ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка навчальної гри “Memory Game” для вивчення англійських слів»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Володимир ЯЦЕНКО
(підпис)

Виконав: здобувач вищої освіти групи ПД-41

Володимир ЯЦЕНКО

Керівник: Юрій ЗАДОНЦЕВ
канд.техн.наук, доц.

Рецензент: _____

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«____ » _____2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Яценко Володимиру Анатолійовичу

1. Тема кваліфікаційної роботи: «Розробка навчальної гри “Memory Game” для вивчення англійських слів».
керівник кваліфікаційної роботи *канд.техн.наук., доц.* Юрій ЗАДОНЦЕВ
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 51.

2. Строк подання кваліфікаційної роботи «29» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про процес вивчення англійської лексики, використання інтерактивних ігрових засобів у навчанні, принципи роботи карткових систем запам’ятовування, підходи до організації адаптивного повторення слів, особливості побудови клієнт-серверних web-застосунків, технічна документація щодо використання технологій Java, Spring Boot, Spring Security, JWT, Spring Data JPA, PostgreSQL, Liquibase, React, Vite, TypeScript, Axios, React Router та REST API.

4. Зміст розрахунково-пояснювальної записки:

1. Огляд та аналіз предметної галузі вивчення англійської лексики за допомогою інтерактивних ігрових засобів.
2. Аналіз існуючих програмних засобів для вивчення англійських слів, роботи з картками та контролю прогресу користувача.
3. Огляд засобів реалізації web-застосунку MemoryGame.
4. Проєктування структури, бази даних та основних модулів системи.
5. Програмна реалізація та опис функціонування web-застосунку.
6. Тестування web-застосунку MemoryGame.
7. Апробація результатів дослідження.
8. Перелік графічного матеріалу: презентація.

5. Вимоги до web-застосунку MemoryGame.

1. Логічна структура системи.
2. Діаграма варіантів використання web-застосунку MemoryGame.
3. Блок-схема процесу проходження гри користувачем.
4. Блок-схема формування адаптивного словникового маршруту.
5. Структура бази даних web-застосунку.
6. Архітектура web-застосунку.
7. Схема обміну даними між клієнтською та серверною частинами через REST API.
8. Діаграма розгортання web-застосунку.
9. Діаграма класів модулів застосунку.
10. Схема та результати тестування програмного забезпечення.
11. Апробація результатів дослідження.

6. Дата видачі завдання«23» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури з теми вивчення англійської лексики та інтерактивних ігрових засобів навчання	23.02–01.03.2026	
2	Аналіз існуючих аналогів програмного забезпечення для вивчення англійських слів, роботи з картками та контролю прогресу	02.03–08.03.2026	
3	Аналіз предметної галузі вивчення англійської лексики та визначення вимог до web-застосунку MemoryGame	09.03–16.03.2026	

4	Огляд засобів реалізації web-застосунку MemoryGame	17.03–23.03.2026	
5	Проєктування web-застосунку MemoryGame, структури системи, бази даних та UML-діаграм	24.03–06.04.2026	
6	Програмна реалізація серверної та клієнтської частин web-застосунку	07.04–27.04.2026	
7	Реалізація адаптивного словникового маршруту, модуля прогресу та адміністративної панелі	21.04–27.04.2026	
8	Тестування web-застосунку та оформлення результатів тестування	28.04–04.05.2026	
9	Оформлення роботи: вступ, висновки, реферат, список джерел та додатки	05.05–11.05.2026	
10	Розробка демонстраційних матеріалів	12.05–17.05.2026	
11	Попередній захист роботи	18.05–22.05.2026	

Здобувач вищої освіти _____ Володимир ЯЦЕНКО
(підпис)

Керівник
кваліфікаційної роботи _____ Юрій ЗАДОНЦЕВ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 53 стор., 3 табл., 23 рис., 24 джерел.

Мета роботи – спрощення вивчення англійської лексики за рахунок розробки веб-застосунку Memory Game з інтерактивними картками, перевіркою відповідей і фіксацією результатів.

Об’єкт дослідження – процес вивчення англійської лексики за допомогою інтерактивних ігрових засобів.

Предмет дослідження – програмне забезпечення для вивчення англійської лексики у форматі інтерактивної гри Memory Game.

У роботі проаналізовано предметну галузь вивчення англійської лексики та визначено основні проблеми, пов’язані з механічним заучуванням слів, недостатньою залученістю користувача, відсутністю регулярної перевірки засвоєння матеріалу та складністю відстеження навчального прогресу. Встановлено, що інтерактивний ігровий формат дозволяє зробити процес запам’ятовування слів більш зручним, наочним і мотивуючим для користувача.

Проаналізовано існуючі програмні засоби для вивчення англійської лексики: Duolingo, Quizlet, Memrise, Wordwall та Anki. Визначено їх переваги, недоліки та функціональні обмеження. Особливу увагу приділено наявності карткової моделі, ігрової механіки, контролю прогресу, можливості керування словниковими картками, адаптивного повторення слабких слів та адміністративного керування навчальним контентом.

Розроблено структуру web-застосунку MemoryGame та визначено його ключові функціональні можливості: реєстрацію та авторизацію користувачів, вибір рівня складності, запуск гри, відкриття карток, пошук пар між англійським словом і українським перекладом, автоматичну перевірку відповідей, збереження

результатів гри, перегляд навчального прогресу, адаптивне тренування, редагування профілю користувача та адміністрування словникової бази.

У роботі використано Java та Spring Boot для реалізації серверної частини, Spring Security і JWT для авторизації та розмежування доступу, Spring Data JPA і Hibernate для роботи з базою даних, PostgreSQL для зберігання даних, Liquibase для керування міграціями бази даних, React, Vite, TypeScript, Axios і React Router для створення клієнтського інтерфейсу, а також REST API для організації взаємодії між клієнтською та серверною частинами.

Сферою використання застосунку є індивідуальне вивчення англійської лексики, навчальні проєкти, мовні курси, освітні платформи та інші середовища, де потрібен зручний інструмент для запам’ятовування слів, контролю результатів і персоналізованого повторення складної лексики.

КЛЮЧОВІ СЛОВА: MEMORYGAME, WEB-ЗАСТОСУНОК, АНГЛІЙСЬКА ЛЕКСИКА, НАВЧАЛЬНА ГРА, КАРТКИ, АДАПТИВНЕ ТРЕНУВАННЯ, ПРОГРЕС КОРИСТУВАЧА, JAVA, SPRING BOOT, SPRING SECURITY, JWT, POSTGRESQL, LIQUIBASE, REACT, TYPESCRIPT, REST API.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – інтерфейс прикладного програмування, що використовується для взаємодії між програмними компонентами системи.

REST – архітектурний стиль організації обміну даними між клієнтською та серверною частинами web-застосунку.

HTTP – протокол передавання даних у мережі Інтернет.

HTTPS – захищена версія протоколу HTTP, що забезпечує безпечний обмін даними між клієнтом і сервером.

JSON – текстовий формат обміну даними, який використовується для передавання інформації між frontend- і backend-частинами системи.

UI – користувацький інтерфейс програмного забезпечення.

UX – користувацький досвід під час взаємодії з програмним продуктом.

HTML – мова розмітки гіпертексту, що використовується для створення структури web-сторінок.

CSS – мова стилів, що використовується для оформлення зовнішнього вигляду web-сторінок.

JavaScript – мова програмування, що використовується для створення інтерактивної поведінки web-сторінок.

TypeScript – мова програмування, що розширює JavaScript за рахунок статичної типізації.

React – бібліотека JavaScript для створення компонентного користувацького інтерфейсу.

Vite – інструмент для швидкого створення, запуску та збирання frontend-застосунків.

Axios – бібліотека для виконання HTTP-запитів із клієнтської частини web-застосунку.

React Router – бібліотека для організації маршрутизації між сторінками React-застосунку.

Java – об’єктно-орієнтована мова програмування, що використовується для реалізації серверної частини застосунку.

Spring – фреймворк для створення серверних Java-застосунків.

Spring Boot – інструмент екосистеми Spring, що спрощує створення та запуск серверних web-застосунків.

Spring Security – модуль Spring для реалізації автентифікації, авторизації та захисту доступу до ресурсів системи.

JWT – формат токена, що використовується для безпечної передачі даних авторизації між клієнтом і сервером.

JPA – специфікація Java для роботи з реляційними базами даних через об’єктну модель.

Hibernate – фреймворк об’єктно-реляційного відображення, який забезпечує зв’язок між Java-класами та таблицями бази даних.

PostgreSQL – об’єктно-реляційна система керування базами даних.

ORM – технологія об’єктно-реляційного відображення, що забезпечує зв’язок між об’єктами програми та таблицями бази даних.

CRUD – базові операції роботи з даними: створення, читання, оновлення та видалення.

Liquibase – інструмент для керування змінами структури бази даних за допомогою міграцій.

SPA – односторінковий web-застосунок, у якому основна взаємодія з користувачем відбувається без повного перезавантаження сторінки.

IDE – інтегроване середовище розробки програмного забезпечення.

Git – система контролю версій, що використовується для відстеження змін у програмному коді.

Memory Game – ігровий формат, у якому користувач відкриває картки та шукає правильні пари між словами й перекладами.

ВСТУП

У сучасних умовах розвитку інформаційних технологій та цифровізації освітнього процесу все більшого значення набувають інтерактивні засоби навчання. Вивчення іноземних мов, зокрема англійської, потребує не лише засвоєння граматичних правил, а й регулярного поповнення словникового запасу. Саме лексика є основою для розуміння текстів, побудови речень, усного мовлення та сприйняття інформації на слух. Проте традиційне механічне заучування слів часто є недостатньо ефективним, оскільки воно не завжди передбачає активну взаємодію користувача з навчальним матеріалом, регулярне повторення та контроль засвоєння [1; 4].

Однією з поширених проблем у процесі вивчення англійської лексики є швидке забування нових слів через відсутність системного повторення. Користувач може переглядати списки слів або навчальні матеріали, але без перевірки результатів, фіксації помилок і повернення до складних слів ефективність такого навчання знижується. Крім того, звичайне заучування часто є монотонним, що негативно впливає на мотивацію користувача та зменшує регулярність занять [2; 3].

У зв’язку з цим актуальним є використання інтерактивних ігрових засобів, які дозволяють зробити процес вивчення лексики більш наочним, динамічним і зручним. Ігрові механіки сприяють залученню користувача до навчального процесу, створюють додаткову мотивацію до повторення матеріалу та дозволяють поєднати навчання з елементами змагання, досягнень і поступового ускладнення завдань. Особливо ефективним для вивчення слів є картковий підхід, коли користувач працює з окремими навчальними одиницями: словом, перекладом, підказкою або прикладом використання [1; 3].

Формат Memory Game є одним із зручних варіантів реалізації такого підходу. У межах цієї гри користувач відкриває картки та шукає відповідні пари між англійським словом і українським перекладом. Така взаємодія дозволяє не лише бачити слово, а й активно працювати з ним у процесі гри. Завдяки цьому

користувач тренує пам’ять, повторює лексику, отримує результат після проходження рівня та може повторно працювати зі словами, які були засвоєні гірше [2; 3].

Актуальність теми кваліфікаційної роботи зумовлена необхідністю створення зручного web-застосунку для вивчення англійської лексики, який поєднує ігрову взаємодію, роботу з картками, фіксацію результатів і персоналізоване повторення слабких слів. На відміну від звичайних списків слів або загальних мовних платформ, розроблюваний застосунок MemoryGame орієнтований саме на інтерактивне запам’ятовування лексики через гру та подальший аналіз результатів користувача [1; 2; 4].

Web-застосунок MemoryGame передбачає роботу двох основних ролей: USER та ADMIN. Звичайний користувач може зареєструватися, увійти до системи, обрати рівень складності, пройти гру, переглянути результати, відстежити прогрес, скористатися адаптивним тренуванням і редагувати профіль. Адміністратор має можливість керувати словниковими картками, рівнями складності, користувачами, активністю навчальних матеріалів та переглядати статистику платформи. Такий розподіл ролей дозволяє відокремити навчальну діяльність користувача від процесу адміністрування системи.

Особливою функціональною можливістю застосунку є адаптивне тренування. Система зберігає статистику по кожному слову: кількість спроб, кількість помилок, кількість успішних повторень і рівень засвоєння. На основі цих даних формується персональний набір карток для повторення. Завдяки цьому користувач більше часу працює саме з тими словами, які викликали труднощі, а не повторює весь словник випадковим чином. Це дозволяє зробити навчання більш індивідуалізованим і результативним [2; 3].

Мета роботи – спрощення вивчення англійської лексики за рахунок розробки web-застосунку MemoryGame з інтерактивними картками, перевіркою відповідей, фіксацією результатів та формуванням адаптивного словникового маршруту.

Об’єкт дослідження – процес вивчення англійської лексики за допомогою інтерактивних ігрових засобів [1; 3].

Предмет дослідження – засоби, методи та технології реалізації web-застосунку MemoryGame для вивчення англійських слів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Проаналізувати предметну галузь вивчення англійської лексики за допомогою інтерактивних ігрових засобів [1; 3].

2. Дослідити існуючі програмні засоби для вивчення англійських слів, роботи з картками та контролю прогресу користувача.

3. Визначити функціональні та нефункціональні вимоги до web-застосунку MemoryGame.

4. Обґрунтувати вибір засобів реалізації клієнтської частини, серверної частини та бази даних.

5. Спроєктувати логічну структуру системи, базу даних та UML-діаграми web-застосунку MemoryGame [24].

6. Реалізувати основні модулі системи: гру, профіль користувача, прогрес, адаптивне тренування та адміністративну панель [2; 3].

7. Розробити механізм адаптивного словникового маршруту на основі помилок і результатів користувача.

8. Провести тестування web-застосунку MemoryGame та проаналізувати результати перевірки основних функцій.

Методи дослідження. У бакалаврській роботі використовуються такі методи: аналіз предметної галузі, порівняльний аналіз існуючих програмних засобів для вивчення англійської лексики, об’єктно-орієнтоване проєктування, UML-моделювання, проєктування структури бази даних, розробка клієнт-серверного web-застосунку та тестування програмного забезпечення [24].

Наукова новизна одержаних результатів полягає у розробці структури web-застосунку MemoryGame, який поєднує ігрове вивчення англійської лексики, карткову модель навчання, збереження персонального прогресу користувача та адаптивне повторення слів на основі попередніх помилок і рівня засвоєння.

Практична значущість одержаних результатів полягає в тому, що розроблений web-застосунок може бути використаний для індивідуального вивчення англійської лексики, у навчальних проєктах, мовних курсах або освітніх платформах. У майбутньому систему можна розширити додаванням нових мов, тематичних словників, рейтингової системи, досягнень, аудіовимови слів, статистичних звітів та інтеграції з іншими освітніми сервісами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Загальна характеристика предметної галузі

Вивчення англійської лексики є однією з основних складових процесу опанування іноземної мови. Саме словниковий запас визначає можливість користувача розуміти текстову інформацію, сприймати мовлення, формувати власні речення та використовувати мову в навчальних, професійних або побутових ситуаціях. При цьому засвоєння нових слів потребує не лише одноразового ознайомлення з ними, а й регулярного повторення, перевірки правильності запам’ятовування та поступового закріплення матеріалу.

Традиційний підхід до вивчення лексики часто полягає у перегляді списків слів, записуванні перекладів або механічному повторенні матеріалу. Такий спосіб може бути корисним на початковому етапі, однак він не завжди забезпечує достатній рівень залученості користувача. Без інтерактивної взаємодії, візуального представлення матеріалу та контролю результатів користувач швидко втрачає мотивацію, а нові слова можуть забуватися вже через короткий проміжок часу.

У сучасних умовах для вирішення цієї проблеми активно використовуються цифрові освітні засоби: мобільні застосунки, web-платформи, електронні картки, тренажери, тести та навчальні ігри. Такі рішення дозволяють організувати процес навчання більш гнучко, оскільки користувач може самостійно обирати темп роботи, рівень складності, кількість повторень і спосіб перевірки знань [4].

Особливе значення у цій предметній галузі мають ігрові підходи до навчання. Вони дають змогу перетворити процес запам’ятовування слів із монотонного повторення на активну взаємодію з навчальним матеріалом. У межах ігрового підходу користувач не просто переглядає слово та переклад, а виконує певні дії: відкриває картки, шукає правильні пари, отримує результат, бачить кількість помилок і може повторно пройти рівень, процес роботи зображено на рисунку 1.1.

Одним із таких підходів є гра типу Memory Game. Її основна ідея полягає у тому, що користувач відкриває картки та повинен знайти відповідні пари. У контексті вивчення англійської лексики пара може складатися з англійського слова та його українського перекладу. Такий формат поєднує навчання, тренування пам’яті та ігрову взаємодію, що робить процес засвоєння лексики більш наочним і зручним.

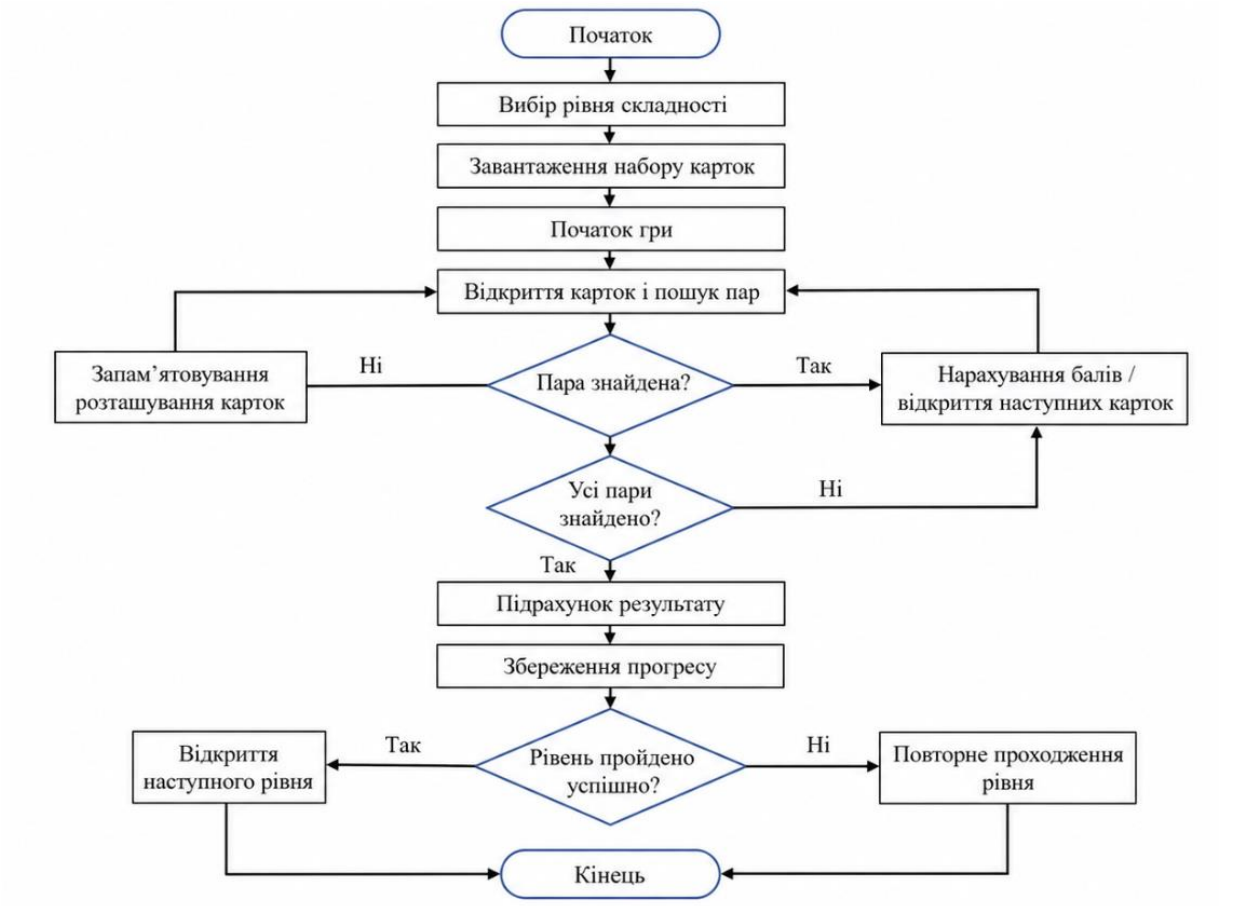


Рис. 1.1 Загальна схема процесу вивчення англійської лексики за допомогою гри Memory Game

1.2 Актуальність створення web-застосунку для вивчення англійської лексики

Актуальність створення web-застосунку для вивчення англійської лексики зумовлена необхідністю підвищення зручності, регулярності та результативності

навчального процесу. Більшість користувачів, які самостійно вивчають англійську мову, стикаються з проблемою нерегулярного повторення слів, відсутності контролю помилок і складністю відстеження власного прогресу.

Якщо користувач вивчає слова без системи фіксації результатів, він не завжди розуміє, які слова вже засвоєні добре, а які потребують додаткового повторення. Через це навчання може бути менш ефективним, оскільки користувач витрачає однакову кількість часу як на прості, так і на складні для себе слова. У результаті частина лексики повторюється надмірно, а частина, навпаки, не закріплюється достатньо.

Web-застосунок MemoryGame дозволяє вирішити цю проблему за рахунок збереження результатів проходження гри, обліку помилок, кількості спроб, успішних повторень і рівня засвоєння окремих слів. Завдяки цьому система може не лише показувати загальний результат користувача, а й формувати персоналізований навчальний маршрут.

Важливою перевагою web-застосунку є те, що користувачу не потрібно встановлювати окрему програму на пристрій. Доступ до системи може здійснюватися через браузер, що робить застосунок зручним для використання на різних пристроях. Крім того, web-формат дозволяє централізовано зберігати словникову базу, результати, профілі користувачів і статистику проходження рівнів.

Для адміністратора така система також є корисною, оскільки вона дає змогу керувати навчальним контентом: додавати нові слова, редагувати картки, змінювати рівні складності, активувати або деактивувати навчальні матеріали, переглядати статистику платформи та контролювати користувачів. Таким чином, web-застосунок MemoryGame може використовуватися не лише окремими користувачами, а й у навчальних проєктах або мовних курсах.

1.3 Цільова аудиторія та основні користувачі системи

Цільовою аудиторією web-застосунку MemoryGame є користувачі, які вивчають англійську лексику та потребують простого інтерактивного інструменту для запам’ятовування слів. До такої аудиторії можна віднести учнів, студентів, користувачів, які самостійно вивчають англійську мову, а також викладачів або адміністраторів навчальних платформ, які хочуть організувати роботу зі словниковими картками.

У системі передбачено дві основні ролі: USER та ADMIN.

Користувач з роллю USER працює з навчальним функціоналом системи. Він може зареєструватися, увійти до системи, обрати рівень складності, запустити гру, відкривати картки, шукати правильні пари, переглядати результати, аналізувати прогрес, проходити адаптивне тренування та редагувати власний профіль [2; 3].

Користувач з роллю ADMIN відповідає за керування системою та навчальним контентом. Адміністратор може переглядати статистику платформи, додавати нові слова, редагувати картки, активувати або деактивувати їх, керувати рівнями складності, переглядати користувачів, змінювати ролі, блокувати або активувати акаунти, переглянути на рисунку 1.2.

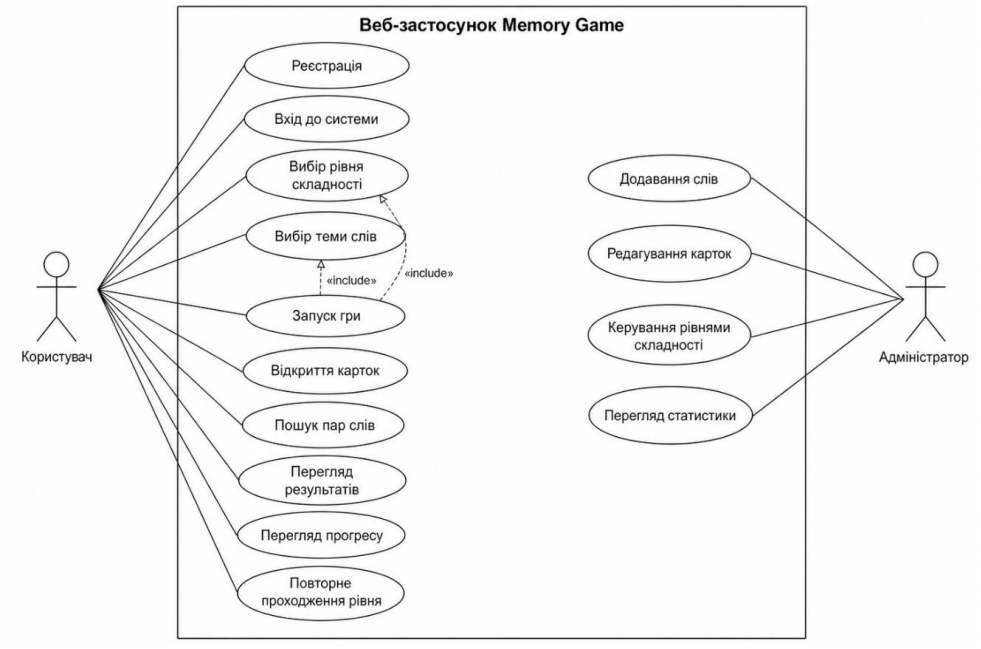


Рис. 1.2 Діаграма варіантів використання web-застосунку MemoryGame

Такий розподіл ролей дозволяє відокремити навчальну взаємодію від адміністративних функцій. Звичайний користувач отримує простий інтерфейс для навчання, а адміністратор - інструменти для підтримки словникової бази та контролю роботи системи.

1.4 Основні функції та бізнес-логіка системи

Основна бізнес-логіка web-застосунку MemoryGame пов’язана з організацією навчального процесу через ігрову взаємодію з картками. Користувач входить до системи, обирає рівень складності та запускає гру. Після цього система формує набір карток, серед яких є англійські слова та їх українські переклади. Користувач відкриває картки та намагається знайти правильні пари.

Під час проходження гри система фіксує кількість ходів, помилки, знайдені пари, час проходження та підсумковий результат. Після завершення гри ці дані зберігаються в базі даних. Надалі вони використовуються для відображення статистики користувача та формування адаптивного тренування, процес зображено на рисунку 1.3.

Окремою функціональною частиною є адаптивне тренування. Його логіка полягає у тому, що система аналізує результати користувача по кожному слову: кількість спроб, кількість помилок, кількість успішних повторень і рівень засвоєння. На основі цих показників формується набір слів, які користувачу потрібно повторити. Це дозволяє не проходити весь словник випадково, а зосередитися саме на тих словах, які були засвоєні гірше [2; 3].

До основних функцій системи належать:

– реєстрація та авторизація користувача;

– вибір рівня складності; запуск ігрового режиму;

– відкриття карток і пошук пар;

– перевірка правильності вибраної пари;

– підрахунок результату гри;

– збереження результатівперегляд прогресу користувача;

– формування адаптивного тренування;

– редагування профілю;

– керування словниковими картками;

– адміністрування користувачів і рівнів складності перегляд статистики системи.

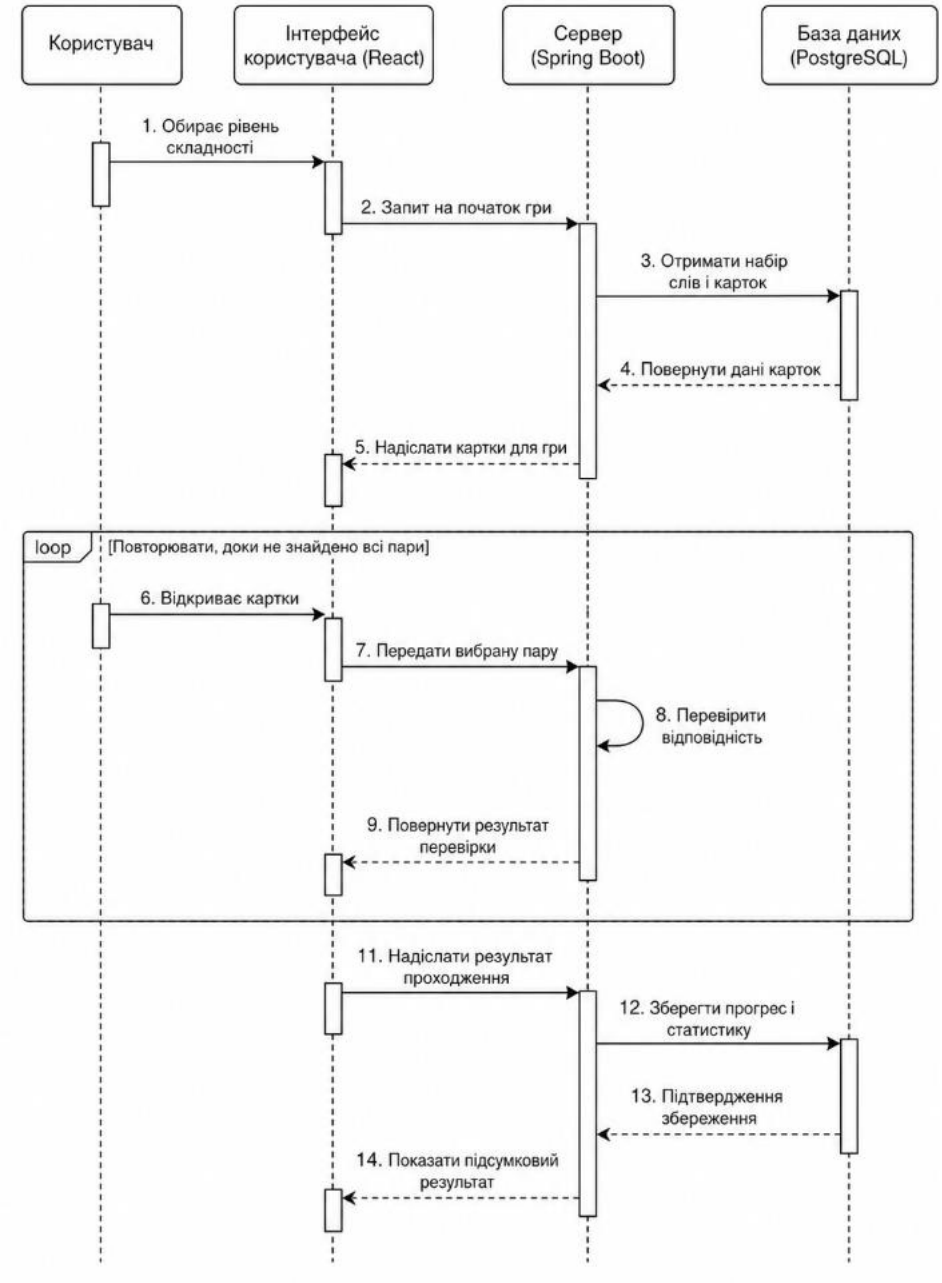


Рис. 1.3 Блок-схема процесу проходження гри користувачем

1.5 Способи вирішення задачі та роль інфокомунікаційних засобів

Задача вивчення англійської лексики може вирішуватися різними способами. Найпростішим варіантом є використання паперових словників, списків слів або звичайних конспектів. Такий підхід не потребує спеціального програмного забезпечення, однак має суттєві обмеження: відсутність автоматичної перевірки, складність відстеження прогресу, неможливість швидко визначити слабкі слова та низький рівень інтерактивності.

Іншим способом є використання електронних таблиць або текстових файлів зі словами та перекладами. Такий варіант дозволяє структурувати матеріал краще, ніж паперові записи, однак не забезпечує повноцінної ігрової взаємодії, автоматичного підрахунку результатів і персоналізованого повторення.

Більш сучасним рішенням є використання спеціалізованих цифрових платформ, мобільних застосунків і web-сервісів. Вони можуть містити картки, тести, вправи на відповідність, рейтинги, статистику та інші елементи, переглянути процес на рисунку 1.4. Проте більшість готових рішень або є надто загальними, або не орієнтовані саме на формат Memory Game, або мають обмежені можливості керування навчальним контентом.

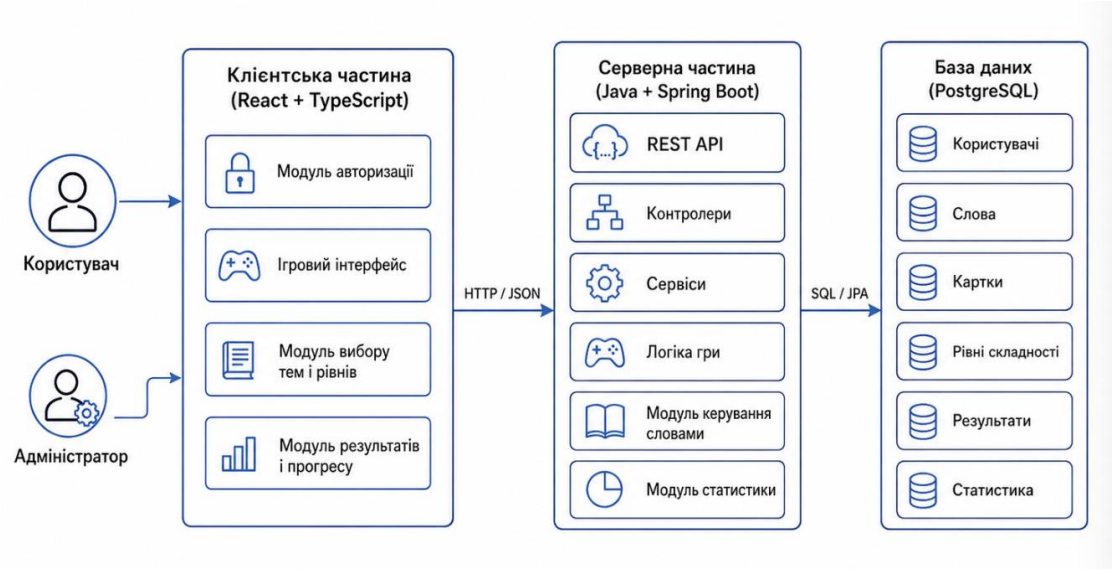


Рис. 1.4 Схема обміну даними між клієнтською та серверною частиною через REST API

Розробка власного web-застосунку MemoryGame дозволяє поєднати переваги цифрового навчального середовища з конкретною ігровою механікою. За рахунок клієнт-серверної архітектури система може зберігати дані користувачів, результати ігор, словникові картки та прогрес у централізованій базі даних. Використання REST API забезпечує обмін даними між клієнтською та серверною частинами, а web-інтерфейс робить систему доступною через браузер [22; 23].

1.6 Основні вимоги до програмного продукту

На основі аналізу предметної галузі можна визначити основні вимоги до майбутнього web-застосунку MemoryGame. Система повинна бути зрозумілою для користувача, забезпечувати швидкий запуск гри, підтримувати вибір рівня складності, правильно перевіряти пари карток і зберігати результати проходження.

Функціональні вимоги до системи включають можливість реєстрації та авторизації користувача, запуску гри, вибору складності, проходження рівнів, перегляду результатів, формування адаптивного тренування та редагування профілю. Для адміністратора мають бути реалізовані функції керування словниковою базою, рівнями складності, користувачами та статистикою.

Нефункціональні вимоги пов’язані зі стабільністю, швидкодією, зручністю інтерфейсу та можливістю розширення системи. Основні сторінки мають завантажуватися не більше ніж за 3 секунди, відповідь сервера на запит має займати не більше 1 секунди, а інтерфейс повинен коректно працювати на екранах від 360 px до 1920 px. Також важливо забезпечити підтримку основних браузерів, збереження результатів після кожної завершеної гри та можливість розширення словникової бази щонайменше до 1000 навчальних одиниць.

Особливу увагу слід приділити захисту даних користувачів. Для цього в системі передбачається використання авторизації, розмежування ролей USER і ADMIN, а також збереження паролів у захищеному вигляді. Адміністративні функції мають бути доступні лише користувачам із відповідною роллю.

1.7 Аналіз проблем і недоліків існуючих підходів

Існуючі підходи до вивчення англійської лексики мають низку обмежень. Механічне заучування списків слів не забезпечує достатнього рівня взаємодії користувача з матеріалом. Користувач може запам’ятати слово на короткий час, але без регулярного повторення та практичного використання воно швидко забувається.

Готові навчальні платформи часто мають широкий функціонал, однак не завжди відповідають конкретній задачі простого ігрового запам’ятовування слів. Наприклад, частина сервісів орієнтована на повноцінне вивчення мови, інші — на створення карток або тестів, але не завжди мають механіку Memory Game, адаптивне повторення слабких слів або адміністративне керування навчальним контентом.

Також проблемою є те, що в багатьох системах користувач не бачить детального прогресу по кожному слову. Загальна статистика може показувати кількість пройдених вправ або набрані бали, але не завжди дає відповідь на питання, які саме слова користувач засвоїв гірше. Через це навчання може залишатися недостатньо персоналізованим.

У web-застосунку MemoryGame ці недоліки враховуються за рахунок збереження результатів гри, деталізації статистики по картках, формування user_word_progress та адаптивного тренування. Система дозволяє не лише проходити гру, а й аналізувати навчальний результат, повторювати складні слова та поступово покращувати рівень засвоєння лексики.

Отже, аналіз предметної галузі показав, що розробка web-застосунку MemoryGame є доцільною, оскільки система поєднує інтерактивну гру, карткову модель навчання, контроль результатів, персоналізоване повторення та адміністративне керування словниковою базою. Це дозволяє зробити процес вивчення англійської лексики більш зручним, структурованим і результативним.

2 АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ ЛЕКСИКИ

У межах розробки інтерактивної ігрової платформи для вивчення англійської лексики доцільно проаналізувати вже існуючі програмні засоби, які частково або повністю вирішують подібні задачі. До таких засобів можна віднести мобільні застосунки для вивчення мов, онлайн-платформи з картками, сервіси для створення навчальних ігор та системи для повторення слів.

Основна задача таких програмних продуктів полягає в тому, щоб допомогти користувачу вивчати, повторювати та закріплювати нову лексику. Однак з точки зору розробника програмного забезпечення важливо оцінювати не лише зовнішній функціонал, а й логіку побудови системи: структуру інтерфейсу, організацію навчального процесу, способи зберігання результатів, масштабованість, адаптивність, наявність ролей користувачів, можливість розширення функціоналу та зручність підтримки системи.

Для аналізу було обрано такі програмні засоби:

– Duolingo

– Quizlet

– Memrise

– Wordwall

– Anki.

2.1 Duolingo

Duolingo є одним із найпоширеніших застосунків для вивчення іноземних мов, переглянути на рисунку 2.1. Його основна ідея полягає у проходженні коротких уроків, які поєднують вправи на читання, аудіювання, письмо, переклад і запам’ятовування лексики. Платформа активно використовує елементи гейміфікації: рівні, серії занять, нагороди, рейтинги, візуальне відображення

прогресу та систему мотивації користувача. Офіційний опис Duolingo вказує, що сервіс пропонує короткі уроки для практики говоріння, читання, слухання й письма та розвитку лексичних і граматичних навичок [5].

З точки зору предметної галузі Duolingo добре підходить для аналізу, оскільки він демонструє, як мовне навчання може бути представлене у формі ігрової системи. Користувач не просто отримує список слів, а проходить послідовність завдань, бачить результат, отримує зворотний зв'язок і поступово переходить до складніших рівнів. Такий підхід є корисним для розробки Memory Game, оскільки показує важливість поєднання навчального змісту, мотиваційної механіки та контролю прогресу.

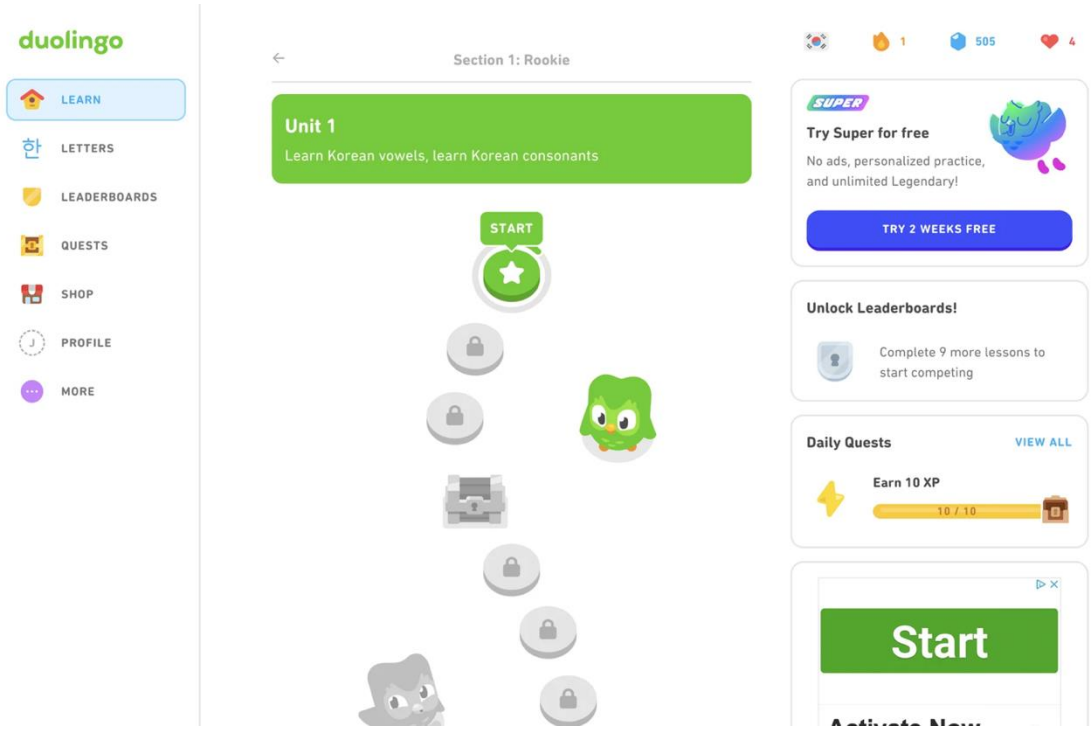


Рис. 2.1 Приклад інтерфейсу застосунку Duolingo

До переваг Duolingo можна віднести: [5]

– зрозумілий та привабливий інтерфейс;

– активне використання ігрових елементів;

– наявність рівнів, досягнень і візуального прогресу;

– короткі навчальні сесії, які не перевантажують користувача;

– підтримку різних типів мовних вправ;

До недоліків Duolingo можна віднести:

– надмірну складність для реалізації в межах бакалаврської роботи;

– обмежені можливості ручного налаштування власних наборів слів;

– орієнтацію на загальний мовний курс, а не на окрему механіку запам’ятовування карток;

– залежність навчальної логіки від внутрішніх алгоритмів платформи;

– частину розширених можливостей, які можуть бути доступні лише в платній версії.

2.2 Quizlet

Quizlet є онлайн-платформою для створення та використання навчальних карток, переглянути на рисунку 2.2. Користувач може створювати власні набори термінів, використовувати готові матеріали, проходити тести, тренування та ігрові режими. Офіційна сторінка Quizlet описує сервіс як платформу з флешкартками, навчальними режимами та класними іграми, зокрема Quizlet Live. Також у описі застосунку зазначено можливість перетворювати картки на інтерактивні ігри, тести та режими навчання [6].

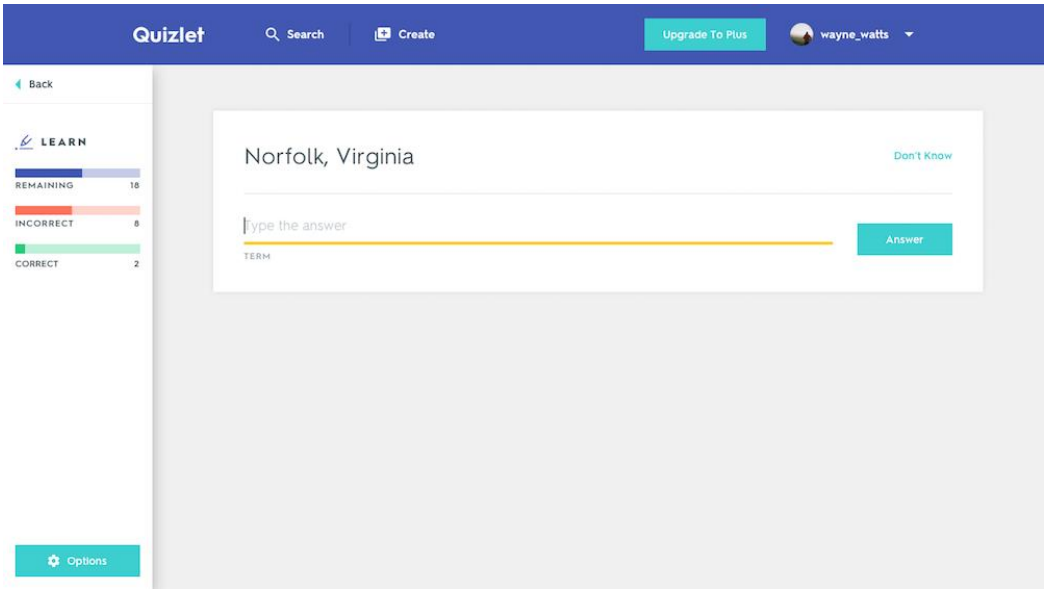


Рис. 2.2 Приклад інтерфейсу платформи Quizlet

Для теми розробки Memory Game Quizlet є одним із найближчих аналогів, оскільки в його основі лежить робота з картками. Карткова модель добре підходить для вивчення англійських слів, тому що дозволяє поєднувати слово, переклад, зображення, аудіо або пояснення. Такий підхід можна використати і в розроблюваному веб-застосунку, де картки будуть основним елементом ігрової взаємодії.

Переваги Quizlet: [6]

– можливість створювати власні набори карток;

– наявність готових навчальних матеріалів;

– підтримка різних режимів вивчення;

– використання тестів та інтерактивних вправ;

– зручність для самостійного навчання;

– можливість застосування в освітньому процесі.

Недоліки Quizlet:

– частина функцій може бути обмежена платною підпискою;

– система не орієнтована виключно на англійську лексику;

– велика кількість готових матеріалів може мати різну якість;

– ігровий компонент не завжди є основним сценарієм взаємодії;

– інтерфейс може бути надлишковим для користувача, якому потрібна проста гра для запам’ятовування слів.

2.3 Memrise

Memrise - це застосунок для вивчення іноземних мов, який поєднує лексику, фрази, відео носіїв мови та повторення матеріалу, переглянути на рисунку 2.3. Офіційний сайт Memrise акцентує увагу на тому, що користувачі навчаються за допомогою відео з реальними носіями мови, а також отримують персоналізований навчальний досвід. Для англійської мови Memrise описує підхід через занурення у мовне середовище та віртуальну практику [7].

Memrise підходить для аналізу тому, що він демонструє інший підхід до вивчення лексики - через контекст, приклади використання та повторення. Якщо Duolingo більше орієнтований на короткі уроки, а Quizlet - на картки, то Memrise намагається зробити запам’ятовування більш природним за рахунок відео, фраз і повторення.

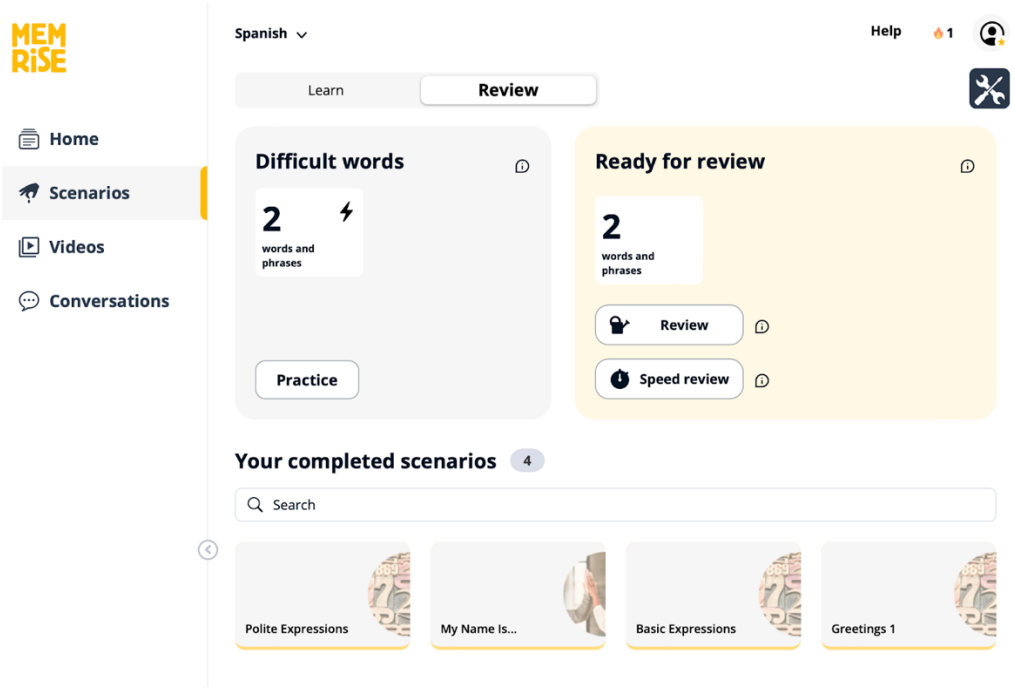


Рис. 2.3 Приклад інтерфейсу застосунку Memrise

Переваги Memrise: [7]

– використання реальних мовних прикладів;

– орієнтація на практичну лексику;

– наявність повторення матеріалу;

– зручний мобільний формат;

– персоналізація навчання;

– поєднання лексики з контекстом.

Недоліки Memrise:

– основний акцент зроблено не на ігровій механіці Memory Game;

– мультимедійний контент ускладнює структуру системи;

– для повної реалізації подібного підходу потрібні великі обсяги якісного навчального матеріалу;

– система більше орієнтована на готові курси, ніж на просту взаємодію з картками;

– частина функцій може бути доступна лише у платній версії.

2.4 Wordwall

Wordwall — це онлайн-платформа для створення інтерактивних навчальних вправ, переглянути на рисунку 2.4. Вона дозволяє створювати різні типи активностей: вікторини, вправи на відповідність, флешкартки, завдання на введення відповіді, групування та інші формати. Офіційний сайт Wordwall зазначає, що платформа має понад 30 типів активностей, зокрема quiz, match up, flash cards, type the answer та інші. Також Wordwall підтримує інтерактивні та друковані активності, які можна використовувати на різних пристроях [8].

.

Рис. 2.4 Приклад інтерфейсу платформи Wordwall

Для розроблюваної теми Wordwall є важливим аналогом, оскільки він безпосередньо пов’язаний із навчальними іграми. На відміну від Duolingo або Memrise, Wordwall більше орієнтований не на повний мовний курс, а на створення окремих інтерактивних вправ. Це наближує його до ідеї Memory Game, де головною одиницею взаємодії є конкретна навчальна активність

Переваги Wordwall [8]:

– велика кількість шаблонів навчальних ігор;

– можливість швидко створювати інтерактивні вправи;

– підтримка різних типів завдань;

– зручність для викладачів;

– можливість використання на різних пристроях;

– наявність готових матеріалів спільноти.

Недоліки Wordwall:

– платформа є більш загальною, а не спеціалізованою саме для вивчення англійської лексики;

– велика кількість шаблонів може ускладнювати інтерфейс;

– якість готових вправ залежить від користувачів, які їх створюють;

– розширені можливості можуть бути обмежені умовами тарифного плану;

– система більше орієнтована на викладача як автора активностей, ніж на індивідуального користувача, який проходить гру самостійно.

2.5 Anki

Anki - це програмний засіб для створення електронних карток і повторення матеріалу, переглянути на рисунку 2.5. Основний принцип роботи Anki пов’язаний із регулярним повторенням інформації через картки. Користувач створює або імпортує набори карток, після чого система допомагає повторювати їх у певній послідовності [9].

Для теми Memory Game Anki є корисним прикладом не з точки зору ігрової форми, а з точки зору організації карткової структури. У подібних системах

важливо правильно зберігати навчальні одиниці: слово, переклад, приклад, категорію, рівень складності та історію повторень. Саме ці сутності можуть бути використані при проєктуванні бази даних власного застосунку.

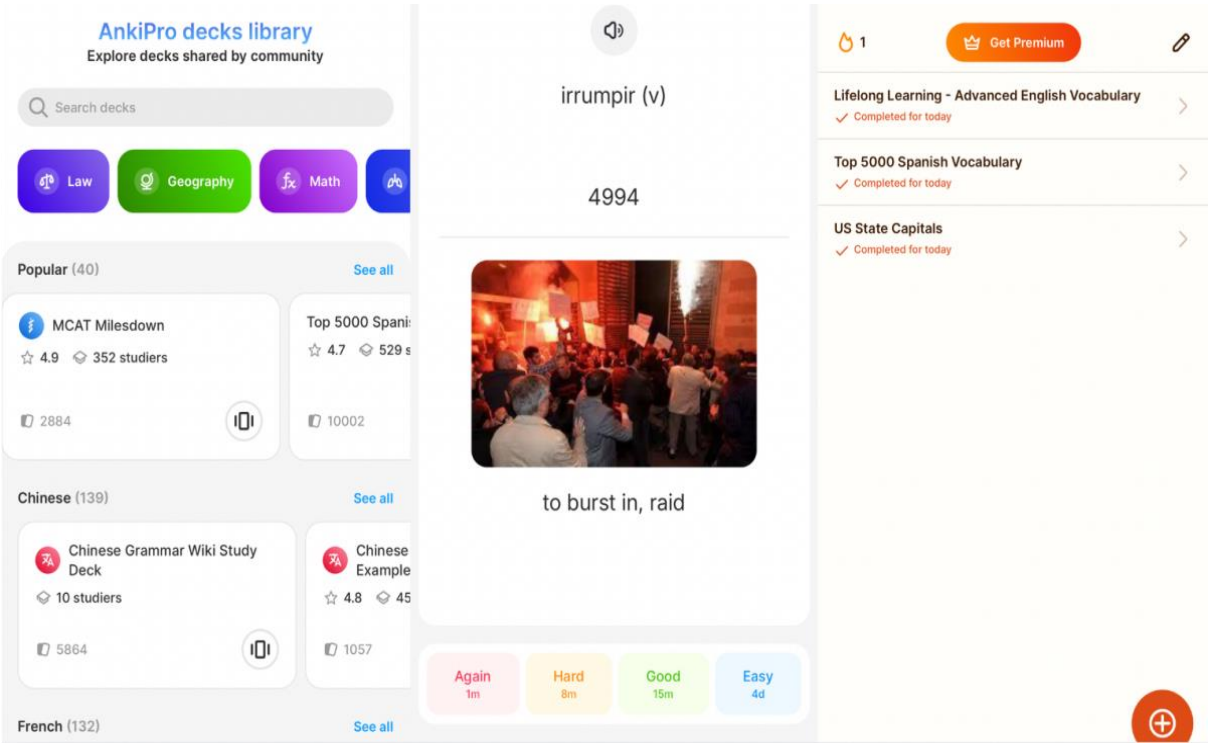


Рис. 2.5 Приклад інтерфейсу програмного засобу Anki

Переваги Anki [9]:

– гнучка система створення карток;

– можливість працювати з великими наборами слів;

– підтримка різних типів контенту в картках;

– ефективність для довготривалого запам’ятовування;

– можливість імпорту та експорту даних;

– наявність версій для різних платформ.

Недоліки Anki:

– інтерфейс може бути складним для нових користувачів

– відсутня повноцінна ігрова механіка;

– візуальна частина менш приваблива порівняно з гейміфікованими застосунками;

– користувач часто має самостійно створювати або налаштовувати набори карток;

– система більше орієнтована на повторення, ніж на інтерактивне проходження рівнів.

2.6 Узагальнення результатів аналізу

Проведений аналіз показує, що існуючі програмні засоби мають різні підходи до вивчення англійської лексики. Duolingo робить акцент на гейміфікованих коротких уроках, Quizlet - на картках і тестуванні, Memrise - на контексті та відео з носіями мови, Wordwall - на створенні інтерактивних навчальних активностей, а Anki - на повторенні матеріалу за допомогою карток [5-9].

З погляду розробки власного веб-застосунку найбільш корисними є такі ідеї:

– використання карток як основної одиниці навчального матеріалу;

– поділ слів за темами та рівнями складності;

– ігрова взаємодія з користувачем;

– підрахунок результату після проходження рівня;

– збереження прогресу користувача;

– можливість подальшого розширення системи.

Водночас більшість існуючих рішень або є надто загальними, або мають надмірний функціонал, або не зосереджені саме на механіці Memory Game. Тому розробка окремого веб-застосунку для вивчення англійської лексики у форматі гри на запам’ятовування є доцільною.

Таблиця 2.1

Порівняльна таблиця з аналогами

Показник	**Duolingo**	**Quizlet**	**Memrise**	**Wordwall**	**MemoryGame**
Карткова модель	обмежено	+	обмежено	+	+
Ігрова механіка	+	окремі режими	окремі елементи	+	+
Вибір складності	+	через налаштування	+	через шаблони	+
Орієнтація на англійську лексику	мовний курс	різні предмети	мовний курс	різні теми	+
Простота інтерфейсу	+	потребує адаптації	потребує адаптації	потребує адаптації	+
Зручність моделі даних	складна	зручна	комбінована	шаблонна	оптимальна
Роль адміністратора	-	автор наборів	-	+	+
Керування картками	-	+	-	+	+
Повторення слабких слів	-	-	-	-	+
Журнал дій адміністратора	-	-	-	-	+

3 ОГЛЯД ЗАСОБІВ РЕАЛІЗАЦІЇ WEB-ЗАСТОСУНКУ MEMORYGAME

3.1 Мова програмування Java

Для реалізації серверної частини web-застосунку MemoryGame обрано мову програмування Java. Java є однією з найбільш поширених мов для створення серверних, корпоративних та web-орієнтованих програмних систем. Вона підтримує об’єктно-орієнтований підхід, що дозволяє будувати програму у вигляді окремих класів, сервісів, контролерів, моделей даних та репозиторіїв [10].

У межах проєкту MemoryGame Java використовується для реалізації backend-логіки системи. Саме серверна частина відповідає за обробку запитів користувачів, авторизацію, роботу з ролями USER та ADMIN, отримання словникових карток, збереження результатів гри, формування прогресу користувача та підбір слів для адаптивного тренування.

Перевагою Java для такого проєкту є чітка структура коду та можливість розділити програму на логічні частини. Наприклад, сутності користувача, словникової картки, результату гри, прогресу по словах і журналу дій адміністратора можуть бути описані окремими класами.

У web-застосунку MemoryGame Java використовується разом із фреймворком Spring Boot, що дозволяє швидше створити серверну частину, налаштувати REST API, підключити базу даних та реалізувати взаємодію з клієнтською частиною [11].

3.2 Spring Boot та Spring Security

Для створення серверної частини застосунку використовується Spring Boot. Це фреймворк, який спрощує розробку Java-застосунків за рахунок готової структури, автоматичного налаштування компонентів і зручної інтеграції з іншими модулями екосистеми Spring [11].

У проєкті MemoryGame Spring Boot використовується для реалізації REST API, через яке frontend-частина отримує та передає дані. Наприклад, клієнтська частина надсилає запити для входу користувача, реєстрації, отримання карток певного рівня складності, збереження результату гри, перегляду прогресу або виконання адміністративних дій [11].

Основними компонентами backend-частини є:

– контролери, які приймають HTTP-запити від клієнтської частини;

– сервіси, які містять бізнес-логіку застосунку;

– репозиторії, які забезпечують доступ до бази даних;

– сутності, які описують таблиці бази даних;

– DTO-об’єкти, які використовуються для передавання даних між клієнтом і сервером [22; 23].

Окрему роль у проєкті відіграє Spring Security. Цей модуль використовується для захисту системи, автентифікації користувачів і розмежування доступу між ролями USER та ADMIN. Завдяки цьому звичайний користувач може працювати з грою, прогресом і профілем, але не має доступу до адміністративної панелі. Адміністратор, у свою чергу, отримує доступ до керування словником, картками, рівнями складності, користувачами та статистикою платформи [12].

Для авторизації використовується JWT. Після успішного входу користувача система формує токен, який надалі використовується під час звернення до захищених ресурсів. Це дозволяє не зберігати стан сесії на сервері та зручно організувати взаємодію між frontend- і backend-частинами [13].

3.3 Spring Data JPA та Hibernate

Для роботи з базою даних у web-застосунку MemoryGame використовується Spring Data JPA. Цей інструмент дозволяє спростити доступ до реляційної бази даних за рахунок використання репозиторіїв та об’єктної моделі. Замість написання великої кількості SQL-запитів вручну розробник може працювати з Java-класами, які відповідають таблицям бази даних [14].

Hibernate використовується як реалізація ORM-підходу. ORM дозволяє пов’язати об’єкти програми з таблицями бази даних. Наприклад, клас User відповідає таблиці users, клас WordCard - таблиці word_cards, клас GameResult - таблиці game_results, а клас UserWordProgress - таблиці user_word_progress [15].

Такий підхід є зручним для проєкту MemoryGame, оскільки система працює з великою кількістю пов’язаних даних. Один користувач може мати багато результатів ігор, багато записів прогресу по словах, а адміністратор може створювати багато словникових карток. Використання JPA та Hibernate спрощує опис таких зв’язків і дозволяє працювати з ними на рівні Java-коду.

Spring Data JPA також забезпечує зручне створення методів для пошуку, фільтрації та сортування даних. Наприклад, у системі можна отримувати активні картки певного рівня складності, знаходити результати конкретного користувача, формувати список слабких слів або отримувати журнал дій адміністратора [14].

3.4 PostgreSQL

Для зберігання даних у web-застосунку MemoryGame використовується PostgreSQL. Це реляційна система керування базами даних, яка добре підходить для зберігання структурованої інформації та підтримує зв’язки між таблицями [16].

У проєкті база даних зберігає такі основні дані:

– акаунти користувачів та адміністраторів;

– словникові картки;

– результати завершених ігор;

– деталізацію результатів по кожній картці;

– персональний прогрес користувача по кожному слову;

– журнал дій адміністратора.

Використання PostgreSQL є доцільним, оскільки застосунок має чітко структуровані сутності та зв’язки між ними. Наприклад, один користувач може мати багато результатів гри, один результат гри може містити багато карток, одна

картка може входити до різних результатів, а прогрес користувача формується окремо по кожному слову [16].

Також PostgreSQL дозволяє забезпечити цілісність даних за рахунок первинних і зовнішніх ключів. Це важливо для коректної роботи системи, оскільки результати гри, прогрес і словникові картки повинні бути пов’язані між собою [16].

3.5 Liquibase

Для керування змінами структури бази даних використовується Liquibase. Цей інструмент дозволяє описувати зміни в базі даних у вигляді міграцій. Завдяки цьому структура бази даних може змінюватися контрольовано та послідовно [17].

У проєкті MemoryGame Liquibase може використовуватися для створення таблиць users, word_cards, game_results, game_result_cards, user_word_progress та activity_logs. Також за допомогою міграцій можна додавати нові поля, змінювати структуру таблиць, створювати зв’язки між сутностями або додавати початкові тестові дані [17].

Перевага використання Liquibase полягає в тому, що всі зміни бази даних зберігаються в проєкті разом із програмним кодом. Це спрощує запуск застосунку на іншому комп’ютері або сервері, оскільки структура бази може бути відтворена автоматично [17].

Для MemoryGame це особливо корисно, оскільки система має кілька пов’язаних таблиць і потребує чіткої організації даних. У майбутньому, якщо буде додано нові функції, наприклад досягнення, рейтинги або аудіовимову слів, структуру бази можна буде розширити через нові міграції.

3.6 React, Vite та TypeScript

Клієнтська частина web-застосунку MemoryGame реалізується з використанням React. React дозволяє створювати інтерфейс із окремих компонентів, що є зручним для побудови інтерактивної гри. Наприклад, у

застосунку окремими компонентами можуть бути картка, ігрове поле, панель результатів, сторінка прогресу, форма входу, профіль користувача та адміністративна панель [18].

Для MemoryGame компонентний підхід є особливо доцільним, оскільки інтерфейс містить багато повторюваних елементів. Картки можуть багаторазово використовуватися в основній грі, адаптивному тренуванні або адміністративному перегляді словника. Це зменшує дублювання коду та робить frontend-частину більш зручною для підтримки.

Vite використовується як інструмент для швидкого запуску та збирання клієнтської частини. Він забезпечує швидкий старт проєкту, зручний режим розробки та оптимізоване збирання готового застосунку [19].

TypeScript використовується для підвищення стабільності frontend-коду. Завдяки типізації можна чітко описати структуру даних, які надходять із backend-частини: користувача, картку, результат гри, прогрес по слову або статистику. Це зменшує кількість помилок під час розробки та полегшує роботу з великими об’єктами даних.

3.7 Axios, React Router та REST API

Для обміну даними між клієнтською та серверною частинами використовується REST API. Frontend-частина надсилає HTTP-запити до backend-частини, а сервер повертає відповіді у форматі JSON, переглянути на рисунку 3.1. Такий підхід дозволяє відокремити інтерфейс користувача від серверної логіки [22; 23].

У проєкті MemoryGame через REST API виконуються такі операції: [22; 23]

– реєстрація та авторизація користувача;

– отримання даних профілю;

– отримання карток для гри;

– збереження результату проходження рівня;

– отримання статистики прогресу;

– формування адаптивного тренування;

– керування словниковими картками;

– керування користувачами в адміністративній панелі.

Для виконання HTTP-запитів на клієнтській частині використовується Axios. Він дозволяє зручно надсилати запити до API, передавати JWT-токен, отримувати відповіді сервера та обробляти помилки [20].

React Router використовується для організації переходів між сторінками застосунку. У системі передбачені такі основні маршрути: [21]

– /login - сторінка входу;

– /register - сторінка реєстрації;

– /dashboard - головна сторінка;

– /game - сторінка гри;

– /adaptive - сторінка адаптивного тренування;

– /progress - сторінка перегляду прогресу;

– /profile - профіль користувача;

– /admin - панель адміністратора.

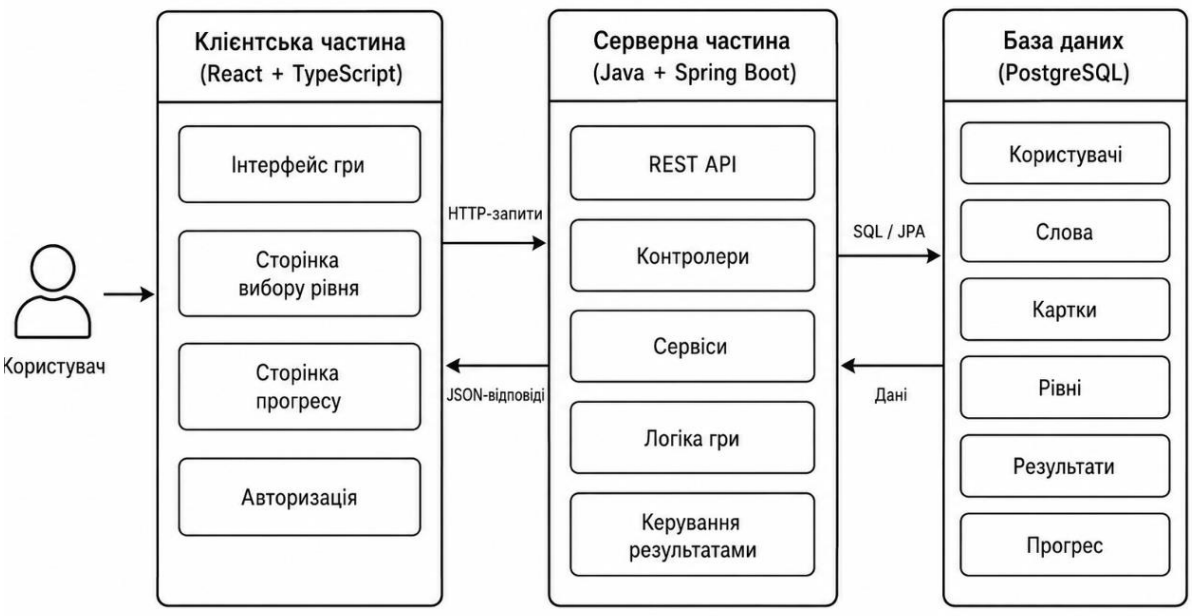


Рис. 3.1 Схема обміну даними між клієнтською та серверною частиною через REST API

3.8 Узагальнення засобів реалізації

Обрані засоби реалізації дозволяють побудувати web-застосунок MemoryGame як клієнт-серверну систему з чітким розподілом відповідальності між окремими частинами. Клієнтська частина на React відповідає за інтерфейс, взаємодію з картками, маршрутизацію між сторінками та відображення результатів. Серверна частина на Java та Spring Boot обробляє запити, реалізує бізнес-логіку, забезпечує авторизацію, керує ролями та працює з базою даних. PostgreSQL використовується для збереження структурованих даних, а Liquibase - для керування змінами структури бази, переглянути на рисунку 3.2[11].

Загальна архітектура застосунку дозволяє надалі розширювати систему. Наприклад, до MemoryGame можна додати нові мови, аудіовимову слів, рейтинги користувачів, досягнення, додаткові режими тренування або розширену аналітику навчального прогресу. Обраний технологічний стек є достатньо гнучким для реалізації як базових функцій гри, так і подальшого розвитку платформи.

Рис. 3.2 - Технологічний стек розробки web-застосунку MemoryGame

4 ПРОЄКТУВАННЯ WEB-ЗАСТОСУНКУ MEMORYGAME

4.1 Логічна структура системи

Проєктування web-застосунку MemoryGame передбачає визначення загальної структури системи, основних модулів, ролей користувачів, зв’язків між компонентами та способу обміну даними між клієнтською і серверною частинами. Оскільки застосунок призначений для вивчення англійської лексики через інтерактивну гру, його структура повинна забезпечувати не лише запуск ігрового процесу, а й збереження результатів, формування прогресу, адаптивне повторення слів та адміністрування навчального контенту.

Web-застосунок MemoryGame побудований за клієнт-серверною архітектурою. Клієнтська частина відповідає за відображення інтерфейсу користувача, маршрутизацію між сторінками, взаємодію з картками та передавання запитів до серверної частини. Серверна частина реалізує основну бізнес-логіку: авторизацію, перевірку ролей, роботу з картками, збереження результатів гри, формування прогресу користувача та підбір слів для адаптивного тренування. База даних використовується для збереження користувачів, словникових карток, результатів ігор, персонального прогресу та журналу дій адміністратора.

До основних модулів системи належать:

– модуль автентифікації та авторизації;

– модуль користувача;

– ігровий модуль;

– модуль прогресу;

– модуль адаптивного тренування;

– адміністративний модуль;

– модуль статистики;

– модуль роботи з базою даних.

Модуль автентифікації забезпечує реєстрацію, вхід користувача до системи, перевірку облікових даних та формування JWT-токена. Модуль користувача відповідає за перегляд і редагування профілю, збереження персональних налаштувань і навчальних цілей. Ігровий модуль реалізує процес проходження гри: вибір рівня складності, отримання набору карток, відкриття карток, пошук пар, фіксацію ходів і помилок [13].

Модуль прогресу забезпечує відображення результатів користувача, зокрема кількості пройдених ігор, найкращого результату, середнього балу, кількості опрацьованих слів і динаміки навчання. Модуль адаптивного тренування аналізує статистику по кожному слову та формує персоналізований набір карток для повторення. Адміністративний модуль дозволяє керувати словниковими картками, рівнями складності, користувачами та статистикою платформи.

4.2 Діаграма варіантів використання web-застосунку MemoryGame

Для опису взаємодії користувачів із системою доцільно використати діаграму варіантів використання. Вона дозволяє показати основні ролі системи та функції, які доступні кожній із них [24].

У web-застосунку MemoryGame передбачено дві основні ролі: USER та ADMIN. Роль USER призначена для звичайного користувача, який використовує систему для вивчення англійських слів. Роль ADMIN призначена для адміністратора, який відповідає за підтримку навчального контенту, керування користувачами та перегляд статистики.

Користувач з роллю USER може виконувати такі дії:

– зареєструватися в системі;

– увійти до облікового запису;

– обрати рівень складності гри;

– запустити гру;

– відкривати картки;

– шукати правильні пари між англійським словом і українським перекладом;

– переглядати результат проходження гри;

– повторно проходити рівень;

– переглядати навчальний прогрес;

– проходити адаптивне тренування;

– редагувати профіль.

Адміністратор має ширший набір функцій, пов’язаних із керуванням системою. Він може переглядати статистику платформи, додавати нові слова, редагувати картки, активувати або деактивувати навчальні матеріали, керувати рівнями складності, переглядати користувачів, змінювати їх ролі, блокувати або активувати акаунти.

Діаграма варіантів використання дозволяє наочно показати, що основна навчальна логіка зосереджена навколо користувача, тоді як адміністративна логіка винесена в окремий блок. Такий поділ є важливим для безпеки та зручності системи, оскільки звичайний користувач не повинен мати доступ до функцій зміни словникової бази або керування іншими користувачами.

4.3 Проєктування бази даних

База даних web-застосунку MemoryGame призначена для збереження структурованої інформації про користувачів, словникові картки, результати проходження ігор, деталізацію результатів, персональний прогрес і журнал дій адміністратора. Для реалізації бази даних використовується PostgreSQL [16].

Основними таблицями бази даних є:

– users;

– word_cards;

– game_results;

– game_result_cards;

– user_word_progress;

– activity_logs.

Таблиця users зберігає дані користувачів і адміністраторів. Вона містить email, хеш пароля, ім’я користувача, роль, телефон, колір аватара, навчальну ціль, денну ціль, статус активності акаунта та службові дати створення й оновлення запису. Поле role визначає тип користувача: USER або ADMIN.

Таблиця word_cards використовується для збереження словникових карток. Кожна картка містить англійське слово, український переклад, категорію, рівень складності, підказку, приклад речення, колір картки, порядок сортування, статус активності та посилання на адміністратора, який створив картку.

Таблиця game_results зберігає загальний результат завершеної гри. До неї входять користувач, рівень складності, кількість ходів, кількість помилок, кількість знайдених пар, підсумковий бал, номер ігрового дня, тривалість проходження та дата створення запису.

Таблиця game_result_cards деталізує результат гри по кожній картці. Вона дозволяє зберігати, скільки спроб було зроблено для конкретного слова, скільки помилок допущено та чи була пара знайдена.

Таблиця user_word_progress зберігає персональний прогрес користувача по кожному слову. Саме ця таблиця є основою для адаптивного тренування, оскільки містить кількість спроб, помилок, успішних повторень, рівень засвоєння слова та дату останнього повторення.

Таблиця activity_logs використовується для збереження журналу дій адміністратора. Вона містить інформацію про адміністратора, тип виконаної дії, її опис і дату створення запису. Це дозволяє відстежувати зміни, які виконуються в адміністративній панелі, переглянути структуру на рисунку 4.1.

Основні зв’язки між таблицями такі:

– один користувач може мати багато результатів гри;

– один користувач може мати багато записів прогресу по словах;

– один адміністратор може створювати багато словникових карток;

– один результат гри може містити багато записів по картках;

– одна словникова картка може входити до багатьох результатів гри;

– одна словникова картка може мати багато записів прогресу користувачів;

– один адміністратор може мати багато записів у журналі дій.

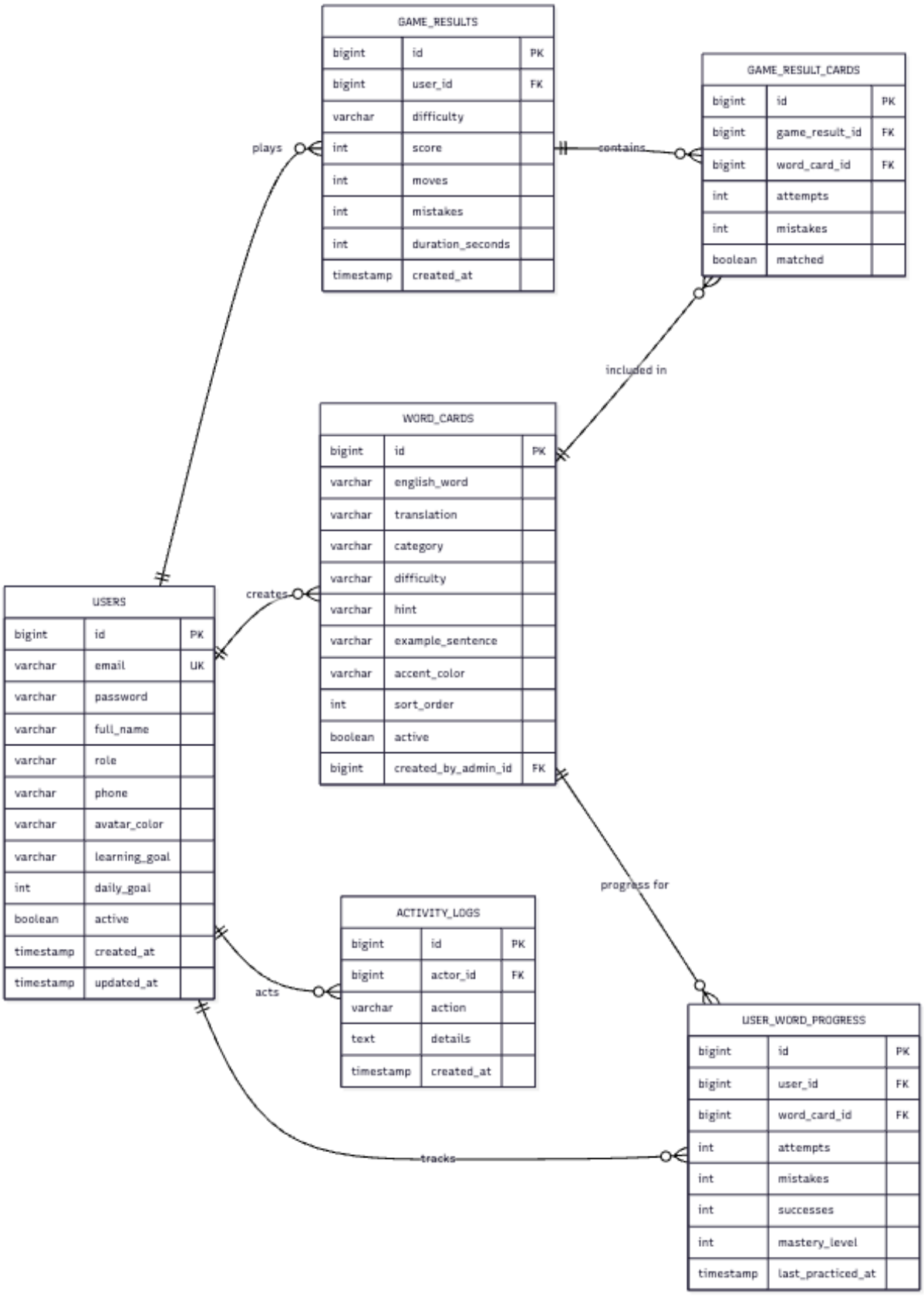


Рис. 4.1 Структура бази даних web-застосунку MemoryGame

4.4 Архітектура web-застосунку MemoryGame

Архітектура web-застосунку MemoryGame побудована за багаторівневим принципом. Такий підхід дозволяє розділити систему на окремі рівні, кожен із яких відповідає за власну частину логіки, переглянути архітектуру застосунку на рисунку 4.2.

У системі можна виділити три основні рівні:

– рівень представлення;

– рівень бізнес-логіки;

– рівень даних.

Рівень представлення відповідає за взаємодію користувача з web-застосунком. До нього належать сторінка входу, сторінка реєстрації, головна сторінка, сторінка гри, сторінка адаптивного тренування, сторінка прогресу, профіль користувача та адміністративна панель. Цей рівень реалізується за допомогою React, Vite, TypeScript, Axios і React Router [19].

Рівень бізнес-логіки реалізується на серверній частині за допомогою Java та Spring Boot. Він містить модулі автентифікації, користувача, гри, прогресу, адаптивного тренування, адміністративний модуль та аналітичний модуль. Саме цей рівень обробляє запити, перевіряє права доступу, формує набори карток, зберігає результати та виконує логіку підбору слабких слів [11].

Рівень даних відповідає за доступ до бази даних. Він включає репозиторії, JPA / Hibernate та PostgreSQL. Через цей рівень серверна частина отримує користувачів, картки, результати, прогрес і журнали дій [16].

Взаємодія між рівнями відбувається за допомогою REST API та механізмів доступу до бази даних. Клієнтська частина надсилає HTTP-запити до backend-частини, сервер обробляє їх, звертається до бази даних через JPA / Hibernate і повертає відповідь у форматі JSON [22; 23].

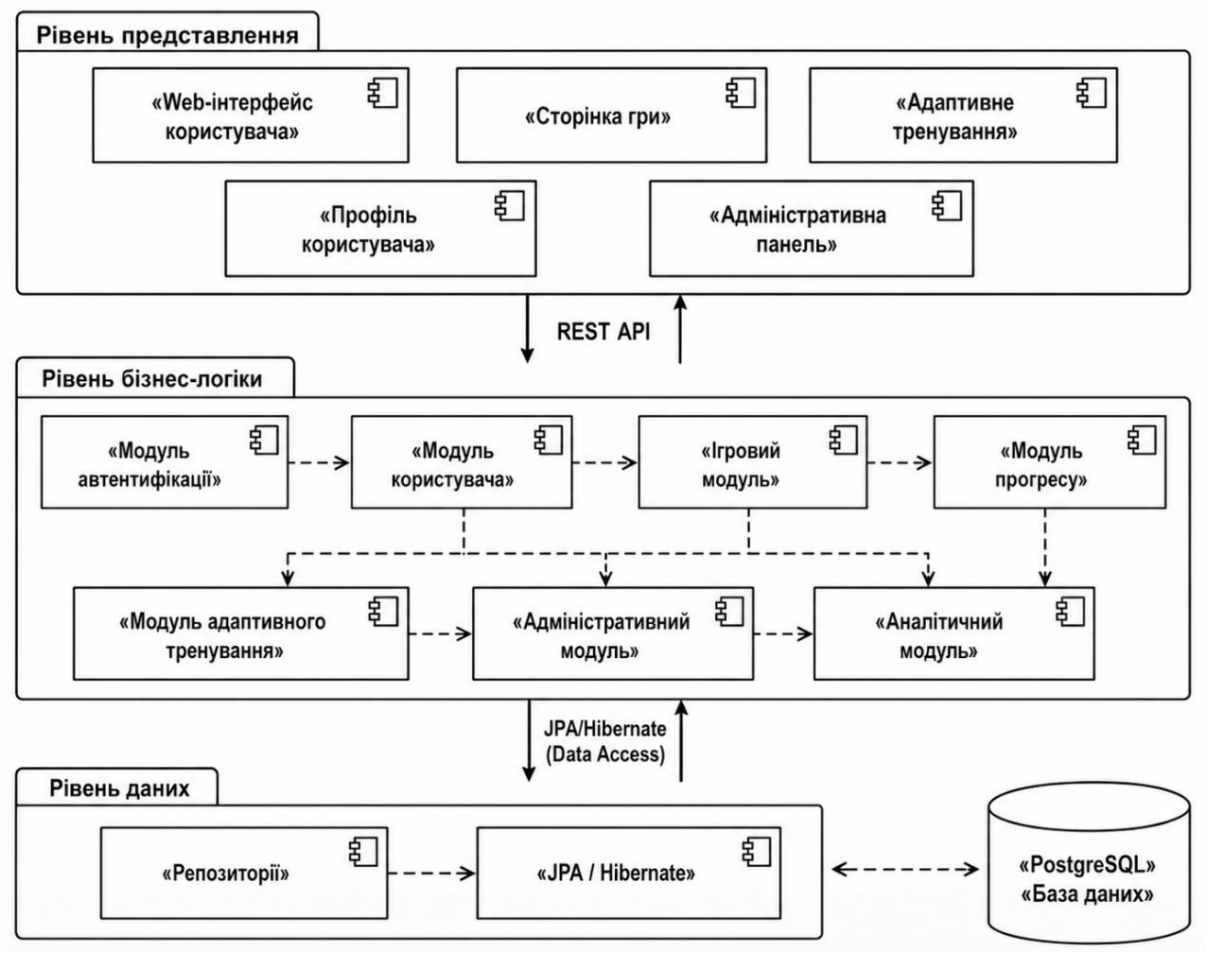


Рис. 4.2 Архітектура web-застосунку MemoryGame

4.5 Діаграма класів web-застосунку MemoryGame

Діаграма класів використовується для опису основних програмних сутностей web-застосунку та зв’язків між ними. У системі MemoryGame класи можна умовно поділити на класи запуску застосунку, контролери, сервіси та сутності бази даних, переглянути діаграму класів на рисунку 4.3.

Центральним класом є MemoryGameApplication, який відповідає за запуск Spring Boot-застосунку. Контролери приймають HTTP-запити від клієнтської частини та передають їх до відповідних сервісів. Сервіси реалізують бізнес-логіку системи [11].

До основних контролерів належать:

– AuthController - відповідає за реєстрацію та вхід користувача;

– GameController - обробляє запити, пов’язані з грою, картками, результатами та прогресом;

– UserController - відповідає за отримання та оновлення даних поточного користувача;

– AdminController - забезпечує адміністративні функції: керування користувачами, картками та статистикою.

Відповідні сервіси реалізують внутрішню логіку цих контролерів. AuthService виконує реєстрацію та авторизацію, GameService формує набори карток, зберігає результати і розраховує прогрес, UserService працює з профілем користувача, а AdminService виконує адміністративні операції.

До основних сутностей системи належать:

– User - користувач або адміністратор системи;

– WordCard - словникова картка;

– GameResult - результат завершеної гри;

– UserWordProgress - персональний прогрес користувача по слову.

Діаграма класів дозволяє показати, як контролери пов’язані із сервісами, а сервіси - із сутностями даних. Така структура відповідає типовій архітектурі Spring Boot-застосунку та спрощує підтримку системи [11].

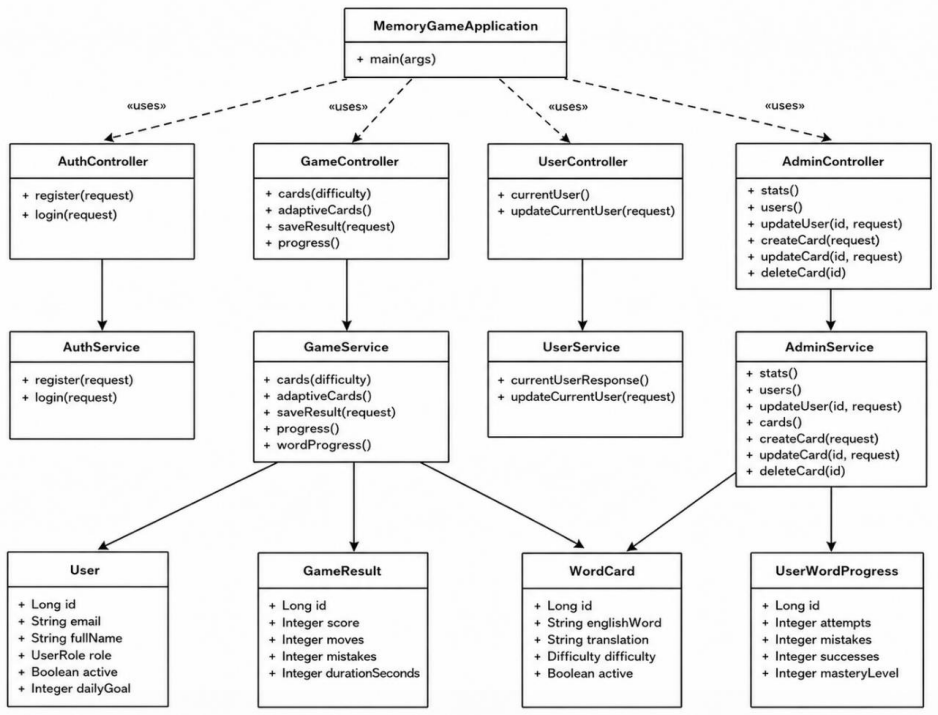


Рис. 4.3 Діаграма класів web-застосунку MemoryGame

4.6 Діаграма розгортання web-застосунку

Діаграма розгортання використовується для відображення фізичної або логічної структури розміщення компонентів системи. Для web-застосунку MemoryGame вона демонструє, які частини системи запускаються окремо та як вони взаємодіють між собою, переглянути діаграму на рисунку 4.4.

Користувач працює із системою через web-браузер. Клієнтська частина запускається як React-застосунок і доступна за адресою http://127.0.0.1:3001/. Через інтерфейс користувач надсилає запити до backend-частини [18].

Серверна частина працює як Spring Boot-застосунок і надає API за адресою http://127.0.0.1:8080/api. Backend обробляє запити, перевіряє JWT-токени, виконує бізнес-логіку та звертається до бази даних [11].

База даних PostgreSQL використовується для збереження всієї основної інформації системи: користувачів, карток, результатів, прогресу та журналу дій адміністратора. Взаємодія між backend-частиною та базою даних здійснюється через Spring Data JPA та Hibernate [14].

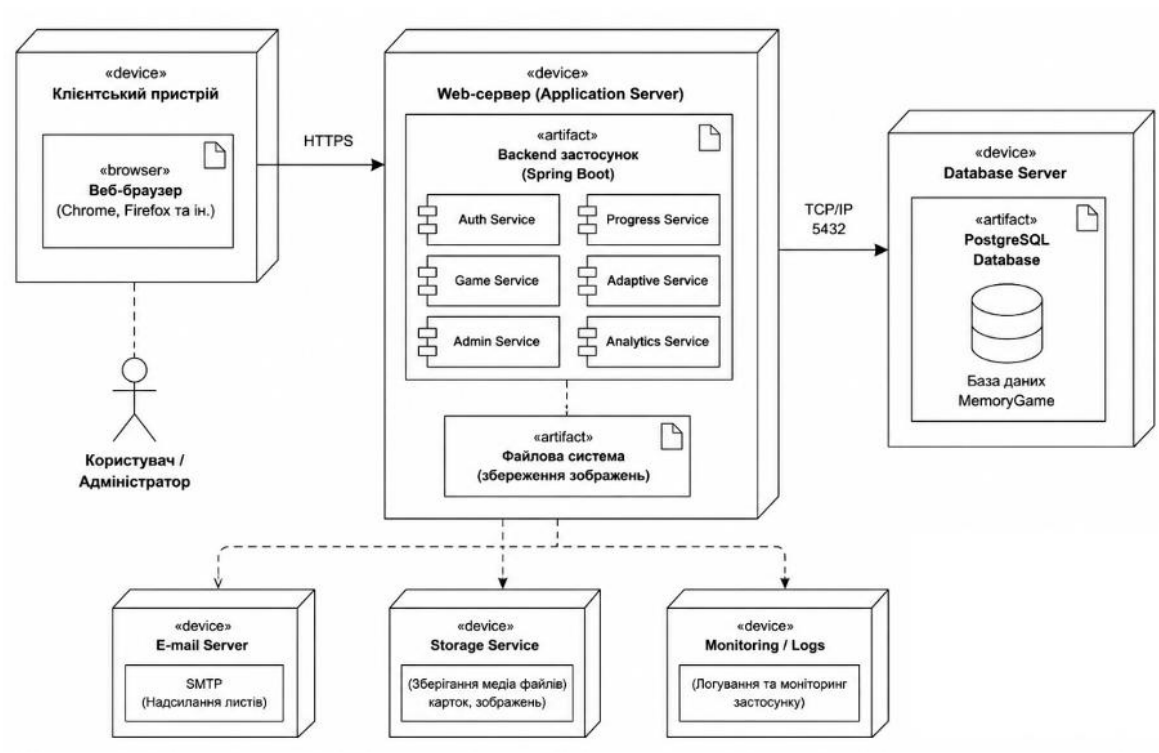


Рис. 4.4 Діаграма розгортання web-застосунку MemoryGame

5 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ WEB-ЗАСТОСУНКУ MEMORYGAME

5.1 Реалізація серверної частини

Серверна частина web-застосунку MemoryGame реалізована мовою програмування Java з використанням фреймворку Spring Boot. Backend-частина відповідає за обробку запитів клієнтської частини, автентифікацію користувачів, перевірку ролей, роботу зі словниковими картками, збереження результатів проходження гри, формування навчального прогресу та виконання адміністративних функцій [11].

Серверна частина побудована за модульним принципом. Основна логіка розділена між контролерами, сервісами, репозиторіями та сутностями бази даних. Контролери приймають HTTP-запити від клієнтської частини, сервіси виконують бізнес-логіку, репозиторії забезпечують доступ до бази даних, а сутності описують структуру таблиць PostgreSQL [16].

До основних серверних модулів належать:

– модуль автентифікації та авторизації;

– ігровий модуль;

– модуль роботи з користувачем;

– модуль навчального прогресу;

– модуль адаптивного тренування;

– адміністративний модуль;

– модуль журналу дій адміністратора.

Модуль автентифікації забезпечує реєстрацію користувача, вхід до системи та формування JWT-токена. Після успішної авторизації користувач отримує токен, який використовується для подальшого доступу до захищених ресурсів системи. Це дозволяє реалізувати розмежування доступу між звичайним користувачем і адміністратором [13].

У системі передбачено дві основні ролі: USER та ADMIN. Користувач з роллю USER має доступ до ігрового режиму, перегляду прогресу, адаптивного тренування та редагування профілю. Користувач з роллю ADMIN має доступ до адміністративної панелі, де може керувати словниковими картками, рівнями складності, користувачами та переглядати статистику платформи.

Ігровий модуль відповідає за отримання набору карток відповідного рівня складності, запуск гри, обробку результатів проходження та збереження статистики. Після завершення гри сервер отримує дані про кількість ходів, помилок, знайдених пар, підсумковий бал і тривалість проходження. Ці дані зберігаються в таблиці game_results, а деталізація по кожній картці - у таблиці game_result_cards.

Окрему роль виконує модуль адаптивного тренування. Він аналізує персональний прогрес користувача по кожному слову та формує набір карток для повторення. При цьому враховуються кількість спроб, кількість помилок, кількість успішних повторень і рівень засвоєння слова. Завдяки цьому система може пропонувати користувачу саме ті слова, які були засвоєні гірше.

5.2 Реалізація клієнтської частини

Клієнтська частина web-застосунку MemoryGame реалізована з використанням React, Vite, TypeScript/JavaScript, Axios та React Router. Вона відповідає за відображення інтерфейсу користувача, маршрутизацію між сторінками, взаємодію з картками, надсилання запитів до серверної частини та відображення результатів [19].

Frontend-частина має сторінкову структуру. Кожна основна функція системи винесена в окрему сторінку або компонент. Це дозволяє зробити інтерфейс зрозумілим для користувача та спростити підтримку програмного коду.

Основними сторінками web-застосунку є:

– /login - сторінка входу;

– /register - сторінка реєстрації;

– /dashboard - головна сторінка користувача;

– /game - сторінка гри;

– /adaptive - сторінка адаптивного тренування;

– /progress - сторінка перегляду навчального прогресу;

– /profile - сторінка профілю користувача;

– /admin - адміністративна панель.

Сторінка входу містить форму авторизації, де користувач вводить email та пароль,переглянути на рисунку 5.1. Після успішного входу система отримує JWT-токен і зберігає його для подальших запитів до захищених ресурсів. Якщо користувач має роль USER, він переходить до головної сторінки. Якщо користувач має роль ADMIN, йому стає доступною адміністративна панель [13].

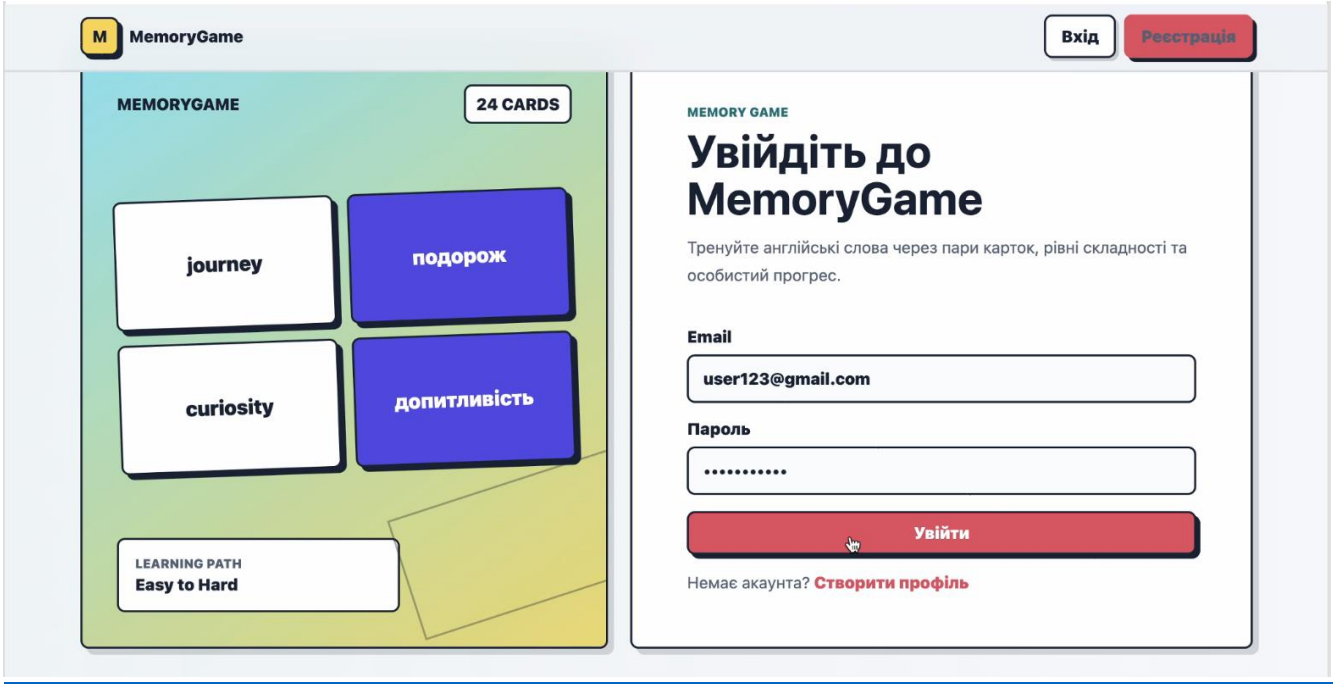


Рис. 5.1 Сторінка входу користувача

Головна сторінка /dashboard відображає основний навчальний маршрут користувача. На ній розміщуються блок адаптивного словникового маршруту, список доступних рівнів, останні результати та переходи до основних розділів системи,переглянути процес на рисунку 5.2. Користувач може швидко перейти до гри, адаптивного тренування, сторінки прогресу або профілю.

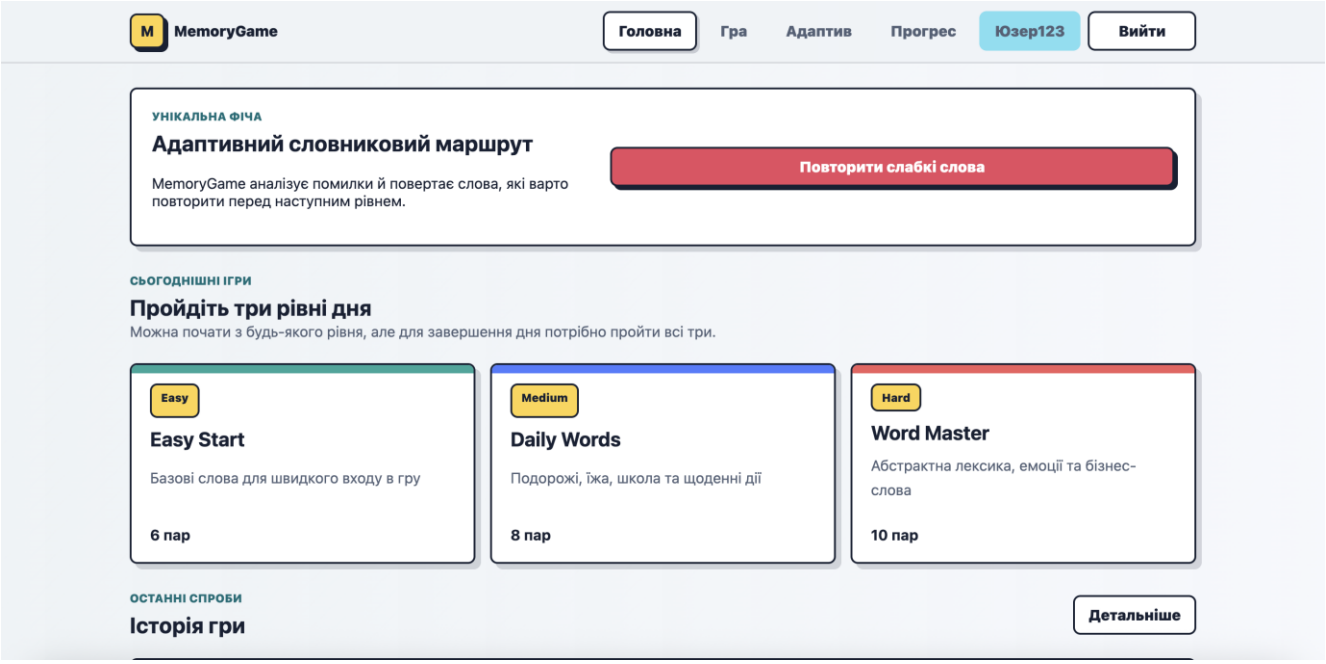


Рис. 5.2 Головна сторінка web-застосунку MemoryGame

Сторінка гри /game є основною частиною застосунку, переглянути процес на рисунку 5.3. На ній користувач обирає рівень складності Easy, Medium або Hard, після чого отримує набір карток. Картки містять англійські слова та їх українські переклади. Завдання користувача- відкривати картки та знаходити правильні пари. Під час гри система відображає кількість ходів, помилок і поточний стан проходження.

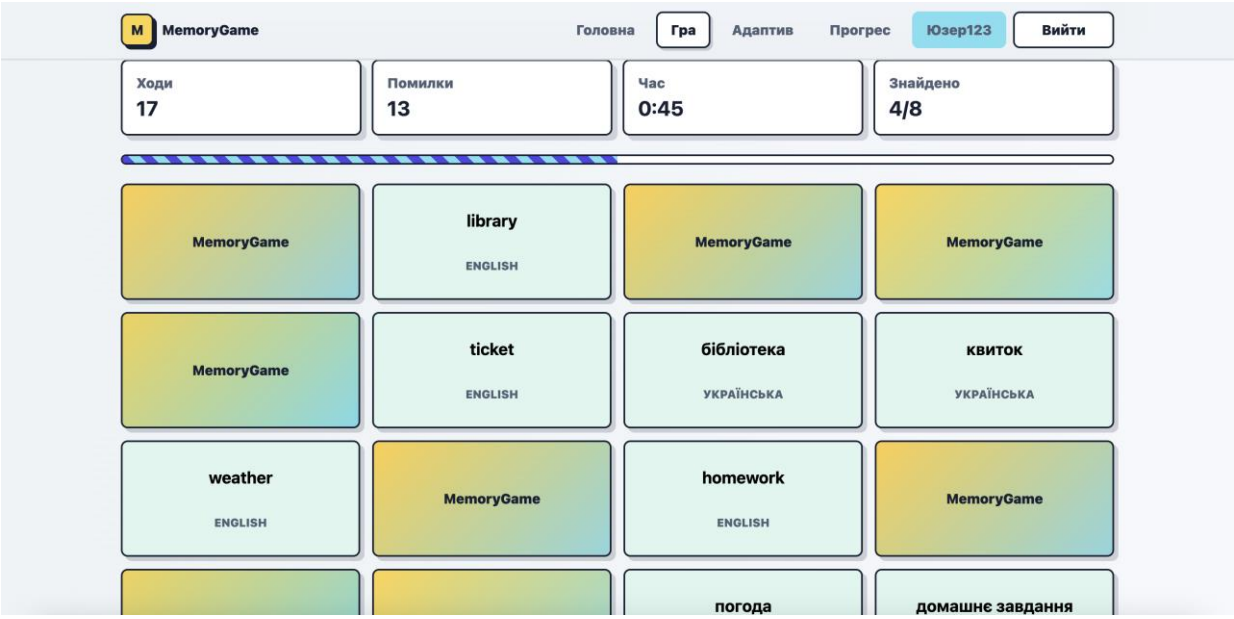


Рис. 5.3 - Процес проходження гри MemoryGame

Сторінка адаптивного тренування /adaptive призначена для повторення слабких слів. Система формує набір карток на основі попередніх помилок користувача та рівня засвоєння окремих слів, переглянути процес на рисунку 5.4. Це дозволяє зробити навчання більш персоналізованим і зосередженим на проблемних словах.

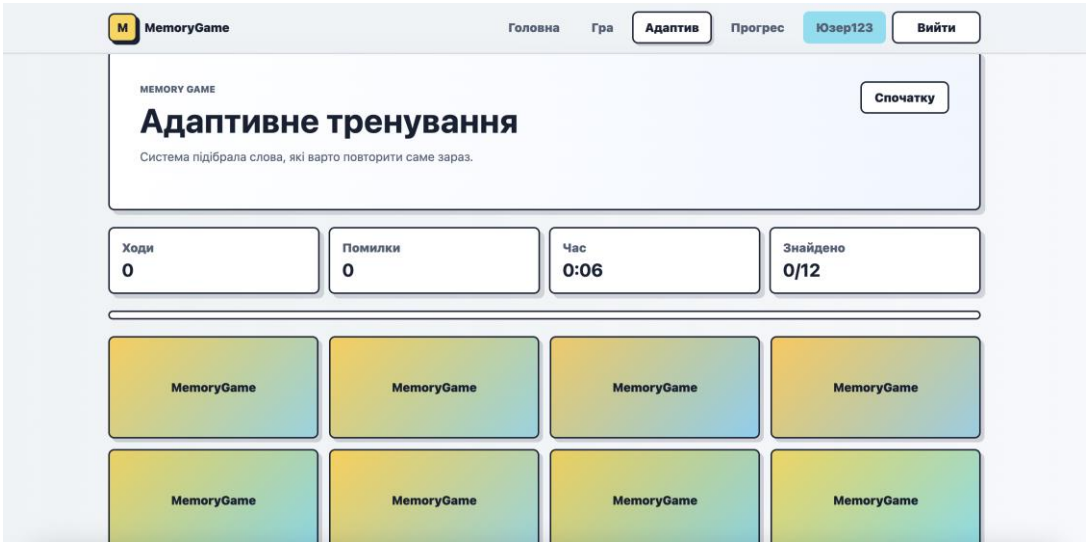


Рис. 5.4 Сторінка адаптивного тренування

Сторінка прогресу /progress відображає результати користувача, переглянути на рисунку 5.5. На ній можна переглянути кількість завершених ігор, найкращий результат, середній бал, кількість опрацьованих слів, слабкі слова, стабільно засвоєні слова та історію проходжень.

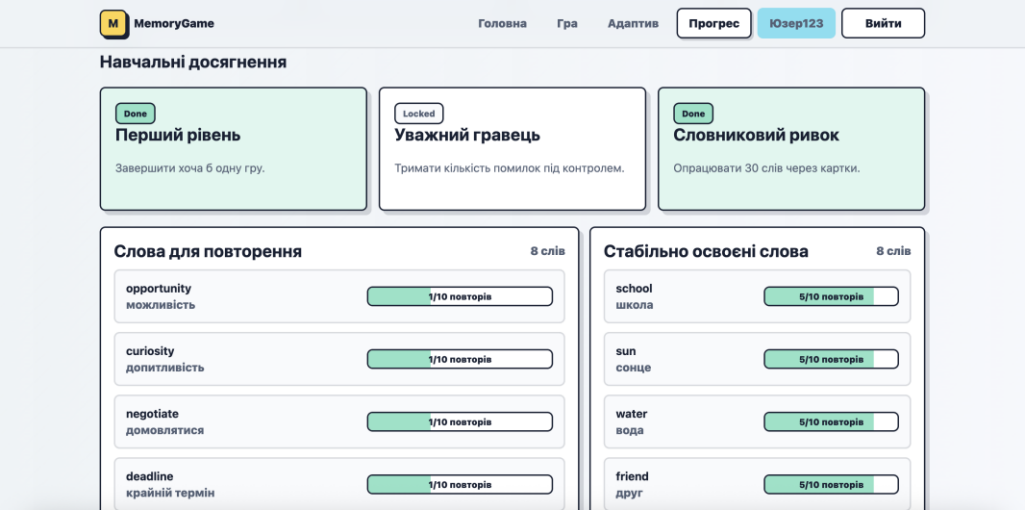


Рис. 5.5 Сторінка перегляду навчального прогресу

Сторінка профілю /profile містить персональні дані користувача: ім’я, телефон, колір аватара, навчальну ціль і денну ціль слів. Також на цій сторінці може відображатися коротка статистика профілю, переглянути на рисунку 5.6.

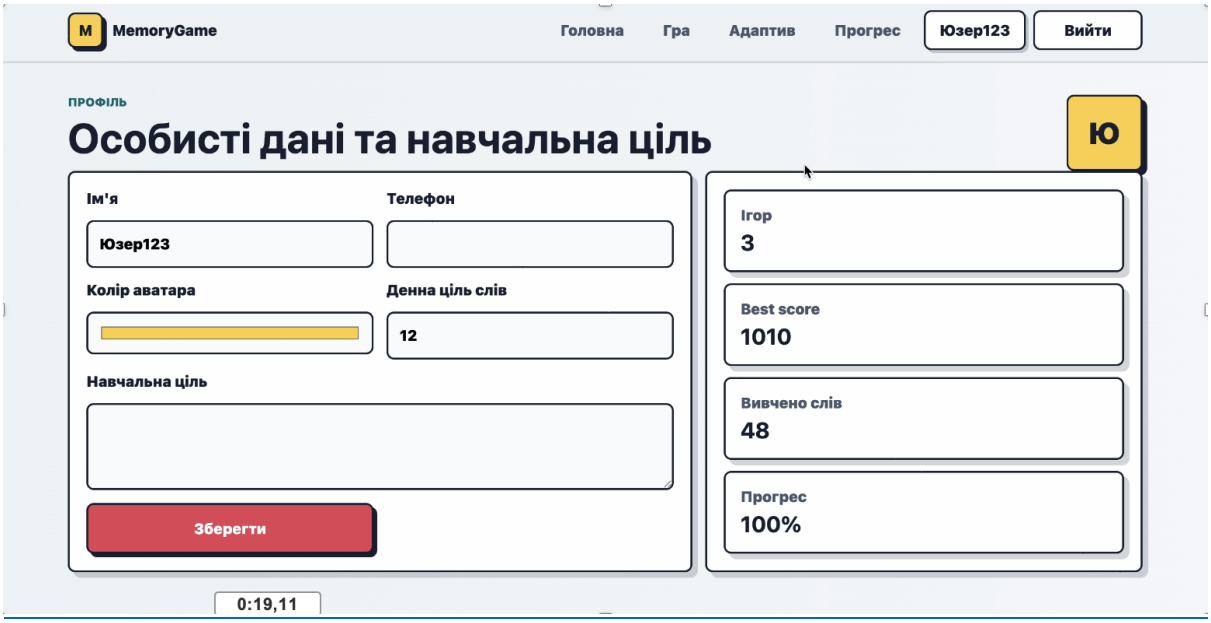


Рис. 5.6 Сторінка профілю користувача

5.3 Реалізація роботи з базою даних

Для зберігання даних у web-застосунку MemoryGame використовується PostgreSQL. Робота з базою даних організована через Spring Data JPA та Hibernate. Це дозволяє працювати з даними на рівні Java-класів, не описуючи всі SQL-запити вручну [14].

У базі даних реалізовано таблиці users, word_cards, game_results, game_result_cards, user_word_progress та activity_logs. Кожна таблиця відповідає окремій сутності системи та використовується для збереження конкретного типу даних.

Таблиця users зберігає облікові записи користувачів і адміністраторів. Для кожного користувача зберігається email, хеш пароля, ім’я, роль, телефон, колір аватара, навчальна ціль, денна ціль, статус активності акаунта та дати створення й оновлення запису.

Таблиця word_cards використовується для збереження словникових карток. Кожна картка містить англійське слово, український переклад, категорію, рівень складності, підказку, приклад речення, колір картки, порядок сортування, статус активності та ідентифікатор адміністратора, який створив запис.

Таблиця game_results містить результати завершених ігор. У ній зберігаються користувач, рівень складності, кількість ходів, кількість помилок, кількість знайдених пар, score, номер ігрового дня, тривалість проходження та дата створення результату.

Таблиця game_result_cards деталізує результат гри по кожному слову. Вона дозволяє визначити, які слова були використані в конкретній грі, скільки спроб було зроблено, скільки помилок допущено та чи була пара знайдена.

Таблиця user_word_progress зберігає персональний прогрес користувача по кожному слову. Саме на основі цієї таблиці працює адаптивне тренування. Якщо користувач часто помиляється в певному слові, рівень засвоєння знижується, і це слово частіше потрапляє до адаптивного повторення.

Таблиця activity_logs використовується для журналювання дій адміністратора. У ній зберігається адміністратор, який виконав дію, тип дії, опис і дата створення запису. Це дозволяє відстежувати зміни, які виконуються в адміністративній панелі.

Для керування змінами структури бази даних використовується Liquibase. Завдяки міграціям структура таблиць може створюватися та оновлюватися автоматично під час запуску проєкту. Це спрощує розгортання застосунку та забезпечує узгодженість структури бази даних із програмним кодом [17].

5.4 Реалізація REST API

Взаємодія між клієнтською та серверною частинами web-застосунку MemoryGame організована через REST API. Клієнтська частина надсилає HTTP-запити до backend-частини, а сервер повертає відповіді у форматі JSON [22; 23].

REST API дозволяє відокремити frontend від backend, що спрощує підтримку системи. Клієнтська частина відповідає за інтерфейс і взаємодію з користувачем, а серверна - за обробку даних, перевірку прав доступу, роботу з базою даних і виконання бізнес-логіки [22; 23].

Основні групи API-запитів можна поділити на такі категорії:

– запити автентифікації;

– запити користувача;

– запити гри;

– запити прогресу;

– запити адаптивного тренування;

– адміністративні запити.

Запити автентифікації використовуються для реєстрації та входу користувача. Після успішного входу сервер повертає JWT-токен. Далі цей токен передається в заголовках HTTP-запитів для доступу до захищених ресурсів [13].

Запити гри використовуються для отримання карток певного рівня складності, запуску ігрової сесії, перевірки результатів і збереження проходження. Наприклад, клієнтська частина може звернутися до серверної частини для отримання карток рівня Medium, після чого відобразити їх на ігровому полі.

Запити прогресу використовуються для отримання статистики користувача. Завдяки ним frontend може показати кількість ігор, найкращий score, середній результат, слабкі слова, засвоєні слова та історію проходжень.

Запити адаптивного тренування дозволяють отримати персональний набір карток, сформований на основі помилок і рівня засвоєння слів. Це є однією з ключових особливостей проєкту.

Адміністративні запити доступні лише користувачам із роллю ADMIN. Вони дозволяють додавати нові слова, редагувати картки, змінювати рівні складності, керувати користувачами, блокувати або активувати акаунти та переглядати статистику платформи.

5.5 Тестування програмного забезпечення

Тестування web-застосунку MemoryGame проводилося для перевірки коректності роботи основних функціональних модулів системи. Основна мета тестування полягає у підтвердженні того, що користувач може зареєструватися, увійти до системи, запустити гру, знайти пари карток, отримати результат, переглянути прогрес і пройти адаптивне тренування, а адміністратор може керувати словниковими картками та користувачами.

Тестування охоплювало такі модулі:

– модуль реєстрації та авторизації;

– ігровий модуль;

– модуль збереження результатів;

– модуль прогресу користувача;

– модуль адаптивного тренування;

– профіль користувача;

– адміністративну панель;

– REST API;

– роботу з базою даних [22; 23].

Під час тестування перевірялося, чи правильно система обробляє вхідні дані, чи коректно відображає інформацію на frontend-частині, чи зберігає результати гри в базі даних, чи правильно розмежовує доступ між USER та ADMIN, а також чи формує адаптивне тренування на основі реальних помилок користувача.

Для перевірки роботи системи були підготовлені тест-кейси. Їх приклади наведено в таблиці 5.1.

Таблиця 5.1

Тест-кейси web-застосунку MemoryGame

№	Модуль	Тестовий сценарій	Очікуваний результат	Статус
1	Авторизація	Вхід користувача USER з коректними даними	Користувач переходить на головну сторінку	Виконано
2	Авторизація	Вхід адміністратора з коректними даними	Адміністратор отримує доступ до панелі ADMIN	Виконано
3	Авторизація	Вхід з неправильним паролем	Система не дозволяє виконати вхід	Виконано
4	Гра	Запуск гри рівня Easy	Система відображає набір карток відповідного рівня	Виконано
5	Гра	Вибір правильної пари карток	Пара зараховується як знайдена	Виконано
6	Гра	Вибір неправильної пари карток	Система фіксує помилку	Виконано
7	Результати	Завершення гри	Результат зберігається в базі даних	Виконано
8	Прогрес	Відкриття сторінки прогресу	Користувач бачить статистику проходжень	Виконано
9	Адаптивне тренування	Формування набору слабких слів	Система показує слова з нижчим рівнем засвоєння	Виконано
10	Профіль	Оновлення даних профілю	Дані користувача змінюються та зберігаються	Виконано
11	Адмін-панель	Додавання нової картки	Картка з’являється у словниковій базі	Виконано
12	Адмін-панель	Деактивація картки	Картка не використовується у грі	Виконано
13	Адмін-панель	Блокування користувача	Користувач втрачає доступ до системи	Виконано
14	Безпека	USER відкриває /admin	Доступ заборонено	Виконано

5.6 Результати тестування

За результатами тестування встановлено, що основні функції web-застосунку MemoryGame працюють коректно. Користувач може зареєструватися, увійти до системи, обрати рівень складності, пройти гру, отримати результат і переглянути

навчальний прогрес. Дані про результати проходження зберігаються в базі даних і надалі використовуються для формування статистики.

Перевірка ігрового модуля показала, що система коректно відображає картки, дозволяє користувачу відкривати їх, визначає правильні та неправильні пари, фіксує кількість ходів і помилок. Після завершення рівня результат зберігається та відображається користувачу.

Тестування адаптивного тренування підтвердило, що система може формувати набір слів для повторення на основі попередніх помилок користувача. Слова з нижчим рівнем засвоєння потрапляють до повторення частіше, що відповідає логіці персоналізованого навчального маршруту.

Перевірка адміністративної панелі показала, що адміністратор може додавати нові слова, редагувати картки, змінювати їх активність, керувати рівнями складності та користувачами. Також було перевірено розмежування ролей: звичайний користувач не має доступу до адміністративного функціоналу.

Результати тестування наведено в таблиці 5.2.

Таблиця 5.2

Результати тестування основних функцій системи

Група функцій	**Кількість перевірених сценаріїв**	**Успішно виконано**	**Помилки**
Авторизація та реєстрація	3	3	0
Ігровий модуль	3	3	0
Збереження результатів	1	1	0
Прогрес користувача	1	1	0
Адаптивне тренування	1	1	0
Профіль користувача	1	1	0
Адміністративна панель	3	3	0
Розмежування доступу	1	1	0
Усього	**14**	**14**	**0**

Отже, тестування підтвердило працездатність основних модулів web-застосунку MemoryGame. Система коректно виконує основні сценарії користувача та адміністратора, забезпечує збереження результатів, підтримує адаптивне тренування і реалізує розмежування доступу відповідно до ролей.

5.7 Напрями подальшого розвитку системи

Розроблений web-застосунок MemoryGame реалізує основні функції, необхідні для вивчення англійської лексики через ігрову взаємодію з картками. Разом з тим система може бути розширена у майбутньому.

До можливих напрямів подальшого розвитку належать:

– додавання аудіовимови англійських слів

– підтримка інших іноземних мов;

– створення тематичних словникових наборів;

– реалізація системи досягнень і нагород;

– додавання рейтингу користувачів;

– розширення аналітики навчального прогресу;

– можливість імпорту словникових наборів;

– додавання режиму змагання між користувачами;

– створення мобільної версії або PWA;

– інтеграція з освітніми платформами.

Особливо перспективним є розширення адаптивного тренування. У майбутньому система може враховувати не лише кількість помилок, а й швидкість відповіді, давність останнього повторення, складність слова та індивідуальну динаміку користувача. Це дозволить зробити навчальний маршрут більш точним і персоналізованим.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено web-застосунок MemoryGame для вивчення англійської лексики за допомогою інтерактивної гри з картками. Розроблена система дозволяє користувачу проходити гру, знаходити пари між англійськими словами та українськими перекладами, зберігати результати проходження, переглядати навчальний прогрес і працювати зі словами, які були засвоєні гірше.

У роботі було проведено аналіз предметної галузі вивчення англійської лексики за допомогою інтерактивних ігрових засобів. Визначено, що традиційне механічне заучування слів має низку недоліків, зокрема низьку залученість користувача, відсутність регулярної перевірки засвоєння матеріалу та складність відстеження прогресу. Обґрунтовано доцільність використання ігрового підходу Memory Game, який поєднує навчання, повторення та активну взаємодію з картками [1; 3].

Було проаналізовано існуючі програмні засоби для вивчення англійської лексики, зокрема Duolingo, Quizlet, Memrise, Wordwall та Anki. У результаті аналізу визначено їхні переваги та недоліки. Встановлено, що більшість аналогів або орієнтовані на комплексне вивчення мови, або не мають достатньої спеціалізації на форматі Memory Game, або не забезпечують повноцінного адаптивного повторення слабких слів та адміністративного керування словниковою базою.

У роботі було сформульовано функціональні та нефункціональні вимоги до web-застосунку MemoryGame. До основних функціональних вимог віднесено реєстрацію та авторизацію користувача, вибір рівня складності, запуск гри, відкриття карток, перевірку правильності пар, збереження результатів, перегляд прогресу, адаптивне тренування, редагування профілю та адміністрування словникових карток. До нефункціональних вимог віднесено швидкодію, адаптивність інтерфейсу, підтримку основних браузерів, захист даних і можливість подальшого розширення системи.

Для реалізації web-застосунку було обґрунтовано вибір технологій. Серверну частину реалізовано на основі Java, Spring Boot, Spring Security, JWT, Spring Data JPA та Hibernate. Для зберігання даних використано PostgreSQL, а для керування змінами структури бази даних - Liquibase. Клієнтську частину реалізовано з використанням React, Vite, TypeScript/JavaScript, Axios та React Router. Такий набір технологій дозволив побудувати клієнт-серверну систему з чітким розподілом відповідальності між frontend, backend і базою даних [12].

У процесі проєктування було розроблено логічну структуру системи, діаграму варіантів використання, діаграми послідовності, структуру бази даних, архітектуру web-застосунку, діаграму класів і діаграму розгортання. База даних системи містить таблиці для збереження користувачів, словникових карток, результатів ігор, деталізації результатів по картках, персонального прогресу користувача та журналу дій адміністратора [24].

Під час програмної реалізації було створено основні модулі web-застосунку: модуль авторизації, ігровий модуль, модуль збереження результатів, модуль прогресу користувача, модуль адаптивного тренування, профіль користувача та адміністративну панель. Звичайний користувач може проходити гру, переглядати результати та працювати зі слабкими словами, а адміністратор може керувати словниковими картками, рівнями складності, користувачами та статистикою платформи.

Особливою функціональною можливістю розробленої системи є механізм адаптивного словникового маршруту. Він базується на аналізі кількості спроб, помилок, успішних повторень і рівня засвоєння кожного слова. Завдяки цьому користувач отримує для повторення саме ті слова, які потребують додаткового закріплення, що робить навчальний процес більш персоналізованим.

Також було проведено тестування web-застосунку MemoryGame. У ході тестування перевірено роботу авторизації, ігрового модуля, збереження результатів, перегляду прогресу, адаптивного тренування, профілю користувача, адміністративної панелі та розмежування доступу між ролями USER і ADMIN. Результати тестування підтвердили коректну роботу основних функцій системи.

Отже, мету кваліфікаційної роботи було досягнуто. Розроблений web-застосунок MemoryGame спрощує процес вивчення англійської лексики за рахунок інтерактивної роботи з картками, перевірки відповідей, збереження результатів і формування адаптивного словникового маршруту.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Яценко В. А., Задонцев Ю.В. Навчальна гра «Memory Game» для вивчення англійської мови. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.425-426

2. Яценко В. А., Задонцев Ю.В. Розробка навчальної гри «Memory Game» для вивчення англійських слів. Матеріали Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в ІКТ», 23 квітня 2026 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2026. С. 117-119.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шабуніна В. В. Гейміфікація у викладанні англійської мови як засіб підвищення мотивації здобувачів освіти. Закарпатські філологічні студії. 2025. Вип. 41. Ч. 2. URL: https://www.zfs-journal.uzhnu.uz.ua/archive/41/part_2/26.pdf.

2. Абсалямова Я. В. Використання онлайн-тестів і флеш-карт для розширення словникового запасу студентів з англійської мови. Педагогічна академія: наукові записки. 2025. URL: https://pedagogical-academy.com/index.php/journal/article/view/1620.

3. Полонська Т. К. Сутність ігрових технологій у навчанні іноземних мов учнів початкової школи на компетентнісних засадах. Проблеми сучасного підручника. 2018. Вип. 20. С. 317–327. URL: https://lib.iitta.gov.ua/710774/1/%21%21%21Polonska_Primary_School_Game.pdf.

4. Михайлова Л. Використання інтерактивних платформ під час викладання англійської мови студентам ЗВО. Актуальні питання гуманітарних наук. 2025. Вип. 84. Ч. 2. URL: https://aphn-journal.in.ua/archive/84_2025/part_2/44.pdf).

5. Duolingo: офіційний сайт платформи для вивчення іноземних мов. URL: https://www.duolingo.com).

6. Quizlet: офіційний сайт платформи для створення навчальних карток і тестів. URL: https://quizlet.com

7. Memrise: офіційний сайт платформи для вивчення іноземних мов. URL: https://www.memrise.com

8. Wordwall: офіційний сайт платформи для створення інтерактивних навчальних вправ. URL: https://wordwall.net).

9. Anki: офіційний сайт програмного засобу для створення електронних карток і повторення навчального матеріалу. URL: https://apps.ankiweb.net.

10. Java Documentation. Object-Oriented Programming Concepts. Oracle. URL: https://docs.oracle.com/javase/tutorial/java/concepts/index.html.

11. Spring Boot Reference Documentation. Spring. URL: https://docs.spring.io/spring-boot/.

12. Spring Security Reference Documentation. Spring. URL: https://docs.spring.io/spring-security/reference/.

13. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519. 2015. URL: https://www.rfc-editor.org/rfc/rfc7519.

14. Spring Data JPA Reference Documentation. Spring. URL: https://docs.spring.io/spring-data/jpa/reference/.

15. Hibernate ORM Documentation. URL: https://hibernate.org/orm/documentation/.

16. PostgreSQL Documentation. PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/.

17. Liquibase Documentation. URL: https://docs.liquibase.com/.

18. React Documentation. Meta Open Source. URL: https://react.dev/.

19. Vite Documentation. URL: https://vite.dev/guide/.

20. Axios Documentation. URL: https://axios-http.com/docs/intro .

21. React Router Documentation. URL: https://reactrouter.com/.

22. MDN Web Docs. HTTP overview. Mozilla. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview.

23. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures: doctoral dissertation. University of California, Irvine, 2000. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

24. Unified Modeling Language Specification. Object Management Group. URL: https://www.omg.org/spec/UML/.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка навчальної гри «Memory game» для вивчення англійських слів

Виконав студент 4 курсу
групи ПД-41
Яценко Володимир Анатолійович
Керівник роботи
доц., канд.техн.наук Задонцев Юрій Вікторович

Київ - 2026

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Визначити потребу в інтерактивному засобі для вивчення англійської лексики, який підвищує залученість користувача та спрощує закріплення нових слів.
2. Проаналізувати існуючі програмні засоби для вивчення англійських слів, роботи з картками та контролю прогресу користувача.
3. Визначити функціональні та нефункціональні вимоги.
4. Обґрунтувати вибір технологій для реалізації клієнтської частини, серверної частини та бази даних.
5. Спроєктувати логічну структуру системи, базу даних та UML-діаграми web-застосунку MemoryGame.
6. Реалізувати основні модулі системи: гру, профіль користувача, прогрес, адаптивне тренування та адміністративну панель.
7. Розробити механізм адаптивного словникового маршруту на основі помилок і результатів користувача.
8. Провести тестування web-застосунку MemoryGame та проаналізувати результати перевірки основних функцій.

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – спрощення вивчення англійської лексики за рахунок розробки веб-застосунку Memory Game з інтерактивними картками, перевіркою відповідей і фіксацією результатів.

Об'єкт дослідження – процес вивчення англійської лексики за допомогою інтерактивних ігрових засобів.

Предмет дослідження – програмне забезпечення для вивчення англійської лексики у форматі інтерактивної гри Memory Game.

АНАЛІЗ АНАЛОГІВ

Показник	Duolingo	Quizlet	Memrise	Wordwall	MemoryGame
Карткова модель	обмежено	+	обмежено	+	+
Ігрова механіка	+	окремі режими	окремі елементи	+	+
Вибір складності	+	через налаштування	+	через шаблони	+
Орієнтація на англійську лексику	мовний курс	різні предмети	мовний курс	різні теми	+
Простота інтерфейсу	+	потребує адаптації	потребує адаптації	потребує адаптації	+
Зручність моделі даних	складна	зручна	комбінована	шаблонна	оптимальна
Роль адміністратора	-	автор наборів	-	+	+
Керування картками	-	+	-	+	+
Повторення слабких слів	-	-	-	-	+
Журнал дій адміністратора	-	-	-	-	+

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги

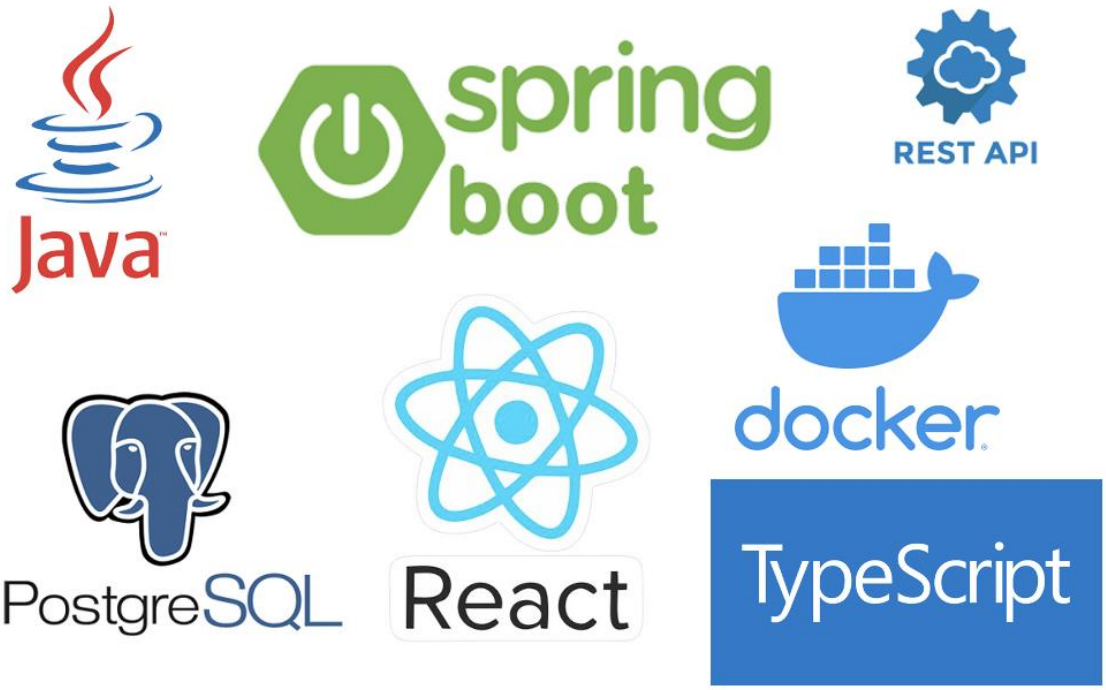
1.Реєстрація та авторизація користувача.
2.Вибір теми англійських слів.
3.Вибір рівня складності гри.
4.Запуск ігрового режиму Memory Game.
5.Відкриття карток та пошук правильних пар.
6.Автоматична перевірка правильності вибору.
7.Підрахунок результату проходження рівня.
8.Збереження прогресу користувача.
9.Перегляд статистики та результатів.
10.Повторне проходження рівня для закріплення слів.

Нефункціональні вимоги

1.Час завантаження основних сторінок — не більше 3 секунд.
2.Час відповіді сервера на запит — не більше 1 секунди.
3.Коректна робота в браузерах Google Chrome, Mozilla Firefox, Microsoft Edge.
4.Адаптивність інтерфейсу для екранів від 360 px до 1920 px.
5.Збереження результатів гри після кожного завершеного рівня.
6.Підтримка одночасної роботи не менше 50 користувачів.
7.Кількість помилок під час перевірки пар карток — не більше 1%.
8.Простота інтерфейсу: користувач має запускати гру не більше ніж за 3 дії.
9.Можливість подальшого розширення бази слів мінімум до 1000 навчальних одиниць.

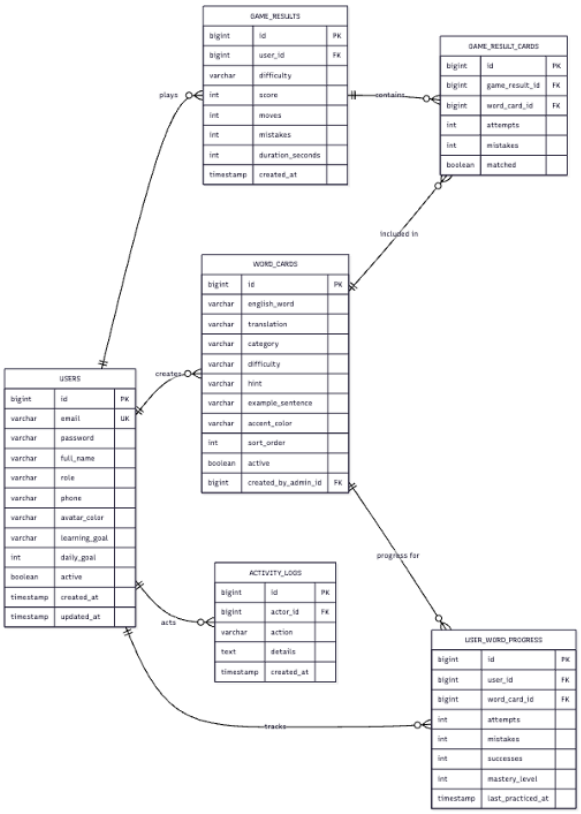
5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



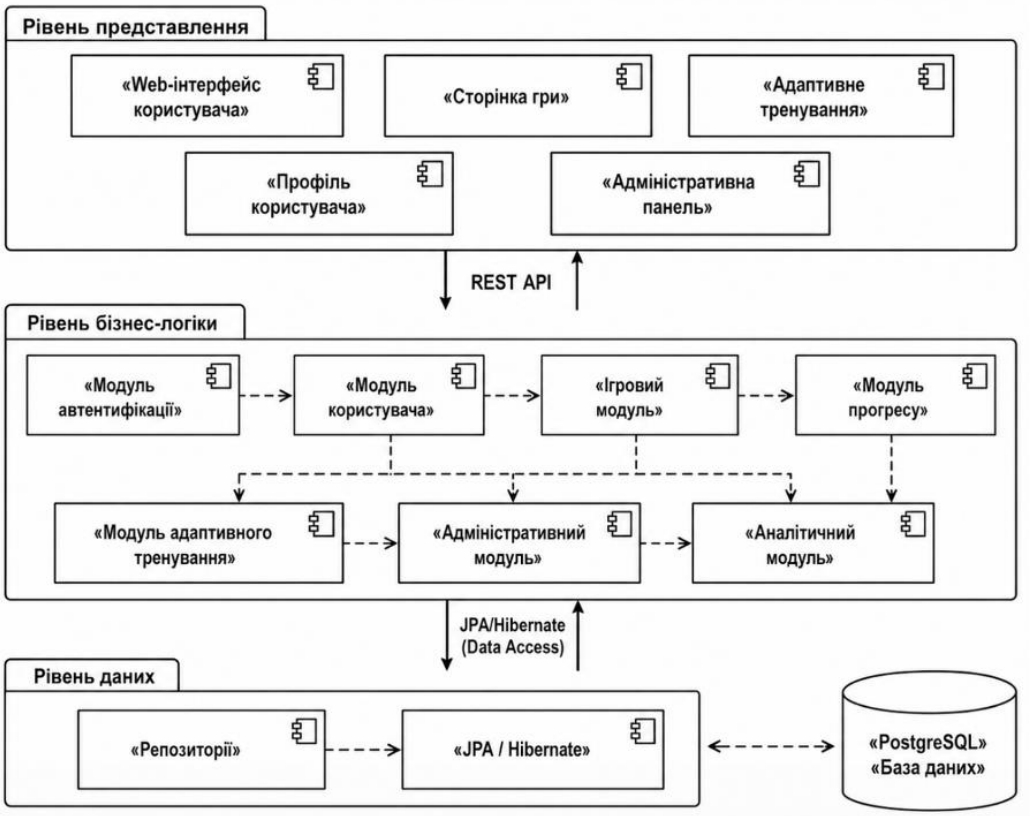
6

СТРУКТУРА БАЗИ ДАНИХ

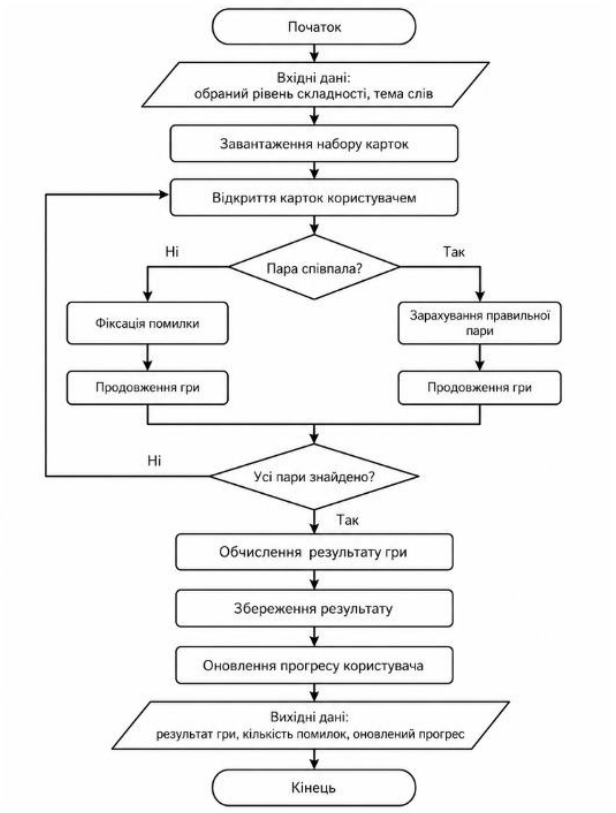


7

АРХІТЕКТУРА ВЕБ-ЗАСТОСУНКУ

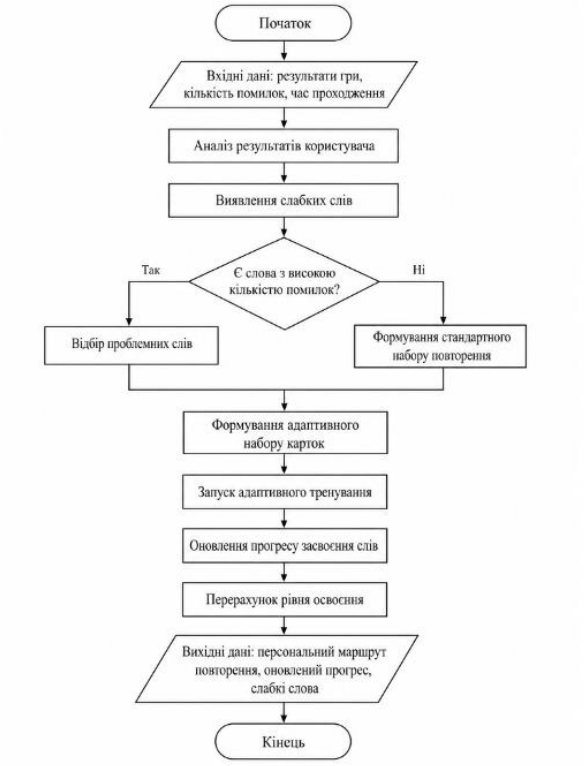


БЛОК-СХЕМА ПРОЦЕСУ ПРОХОДЖЕННЯ ГРИ КОРИСТУВАЧЕМ



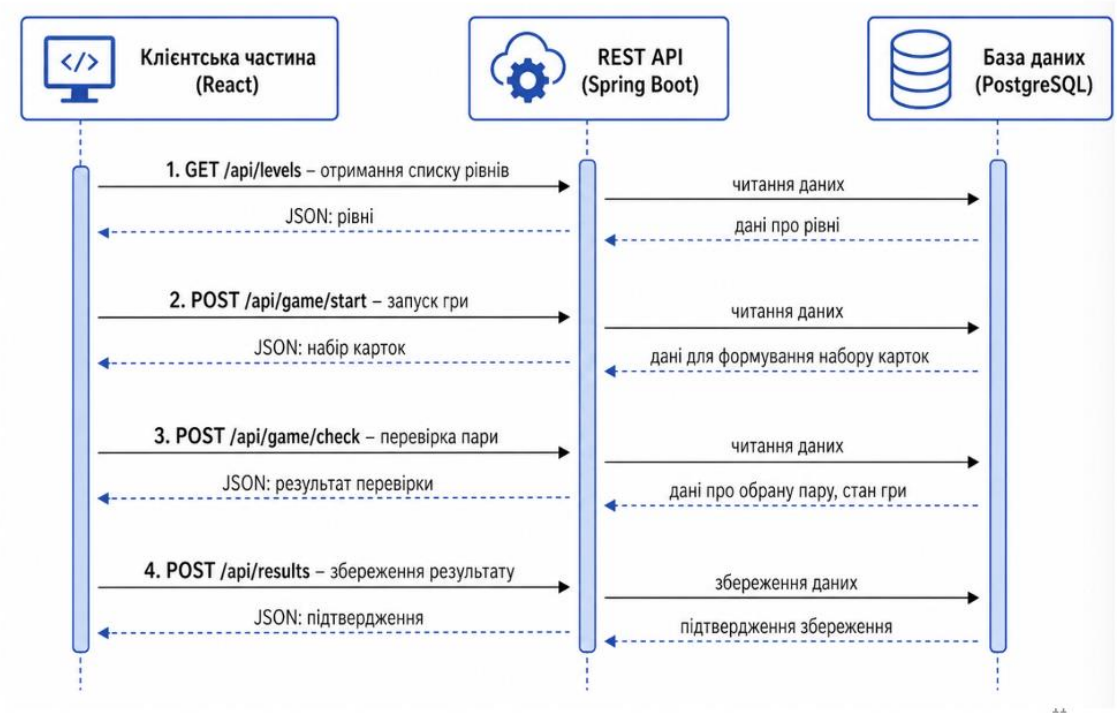
9

БЛОК-СХЕМА ФОРМУВАННЯ АДАПТИВНОГО СЛОВНИКОВОГО МАРШРУТУ

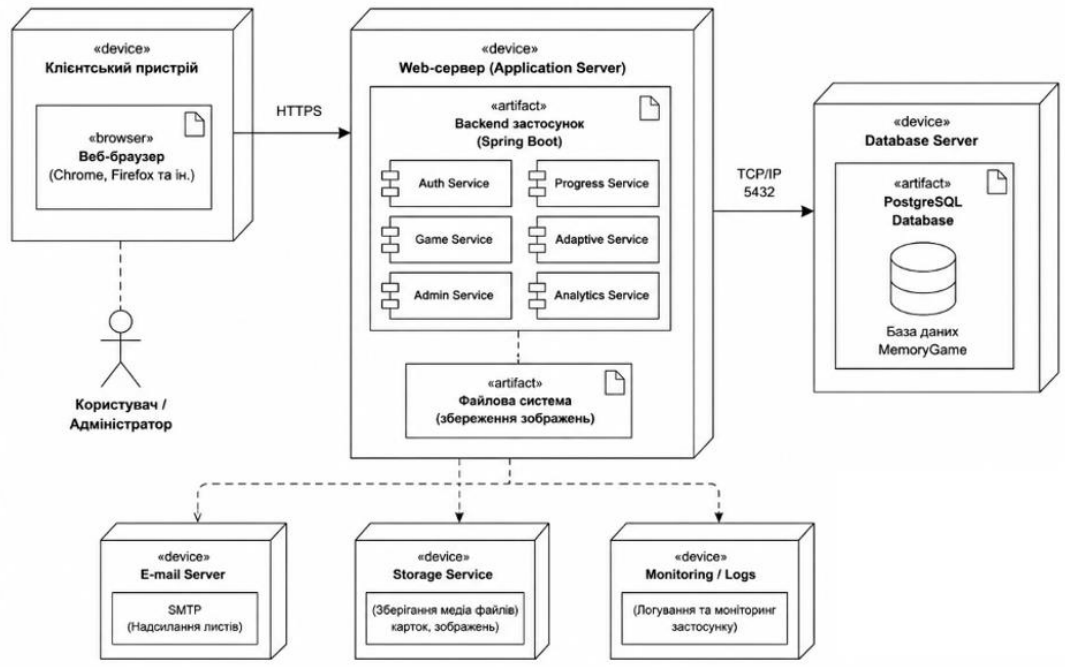


10

СХЕМА ОБМІНУ ДАНИМИ МІЖ КЛІЄНТСЬКОЮ ТА СЕРВЕРНОЮ ЧАСТИНОЮ



ДІАГРАМА РОЗГОРТАННЯ



ДІАГРАМА КЛАСІВ

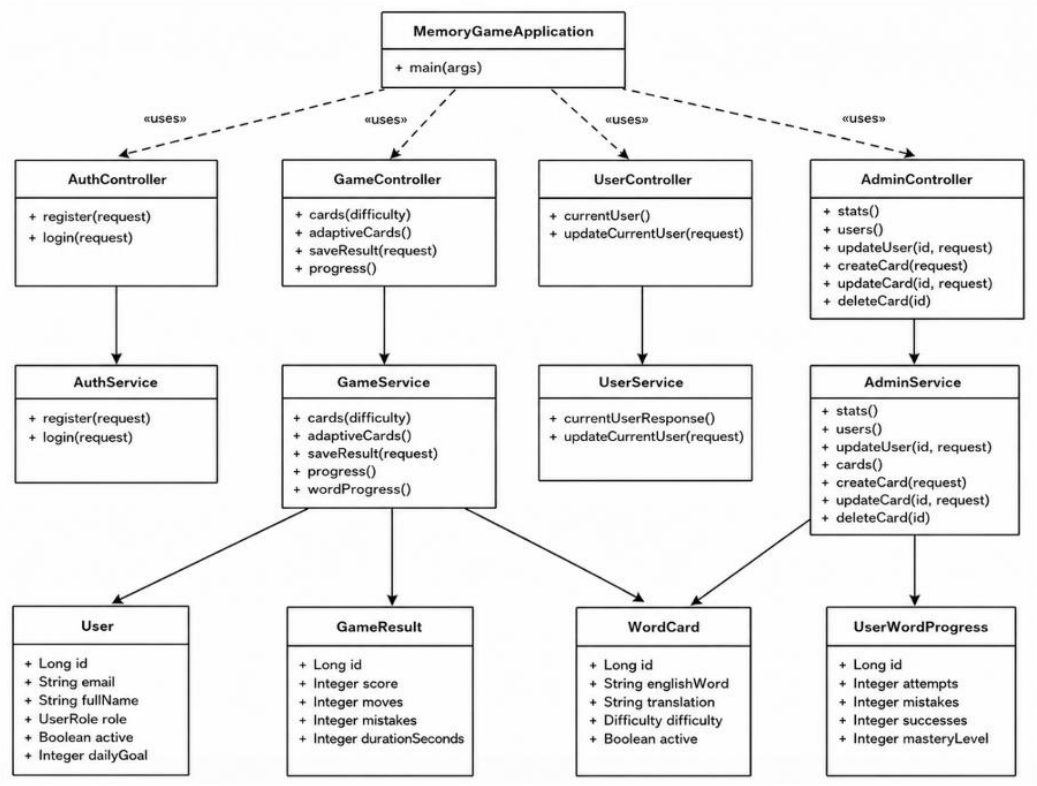
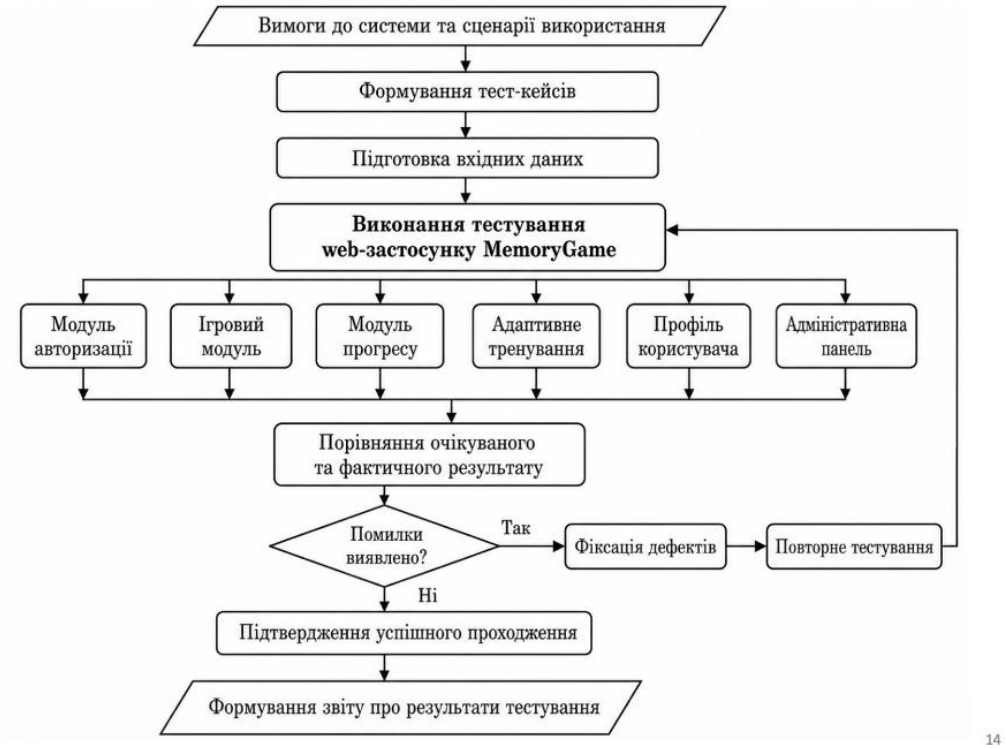
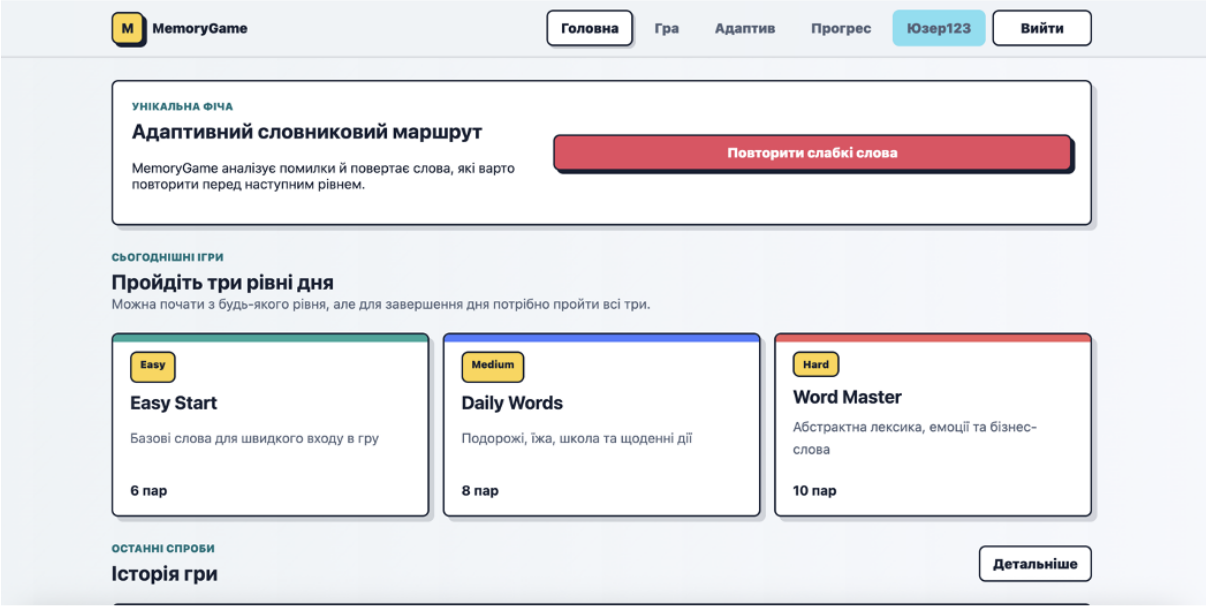


СХЕМА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ MEMORY GAME



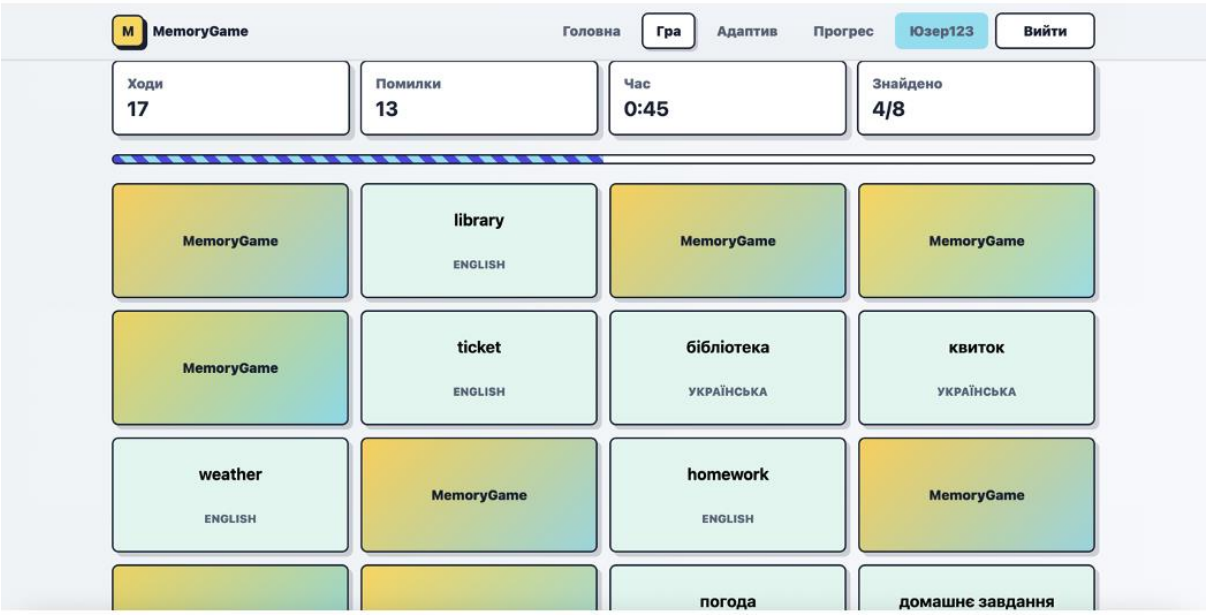
ЕКРАННІ ФОРМИ



Головна сторінка

15

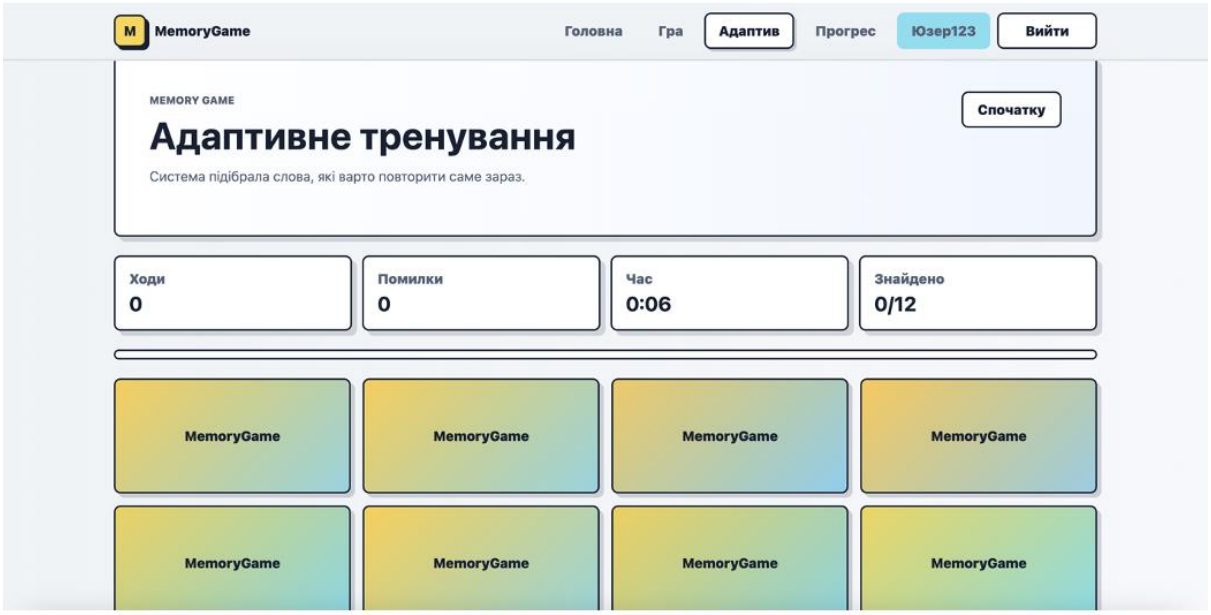
ЕКРАННІ ФОРМИ



Процес гри рівня «Середній»

16

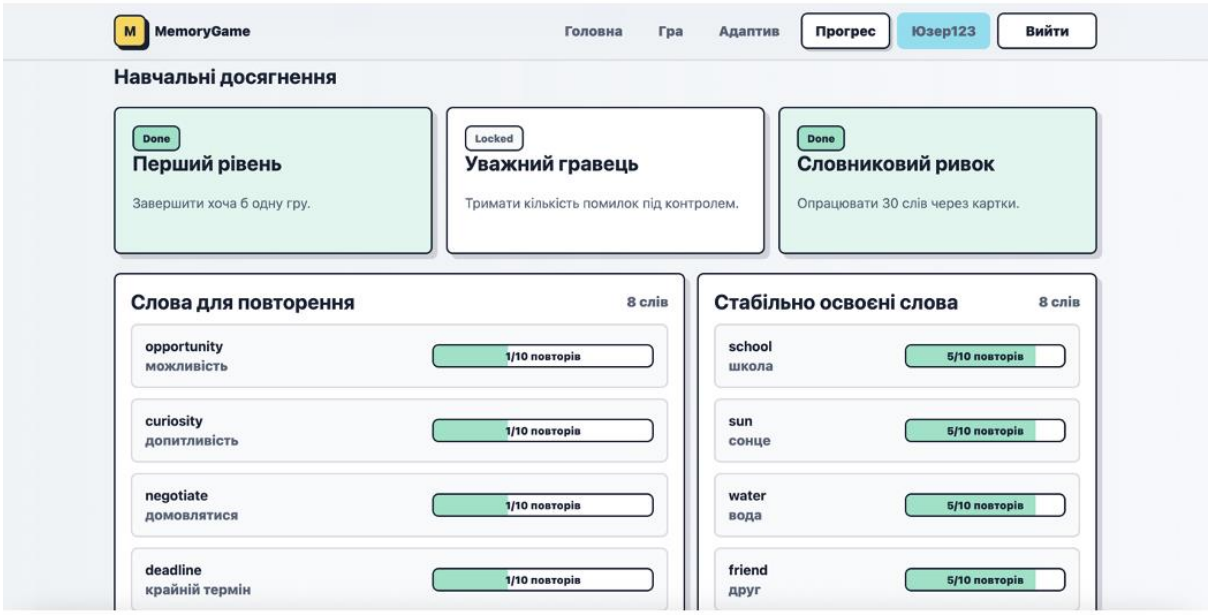
ЕКРАННІ ФОРМИ



Сторінка адаптивного тренування

17

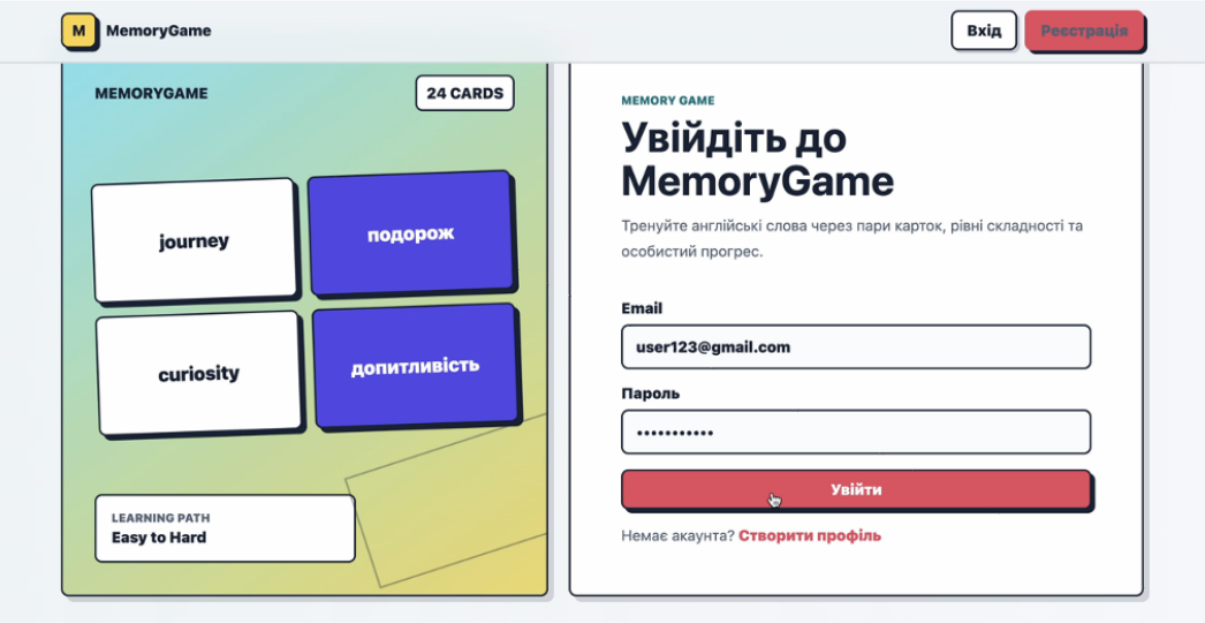
ЕКРАННІ ФОРМИ



Сторінка перегляду навчального прогресу

18

ДЕМОНСТРАЦІЯ РОБОТИ ПРОГРАМИ



19

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

8

Тези доповідей:

1. Яценко В.А,, Задонцев Ю.В. Навчальна гра «Memory Game» для вивчення англійської мови. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2025, С.425-426.

2. Яценко В.А,, Задонцев Ю.В.. Розробка навчальної гри «Memory game» для вивчення англійських слів. Матеріали Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в ІКТ», 24 квітня 2026 року, Київ, ДУІКТ. Збірник тез. *(подано до друку).*

ВИСНОВКИ

1. Проведено аналіз предметної галузі вивчення англійської лексики за допомогою інтерактивних ігрових засобів.
2. Проаналізовано існуючі програмні засоби: Duolingo, Quizlet, Memrise та Wordwall.
3. Визначено функціональні та нефункціональні вимоги до web-застосунку MemoryGame.
4. Обґрунтовано вибір технологій: React, TypeScript, Java, Spring Boot та PostgreSQL.
5. Спроєктовано логічну структуру системи, базу даних та основні UML-діаграми.
6. Реалізовано основні модулі: гру, профіль користувача, прогрес, адаптивне тренування та адміністративну панель.
7. Розроблено механізм адаптивного словникового маршруту на основі помилок користувача.
8. Проведено тестування web-застосунку та підтверджено коректну роботу основних функцій.

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

Листинг 1. GameController.java

```
@RestController
@RequestMapping("/api")
@RequiredArgsConstructor
public class GameController {

  private final GameService gameService;

  @GetMapping("/levels")
  public List<LevelResponse> levels() {
    return gameService.levels();
  }

  @GetMapping("/cards")
  public List<WordCardResponse> cards(
      @RequestParam(defaultValue = "EASY")
Difficulty difficulty) {
    return gameService.cards(difficulty);
  }

  @GetMapping("/adaptive/cards")
  public List<WordCardResponse>
adaptiveCards() {
    return gameService.adaptiveCards();
  }

  @PostMapping("/results")
  public GameResultResponse saveResult(
      @Valid @RequestBody
GameResultRequest request) {
    return gameService.saveResult(request);
  }

  @GetMapping({"/progress", "/dashboard"})
  public ProgressResponse progress() {
    return gameService.progress();
  }

  @GetMapping("/progress/words")
  public List<WordProgressResponse>
wordProgress() {
    return gameService.wordProgress();
  }
}
```

Листинг 2. AdminController.java

```
@RestController
@RequestMapping("/api/admin")
@RequiredArgsConstructor
@PreAuthorize("hasRole('ADMIN')")
public class AdminController {
  private final AdminService adminService;

  @GetMapping("/stats")
  public AdminStatsResponse stats() {
    return adminService.stats();
  }

  @GetMapping("/users")
  public List<UserResponse> users() {
    return adminService.users();
  }

  @PutMapping("/users/{id}")
  public UserResponse updateUser(
      @PathVariable Long id,
      @RequestBody AdminUserUpdateRequest
request) {
    return adminService.updateUser(id, request);
  }

  @GetMapping("/cards")
  public List<WordCardResponse> cards() {
    return adminService.cards();
  }

  @PostMapping("/cards")
  public WordCardResponse createCard(
      @Valid @RequestBody WordCardRequest
request) {
    return adminService.createCard(request);
  }

  @PutMapping("/cards/{id}")
  public WordCardResponse updateCard(
      @PathVariable Long id,
      @Valid @RequestBody WordCardRequest
request) {
    return adminService.updateCard(id, request);
  }

  @DeleteMapping("/cards/{id}")
  public void deleteCard(@PathVariable Long id) {
    adminService.deleteCard(id);
  }
}
```

Листинг 3. User.java

```
@Entity
@Table(name = "users")
@Getter
@Setter
```

```
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false, unique = true)
    private String email;

    @Column(name = "password_hash", nullable = false)
    private String passwordHash;

    @Column(name = "full_name")
    private String fullName;

    private String phone;

    @Column(name = "avatar_color")
    @Builder.Default
    private String avatarColor = "#ffd447";

    @Column(name = "learning_goal")
    private String learningGoal;

    @Column(name = "daily_goal")
    @Builder.Default
    private Integer dailyGoal = 12;

    @Builder.Default
    private Boolean active = true;

    @Enumerated(EnumType.STRING)
    @Builder.Default
    private UserRole role = UserRole.USER;

    @CreationTimestamp
    private LocalDateTime createdAt;

    @UpdateTimestamp
    private LocalDateTime updatedAt;
}
```

Листинг 4. WordCard.java

```
@Entity
@Table(name = "word_cards")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class WordCard {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "english_word", nullable = false)
    private String englishWord;

    @Column(nullable = false)
    private String translation;

    private String category;

    @Enumerated(EnumType.STRING)
    private Difficulty difficulty;

    private String hint;

    @Column(name = "example_sentence")
    private String exampleSentence;

    @Column(name = "accent_color")
    private String accentColor;

    @Builder.Default
    private Boolean active = true;

    @Column(name = "sort_order")
    @Builder.Default
    private Integer sortOrder = 0;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "created_by_admin_id")
    private User createdByAdmin;

    @CreationTimestamp
    private LocalDateTime createdAt;
}
```

Листинг 5. Основная логика адаптивного режима

```
@Transactional(readOnly = true)
public List<WordCardResponse> adaptiveCards() {
    User user = userService.currentUser();
    Map<Long, WordCard> selected = new LinkedHashMap<>();

    userWordProgressRepository
.findTop12ByUserOrderByMasteryLevelAscMistakesDescLastPracticedAtAsc(user)
        .stream()
        .map(UserWordProgress::getWordCard)
        .filter(card -> Boolean.TRUE.equals(card.getActive()))
```

```
    .forEach(card -> selected.putIfAbsent(card.getId(), card));

    for (WordCard card : wordCardRepository
.findByActiveTrueOrderByDifficultyAscSortOrderAscEnglishWordAsc()) {
    if (selected.size() >= 10) {
    break;
    }
    selected.putIfAbsent(card.getId(), card);
    }

    return selected.values()
    .stream()
    .map(this::toCardResponse)
    .toList();
    }
```

Листинг 6. Збереження результату гри

```
@Transactional
public GameResultResponse saveResult(GameResultRequest request) {
    User user = userService.currentUser();
    int score = calculateScore(request);

    GameResult result = gameResultRepository.save(GameResult.builder()
    .user(user)
    .difficulty(request.getDifficulty())
    .moves(request.getMoves())
    .mistakes(request.getMistakes())
    .matchedPairs(request.getMatchedPairs())
    .durationSeconds(request.getDurationSeconds())
    .dayNumber(request.getDayNumber() == null ? 1 : request.getDayNumber())
    .score(score)
    .build());

    saveCardDetailsAndProgress(user, result, request);

    return toResultResponse(result);
    }
```

Листинг 7. Оновлення прогресу слова

```
public void updateWordProgress(
    User user,
    WordCard word,
    int attempts,
    int mistakes,
    boolean matched) {

    UserWordProgress progress = userWordProgressRepository
    .findByUserAndWordCard(user, word)
    .orElseGet(() -> UserWordProgress.builder()
    .user(user)
    .wordCard(word)
    .build());

    progress.setAttempts(progress.getAttempts() + attempts);
    progress.setMistakes(progress.getMistakes() + mistakes);
    progress.setSuccesses(progress.getSuccesses()
+ (matched ? 1 : 0));

    int mastery = Math.max(0, Math.min(100,
    progress.getSuccesses() * 18
    - progress.getMistakes() * 10
    + progress.getAttempts() * 3));

    progress.setMasteryLevel(mastery);

    progress.setLastPracticedAt(LocalDateTime.now());

    userWordProgressRepository.save(progress);
    }
```

Листинг 8. React-маршрути App.jsx

```
    export default function App() {
    return (
<AuthProvider>
<BrowserRouter>
<Navbar />

<main className="app-shell">
<Routes>
<Route path="/" element={<Navigate to="/dashboard" replace />} />
<Route path="/login" element={<Login />} />
<Route path="/register" element={<Register />} />

<Route path="/dashboard" element={
<ProtectedRoute><Dashboard /></ProtectedRoute>
    } />

<Route path="/game" element={
<ProtectedRoute><Game /></ProtectedRoute>
    } />

<Route path="/adaptive" element={
<ProtectedRoute><Game adaptive /></ProtectedRoute>
    } />

<Route path="/progress" element={
<ProtectedRoute><Progress /></ProtectedRoute>
```

```
    } />

<Route path="/profile" element={
<ProtectedRoute><Profile /></ProtectedRoute>
    } />

<Route path="/admin" element={
<ProtectedRoute><Admin /></ProtectedRoute>
    } />
</Routes>
</main>
</BrowserRouter>
</AuthProvider>
    )
    }
```

Листинг 9. API для гри на frontend

```
    import api from './client'

    export async function fetchLevels() {
    const { data } = await api.get('/levels')
    return data
    }

    export async function fetchCards(difficulty) {
    const { data } = await api.get('/cards', {
    params: { difficulty },
    })
    return data
    }

    export async function fetchAdaptiveCards() {
    const { data } = await api.get('/adaptive/cards')
    return data
    }

    export async function
saveGameResult(payload) {
    const { data } = await api.post('/results',
payload)
    return data
    }

    export async function fetchProgress() {
    const { data } = await api.get('/progress')
    return data
    }

    export async function fetchWordProgress() {
    const { data } = await api.get('/progress/words')
    return data
    }
```

Листинг 10. API для адміністратора на frontend

```
    import api from './client'

    export async function fetchAdminStats() {
    const { data } = await api.get('/admin/stats')
    return data
    }

    export async function fetchAdminUsers() {
    const { data } = await api.get('/admin/users')
    return data
    }

    export async function updateAdminUser(id,
payload) {
    const { data } = await
api.put(`/admin/users/${id}`, payload)
    return data
    }

    export async function fetchAdminCards() {
    const { data } = await api.get('/admin/cards')
    return data
    }

    export async function
createAdminCard(payload) {
    const { data } = await api.post('/admin/cards',
payload)
    return data
    }

    export async function updateAdminCard(id,
payload) {
    const { data } = await
api.put(`/admin/cards/${id}`, payload)
    return data
    }

    export async function deleteAdminCard(id) {
    await api.delete(`/admin/cards/${id}`)
    }
```