перевагою є низьке навантаження на систему, швидка обробка аудіофреймів та проста інтеграція з графічним інтерфейсом PyQt6, що дозволяє реалізувати запис мовлення без помітних затримок і втручання у роботу інших модулів комплексу.

Для автоматичного розпізнавання мовлення застосовано модель Whisper, що демонструє високу точність для української мови та здатна коректно працювати з різними особливостями вимови. Whisper підтримує обробку аудіофайлів у стандартних форматах, а також має зручний інтерфейс API, що дозволяє гнучко вбудувати її в навчальний цикл системи. Завдяки цьому модуль ASR отримує текст, придатний для подальшого граматичного аналізу.

Синтез мовлення в комплексі реалізовано з використанням бібліотеки

gTTS (Google Text-to-Speech), яка забезпечує формування озвученого аудіо українською мовою з довільного текстового фрагмента. Такий підхід дозволяє системі відтворювати голосові підказки, пояснення результатів та інші елементи навчального процесу, що підвищує інтерактивність і зручність використання комплексу. Завдяки простому механізму генерації аудіофайлів у форматі MP3 модуль синтезу легко інтегрується з іншими компонентами системи та не потребує складної конфігурації.

Усі обрані технології працюють із врахуванням обмежених ресурсів апаратної платформи та забезпечують безперервність мовної взаємодії, необхідної для реалізації повного навчального процесу.

Вибір моделей штучного інтелекту для граматичного аналізу та генерації навчальних завдань визначений тим, що система має працювати з українським мовленням, коректно інтерпретувати текст користувача та формувати індивідуальні вправи на основі його помилок. Для цих завдань використовується Google Gemini 2.0 Flash, яка демонструє високу точність у контекстному розборі українських речень і дозволяє проводити гнучкий граматичний аналіз у реальному часі. Модель добре справляється з визначенням частин мови, ролей слів у реченні, виявленням граматичних та орфографічних помилок, а також поясненням причин їх появи у формі, зрозумілій користувачу.

Gemini 2.0 Flash обрана також завдяки своїй здатності адаптувати навчальний матеріал під кожного користувача. На основі історії введень, виявлених помилок та рівня підготовки модель може формувати різні типи навчальних завдань: вправи на вибір правильної форми слова, побудову речень, визначення граматичних категорій чи аналіз структури висловлювання. Такий підхід забезпечує персоналізований навчальний процес без необхідності жорстко фіксувати набір статичних вправ.

Перевагою Gemini є також її продуктивність і швидкість роботи, що дозволяє інтегрувати її в інтерфейс навчального комплексу без помітних

затримок. Модель має простий API, який легко поєднується з Python-застосунком, що спрощує формування запитів і обробку відповідей. Важливо й те, що Gemini 2.0 Flash підтримує роботу з короткими та середніми за обсягом текстовими фрагментами, що повністю відповідає формату системи, де взаємодія відбувається невеликими голосовими блоками.

Таким чином, використання Gemini 2.0 Flash забезпечує комплексне виконання двох ключових завдань — граматичного аналізу та генерації адаптивних навчальних матеріалів — і дозволяє поєднати точність мовної обробки з гнучкістю персонального навчання.

Вибір локальної системи керування базами даних має ключове значення для стабільності та організації роботи навчального комплексу. Оскільки система повинна функціонувати автономно та не залежати від постійного доступу до мережі, сховище даних повинно бути простим, надійним і невимогливим до ресурсів. З огляду на ці вимоги було обрано SQLite — вбудовану СУБД, що не потребує серверної інфраструктури та дозволяє швидко зберігати й отримувати дані користувача локально, мінімізуючи залежність від зовнішніх сервісів.

SQLite стабільно працює з обсягами інформації, характерними для навчальних систем: збереженням результатів виконаних вправ, переліком виявлених помилок, рівнем складності, статистикою прогресу та персональними налаштуваннями. Важливо, що кількість операцій зчитування та запису є невеликою, тому навіть на апаратній платформі з обмеженими ресурсами база даних функціонує без затримок і не створює додаткового навантаження на процесор.

Ще однією перевагою є простота інтеграції SQLite з Python. Бібліотека sqlite3, що входить до стандартної комплектації мови, дозволяє працювати з базою даних без встановлення додаткових компонентів або серверного середовища. Такий підхід мінімізує ризики несумісності, спрощує розгортання системи на будь-якому комп’ютері та робить подальшу підтримку комплексу

більш передбачуваною й стабільною.

Оскільки база зберігається у вигляді одного файлу, стає можливою швидка міграція даних, їх резервне копіювання або перенесення на інший пристрій. Для навчального комплексу, який має підтримувати відновлення статистики після оновлень або технічних змін, це є суттєвою перевагою.

Враховуючи поєднання продуктивності, стабільності, простоти інтеграції та відповідності вимогам автономної роботи, SQLite є найдоцільнішим рішенням для організації локального зберігання даних у межах програмно-апаратного комплексу.

Підхід до інтеграції модулів у єдину систему визначається необхідністю забезпечити узгоджену роботу всіх компонентів комплексу, починаючи від аудіозапису та розпізнавання мовлення і завершуючи формуванням навчальних завдань, синтезом підказок та збереженням результатів у базі даних. Оскільки система складається з функціонально різнорідних модулів — аудіообробки, мовних моделей, інтерфейсу, локального сховища та апаратної частини — інтеграція повинна гарантувати стабільність роботи й мінімізувати залежність між компонентами.

Для цього використовується модульний підхід зі строгим поділом відповідальностей: кожен модуль виконує власне завдання та взаємодіє з іншими через заздалегідь визначені інтерфейси. Наприклад, модуль аудіозапису передає результати розпізнавання лише через виклик функції або API, не потребуючи інформації про роботу граматичного аналізатора чи генератора завдань. Це дозволяє уникнути надмірних зв’язків і спрощує оновлення окремих компонентів.

Інтеграція мовних моделей, таких як Gemini 2.0 Flash, здійснюється через зовнішні API, що забезпечує чіткий формат запитів і відповідей. Такий підхід дозволяє ізолювати інтелектуальні функції від решти системи й водночас дає змогу адаптувати окремі модулі без зміни загальної структури комплексу. Локальна база даних SQLite інтегрується через уніфікований шар

доступу, який відповідає за збереження та отримання даних незалежно від логіки будь-якого іншого модуля.

Завдяки такому принципу інтеграції забезпечується масштабованість системи, можливість поступового розширення функціоналу та підтримання стабільної роботи всіх компонентів під час їх взаємодії. Модулі залишаються незалежними один від одного, що дозволяє оновлювати або замінювати будь-яку частину комплексу без ризику порушення роботи всієї програми. Така гнучка інфраструктура спрощує подальший розвиток системи й гарантує можливість адаптації до нових технологічних вимог.

3.2 Розроблення загальної структури програмної складової комплексу

Реалізація програмно-апаратного комплексу розпочинається з формування загальної структури застосунку та побудови графічного інтерфейсу, що забезпечує взаємодію користувача з основними функціональними модулями. Структура проєкту організована у вигляді каталогів, що групують вихідні файли за логічними компонентами системи: модулі роботи з даними, модулі штучного інтелекту, обробники аудіо, елементи графічного інтерфейсу та файли ініціалізації. Така організація коду дозволяє підтримувати читабельність та полегшує подальше внесення змін. Загальну структуру директорій і файлів системи наведено на рис. 3.1.

Інтерфейс застосунку побудовано на основі фреймворку PyQt6, який дає можливість створити багатовкладкове вікно з окремими робочими областями для запису мовлення, перегляду результатів аналізу, виконання навчальних завдань, роботи із синтезом мовлення та управління статистикою користувача. Головним елементом інтерфейсу є клас MainWindow, який успадковує QMainWindow і містить основні компоненти, зокрема меню, вкладки, службові панелі та інтегровані модулі взаємодії з користувачем.

При запуску програми ініціалізується база даних, викликається модуль авторизації, а після успішного входу відкривається головне вікно. Фрагмент

відповідного коду наведено нижче, що демонструє взаємодію між механізмом входу та побудовою інтерфейсу:

```
app = QApplication(sys.argv)
db = Database()
login_dialog = LoginDialog(db)
if login_dialog.exec() == 0:
    sys.exit()
current_user = login_dialog.logged_user
window = MainWindow(db, current_user)
window.show()
sys.exit(app.exec())
```

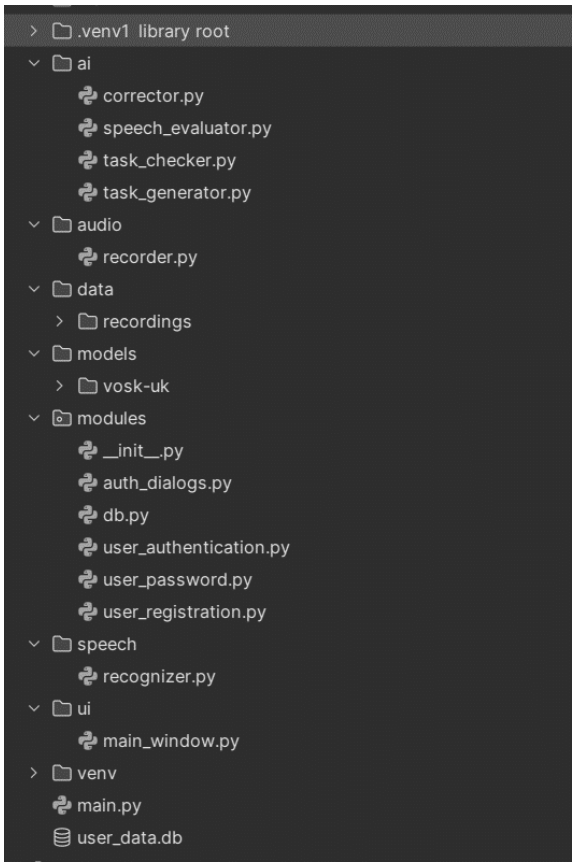

Рисунок 3.1 — Програмна реалізація структурної моделі програмного комплексу

Головне вікно складається з вкладок, кожна з яких реалізована окремим методом класу. Наприклад, функція створення вкладки синтезу мовлення визначає текстове поле введення, кнопки відтворення та збереження аудіо. Це

забезпечує модульність та відокремленість логіки кожної частини інтерфейсу. Загальний вигляд головного вікна програмно-апаратного комплексу наведено на рис. 3.2, де продемонстровано структуру інтерфейсу, основні вкладки, навігаційні елементи та панелі взаємодії з модулями системи.

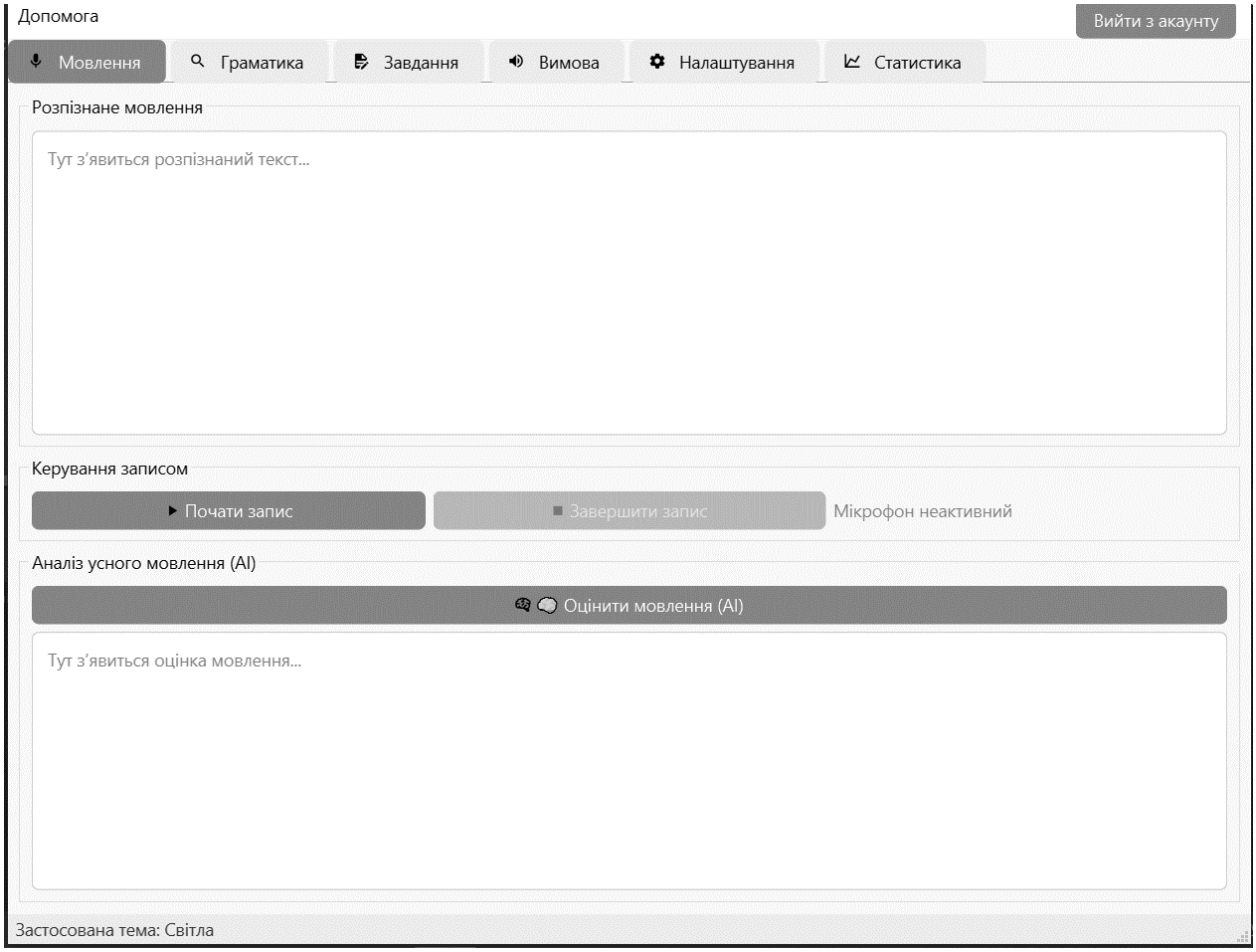

Рис. 3.2 — Головне вікно програмно-апаратного комплексу

Загальну схему взаємодії інтерфейсу з функціональними модулями подано на рис. 3.3. На схемі зображено, як вкладки головного вікна здійснюють виклики відповідних модулів: аудіозапису, транскрипції мовлення, аналізу тексту, генерації навчальних завдань і роботи з базою даних.

Графічний інтерфейс відіграє роль центрального вузла керування системою: він не виконує складних обчислень, але ініціює виклики до відповідних модулів і відображає результати їх роботи. Такий підхід дозволяє гнучко розширювати систему, додаючи нові можливості шляхом створення

нових вкладок або панелей без втручання в основну логіку програми.

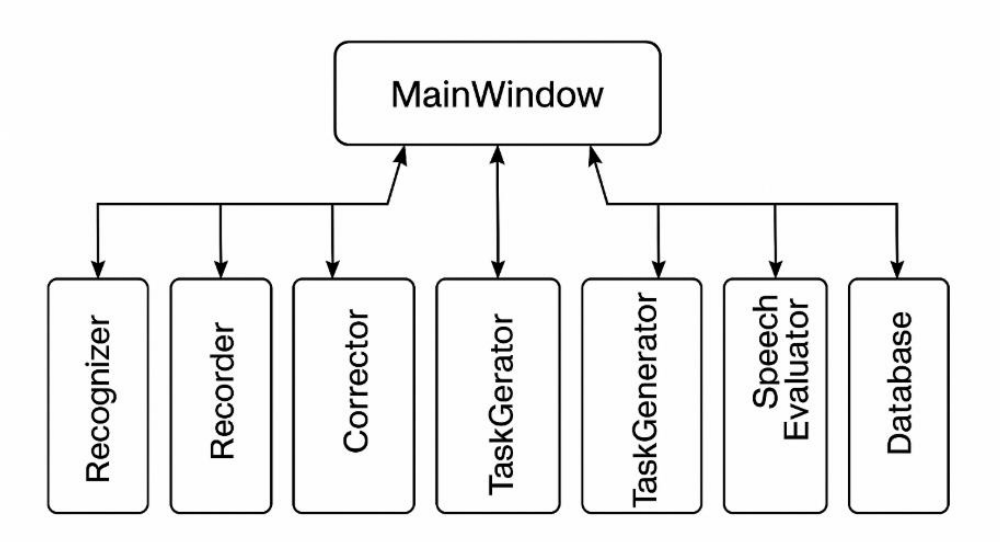

Рисунок 3.3 — Архітектура взаємодії графічного інтерфейсу з модулями системи

Таким чином, побудова інтерфейсу забезпечує цілісність і логічну організацію роботи користувача з програмою, створює основу для інтеграції окремих модулів навчального комплексу та визначає структуру подальших етапів програмної реалізації.

3.3 Модульне проєктування програмної складової комплексу

Підсистема реєстрації та авторизації забезпечує контроль доступу до програмно-апаратного комплексу, формування унікального профілю кожного користувача та збереження індивідуальної статистики навчання. Оскільки система призначена для персональної мовної роботи та накопичення аналітичних даних, надійне керування обліковими записами є важливою складовою загальної архітектури комплексу. Реалізація функціональності виконана у декількох логічних шарах: графічні діалогові вікна, модулі обробки даних та взаємодія з локальною базою SQLite.

Основою підсистеми виступає таблиця users у локальній базі даних, де

зберігаються логін, хешований пароль, час створення акаунту та дата останнього входу. Під час першого запуску програми база даних автоматично створює всі необхідні таблиці, що забезпечує повну автономність роботи комплексу та можливість його використання на будь-якому пристрої без додаткового налаштування з боку користувача. У момент реєстрації введений пароль проходить хешування із використанням випадкової солі, що гарантує надійний захист даних у разі доступу до файлу бази. Для цього використовується спеціальний допоміжний модуль, який інкапсулює операції генерації солі, формування хешу та перевірки пароля під час авторизації.

Графічна складова підсистеми реалізована у вигляді двох діалогових вікон — LoginDialog та RegisterDialog. Обидва вікна інтегровані у загальний інтерфейс PyQt6 і відповідають за введення та первинну валідацію даних. У разі помилкових або порожніх значень користувачу виводяться відповідні повідомлення, а поля підсвічуються стилями помилки для покращення взаємодії. Після успішного введення даних діалог викликає відповідні методи модулів UserRegistration або UserAuthentication. Система реєстрації перевіряє унікальність логіна та додає нового користувача до бази, а модуль авторизації звіряє хеш пароля з еталонним записом у таблиці users. У випадку успішної автентифікації система автоматично оновлює поле last_login та збільшує лічильник сесій користувача у таблиці статистики.

Наведений нижче фрагмент демонструє приклад використання хешування пароля перед збереженням у базі:

```
salt = generate_salt()
pwd_bytes = (salt + password).encode("utf-8")
pwd_hash = hashlib.sha256(pwd_bytes).hexdigest()
stored_value = f"{salt}${pwd_hash}"
```

Після успішної авторизації головне вікно застосунку отримує інформацію про поточного користувача, що дозволяє персоналізувати роботу інших модулів — зокрема облік статистики виконаних завдань та

відображення індивідуальних навчальних рекомендацій. Таким чином, підсистема реєстрації та авторизації не лише обмежує доступ до програми, але й формує фундамент для подальшого аналізу навчального прогресу, інтегруючись у загальний робочий цикл комплексу.

Модуль розпізнавання мовлення забезпечує перетворення аудіосигналу, отриманого від користувача, у текстовий формат, який надалі використовується для граматичного аналізу та генерації навчальних завдань. Реалізація модуля включає дві ключові складові: запис мовлення в реальному часі та подальшу обробку аудіофайлу за допомогою моделі розпізнавання української мови. Загальна схема взаємодії компонентів модуля показана на рис. 3.4, що демонструє послідовність переходу від захоплення аудіо до одержання готового тексту.

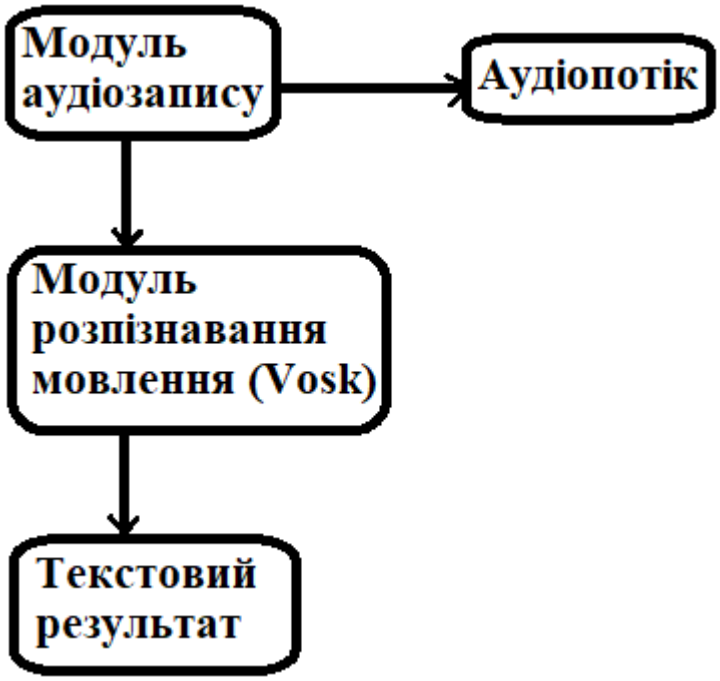

Рисунок 3.4 — Схема роботи модуля розпізнавання мовлення

Запис мовлення реалізовано на базі PyAudio у вигляді окремого потоку, що дає змогу не блокувати основний інтерфейс програми та забезпечує безперервне зчитування даних із мікрофона. Потік збирає окремі фрейми

аудіосигналу, які після завершення запису об’єднуються у суцільний буфер. Нижче подано фрагмент реалізації класу запису аудіо:

```
class AudioRecorder(QThread):
    finished = pyqtSignal(bytes)
    def __init__(self, rate=44100, chunk=1024, parent=None):
        super().__init__(parent)
        self.running = False
        self.rate = rate
        self.chunk = chunk
    def run(self):
        audio = pyaudio.PyAudio()
        stream = audio.open(
            format=pyaudio.paInt16,
            channels=1,
            rate=self.rate,
            input=True,
            frames_per_buffer=self.chunk
        )
        frames = []
        self.running = True
        while self.running:
            data = stream.read(self.chunk, exception_on_overflow=False)
            frames.append(data)
        stream.stop_stream()
        stream.close()
        audio.terminate()
        self.finished.emit(b"".join(frames))
```

Після завершення запису модуль розпізнавання отримує зібраний аудіопотік і зберігає його у WAV-файл, який передається моделі ASR. У системі використовується офлайн-модель Vosk українською мовою, що забезпечує роботу без підключення до мережі та гарантує стабільну швидкодію. Алгоритм обробки передбачає поділ аудіосигналу на блоки та поступову передачу їх моделі з подальшим формуванням текстового результату. Фрагмент коду реалізації наведено нижче:

```
class SpeechRecognizer:
    def __init__(self, model_path="models/vosk-uk"):
```

```
        self.model = Model(model_path)
    def transcribe(self, wav_file):
        wf = wave.open(wav_file, "rb")
        rec = KaldiRecognizer(self.model, wf.getframerate())
        text = ""
        while True:
            data = wf.readframes(4000)
            if len(data) == 0:
                break
            if rec.AcceptWaveform(data):
                text += json.loads(rec.Result())["text"] + " "
        text += json.loads(rec.FinalResult())["text"]
        return text.strip()
```

Модуль Vosk API повертає текст без розділових знаків та з мінімальною кількістю граматичних конструкцій, що є очікуваним для побудови більш точного подальшого аналізу. Усі службові елементи, які можуть виникати у процесі транскрипції, очищуються в межах функціональної логіки модуля. Отриманий текст передається до модуля граматичного аналізу, модуля оцінювання мовлення або використовується для формування навчальних завдань залежно від обраного режиму роботи комплексу.

Розпізнавання мовлення інтегрується в основне вікно програми через сигнально-слотовий механізм. Після того як модуль аудіозапису завершує роботу та передає буфер даних, головне вікно зберігає аудіо у тимчасовий файл і викликає функцію транскрипції. Такий підхід забезпечує чітке розмежування відповідальностей між компонентами та спрощує підтримку системи.

У цілому модуль розпізнавання мовлення працює як самостійний компонент, який може бути повторно використаний або вдосконалений без впливу на інші частини програмно-апаратного комплексу. Завдяки модульній структурі можлива подальша заміна моделі розпізнавання на більш продуктивну без зміни архітектури програми.

Модуль граматичного аналізу відповідає за обробку тексту, отриманого після розпізнавання мовлення, і виконує оцінювання правильності введених

користувачем фраз. Основною задачею цього компонента є формування структурованого звіту про виявлені граматичні, лексичні та стилістичні помилки, а також надання короткого узагальненого висновку, який відображається в інтерфейсі застосунку. Реалізація модуля базується на функції ai_analyze(), що інкапсулює логіку формування запиту до мовної моделі та подальше отримання результату у вигляді текстового опису. Послідовність обробки тексту модулем граматичного аналізу наведено на рис. 3.5.

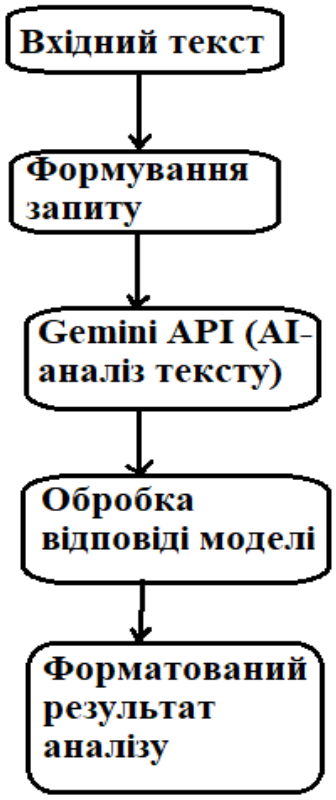

Рисунок 3.5 — Схема роботи модуля граматичного аналізу

Обробка виконується у кілька етапів. Спершу модуль формує промпт, у якому містяться інструкції щодо стилю та рівня суворості аналізу, а також чіткий формат відповіді. Далі цей промпт передається до моделі за допомогою методу generate_content(), після чого результат повертається у вигляді відформатованого тексту. Нижче наведено фрагмент коду, що демонструє реалізацію модуля:

```
from google.generativeai import GenerativeModel
import os
model = GenerativeModel("models/gemini-2.0-flash")
def ai_analyze(text: str) -> str:
    prompt = f"""
Ти — суворий екзаменатор…
(формування умов аналізу)
Текст:
\"\"\"{text}\"\"\".
"""
    response = model.generate_content(
        prompt,
        generation_config={"temperature": 0.3, "max_output_tokens": 500}
    )
    return response.text.strip()
```

Такий підхід дозволяє виконувати складний синтаксичний і морфологічний розбір без реалізації громіздких локальних алгоритмів. Модуль підтримує гнучку адаптацію промпту, тому його можна розширювати новими правилами або додатковими параметрами для зміни стилю аналізу без зміни основної структури програми.

Отримані результати передаються у графічний інтерфейс, де вони виводяться у відповідному полі на вкладці аналізу мовлення. Важливою частиною інтеграції є обробка можливих помилок, пов’язаних із некоректним введенням тексту або тимчасовою недоступністю моделі. У таких випадках модуль повертає безпечне fallback-повідомлення, що дозволяє зберегти стабільність роботи застосунку.

Завдяки інкапсуляції всієї логіки аналізу в одному модулі система залишається розширюваною: для майбутніх удосконалень достатньо змінити лише функцію формування запиту або додати нові режими роботи моделі. Це забезпечує незалежність модуля від інших компонентів, що спрощує подальше тестування, налагодження та інтеграцію в основне вікно застосунку.

Модуль генерації навчальних завдань забезпечує формування персоналізованих вправ. У межах програмно-апаратного комплексу цей

модуль викликається під час переходу користувача до тренувального режиму та працює на основі запиту з параметрами тип завдання та рівень складності.

Розроблена підсистема реалізована у вигляді окремого модуля task_generator.py, який взаємодіє зі ШІ-моделлю Gemini 2.0 Flash. Генерація відбувається повністю динамічно: модель створює нове завдання під кожен запит, перевіряє його на відповідність вимогам і повертає сформований текст разом із правильною відповіддю. Узагальнену структуру роботи модуля зображено на рис. 3.6.

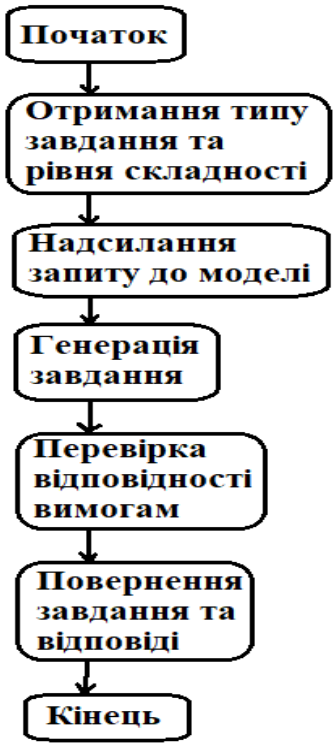

Рисунок 3.6 — Граф-схема роботи модуля генерації навчальних завдань

Основна функція модуля має вигляд:

```
def generate_task(task_type: str, difficulty: str) -> tuple[str, str]:
    prompt = f"""
    Створи одне тестове завдання українською мовою...
    Тип завдання: {task_type}
    Рівень складності: {difficulty}
    ... (правила побудови) ...
    """
```

```
response = model.generate_content(prompt, ...)
text = safe_text(response)
if not text or "ANSWER:" not in text:
    return generate_task(task_type, difficulty)
task, answer = text.split("ANSWER:")
return task.strip(), answer.strip().lower()
```

Функція отримує два параметри: обраний тип завдання (лексика, граматика, орфографія тощо) та рівень складності. На основі цих параметрів формується промпт, що передається мовній моделі. Сформоване завдання повертається у двох частинах: текст вправи та правильна відповідь, що використовується під час перевірки.

Окремо реалізована допоміжна функція safe_text(), яка уніфікує роботу з відповідями моделі та гарантує повернення коректного тексту навіть у випадку часткових або неструктурованих відповідей. Це мінімізує ризики помилкового розбору та забезпечує стабільність роботи комплексу під час багаторазової генерації завдань.

У межах комплексу модуль генерації інтегрується з вікном тренувальних завдань. Після отримання тексту вправи вона відображається у графічному інтерфейсі, а правильний варіант зберігається у внутрішній змінній. Після того як користувач обирає відповідь, інший модуль — перевірки — виконує оцінювання. Взаємодію цих підсистем буде продемонстровано в підрозділі інтеграції.

Загалом модуль забезпечує автоматичне отримання унікальних завдань, що дозволяє будувати індивідуальну траєкторію навчання та уникати повторів у змісті вправ.

Окремим етапом функціонування програмно-апаратного комплексу є аналіз усного мовлення користувача після його транскрибування. На відміну від граматичного аналізу, який працює зі структурою письмового тексту, модуль оцінювання мовлення зосереджується на характеристиках усного висловлювання: чіткості артикуляції, темпі, плавності, логічності висловлення

та природності мовного потоку. Такий підхід дозволяє комплексно оцінювати навички усної комунікації та формувати рекомендації для їх покращення.

Реалізація модуля розміщена у файлі speech_evaluator.py, де використовується мовна модель Gemini 2.0 Flash. Модуль працює у два етапи: отримання тексту від підсистеми ASR та передання цього тексту до мовної моделі зі спеціально сформованим промптом. Промпт містить чіткі правила — не аналізувати орфографію і пунктуацію транскрипту, оскільки вони можуть спотворювати реальну мовну поведінку, а зосереджуватися виключно на усному мовленні.

Базовий фрагмент реалізації подано нижче:

```
def evaluate_speech(text: str) -> str:
    prompt = f"""
Ти — експерт з усного українського мовлення...
(далі інструкція щодо критеріїв оцінювання)
\"\"\"{text}\"\"\".
"""
    response = model.generate_content(
        prompt,
        generation_config={"temperature": 0.4, "max_output_tokens": 400}
    )
    return response.text.strip()
```

У відповідь модель формує розгорнений аналіз, який включає оцінку чіткості вимови, плавності та темпу мовлення, використаного словникового запасу, логічної побудови висловлювання та наявності слів-паразитів. Завершується аналіз коротким переліком рекомендацій та виставленням інтегральної оцінки за десятибальною шкалою.

Даний модуль інтегрований у інтерфейс системи через окрему вкладку, де результат відображається у зручному форматі після завершення обробки аудіозапису. Оскільки він працює на основі тексту, що був отриманий після розпізнавання мовлення, його використання не створює додаткових вимог до апаратної частини та не впливає на швидкодію локальної системи.

Підсистема зберігання даних у програмно-апаратному комплексі реалізована на основі локальної СУБД SQLite, що забезпечує автономність роботи та мінімальні вимоги. Усі операції з даними інкапсульовані в окремому модулі db.py, який відповідає за створення структури таблиць, виконання транзакцій та обробку статистичних даних користувача. Структуру локальної бази даних, що використовується для зберігання інформації про користувачів та їх навчальну активність, зображено на рис. 3.7.

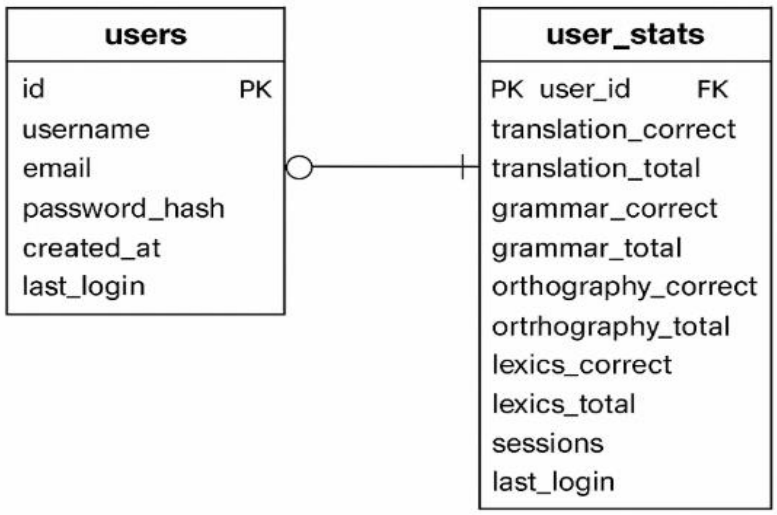

Рисунок 3.7 — Структура локальної бази даних комплексу

Під час ініціалізації програми створюється файл бази даних user_data.db, після чого запускається метод _create_tables(), який формує необхідні таблиці. Основними є таблиця користувачів та таблиця статистики, що містить показники успішності в різних мовних категоріях. Фрагмент ініціалізації бази подано у коді нижче:

```
self.conn = sqlite3.connect(self.db_path)
self.conn.row_factory = sqlite3.Row
cur.execute("""
CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT UNIQUE NOT NULL,
    email TEXT UNIQUE,
    password_hash TEXT NOT NULL,
```

```
    created_at TEXT NOT NULL,
    last_login TEXT
);
""")
```

Збереження статистики виконання вправ реалізовано через окрему таблицю user_stats, у якій для кожного користувача зберігається кількість правильних та загальних відповідей у категоріях “граматика”, “орфографія”, “лексика” та “переклад”. Структура таблиці визначена так, щоб забезпечити розширюваність і швидкий доступ до даних:

```
cur.execute("""
CREATE TABLE IF NOT EXISTS user_stats (
    user_id INTEGER PRIMARY KEY,
    translation_correct INTEGER DEFAULT 0,
    translation_total INTEGER DEFAULT 0,
    grammar_correct INTEGER DEFAULT 0,
    grammar_total INTEGER DEFAULT 0,
    orthography_correct INTEGER DEFAULT 0,
    orthography_total INTEGER DEFAULT 0,
    lexics_correct INTEGER DEFAULT 0,
    lexics_total INTEGER DEFAULT 0,
    sessions INTEGER DEFAULT 0,
    last_login TEXT,
    FOREIGN KEY(user_id) REFERENCES users(id)
);
""")
```

Оновлення статистики відбувається через метод update_task_stats(), який динамічно формує назви стовпців залежно від типу вправи. Такий підхід спрощує масштабування та дозволяє легко додавати нові категорії мовних завдань. При кожному виконанні вправи система збільшує кількість спроб та кількість правильних відповідей:

```
column_total = f"{category}_total"
column_correct = f"{category}_correct"
self.conn.execute(
    f"UPDATE user_stats SET {column_total} = {column_total} + 1
```

```
    WHERE user_id = ?",
        (user_id,)
    )
```

Крім збереження статистики, підсистема відповідає за управління обліковими записами користувачів: створення нового профілю, пошук за логіном, перевірку пароля та оновлення часу останнього входу. Реалізація цих функцій виконується в методах create_user(), get_user_by_username() та update_last_login(). Підключення підсистеми до інтерфейсу здійснюється через передачу єдиного екземпляра Database у всі компоненти програми, що забезпечує узгоджену роботу з даними.

Завдяки використанню SQLite підсистема бази даних працює без зовнішніх залежностей, забезпечуючи стабільність роботи комплексу та можливість швидкого перенесення даних у межах одного файлу. Така архітектура дозволяє відокремити логіку збереження інформації від функціональних модулів і спрощує підтримку програмної системи.

Підсистема налаштувань у програмно-апаратному комплексі забезпечує збереження локальних параметрів роботи застосунку та індивідуальних вподобань користувача. Оскільки комплекс працює автономно, усі конфігурації зберігаються у локальній базі даних SQLite, що дає змогу уникнути залежності від мережевих сервісів та підтримувати швидкий доступ до даних. Налаштування включають вибір типу завдань за замовчуванням, робочий режим інтерфейсу, а також інші внутрішні параметри, необхідні для узгодженої роботи модулів. Читання та оновлення параметрів виконується безпосередньо через об’єкт Database, що виступає єдиною точкою доступу до сховища даних. Завдяки цьому зміна параметрів виконується централізовано й не потребує додаткової конфігурації окремих модулів застосунку.

Другим важливим елементом є підсистема статистики, що відстежує прогрес користувача у межах основних мовних категорій: граматика, орфографія, лексика та переклад. Під час кожного проходження вправи

система викликає метод оновлення статистики, який інкрементує кількість правильних та загальних відповідей. Це дозволяє поступово накопичувати історію результатів і формувати об’єктивну картину успішності навчання. Дані статистики записуються в окрему таблицю user_stats, що створюється автоматично під час реєстрації нового користувача. Кожна категорія має власні лічильники, що спрощує подальший аналіз та відображення результатів у графічному інтерфейсі.

Під час авторизації статистика оновлюється також із технічного боку: фіксується дата останнього входу та кількість сесій. Це дає можливість реалізувати розширені сценарії, такі як нагадування про необхідність тренування чи побудова динаміки регулярності занять. Усі ці операції виконуються локально та не впливають на продуктивність комплексу, оскільки обсяги даних є незначними.

У вікні інтерфейсу статистика відображається у вигляді зведеної таблиці та коротких індикаторів успішності. Це дозволяє користувачу одразу оцінити власний прогрес та визначити, на яких навичках необхідно зосередитися. Завдяки модульній реалізації підсистеми статистики та налаштувань вони не залежать від інших компонентів комплексу й можуть бути розширені додатковими параметрами або новими видами аналітики без зміни логіки програми.

Завершальним етапом розробки програмної частини стало об’єднання всіх функціональних модулів у головному вікні застосунку MainWindow. Саме цей компонент забезпечує взаємодію користувача з підсистемами аудіозапису, розпізнавання мовлення, граматичного аналізу, генерації навчальних завдань, оцінювання усного мовлення та роботи з локальною базою даних. Архітектура побудована таким чином, щоб кожен модуль виконував свою чітко визначену роль, а MainWindow координував виклики функцій, передачу даних і оновлення графічного інтерфейсу.

Під час запуску застосунок отримує доступ до об’єкта бази даних та

інформації про авторизованого користувача. Після цього створюється головне вікно, у межах якого ініціалізуються всі вкладки й інтерфейсні компоненти. У конструкторі MainWindow відбувається прив’язка подій до методів, які викликають відповідні модулі. Фрагмент початкової частини коду демонструє структуру ініціалізації:

```
class MainWindow(QMainWindow):
        def __init__(self, db, user):
super().__init__()
self.db = db
self.user = user
self.recognizer = SpeechRecognizer()
self.tts_engine = pyttsx3.init()
self.setup_ui()
```

У цьому фрагменті видно, що модуль розпізнавання мовлення створюється один раз і зберігається як компонент вікна. Аналогічно ініціалізується механізм синтезу мовлення, а також елементи керування інтерфейсом. Це дозволяє забезпечити доступ до всіх функцій зі спільного контексту, не порушуючи модульності системи.

Окреме місце займає інтеграція аудіозапису з модулем розпізнавання мовлення. Після завершення запису клас AudioRecorder надсилає сигнал, що передає зібрані фрейми аудіо. У головному вікні дані конвертуються у формат і надсилаються до розпізнавача:

```
def on_record_finished(self, audio_bytes):
wav_path = self.save_temp_wav(audio_bytes)
text = self.recognizer.transcribe(wav_path)
self.text_output.setPlainText(text)
```

Таким чином забезпечується повний цикл «запис → обробка → відображення», де MainWindow виступає координатором між модулями AudioRecorder та SpeechRecognizer.

У вкладці граматичного аналізу виклик модуля ai_analyze() відбувається

безпосередньо з головного вікна. Після того, як користувач вводить текст або отримує його через розпізнавання мовлення, метод передає текст у модуль аналізу, отримує структуровану відповідь і відображає її у відповідному полі інтерфейсу. Обробка здійснюється асинхронно з метою уникнення блокування інтерфейсу.

Аналогічна інтеграція реалізована для генератора навчальних завдань. Користувач обирає тип та складність завдання, після чого MainWindow викликає функцію generate_task(), отримує сформований блок із питанням та варіантами відповідей і виводить його на екран. Додатково передбачено механізм перевірки відповідей через функцію check_answer(), який також викликається головним вікном.

Для підсистеми оцінювання усного мовлення інтеграція має двоетапну структуру: спочатку текст отримується через модуль розпізнавання мовлення, після чого передається у функцію evaluate_speech(). Відповідь системи містить рекомендації та бальну оцінку, які негайно виводяться користувачу. Таке поєднання двох модулів у одному інтерфейсному виклику забезпечує безперервність роботи та швидкість взаємодії.

Окремої уваги потребує інтеграція з локальною базою даних. Після кожного входу в систему головне вікно оновлює статистику користувача, а при виконанні навчальних завдань викликає методи update_task_stats(), які фіксують результати вправ у таблиці user_stats. Це дозволяє підтримувати індивідуальний прогрес, зберігати історію та використовувати її в модулях.

Завдяки модульній архітектурі інтеграція не призводить до змішування логіки окремих компонентів. MainWindow виконує роль центрального контролера, який отримує дії користувача, обробляє їх у відповідному модулі й повертає результати у графічний інтерфейс. Такий підхід забезпечує масштабованість, спрощує додавання нових функцій і полегшує супровід комплексу, оскільки кожен модуль залишається незалежним і може вдосконалюватися окремо від інших.

4 ТЕСТУВАННЯ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ

4.1 Тестування функціональних модулів комплексу

Тестування функціональних модулів програмного комплексу має на меті перевірити коректність роботи кожного окремого компонента, виявити можливі помилки у взаємодії між ними та підтвердити відповідність реалізованих алгоритмів визначеним вимогам. Оскільки система складається з незалежних модулів — аудіозапису, розпізнавання мовлення, граматичного аналізу, генерації навчальних завдань, оцінювання мовлення, синтезу мовлення, роботи з локальною базою даних та підсистеми авторизації — тестування проводилося поетапно, з ізоляцією кожного компонента.

Перевірка модуля аудіозапису полягала у фіксації якості збирання аудіофреймів, стабільності роботи потоку та відсутності блокування графічного інтерфейсу під час запису. Для цього було виконано серію тестів із різною тривалістю запису, перевірено формування буфера та його подальше збереження у формат MP3. На рис. 4.1 наведено результат виконання юніт тестів.

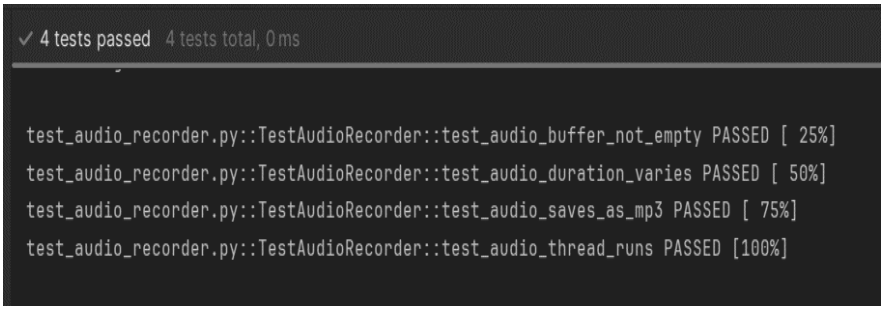

Рисунок 4.1 — Результат виконання юніт тестів модуля аудіозапису

Для модуля розпізнавання мовлення основною метою було переконатися, що офлайн-модель Vosk коректно обробляє аудіофайл і повертає текстовий результат без критичних спотворень. У межах тестування були використані різні голосові фрагменти: монотонні, швидкі, з паузами та шумовим фоном. Тестове порівняння введення та транскрипції демонструє

відповідність модуля очікуваним результатам. На рис. 4.2 наведено результат виконання юніт тестів.

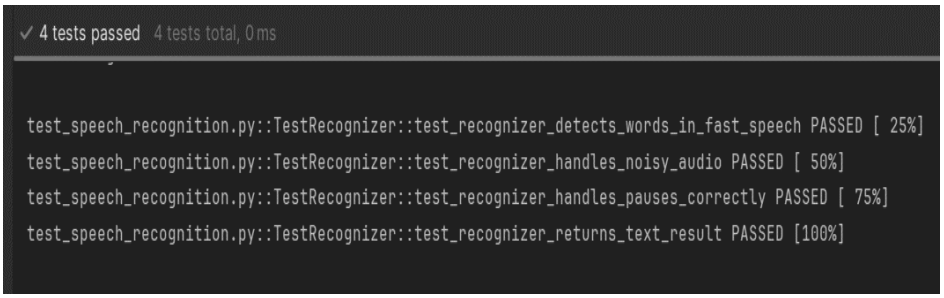

Рисунок 4.2 — Результат виконання юніт тестів модуля розпізнавання мовлення

Модуль граматичного аналізу тестувався із використанням набору контрольних фраз, що містили поширені граматичні, лексичні та синтаксичні помилки. Завданням було перевірити, чи модель Gemini 2.0 Flash виділяє необхідні помилки, чи структура відповіді відповідає заданому формату, а також чи модуль стійкий до неправильного або неповного користувацького введення. Усі результати перевірки відображалися у вкладці аналізу, що дозволило оцінити роботу компонента у реальному інтерфейсному середовищі. На рис. 4.3 наведено результат виконання юніт тестів.

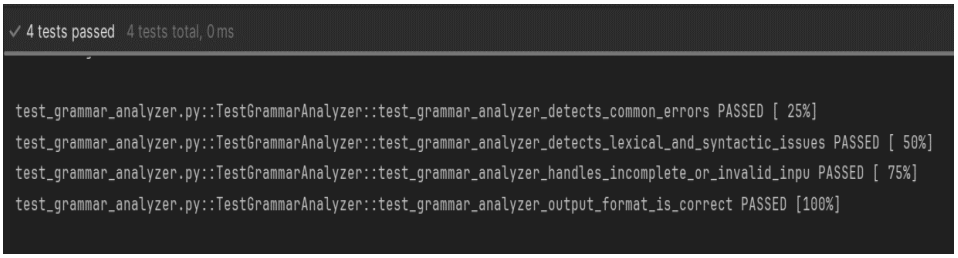

Рисунок 4.3 — Результат виконання юніт тестів модуля граматичного аналізу

Для модуля генерації навчальних завдань тестування передбачало перевірку стабільності формування вправ різних типів і рівнів складності, а також правильності структури відповідей. Було протестовано кількадесят запитів із різними параметрами, після чого оцінено якість сформованих

завдань і коректність виділення правильної відповіді. Особливу увагу приділено роботі механізму повторної генерації у випадку некоректної відповіді моделі. На рис. 4.4 наведено результат виконання юніт тестів.

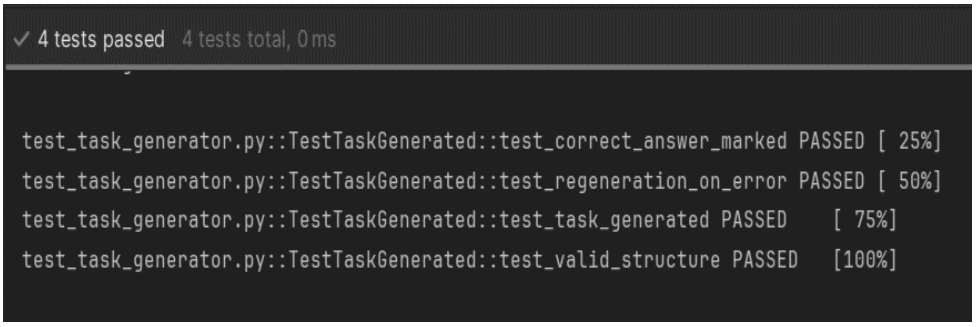

Рисунок 4.4 — Результат виконання юніт тестів модуля генерації навчальних завдань

Під час тестування модуля оцінювання аналізу усного мовлення виконувалася передача різних транскрибованих фрагментів до функції evaluate_speech(). Метою тестів було перевірити, чи модель коректно формує оцінювання темпу, чіткості, плавності та логічності висловлення, а також чи відповіді містять рекомендації у відповідності до формату. Окремо перевірялася стійкість до сильно скорочених або шумових текстів.

Модуль синтезу мовлення перевірявся на коректність генерації аудіо з використанням бібліотеки gTTS, а також на стабільність відтворення та збереження MP3-файлів. У межах тестування генерувалися десятки голосових фрагментів різної довжини, після чого проводилася перевірка їх якості та відсутності пошкоджень файлів. На рис. 4.5 наведено виконання юніт тестів.

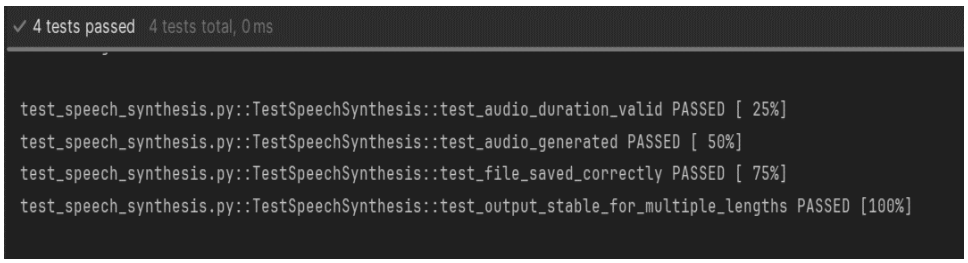

Рисунок 4.5 — Результат виконання юніт тестів модуля синтезу мовлення

Для підсистеми збереження даних виконувалося тестування створення таблиць, читання та запису інформації, а також оновлення статистики користувача. Було протестовано сценарії реєстрації, авторизації, некоректного пароля, дублювання логіна та оновлення статистичних показників. Окремо перевірено реакцію системи на спроби звернення до неіснуючого користувача або неправильних атрибутів даних.

Проведені дослідження показали, що всі модулі працюють стабільно, виконують свої функції відповідно до вимог і забезпечують коректну взаємодію з іншими частинами комплексу. Виявлені під час тестування несуттєві недоліки були усунені, а модульна структура системи дозволила локалізувати проблеми без впливу на інші компоненти. Результати цього етапу створюють основу для системного тестування, яке буде описано в наступному підрозділі.

4.2 Системне тестування програмного комплексу

Системне тестування програмного комплексу має на меті перевірити його працездатність як єдиного цілісного середовища, у якому всі функціональні модулі взаємодіють між собою відповідно до визначених вимог. На цьому етапі основний акцент робиться не на перевірці окремих алгоритмів чи компонентів, а на оцінюванні того, наскільки узгоджено, послідовно та стабільно функціонує система під час виконання реальних користувацьких сценаріїв. Метою системного тестування є виявлення можливих помилок у механізмах взаємодії між модулями, перевірка коректності обробки даних при переході між етапами навчального процесу, а також оцінка стійкості комплексу до непередбачених або граничних ситуацій, що можуть виникати в реальному використанні.

Для досягнення поставленої мети системне тестування проводилося за підходом, який моделює повний робочий цикл користувача — від моменту реєстрації до виконання навчальних завдань та аналізу усного мовлення.

Такий підхід дозволяє охопити всі критично важливі точки взаємодії між модулями, включно з передаванням аудіоданих, формуванням транскрипту, виконанням граматичного аналізу, генерацією завдань, оцінюванням відповідей та оновленням статистики в локальній базі даних. Повноцінне проходження цих етапів забезпечує можливість оцінити поведінку програмного комплексу в умовах реального використання, коли вплив кожного модуля на інші є безпосереднім і нерозривним.

Першим етапом тестування стало перевірення процедури реєстрації нового користувача. Було виконано введення унікального логіна та пароля, після чого система мала коректно створити новий обліковий запис у локальній базі даних. Під час тестування перевірялося оброблення помилкових даних, реакція на дублювання логіна й відповідність графічної форми реєстрації вимогам коректності та доступності інтерфейсу. Також підтверджено, що після успішної реєстрації автоматично створюється запис у таблиці статистики користувача. На рис. 4.6 показано вікно реєстрації нового користувача.

Після завершення реєстрації проводилося тестування модуля авторизації. У ході системної перевірки було підтверджено, що введення коректних облікових даних відкриває доступ до інтерфейсу, а неправильний пароль або логін викликають відповідні попередження. Надійність авторизації була протестована за допомогою кількох послідовних входів та виходів із програми. На рисунку 4.7 показано вікно авторизації користувача.

Після успішного входу виконано тестування головного меню застосунку, яке слугує навігаційним центром системи. Перевірці підлягали правильність відображення вкладок, доступність усіх функцій для авторизованого користувача та стабільність переходів між основними розділами. Було підтверджено, що головне меню коректно ініціалізує створення всіх вкладок, а відсутність затримок у відображенні інтерфейсу підтверджує узгодженість взаємодії між GUI та логічними модулями.

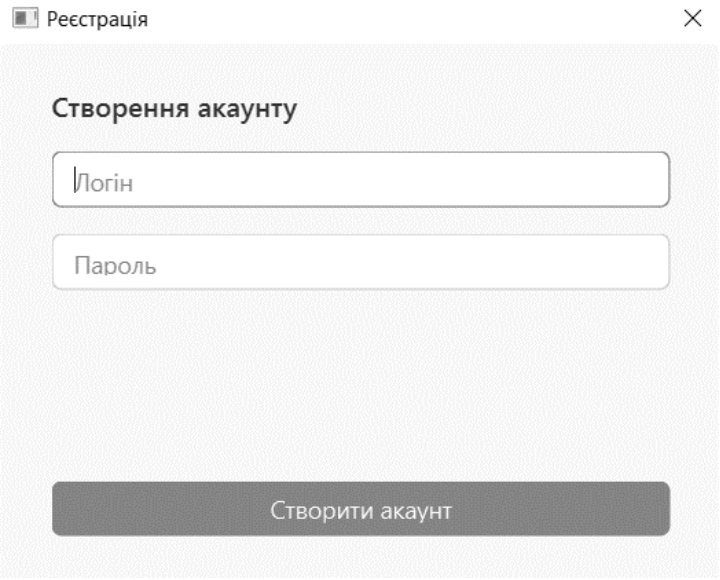

Рисунок 4.6 — Вікно реєстрації користувача

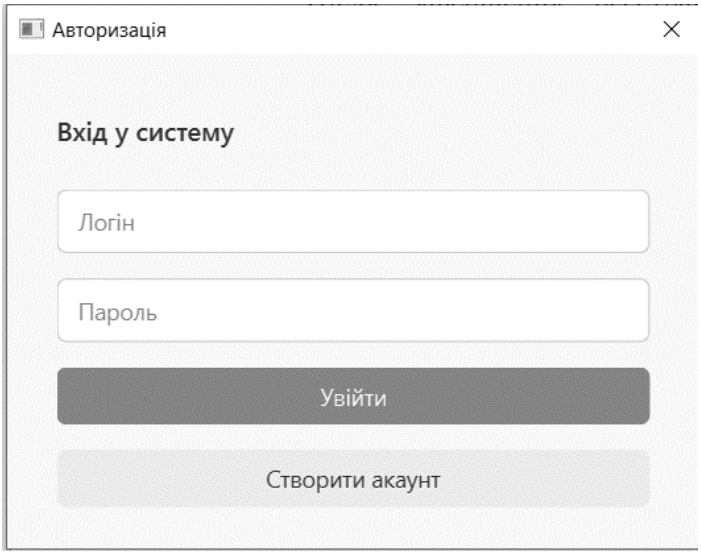

Рисунок 4.7 — Вікно авторизації користувача

Наступним етапом стало тестування вкладки мовлення, де користувач має можливість записати аудіофрагмент і передати його до модуля розпізнавання мовлення. Було проведено тестові записи різної довжини, після чого оцінено стабільність роботи аудіопотоку, моментальність завершення запису, формування тимчасового аудіофайлу та відправлення його на транскрипцію. Результатом тестування стало підтвердження того, що отриманий текст миттєво відображається у відповідному полі, а система

коректно обробляє випадки порожнього або надто короткого запису. На рис. 4.8 показано вкладка мовлення та її функціонал.

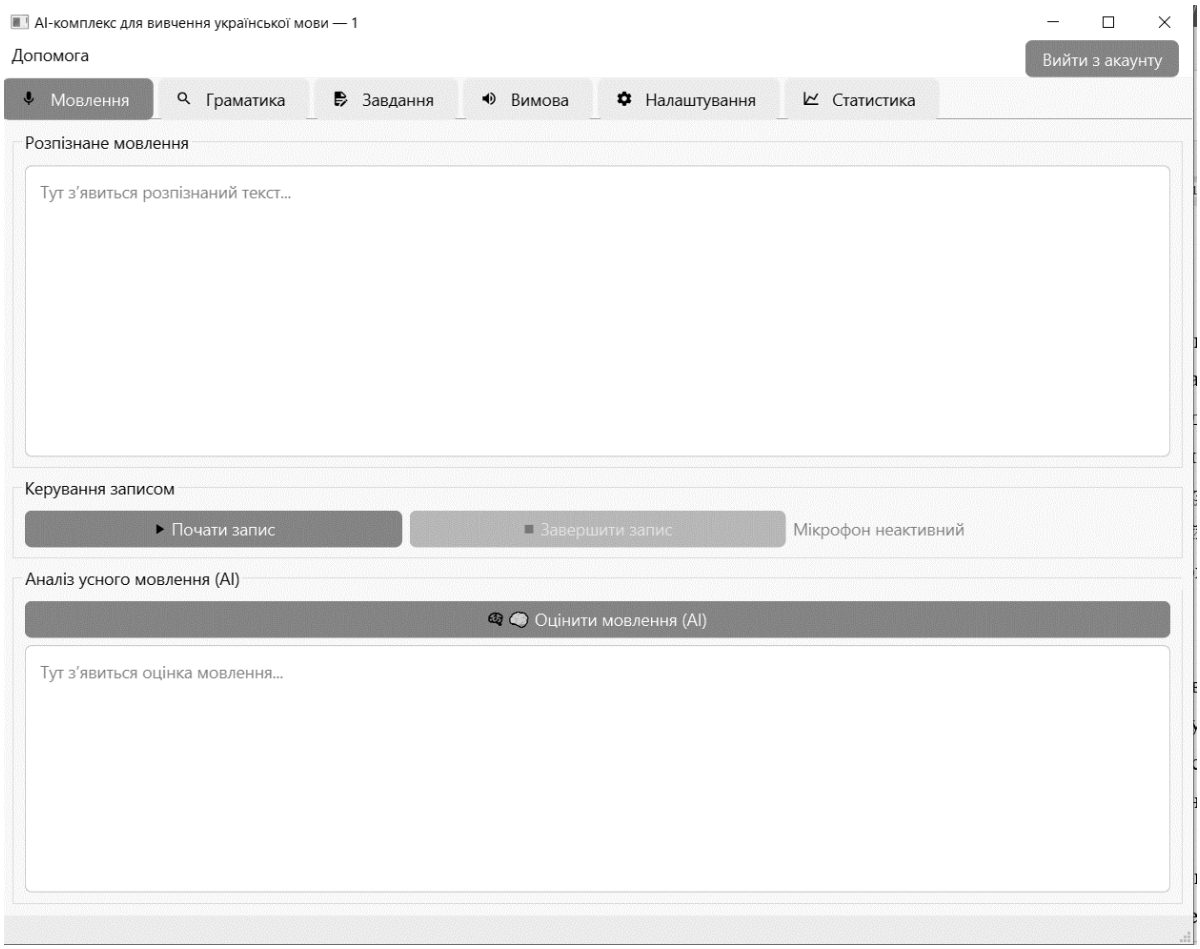

Рисунок 4.8 — Вкладка «Мовлення»

У вкладці граматики перевірялося передавання тексту до модуля граматичного аналізу та отримання структурованого результату від моделі Gemini (рис. 4.9). Під час тестування вводилися як коректні, так і навмисно помилкові фрази, що дозволило оцінити якість і повноту зворотного аналізу. Усі результати аналізу успішно відображалися в інтерфейсі, а система коректно обробляла помилки введення. Окремо підтверджено, що вкладка не блокує основний інтерфейс під час очікування відповіді моделі.

Далі було протестовано вкладку навчальних завдань, де користувач отримує згенеровану вправу та має можливість її виконати. Сценарій тестування передбачав вибір різних типів завдань і рівнів складності.

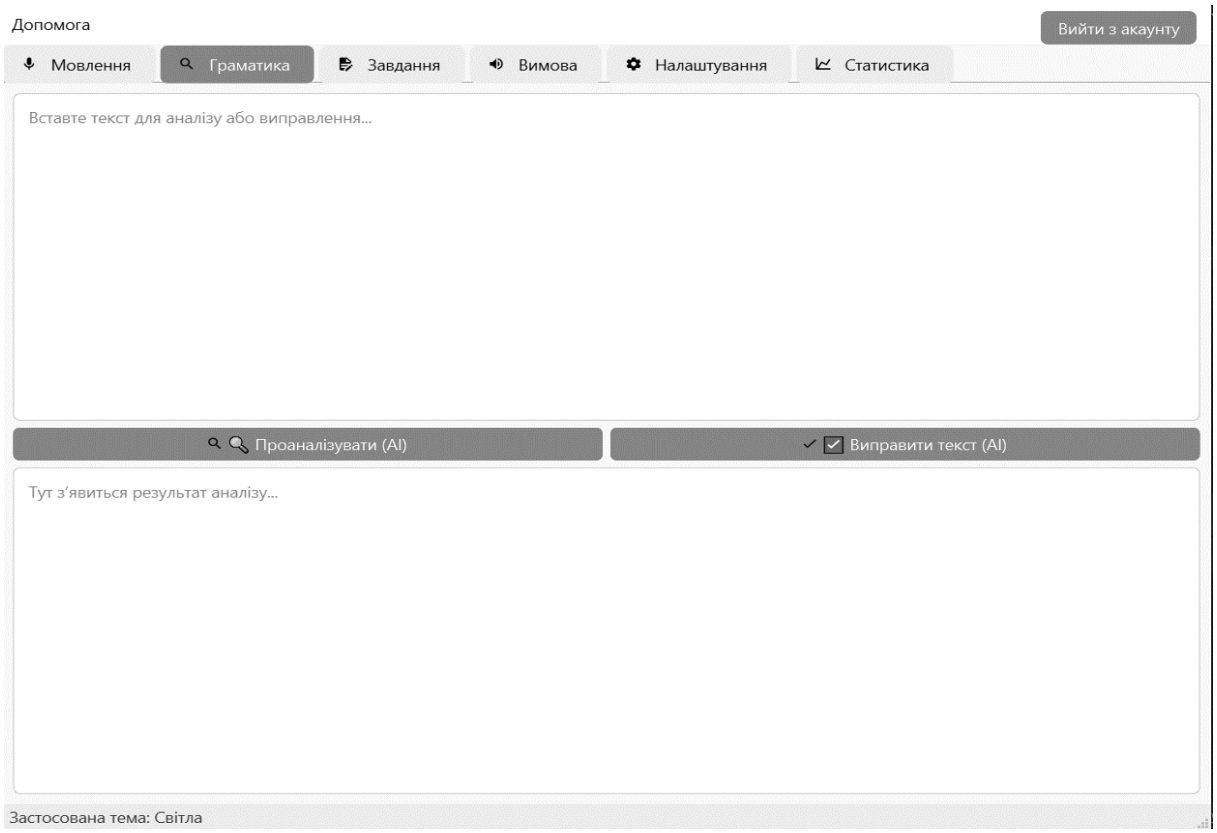

Рисунок 4.9 — Вкладка «Граматика»

Під час тестування модуля генерації навчальних завдань виконувалася не лише перевірка стабільності формування вправ різних типів і рівнів складності, а й оцінювання коректності їх генерації та правильності відпрацювання механізму перевірки відповідей. Додатково було перевірено стабільність роботи модуля під час багаторазового повторення вправ, а також коректність оновлення навчальних даних між послідовними сесіями користувача. Усі завдання формувалися динамічно відповідно до параметрів запиту, а правильна відповідь зберігалася у внутрішній структурі даних до моменту виконання перевірки. Окремо контролювалася обробка граничних випадків, зокрема введення порожніх значень, некоректних символів та частково правильних відповідей. Також перевірялося, що система коректно зберігає проміжні результати та не допускає повторного використання попередніх правильних відповідей під час генерації нових вправ. Крім того, оцінювалася узгодженість повідомлень інтерфейсу із фактичним результатом

перевірки та правильність переходу до наступного завдання після завершення поточного. Система коректно реагувала на неправильні введення, повідомляючи користувача про результат виконання та надаючи підказку щодо подальших дій (рис. 4.10).

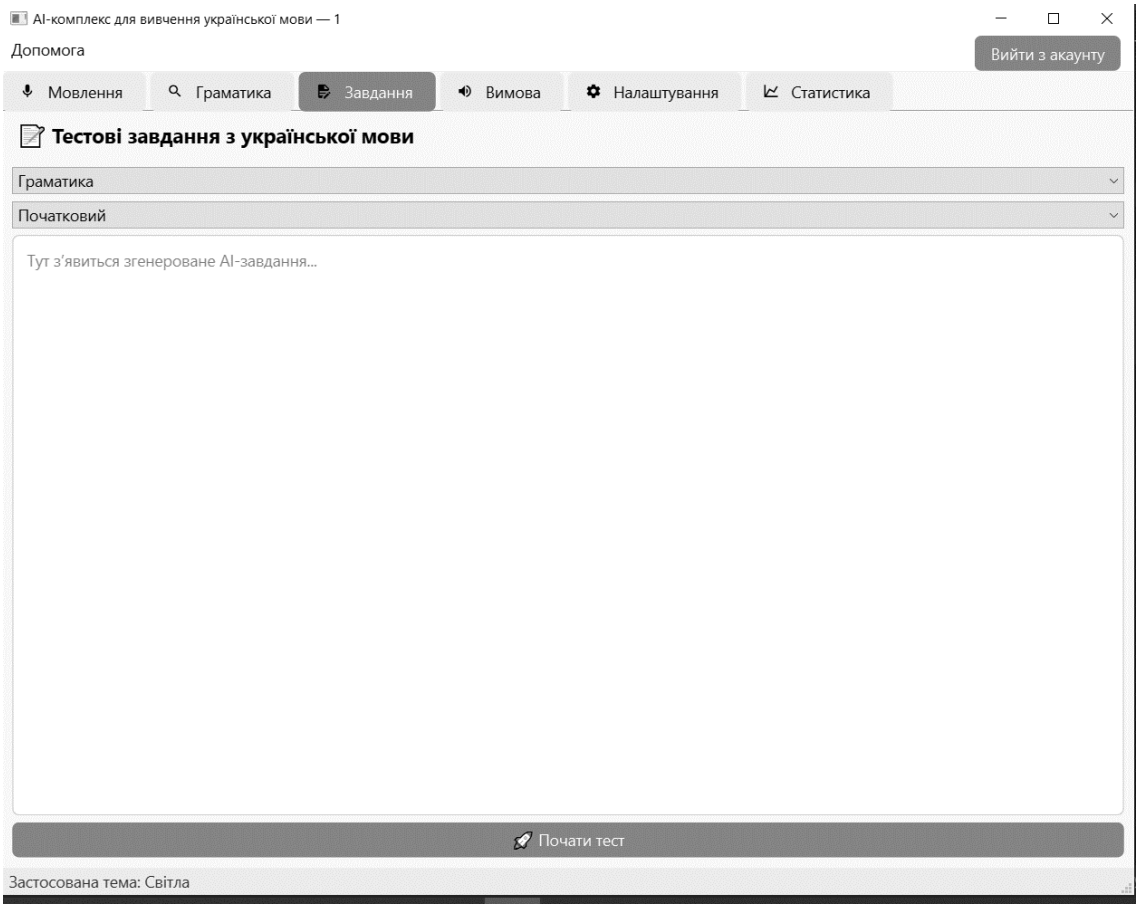

Рисунок 4.10 — Вкладка «Завдання»

Тестування вкладки вимови передбачало повний цикл оцінювання усного мовлення: розпізнавання аудіо, передавання тексту в модуль evaluate_speech() та відображення детального аналізу. Було протестовано кілька аудіофрагментів із різною якістю дикції та темпом мовлення. Окремо перевірялася коректність оброблення результатів розпізнавання на рівні слів і речень, а також відповідність сформованих зауважень фактичним помилкам у вимові користувача. Тестування охоплювало сценарії з чіткою артикуляцією, прискореним темпом мовлення та наявністю пауз, що дозволило оцінити

стійкість алгоритмів аналізу, а також коректність формування рекомендацій щодо покращення вимови. Усі результати відображалися у відповідному форматі, а модуль стабільно працював навіть із короткими або неповними транскрибованими фрагментами (рис. 4.11).

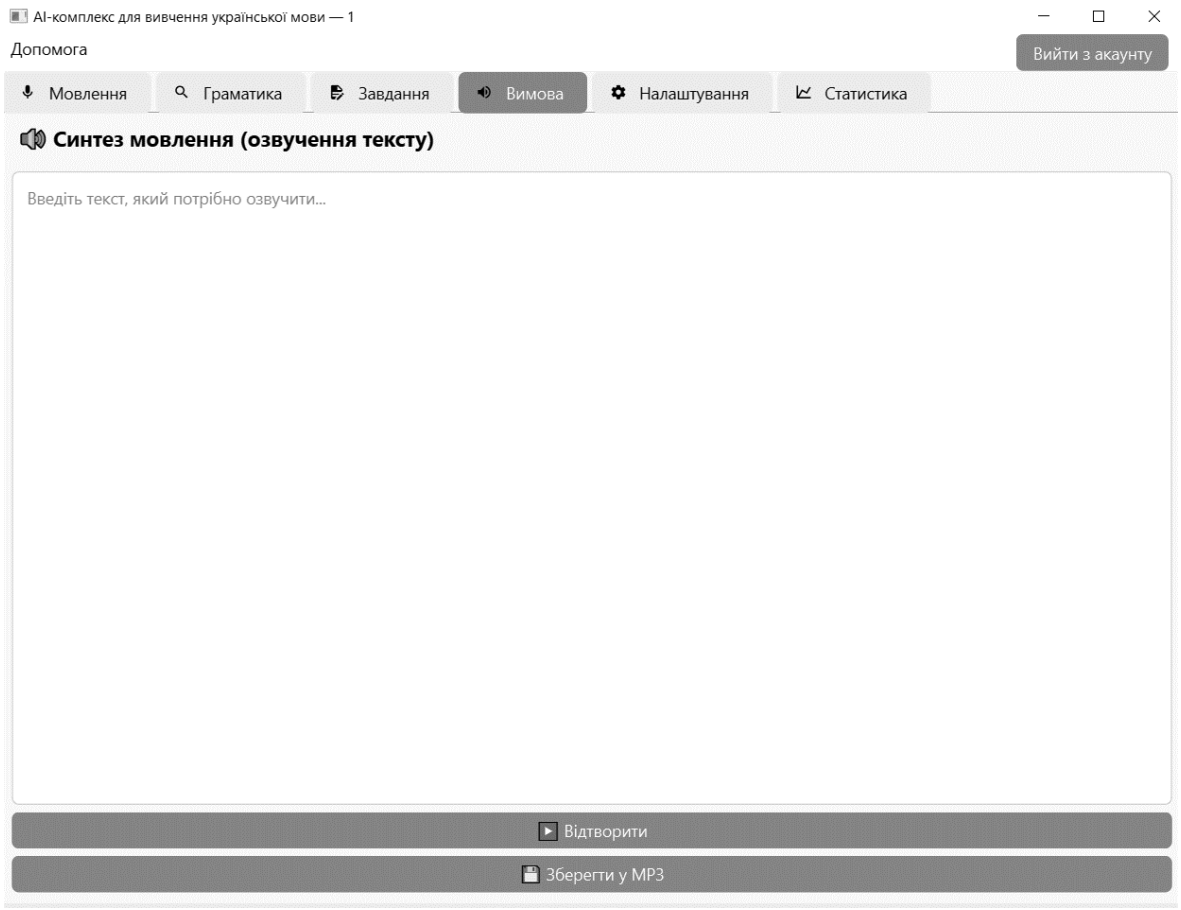

Рисунок 4.11 — Вкладка «Вимова»

У вкладці налаштувань перевірялася можливість зміни параметрів роботи комплексу, зокрема вибору типу завдань за замовчуванням або зміни окремих поведінкових параметрів інтерфейсу. Під час тестування підтверджено, що всі зміни коректно зберігаються у локальній базі даних, а значення налаштувань зчитуються при повторному запуску застосунку. Додатково перевірялася коректність застосування змінених параметрів без необхідності перезапуску застосунку, а також їх вплив на поведінку інших модулів системи. Тестування охоплювало сценарії багаторазової зміни

налаштувань, повернення до значень за замовчуванням і взаємодії з іншими вкладками інтерфейсу. Окрему увагу приділено перевірці стійкості системи до неконсистентних або частково відсутніх конфігураційних параметрів, що дозволило підтвердити надійність механізмів валідації та оброблення помилок. Окремо протестовано роботу з некоректними конфігураційними даними та їх безпечне оброблення (рис. 4.12).

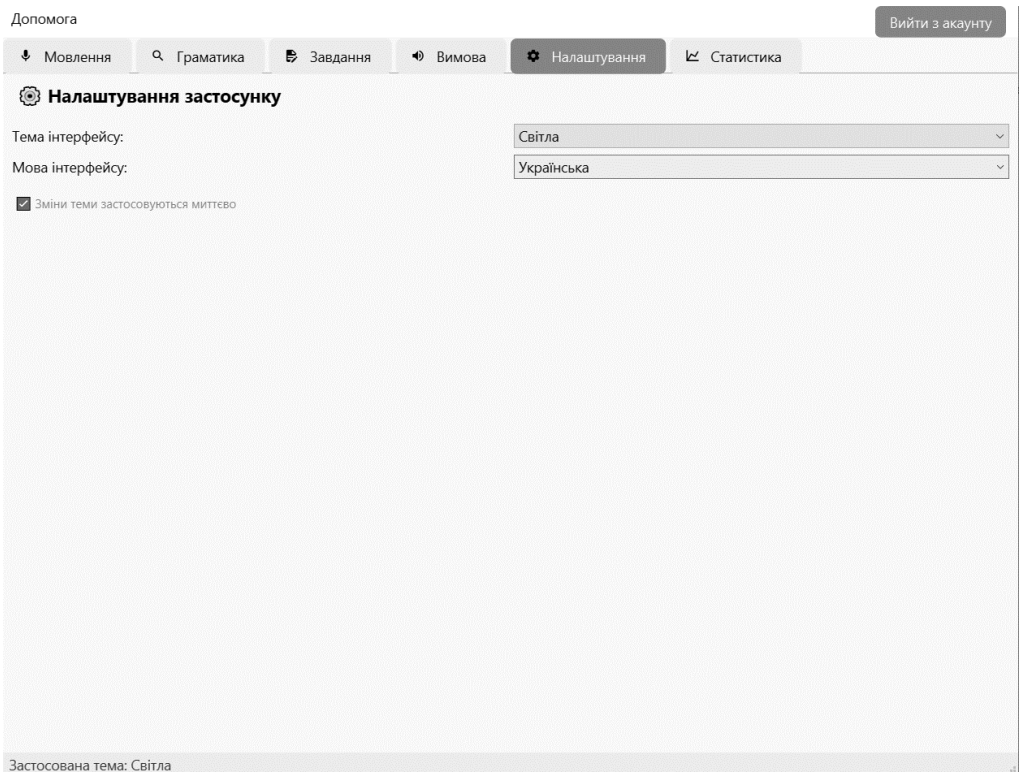

Рисунок 4.12 — Вкладка Налаштування

Перевірка вкладки статистики була зосереджена на відображенні прогресу користувача та коректності обчислення лічильників у таблиці user_stats. У ході тестування виконано кілька вправ у різних режимах, після чого перевірено, чи статистика оновилася відповідно до результатів. Система стабільно зчитувала й відображала дані, а також коректно обробляла випадки, коли користувач не мав історії виконання завдань (рис. 4.13).

Завершальним етапом сценарію стала перевірка механізму виходу з акаунту. Було підтверджено, що після натискання кнопки «Вийти» застосунок

коректно завершує поточну сесію, повертається до діалогового вікна авторизації та очищує внутрішні об’єкти, пов’язані з активним користувачем. Це підтверджує завершеність циклу взаємодії та коректність логічної структури всієї системи.

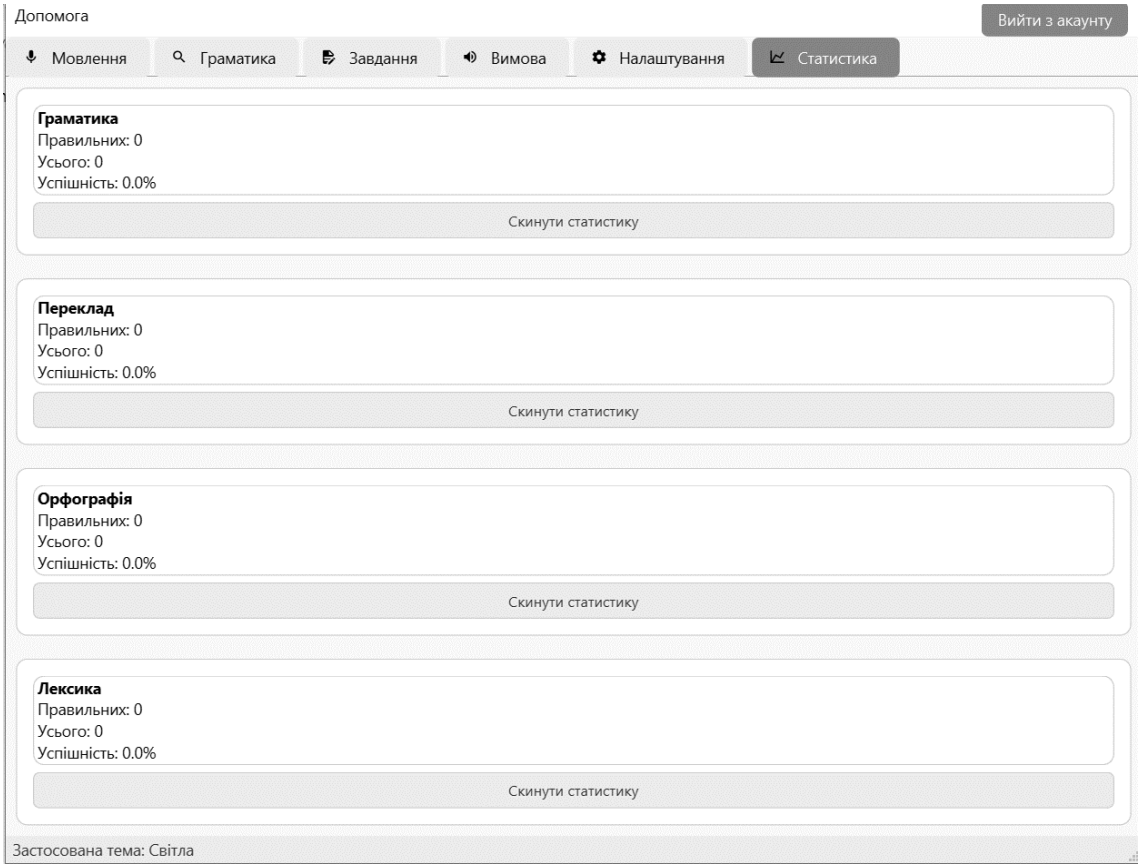

Рисунок 4.13 — Вкладка «Статистика»

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Мета проведення комерційного та технологічного аудиту є Метою магістерської кваліфікаційної роботи є удосконалення процесу вивчення української мови шляхом розроблення програмно-апаратного комплексу, який забезпечує адаптивність, інтерактивність та персоналізацію навчання на основі технологій штучного інтелекту[30].

Для проведення технологічного аудиту було залучено 3-х незалежних експертів Вінницького національного технічного університету, кафедри обчислювальної техніки: професор Захарченко Сергій Михайлович, доцент Войцехівська Олена Валеріївна, доцент Тарновський Микола Геннадійович.

Для проведення технологічного аудиту було використано таблицю 5.1 в якій за п’ятибальною шкалою використовуючи 12 критеріїв здійснено оцінку комерційного потенціалу.

Таблиця 5.1 — Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження табл. 5.1

3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Продовження табл. 5.1

9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію.	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Для оцінки комерційного потенціалу розробки було проведено експертне опитування, результати якого зведено у вигляді середньоарифметичної суми балів (СБ). Залежно від отриманого значення, розробка класифікується за рівнем комерційної перспективності відповідно до наведеної нижче шкали:

Таблиця 5.2 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

В таблиці 5.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 5.3 — Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	проф. Захарченко С. М.	доц. Войцехівська О. В.	доц. Тарновський М. Г.
	Бали, виставлені експертами:		
1	3	3	3
2	2	3	2
3	3	3	3
4	3	4	3
5	2	3	3
6	3	3	3
7	2	3	3
8	3	3	3
9	2	3	4

Продовження таблиці 5.3

10	3	3	3
11	2	2	3
12	3	3	3
Сума балів	$СБ_1$=31	$СБ_2$=34	$СБ_3$=34
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{31+34+34}{3} = 33$		

Середньоарифметична оцінка, отримана на основі експертних висновків, становить 33 бали, і згідно з таблицею 4.2, це вказує на рівень вище середнього комерційного потенціалу результатів проведених досліджень.

Розроблений програмно-апаратний комплекс може бути реалізований у закладах вищої та середньої освіти, на мовних курсах або як індивідуальний навчальний інструмент для іноземців, що вивчають українську мову.

Основними користувачами є викладачі української мови, студенти та самонавчальні користувачі.

Система може бути впроваджена як окремий десктопний додаток або як частина інтерактивного навчального середовища з підтримкою голосового введення, розпізнавання та аналізу мовлення.

В якості аналога було обрано Duolingo — популярну систему для вивчення мов.

Основними недоліками аналога є відсутність підтримки української мови як мови навчання для іноземців, обмежені можливості інтеграції з апаратними засобами (мікрофонами, динаміками) та відсутність глибокого аналізу вимови.

Також до недоліків можна віднести недостатню персоналізацію навчального процесу та орієнтацію на англомовну аудиторію.

У розробці дана проблема вирішується за рахунок інтеграції штучного

інтелекту, який аналізує мовлення користувача в реальному часі, формує зворотний зв’язок українською мовою та генерує вправи з урахуванням рівня знань.

Система випереджає аналог за такими параметрами, як адаптивність навчання, інтерактивна вимова.

В таблиці 5.4 наведені основні техніко-економічні показники аналога і нової розробки.

Проведемо оцінку якості продукції, яка є найефективнішим засобом забезпечення вимог споживачів та порівняємо її з аналогом.

Таблиця 5.4 — Основні параметри нової розробки та товару-конкурента

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
1	2	3	4	5
Точність розпізнавання мовлення, %	80	95	1,2	30
Інтерактивність навчання, бали	6	8	1,3	40
Зручність інтерфейсу, бали	7	8	1,1	30

Визначимо відносні одиничні показники якості по кожному параметру за формулами (5.1) та (5.2) і занесемо їх у відповідну колонку табл. 5.5.

$$q_i = \frac{P_{Hi}}{P_{Бi}} \qquad (5.1)$$

або

$$q_i = \frac{P_{Бi}}{P_{Нi}} \tag{5.2}$$

де $P_{Нi}$, $P_{Бi}$ — числові значення *i*-го параметру відповідно нового і базового виробів.

$$q_1 = \frac{95}{70} = 1{,}2;$$

$$q_2 = \frac{8}{6} = 1{,}3;$$

$$q_3 = \frac{9}{7} = 1{,}3.$$

Відносний рівень якості нової розробки визначаємо за формулою:

$$К_{я.в.} = \sum_{i=1}^{n} q_i \cdot \alpha_i, \tag{5.3}$$

$$К_{я.в.} = 1{,}35 \cdot 0{,}3 + 1{,}3 \cdot 0{,}4 + 1{,}3 \cdot 0{,}3 = 1{,}31$$

Відносний коефіцієнт показника якості нової розробки більший одиниці, отже нова розробка якісніший базового товару-конкурента.

Наступним кроком є визначення конкурентоспроможності товару. Конкурентоспроможність товару є головною умовою конкурентоспроможності підприємства на ринку і важливою основою прибутковості його діяльності.

Однією із умов вибору товару споживачем є збіг основних ринкових характеристик виробу з умовними характеристиками конкретної потреби покупця. Такими характеристиками найчастіше вважають нормативні та технічні параметри, а також ціну придбання та вартість споживання товару.

В табл. 5.5 наведено технічні та економічні показники для розрахунку

конкурентоспроможності нової розробки відносно товару-аналога, технічні дані взяті з попередніх розрахунків.

Таблиця 5.5 — Нормативні, технічні та економічні параметри нової розробки і товару-виробника

Показники	Варіанти	
	Базовий (товар-конкурент)	Новий (інноваційне рішення)
1	2	3
1. Нормативно-технічні показники		
Точність розпізнавання мовлення, %	70	95
Інтерактивність навчання, бали	6	8
Зручність інтерфейсу, бали	7	9
2. Економічні показники		
Ціна придбання, грн	840	630

Загальний показник конкурентоспроможності інноваційного рішення (К) з урахуванням вищезазначених груп показників можна визначити за формулою:

$$K = \frac{I_{m.n.}}{I_{e.n.}}, \tag{5.4}$$

де $I_{m.n.}$ — індекс технічних параметрів;

$I_{e.n.}$ — індекс економічних параметрів.

Індекс технічних параметрів є відносним рівнем якості інноваційного рішення. Індекс економічних параметрів визначається за формулою (5.5)

$$I_{e.n.} = \frac{\sum_{i=1}^{n} P_{Hei}}{\sum_{i=1}^{n} P_{Бei}}, \quad (5.5)$$

де P_{Hei}, $P_{Бei}$ — економічні параметри (ціна придбання та споживання товару) відповідно нового та базового товарів.

$$\mathrm{I}_{\text{е.п.}} = \frac{840}{630} = 1{,}3;$$

$$\mathrm{К} = \frac{1{,}31}{1{,}3} = 1{,}15.$$

Зважаючи на розрахунки, можна зробити висновок, що нова розробка буде конкурентоспроможніше, ніж конкурентний товар.

Розроблений програмно-апаратний комплекс може бути реалізований у закладах вищої та середньої освіти, на мовних курсах або як індивідуальний навчальний інструмент для іноземців, що вивчають українську мову.

Основними користувачами є викладачі української мови, студенти та самонавчальні користувачі.

Система може бути впроваджена як окремий десктопний додаток або як частина інтерактивного навчального середовища з підтримкою голосового введення, розпізнавання та аналізу мовлення.

5.2 Прогнозування витрат на виконання науково-дослідної роботи

До складу витрат на проведення НДР входять: оплата праці виконавців, обов’язкові соціальні нарахування, витрати на матеріали, паливо та енергію, необхідні для науково-виробничого процесу, кошти на службові відрядження, придбання програмного забезпечення, що використовується у дослідницькій діяльності, інші супутні видатки, а також накладні витрати.

Основна заробітна плата кожного із дослідників $З_О$, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_О = \frac{М}{Т_р} * t \text{ (грн)} \quad (5.6)$$

де M — місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

T_P — число робочих днів в місяці; приблизно $T_P \approx 21...23$ дні;

t — число робочих днів роботи дослідника.

Зведемо сумарні розрахунки до таблиця 5.6.

Таблиця 5.6 — Заробітна плата дослідника в науковій установі бюджетної сфери

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Керівник	18000	857,1	5	4286
Програміст	19000	904,8	25	22619
Всього				26905

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (5.7)$$

де C_i — погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i — час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна

визначити за формулою:

$$C_i = \frac{М_м \cdot К_і \cdot К_с}{Т_р \cdot t_{зм}}, \quad (5.8)$$

де $М_М$ — розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), грн;

$К_і$ — коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

$К_с$ — мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$Т_р$ — середня кількість робочих днів в місяці, приблизно Тр = 21…23 дні;

$t_{зм}$ — тривалість зміни, год.

Таблиця 5.7 — Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника, грн
1. Розробка алгоритму	3	1	47,6	142,9
2. Написання коду	2	3	64,3	128,6
3. Налагодження програми	3	5	81,0	242,9
4. Випробувальні	5	2	52,4	261,9
5. Документування системи	2	4	71,4	142,9
Всього				919,0

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується як 10 - 12 % від основної заробітної плати робітників.

На даному підприємстві додаткова заробітна плата начисляється в розмірі 11% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{дод}}{100\%} \quad (5.9)$$

$$З_д = 0{,}11 * (26905 + 919{,}0) = 3060{,}62 \text{ (грн)}$$

Нарахування на заробітну плату $Н_{ЗП}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою (5.10):

$$Н_{ЗП} = (З_о + З_р + З_д) * \frac{\beta}{100} \text{ (грн)} \quad (5.10)$$

де $З_о$ — основна заробітна плата розробників, грн.;

$З_д$ — додаткова заробітна плата всіх розробників та робітників, грн.;

$З_р$ — основну заробітну плату робітників, грн.;

β — ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді (5.11)**:**

$$Н_{ЗП} = (26905 + 919{,}0 + 3060{,}62) * \frac{22}{100} = 6794{,}57 \text{ (грн)} \quad (5.11)$$

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби й предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за прямим призначенням згідно з нормами їх витрачання, а також витрачені придбані напівфабрикати, що підлягають монтажу або виготовленню й додатковій обробці в цій організації, чи дослідні зразки, що виготовляються виробниками за документацією наукової організації.

Витрати на матеріали (М) у вартісному вираженні розраховуються

окремо для кожного виду матеріалів за формулою:

$$М = \sum_{i=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{i=1}^{n} В_j \cdot Ц_{вj}, \quad (5.11)$$

де Нj — норма витрат матеріалу j-го найменування, кг;

n — кількість видів матеріалів;

Цj — вартість матеріалу j-го найменування, грн/кг;

Кj — коефіцієнт транспортних витрат, (Кj = 1,1 … 1,15);

Вj — маса відходів j-го найменування, кг;

Цвj — вартість відходів j-го найменування, грн/кг.

Проведені розрахунки зведені в таблицю 4.8.

Таблиця 5.8 — Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, шт	Вартість витраченого матеріалу, грн
Папір	195	1	195
Ручка	18	1	18
Блокнот	35	1	35

Продовження табл. 5.8

Флешка	350	1	350
З врахуванням коефіцієнта транспортування			657,8

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, можуть бути розраховані з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.12)$$

де $Ц_б$ — балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ — термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ — строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Проведені розрахунки необхідно звести до таблиці 5.9.

Таблиця 5.9 — Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн

Продовження табл. 5.9

Ноутбук Lenovo IdeaPad 5	25000	2	2	2083,33
Всього				2083,33

До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень(5.13).

$$В_е = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_і} \quad (5.13)$$

де W_{yt} — встановлена потужність обладнання на певному етапі розробки, кВт;

t_i — тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ — вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ — коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

$\eta_і$ — коефіцієнт корисної дії обладнання, $\eta_і < 1$.

Для написання магістерської роботи використовується персональний комп'ютер для якого розрахуємо витрати на електроенергію.

$$В_е = \frac{0{,}5 \cdot 250 \cdot 12{,}69 \cdot 0{,}5}{0{,}8} = 991{,}41$$

Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників за формулою(5.14):

$$В_{св} = (З_о + З_р) * \frac{Н_{св}}{100\%}, \quad (5.14)$$

де $Н_{св}$ — норма нарахування за статтею «Службові відрядження».

$$В_{св} = 0{,}2 * (26905 + 919{,}0) = 5564{,}76$$

Накладні (загальновиробничі) витрати Внзв охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати Внзв можна прийняти як (100…150)% від суми основної заробітної плати розробників та робітників, які виконували дану

МКНР, тобто:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.15)$$

де $Н_{нзв}$ — норма нарахування за статтею «Інші витрати».

$$В_{нзв} = (26905 + 919{,}0) \cdot \frac{100}{100\%} = 27823{,}81 \text{ грн}$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКНР

В=26905+919,0+3060,62+6794,57+657,8+2083,33+991,41+5565,76+27823,81= 75130,11грн

Прогнозування загальних втрат ЗВ на виконання та впровадження результатів виконаної МКНР здійснюється за формулою:

$$ЗВ = \frac{В}{\eta}, \quad (5.16)$$

де η — коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії науково-дослідних робіт, то коефіцієнт β = 0,7.

Звідси:

$$ЗВ = \frac{75130{,}11}{0{,}7} = 107328{,}73 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки

У даному підрозділі кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової

роботи. Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою

$$\Delta П_i = \sum_1^n \left(\Delta Ц_o \cdot N + Ц_o \cdot \Delta N\right)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100}\right) \quad (5.17)$$

де $\Delta Ц_0$ — покращення основного оціночного показника від впровадження результатів розробки у даному році.

N — основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN — покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

$Ц_о$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n — кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

л — коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт *л* = 0,8333.

р — коефіцієнт, який враховує рентабельність продукту. р = 0,25;

х — ставка податку на прибуток. У 2025 році — 18%.

Припустимо, що ціна зросте на 100 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року на 900 шт., протягом другого року — на 1000 шт., протягом третього року на 1500 шт. Реалізація продукції до впровадження розробки складала 1 шт., а її ціна до 630 грн. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [100 \cdot 1 + (630 + 100) \cdot 900] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right)$$
$$= 112250{,}09 \text{ грн.}$$

$$\Delta П_2 = [100 \cdot 1 + (630 + 100) \cdot (900 + 1000)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right)$$
$$= 237036{,}36 \text{ грн.}$$
$$\Delta П_3 = [100 \cdot 1 + (630 + 100) \cdot (900 + 1000 + 1500)] \cdot 0{,}833 \cdot 0{,}25$$
$$\cdot \left(1 + \frac{18}{100}\right) = 424091{,}37 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо основні показники, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахуємо величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки.

$$PV = k_{\text{інв}} \cdot \text{ЗВ}, \qquad (5.18)$$

$k_{\text{інв}}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо ($k_{\text{інв}}$ = 2…5).

$$PV = 2 \cdot 107328{,}73 = 214657{,}47$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$E_{абс} = (ПП - PV) \qquad (5.19)$$

де ПП — приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки, грн.;

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t},$$

(5.20)

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t — період часу (в роках).

$$ПП = \frac{112250,09}{(1+0,2)^1} + \frac{237036,36}{(1+0,2)^2} + \frac{424091,37}{(1+0,2)^3} = 504715,07 \text{ грн.}$$

$$Е_{абс} = (504715,07 - 214657,47) = 290057,61 \text{ грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1,$$

(5.21)

де $T_ж$ – життєвий цикл наукової розробки, роки.

$$\text{Е}_{\text{в}} = \sqrt[3]{1 + \frac{290057{,}61}{214657{,}47}} - 1 = 0{,}55 = 55\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f,$$

(5.22)

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d= (0,14…0,2);

f − показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{\min} = 0{,}18 + 0{,}05 = 0{,}23$$

Так як $E_в > \tau_{\min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в} \qquad (5.23)$$

$$\text{Т}_{\text{ок}} = \frac{1}{0{,}55} = 1{,}7 \text{ роки}$$

Так як $T_{ок} \leq$ 3...5-ти років, то фінансування даної наукової розробки в принципі є доцільним.

ВИСНОВКИ

У магістерській кваліфікаційній роботі було спроєктовано та розроблено програмно-апаратний комплекс для вивчення української мови з використанням інструментів штучного інтелекту. На основі аналізу сучасних підходів до побудови інтерактивних мовних систем сформовано вимоги до архітектури комплексу, який поєднує модулі автоматичного розпізнавання мовлення, аналізу граматики, генерації навчальних завдань та синтезу мовлення. Виконано дослідження існуючих систем і технологій, що дозволило визначити їхні переваги та недоліки й обґрунтувати необхідність створення інтегрованої системи, орієнтованої на індивідуальне та адаптивне навчання української мови.

У роботі розроблено структуру програмно-апаратного комплексу, що включає модулі реєстрації та авторизації користувача, модуль аналізу мовлення на основі моделі Whisper, модуль граматичного аналізу, підсистему генерації навчальних матеріалів і завдань, а також підсистему оцінювання результатів навчання. Окрему увагу приділено модулю синтезу мовлення, який дозволяє формувати аудіоматеріали для вимови та тренування слухового сприйняття. Для кожного програмного модуля наведено схеми взаємодії та інтерфейси, що забезпечують повноцінну інтеграцію підсистем у єдину функціональну архітектуру.

На основі аналізу алгоритмів машинного навчання було обґрунтовано вибір моделей для розпізнавання мовлення, синтезу звуку та граматичного аналізу. Створено концептуальну та логічну моделі даних, реалізовані у вигляді локальної бази SQLite, яка забезпечує зберігання профілів користувачів, історії навчання та результатів виконання завдань. Розроблено інтерфейс користувача з використанням фреймворку PyQt6, що забезпечує доступ до всіх навчальних функцій комплексу та зручну навігацію між ними.

Обґрунтовано вибір програмних засобів для реалізації комплексу.

Основною мовою програмування визначено Python завдяки широкій екосистемі бібліотек для штучного інтелекту та обробки аудіосигналів. Для розпізнавання мовлення використано модель Whisper, для синтезу — gTTS, а для побудови інтерфейсу — PyQt6. Окремо розглянуто інтеграцію комплексу з апаратною платформою Raspberry Pi для можливості подальшого створення портативного навчального пристрою.

Було проведено комплексне тестування розробленої системи. Здійснено модульне тестування окремих складових, зокрема модулів розпізнавання та синтезу мовлення, граматичного аналізу та генерації навчальних завдань. Проведено системне тестування “повного циклу роботи користувача”, де оцінювалася послідовність і коректність взаємодії всіх модулів. Підтверджено стабільність роботи комплексу, правильність формування результатів навчання та коректність збереження даних у базі.

Ключовими перевагами розробленого комплексу є інтегрована підтримка усіх основних аспектів навчання української мови: слухання, говоріння, граматики, виконання завдань і одержання статистики прогресу. Система використовує сучасні мовні моделі, забезпечує адаптивність завдань, автоматичну перевірку результатів, формування індивідуальних рекомендацій та можливість подальшого масштабування за рахунок додавання нових навчальних модулів.

Результати здійсненого технологічного аудиту вказують на рівень вище середнього комерційного потенціалу. У порівнянні з аналогічним виробом виявлено, що нова розробка вищої якості і більш конкурентоспроможна, як з технічних, так і економічних позначень.

Вкладені інвестиції в даний проект окупляться через 1,7 роки. Загальні витрати складають 107328,73 грн. Прогнозований прибуток за три роки складає 504715,07 грн.

Отже, усі задачі, поставлені у магістерській кваліфікаційній роботі, було виконано в повному обсязі, а розроблений програмно-апаратний комплекс

підтвердив свою ефективність як інструмент для інтерактивного та інтелектуального вивчення української мови.

Результати виконання магістерської кваліфікаційної роботи оформлені згідно з вимогами[31].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Тетерев В.І., Азарова А.О. Програмно-апаратний комплекс для вивчення української мови з використанням штучного інтелекту. Науково-технічна конференція ВНТУ Молодь в науці : Електронне наукове видання матеріалів конференції, м. Вінниця, 2025. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/26626

2. Thi-Nhu Ngo T, Hao-Jan Chen H, Kuo-Wei Lai K. The effectiveness of automatic speech recognition in ESL/EFL pronunciation: A meta-analysis. ReCALL. 2024. 36(1). P. 4 – 21. [Електронний ресурс]. Режим доступу: https://doi.org/10.1017/S0958344023000113

3.Shadiev, R., Liu, T. (2023). Review of speech recognition technology in assistive language learning [Електронний ресурс]. Режим доступу: https://doi.org/10.1017/S095834402200012X

4. Hou, Z., Min, S. (2025). Dialogue-based computer-assisted language learning systems for second language speaking development: A three-level meta-analysis [Електронний ресурс]. Режим доступу: https://doi.org/10.1017/S0958344025000059

5. Jiang, H., Chen, S., Huang, X. (2023). Mobile-based dictation: How ASR technology facilitates oral accuracy and fluency… [Електронний ресурс]. Режим доступу: https://doi.org/10.1111/jcal.12732

6. Sun, J., Liu, J., Zhang, Y. et al. (2023). Automatic speech recognition combined with peer correction improves L2 pronunciation… [Електронний ресурс]. Режим доступу: https://doi.org/10.3389/fpsyg.2023.1210187

7. Evers, K., Chen, S. (2022). Effects of an automatic speech recognition system with peer feedback on pronunciation instruction for adults [Електронний ресурс]. Режим доступу: https://doi.org/10.1080/09588221.2020.1839504

8. Evers, K., Chen, S. (2021). Effects of Automatic Speech Recognition Software on Pronunciation for Adults With Different Learning Styles [Електронний ресурс]. Режим доступу: https://doi.org/10.1177/0735633120972011

9. Timpe-Laughlin, V., Sydorenko, T., Daurio, P. (2022). Using spoken dialogue technology for L2 speaking practice: What do teachers think? [Електронний ресурс]. Режим доступу: https://doi.org/10.1080/09588221.2020.1774904

10. Huang, W., Hew, K. F., Fryer, L. K. (2022). Chatbots for language learning—Are they really useful? A systematic review… [Електронний ресурс]. Режим доступу: https://doi.org/10.1111/jcal.12610

11. Jeon, J. (2024). Exploring AI chatbot affordances in the EFL classroom: young learners' experiences… [Електронний ресурс]. Режим доступу: https://doi.org/10.1080/09588221.2021.2021241

12. Annamalai, N., Rashid, R. A., Hashmi, U. M. et al. (2023). Using chatbots for English language learning in higher education [Електронний ресурс]. Режим доступу: https://doi.org/10.1016/j.caeai.2023.100153

13. Annamalai, N., Eltahir, M. E., Zyoud, S. H. et al. (2023). Exploring English language learning via chatbot: A self-determination theory perspective [Електронний ресурс]. Режим доступу: https://doi.org/10.1016/j.caeai.2023.100148

14. Du, J., Daniel, B. K. (2024). Transforming language education: A systematic review of AI-powered chatbots for EFL speaking practice [Електронний ресурс]. Режим доступу: https://doi.org/10.1016/j.caeai.2024.100230

15. Lai, W. Y. W., Lee, J. S. (2024). A systematic review of conversational AI tools in ELT… [Електронний ресурс]. Режим доступу: https://doi.org/10.1016/j.caeai.2024.100291

16. Shi, H., Chai, C. S., Zhou, S., Aubrey, S. (2025). Comparing the effects of ChatGPT and automated writing evaluation on students' writing… [Електронний ресурс]. Режим доступу: https://doi.org/10.1080/09588221.2025.2454541

17. Google Speech-to-Text API [Електронний ресурс]. Режим доступу: https://cloud.google.com/speech-to-text

18. Docker Inc. Docker Documentation: Microservices Architecture

[Електронний ресурс]. Режим доступу: https://docs.docker.com/get-started/overview/

19. Jurafsky D., Martin J. H. Speech and Language Processing. 3rd ed. Draft. Stanford University, 2023 [Електронний ресурс]. Режим доступу: https://web.stanford.edu/~jurafsky/slp3/

20. Brown T. B., et al. Language Models are Few-Shot Learners. NeurIPS, 2020 [Електронний ресурс]. Режим доступу: https://arxiv.org/abs/2005.14165

21. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL, 2019 [Електронний ресурс]. Режим доступу: https://arxiv.org/abs/1810.04805

22. Vaswani A., et al. Attention Is All You Need. Advances in Neural Information Processing Systems, 2017 [Електронний ресурс]. Режим доступу: https://arxiv.org/abs/1706.03762

23. Hugging Face Model Hub. Ukrainian language models and datasets [Електронний ресурс]. Режим доступу: https://huggingface.co

24. Single Board Computer Market: Press Release. Publ. 2022, Nov. 11 [Електронний ресурс]. Режим доступу: https://www.marketwatch.com/press-release/single-board-computer-marketsize-trends-growth-status-share-research-and-forecast-2030-2022-11-11?mod=search_headline

25. NVIDIA Corporation. Jetson Nano Developer Kit — AI Education Platform [Електронний ресурс]. Режим доступу: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

26. Orange Pi 5 [Електронний ресурс]. Режим доступу: http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/details/Orange-Pi-5.html

27. Google Coral. Edge TPU for On-device Machine Learning [Електронний ресурс]. Режим доступу: https://coral.ai

28. Mozilla Common Voice. Ukrainian speech datasets [Електронний ресурс]. Режим доступу: https://commonvoice.mozilla.org

29. Radford A., et al. Robust Speech Recognition via Large-Scale Weak Supervision. OpenAI, 2022 [Електронний ресурс]. Режим доступу: https://arxiv.org/abs/2212.04356

30. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. — Вінниця : ВНТУ, 2021. — 42 с.

31. Методичні вказівки до виконання магістерських кваліфікаційних робіт студентами спеціальності 123 “Комп’ютерна інженерія”. / Укладачі О.Д. Азаров, О.В. Дудник, С.І. Швець − Вінниця : ВНТУ, 2023. − 57 с.

ДОДАТОК А

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри обчислювальної техніки

_____проф., д.т.н. О. Д. Азаров

«___» _____ 2025 року

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

Програмно-апаратний комплекс для вивчення української мови на основі штучного інтелекту

Науковий керівник к.т.н., проф. каф. ОТ

_____ Азарова А. О.

Студента групи 1КІ-24м

_____ Тетерева В. І.

ВНТУ 2025

1 Підставою для виконання магістерської кваліфікаційної роботи є наказ про затвердження теми дипломної роботи, а також актуальність створення програмно-апаратного комплексу для вивчення української мови, який забезпечує інтерактивне навчання, автоматичне оцінювання результатів та підвищення ефективності засвоєння мовного матеріалу.

2 Мета і призначення МКР:

— удосконалення процесу вивчення української мови шляхом розроблення програмно-апаратного комплексу;

— призначенням розробки є виконання магістерської кваліфікаційної роботи з можливістю подальшого впровадження та масштабування.

3 Вихідні дані для виконання МКР:

— розробити програмно-апаратний комплекс для вивчення української мови з використанням штучного інтелекту;

— основна технологія PyQt6;

— середовище програмування PyCharm 2025;

— мова програмування Phython, SQL.

4 Технічні вимоги до виконання МКР:

— розроблення модульної програмної частини для обробки мовлення, аналізу тексту та генерації навчальних завдань.

— створення інтерфейсу користувача.

5 Етапи МКР та очікувані результати (табл. А.1).

Таблиця А.1 — Етапи роботи

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз методів обробки мовлення та мовних технологій ІШІ.	01.09.2025	09.09.2025	Розділ 1
2	Проєктування архітектури комплексу та його функціональних модулів	12.09.2025	26.09.2025	Розділ 2

Продовження таблиці А.1

3	Вибір технологій розроблення та підготовка технічних рішень	3.10.2025	10.10.2025	Розділ 3 частково
4	Реалізація модулів комплексу та інтеграція системи	17.10.2025	31.10.2025	Розділ 3 повністю
5	Підготовка економічної частини	7.11.2025	14.11.2025	Розділ 4
6	Оформлення матеріалів до захисту МКР	21.11.2025	1.12.2025	Пояснювальна записка, графічний матеріал, лістинг, презентація

6 Матеріали, що подаються до захисту МКР:

— пояснювальна записка МКР;

— графічні і ілюстративні матеріали;

— протокол попереднього захисту МКР на кафедрі;

— відгук наукового керівника;

— рецензія на виконану роботу;

— анотації до МКР українською та іноземною мовами;

— нормоконтроль про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР:

— виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами;

— захист МКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення МКР викладені в методичних вказівках до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія, ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання», ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання».

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Програмно-апаратний комплекс для вивчення української мови з використанням штучного інтелекту

Тип роботи: магістерська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра обчислювальної техніки, ФІТКІ, 1КІ-24м
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) 1%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.
- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- ☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

Азаров О. Д., д.т.н., зав. каф. ОТ _____
(прізвище, ініціали, посада) (підпис)

Мартинюк Т. Б., д.т.н., проф. каф. ОТ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Захарченко С. М.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Азарова А.О., к.т.н., проф. каф. ОТ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Тетерев В.І.
(підпис) (прізвище, ініціали)

ДОДАТОК В

Структура програмно-апаратного комплексу

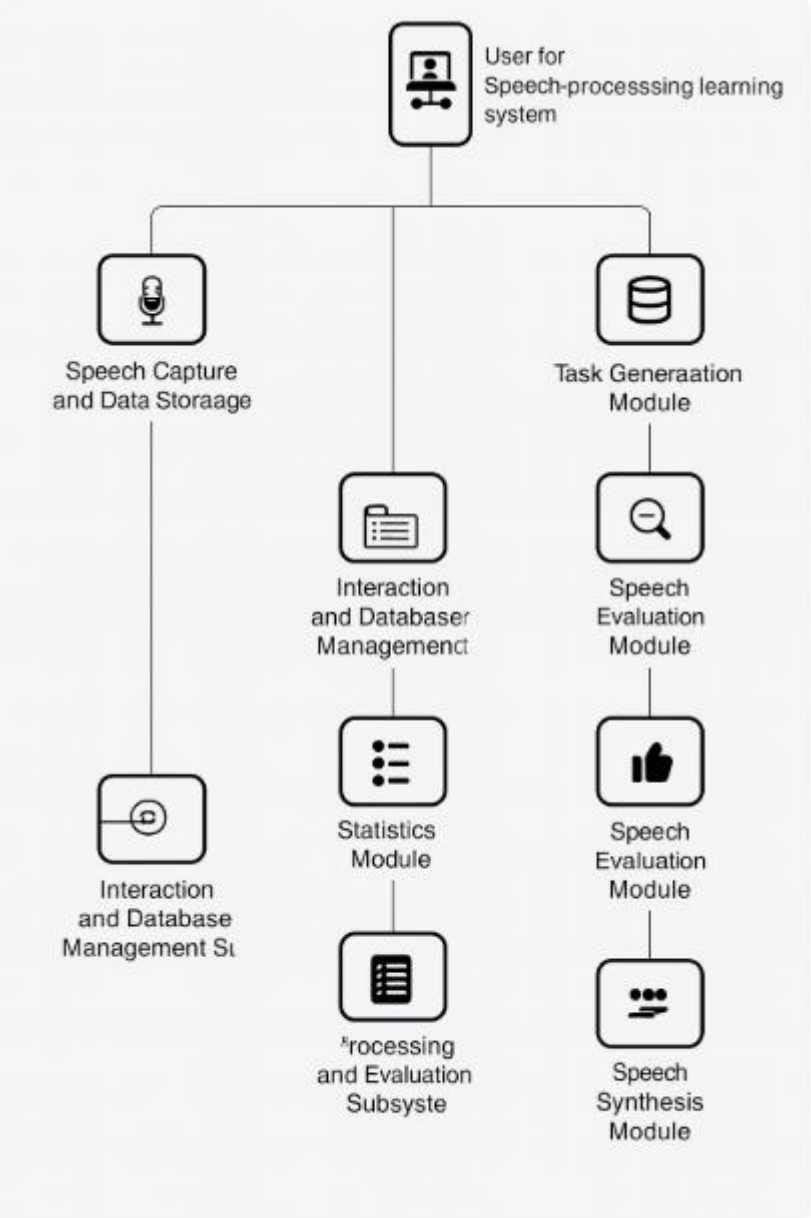

Рисунок В.1 — Структурна схема програмно-апаратного комплексу

ДОДАТОК Г

Лістинг модуля розпізнавання мовлення

```
def start_recording(self):
    self.recorder = AudioRecorder()
    self.recorder.finished.connect(self.save_recording)
    self.mic_status.setText("Запис триває...")
    self.mic_status.setStyleSheet("color: green; font-weight: bold;")
    self.start_btn.setEnabled(False)
    self.stop_btn.setEnabled(True)
    self.recorder.start()
    self.status.showMessage("Запис розпочато")
def stop_recording(self):
    if hasattr(self, "recorder"):
        self.recorder.stop()
    self.mic_status.setText("Мікрофон неактивний")
    self.mic_status.setStyleSheet("color: gray;")
    self.start_btn.setEnabled(True)
    self.stop_btn.setEnabled(False)
    self.status.showMessage("Запис завершено")
def save_recording(self, audio_data):
    import os, wave, pyaudio
    os.makedirs("data/recordings", exist_ok=True)
    output_file = "data/recordings/record.wav"
    wf = wave.open(output_file, "wb")
    wf.setnchannels(1)
    wf.setsampwidth(pyaudio.PyAudio().get_sample_size(pyaudio.paInt16))
    wf.setframerate(44100)
    wf.writeframes(audio_data)
    wf.close()
    self.status.showMessage(f"Аудіо збережено: {output_file}")
    recognizer = SpeechRecognizer()
    text = recognizer.transcribe(output_file)
    self.speech_text.setText(text)
    self.status.showMessage("Розпізнавання завершено ✅")
def create_speech_tab(self):
    widget = QWidget()
    layout = QVBoxLayout()
    from PyQt6.QtWidgets import QGroupBox
    text_group = QGroupBox("Розпізнане мовлення")
    text_layout = QVBoxLayout()
    self.speech_text = QTextEdit()
    self.speech_text.setPlaceholderText("Тут з’явиться розпізнаний текст...")
```

```
        self.speech_text.setReadOnly(True)
        text_layout.addWidget(self.speech_text)
        text_group.setLayout(text_
        record_group = QGroupBox("Керування записом")
        record_layout = QHBoxLayout()
        self.start_btn = QPushButton("Почати запис")
        self.stop_btn = QPushButton("Завершити запис")
        self.stop_btn.setEnabled(False)
        self.start_btn.setIcon(qta.icon("mdi.play"))
        self.stop_btn.setIcon(qta.icon("mdi.stop"))
        self.start_btn.clicked.connect(self.start_recording)
        self.stop_btn.clicked.connect(self.stop_recording)
        self.mic_status = QLabel("Мікрофон неактивний")
        self.mic_status.setStyleSheet("color: gray;")
        record_layout.addWidget(self.start_btn)
        record_layout.addWidget(self.stop_btn)
        record_layout.addWidget(self.mic_status)
        record_group.setLayout(record_layout)
        ai_group = QGroupBox("Аналіз усного мовлення (AI)")
        ai_layout = QVBoxLayout()
        self.evaluate_btn = QPushButton("□ Оцінити мовлення (AI)")
        self.evaluate_btn.setIcon(qta.icon("mdi.brain"))
        self.evaluate_btn.clicked.connect(self.evaluate_speech_ai)
        self.speech_feedback = QTextEdit()
        self.speech_feedback.setReadOnly(True)
        self.speech_feedback.setPlaceholderText("Тут з’явиться оцінка мовлення...")
        ai_layout.addWidget(self.evaluate_btn)
        ai_layout.addWidget(self.speech_feedback)
        ai_group.setLayout(ai_layout)
        layout.addWidget(text_group)
        layout.addWidget(record_group)
        layout.addWidget(ai_group)
        widget.setLayout(layout)
        return widget
    class SpeechRecognizer:
        def __init__(self, model_path="models/vosk-uk"):
            self.model = Model(model_path)
        def transcribe(self, wav_file):
            wf = wave.open(wav_file, "rb")
            rec = KaldiRecognizer(self.model, wf.getframerate())
            text = ""
            while True:
                data = wf.readframes(4000)
```

```
            if len(data) == 0:
                break
            if rec.AcceptWaveform(data):
                text += json.loads(rec.Result())["text"] + " "
        text += json.loads(rec.FinalResult())["text"]
        return text.strip()
# audio/recorder.py
from PyQt6.QtCore import QThread, pyqtSignal
import pyaudio
import wave
import io
class AudioRecorder(QThread):
    finished = pyqtSignal(bytes)
    def __init__(self, parent=None):
        super().__init__(parent)
        self.running = False
        self.frames = []
        self.chunk = 1024
        self.format = pyaudio.paInt16
        self.channels = 1
        self.rate = 44100
    def run(self):
        self.running = True
        self.frames = []
        audio = pyaudio.PyAudio()
        stream = audio.open(format=self.format,
                            channels=self.channels,
                            rate=self.rate,
                            input=True,
                            frames_per_buffer=self.chunk)
        while self.running:
            data = stream.read(self.chunk)
            self.frames.append(data)
        stream.stop_stream()
        stream.close()
        audio.terminate()
        audio_data = io.BytesIO()
        wf = wave.open(audio_data, 'wb')
        wf.setnchannels(self.channels)
        wf.setsampwidth(audio.get_sample_size(self.format))
        wf.setframerate(self.rate)
        wf.writeframes(b".join(self.frames))
        wf.close()
```

```
        self.finished.emit(audio_data.getvalue())
    def stop(self):
        self.running = False
# ai/speech_evaluator.py
def evaluate_speech(text: str) -> str:
    """
    Проста евристична функція оцінки мовлення.
    Порівнює розпізнаний текст із базовими критеріями правильності.
    Повертає зворотний зв'язок для користувача.
    """
    if not text:
        return "⚠ Текст не виявлено. Спробуйте ще раз."
    feedback = []
    word_count = len(text.split())
    if word_count < 5:
        feedback.append("💬 Ви сказали дуже мало слів. Намагайтеся формулювати повніші речення.")
    if text[0].islower():
        feedback.append("🔤 Речення має починатися з великої літери.")
    if not text.endswith(('.', '!', '?')):
        feedback.append("🔑 Речення має закінчуватися розділовим знаком.")
    if not feedback:
        return "✅ Ваше мовлення виглядає добре! Продовжуйте в тому ж дусі."
    return "\n".join(feedback)
```

ДОДАТОК Д

Лістинг модуля граматичного аналізу

```
def create_grammar_tab(self):
    widget = QWidget()
    layout = QVBoxLayout()
    self.grammar_text = QTextEdit()
    self.grammar_text.setPlaceholderText("Вставте текст для аналізу або виправлення...")
    self.grammar_result = QTextEdit()
    self.grammar_result.setReadOnly(True)
    analyze_btn = QPushButton("🔍 Проаналізувати (AI)")
    analyze_btn.clicked.connect(self.run_grammar_check)
    correct_btn = QPushButton("✅ Виправити текст (AI)")
    correct_btn.clicked.connect(self.auto_correct)
    layout.addWidget(self.grammar_text)
    layout.addWidget(analyze_btn)
    layout.addWidget(correct_btn)
    layout.addWidget(self.grammar_result)
    widget.setLayout(layout)
    return widget
def run_grammar_check(self):
    text = self.grammar_text.toPlainText().strip()
    if not text:
        self.status.showMessage("Поле порожнє")
        return
    try:
        self.status.showMessage("AI аналізує текст...")
        analysis = ai_analyze(text)
        self.grammar_result.setText(analysis)
        self.status.showMessage("Аналіз граматики завершено ✅")
    except Exception as e:
        QMessageBox.warning(self, "Помилка AI", f"Не вдалося виконати аналіз:\n{e}")
        self.status.showMessage("Помилка під час аналізу AI")
def auto_correct(self):
    text = self.grammar_text.toPlainText().strip()
    if not text:
        self.status.showMessage("Поле порожнє")
        return
    try:
        self.status.showMessage("AI виконує виправлення...")
```

```
        corrected = ai_correct(text)
        self.grammar_result.setText(corrected)
        self.status.showMessage("AI виправив текст ✅")
    except Exception as e:
        QMessageBox.warning(self, "Помилка AI", f"Не вдалося виправити текст:\n{e}")
        self.status.showMessage("Помилка під час виправлення AI")
import torch
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer
model_name = "ainize/bart-base-cnn-german"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
def ai_analyze(text):
    return f"AI: Виявлено помилки у реченні: '{text}'"
def ai_correct(text):
    inputs = tokenizer([text], return_tensors="pt", truncation=True)
    with torch.no_grad():
        outputs = model.generate(**inputs, max_length=256)
    return tokenizer.decode(outputs[0], skip_special_tokens=True)
from PyQt6.QtWidgets import (
    QMainWindow, QWidget, QVBoxLayout, QTextEdit, QPushButton,
    QMenuBar, QStatusBar, QMessageBox
)
from ai.corrector import ai_analyze, ai_correct
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("AI-комплекс для вивчення української мови")
        self.resize(1000, 700)
        self.status = QStatusBar()
        self.setStatusBar(self.status)
        self.init_menu()
        self.init_tabs()
def init_menu(self):
    menu_bar = self.menuBar()
    file_menu = menu_bar.addMenu("Файл")
    help_menu = menu_bar.addMenu("Допомога")
    about_action = QAction("Про програму", self)
    help_menu.addAction(about_action)
    about_action.triggered.connect(self.show_about)
def show_about(self):
    QMessageBox.information(
        self,
```

```
        "Про застосунок",
        "AI-комплекс для вивчення української мови\n"
        "Розробник: Тетерев В.І., 2025"
    )
def init_tabs(self):
    tabs = QTabWidget()
    tabs.addTab(self.create_grammar_tab(), qta.icon("mdi.magnify"), "Граматика")
    self.setCentralWidget(tabs)
def apply_theme(self, theme_name):
    if theme_name == "Темна":
        self.setStyleSheet(DARK_THEME)
    else:
        self.setStyleSheet(LIGHT_THEME)
    self.status.showMessage(f"Застосована тема: {theme_name}")
def validate_text_input(self, text):
    if not text.strip():
        QMessageBox.warning(self, "Помилка", "Поле не може бути порожнім.")
        self.status.showMessage("Введіть текст для аналізу або виправлення")
        return False
    return True
def show_analysis_result(self, result_text):
    self.grammar_result.setText(result_text)
    self.status.showMessage("Результат обробки виведено")
def open_file(self):
    file_path, _ = QFileDialog.getOpenFileName(
        self,
        "Відкрити текстовий файл",
        "",
        "Text Files (*.txt);;Усі файли (*.*)"
    )
    if file_path:
        try:
            with open(file_path, "r", encoding="utf-8") as f:
                content = f.read()
            self.speech_text.setText(content)
            self.status.showMessage(f"Файл відкрито: {file_path}")
        except Exception as e:
            QMessageBox.warning(self, "Помилка", f"Не вдалося прочитати
файл:\n{e}")
```

ДОДАТОК Е

Лістинг модуля генерації навчальних завдань

```
def create_tasks_tab(self):
    widget = QWidget()
    layout = QVBoxLayout(widget)
    title = QLabel("📝 Тестові завдання з української мови")
    title.setStyleSheet("font-size: 18px; font-weight: bold; margin-bottom: 10px;")
    layout.addWidget(title)
    self.task_type = QComboBox()
    self.task_type.addItems(["Граматика", "Орфографія", "Лексика",
"Переклад"])
    layout.addWidget(self.task_type)
    self.task_difficulty = QComboBox()
    self.task_difficulty.addItems(["Початковий", "Середній", "Просунутий"])
    layout.addWidget(self.task_difficulty)
    self.tasks_text = QTextEdit()
    self.tasks_text.setReadOnly(True)
    self.tasks_text.setPlaceholderText("Тут з’явиться згенероване AI-завдання...")
    layout.addWidget(self.tasks_text)
    self.answer_group = QButtonGroup(widget)
    self.opt_a = QRadioButton("а)")
    self.opt_b = QRadioButton("б)")
    self.opt_c = QRadioButton("в)")
    self.opt_d = QRadioButton("г)")
    self.answer_group.addButton(self.opt_a, 1)
    self.answer_group.addButton(self.opt_b, 2)
    self.answer_group.addButton(self.opt_c, 3)
    self.answer_group.addButton(self.opt_d, 4)
    layout.addWidget(self.opt_a)
    layout.addWidget(self.opt_b)
    layout.addWidget(self.opt_c)
    layout.addWidget(self.opt_d)
    self.opt_a.hide()
    self.opt_b.hide()
    self.opt_c.hide()
    self.opt_d.hide()
    self.start_test_btn = QPushButton("🚀 Почати тест")
    self.start_test_btn.clicked.connect(self.start_test)
    self.check_btn = QPushButton("✅ Перевірити відповідь")
    self.check_btn.clicked.connect(self.check_task_answer_ai)
    self.check_btn.hide()
```

```
    layout.addWidget(self.start_test_btn)
    layout.addWidget(self.check_btn)
    return widget
def generate_task_ai(self):
    try:
        from ai.task_generator import generate_task
        task_type = self.task_type.currentText()
        difficulty = self.task_difficulty.currentText()
        self.status.showMessage("AI генерує завдання...")
        task_text, correct = generate_task(task_type, difficulty)
        self.tasks_text.setText(task_text)
        self.correct_answer = correct
        self.opt_a.show()
        self.opt_b.show()
        self.opt_c.show()
        self.opt_d.show()
        self.check_btn.show()
        self.answer_group.setExclusive(False)
        self.opt_a.setChecked(False)
        self.opt_b.setChecked(False)
        self.opt_c.setChecked(False)
        self.opt_d.setChecked(False)
        self.answer_group.setExclusive(True)
        self.status.showMessage("Завдання згенеровано ✅")
    except Exception as e:
        QMessageBox.warning(self, "Помилка AI", str(e))
        self.status.showMessage("Не вдалося згенерувати завдання ❌")
def start_test(self):
    """Почати новий тест із 10 питань"""
    self.test_active = True
    self.current_question = 1
    self.correct_count = 0
    self.start_test_btn.hide()
    self.generate_task_ai()
    self.opt_a.show()
    self.opt_b.show()
    self.opt_c.show()
    self.opt_d.show()
    self.check_btn.show()
    self.next_btn.hide()
    self.status.showMessage(f"Питання {self.current_question} із
{self.total_questions}")
def check_task_answer_ai(self):
```

```
    task = self.tasks_text.toPlainText().strip()
    selected = self.get_selected_answer()

    if not task:
        QMessageBox.warning(self, "Помилка", "Спочатку згенеруйте завдання ✅")
        return
    if not selected:
        QMessageBox.warning(self, "Помилка", "Оберіть варіант відповіді ✅")
        return
    if not self.correct_answer:
        QMessageBox.warning(self, "Помилка", "Відсутня правильна відповідь ⚠")
        return
    if selected == self.correct_answer:
        QMessageBox.information(self, "Результат", "✅ Правильно!")
        self.correct_count += 1
    else:
        QMessageBox.information(
            self,
            "Результат",
            f"❌ Неправильно.\nПравильна відповідь: {self.correct_answer}"
        )
    self.check_btn.hide()
    self.next_btn.show()
def get_selected_answer(self):
    checked = self.answer_group.checkedId()
    if checked == -1:
        return None
    return {1: "а", 2: "б", 3: "в", 4: "г"}[checked]
```

ДОДАТОК Ж

УЗГОДЖЕНО
Декан факультету інформаційних технологій та комп'ютерної інженерії, к.пед.н., доцент
_____Світлана КИРИЛАЩУК
«15» грудня 2025 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри мовознавства Вінницького національного технічного університету
_____Лариса АЗАРОВА
«15» грудня 2025 р.

АКТ ВПРОВАДЖЕННЯ № 1

результатів науково – дослідних робіт

Замовник кафедра мовознавства Вінницького національного технічного університету. Цим актом підтверджується, що результати роботи – «**Програмно-апаратний комплекс для вивчення української мови з використанням штучного інтелекту**», яку виконано студентом гр. 1КІ – 24М, Вінницького національного технічного університету, Тетеревим В. І. за договором про творчу співдружність без взаємних грошових розрахунків, на громадських засадах, впроваджено на кафедрі мовознавства Вінницького національного технічного університету.

1.Вид впроваджених результатів експлуатація технології

2. Характеристика масштабу впровадження _____унікальне_____

3. Форма впровадження _____програмний продукт _____

4. Новизна результатів науково – дослідної роботи модернізація старих розробок
(піонерські, принципово нові, якісно нові, модифікації, модернізація старих розробок)

5. Впроваджені: _____у навчальному процесі кафедри мовознавства_____

6. Річний економічний ефект__ не розраховувався_____

7. Соціальний та науково-технічний ефект спеціальне призначення (удосконалення методу навчання української мови)

Від виконавця: **Від зво:**

_____Тетерев В. І. _____ Азарова Л. Є.