Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: Розробка програмного забезпечення для розвиваючої вебсистеми з вивчення англійської мови

Виконав: студент 4 курсу
групи 2ПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки спеціальності)
Причепа В.О.
(прізвище та ініціали)
Керівник: PhD, ст. вик. каф. ПЗ Стахов О.Я.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент: к.т.н, доц. каф. ОТ Крупельницький Л.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри ПЗ
д.т.н., проф. Романюк О. Н.
«09» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
ПЗ О. Н. Романюк
« 21 » березня 2025 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Причепі Владиславу Олеговичу

1. Тема роботи «Розробка програмного забезпечення для розвиваючої вебсистеми з вивчення англійської мови».

Керівник роботи: Стахов Олексій Ярославович, доктор філософії PHD, старш.викл. кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. № 97.

2. Строк подання студентом роботи 01 червня 2025.

3. Вихідні дані до роботи: тип програми вебзастосунок; модель розробки ітеративна; засоби організації даних програми хмарна та віртуальна бази даних; вхідні дані: мовлення користувача, вибрані варіанти у тестах; вихідні дані: оцінка вимови, основні граматичні правила, персоналізовані рекомендації для покращення, звіти після виконання, прогрес; середовище розробки JetBrains WebStorm; мова програмування JavaScript.

4. Зміст текстової частини: вступ; обґрунтування вибору методу розробки та постановка задач дослідження; розробка моделі та алгоритмів програмного застосунку; розробка функціональних модулів застосунку; тестування програмного застосунку; інструкція користувача; висновки; список використаних джерел; додатки; графічна частина.

5. Перелік ілюстративного матеріалу: титульний слайд; мета, предмет та об'єкт дослідження; актуальність розробки; постановка задач; новизна отриманих результатів; практична цінність; порівняння існуючих аналогів; використані технології; модель роботи вебсистеми; блок-схема методу формування та відображення загального прогресу; демонстрація роботи вебсистеми; апробація та публікація матеріалів бакалаврської

кваліфікаційної роботи; висновки; фінальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Стахов О.Я.	24.03.2025	01.06.2025

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану питання та постановка задач дослідження	25.03.2025 09.04.2025	вик.
2	Аналіз вхідних та вихідних даних системи	10.04.2025 17.04.2025	вик.
4	Розробка моделі та алгоритмів системи	18.04.2025 27.04.2025	вик.
5	Розробка інтерфейсу	28.04.2025 05.05.2025	вик.
6	Розробка функціональних модулів	06.05.2025 16.05.2025	вик.
7	Тестування вебсистеми	17.05.2025 21.05.2025	вик.
8	Оформлення матеріалів до захисту БКР	22.05.2025 30.05.2025	вик.

Студент _____ Причепа В.О.
(підпис) (ініціали та прізвище)

Керівник бакалаврської кваліфікаційної роботи _____ Стахов О.Я.
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 57 сторінок формату А4, на яких є 27 рисунків, 2 таблиці, список використаних джерел містить 20 найменувань.

У бакалаврській кваліфікаційній роботі було проаналізовано сферу навчальних вебсистем для вивчення іноземних мов, а зокрема англійської. Проведено аналіз аналогів, визначено їх переваги і недоліки та доведено актуальність створення власної розробки.

Розроблена вебсистема дозволяє вивчати англійську мову шляхом виконання вправ різних типів, наприклад: тестові завдання, граматика або тренування мовлення.

Вебсистему було створено за допомогою мови програмування JavaScript, з використанням стилів CSS та мови гіпертекстової розмітки HTML. База даних реалізована за допомогою сервісу Firebase, а зокрема інструментом Firestore Database. Середовищем розробки було обрано JetBrains WebStorm.

В результаті виконання бакалаврської кваліфікаційної роботи було розроблено розвиваючу вебсистему, що підвищить ефективність процесу вивчення англійської мови.

Отримані в бакалаврській кваліфікаційній роботі результати можна використати для покращення процесу вивчення англійської мови.

Ключові слова: вебсистема, вивчення англійської мови, онлайн-навчання, JavaScript, HTML, CSS.

ABSTRACT

The bachelor's thesis consists of 57 A4 pages, with 27 figures, 2 tables, and a list of references containing 20 titles.

The bachelor's thesis analyzes the field of educational web systems for learning foreign languages, in particular English. The author analyzed analogs, identified their advantages and disadvantages, and proved the relevance of creating his own development.

The developed web system allows you to learn English by performing exercises of various types, such as: test tasks, grammar or speech training.

The web system was created with the help of the JavaScript programming language, using CSS styles and the HTML hypertext markup language. The database was implemented with the help of the Firebase service, and in particular, the Firestore Database tool. JetBrains WebStorm was chosen as the development environment.

As a result of the bachelor's thesis, a web-based development system was developed that will increase the efficiency of the English language learning process.

The results obtained in the bachelor's thesis can be used to improve the process of learning English.

Keywords: web system, learning English, online learning, JavaScript, HTML, CSS.

ЗМІСТ

ВСТУП ... 3
1. АНАЛІЗ РОЗВИТКУ СИСТЕМ ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ ... 6
1.1 Аналіз стану вебсистем для вивчення англійської мови ... 6
1.2 Порівняльний аналіз аналогів ... 7
1.3 Аналіз методів розв'язання задачі ... 10
1.4 Постановка задач для розробки розвиваючої вебсистеми ... 11
1.5 Висновки ... 12
2 РОЗРОБКА МОДЕЛІ ТА АЛГОРИТМІВ ВЕБСИСТЕМИ ... 13
2.1 Аналіз даних та методів розробки ... 13
2.2 Розробка моделі системи ... 13
2.3 Розробка методу для формування та відображення загального прогресу ... 16
2.4 Розробка загального алгоритму роботи вебсистеми ... 19
2.5 Висновки ... 21
3 РОЗРОБКА МОДУЛІВ ВЕБСИСТЕМИ ... 22
3.1 Варіантний аналіз і обґрунтування вибору засобі для реалізації вебсистеми ... 22
3.2 Розробка модуля для розпізнавання мовлення ... 26
3.3 Розробка модуля для вивчення граматики ... 28
3.4 Розробка модуля для проходження тестів ... 31
3.5 Розробка структури інтерфейсу вебсистеми ... 34
3.6 Розробка інтерфейсу вебсистеми ... 37
3.7 Висновки ... 43
4 ТЕСТУВАННЯ ВЕБСИСТЕМИ ... 46
4.1 Аналіз методів тестування ... 46
4.2 Тестування вебсистеми ... 48
4.3 Розробка інструкції користувача ... 53
4.3 Висновки ... 54
ВИСНОВКИ ... 55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ... 56
ДОДАТКИ ... 58
ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ ... 59
ДОДАТОК Б – ПЕРЕВІРКА НА ПЛАГІАТ ... 62
ДОДАТОК В – ЛІСТИНГ ПРОГРАМИ ... 63
ДОДАТОК Г – ІЛЮСТРАТИВНА ЧАСТИНА ... 91

ВСТУП

Сучасні інформаційні технології, що постійно розвиваються, створюють нові можливості для вдосконалення методів вивчення іноземних мов, зокрема англійської. З появою легкого доступу до інтернету, навчання стало доступнішим і гнучкішим, однак традиційні методи, що базуються на використанні статичних матеріалів, часто не можуть задовольнити потреби сучасного учня сповна. Вони не здатні адаптуватися до індивідуальних особливостей, швидкості засвоєння матеріалу та специфічних потреб кожного користувача. Ці обмеження спонукають до розробки нових інтерактивних веб рішень, які можуть ефективно поєднувати сучасні технології з інноваційними підходами до навчання.

Англійська мова, як глобальний засіб комунікації, має ключове значення для професійного зростання, академічних досягнень та соціальної інтеграції, що робить актуальним пошук ефективних способів її вивчення.

Сучасні вебтехнології та методи штучного інтелекту дозволяють створювати інтелектуальні системи, здатні адаптуватися до рівня знань, темпу навчання та інтересів користувача. Розробка вебсистеми з вивчення англійської мови, яка поєднує різні вправи, аналіз прогресу та автоматизовані рекомендації, може значно підвищити ефективність навчання порівняно зі статичними курсами чи класичними підручниками.

Враховуючи усе вищеперераховане, обрана тема є актуальною через поєднання поточної значущості вивчення англійської мови з технологічними можливостями створення інноваційних навчальних платформ. Її реалізація може сприяти розвитку електронної освіти та стати корисною для всіх, хто прагне покращити свої мовні навички.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета дослідження. Метою дослідження є покращення умов для вивчення англійської мови, за рахунок використання інтерактивного підходу до вивчення, що охоплює різні аспекти. Система передбачає розробку різноманітних модулів для навчання, включаючи вивчення граматики, різні види тестів та розвитку навичок сприйняття мови на слух, що дозволяє користувачам не лише засвоювати матеріал, але й автоматично отримувати зворотний зв'язок про свої досягнення та області, які потребують додаткової уваги.

Задачі дослідження:

1. Розробити модель розвиваючої вебсистеми для вивчення англійської мови, яка відображатиме основний функціонал застосунку.

2. Реалізувати необхідний функціонал вебсистеми для вивчення англійської мови.

3. Розробити привабливий та зручний інтерфейс користувача.

4. Провести тестування вебсистеми.

5. Розробити інструкцію користувача.

Об'єктом дослідження є процес розробки розвиваючої вебсистеми для вивчення англійської мови.

Предметом дослідження є методи та засоби реалізації розвиваючої вебсистеми для вивчення англійської мови.

Методи дослідження. У процесі дослідження використовувались: теорія алгоритмів та структур даних для розробки ефективних методів обробки навчальних матеріалів; методи об'єктно-орієнтованого програмування та патернів проектування для створення модульної архітектури вебсистеми; теорія людино-машинної взаємодії та принципи UX/UI дизайну для розробки інтуїтивного інтерфейсу користувача; методи розпізнавання та синтезу мовлення на основі Web Speech API для реалізації модуля перевірки вимови; теорія баз даних та методи синхронізації для організації зберігання даних у Firebase Firestore з резервним копіюванням у localStorage; методи тестування для перевірки коректності роботи всіх модулів системи.

Новизна отриманих результатів:

1. Вперше запропоновано алгоритм комбінованої оцінки точності вимови, який поєднує посимвольне порівняння з аналізом лексичної схожості та враховує рівень складності фрази, що дозволяє підвищити точність оцінювання вимови та надавати персоналізовані рекомендації щодо покращення мовленнєвих навичок.

2. Подальшого розвитку отримав метод візуалізації прогресу навчання, в якому, на відміну від існуючих рішень, реалізовано динамічну систему розрахунку поточного рівня володіння мовою з анімованим відображенням статистики та інтеграцією з системою досягнень, що підвищує мотивацію користувачів та збільшує їх залученість у навчальний процес.

Практична цінність. Практична цінність розробки полягає в її можливостях для використання у різних сферах освіти. Вебсистема може бути застосована в рамках шкільних програм, вищих навчальних закладів, а також для корпоративних тренінгів, що дозволить значно покращити процес навчання. Завдяки адаптивному підходу система може підтримувати учнів на різних рівнях знань та допомогти швидко та ефективно освоїти англійську мову. Крім того, автоматизація перевірки завдань та наочне відображення результатів дозволяє користувачам зручно оцінювати свої успіхи та відслідковувати прогрес.

Особистий внесок здобувача. Наукові результати, викладені у бакалаврській кваліфікаційній роботі, отримані автором особисто. У друкованій роботі [1], опублікованій у співавторстві, автору належать такі результати: дослідження про сучасні технології в електронному навчанні, а зокрема у вивченні мов.

Апробація матеріалів бакалаврської кваліфікаційної роботи. Матеріали з дослідження доповідалися на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)».

Публікації. Матеріали дослідження були опубліковані в матеріалах конференції [1].

1. АНАЛІЗ РОЗВИТКУ СИСТЕМ ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану вебсистем для вивчення англійської мови

Сучасний етап розвитку цифрових технологій вимагає постійного вдосконалення методів навчання іноземних мов, зокрема англійської. Традиційні методи викладання, хоча й залишаються актуальними, часто обмежені пасивними формами навчання та недостатньою індивідуалізацією процесу. У цьому контексті використання інтерактивних вебсистем, здатних забезпечити комплексний підхід до вивчення мови, стає дедалі популярнішим [2].

Одним із ключових аспектів розвитку сучасних мовних платформ є інтеграція штучного інтелекту та машинного навчання. Сучасні обчислювальні системи дозволяють аналізувати прогрес кожного студента, адаптувати навчальні матеріали та надавати персоналізовані рекомендації, що значно підвищує ефективність навчання.

Сучасні вебсистеми для вивчення англійської мови пропонують значні переваги порівняно з класичними методами. Головною перевагою є можливість створення середовища, яке поєднує різні аспекти мови - лексику, граматику, аудіювання та говоріння. Такі системи дозволяють точно оцінювати рівень знань та визначати проблемні зони кожного учня.

Важливою характеристикою сучасних платформ є їх багатомодальність. Вони можуть інтегрувати різні типи контенту - текстові матеріали, аудіозаписи, відео, інтерактивні вправи, що забезпечує комплексний підхід до вивчення мови. Це дозволяє враховувати різні стилі навчання та забезпечує більш глибоке засвоєння матеріалу.

Ще однією суттєвою перевагою вебсистем є можливість точної оцінки прогресу. У сфері мовної освіти об'єктивні дані про досягнення студентів відіграють вирішальну роль у плануванні навчального процесу. Зокрема, точний аналіз рівня володіння мовою, виявлення проблемних аспектів та моніторинг прогресу є критично важливими для ефективного навчання.

Таким чином, аналіз сучасних вебсистем для вивчення англійської мови підтверджує важливість переходу до більш інтерактивних та персоналізованих методів навчання. Розробка програмних рішень, що поєднують передові технології з педагогічними підходами, стає необхідністю для підвищення ефективності мовної освіти. Враховуючи стрімкий розвиток освітніх технологій, важливо інтегрувати останні досягнення у галузі аналітики даних та UX-дизайну для створення конкурентноздатних рішень, що забезпечать якісний користувацький досвід.

1.2 Порівняльний аналіз аналогів

У сучасних умовах стрімкого розвитку цифрових технологій все більшої популярності набувають інтерактивні вебсистеми для вивчення іноземних мов, зокрема англійської. Ці технології інтегруються в різноманітні освітні платформи та додатки, пропонуючи нові підходи до мовної освіти.

Розробка сучасних програмних рішень для вивчення англійської мови на основі веб-технологій дозволяє забезпечити ефективність та індивідуалізацію навчального процесу. Такі системи дають змогу точніше аналізувати рівень знань учня, швидше виявляти проблемні аспекти та пропонувати персоналізовані навчальні траєкторії, що значно підвищує якість мовної підготовки.

До найбільш відомих та ефективних рішень у цій галузі належать:

- «Duolingo»;
- «Babbel»;
- «Busuu».

«Duolingo» – одна з найпопулярніших безкоштовних платформ для вивчення мов, яка використовує ігровий підхід. Запущена у 2012 році, налічує понад 500 мільйонів користувачів [3]. Особливістю є система "життів" та щоденних цілей, що мотивують до регулярних занять. «Duolingo» також має платну підписку «Duolingo Plus», яка прибирає рекламу та додає деякі додаткові

функції. Сервіс підходить як для початківців, так і для тих, хто хоче підтримувати знання мови у форматі щоденних тренувань (див.рисунок 1.1).

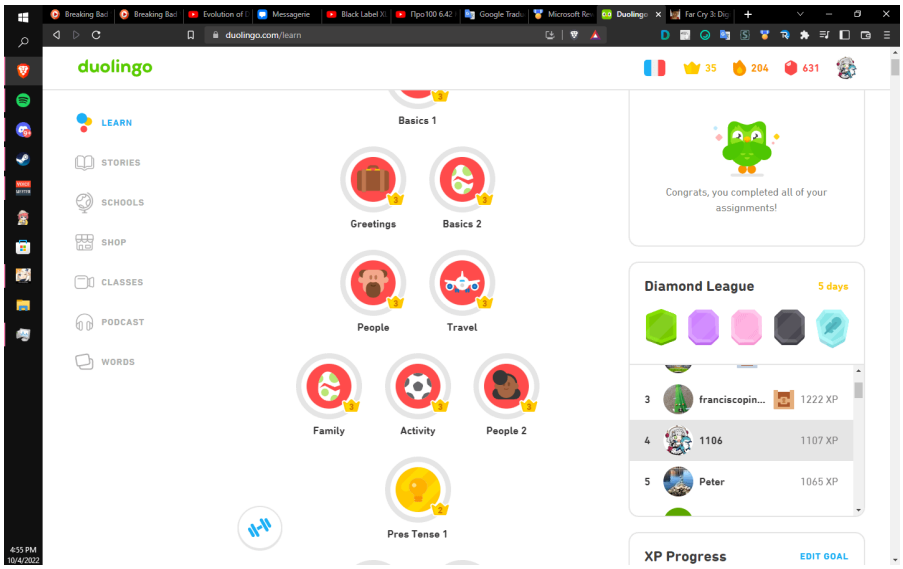


Рисунок 1.1 – Інтерфейс застосунку «Duolingo»

«Babbel» – платна платформа, що спеціалізується на практичному застосуванні мови. Розроблена лінгвістами, пропонує курси для 14 мов. Особливістю є акцент на реальних діалогах та ситуаціях [4].

Інтерфейс застосунку «Babbel» наведено на рисунку 1.2.

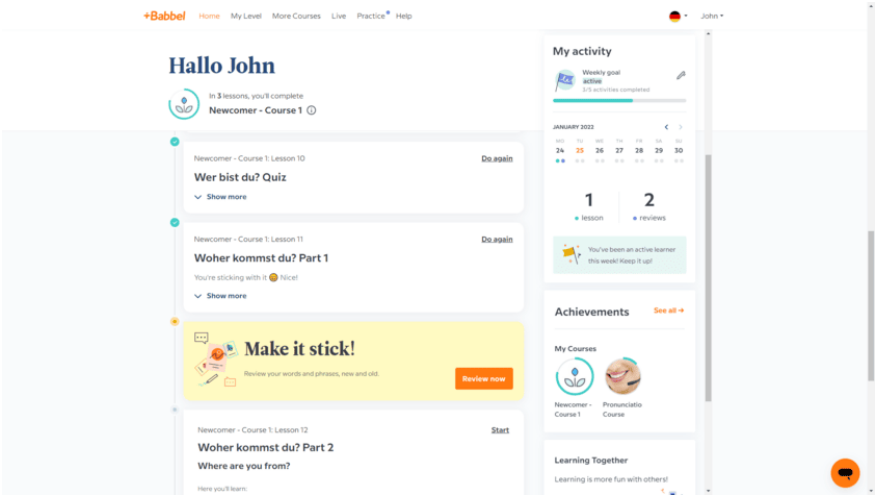


Рисунок 1.2 – Інтерфейс застосунку «Babbel»

«Busuu» – це популярна онлайн-платформа для вивчення іноземних мов. Вона пропонує курси з різних мов через вебсайт та мобільний додаток [5]. Серед доступних мов – іспанська, англійська, французька, німецька, італійська, японська, китайська, арабська та інші.

Платформа пропонує як безкоштовний доступ з обмеженими функціями, так і платну підписку з розширеними можливостями. Busuu підходить для різних рівнів – від початківців до тих, хто хоче підвищити свою мовну майстерність.

Інтерфейс застосунку «Busuu» наведено на рисунку 1.3.

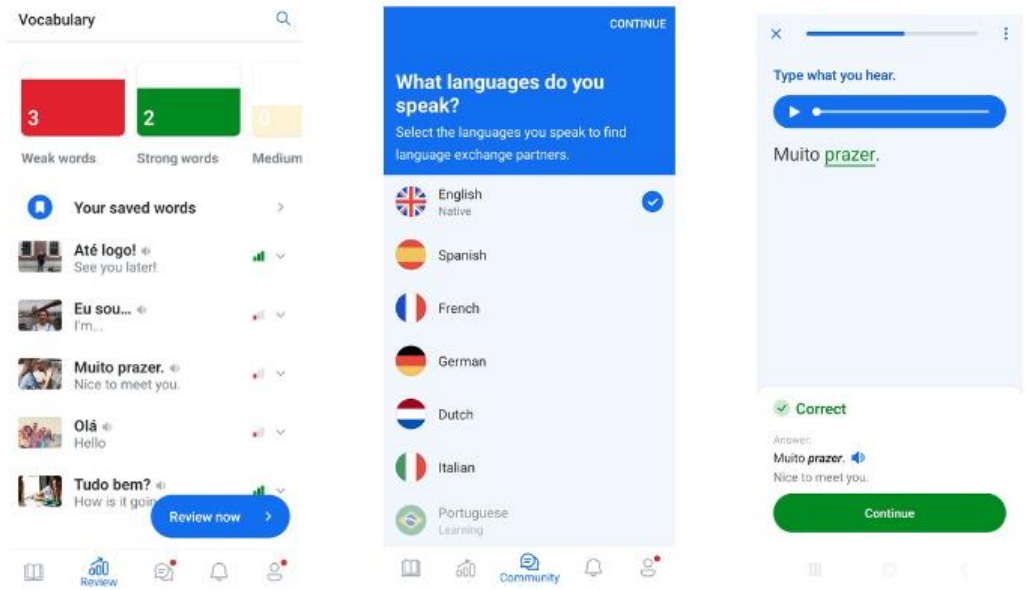


Рисунок 1.3 – Інтерфейс застосунку «Busuu»

Розробка програмних засобів для вивчення англійської мови з використанням вебтехнологій є комплексним процесом, який вимагає глибоких знань у галузі сучасного веброзробки. Для створення ефективної навчальної платформи необхідно реалізувати інтерактивний інтерфейс, що поєднуватиме мультимедійні елементи зі зручною системою навігації. Важливим аспектом є інтеграція механізмів зворотного зв'язку, які дозволять користувачам отримувати миттєву реакцію на їхні дії та помилки.

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 Порівняльна характеристика аналогів

Додатки / Критерії	«Duolingo»	«Babbel»	«Busuu»	Власна розробка
Персоналізація навчання	+	+	-	+
Інтерактивні вправи	-	-	+	+
Розмовна практика	+	+	-	+
Наявність платної підписки	+	+	+	-
Мобільність	+	+	+	+

Відповідно до проведеного порівняльного аналізу, розробка власної вебсистеми для вивчення англійської мови є цілком виправданою. Запропоноване рішення дозволить усунути недоліки існуючих платформ та запропонує користувачам розширені можливості для ефективного вивчення мови.

1.3 Аналіз методів розв’язання задачі

Для успішної розробки розвиваючої вебсистеми з вивчення англійської мови необхідно ретельно підібрати ефективні методи та інструменти, які забезпечать реалізацію всіх функціональних вимог. У даному розділі проаналізовано оптимальні підходи до вирішення ключових завдань вебсистеми та обґрунтовано вибір конкретних технологій.

Технічна реалізація вебсистеми ґрунтується на сучасній вибірці з використанням JavaScript для бекенду, що забезпечує швидку розробку та ефективну обробку даних. База даних реалізована за допомогою сервісу Firebase [6], зокрема утиліти Firestore Database. Фронтенд реалізований за допомогою

стандартних веб-технологій HTML та CSS, які дозволяють створити зручний та інтуїтивно зрозумілий інтерфейс для користувачів з різним рівнем технічної підготовки.

Модуль аналізу мовлення використовує Web Speech API для перетворення голосового введення в текст та аналізу вимови [7]. Ця технологія дозволяє ідентифікувати основні помилки у вимові та надавати корисні рекомендації щодо їх виправлення. Результати аналізу відображаються у вигляді зрозумілої візуалізації, що полегшує користувачам процес корекції власної вимови.

Для вивчення граматики реалізовано модуль на основі наборів правил та шаблонів з прикладами завдань, розроблених на JavaScript.

Для виконання тестів також було розроблено окремий модуль на JavaScript, який містить набір правил та шаблонів для їх перевірки.

Вибір технологічного стеку обґрунтований його ефективністю, простотою інтеграції та відповідністю сучасним вимогам до вебдодатків. Використання перевірених рішень дозволило створити стабільну платформу, яка поєднує академічний підхід до вивчення мови з інтерактивними можливостями сучасних вебтехнологій.

1.4 Постановка задач для розробки розвиваючої вебсистеми

На основі аналізу існуючих рішень для вивчення англійської мови було визначено ключові напрямки розробки нашої платформи. Основною метою є створення ефективної системи аналізу мовлення з використанням Web Speech API, яка забезпечуватиме оцінку вимови, інтонації та темпу мовлення користувача з подальшим наданням детального зворотного зв'язку.

Для вивчення граматики та тестів реалізовано спеціалізовані модулі на базі JavaScript.

Одним з аспектів платформи є розробка адаптивного механізму навчання, який аналізує прогрес користувача та автоматично підбирає індивідуальні навчальні матеріали з урахуванням особливостей засвоєння інформації та проблемних тем.

Особливу увагу приділено створенню зручного та інтуїтивно зрозумілого інтерфейсу, який поєднує функціональність професійного навчального інструменту з простотою використання. Технічна реалізація базується на сучасному вебстеку з використанням JavaScript, HTML та CSS, що забезпечує стабільну роботу системи. База даних реалізована за допомогою Firestore Database.

Автоматизована система оцінювання навчальних досягнень дозволяє об'єктивно аналізувати прогрес користувачів та формувати персоналізовані рекомендації для подальшого навчання. Завершальним етапом розробки передбачено комплексне тестування всіх функціональних модулів, інтерфейсу користувача та перевірку продуктивності системи за різних умов навантаження.

1.5 Висновки

У першому розділі дослідження було проведено огляд сучасних платформ для вивчення англійської мови, серед яких аналізувалися такі популярні рішення як «Duolingo», «Babbel» та «Busuu». Кожна з цих платформ демонструє унікальний підхід до організації навчального процесу, маючи при цьому як сильні сторони, так і певні обмеження.

Детальний аналіз функціональних можливостей існуючих систем дозволив виявити ключові напрями для вдосконалення та сформулювати завдання для розробки власної інноваційної платформи. Основними аспектами, які потребують уваги, є створення більш гнучкої системи персоналізації навчання, вдосконалення механізмів оцінки мовленнєвих навичок та розробка інтуїтивного інтерфейсу.

Проведене дослідження підтвердило доцільність розробки власного програмного рішення, яке зможе ефективно поєднувати переваги існуючих систем та усувати їхні недоліки. На основі отриманих даних було визначено перелік ключових завдань, серед яких особливу увагу приділено розробці інтелектуальних алгоритмів аналізу мовлення, створенню адаптивної системи навчання та реалізації зручного користувацького інтерфейсу.

2 РОЗРОБКА МОДЕЛІ ТА АЛГОРИТМІВ ВЕБСИСТЕМИ

2.1 Аналіз даних та методів розробки

Для створення інтерактивної веб-платформи використовується сучасний технологічний стек на базі JavaScript для бекенду та HTML/CSS для фронтенду. В якості бази даних використовується Firestore Database, дані також зберігаються до localStorage. Це дозволить реалізувати зручний та продуктивний інтерфейс користувача без необхідності застосування складних фреймворків, а також ефективно зберігати дані користувача.

Основним інструментом аналізу мовлення виступає Web Speech API, який забезпечує точне розпізнавання та оцінку вимови, інтонації та ритму мовлення. Для перевірки граматики та тестування реалізовано спеціалізовані модулі на чистому JavaScript, що дозволяє ефективно вивчати граматику та .

Технічна архітектура платформи включає:

1. Бекенд на JavaScript для обробки запитів.
2. Фронтенд на HTML/CSS.
3. Web Speech API для всіх функцій аналізу мовлення.
4. Базу даних Firestore Database.
5. Резервне копіювання в localStorage.

Даний підхід дозволяє створити ефективну навчальну платформу з мінімальними технічними накладними витратами, зберігаючи при цьому всі ключові функціональні можливості для якісного вивчення англійської мови.

2.2 Розробка моделі системи

UML (Unified Modeling Language) — це стандартизована мова моделювання, яка використовується для візуалізації, специфікації, конструювання та документування програмних систем [8]. Вона надає різноманітні типи діаграм, кожна з яких відображає певний аспект системи:

1. Діаграми поведінки (наприклад, діаграми діяльності, послідовності, станів) — показують динамічні процеси та взаємодії.

2. Діаграми структури (наприклад, класів, компонентів) — описують статичну архітектуру системи.

3. Діаграми взаємодії (наприклад, комунікації, часові) — акцентують увагу на обміні даними між об’єктами.

Діаграма діяльності, яка була розроблена, належить до поведінкових діаграм і ілюструє послідовність кроків у процесі, що особливо корисне для моделювання робочих потоків у вебдодатках.

UML-діаграма діяльності вебсайту детально відображає логіку роботи системи для користувача. Процес починається з ініціалізації, коли система підключається до Firestore Database для синхронізації даних. Одночасно дані зберігаються у localStorage, що забезпечує резервний варіант у разі проблем із зовнішньою базою даних. Після успішного підключення користувач отримує доступ до навчальних модулів.

Основним функціоналом є робота з навчальними модулями. Користувач обирає потрібний розділ, після чого система надає доступ до завдань. Завдання можуть включати різні форми виконання: запис власного мовлення для аналізу вимови, вивчення граматичних тем із прикладами або тестування з вибором правильного варіанту.

Після виконання завдання система аналізує результати, перевіряє відповіді, оцінює вимову (якщо це аудіозавдання) та формує зворотній зв’язок. Користувач отримує детальний аналіз помилок, рекомендації щодо покращення та можливість перейти до наступного рівня. Усі результати фіксуються у модулі «Прогрес», що дозволяє відстежувати досягнення.

Діаграма наочно ілюструє структурований підхід до вивчення англійської мови, поєднуючи інтерактивні завдання, автоматичну перевірку та персоналізований зворотній зв’язок для ефективного навчання.

UML-діаграма наочно демонструє підхід до організації процесу вивчення англійської мови (див. рисунок 2.1).

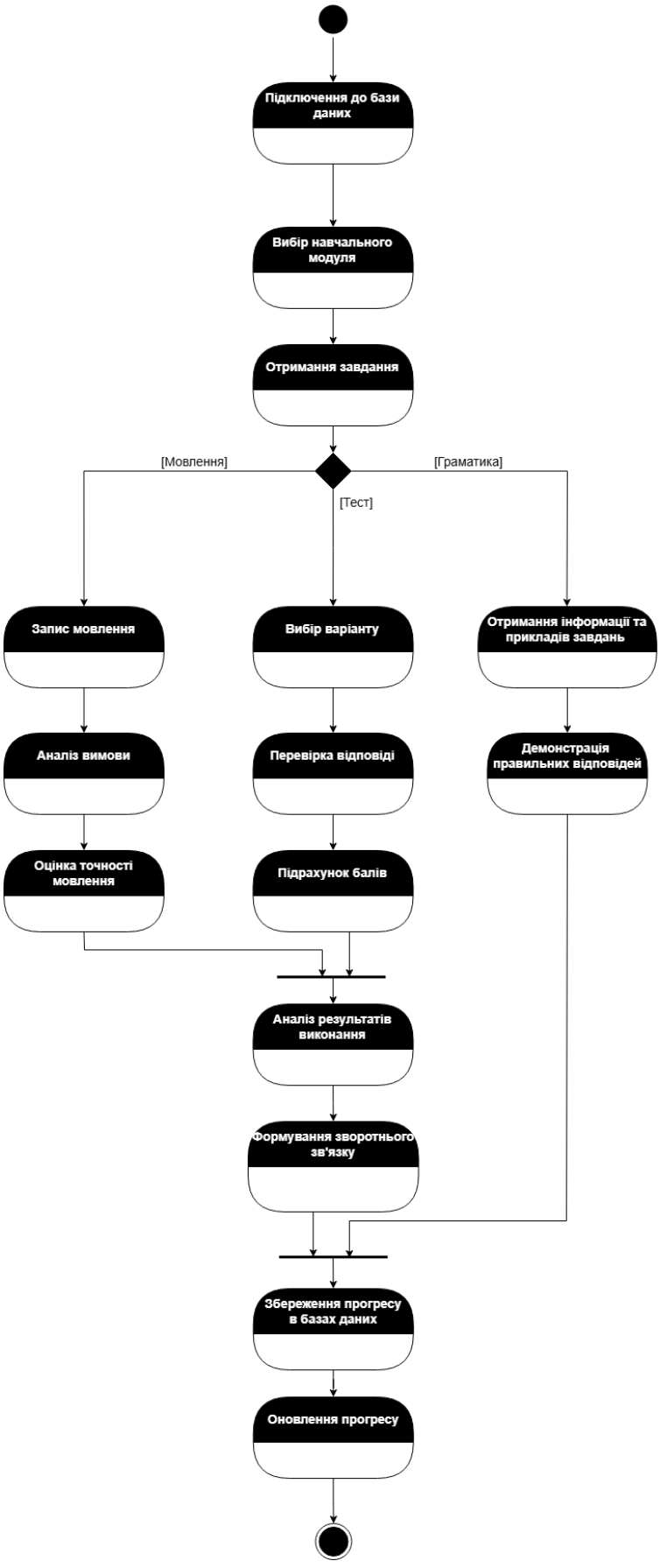


Рисунок 2.1 – Модель роботи системи у вигляді діаграми діяльності

Розробка діаграми діяльності та усі інші блок-схеми були виконані у програмному забезпеченні Diagrams.net [9].

2.3 Розробка методу для формування та відображення загального прогресу

Метод updateOverallStats() призначений для:

1. Оновлення статистики прогресу користувача у веб-інтерфейсі.

2. Візуалізації даних (дні навчання, бали, поточна серія, рівень) з анімацією.

3. Обробки спеціального випадку для відображення рівня та прогресу до наступного рівня.

Вхідні та вихідні дані:

Вхідні:

1. Об’єкт progress з даними, отриманими з app.getProgress() (містить daysLearning, streakDays, grammarProgress тощо).

Вихідні:

1. Оновлений інтерфейс користувача (DOM-елементи з анімованими значеннями).

Метод починає роботу з виклику app.getProgress(), який повертає об’єкт з актуальною статистикою користувача. Ця статистика включає кількість днів навчання, поточну серію безперервних занять, результати тестувань, прогрес у вивченні граматики та інші показники. Дані зберігаються у змінній progress для подальшої обробки.

На основі отриманих даних формується словник elements, де ключами виступають ідентифікатори DOM-елементів, а значеннями — відповідні дані з прогресу. Для визначення поточного рівня користувача викликається метод calculateCurrentLevel(), який аналізує загальну кількість балів і повертає інформацію про поточний рівень (наприклад, "B1"), наступний рівень (наприклад, "B2") та прогрес до нього у відсотках.

Для кожного елементу зі словника elements виконується перевірка його типу. Якщо елемент відповідає рівню (level-progress), його значення оновлюється текстом, що відображає поточний рівень та прогрес до наступного. Для інших елементів (наприклад, кількість днів навчання або поточна серія) запускається анімація зміни числа від нуля до кінцевого значення. Анімація реалізована за допомогою функції animateCounter(), яка використовує requestAnimationFrame для плавного оновлення значень.

Після оновлення всіх елементів інтерфейсу метод завершує свою роботу. Користувач бачить актуальну статистику свого прогресу, яка включає анімовані числа та чіткі індикатори рівня.

Для забезпечення плавної зміни чисел використовується функція animateCounter(), яка реалізує анімацію за допомогою requestAnimationFrame. Ця функція приймає початкове та кінцеве значення, тривалість анімації та застосовує ефект плавного прискорення через функцію easeOutCubic. Це створює візуально приємний ефект зростання чисел.

Метод calculateCurrentLevel() визначає поточний рівень на основі загальної кількості балів. Рівні мають чіткі порогові значення (наприклад, 100 балів для "A2", 300 балів для "B1"). Для розрахунку прогресу до наступного рівня використовується формула:

Якщо дані про прогрес відсутні (наприклад, при першому вході), метод може відображати стартові значення (0 днів, рівень "A1") з підказкою для користувача.

Це дозволяє точно відобразити, наскільки користувач близький до досягнення наступного рівня.

У разі помилки завантаження даних (наприклад, відсутнє інтернет-з’єднання), метод може показувати кешовані значення або повідомлення про тимчасову недоступність статистики.

Таким чином, реалізований метод забезпечує комплексне відображення прогресу навчання, включаючи етапи отримання даних, аналіз результатів, візуалізацію статистики та надання персоналізованої інформації про досягнення

користувача (див. рисунок 2.2). Метод автоматично оновлює інтерфейс, застосовуючи інтерактивні елементи та анімацію для наочної демонстрації динаміки навчання.

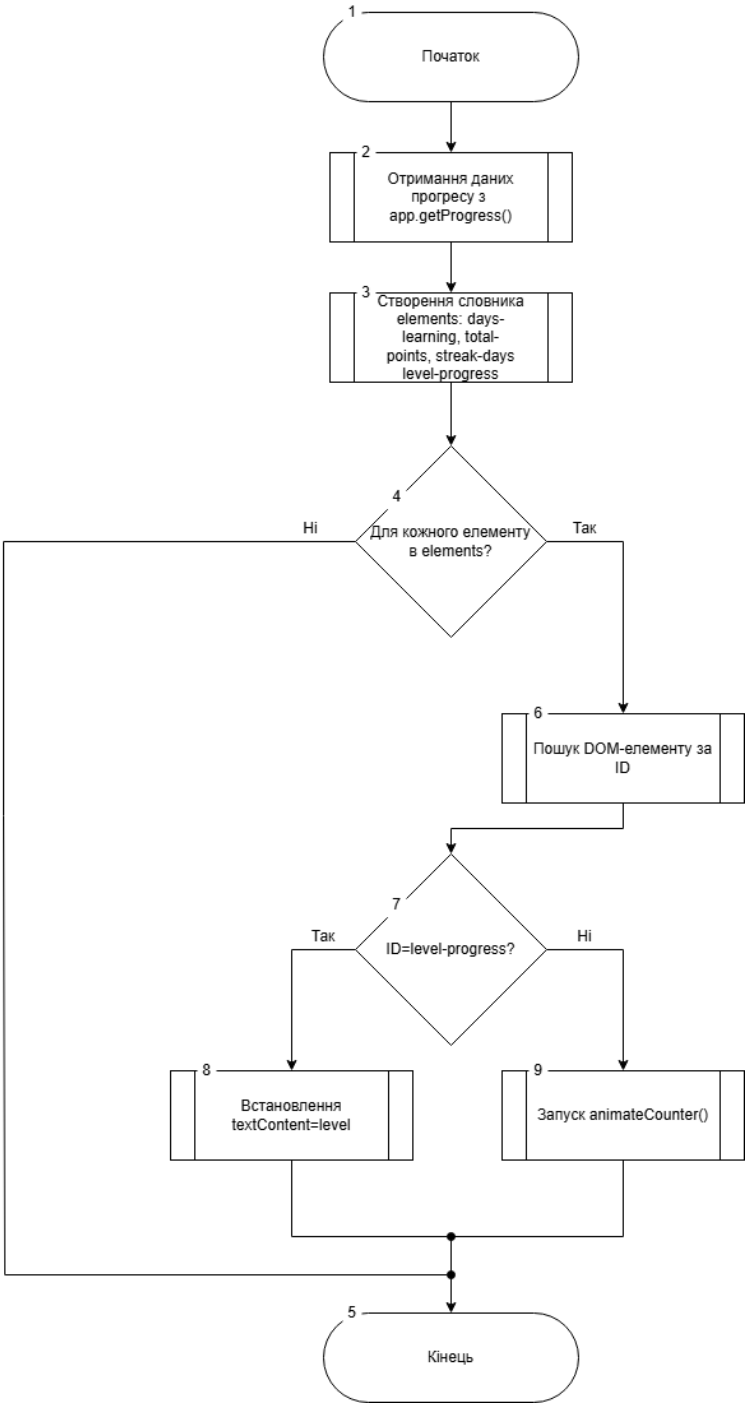


Рисунок 2.2 – Блок-схема методу для формування та відображення загального прогресу

Реалізований метод забезпечує повний цикл аналізу прогресу: від збору даних до наочної візуалізації. Система автоматично обчислює поточний рівень користувача, аналізує динаміку навчання та відображає результати у зручному форматі.

2.4 Розробка загального алгоритму роботи вебсистеми

Робота з вебсистемою розпочинається з технічної підготовки системи, де ініціалізуються всі необхідні модулі та підключаються зовнішні сервіси зберігання даних (див. рисунок 2.3).

На етапі автентифікації система перевіряє наявність облікового запису користувача у базі даних Firestore. Для нових користувачів автоматично створюється персональний профіль. Цей профіль синхронізується з хмарним сховищем.

Після успішної ініціалізації користувач потрапляє на головну сторінку. Система пропонує три основні напрямки розвитку: вивчення граматики, проходження тестів та розвиток мовленнєвих навичок через ситуативні фрази.

При виборі граматичного модуля система надає структурований теоретичний матеріал, що супроводжується прикладами вправам для рукописного виконання.

Мовленнєвий модуль відрізняється більш практичною спрямованістю, пропонуючи користувачеві розіграш реальних комунікативних ситуацій. Система аналізує правильність відповідей, вимову, темп, що дозволяє формувати комплексні навички спілкування.

Після завершення будь-якого навчального блоку система перевіряє, чи було досягнуто певних критеріїв для отримання нових досягнень. У разі позитивного результату користувач отримує візуальне сповіщення.

Окремий розділ платформи присвячений аналітиці прогресу, де представлена уся наявна статистика.

Завершення роботи з системою супроводжується фінальною синхронізацією всіх даних, що гарантує збереження поточного стану навчання.

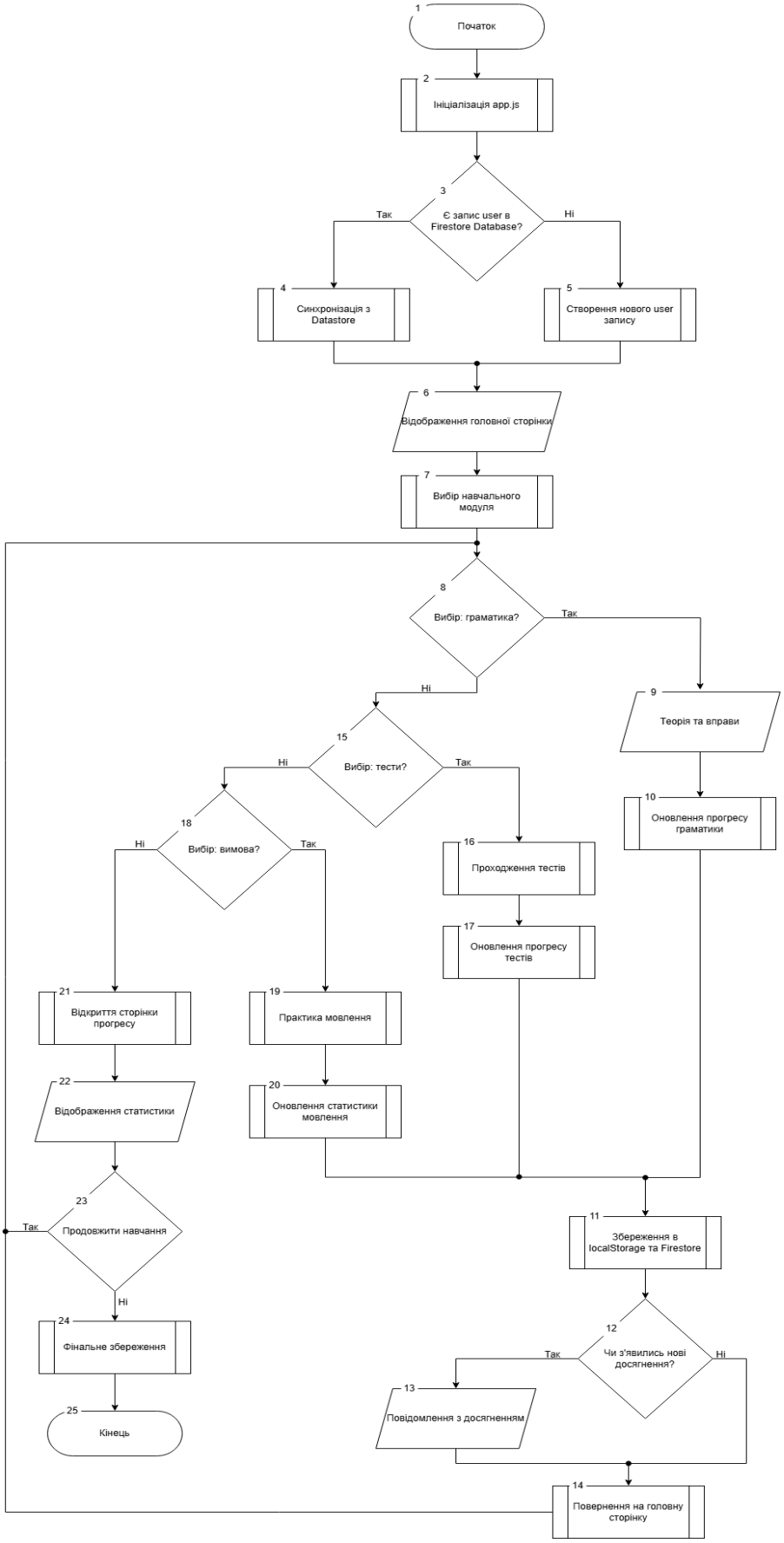


Рисунок 2.3 – Блок-схема загального алгоритму роботи вебсистеми

Робота вебсистеми організована так, що вона дозволяє користувачеві постійно вдосконалювати свої навички, отримувати зворотний зв’язок та мотивацію у вигляді досягнень.

2.5 Висновки

У другому розділі було розроблено модель та алгоритми функціонування вебсистеми для вивчення англійської мови. На основі аналізу сучасних технологій було обрано оптимальний стек: JavaScript для бекенду, HTML/CSS для фронтенду, Firestore Database для зберігання даних із резервним копіюванням у localStorage. Використання Web Speech API дозволило реалізувати точний аналіз мовлення, а спеціалізовані модулі на JavaScript забезпечили ефективне тестування та вивчення граматики.

Модель системи була представлена за допомогою UML-діаграми діяльності, яка наочно відображає логіку взаємодії користувача з вебсистемою. Діаграма охоплює всі етапи: від ініціалізації та синхронізації даних до виконання навчальних завдань, аналізу результатів та відображення прогресу.

Розроблений метод updateOverallStats() забезпечує автоматичне оновлення статистики користувача, включаючи анімацію показників, розрахунок поточного рівня та прогрес до наступного.

Загальний алгоритм роботи системи охоплює всі ключові етапи: від вибору навчального модуля до аналізу результатів і синхронізації даних. Система інтегрує різні форми навчання — граматику, тестування та розвиток мовленнєвих навичок, забезпечуючи комплексний підхід до вивчення мови.

Таким чином, у розділі вдалося реалізувати ефективну модель вебсистеми, яка поєднує інтерактивність, автоматизований аналіз даних та зручний інтерфейс для користувача. Запропоновані рішення дозволяють забезпечити стабільну роботу платформи, гнучкість у навчанні та мотивацію через систему прогресу.

3 РОЗРОБКА МОДУЛІВ ВЕБСИСТЕМИ

3.1 Варіантний аналіз і обґрунтування вибору засобі для реалізації вебсистеми

Ефективність розробки вебсистеми залежить від вибору мови програмування та середовища розробки. Розглядались такі мови програмування: Python, JavaScript та PHP.

Python — інтерпретована об'єктно-орієнтована мова програмування високого рівня із суворою динамічною типізацією[10] .

Переваги:

1. Простота та читабельність: Чистий синтаксис робить його ідеальним для початківців.

2. Універсальність: Може використовуватись для вебу (Django, Flask), аналітики даних, штучного інтелекту тощо.

3. Велика спільнота: Багато бібліотек (наприклад, NumPy, Pandas для аналітики) та фреймворків.

4. Крос-платформеність: Працює на Windows, Linux, macOS без змін у коді.

Недоліки:

1. Повільність: Python — інтерпретована мова, тому вона повільніша за компільовані (C++, Java).

2. Не ідеальний для вебу: Для високонавантажених вебдодатків може знадобитись додаткове масштабування.

3. Глобальна блокувальна інтерпретатора (GIL): Обмежує паралельні обчислення в багатопотокових додатках.

PHP – скриптова мова програмування, була створена для генерації HTML-сторінок на стороні вебсервера [11].

Переваги:

1. Спеціалізована для вебу: Створена саме для веброзробки, має вбудовані функції для роботи з HTTP, формами та базами даних.

2. Простота вивчення: Синтаксис схожий на C/Java, що робить її доступною для початківців.

3. Широке використання: Велика кількість CMS (WordPress, Drupal) та фреймворків (Laravel, Symfony) побудовано на PHP.

4. Інтеграція з базами даних: Легко працює з MySQL, PostgreSQL.

Недоліки:

1. Серверна мова: Потрібен вебсервер (Apache, Nginx) для виконання коду.

2. Проблеми з продуктивністю: У порівнянні з сучасними технологіями (Node.js, Go), PHP може працювати повільніше.

3. Застарілий код: Багато старих проектів використовують застарілі практики, що ускладнює підтримку.

JavaScript – динамічна, об'єктно-орієнтована прототипна мова програмування [12].

Переваги:

1. Клієнтська обробка: Виконується безпосередньо в браузері, що дозволяє створювати інтерактивні інтерфейси без перезавантаження сторінки.

2. Універсальність: Може використовуватись як для фронтенду (HTML/CSS), так і для бекенду (Node.js).

3. Велика кількість бібліотек та фреймворків: Доступні такі інструменти, як React, Vue, Angular для розробки складних додатків.

4. Швидкість розробки: Не потребує додаткових інструментів для запуску (достатньо браузера).

5. Сумісність: Працює у всіх сучасних браузерах без додаткових налаштувань.

Недоліки:

1. Залежність від браузера: Різні браузери можуть інтерпретувати код по-різному.

2. Проблеми з безпекою: Вразливості, такі як XSS (міжсайтовий скриптинг), можуть виникати при неправильному використанні.

3. Складність масштабування: Для великих проектів може знадобитись додаткові інструменти (наприклад, TypeScript для типізації).

Отже, JavaScript — найкращий вибір, оскільки він інтегрований з браузером і не вимагає серверної частини.

Розглянемо середовища розробки, такі як Visual Studio Code, WebStorm та Atom.

Visual Studio Code – це безкоштовний редактор коду з відкритим вихідним кодом, розроблений Microsoft [13]. Він підтримує роботу з різними мовами програмування, включаючи JavaScript, TypeScript, HTML і CSS.

Переваги:

VS Code має легку та швидку роботу, що робить його популярним серед розробників. Він підтримує велику кількість розширень, які дозволяють налаштувати середовище під конкретні потреби. Вбудована підтримка Git спрощує роботу з версіями коду, а інтегрований термінал дозволяє виконувати команди без переключення між вікнами.

Недоліки:

Для повноцінної роботи часто потрібні додаткові плагіни, що може ускладнити налаштування. Деякі розширення можуть сповільнювати роботу, особливо на слабких комп’ютерах.

WebStorm — це потужне IDE від JetBrains, спеціалізоване для веброзробки. Воно підтримує JavaScript, TypeScript, HTML, CSS, а також популярні фреймворки, такі як React, Angular та Vue [14].

Переваги:

WebStorm пропонує вбудовані інструменти для налагодження, рефакторингу коду та автодоповнення, що значно прискорює розробку. Інтелектуальна підказка коду, інтеграція з системами контролю версій (Git) та можливість запуску тестів без додаткових налаштувань роблять його зручним для професійних розробників.

Недоліки:

WebStorm є платним продуктом, що може бути незручно для початківців або малих проектів. Також він вимагає більше ресурсів системи порівняно з легкими редакторами, тому на слабких машинах може працювати повільніше.

Atom — це безкоштовний текстовий редактор з відкритим вихідним кодом, розроблений GitHub. Він підтримує різні мови програмування та має можливість глибокого налаштування через плагіни [15].

Переваги:

Atom відрізняється простотою інтерфейсу та гнучкістю, оскільки користувачі можуть встановлювати різні теми та розширення. Він підходить для тих, хто любить налаштовувати середовище під себе.

Недоліки:

Atom має обмежену продуктивність, особливо при роботі з великими проектами або на слабких комп’ютерах. У 2022 році GitHub припинив його підтримку, тому оновлення та нові функції більше не надходять.

Серед розглянутих середовищ WebStorm є найкращим вибором для професійної веброзробки на JavaScript, HTML та CSS. На відміну від VS Code, який потребує додаткових налаштувань, WebStorm пропонує вже готовий набір інструментів для ефективної роботи. Він має потужні функції, такі як інтелектуальне автодоповнення, налагодження коду та інтеграція з фреймворками, що робить його ідеальним для складних проектів.

Хоча WebStorm є платним, його можна використовувати безкоштовно протягом пробного періоду, а для студентів та викладачів JetBrains пропонує безплатні ліцензії. Atom вже не підтримується, а VS Code, хоча й популярний, потребує додаткових зусиль для налаштування.

Таким чином, WebStorm є оптимальним середовищем для розробки вебсистеми на чистому JavaScript, HTML та CSS, забезпечуючи високу продуктивність та зручність у роботі.

Отже, після аналізу мов програмування та середовищ розробки, для реалізації вебсистеми було обрано JavaScript та середовище розробки JetBrains WebStorm.

3.2 Розробка модуля для розпізнавання мовлення

Модуль розпізнавання голосу є ключовим інтерактивним компонентом вебсистеми для вивчення англійської мови, орієнтованим на вдосконалення навичок вимови. Він інтегрує сучасні веб-технології мовного синтезу та розпізнавання для створення навчального середовища з ефективним зворотним зв'язком (див. рисунок 3.1).

Модуль побудований на базі Web Speech API, що включає два основних інтерфейси для синтезу мови (Text-to-Speech) та розпізнавання (Speech Recognition). При ініціалізації система перевіряє підтримку цих можливостей у браузері користувача, надаючи відповідні повідомлення у разі їх відсутності. Для роботи з мовним контентом реалізовано структуроване зберігання фраз, організоване за категоріями (привітання, питання, побутові ситуації тощо) з різними рівнями складності.

Користувач отримує можливість вибору конкретної фрази для практики, яка включає оригінальний англійський варіант та переклад. Система пропонує прослухати зразок правильної вимови через механізм синтезу мови з регульованими параметрами відтворення. Під час запису власного варіанту модуль аналізує аудіопотік, виділяючи текстову транскрипцію та оцінюючи її відповідність еталону.

Для визначення точності вимови використовується комбінований підхід, що поєднує методи порівняння на рівні окремих символів та аналіз лексичної схожості. Система враховує рівень складності фрази, адаптуючи критерії оцінки для складніших конструкцій допускається більша похибка. Результати аналізу представляються у візуальній формі з індикацією відсоткової точності та якісними оцінками.

Модуль інтегрований із загальною системою відстеження прогресу, фіксуючи статистику виконаних завдань, середній рівень точності та найкращі досягнення. Особлива увага приділяється історії виконання вправ, що дозволяє аналізувати динаміку навчання. Для підтримки мотивації реалізовано механізм

визначення "засвоєних" фраз тих, точність вимови яких перевищує встановлений поріг.

Візуальна частина модуля включає інтуїтивні елементи керування процесом навчання, зокрема кнопки запису та відтворення, індикатори активності системи, таб-панель для навігації між категоріями фраз. Результати аналізу представлені у формі порівняння очікуваного та розпізнаного тексту з кольоровою індикацією рівня відповідності.

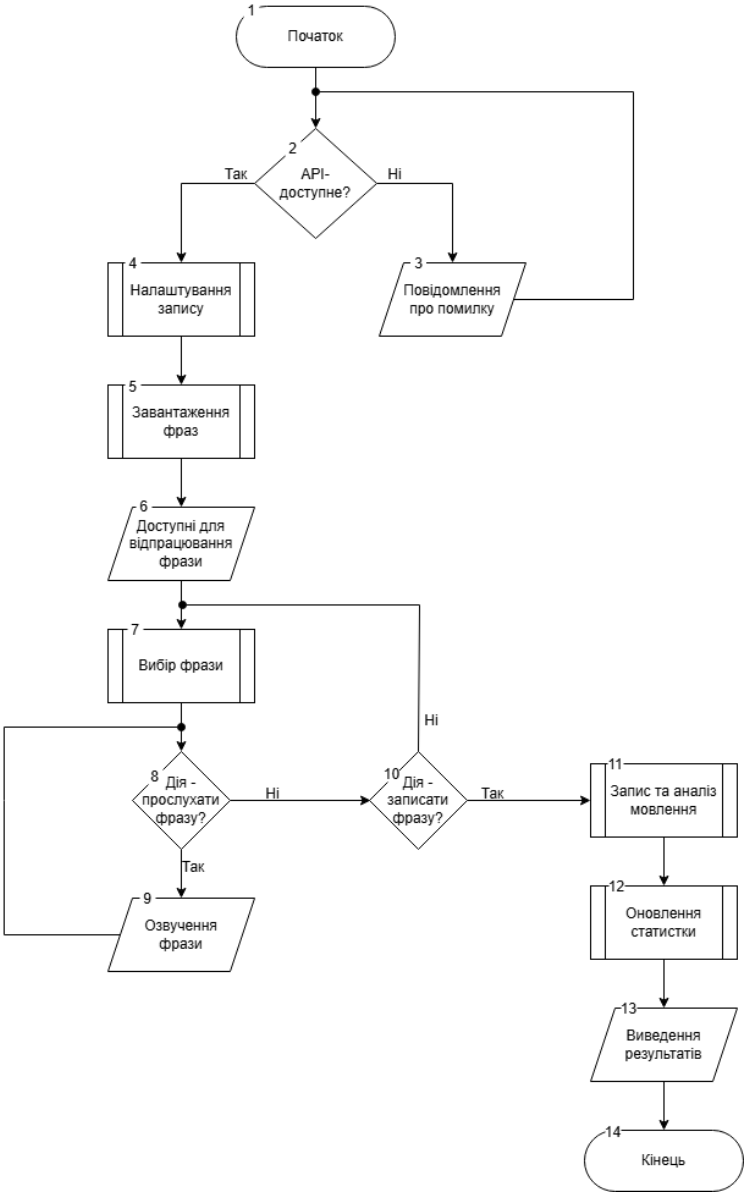


Рисунок 3.1 – Загальна блок-схема алгоритму роботи модуля для розпізнавання та аналізу мовлення

Модуль містить розгалужену систему обробки виняткових ситуацій, що виникають при роботі з аудіопристроями, включаючи відсутність дозволів на доступ до мікрофона, проблеми з мережевим з'єднанням та інші технічні обмеження. Для кожного типу помилки передбачено зрозумілі повідомлення з рекомендаціями щодо усунення проблем.

3.3 Розробка модуля для вивчення граматики

Модуль для вивчення граматики є комплексним компонентом для вивчення граматики англійської мови, який було розроблено та інтегровано до вебсистеми. Цей модуль пропонує структурований підхід до вивчення граматичних тем з інтерактивними вправами, відстеженням прогресу та зручною навігацією. Алгоритм його роботи представлено на рисунку .

Модуль граматики надає такі ключові можливості:

1. Шість граматичних тем для вивчення: Present Simple, Past Simple, Future Simple, Articles, Present Continuous та Modal Verbs.

2. Інтерактивний інтерфейс з можливістю навігації між темами.

3. Приклади практичних вправв з зворотнім зв'язком та поясненнями.

4. Система відстеження прогресу, яка зберігає дані про вивчені теми та виконані вправи.

5. Інтеграція з основним застосунком через об'єкт app.

Модуль надає кілька методів для роботи з темами:

- selectTopic(topicId) - вибір активної теми;

- loadTopic(topicId) - завантаження контенту теми;

- loadCurrentTopic() - завантаження збереженої теми з localStorage;

- navigateLesson(direction) - навігація між темами (вперед/назад).

Система навігації побудована на принципах максимальної зручності. Користувачі переміщаються між темами через інтуїтивну бічну панель або за допомогою спеціальних кнопок навігації. Розумний алгоритм зберігає прогрес, дозволяючи повертатися до навчання з того ж місця.

Особливістю модуля є його високий рівень інтерактивності. Вбудована система вправ надає миттєвий зворотний зв'язок, а детальні аналізи помилок допомагають глибше зрозуміти матеріал. Візуальні підказки та анімації полегшують сприйняття складних граматичних явищ.

Складова система аналітики відстежує всі аспекти навчального процесу. Візуалізація прогресу реалізована через індикатори завершення, що відображають ступінь опанування кожної теми. Спеціальні позначки відзначають досягнення користувачів, створюючи додаткову мотивацію.

Кожен граматичний розділ структурований за єдиним принципом: від загальних правил до конкретних випадків використання. Особлива увага приділяється типовим помилкам та складним моментам. Додаткові довідкові матеріали, такі як зведені таблиці та списки винятків, полегшують процес запам'ятовування.

Метод handleExerciseClick(button) обробляє кліки по кнопках вправ:

1. Отримує правильну відповідь та пояснення з атрибутів кнопки.
2. Відображає цю інформацію під кнопкою.
3. Позначає вправу як виконану.
4. Оновлює прогрес користувача.
5. Перевіряє чи завершена вся тема.

Для користувачів модуль пропонує персоналізований підхід до навчання, де кожен може вибирати оптимальний темп роботи. Розробники отримують гнучку платформу, здатну до масштабування та інтеграції додаткових функцій. Адаптивний дизайн забезпечує однаково комфортне використання на будь-яких пристроях.

Цей граматичний модуль представляє собою сучасне рішення для ефективного вивчення англійської граматики. Поєднання теоретичної бази з практичними механіками засвоєння матеріалу створює ідеальний баланс між навчанням і застосуванням знань. Інтерактивні елементи та система мотивації перетворюють процес навчання на захоплюючий досвід, що сприяє глибшому засвоєнню матеріалу та тривалому збереженню знань.

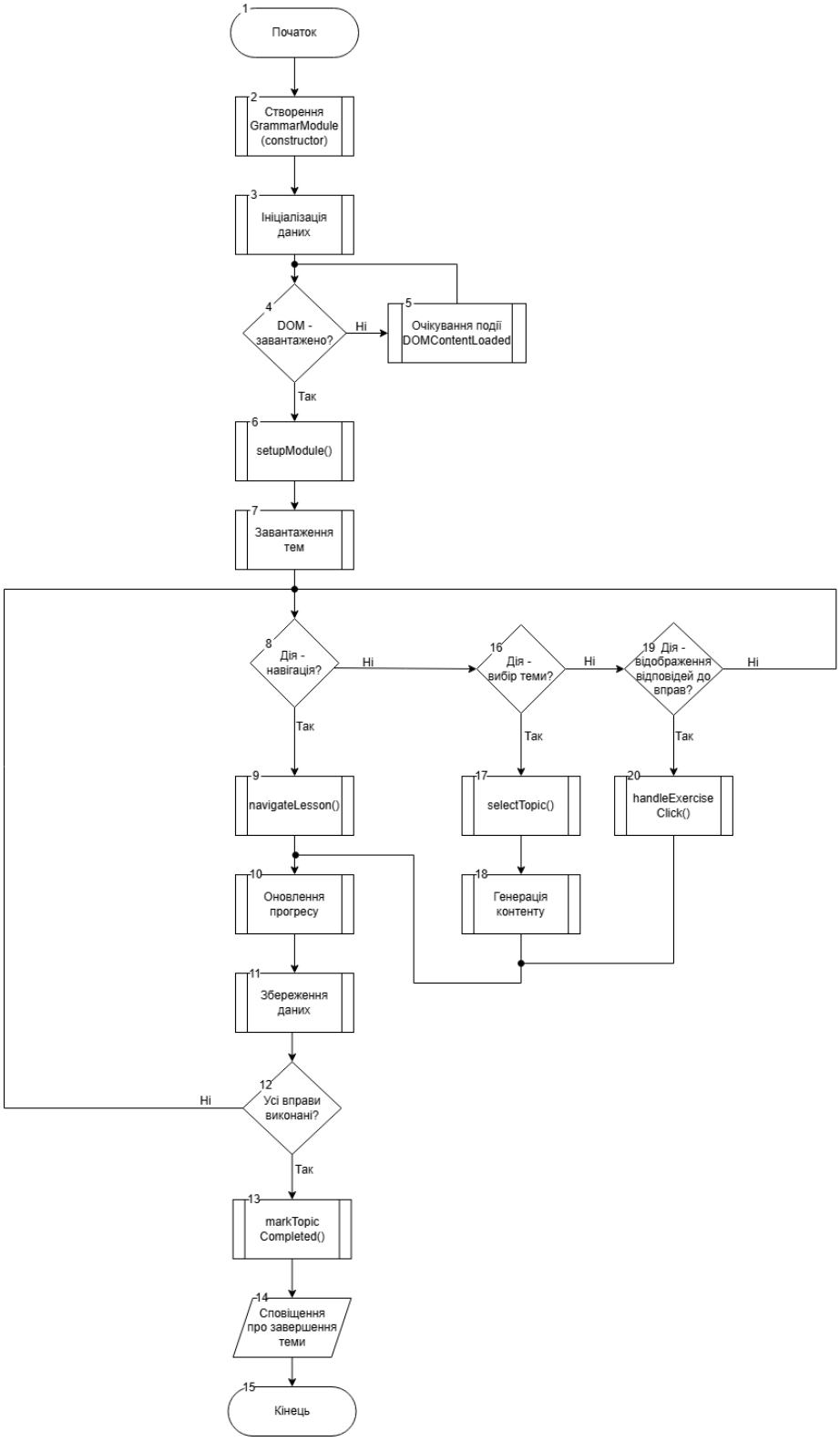


Рисунок 3.2 – Загальна блок-схема алгоритму роботи модуля для вивчення граматики

Гнучкість та адаптивність модуля дозволяють легко інтегрувати його в різні навчальні платформи, забезпечуючи інтерактивне вивчення граматики англійської мови.

3.4 Розробка модуля для проходження тестів

Модуль TestsModule реалізує інтерактивну систему тестування, яка адаптується до потреб користувача, пропонуючи персоналізований підхід до навчання. Він автоматично генерує тести на основі обраних параметрів, забезпечуючи різноманітність завдань для кращого засвоєння матеріалу. Кожне питання супроводжується зрозумілими поясненнями, що допомагає не лише виявити помилки, але й зрозуміти їх причини.

Модуль підтримує різні типи питань, включаючи вибір правильної відповіді з кількох варіантів, що дозволяє охопити всі аспекти вивчення мови — від базової граматики до складної лексики. Для зручності користувача інтерфейс тестування включає інтуїтивну навігацію, можливість повертатися до попередніх завдань та змінювати відповіді до завершення тесту.

Модуль надає можливості:

- почати новий тест;
- переміщуватися між питаннями;
- обирати відповіді;
- завершувати тест;
- перезапускати тест;
- переглядати результати.

Також у цьому модулі було реалізовано обмеження часу на проходження тесту з відображенням залишкового часу.

При ініціалізації модуль створює об'єкт, який керує всіма аспектами тестування. Він зберігає поточний стан тесту, включаючи обраний рівень складності, категорію питань, лічильник часу та прогрес користувача. Модуль використовує систему подій для обробки взаємодії з інтерфейсом, що дозволяє реалізувати інтуїтивно зрозуміле керування процесом тестування.

Користувач має можливість налаштувати параметри тесту перед початком, обравши бажану кількість питань та обмеження часу. Під час тестування модуль послідовно відображає питання, надаючи варіанти відповідей у зручному для сприйняття форматі. Система відстежує обрані відповіді, забезпечує навігацію між питаннями та відображає прогрес проходження тесту.

Після завершення тесту модуль автоматично аналізує результати, порівнюючи відповіді користувача з правильними варіантами. Він розраховує відсоток правильних відповідей, час виконання та інші метрики ефективності. Результати презентуються у візуально привабливій формі з індикаторами якості виконання (відмінно, добре, задовільно, погано).

Модуль пропонує розширені можливості для аналізу результатів, включаючи детальний перегляд усіх питань тесту з підсвічуванням правильних та неправильних відповідей. Користувач може бачити пояснення до кожного питання, що сприяє ефективнішому засвоєнню матеріалу. Також реалізовано функції повторного проходження тесту або створення нового тесту з іншими параметрами.

Модуль тісно інтегрується з основним об'єктом додатка, зберігаючи історію тестувань та оновлюючи загальну статистику користувача. Це дозволяє відстежувати прогрес у навчанні та аналізувати динаміку удосконалення знань.

Модуль включає власні стилі для відображення модальних вікон перегляду результатів, що забезпечує послідовність інтерфейсу та зручність користування. Елементи інтерфейсу мають інтерактивні ефекти, що покращують взаємодію з користувачем.

Реалізація модуля використовує сучасні підходи веб-розробки, включаючи динамічне створення DOM-елементів, обробку подій та асинхронні операції. Код структурований з урахуванням принципів чистого коду та легкості подальшої підтримки.

Додатково модуль забезпечує мотиваційний ефект завдяки системі оцінювання, яка враховує не лише кількість правильних відповідей, але й швидкість виконання.

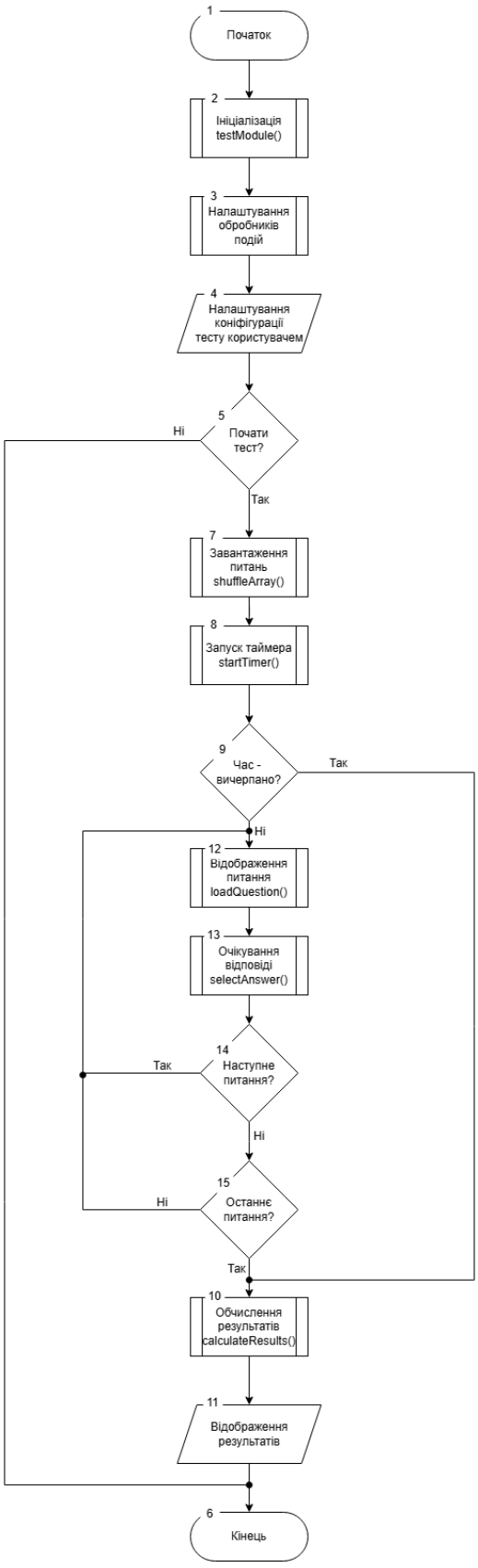


Рисунок 3.3 – Загальна блок-схема алгоритму роботи модуля для проходження тестів

Архітектура модуля дозволяє легко розширювати базу тестових запитань без змін у основній логіці роботи. Система підтримує додавання нових категорій питань та рівнів складності, що робить її придатною для майбутнього розвитку додатка.

3.5 Розробка структури інтерфейсу вебсистеми

Створення продуманої та логічно вибудованої структури інтерфейсу є одним із найважливіших етапів у розробці сучасних вебсайтів. Вона виконує роль свого роду «архітектурного каркасу», який визначає розташування, взаємозв'язки та функціональність усіх елементів системи. Грамотно побудована структура забезпечує не лише зручність навігації, а й високий рівень ефективності взаємодії користувача з цифровим продуктом.

Одним із найпоширеніших підходів до організації інформації є створення карти сайту (sitemap) або дерева навігації. Ці інструменти дозволяють розробникам та дизайнерам наочно відобразити ієрархію сторінок, розділів та підрозділів, а також логічні зв'язки між ними [16].

Карта сайту це схематичне представлення всієї архітектури вебресурсу, яке може бути виконане у вигляді текстового переліку (наприклад, у форматі XML для пошукових систем) або у вигляді візуальної діаграми (у Figma, Miro або спеціалізованих інструментах на кшталт Slickplan).

Дерево навігації більш деталізований варіант, який показує, як користувач переміщатиметься між різними рівнями інтерфейсу.

Такі моделі допомагають:

- оптимізувати розміщення контенту;
- уникнути хаотичності в навігації;
- забезпечити логічний перехід між сторінками;
- покращити SEO-оптимізацію за рахунок чіткої структури.

Щоб створити дійсно ефективну систему, необхідно враховувати три основні аспекти:

1. Врахування потреб користувачів (User-Centered Design)

2. Логічна організація інформації (Information Architecture)

3. Візуальна зручність та інтуїтивність (UI/UX-оптимізація)

Для створення структури вебсистеми було обрано карту сайту (див. рисунок 3.4).

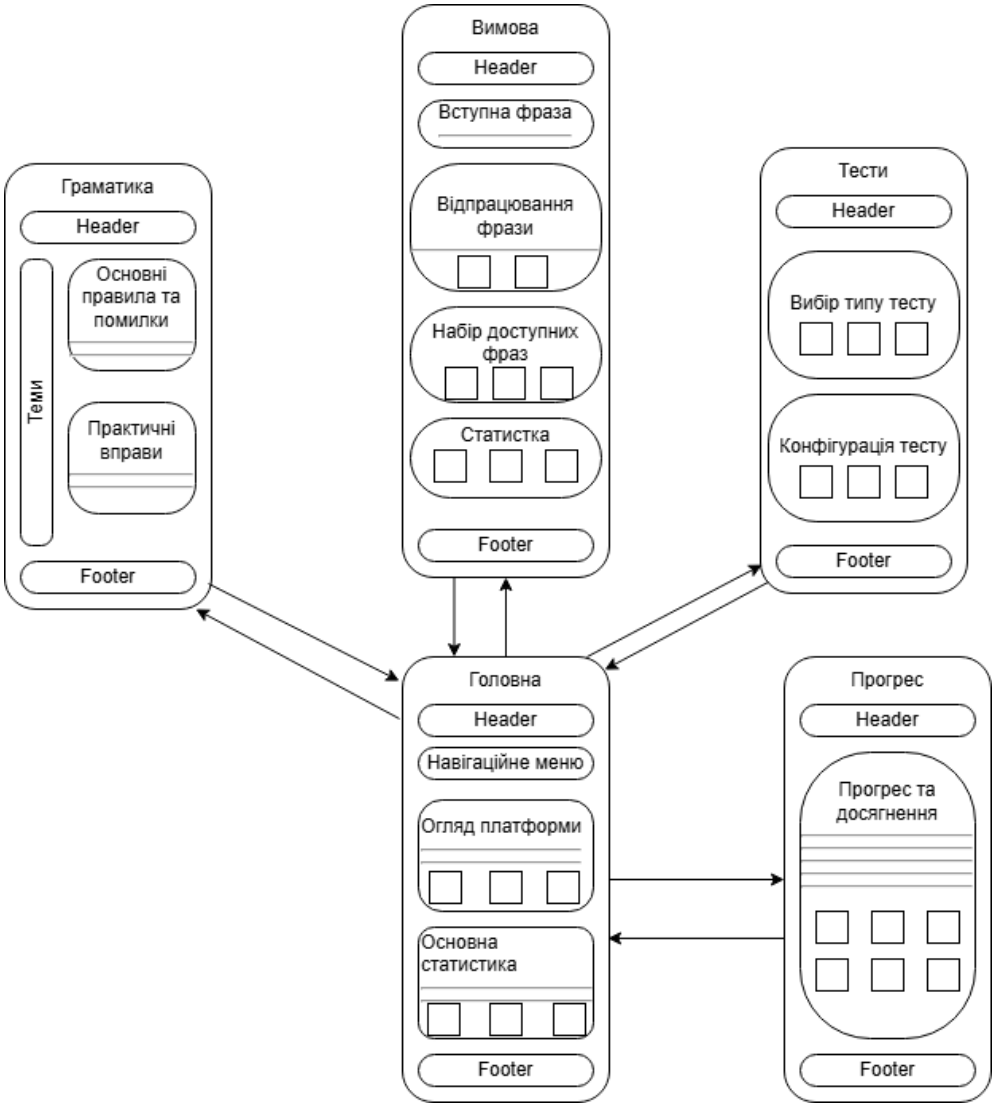


Рисунок 3.4 – Карта сайту

За основу структури інтерфейсу обрано класичний підхід, де головна веб-сторінка містить посилання на інші сторінки сайту, а допоміжні документи, у свою чергу, мають посилання на основну сторінку.

Ще одним ефективним способом проектування інтерфейсу є створення прототипів. Вони дають змогу наочно представити та протестувати взаємодію між різними елементами інтерфейсу ще до їх реалізації. Прототипи бувають:

- статичними (наприклад, зображення макетів);
- інтерактивними (з функцією взаємодії з користувачем).

На рисунку 3.8 представлено статичний прототип інтерфейсу головної сторінки розвиваючої вебсистеми, що був розроблений в межах проекту.

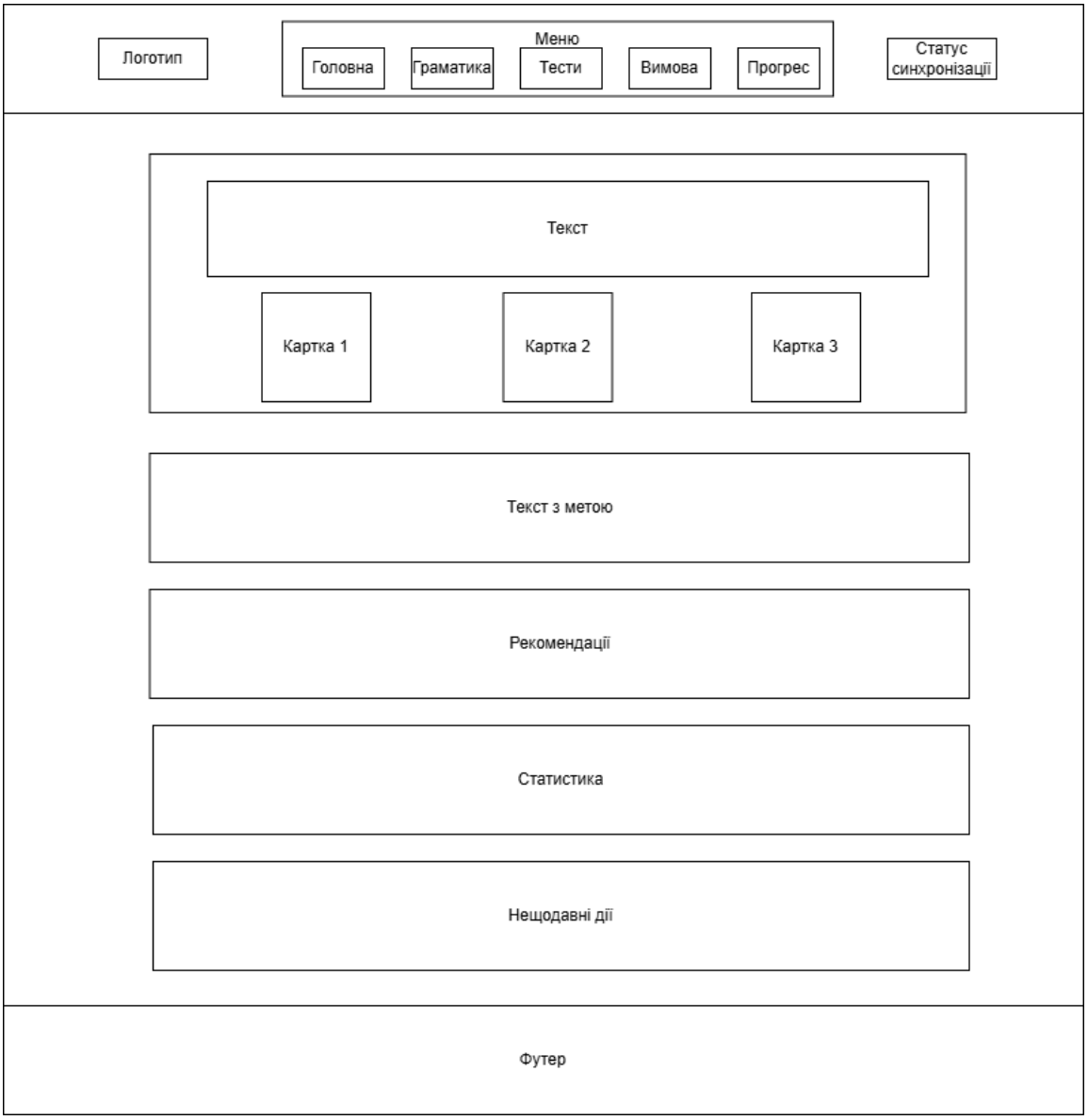


Рисунок 3.5 – Структурна схема головної сторінки

Використання прототипів дозволяє виявити можливі недоліки та оперативно внести корективи на початкових етапах розробки.

3.6 Розробка інтерфейсу вебсистеми

Під час розробки інтерфейсу розвиваючої вебсистеми було приділено особливу увагу його простоті, інтуїтивності та зручності для користувача. Основними кольорами інтерфейсу є синій, фіолетовий та білий. У більшості елементів використовується сам кольоровий градієнт з синього у фіолетовий, що створює дуже гармонічну картинку для користувача

Після запуску вебсайту користувача зустрічає головна сторінка з навігаційним меню у верхній частині, герой-секцією посередині та додатковими розділами внизу (див. рисунок 3.6).

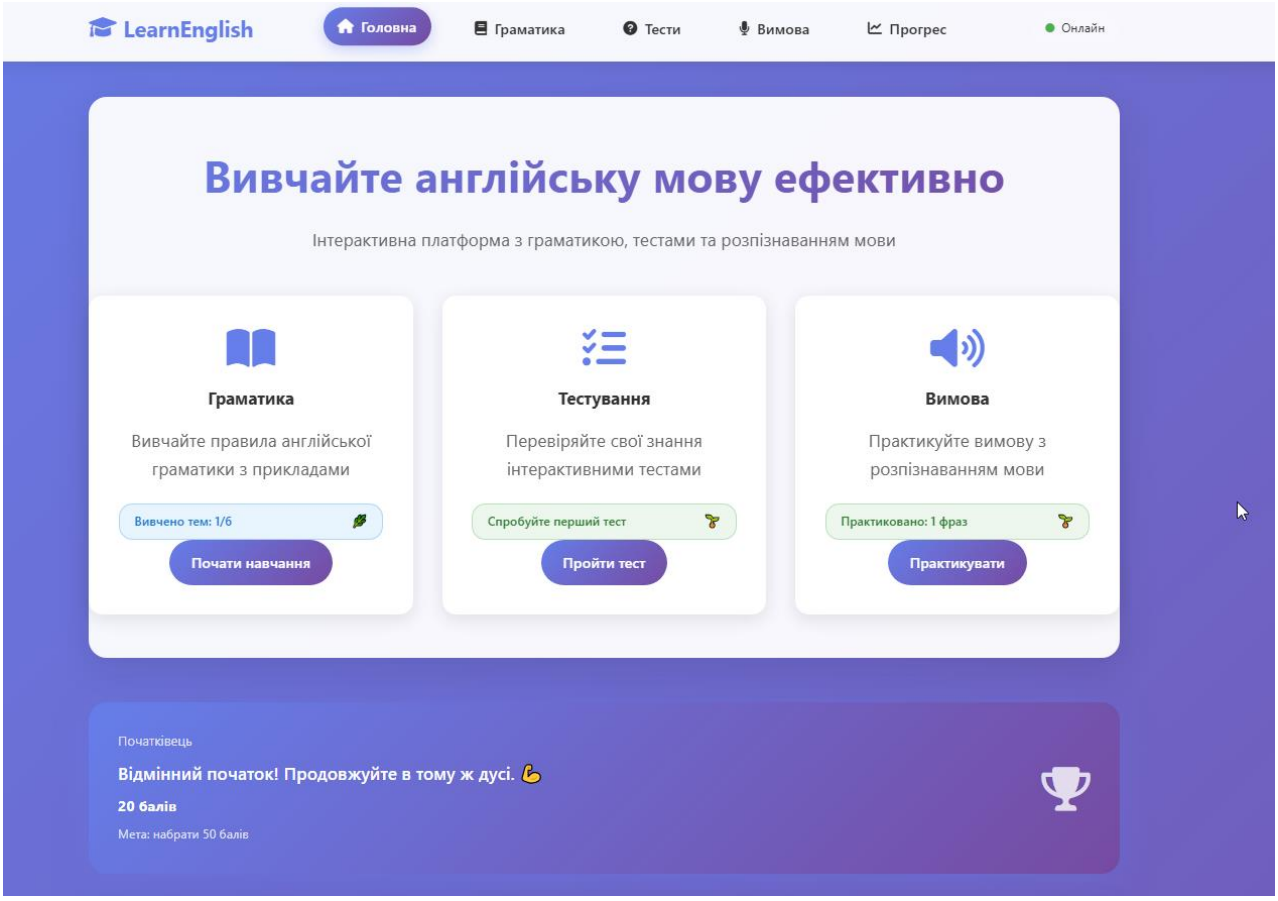


Рисунок 3.6 – Головна сторінка

На цій сторінці користувачу доступне повне навігаційне меню, а також блок з картками з інформацією та кнопками, за допомогою яких можна одразу перейти до вивчення англійської мови. Також нижче розташований блок з метою, а ще нижче можна знайти персональні рекомендації, коротку статистику та останні дії користувача.

Розділ граматики організований у вигляді двох панелей: бічної панелі зі списком тем та основної області для вивчення матеріалів (див. рисунок 3.7). Користувач може вибирати теми, такі як Present Simple, Past Simple, артиклі чи модальні дієслова, і вивчати їх у структурованому форматі з поясненнями та прикладами. Навігація між темами реалізована за допомогою кнопок "Попередня тема" та "Наступна тема", що забезпечує послідовне вивчення матеріалу.

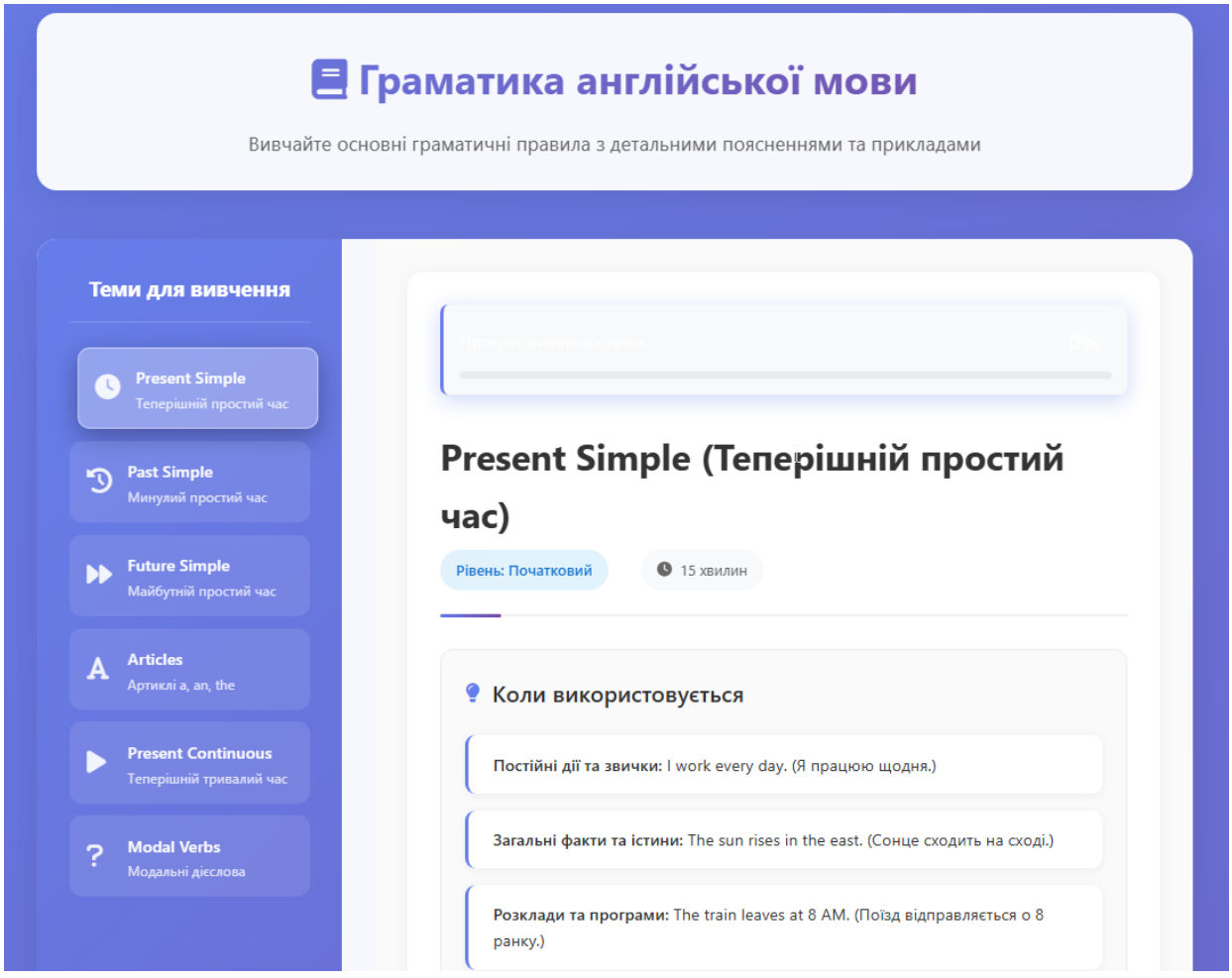


Рисунок 3.7 – Сторінка для вивчення граматики

Розділ для проходження тестів пропонує різні варіанти перевірки знань: тести з граматики, словникового запасу або змішані завдання (див. рисунок 3.8). Користувач може налаштувати рівень складності, та обмеження часу, що дозволяє адаптувати тестування під індивідуальні потреби. Під час проходження тесту відображається індикатор прогресу, таймер та інтерфейс для відповідей на питання. Після завершення тесту користувач отримує детальні результати з аналізом правильних та неправильних відповідей, що допомагає визначити слабкі місця.

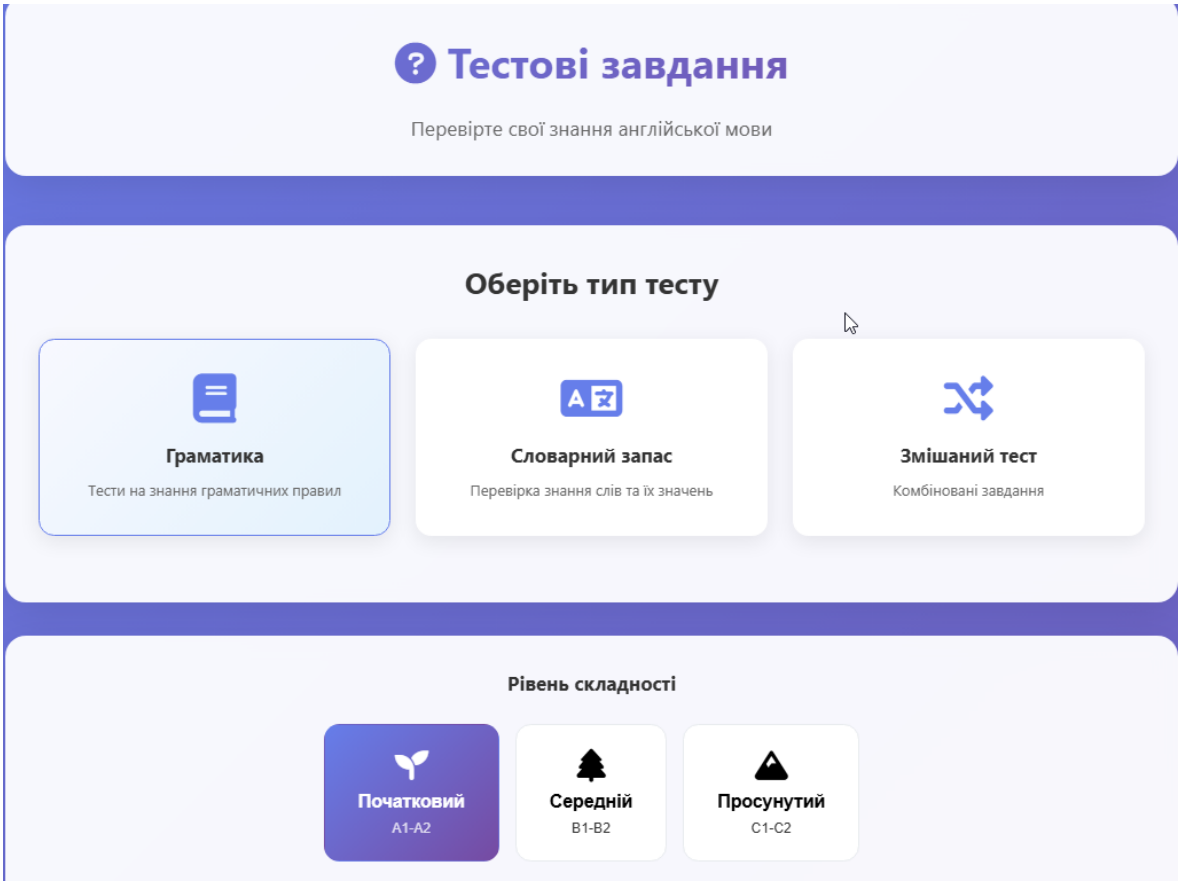


Рисунок 3.8 – Сторінка для проходження тестів

Інструмент для практики вимови використовує технологію розпізнавання мови для аналізу вимови користувача. Він пропонує фрази різного рівня складності, які можна прослухати, записати власний варіант та отримати зворотний зв'язок щодо точності вимови (див. рисунок 3.9). Результати

аналізуються за кількома параметрами, включаючи відсоток відповідності до еталонного варіанту та впевненість розпізнавання. Користувач може вибирати фрази з різних категорій, таких як привітання, питання або бізнес-лексика, що робить навчання більш цілеспрямованим.

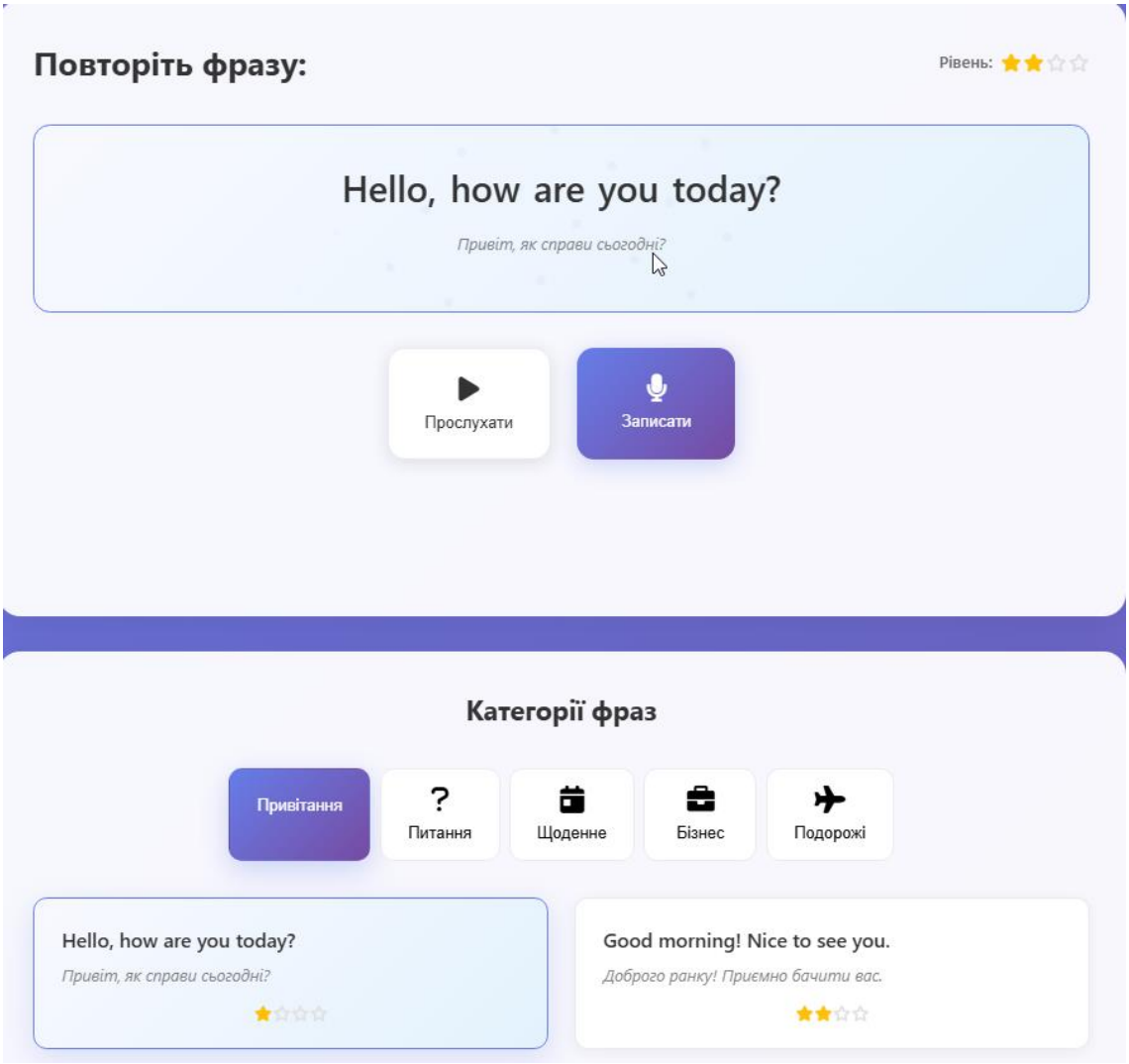


Рисунок 3.9 – Сторінка для практики вимови

Розділ прогресу надає детальну статистику навчання, включаючи кількість днів, проведених на платформі, загальний бал, поточний рівень володіння мовою та серію днів безперервного навчання. Тут також відображається прогрес у вивченні окремих тем граматики, результати тестів та досягнення у вимові. Візуалізація даних реалізована за допомогою діаграм, індикаторів прогресу та

календаря активності, що дозволяє легко оцінити свої успіхи. Досягнення, такі як "Перші кроки" або "Знавець граматики", додають елемент гри та мотивації.

Блоки з загальною статистикою та прогресом у граматиці представлено на рисунку 3.10.

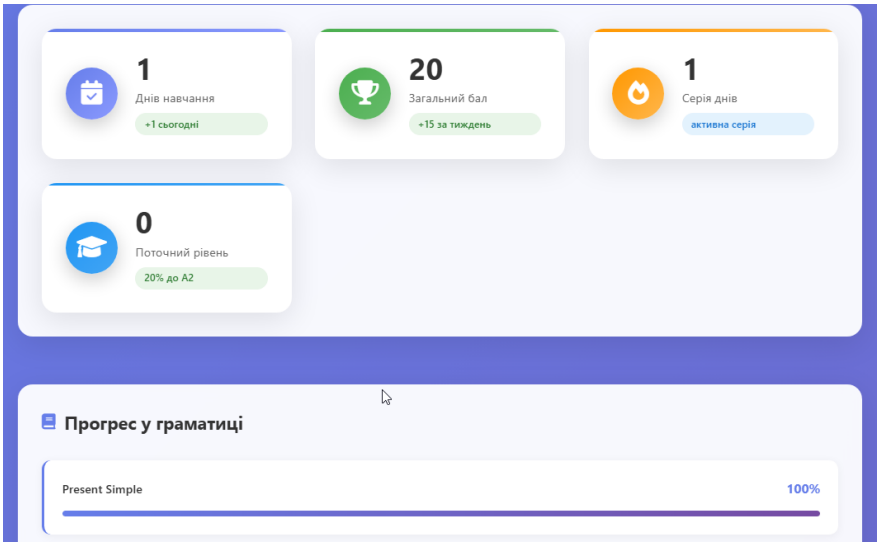


Рисунок 3.10 – Загальний прогрес та прогрес вивчення граматики

Інформаційний блок з статистикою за проходженням тестів представлено на рисунку 3.11.

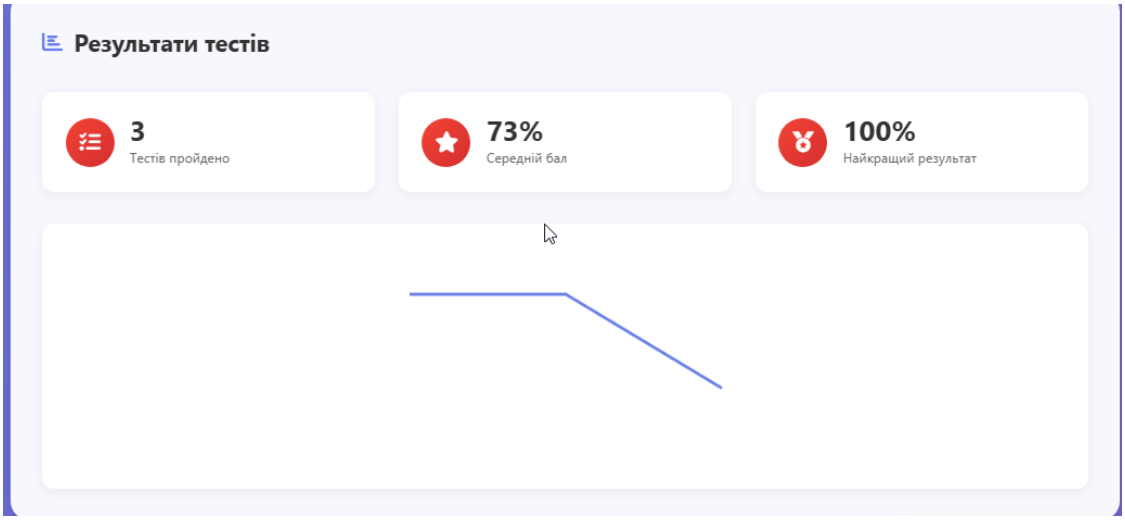


Рисунок 3.11 – Інформаційний блок зі статистикою тестувань

Блок зі статистикою та прогресом у вимові представлено на рисунку 3.12.

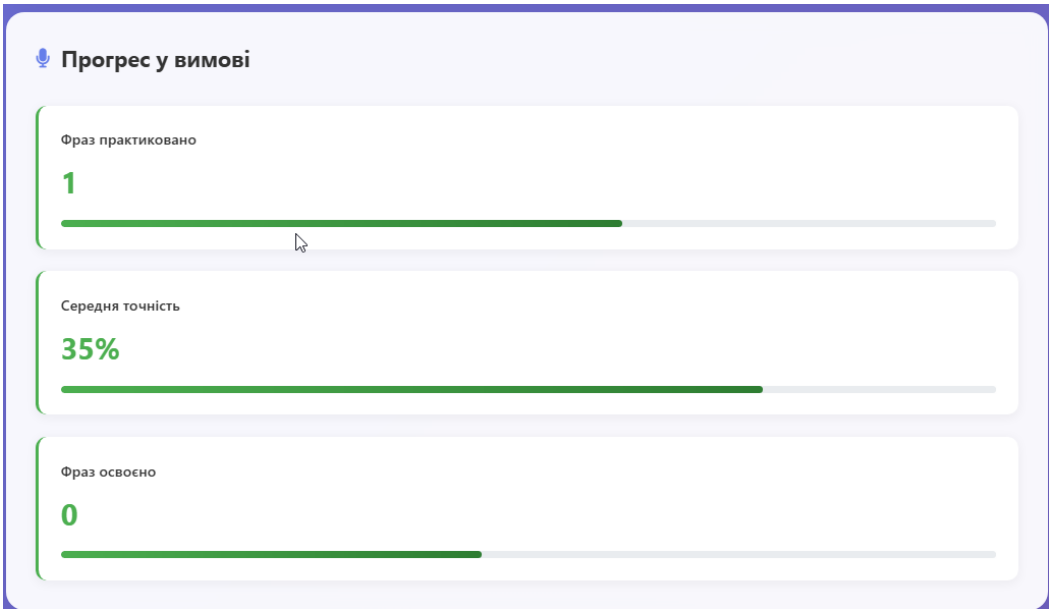


Рисунок 3.12 – Статистика практики вимови

Блок з календарем активності представлено на рисунку 3.13.

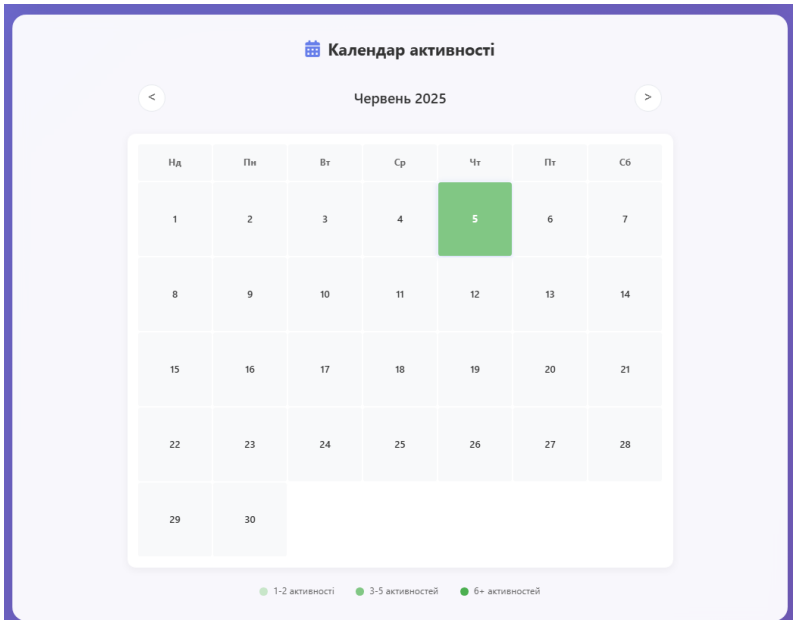


Рисунок 3.13 – Календар активності

Блок з досягненнями користувача продемонстровано на рисунку 3.8.

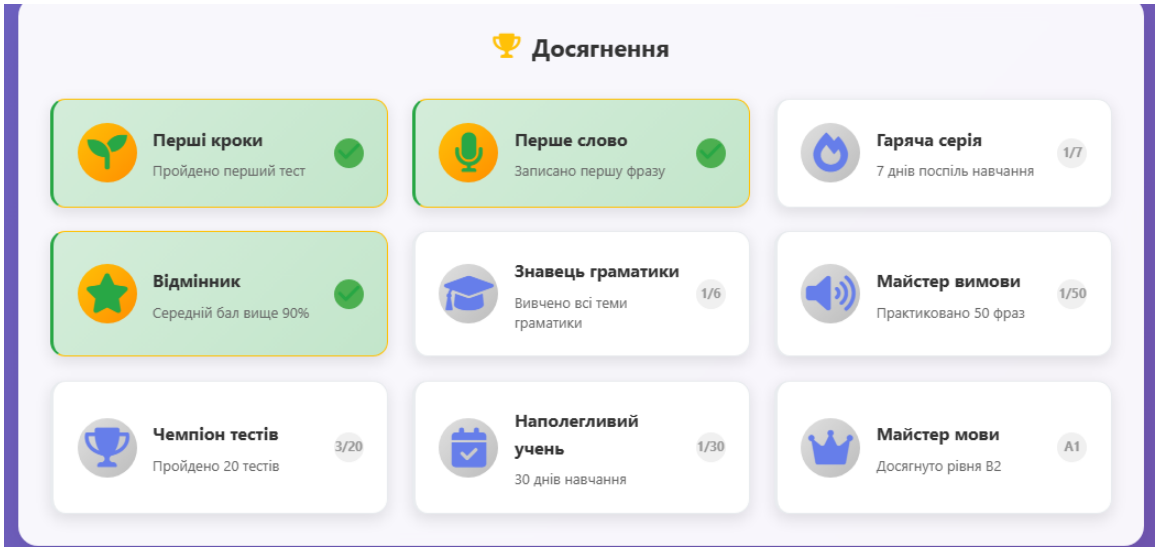


Рисунок 3.14 – Блок із досягненнями

Графічний інтерфейс для розвиваючої вебсистеми з вивчення англійської мови був розроблений за допомогою мови гіпертекстової розмітки HTML [17] та стилів CSS [18]. Було розроблено інтерфейс, який є зрозумілим та приємним для ока користувача.

3.7 Висновки

У третьому було проведено комплексний аналіз та обґрунтування технологічних рішень, що лягли в основу розробки вебсистеми для вивчення англійської мови. Детальне дослідження функціональних вимог до програмного забезпечення дозволило спроектувати оптимальну архітектуру системи, яка забезпечує високий рівень зручності використання, стабільність роботи та можливість горизонтального масштабування відповідно до зростаючих потреб користувачів.

Серверна частина системи була реалізована з використанням JavaScript у поєднанні з хмарною базою даних Firebase Firestore. Модуль database.js реалізує комплексну систему управління даними, що включає автоматичну

синхронізацію між локальним сховищем та хмарною базою даних. Ця архітектура забезпечує безперебійну роботу навіть при відсутності інтернет-з'єднання, накопичуючи зміни у локальній черзі та синхронізуючи їх при відновленні підключення. Кожен користувач отримує унікальний ідентифікатор, що дозволяє персоналізувати досвід навчання та зберігати індивідуальний прогрес.

Особливу увагу було приділено створенню інтуїтивно зрозумілого користувацького інтерфейсу, який адаптується до різних пристроїв. Система аналізує досягнення студента та пропонує персоналізовані рекомендації щодо наступних кроків у навчанні, визначаючи пріоритети на основі слабких місць у знаннях.

Граматичний модуль grammar.js представляє собою комплексну систему вивчення граматичних правил англійської мови. Він охоплює шість основних тем: Present Simple, Past Simple, Future Simple, Articles, Present Continuous та Modal Verbs. Кожна тема містить детальні пояснення правил, численні приклади використання та інтерактивні вправи для закріплення матеріалу. Система автоматично відстежує прогрес вивчення кожної теми та візуально відображає його через прогрес-бари та індикатори завершення.

Модуль тестування tests.js реалізує гнучку систему перевірки знань з можливістю налаштування параметрів тесту. Користувачі можуть обирати категорію тестування (граматика, словниковий запас або змішаний тест), рівень складності (початковий, середній або просунутий) та кількість питань. Система підтримує обмеження часу на проходження тесту та автоматично зберігає результати для подальшого аналізу. Після завершення тесту користувач отримує детальний звіт з можливістю перегляду правильних відповідей та пояснень до кожного питання.

Інноваційним компонентом системи є модуль розпізнавання мови speech.js, який інтегрує Web Speech API для практики вимови. Користувачі можуть прослуховувати еталонну вимову фраз різних категорій (привітання, щоденні фрази, бізнес-комунікація, подорожі), записувати власну вимову та

отримувати миттєвий зворотній зв'язок щодо точності. Алгоритм аналізу використовує відстань між символами для порівняння розпізнаного тексту з еталонним, враховуючи як посимвольну схожість, так і схожість на рівні слів. Результати аналізу візуалізуються через кольорові індикатори та відсоткові

Система реалізує складну логіку персоналізації навчання, яка автоматично адаптує навчальний контент до індивідуальних потреб користувача. Аналізуючи результати тестів та вправ, система виявляє проблемні області та автоматично включає додаткові матеріали для їх опрацювання. Наприклад, якщо користувач систематично помиляється у використанні артиклів, система збільшує частоту вправ на цю тему та пропонує додаткові пояснення.

Інтерфейс системи розроблений з урахуванням сучасних принципів UX/UI дизайну. Використання градієнтних кольорів, плавних анімацій та інтуїтивної навігації створює привабливе та зручне середовище для навчання. Адаптивний дизайн забезпечує оптимальне відображення на пристроях з різними розмірами екранів, від смартфонів до десктопних комп'ютерів.

Таким чином, результати реалізації третього розділу демонструють успішне створення комплексної розвиваючої вебсистеми для вивчення англійської мови, яка поєднує передові технології веброзробки з інноваційними підходами до онлайн-освіти. Система забезпечує високий рівень персоналізації, інтерактивності та адаптивності, що робить процес вивчення мови ефективним та захоплюючим для користувачів різного рівня підготовки.

4 ТЕСТУВАННЯ ВЕБСИСТЕМИ

4.1 Аналіз методів тестування

У сучасному цифровому світі програмні рішення охоплюють усі сфери діяльності. Однак нерідко зустрічаються ситуації, коли програмні продукти працюють некоректно, що може спричинити серйозні наслідки для різних аспектів людського життя.

Забезпечення якості програмного забезпечення передбачає комплексний підхід до виявлення недоліків та оцінки його роботи. Багато хто помилково вважає, що цей процес обмежується лише запуском тестів та аналізом результатів. Насправді він включає широкий спектр заходів і тісно пов’язаний із етапами створення програмного продукту [19].

Також існує думка, що тестування зосереджене виключно на перевірці функціоналу. Проте воно охоплює не лише верифікацію (відповідність технічним вимогам), а й валідацію (відповідність реальним потребам користувачів).

Основні види тестування:

- динамічне тестування перевірка роботи програми під час її виконання;

- статичне тестування аналіз коду та документації без запуску програми (наприклад, рев’ю коду).

Ці процеси вимагають не лише технічних навичок, але й ретельного планування, контролю та оцінки результатів.

Для ефективного тестування використовуються різні підходи, які допомагають визначити, що саме перевіряти та як це робити. Вони дозволяють створити оптимальний набір тестів, визначити умови їх проведення та підібрати відповідні дані.

Методи «чорної скриньки» базуються на аналізі зовнішньої поведінки системи без урахування її внутрішньої структури. До них належать:

1. Класифікація еквівалентності групування вхідних даних за спільними ознаками.

2. Перевірка граничних умов тестування значень на межах допустимих діапазонів.

3. Таблиці рішень аналіз логіки програми на основі комбінацій вхідних параметрів.

4. Тестування станів системи перевірка реакції програми на зміну станів.

5. Тестування сценаріїв використання імітація реальних умов роботи з продуктом.

Методи «білої скриньки» орієнтовані на внутрішню структуру програми та логіку її роботи. Тести розробляються після створення архітектури або написання коду.

Досвідчені методи ґрунтуються на експертних знаннях тестувальників і дозволяють виявляти помилки, які можуть залишитися непоміченими при інших підходах. До них відносяться:

1. Припущення про помилки прогнозування потенційних проблем на основі досвіду.

2. Чек-листи використання стандартизованих переліків для систематичної перевірки.

3. Дослідницьке тестування інтерактивний пошук дефектів без жорстких сценаріїв.

Для повноцінного тестування необхідно створити детальну документацію, яка включає:

1. Тест-кейси детальні інструкції з перевірки окремих функцій.

2. Чек-листи структуровані списки критеріїв для контролю якості.

Ці інструменти допомагають мінімізувати ризики виникнення помилок у робочому середовищі та забезпечують високу якість продукту для кінцевих користувачів.

Таким чином, комбінація різних методів тестування дозволяє досягти максимальної ефективності у забезпеченні якості програмного забезпечення.

4.2 Тестування вебсистеми

Тестування програмного забезпечення є критично важливим етапом життєвого циклу створення вебзастосунку, адже саме воно дозволяє виявити логічні та технічні помилки, перевірити стабільність системи за різних умов експлуатації та забезпечити відповідність реалізованого функціоналу попередньо визначеним вимогам. Основною метою тестування розробленої вебсистеми для вивчення англійської мови є гарантування її коректної, безперебійної та ефективної роботи як на стороні клієнта, так і на стороні сервера.

Перед початком повноцінного тестування була підготовлена стабільна версія застосунку, яка розгорталась локально. Це дозволило провести перевірку роботи інтерфейсу на різних платформах, включаючи десктопні браузери (Google Chrome, Mozilla Firefox, Microsoft Edge). Окрему увагу було приділено адаптивності дизайну, перевірці інтерактивних елементів та стійкості системи до нестандартних сценаріїв взаємодії з користувачем.

Основними середовищами для тестування стали:

- веб-браузери з інструментами розробника «DevTools», що дозволяли переглядати помилки в консолі, відстежувати запити, аналізувати логіку JavaScript-коду [20];
- утиліта «Lighthouse», яка вбудована у «DevTools» браузера «Google Chrome».

Процес тестування умовно поділявся на кілька ключових етапів:

1. Функціональне тестування: на цьому етапі перевірялися основні можливості вебсистеми — коректність голосового введення та розпізнавання мовлення, обробка результату API, відображення граматичних підказок, коректне формування звітів і статистики навчання. Особливу увагу приділяли стабільності роботи модуля аналізу мовлення.

2. Тестування на різних пристроях: проводилася перевірка адаптивності інтерфейсу, доступності усіх елементів у мобільній версії, а також стабільності роботи у різних браузерах.

3. Тестування інтерактивних компонентів: перевірялась коректна робота кнопок («Записати», «Показати відповідь», «Почати тест», «Наступна тема» та ін.). Також перевірялась правильність формування звіту після проходження тесту.

На рисунку 4.1 головну сторінку веб-сайту та її відображення на реальному пристрої з браузеру Google Chrome.

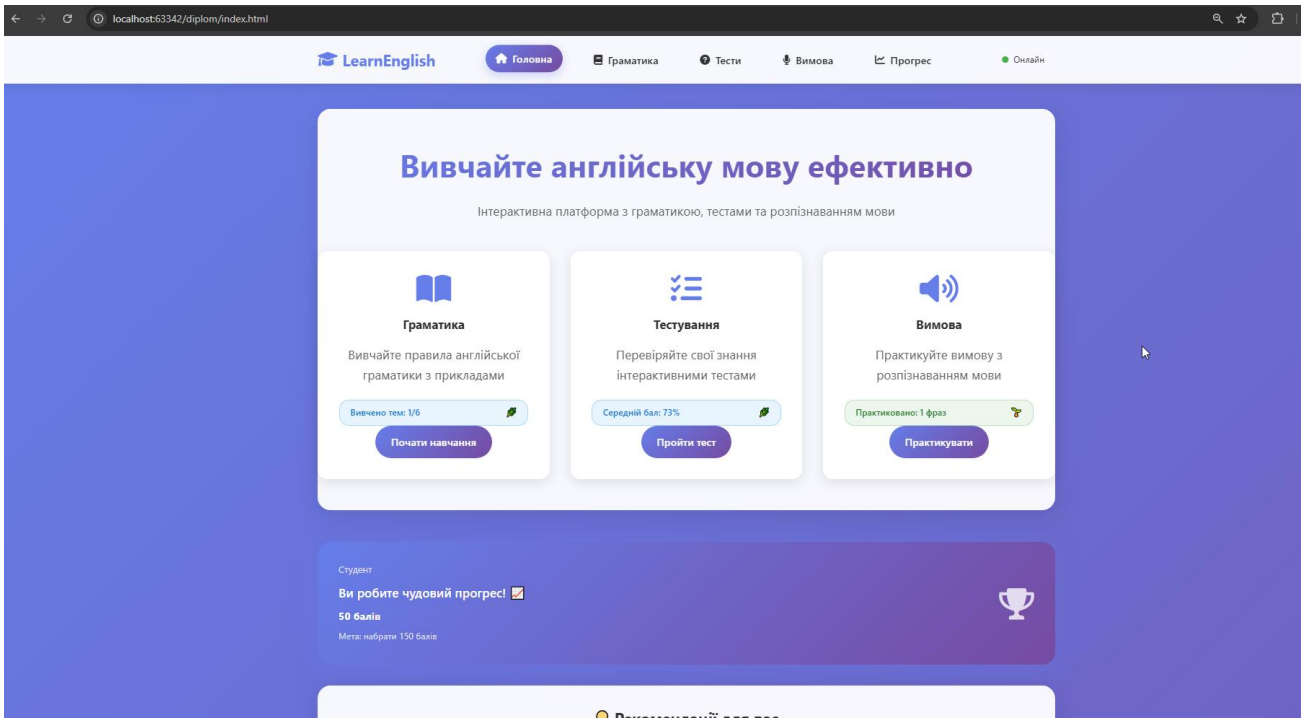


Рисунок 4.1 – Головна сторінка веб-сайту на локальному хості

Тестування для мобільних пристроїв проводилось у вбудованому емуляторі «DevTools». Вигляд головної сторінки з телефону представлено на рисунку 4.2.

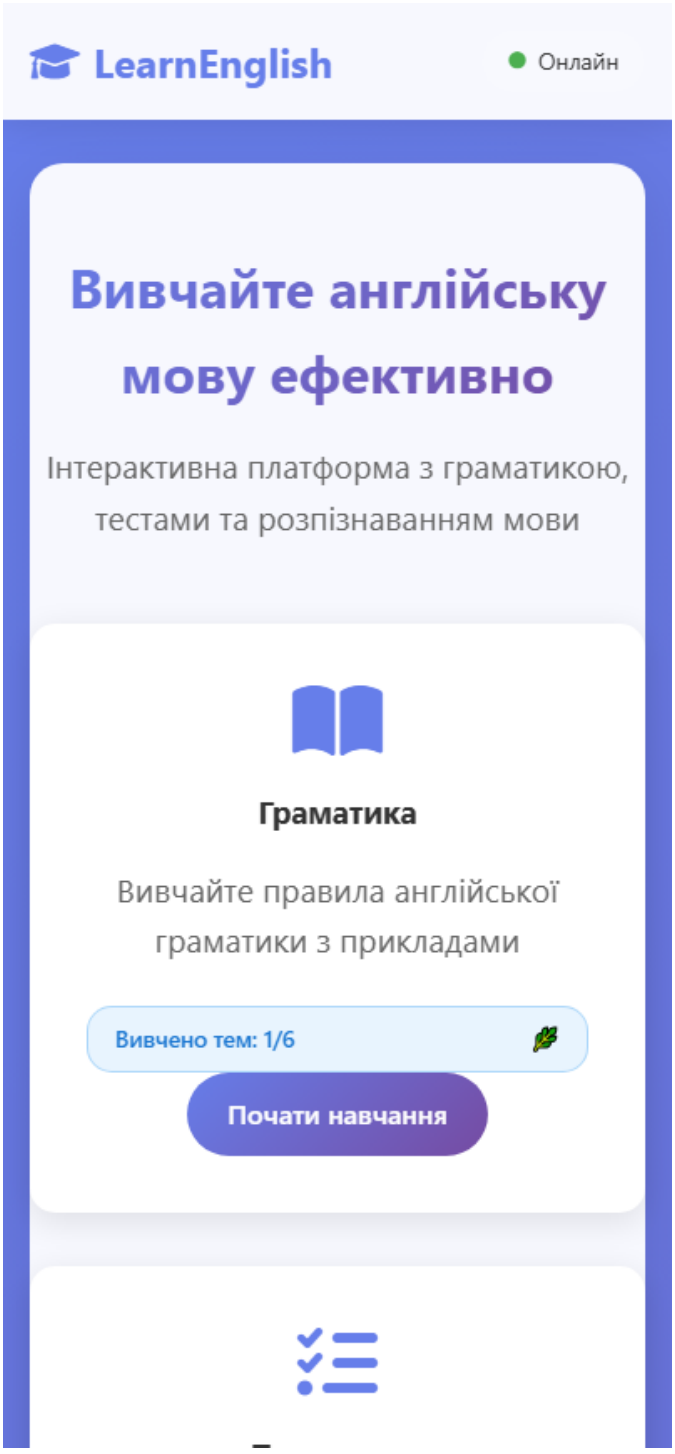


Рисунок 4.2 – Вигляд головної сторінки з мобільного пристрою

Тестування продуктивності та швидкодії. Наступним етапом розробки стало тестування продуктивності вебсистеми. Швидкодія була перевірена за допомогою «Lighthouse», результати тестування головної сторінки представлено на рисунку 4.3.

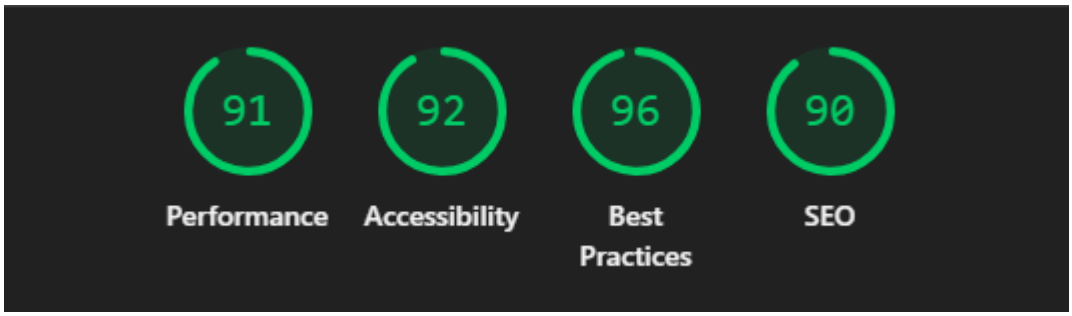


Рисунок 4.3 – Результати тестування в «Lighthouse»

Результати тестування на фізичному пристрої (ПК з Windows 10, браузер Google Chrome) показали, що головна сторінка вебсистеми завантажується в середньому за 0,7 секунд.

На рисунку 4.4 представлено рекомендації після аналізу «Lighthouse», щодо можливих покращень.

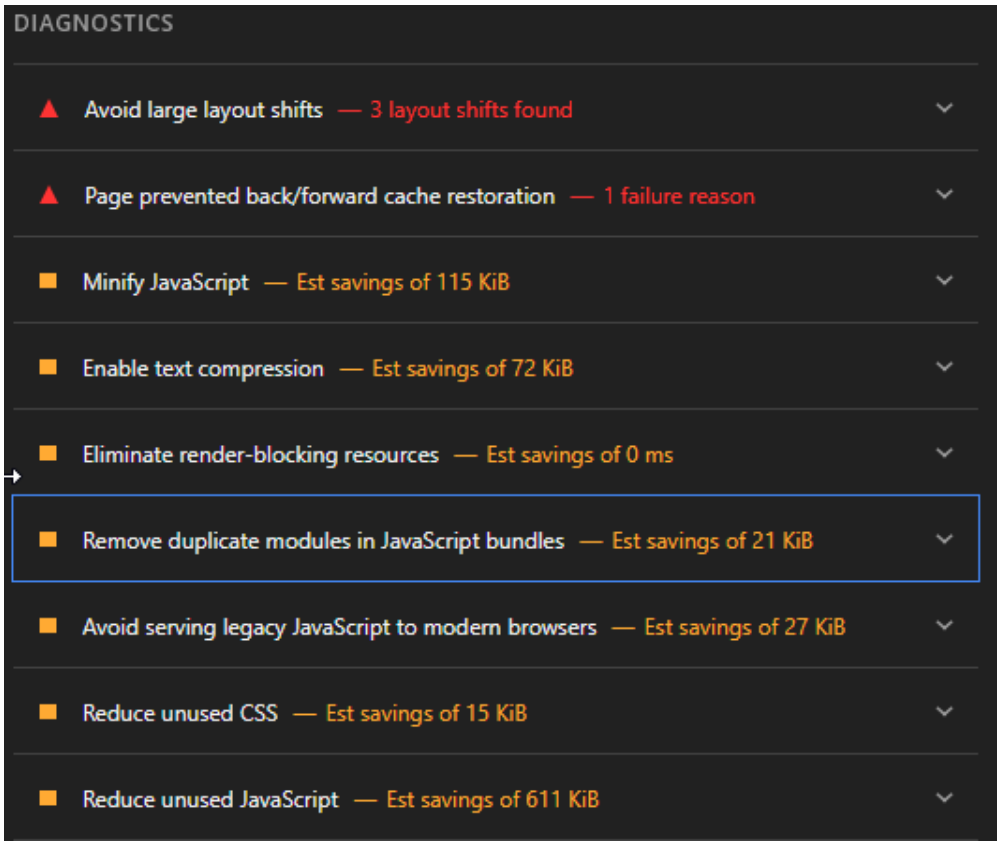


Рисунок 4.4 – Рекомендації щодо покращення від «Lighthouse»

Також у вебсистемі було реалізовано функціонал, який надсилає повідомлення до консолі, що може бути корисним для виявлення різних помилок та тестування вебсистеми. Результати запуску вебсистеми та логи продемонстровано на рисунку 4.5.

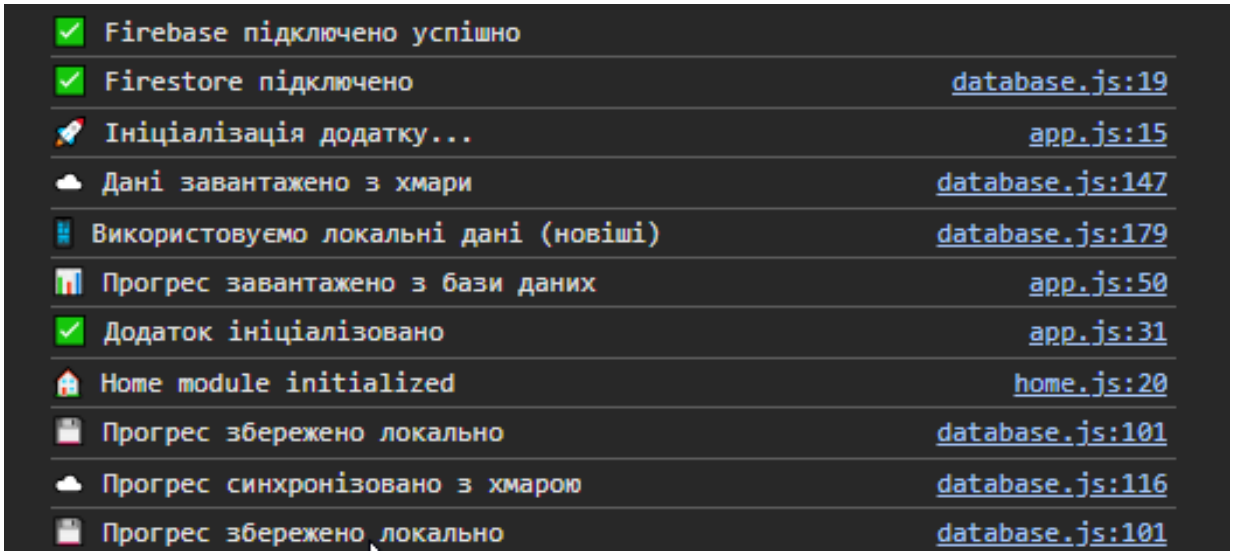


Рисунок 4.5 – Консольні логи при запуску вебсистеми

Тестування сумісності: було проведено перевірку сумісності з різними браузерами. Тестування охоплювало такі середовища:

1. Google Chrome (версії 112124, Windows 10);
2. Mozilla Firefox (версії 98115, Windows 10);
3. Microsoft Edge (версія 122, Windows 10).

Усі основні функції вебсистеми, включаючи голосовий ввід, відображення інтерфейсу, та оновлення статистики.

Тестування поведінки при втраті з’єднання: проводилася імітація втрати доступу до Інтернету та обмеженого доступу до сторонніх сервісів.

У таких ситуаціях система коректно виводила повідомлення про помилку та зберігала дані до локального сховища (див. рисунок 4.6).

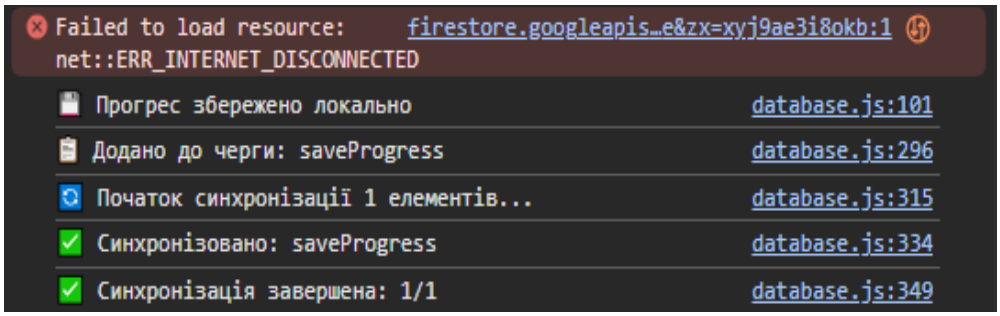


Рисунок 4.6 – Логи при відключені від інтернету

Завдяки ретельному тестуванню продуктивності, сумісності та поведінки в критичних умовах вебсистема для вивчення англійської мови довела свою стабільність, ефективність та зручність користування в широкому діапазоні середовищ і сценаріїв.

4.3 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для коректного запуску вебсистеми для вивчення англійської мови на різних пристроях і платформах. Оскільки система є веб-орієнтованою, основними факторами є сумісність із браузером, швидкодія клієнтського пристрою та наявність стабільного підключення до Інтернету. Детальні параметри мінімальної та рекомендованої конфігурації наведено в таблиці 4.1.

Таблиця 4.1 Програмно-апаратні вимоги

Варіант	Мінімальні	Рекомендовані
Клієнтський пристрій	ОС: Windows 10 / Windows 11 / iOS 13 Браузер: Chrome 90+, Firefox 88+, Safari 13+ Процесор: 2 ядра, 1.8 ГГц ОЗП: 2 ГБ	ОС: Windows 11 / Android 12 / iOS 15 Браузер: Chrome 115+, Firefox 115+, Safari 15+ Процесор: 4 ядра, 2.4 ГГц ОЗП: 4 ГБ

У головному меню користувач має змогу:

1. обрати розділ для навчання (граматика, тести, вимова);
2. переглянути прогрес;
3. переглянути статус синхронізації з базою даних.

Для початку запису мовлення потрібно натиснути кнопку «Записати» у модулі «Вимова», дозволити доступ до мікрофона (якщо не зроблено раніше) (див. рисунок 4.7).

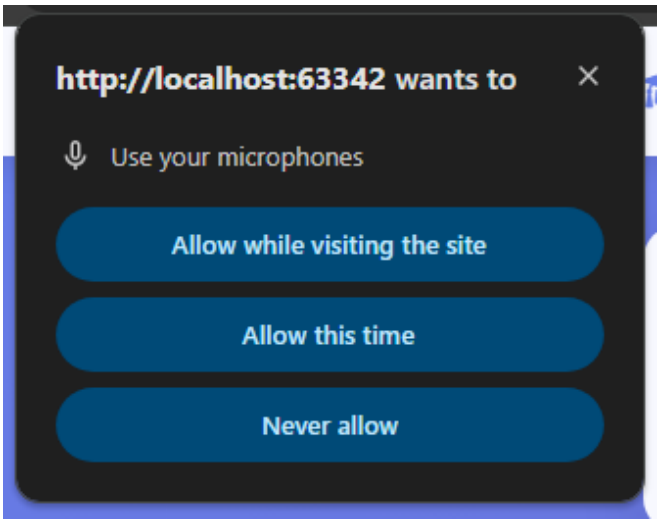


Рисунок 4.7 – Запит на дозвіл використання мікрофону

Після цього користувач може перейти до відповідного розділу, озвучити фразу, і отримати зворотний зв’язок щодо її правильності та граматичної структури.

4.3 Висновки

У четвертому розділі було проведено тестування розвиваючої вебсистеми для вивчення англійської мови. Було перевірено основні функції, поведінку вебсистеми при відключенні інтернету, а також перевірено поведінку інтерфейсу при нестандартних сценаріях використання. Крім того, розроблено інструкцію користувача, яка містить технічні вимоги до клієнтських пристроїв та детально описує основні етапи взаємодії з функціоналом веб-застосунку.

ВИСНОВКИ

У результаті дослідження було розроблено розвиваючу вебсистему з вивчення англійської мови «LearnEnglish», яка забезпечує ефективне та цікаве вивчення англійської мови для кінцевого користувача, використовуючи багато метрик статистики та систему досягнень, що збільшує інтерес до вивчення матеріалу.

Було проведено аналіз стану вебсистем для вивчення англійської мови, зокрема були розглянуті існуючі рішення серед засобів для вивчення англійської мови, були розглянуті їх переваги та недоліки в порівнянні з розроблюваною системою та встановлено, що реалізація власної розвиваючої вебсистеми з вивчення англійської мови є доцільною.

Беручи до уваги результати аналізу, були визначені основні задачі бакалаврської кваліфікаційної роботи, були обрані методи для розробки вебсистеми. Було проведено аналіз та обґрунтування вибору засобів для реалізації вебсистеми. В якості мови програмування було обрано JavaScript, середовище розробки JetBrains WebStorm.

Відповідно до поставлених задач було виконано такі завдання:

1. Розроблено модель розвиваючої вебсистеми для вивчення англійської мови, яка відображає основний функціонал.

2. Реалізовано необхідний функціонал вебсистеми для вивчення англійської мови.

3. Розроблено привабливий та зручний інтерфейс користувача.

4. Проведено тестування вебсистеми.

5. Розроблено інструкцію користувача.

Також було розроблено модулі для тестування, вивчення граматики та практики вимови, які забезпечують основний функціонал вебсистеми.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Причепа В.О. Сучасні технології в електронному навчанні: розробка вебсистем для вивчення мов / В.О. Причепа, О.Я. Стахов // Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)», Вінниця, 2025. [Електронний ресурс] – режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/24980 (дата звернення: 14.05.2025).

2. Колонуто Д.К. Сучасні технології у викладанні іноземних мов / Д.К. Колонуто // Науковий вісник Ужгородського національного університету – с. 92-94.

3. Duolingo. [Електронний ресурс] – режим доступу: https://uk.duolingo.com/ (дата звернення: 14.05.2025).

4. Babbel. [Електронний ресурс] – режим доступу: https://ua.babbel.com/ (дата звернення: 04.04.2025).

5. Busuu. [Електронний ресурс] – режим доступу: https://www.busuu.com/en (дата звернення: 05.04.2025).

6. Firebase Console [Електронний ресурс] – режим доступу: https://firebase.google.com/ (дата звернення: 06.04.2025).

7. Using the Web Speech API. [Електронний ресурс] – режим доступу: https: developer.mozilla.org/enUS/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API (дата звернення: 12.04.2025).

8. What is a UML diagram. [Електронний ресурс] – режим доступу: https://miro.com/diagramming/what-is-a-uml-diagram/ (дата звернення: 16.04.2025).

9. Додаток Diagrams.net. [Електронний ресурс] – режим доступу: https://app.diagrams.net/ (дата звернення: 20.04.2025).

10. Ерік М. Пришвидшений курс Python. Львів: Видавництво старого лева, 2021. 600с.

11. PHP Tutorial. [Електронний ресурс] – режим доступу: https: https://www.w3schools.com/php/ (дата звернення: 23.04.2025).

12. David Flanagan, JavaScript: The Definitive Guide: Master the World’s Most-Used Programming Language 7th Edition, O’Rielly, 2020. 706 p.

13. Visual Studio Code. [Електронний ресурс] – режим доступу: https: https://code.visualstudio.com/ (дата звернення: 01.05.2025).

14. WebStorm IDE. [Електронний ресурс] – режим доступу: https://www.jetbrains.com/ru-ru/webstorm/ (дата звернення: 04.05.2025).

15. Atom IDE. [Електронний ресурс] – режим доступу: https://atom-editor.cc/ (дата звернення: 05.05.2025).

16. Що таке мапа сайту? [Електронний ресурс] – режим доступу: https://theinweb.media/what-is-sitemap/ (дата звернення: 10.05.2025).

17. Kevi Wilson, The Absolute Beginner's Guide to HTML and CSS: A Step-by-Step Guide with Examples and Lab Exercises, Apress, 2023. 240 p.

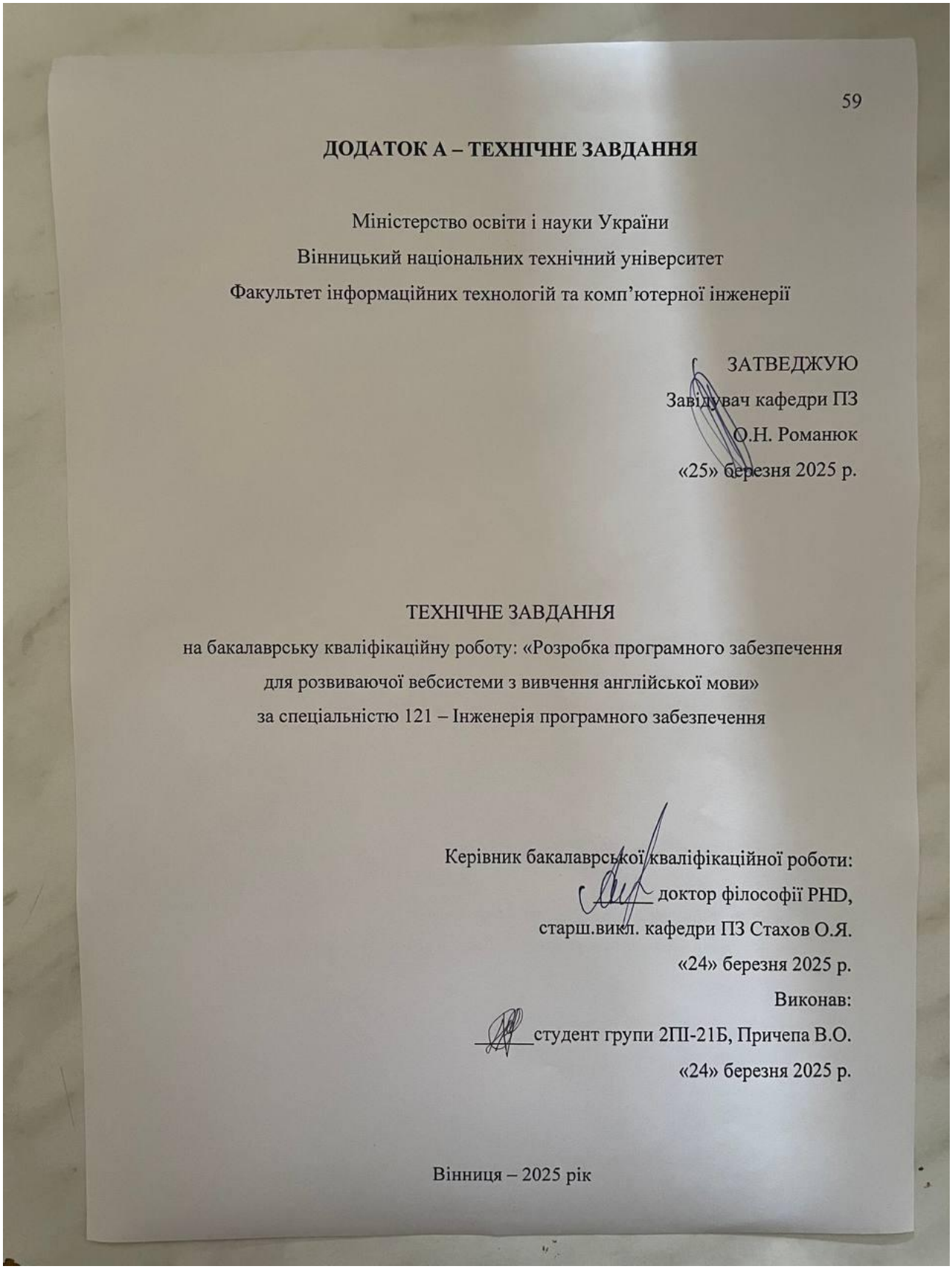
18. Mike McGrath, CSS in easy steps, In Eeasy Steps, 2020. 192 p.

19. Тестування програмного забезпечення: типи, види та застосування. [Електронний ресурс] – режим доступу: https://foxminded.ua/testuvannia-prohramnoho-zabezpechennia/ (дата звернення: 20.05.2025).

20. Chrome DevTools: налаштування, можливості та способи перевірки коду. URL: https://dou.ua/lenta/articles/chrome-dev-tools-guide/ (дата звернення: 22.05.2025).

ДОДАТКИ

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ



59

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ

Міністерство освіти і науки України
Вінницький національних технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕДЖУЮ
Завідувач кафедри ПЗ
О.Н. Романюк
«25» березня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на бакалаврську кваліфікаційну роботу: «Розробка програмного забезпечення для розвиваючої вебсистеми з вивчення англійської мови»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
доктор філософії PHD,
старш.викл. кафедри ПЗ Стахов О.Я.
«24» березня 2025 р.
Виконав:
студент групи 2ПІ-21Б, Причепа В.О.
«24» березня 2025 р.

Вінниця – 2025 рік

1. Найменування та галузь застосування.

Бакалаврська кваліфікаційна робота: «Розробка програмного забезпечення для розвиваючої вебсистеми з вивчення англійської мови».

Галузь застосування – навчальні та розвиваючі вебресурси.

2. Підстава для розробки.

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 від «20» березня 2025 р. від вищого навчального закладу.

3. Мета та призначення розробки

Метою дослідження є покращення умов для вивчення англійської мови, за рахунок використання інтерактивного підходу до вивчення, що охоплює різні аспекти.

Призначення роботи – розробка вебсистеми для вивчення англійської мови.

4.Вихідні дані для проведення НДР

1. 10 tips for creating the perfect eLearning website [Електронний ресурс] – режим доступу: https://elearningindustry.com/10-tips-creating-perfect-elearning-website

2. David Flanagan, JavaScript: The Definitive Guide: Master the World’s Most-Used Programming Language 7th Edition, O’Rielly, 2020. 706 p.

3. Malcolm Mann, Steve Taylore-Knowles, Destination C1-C2, Macmillan Education, 2020. 312 p.

5. Технічні вимоги

ОС: Windows 10 / Windows 11 / iOS 13

Браузер: Chrome 90+, Firefox 88+, Safari 13+

Процесор: 2 ядра, 1.8 ГГц

ОЗП: 2 ГБ

6. Конструктивні вимоги

Вебсистема має відповідати ергономічним та технічним вимогам. Текстова та графічна документація повинна відповідати ДСТУ.

Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до бакалаврської кваліфікаційної роботи;

– технічне завдання;

– лістинги програми.

7. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

8. Стадії та етапи розробки

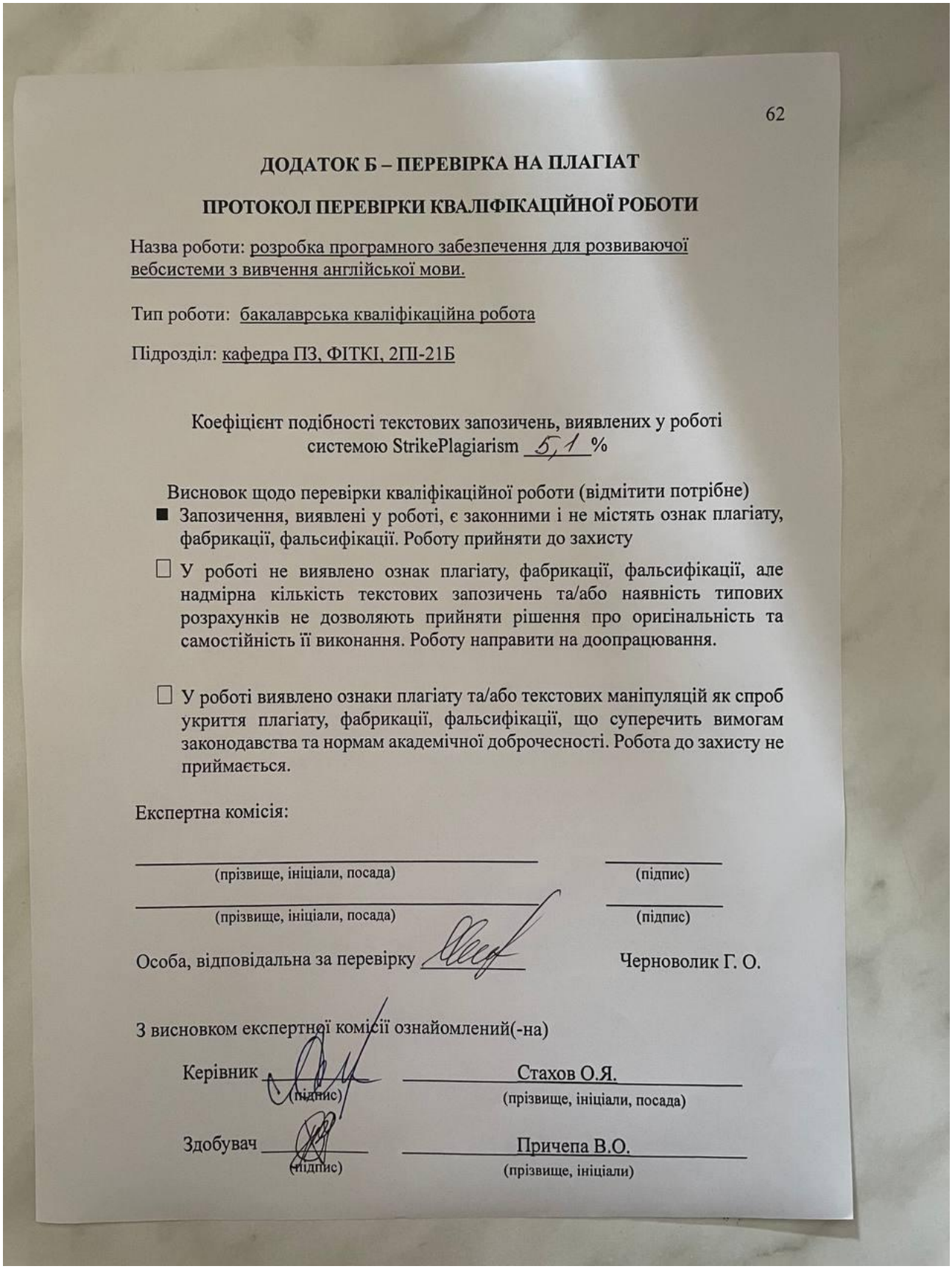
№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз стану питання та постановка задач дослідження	25.03.2025 09.04.2025
2	Аналіз вхідних та вихідних даних системи	10.04.2025 17.04.2025
4	Розробка моделі та алгоритмів системи	18.04.2025 27.04.2025
5	Розробка інтерфейсу	28.04.2025 05.05.2025
6	Розробка функціональних модулів	06.05.2025 16.05.2025
7	Тестування вебсистеми	17.05.2025 21.05.2025
8	Оформлення матеріалів до захисту БКР	22.05.2025 30.05.2025

9. Порядок контролю та прийняття

Всі етапи бакалаврської кваліфікаційної роботи контролюються науковим керівником згідно плану. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

Дозволяється корегування бакалаврської кваліфікаційної роботи.

ДОДАТОК Б – ПЕРЕВІРКА НА ПЛАГІАТ



62

ДОДАТОК Б – ПЕРЕВІРКА НА ПЛАГІАТ

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: розробка програмного забезпечення для розвиваючої вебсистеми з вивчення англійської мови.

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ: кафедра ПЗ, ФІТКІ, 2ПІ-21Б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 5,1 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак 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 плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Стахов О.Я.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Причепа В.О.
(підпис) (прізвище, ініціали)

ДОДАТОК В – ЛІСТИНГ ПРОГРАМИ

Лістинг файлу home.js

```
import app from './app.js';

class HomeModule {
    constructor() {
        this.globalStats = null;
        this.init();
    }

    async init() {

        await this.waitForApp();

        this.setupWelcomeMessage();
        this.setupQuickActions();
        this.setupProgressOverview();
        this.setupRecommendations();
        this.loadGlobalStats();

        console.log('🏠 Home module initialized');
    }

    async waitForApp() {
        while (!app.isInitialized) {
            await new Promise(resolve => setTimeout(resolve, 100));
        }
    }

    setupWelcomeMessage() {
        const progress = app.getProgress();
        const welcomeElement = document.querySelector('.hero h1');

        if (welcomeElement && progress.daysLearning > 1) {
            const timeOfDay = this.getTimeOfDay();
            const streakMessage = progress.streakDays > 1 ? ` Ваша серія: ${progress.streakDays} днів! 🔥` : '';
            welcomeElement.textContent = `${timeOfDay}! Продовжуємо вивчати англійську.${streakMessage}`;
        }
    }
```

```
getTimeOfDay() {
    const hour = new Date().getHours();
    if (hour < 12) return 'Доброго ранку';
    if (hour < 17) return 'Добрий день';
    if (hour < 22) return 'Добрий вечір';
    return 'Доброї ночі';
}

setupQuickActions() {
    // Відстеження кліків по картках
    document.querySelectorAll('.card-btn').forEach(btn => {
        btn.addEventListener('click', (e) => {
            const card = e.target.closest('.card');
            const cardTitle = card.querySelector('h3').textContent;
            this.trackQuickAction(cardTitle);
        });
    });


    this.addDynamicRecommendations();
}

addDynamicRecommendations() {
    const progress = app.getProgress();
    const cards = document.querySelectorAll('.card');

    cards.forEach(card => {
        const title = card.querySelector('h3').textContent;
        let recommendation = '';
        let progress_level = '';

        switch(title) {
            case 'Граматика':
                const grammarTopics = progress.grammarTopicsStudied || 0;
                if (grammarTopics === 0) {
                    recommendation = 'Почніть з Present Simple';
                    progress_level = 'beginner';
                } else {
                    recommendation = `Вивчено тем: ${grammarTopics}/6`;
                    progress_level = grammarTopics >= 4 ? 'advanced' : 'intermediate';
```

```
            }
            break;

        case 'Тестування':
            const testsCompleted = progress.testsCompleted || 0;
            if (testsCompleted === 0) {
                recommendation = 'Спробуйте перший тест';
                progress_level = 'beginner';
            } else {
                const avgScore = Math.round(app.getAverageScore());
                recommendation = `Середній бал: ${avgScore}%`;
                progress_level = avgScore >= 80 ? 'advanced' : avgScore >= 60 ? 'intermediate' : 'beginner';
            }
            break;

        case 'Вимова':
            const phrasesPracticed = progress.phrasesPracticed || 0;
            if (phrasesPracticed === 0) {
                recommendation = 'Запишіть першу фразу';
                progress_level = 'beginner';
            } else {
                recommendation = `Практиковано: ${phrasesPracticed} фраз`;
                progress_level = phrasesPracticed >= 20 ? 'advanced' : phrasesPracticed >= 5 ? 'intermediate' : 'beginner';
            }
            break;
    }

    if (recommendation) {
        const existing = card.querySelector('.card-recommendation');
        if (existing) existing.remove();

        const recElement = document.createElement('div');
        recElement.className = `card-recommendation ${progress_level}`;
        recElement.innerHTML = `
            <span class="rec-text">${recommendation}</span>
            <span class="rec-level">${this.getLevelIcon(progress_level)}</span>
        `;
        card.querySelector('p').after(recElement);
    }
  });
}
```

```
getLevelIcon(level) {
    switch(level) {
        case 'beginner': return '🌱';
        case 'intermediate': return '🌿';
        case 'advanced': return '🌳';
        default: return '';
    }
}

setupProgressOverview() {
    const progress = app.getProgress();


    this.addProgressCircles();


    this.addRecentActivity();


    this.addMotivationalMessage();


    this.addQuickActions();
}

addProgressCircles() {
    const statItems = document.querySelectorAll('.stat-item');

    statItems.forEach(item => {
        const icon = item.querySelector('i');
        if (icon && !item.querySelector('.progress-ring')) {
            const progressRing = this.createProgressRing(this.getStatProgress(item));
            icon.parentElement.style.position = 'relative';
            icon.parentElement.appendChild(progressRing);
        }
    });
}

createProgressRing(percentage) {
    const ring = document.createElement('div');
```

```
    ring.className = 'progress-ring';
    ring.innerHTML = `
        <svg width="60" height="60">
            <circle cx="30" cy="30" r="25" fill="transparent" stroke="#e0e0e0" stroke-width="4"/>
            <circle cx="30" cy="30" r="25" fill="transparent" stroke="#667eea" stroke-width="4"
                stroke-dasharray="${2 * Math.PI * 25}"
                stroke-dashoffset="${2 * Math.PI * 25 * (1 - percentage / 100)}"
                transform="rotate(-90 30 30)"
                style="transition: stroke-dashoffset 1s ease"/>
        </svg>
    `;

    return ring;
}

getStatProgress(statItem) {
    const value = statItem.querySelector('span').textContent;
    const numValue = parseInt(value);

    if (statItem.textContent.includes('Тестів')) {
        return Math.min((numValue / 10) * 100, 100);
    } else if (statItem.textContent.includes('Фраз')) {
        return Math.min((numValue / 50) * 100, 100);
    } else {
        return Math.min(numValue, 100);
    }
}

addRecentActivity() {
    const progress = app.getProgress();
    const activitySection = document.createElement('div');
    activitySection.className = 'recent-activity';
    activitySection.innerHTML = `
        <h3>📈 Недавня активність</h3>
        <div class="activity-list">
            ${this.generateActivityList(progress)}
        </div>
    `;

    document.querySelector('.stats-overview').after(activitySection);
}
```

```
generateActivityList(progress) {
    const activities = [];
    const today = new Date().toLocaleDateString('uk-UA');

    if (progress.testsCompleted > 0) {
        activities.push({
            icon: 'fas fa-check-circle',
            text: `Пройдено тестів: ${progress.testsCompleted}`,
            subtext: `Середній бал: ${Math.round(app.getAverageScore())}%`,
            date: today,
            type: 'test'
        });
    }

    if (progress.phrasesPracticed > 0) {
        activities.push({
            icon: 'fas fa-microphone',
            text: `Практиковано фраз: ${progress.phrasesPracticed}`,
            subtext: `Точність: ${Math.round(progress.speechAccuracy || 0)}%`,
            date: today,
            type: 'speech'
        });
    }

    if (progress.grammarTopicsStudied > 0) {
        activities.push({
            icon: 'fas fa-book',
            text: `Вивчено граматичних тем: ${progress.grammarTopicsStudied}`,
            subtext: 'Продовжуйте вивчення!',
            date: today,
            type: 'grammar'
        });
    }

    if (progress.streakDays > 1) {
        activities.push({
            icon: 'fas fa-fire',
            text: `Серія навчання: ${progress.streakDays} днів`,
            subtext: 'Продовжуйте в тому ж дусі!',
            date: today,
```

```
            type: 'streak'
        });
    }

    if (activities.length === 0) {
        return '<div class="no-activity">🚀 Почніть навчання, щоб побачити активність</div>';
    }

    return activities.map(activity => `
        <div class="activity-item ${activity.type}">
            <i class="${activity.icon}"></i>
            <div class="activity-content">
                <div class="activity-text">${activity.text}</div>
                <div class="activity-subtext">${activity.subtext}</div>
                <div class="activity-date">${activity.date}</div>
            </div>
        </div>
    `).join('');
}

addMotivationalMessage() {
    const progress = app.getProgress();
    const totalPoints = app.getTotalPoints();

    let message = '';
    let level = '';
    let nextGoal = '';

    if (totalPoints === 0) {
        message = 'Розпочніть своє навчання сьогодні! 🎯';
        level = 'Новачок';
        nextGoal = 'Пройдіть перший тест або урок граматики';
    } else if (totalPoints < 50) {
        message = 'Відмінний початок! Продовжуйте в тому ж дусі. 💪';
        level = 'Початківець';
        nextGoal = 'Мета: набрати 50 балів';
    } else if (totalPoints < 150) {
        message = 'Ви робите чудовий прогрес! 📈';
        level = 'Студент';
        nextGoal = 'Мета: набрати 150 балів';
    } else if (totalPoints < 300) {
```

```
            message = 'Ваші зусилля приносять результат! 🌟';
            level = 'Ентузіаст';
            nextGoal = 'Мета: набрати 300 балів';
        } else {
            message = 'Ви справжній майстер навчання! 🏆';
            level = 'Експерт';
            nextGoal = 'Продовжуйте вдосконалюватися!';
        }

        const motivationCard = document.createElement('div');
        motivationCard.className = 'motivation-card';
        motivationCard.innerHTML = `
            <div class="motivation-content">
                <div class="motivation-level">${level}</div>
                <div class="motivation-message">${message}</div>
                <div class="motivation-points">${totalPoints} балів</div>
                <div class="motivation-goal">${nextGoal}</div>
            </div>
            <div class="motivation-icon">
                <i class="fas fa-trophy"></i>
            </div>
        `;

        document.querySelector('.hero').after(motivationCard);
    }

    addQuickActions() {
        const quickActionsSection = document.createElement('section');
        quickActionsSection.className = 'quick-actions';
        quickActionsSection.innerHTML = `
            <h2>⚡ Швидкі дії</h2>
            <div class="actions-grid">
                <button class="action-btn" onclick="this.exportData()">
                    <i class="fas fa-download"></i>
                    <span>Експорт даних</span>
                </button>
                <button class="action-btn" onclick="this.importData()">
                    <i class="fas fa-upload"></i>
                    <span>Імпорт даних</span>
                </button>
                <button class="action-btn" onclick="this.showConnectionStatus()">
```

```
            <i class="fas fa-wifi"></i>
            <span>Статус підключення</span>
        </button>
        <button class="action-btn" onclick="this.syncNow()">
            <i class="fas fa-sync"></i>
            <span>Синхронізувати</span>
        </button>
    </div>
`;

document.querySelector('.recent-activity').after(quickActionsSection);


this.setupQuickActionHandlers();
}

setupQuickActionHandlers() {
    // Експорт даних
    window.exportData = async () => {
        if (app.dbManager) {
            await app.dbManager.exportUserData();
        } else {
            alert('Функція експорту недоступна');
        }
    };


    window.importData = () => {
        const input = document.createElement('input');
        input.type = 'file';
        input.accept = '.json';
        input.onchange = async (e) => {
            const file = e.target.files[0];
            if (file && app.dbManager) {
                const result = await app.dbManager.importUserData(file);
                alert(result.message);
                if (result.success) {
                    location.reload();
                }
            }
        };
```

```
        input.click();
    };


    window.showConnectionStatus = () => {
        const status = app.dbManager ? app.dbManager.getConnectionStatus() : null;
        if (status) {
            alert(`🔌 Статус підключення:\n\n` +
                `Онлайн: ${status.isOnline ? 'Так' : 'Ні'}\n` +
                `База даних: ${status.hasDatabase ? 'Підключена' : 'Недоступна'}\n` +
                `Очікує синхронізації: ${status.queuedItems}\n` +
                `Остання синхронізація: ${status.lastSync}\n` +
                `ID користувача: ${status.userId}`);
        } else {
            alert('❌ Інформація про підключення недоступна');
        }
    };


    window.syncNow = async () => {
        if (app.dbManager && app.dbManager.database) {
            try {
                await app.dbManager.syncOfflineData();
                alert('✅ Синхронізація завершена!');
            } catch (error) {
                alert('❌ Помилка синхронізації: ' + error.message);
            }
        } else {
            alert('⚠️ Синхронізація недоступна (немає підключення до бази даних)');
        }
    };
}

async loadGlobalStats() {
    if (!app.dbManager) return;

    try {
        this.globalStats = await app.dbManager.getGlobalStats();

        if (this.globalStats.isGlobal && this.globalStats.totalUsers > 1) {
            this.displayGlobalStats();
```

```
            this.addLeaderboard();
        }
    } catch (error) {
        console.error('Помилка завантаження глобальної статистики:', error);
    }
}

displayGlobalStats() {
    const globalStatsSection = document.getElementById('global-stats');
    if (globalStatsSection && this.globalStats) {
        globalStatsSection.style.display = 'block';

        document.getElementById('total-users').textContent = this.globalStats.totalUsers;
        document.getElementById('global-average').textContent = this.globalStats.averageScore + '%';
        document.getElementById('your-rank').textContent = this.globalStats.currentUserRank;
    }
}

addLeaderboard() {
    if (!this.globalStats || !this.globalStats.topScores) return;

    const leaderboardSection = document.createElement('section');
    leaderboardSection.className = 'leaderboard';
    leaderboardSection.innerHTML = `
        <h2>🏆 Топ учнів</h2>
        <div class="leaderboard-list">
            ${this.globalStats.topScores.slice(0, 5).map((user, index) => `
                <div class="leaderboard-item ${index < 3 ? 'top-three' : ''}">
                    <div class="rank">
                        ${this.getRankIcon(index + 1)}
                        <span>${index + 1}</span>
                    </div>
                    <div class="user-info">
                        <div class="user-score">${user.score} балів</div>
                        <div class="user-tests">${user.testsCompleted} тестів</div>
                    </div>
                </div>
            `).join('')}
        </div>
        <div class="leaderboard-note">
            ${this.globalStats.currentUserRank !== '-' ?
```

```
        `Ви на ${this.globalStats.currentUserRank} місці з ${this.globalStats.totalUsers} учнів` :
        'Пройдіть тести, щоб потрапити в рейтинг'}
        </div>
    `;

    const globalStats = document.getElementById('global-stats');
    if (globalStats) {
        globalStats.after(leaderboardSection);
    }
}

getRankIcon(rank) {
    switch(rank) {
        case 1: return '🥇';
        case 2: return '🥈';
        case 3: return '🥉';
        default: return '🏅';
    }
}

setupRecommendations() {
    const progress = app.getProgress();
    const recommendations = this.generateRecommendations(progress);

    if (recommendations.length > 0) {
        this.addRecommendationsSection(recommendations);
    }
}

generateRecommendations(progress) {
    const recommendations = [];


    if (progress.grammarTopicsStudied === 0) {
        recommendations.push({
            type: 'grammar',
            title: 'Почніть з граматики',
            description: 'Вивчіть базові часи англійської мови',
            action: 'Перейти до граматики',
            link: 'grammar.html',
            priority: 1,
```

```
            icon: 'fas fa-book'
        });
    } else if (progress.grammarTopicsStudied < 6) {
        recommendations.push({
            type: 'grammar',
            title: 'Продовжіть вивчення граматики',
            description: `Вивчено ${progress.grammarTopicsStudied} з 6 тем`,
            action: 'Продовжити',
            link: 'grammar.html',
            priority: 2,
            icon: 'fas fa-book'
        });
    }


    if (progress.testsCompleted === 0) {
        recommendations.push({
            type: 'test',
            title: 'Пройдіть перший тест',
            description: 'Оцініть свій поточний рівень знань',
            action: 'Пройти тест',
            link: 'tests.html',
            priority: 2,
            icon: 'fas fa-tasks'
        });
    } else if (app.getAverageScore() < 70) {
        recommendations.push({
            type: 'improvement',
            title: 'Покращіть результати',
            description: `Поточний середній бал: ${Math.round(app.getAverageScore())}%`,
            action: 'Повторити граматику',
            link: 'grammar.html',
            priority: 1,
            icon: 'fas fa-chart-line'
        });
    }


    if (progress.phrasesPracticed === 0) {
        recommendations.push({
            type: 'speech',
```

```
            title: 'Практикуйте вимову',
            description: 'Розпочніть з простих фраз',
            action: 'Практикувати вимову',
            link: 'speech.html',
            priority: 3,
            icon: 'fas fa-microphone'
        });
    } else if (progress.phrasesPracticed < 10) {
        recommendations.push({
            type: 'speech',
            title: 'Продовжіть практику вимови',
            description: `Практиковано ${progress.phrasesPracticed} фраз`,
            action: 'Продовжити практику',
            link: 'speech.html',
            priority: 3,
            icon: 'fas fa-microphone'
        });
    }


    if (progress.streakDays === 1) {
        recommendations.push({
            type: 'streak',
            title: 'Збільшуйте серію навчання',
            description: 'Навчайтеся щодня для кращих результатів',
            action: 'Продовжити навчання',
            link: '#',
            priority: 2,
            icon: 'fas fa-fire'
        });
    }

    return recommendations.sort((a, b) => a.priority - b.priority).slice(0, 3);
}

addRecommendationsSection(recommendations) {
    const section = document.createElement('section');
    section.className = 'recommendations-section';
    section.innerHTML = `
        <h2>💡 Рекомендації для вас</h2>
        <div class="recommendations-grid">
```

```
            ${recommendations.map(rec => `
                <div class="recommendation-card ${rec.type}">
                    <div class="rec-icon">
                        <i class="${rec.icon}"></i>
                    </div>
                    <div class="rec-content">
                        <h3>${rec.title}</h3>
                        <p>${rec.description}</p>
                        <a href="${rec.link}" class="recommendation-btn">${rec.action}</a>
                    </div>
                </div>
            `).join('')}
        </div>
    `;

    document.querySelector('.stats-overview').before(section);
  }

  trackQuickAction(action) {
    console.log(`Quick action: ${action}`);

    // Можна додати аналітику
    if (app.dbManager) {
      app.dbManager.queueForSync('trackAction', {
        action,
        timestamp: new Date().toISOString(),
        page: 'home'
      });
    }
  }
}
document.addEventListener('DOMContentLoaded', () => {
  if (document.querySelector('.hero') && document.querySelector('.feature-cards')) {
    new HomeModule();
  }
```

Лістинг файлу app.js

```
import dbManager from './database.js';

class EnglishLearningApp {
  constructor() {
```

```
    this.userProgress = null;
    this.currentPage = this.getCurrentPage();
    this.dbManager = dbManager;
    this.isInitialized = false;

    this.init();
}

async init() {
    console.log('🚀 Ініціалізація додатку...');


    await this.waitForDatabase();


    await this.loadUserProgress();


    this.setupGlobalEventListeners();
    this.updateGlobalStats();


    this.loadGlobalStats();

    this.isInitialized = true;
    console.log('✅ Додаток ініціалізовано');
}

async waitForDatabase(maxWait = 5000) {
    const startTime = Date.now();

    while (!this.dbManager && (Date.now() - startTime) < maxWait) {
        await new Promise(resolve => setTimeout(resolve, 100));
    }

    if (!this.dbManager) {
        console.warn('⚠️ База даних не ініціалізована, використовуємо localStorage');
    }
}

async loadUserProgress() {
```

```
    try {
        if (this.dbManager) {
            this.userProgress = await this.dbManager.loadProgress();
            console.log('📊 Прогрес завантажено з бази даних');
        } else {

            const saved = localStorage.getItem('englishLearningProgress');
            this.userProgress = saved ? JSON.parse(saved) : this.getDefaultProgress();
            console.log('📱 Прогрес завантажено з localStorage');
        }
    } catch (error) {
        console.error('❌ Помилка завантаження прогресу:', error);
        this.userProgress = this.getDefaultProgress();
    }
}

getCurrentPage() {
    const path = window.location.pathname;
    const page = path.split('/').pop().replace('.html', '') || 'index';
    return page === 'index' ? 'home' : page;
}

setupGlobalEventListeners() {

    this.updateActiveNavigation();


    document.addEventListener('visibilitychange', () => {
        if (!document.hidden) {
            this.updateGlobalStats();
        }
    });


    window.addEventListener('beforeunload', () => {
        this.saveProgress();
    });


    setInterval(() => {
        if (this.userProgress) {
```

```
            this.saveProgress();
        }
    }, 30000);
}

updateActiveNavigation() {
    const currentPage = this.getCurrentPage();
    const navLinks = document.querySelectorAll('.nav-link');

    navLinks.forEach(link => {
        link.classList.remove('active');
        const href = link.getAttribute('href');

        if (href.includes(currentPage) ||
            (currentPage === 'home' && href === 'index.html')) {
            link.classList.add('active');
        }
    });
}



async saveProgress() {
    if (!this.userProgress) return;

    try {
        this.userProgress.lastActivity = new Date().toDateString();
        this.userProgress.lastUpdated = new Date().toISOString();

        if (this.dbManager) {
            await this.dbManager.saveProgress(this.userProgress);
        } else {
            localStorage.setItem('englishLearningProgress', JSON.stringify(this.userProgress));
        }
    } catch (error) {
        console.error('❌ Помилка збереження прогресу:', error);
    }
}

updateProgress(category, value = 1, additionalData = {}) {
    if (!this.userProgress) return;
```

```
switch (category) {
    case 'testsCompleted':
        this.userProgress.testsCompleted += value;
        break;

    case 'totalScore':
        this.userProgress.totalScore += value;
        break;

    case 'phrasesPracticed':
        this.userProgress.phrasesPracticed += value;
        break;

    case 'grammarTopicsStudied':
        this.userProgress.grammarTopicsStudied += value;
        break;

    case 'speechAccuracy':
        this.userProgress.speechAttempts += 1;
        const prevAccuracy = this.userProgress.speechAccuracy || 0;
        const attempts = this.userProgress.speechAttempts;
        this.userProgress.speechAccuracy =
            ((prevAccuracy * (attempts - 1)) + value) / attempts;
        break;

    case 'grammarProgress':
        if (additionalData.topic) {
            if (!this.userProgress.grammarProgress) {
                this.userProgress.grammarProgress = {};
            }
            this.userProgress.grammarProgress[additionalData.topic] = value;
        }
        break;

    case 'testHistory':
        if (!this.userProgress.testHistory) {
            this.userProgress.testHistory = [];
        }
        const testResult = {
            date: new Date().toISOString(),
            score: value,
```

```
            level: additionalData.level || 'beginner',
            category: additionalData.category || 'mixed'
        };
        this.userProgress.testHistory.push(testResult);


        if (this.dbManager) {
            this.dbManager.saveTestResult(testResult);
        }
        break;

    case 'speechHistory':
        if (!this.userProgress.speechHistory) {
            this.userProgress.speechHistory = [];
        }
        this.userProgress.speechHistory.push({
            date: new Date().toISOString(),
            phrase: additionalData.phrase,
            accuracy: value
        });
        break;
  }


  this.updateActivityTracking();


  this.saveProgress();


  this.updateGlobalStats();


  this.checkAchievements();
}

updateActivityTracking() {
  const today = new Date().toDateString();
  const lastActivity = this.userProgress.lastActivity;

  if (lastActivity !== today) {
```

```
        const yesterday = new Date();
        yesterday.setDate(yesterday.getDate() - 1);

        if (lastActivity === yesterday.toDateString()) {

            this.userProgress.streakDays += 1;
        } else {

            this.userProgress.streakDays = 1;
        }

        this.userProgress.daysLearning += 1;
        this.userProgress.lastActivity = today;
    }
}

updateGlobalStats() {
    if (!this.userProgress) return;

    const elements = {
        'tests-completed': this.userProgress.testsCompleted || 0,
        'home-tests-completed': this.userProgress.testsCompleted || 0,
        'average-score': this.getAverageScore(),
        'home-average-score': this.getAverageScore(),
        'phrases-practiced': this.userProgress.phrasesPracticed || 0,
        'home-phrases-practiced': this.userProgress.phrasesPracticed || 0,
        'grammar-topics-studied': this.userProgress.grammarTopicsStudied || 0,
        'days-learning': this.userProgress.daysLearning || 1,
        'streak-days': this.userProgress.streakDays || 1,
        'total-points': this.getTotalPoints()
    };

    Object.entries(elements).forEach(([id, value]) => {
        const element = document.getElementById(id);
        if (element) {
            if (id.includes('score') || id.includes('accuracy')) {
                element.textContent = Math.round(value) + '%';
            } else {
                element.textContent = value;
            }
        }
```

```
    });
}

async loadGlobalStats() {
    if (!this.dbManager) return;

    try {
        const stats = await this.dbManager.getGlobalStats();
        this.displayGlobalStats(stats);
    } catch (error) {
        console.error('Помилка завантаження глобальної статистики:', error);
    }
}

displayGlobalStats(stats) {
    const globalStatsSection = document.getElementById('global-stats');
    if (!globalStatsSection) return;


    if (stats.isGlobal && stats.totalUsers > 1) {
        globalStatsSection.style.display = 'block';


        const totalUsersEl = document.getElementById('total-users');
        const globalAverageEl = document.getElementById('global-average');
        const yourRankEl = document.getElementById('your-rank');

        if (totalUsersEl) totalUsersEl.textContent = stats.totalUsers;
        if (globalAverageEl) globalAverageEl.textContent = stats.averageScore + '%';
        if (yourRankEl) yourRankEl.textContent = stats.currentUserRank;
    }
}

getAverageScore() {
    if (!this.userProgress || this.userProgress.testsCompleted === 0) return 0;
    return Math.round(this.userProgress.totalScore / this.userProgress.testsCompleted);
}

getTotalPoints() {
    if (!this.userProgress) return 0;
    return (this.userProgress.testsCompleted * 10) +
```

```
        (this.userProgress.phrasesPracticed * 5) +
        (this.userProgress.grammarTopicsStudied * 15);
}

checkAchievements() {
    const achievements = [
        {
            id: 'first-test',
            name: 'Перші кроки',
            description: 'Пройдено перший тест',
            condition: () => this.userProgress.testsCompleted >= 1,
            icon: 'fas fa-seedling'
        },
        {
            id: 'first-speech',
            name: 'Перше слово',
            description: 'Записано першу фразу',
            condition: () => this.userProgress.phrasesPracticed >= 1,
            icon: 'fas fa-microphone'
        },
        {
            id: 'week-streak',
            name: 'Гаряча серія',
            description: '7 днів поспіль навчання',
            condition: () => this.userProgress.streakDays >= 7,
            icon: 'fas fa-fire'
        },
        {
            id: 'excellent-student',
            name: 'Відмінник',
            description: 'Середній бал вище 90%',
            condition: () => this.getAverageScore() >= 90,
            icon: 'fas fa-star'
        },
        {
            id: 'grammar-master',
            name: 'Знавець граматики',
            description: 'Вивчено всі теми граматики',
            condition: () => {
                const grammarProgress = this.userProgress.grammarProgress || {};
                return Object.keys(grammarProgress).length >= 6 &&
```

```
                Object.values(grammarProgress).every(score => score > 0);
            },
            icon: 'fas fa-graduation-cap'
        },
        {
            id: 'speech-master',
            name: 'Майстер вимови',
            description: 'Практиковано 50 фраз',
            condition: () => this.userProgress.phrasesPracticed >= 50,
            icon: 'fas fa-volume-up'
        },
        {
            id: 'test-champion',
            name: 'Чемпіон тестів',
            description: 'Пройдено 20 тестів',
            condition: () => this.userProgress.testsCompleted >= 20,
            icon: 'fas fa-trophy'
        },
        {
            id: 'persistent-learner',
            name: 'Наполегливий учень',
            description: '30 днів навчання',
            condition: () => this.userProgress.daysLearning >= 30,
            icon: 'fas fa-calendar-check'
        }
    ];

    const currentAchievements = this.userProgress.achievements || [];

    achievements.forEach(achievement => {
        if (achievement.condition() && !currentAchievements.includes(achievement.id)) {
            currentAchievements.push(achievement.id);
            this.showAchievementNotification(achievement);
        }
    });

    this.userProgress.achievements = currentAchievements;
}

showAchievementNotification(achievement) {
    const notification = document.createElement('div');
```

```
notification.className = 'achievement-notification';
notification.innerHTML = `
    <div class="achievement-content">
        <i class="${achievement.icon}"></i>
        <div>
            <h4>🏆 Нове досягнення!</h4>
            <p><strong>${achievement.name}</strong></p>
            <small>${achievement.description}</small>
        </div>
    </div>
`;

notification.style.cssText = `
    position: fixed;
    top: 100px;
    right: 20px;
    background: linear-gradient(135deg, #667eea, #764ba2);
    color: white;
    padding: 1rem;
    border-radius: 10px;
    box-shadow: 0 10px 30px rgba(0,0,0,0.3);
    z-index: 10000;
    animation: slideInRight 0.5s ease-out;
    max-width: 300px;
`;

if (!document.querySelector('#achievement-animations')) {
    const style = document.createElement('style');
    style.id = 'achievement-animations';
    style.textContent = `
        @keyframes slideInRight {
            from { transform: translateX(100%); opacity: 0; }
            to { transform: translateX(0); opacity: 1; }
        }
        .achievement-notification .achievement-content {
            display: flex;
            align-items: center;
            gap: 1rem;
        }
        .achievement-notification i {
            font-size: 2rem;
```

```
            }
        `;
        document.head.appendChild(style);
    }

    document.body.appendChild(notification);

    setTimeout(() => {
        notification.style.animation = 'slideInRight 0.5s ease-out reverse';
        setTimeout(() => notification.remove(), 500);
    }, 5000);
}


formatTime(seconds) {
    const minutes = Math.floor(seconds / 60);
    const remainingSeconds = seconds % 60;
    return `${minutes}:${remainingSeconds.toString().padStart(2, '0')}`;
}

shuffleArray(array) {
    const shuffled = [...array];
    for (let i = shuffled.length - 1; i > 0; i--) {
        const j = Math.floor(Math.random() * (i + 1));
        [shuffled[i], shuffled[j]] = [shuffled[j], shuffled[i]];
    }
    return shuffled;
}



levenshteinDistance(str1, str2) {
    const matrix = [];

    for (let i = 0; i <= str2.length; i++) {
        matrix[i] = [i];
    }

    for (let j = 0; j <= str1.length; j++) {
        matrix[0][j] = j;
    }
```

```
    for (let i = 1; i <= str2.length; i++) {
        for (let j = 1; j <= str1.length; j++) {
            if (str2.charAt(i - 1) === str1.charAt(j - 1)) {
                matrix[i][j] = matrix[i - 1][j - 1];
            } else {
                matrix[i][j] = Math.min(
                    matrix[i - 1][j - 1] + 1,
                    matrix[i][j - 1] + 1,
                    matrix[i - 1][j] + 1
                );
            }
        }
    }

    return matrix[str2.length][str1.length];
}

getProgress() {
    return { ...this.userProgress };
}

async resetProgress() {
    if (confirm('⚠ Ви впевнені, що хочете скинути весь прогрес? Цю дію неможливо скасувати.')) {
        if (this.dbManager) {
            await this.dbManager.clearAllData();
        } else {
            localStorage.clear();
            location.reload();
        }
    }
}

getDefaultProgress() {
    return {
        testsCompleted: 0,
        totalScore: 0,
        phrasesPracticed: 0,
        grammarTopicsStudied: 0,
        speechAccuracy: 0,
        speechAttempts: 0,
```

```
            daysLearning: 1,
            streakDays: 1,
            lastActivity: new Date().toDateString(),
            lastUpdated: new Date().toISOString(),
            achievements: [],
            testHistory: [],
            speechHistory: [],
            grammarProgress: {
                'present-simple': 0,
                'past-simple': 0,
                'future-simple': 0,
                'articles': 0,
                'present-continuous': 0,
                'modal-verbs': 0
            }
        };
    }

}
const app = new EnglishLearningApp();
window.englishApp = app;
export default app;
```

ДОДАТОК Г – ІЛЮСТРАТИВНА ЧАСТИНА

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗВИВАЮЧОЇ ВЕБСИСТЕМИ З ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ

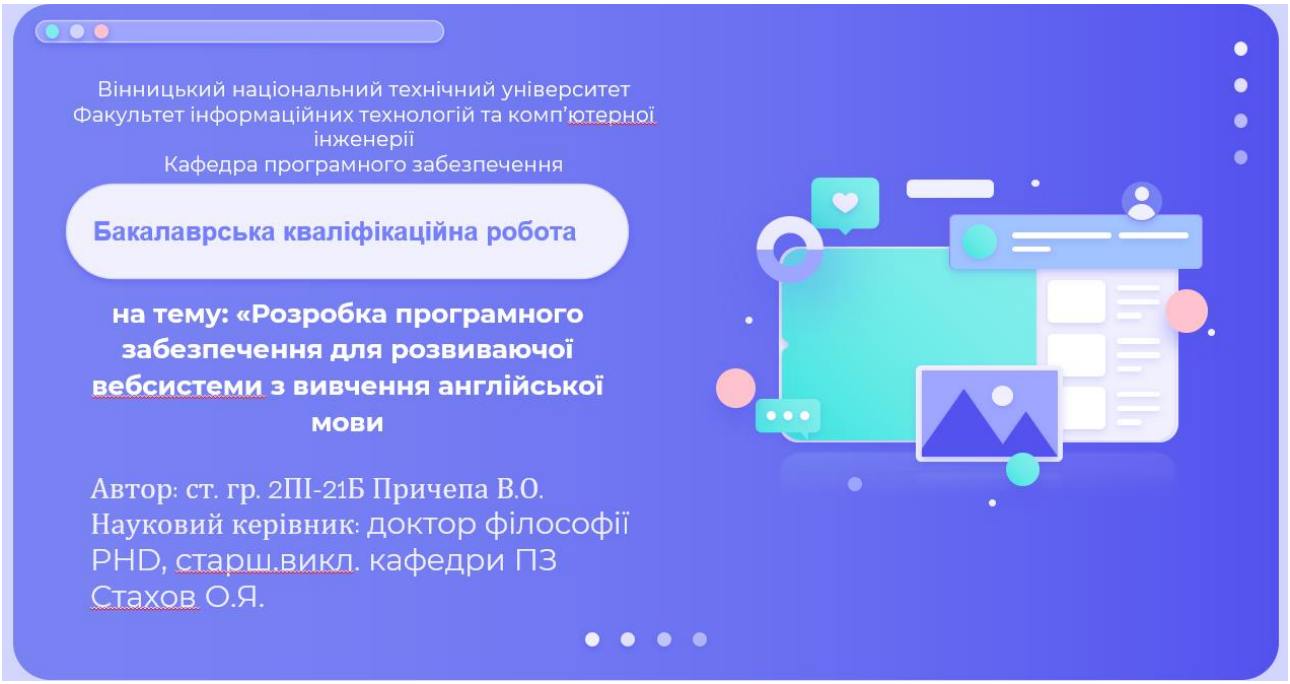


Рисунок Г.1 – Титульний слайд

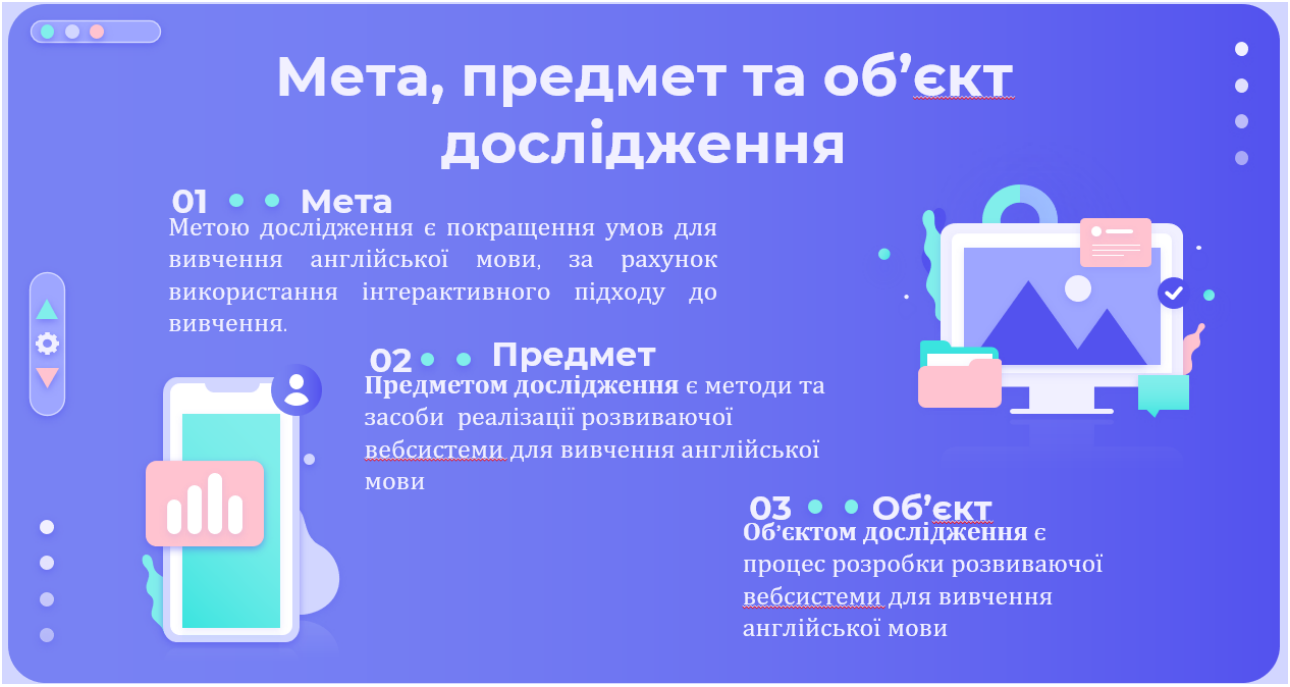


Рисунок Г.2 – Мета, предмет та об'єкт дослідження

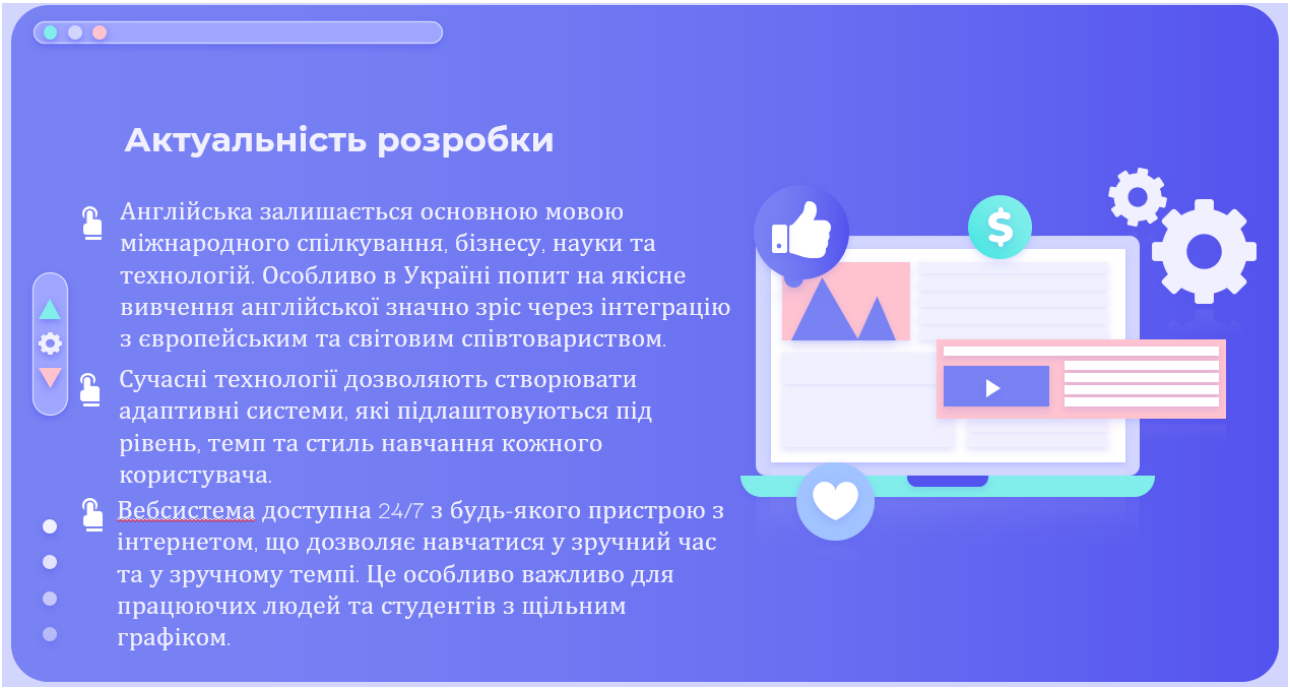


Рисунок Г.3 – Актуальність розробки

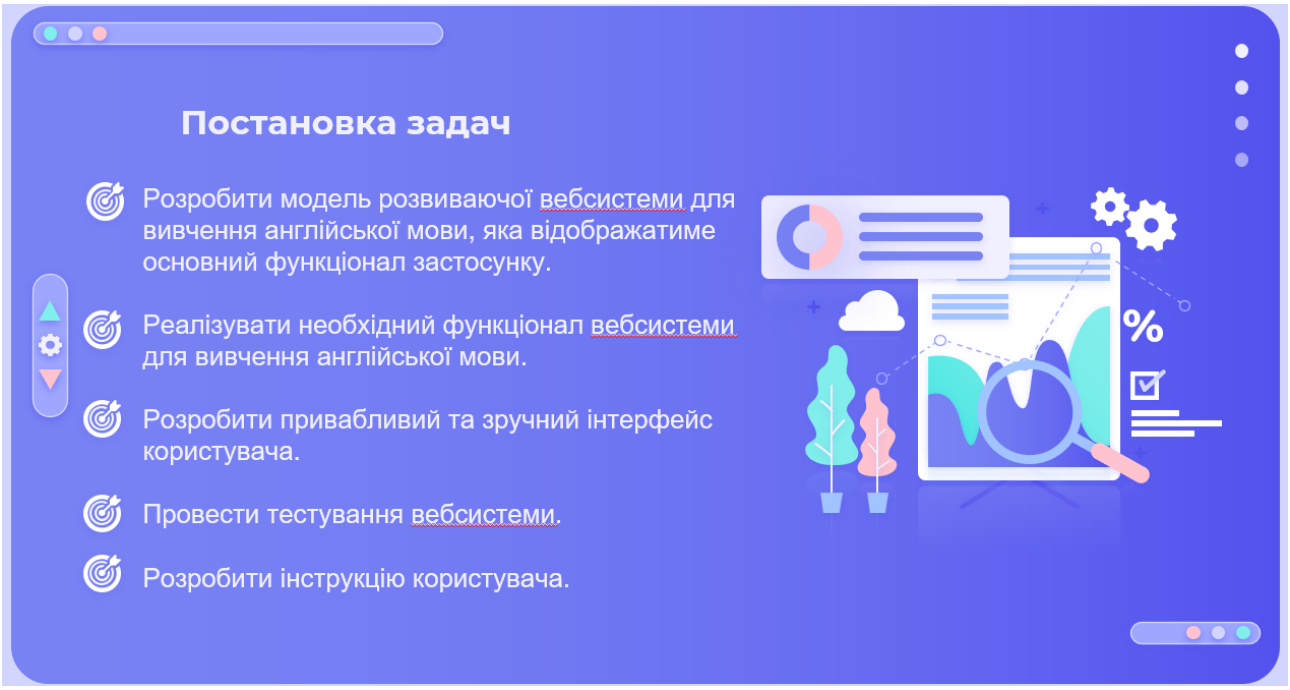


Рисунок Г.4 – Постановка задач

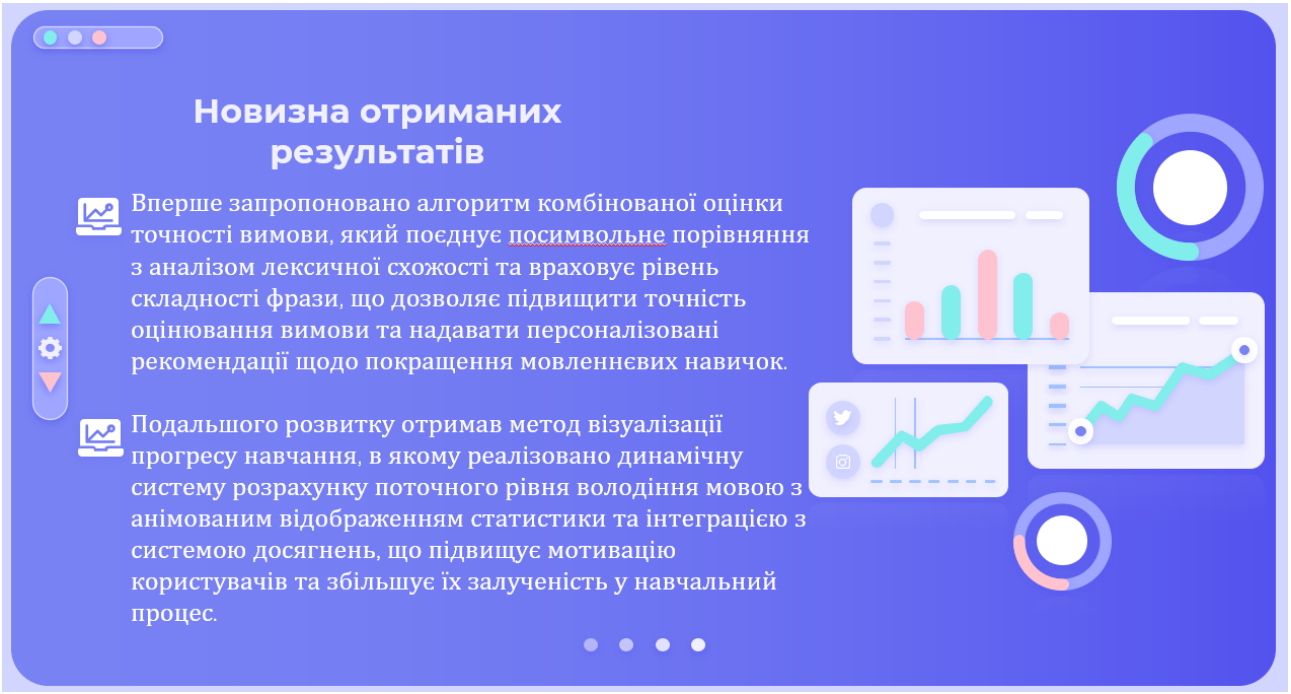


Рисунок Г.5 – Новизна отриманих результатів

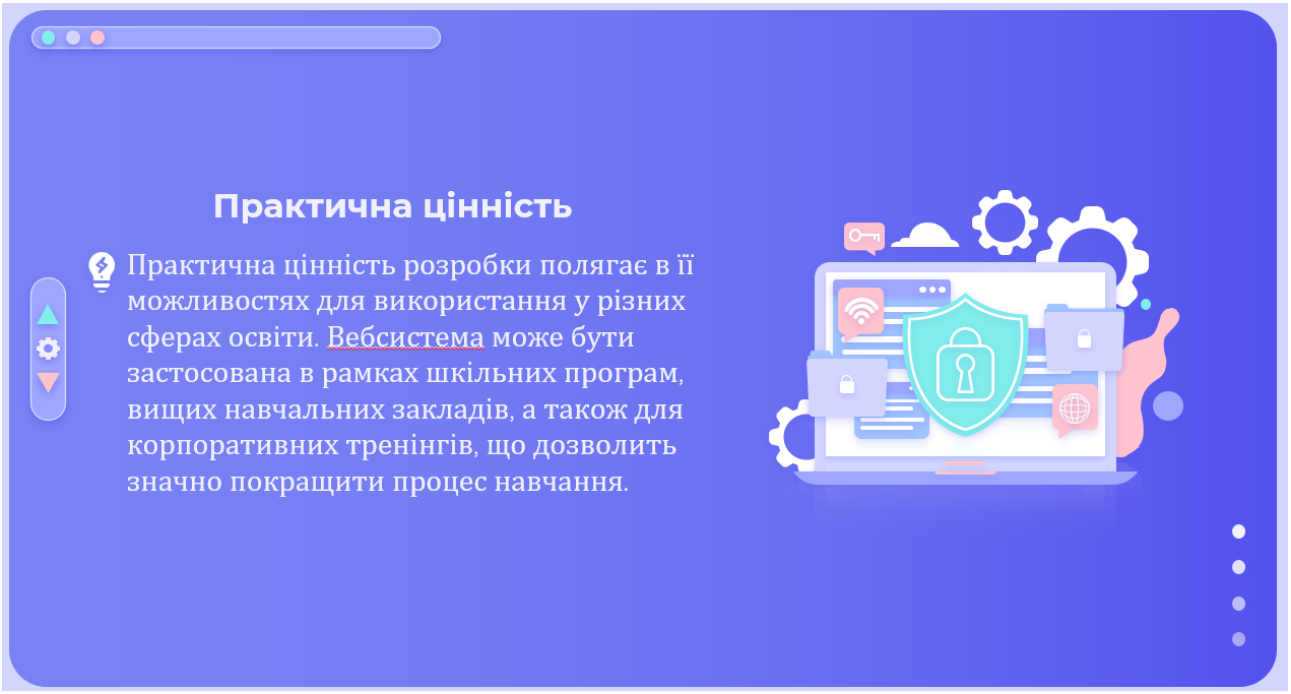


Рисунок Г.6 – Практична цінність

Рисунок Г.7 – Порівняння аналогів

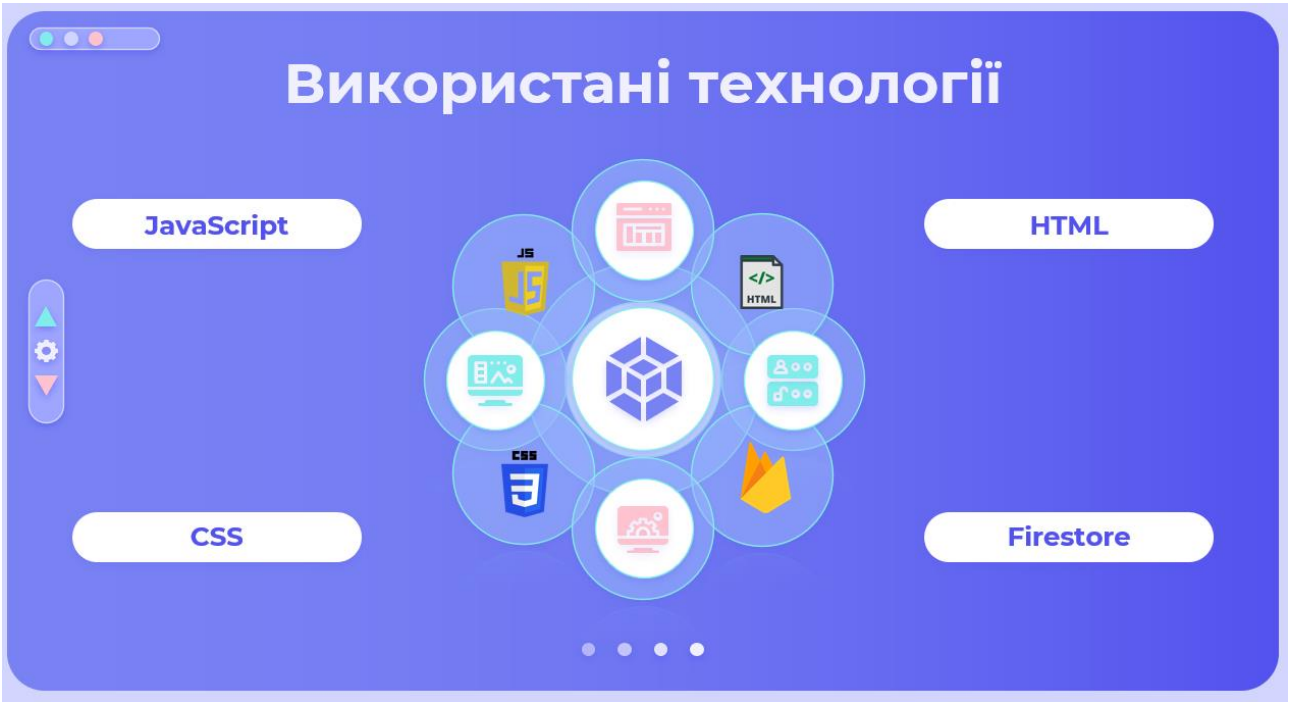


Рисунок Г.8 – Використані технології

Рисунок Г.9 – Модель роботи вебсистеми

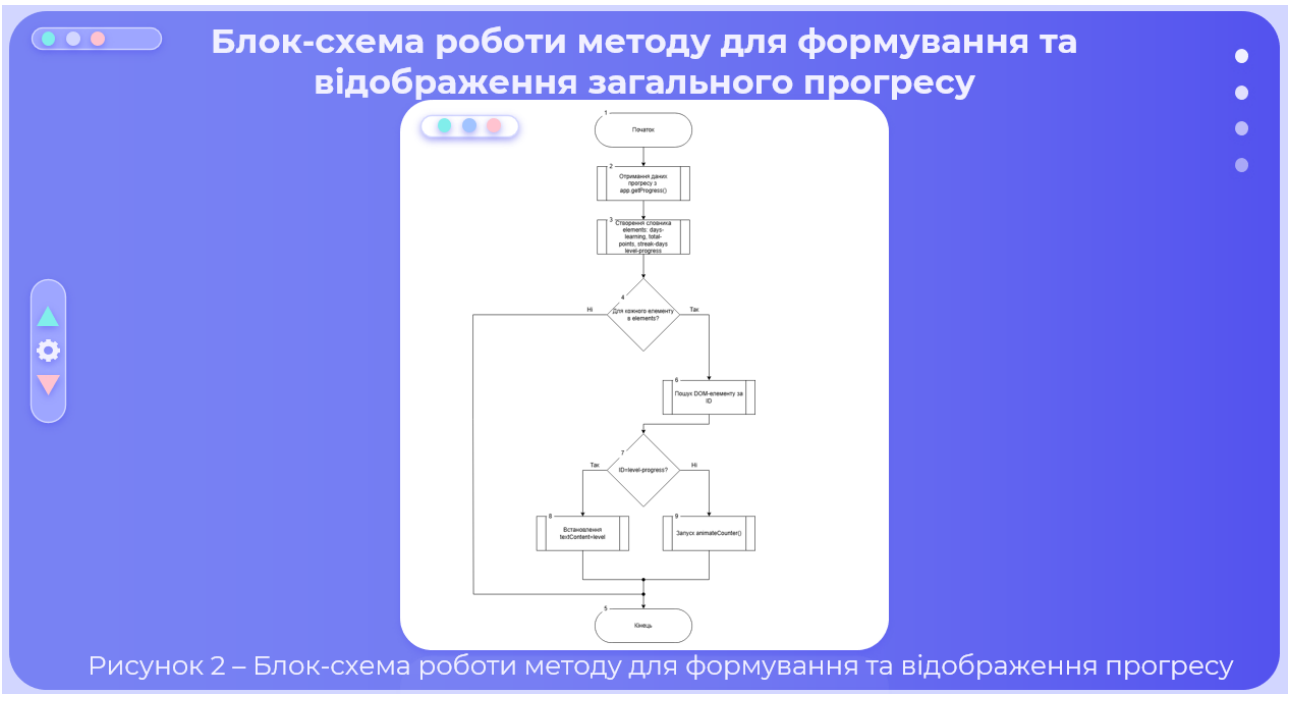


Рисунок Г.10 – Блок-схема методу для формування та відображення прогресу

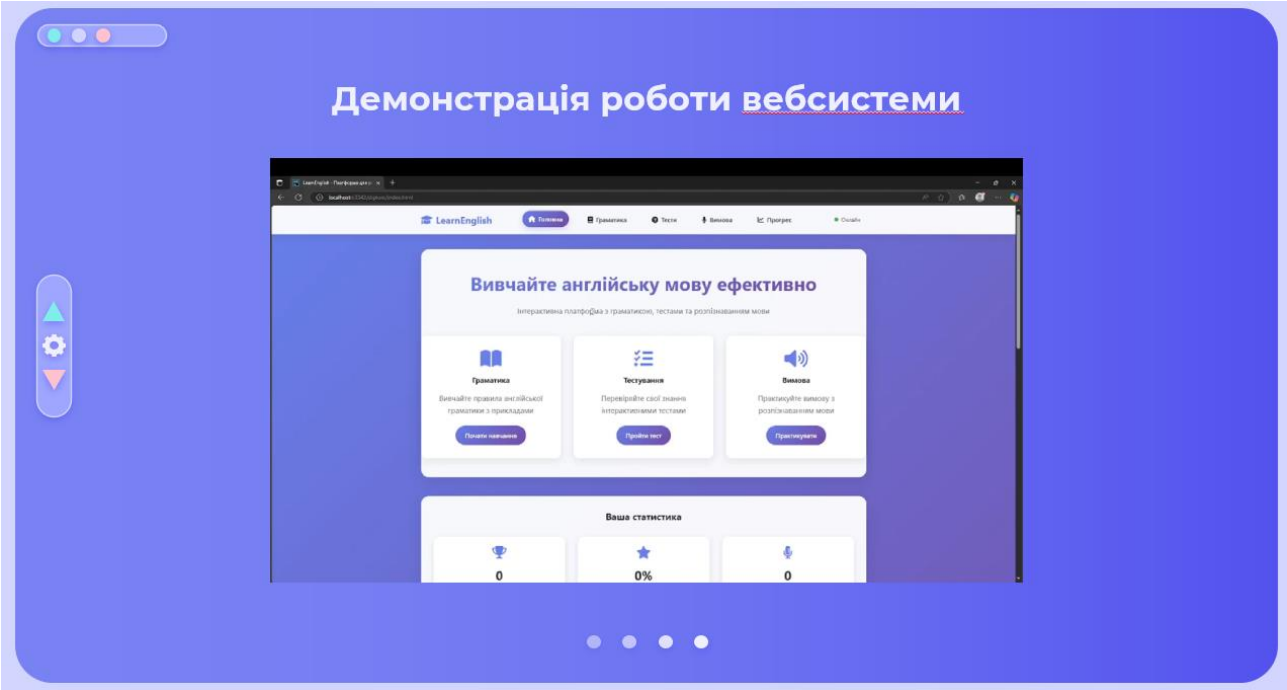


Рисунок Г.11 – Демонстрація роботи вебсистеми

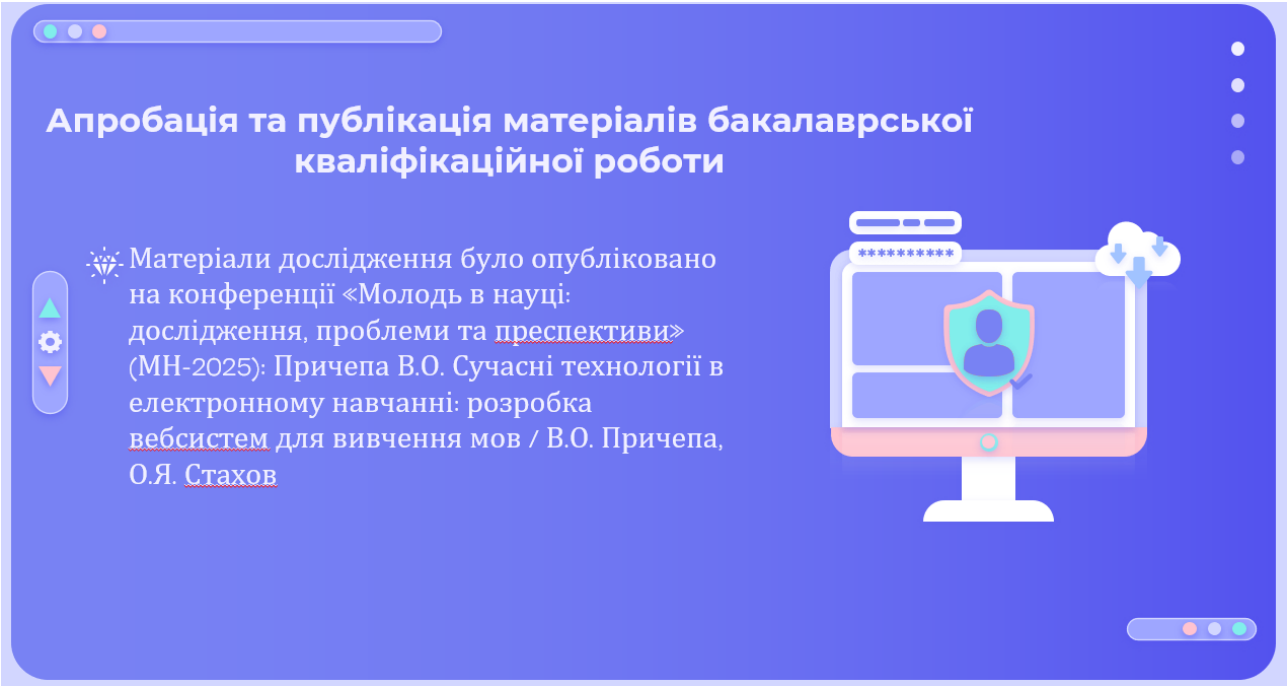


Рисунок Г.12 – Апробація та публікація матеріалів бакалаврської кваліфікаційної роботи

Рисунок Г.13 - Висновки

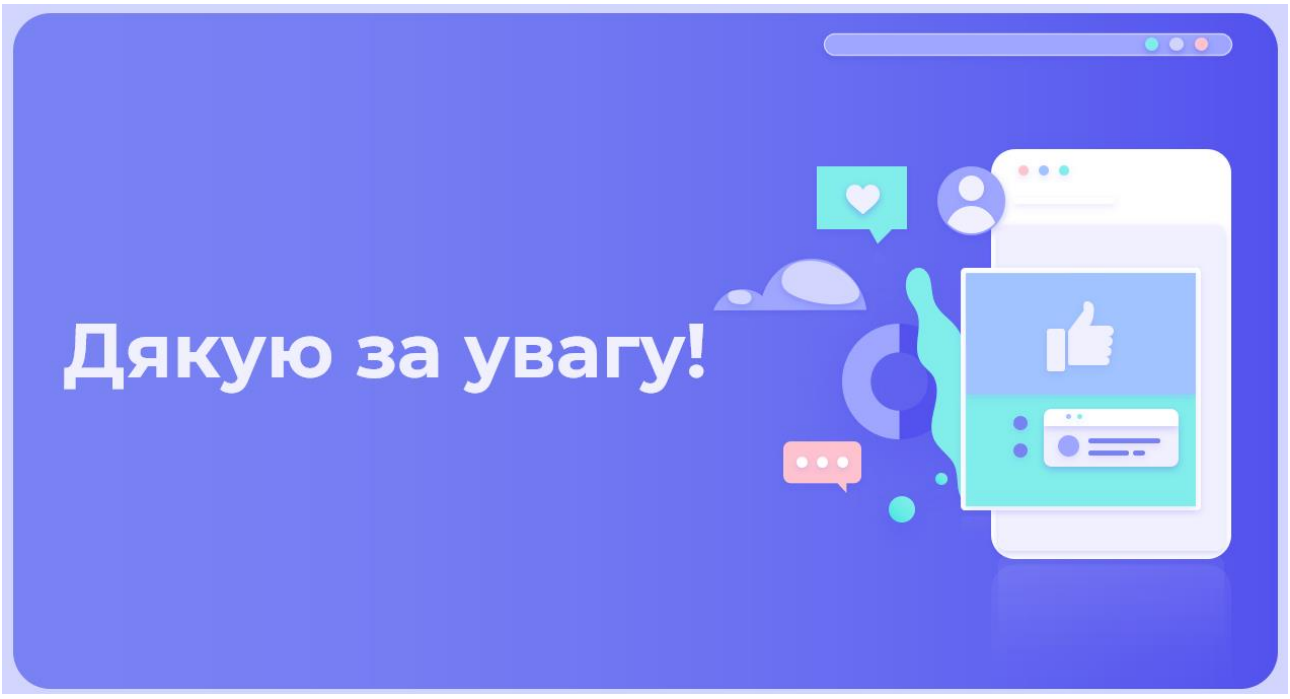


Рисунок Г.14 – Фінальний слайд