Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та

комп'ютерних технологій і дизайну

(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних технологій

(повна назва кафедри (предметної, циклової комісії)

Пояснювальна записка

до дипломної роботи

другий (магістерський)

(рівень вищої освіти)

на тему: "Веб-система для вивчення граматики та лексики іноземних мов на прикладі англійської"

Виконав: студент 6 курсу групи КН-61м

спеціальності

122 "Комп'ютерні науки"

(шифр і назва напряму підготовки, спеціальності)

Юрчук О.Ю.

Керівник к.т.н, доцент кафедри Левкович М.В.

(прізвище та ініціали)

Рецензент

(прізвище та ініціали)

Львів – 2022

Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну
Кафедра інформаційних технологій
Рівень вищої освіти другий (магістерський)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Крошний І. М.
"____" _____20___ року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Юрчук Олег Юрійович
(прізвище, ім'я, по батькові)

1. Тема роботи *Веб-система для вивчення граматики та лексики іноземних мов на прикладі англійської*

керівник роботи *к.т.н, доцент кафедри Левкович М.В.*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "__"_____ 20__ року № _____
2. Термін подання студентом роботи
3. Вихідні дані до роботи *літературні джерела*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1. Опис області веб-систем для вивчення іноземних мов
2. Постановка задачі розробки веб-системи для граматики та лексики іноземних мов на прикладі англійської
3. Архітектура та проектування веб-системи
4. Реалізація та тестування веб-системи
5. Економічна частина
6. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Діаграма розгортання, діаграма прецедентів

6. Дата видачі завдання 09.09.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження продуктів-аналогів та предметної області вибраної теми роботи	09.09.2022 - 25.09.2022	
2	Розроблення вимог до ПЗ та їх письмове оформлення в специфікації вимог до нього	29.09.2022 - 11.10.2022	
3	Вибір стратегії розробки, основних технологій та шаблонів, на основі яких буде створюватись програмне забезпечення	14.10.2022 - 30.10.2022	
4	Розроблення програмного забезпечення	01.11.2022 - 23.11.2022	
5	Тестування програмного забезпечення	23.11.2022 - 28.11.2022	
6	Оформлення пояснювальної записки	30.11.2022 - 05.12.2022	

Студент _____ Юрчук О.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Левкович М.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Темою даної магістерської кваліфікаційної роботи є веб-система для вивчення граматики та лексики іноземних мов на прикладі англійської. Дана тема обрана у зв'язку з потребою ринку в даному програмному забезпеченні, так як включає у себе велику кількість класів користувачів та необхідність застосування сучасних технологій у процесах вивчення іноземних мов.

Тому метою роботи стало розроблення затребуваної ринком веб-системи для вивчення іноземних мов, а саме процесів вивчення граматики та лексики, як таких, які можуть бути повністю автоматизованими. Загалом робота включає проектування, реалізацію та тестування платформо-незалежної веб-системи у вигляді веб-серверу; пояснювальну записку обсягом 66 сторінок; додатки з лістингами коду, описами тестів та інструкцією користувача.

Веб-система дозволяє користувачу переглядати заняття створені іншими користувачами, проходити тести до них, створювати свої заняття та тести, вести власний словник, а також тренувати слова зі словника. Таким чином система поділена на два модуля: граматика та лексика. Усі функції пов'язані з заняття відносяться до модулю граматики, а функції пов'язані зі словником та його тренування до модулю лексики. Модуль граматики дозволяє знаходити новий матеріал для підвищення рівня граматики іноземної мови, а також реалізовувати вже набуті знання та творчий потенціал шляхом створення власних уроків та тестів для них, а модуль лексики дозволяє ввести налагоджений та ефективний процес підтримки та підвищення рівня словникового запасу користувача використовуючи віртуальний словник.

Система була успішно описана, спроектована, реалізована і протестована. Також була опрацьована економічна частина, в якій було підраховано та проаналізовано необхідні ресурси та затрати для розроблення програмного продукту, а також доведена його економічна вигода і доцільність.

Ключові слова: Java, Spring, Thymeleaf, Граматика, Лексика, Автоматизована система, Code First, клієнт-сервер, Java Persistence API

ABSTRACT

The topic of this magister’s thesis is a web-system for learning grammar and vocabulary of foreign languages on the example of English. This topic was chosen due to the market's need for this software, as it includes a large number of user classes and the need to use modern technologies in foreign language learning processes.

Therefore, the aim of the work was to develop a market-demanded web system for learning foreign languages, namely the processes of learning grammar and vocabulary as such, which can be fully automated. Such a system would facilitate the learning and learning of foreign languages for both teachers and students.

In general, the work includes the design, implementation and testing of a platform-independent web system in the form of a web server; an explanatory note of 66 pages; applications with code listings, test descriptions and user instructions.

The web system allows the user to view lessons created by other users, take tests for them, create their own lessons and tests, maintain their own dictionary, as well as practice words from the dictionary. Thus, the system is divided into two modules: grammar and vocabulary. All lessons related to the lessons are related to the grammar module, and the functions related to the vocabulary and its training are related to the vocabulary module. The grammar module allows you to find new material to improve the level of grammar of a foreign language, as well as implement the acquired knowledge and creativity by creating your own lessons and tests for them, and the vocabulary module allows you to introduce a well-established and effective process of maintaining the dictionary.

The system has been successfully described, designed, implemented and tested. The economic part was also studied, which calculated and analyzed the necessary resources and costs for the development of the software product, as well as proved its economic benefits and feasibility.

Keywords:

Java, Spring, Thymeleaf, Grammar, Vocabulary, Automated system, Code First, client-server, Java Persistence API.

ТЕХНІЧНЕ ЗАВДАННЯ

Необхідно розробити веб-систему для вивчення іноземних мов на прикладі англійської, а саме:

1. Провести аналіз ринку та визначити основні особливості веб-систем для вивчення іноземних мов;
2. Розробити вимоги до веб-системи та їх письмово оформити їх в специфікації вимог;
3. Вибрати стратегію розробки, основні технологій та шаблони, на основі яких буде створюватись веб-система;
4. Спроектувати, реалізувати та протестувати веб-систему.

Зміст

ВСТУП

На сьогоднішній час з розвитком комп’ютерних технологій та мереж все більше людей вивчають іноземні мови, так як інформація, яка може міститися на іншому куті світу в інших країнах, є легкодоступною.

Такий розвиток сприяє все більшому бажанню людей набувати знання та навички у певних сферах через швидкий доступ до цих знань. Тому факт того, що багато людей сьогодні віддають перевагу автоматизованим системам навчання на заміну традиційним, не є чимось неочікуваним. А застосування мереж та, в особливості, мережі Інтернет в автоматизованих системах робить процес набуття навичок не залежить від місцезнаходження користувача системи і відкриває багато нових можливостей для значних покращень та удосконалень цього процесу. Це може бути корисним у сфері вивчення іноземних мов, так як це відкриває кордони до іноземних країн та онлайн-комунікації з ними.

Тому застосування автоматизованих навчальних систем, які працюють через мережу Інтернет, у сфері вивчення іноземних мов є очевидним наступним кроком в процесі розвитку даної сфери. Також на даний момент для вчителів іноземних мов, незалежно від їхнього місця зайнятості, застосування автоматизованих систем з доступом в Інтернет може зробити їх роботу набагато легшою та ефективнішою. Адже автоматизовані системи зменшують або зовсім прибирають необхідність використання фізичних об’єктів для збереження інформації та подальшого контролю за цими об’єктами.

Певною мірою автоматизується процес перевірки робіт і унеможливлюється фактор помилки при цій перевірці. Також зникає загроза неможливості проведення процесу навчання при ситуаціях, які потребують віддаленого навчання, наприклад, карантинні заходи. Таким чином з’являється необхідність у програмному продукту, який би вирішував вище згадані проблеми, а саме у певній веб-орієнтованій автоматизованій системі навчання для вивчення іноземних мов.

РОЗДІЛ 1. ОПИС ОБЛАСТІ ВЕБ-СИСТЕМ ДЛЯ ВИВЧЕННЯ ІНОЗЕМНИХ МОВ

1.1. Опис предметної області

Процес вивчення іноземної мови являє собою набуття людиною навичок володіння невідомої їй мови за допомогою вже їй відомої. Тобто у цьому процесі можна виділити дві змінні: мова, яку вивчають, та мова, якою подано матеріал і яка є посиланням для порівняння. Найчастіше виділяють 6 навичок володіння конкретною мовою:

- граматика;
- лексика;
- говоріння;
- аудіювання;
- читання;
- письмо.

Ці навички надалі використовують як теми при вивченні іноземної мови.

Існує безліч методики вивчення іноземних мов. І не дивлячись на те, що вивчення іноземної мови є комплексним процесом і найчастіше вимагає вчителя або іншої досвідченої особи, певні під процеси можуть бути автоматизовані. Найкраще для цього підходять теми граматика та лексика.

Граматика – розділ мовознавства, який вивчає граматичну будову мови[1]. Це правила мови, що регулюють звуки, слова, речення та інші елементи, а також їх поєднання та тлумачення. Слово граматика також означає вивчення цих абстрактних ознак. У обмеженому розумінні цей термін стосується лише вивчення будови речень та слів (синтаксису та морфології), виключаючи лексику та вимову.

Лексика – сукупність слів тієї чи іншої мови або частини мови. Лексика є центральною частиною мови, що іменує, формує і передає знання про будь-які об'єкти, явища. Вивченням лексики займається наука лексикологія[2].

1.2. Поняття про автоматизовані навчальні системи

Автоматизована навчальна система – комплекс програм, направлених на вивчення, контроль та оцінку певної навчальної області[3]. Основними перевагами автоматизованих систем над традиційними методами навчання є: миттєвий зворотній зв'язок, швидка навігація та пошук матеріалу, використання сучасних можливостей для динамічного і інтерактивного моделювання та демонстрації матеріалу. Також більшість навчальних систем дозволяє миттєво отримати оцінку знань у вибраній області. Щодо недоліків автоматизованих навчальних систем, то можна виділити високу вартість кінцевого продукту.

Одними з найважливіших критеріїв оцінки автоматизованих навчальних систем є зручність інтерфейсу, що є вкрай важливим у таких типах систем, та повнота розкриття можливостей навчання вибраної області, тобто наскільки дана система дає можливість ознайомитися і вивчити поданий матеріал. Але відповіді на те, якою має бути структура ідеальної навчальної системи різняться. Найчастіше від навчальної системи вимагають 3 основних режимах роботи:

- Навчання без перевірки (перегляд теорії у вигляді тексту, зображень, відео або аудіо)
- Навчання з перевіркою (в кінці певного структурного блоку пропонується пройти тест для перевірки здобутих знань)
- Тестовий контроль (це тестові завдання для підсумкового контролю знань)

Тобто автоматизовані навчальні системи дозволяють не тільки вивчити теоретичний матеріал, але й сформувати практичні навички у вирішенні певних задач.

Тестування є одним з найпопулярнішим методом оцінки знань у наш час у педагогіці і використання нових технологій дає можливість якісно вирішити цю проблему, підвищуючи ефективність навчання.

Автоматизовані навчальні системи сьогодні найчастіше зустрічаються у вигляді прикладних програмних систем, веб-систем або мобільних додатків. Умовно їх можна розділити на дві групи: з доступом до Інтернету та без нього.

Доступ до Інтернету набагато розширює можливості навчальної системи додаючи можливість віддаленої взаємодії вчитель-учень, вчитель-вчитель і учень-учень. Також доступ до Інтернету відкриває для навчальної системи можливість бути платформою для авторів навчальних матеріалів.

1.3. Аналіз існуючих рішень

Серед існуючих рішень можна виділити наступні[4,5]:

LinguaLeo - це безкоштовна онлайн-платформа, що пропонує послугу з вивчення англійської мови для носіїв російської, португальської та бразильської мов. LinguaLeo персоналізує навчальну програму кожного користувача, щоб зробити вивчення англійської мови більш ефективним.

По-перше, LinguaLeo пропонує користувачам тест на розміщення, щоб визначити рівень їхньої мовної майстерності. Потім служба розробляє програму особистого навчання, яка враховує навички, цілі та уподобання користувача.

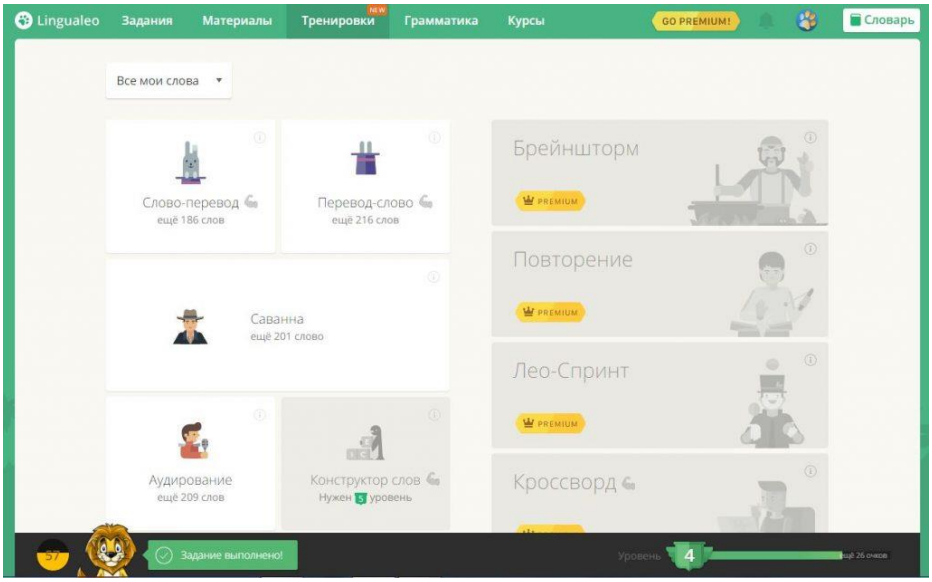

Рис. 1.1. Інтерфейс веб-сайту LinguaLeo.

Переваги сервісу:

- велика кількість вправ;

- можливість вивчати граматику;
- велика кількість статей та навчальних відео.

Недоліки сервісу:

- частина функціоналу є платною;
- можливість вивчати тільки англійську мову.

Duolingo - це веб-сайт та мобільний додаток для вивчення іноземних мов. Компанія використовує модель freemium: додаток та веб-сайт доступні безкоштовно, хоча Duolingo також пропонує преміум-послугу за окрему плату. Особливістю ресурсу є можливість вивчати більш ніж 30 мов, використовуючи при цьому більш ніж 10 мов для вивчення. Duolingo імітує структуру відеоігор кількома способами, щоб залучити своїх користувачів.

Наявна система винагород, в якій користувачі отримують ігрову валюту, яку вони можуть витратити на такі функції, як налаштування персонажа або рівні бонусів (обидві доступні лише в мобільному додатку).

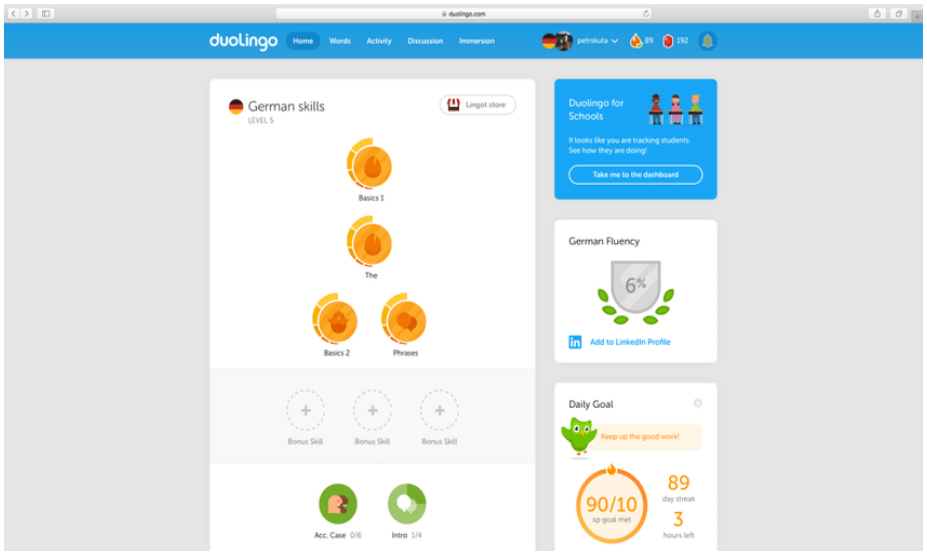

Рис. 1.2. Інтерфейс веб-сайту Duolingo.

Переваги сервісу:

- безкоштовний;
- зрозуміла структура уроків;
- можливість підібрати рівень та пропустити певний матеріал;

- велика кількість мов.

Недоліки сервісу:

- курси певних мов мають невелику кількість матеріалу.

Busuu - це платформа для вивчення мови, яка дозволяє користувачам взаємодіяти з носіями мови. Доступна у вигляді веб-сайту та мобільного додатку. Користувачі працюють на самостійних уроках однієї або декількох мов курсу. Уроки включають вивчення лексики та граматики. Наприкінці кожного уроку користувачі можуть потренуватися з носіями мови, яку вони вивчають, в письмовому або розмовному вигляді.

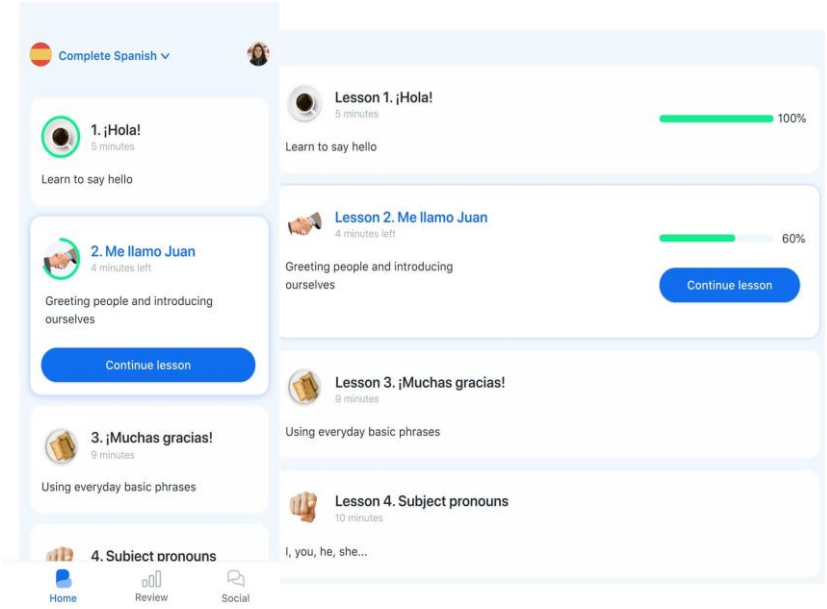

Рис. 1.3. Інтерфейс веб-сайту Busuu.

Переваги сервісу:

- наявність розмовних уроків;
- присутні деталі соціальної мережі;
- можливість створення особистого навчального плану.

Недоліки сервісу:

- курси різняться по якості в залежності від мови;
- відсутність розважальної складової.

Memrise – це веб-сайт та додаток для вивчення лексики іноземних мов. Сервіс працює за системою флеш-карт. У класичній формі це аркуш паперу, на

якому на одній стороні написано слово чи фраза на вашій мові, а на іншій - переклад. В основному сервіс зосереджений на вивченні мов. Тим не менш, ви також можете використовувати програму для запам’ятовування та відпрацювання термінів з інших предметів та сфер. Memrise вкорінює незнайомі слова та фрази у ваш мозок, використовуючи мнемотехніку та повторювані інтервали, які вони показують, висаджуючи квіти. Підводячи підсумок, це інструмент збільшення словникового запасу, і Memrise вдосконалює цей процес за допомогою сучасних технологій.

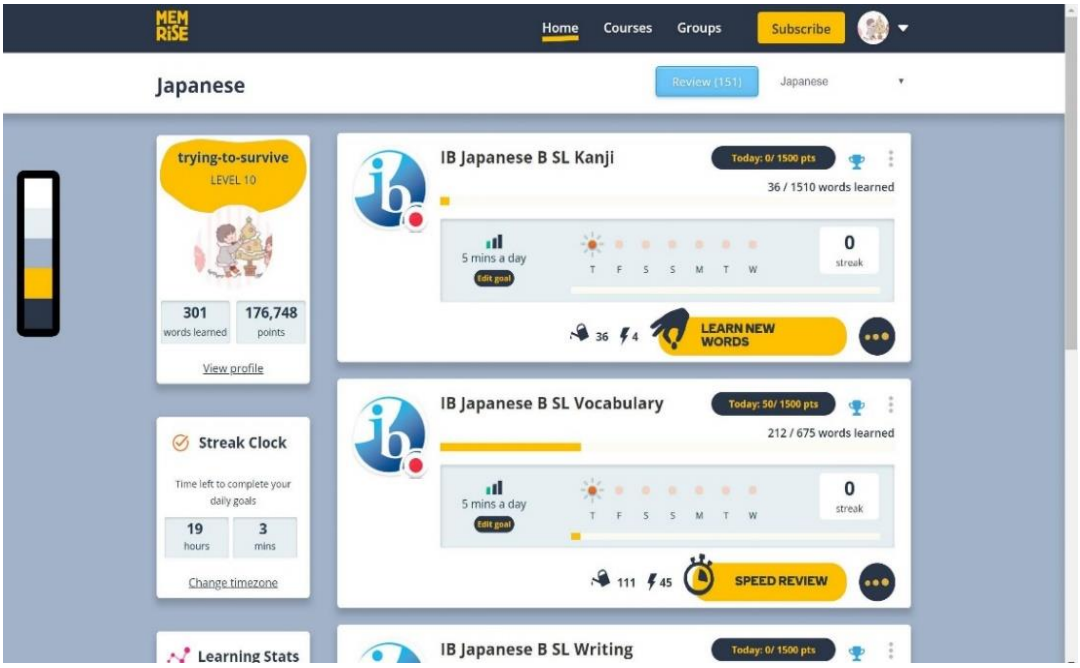

Рис. 1.4. Інтерфейс веб-сайту Memrise.

Переваги сервісу:

- велика кількість матеріалу;
- багато можливостей кастомізації в налаштуваннях;
- можливість створення контенту користувачами.

Недоліки сервісу:

- немає гарантій щодо якості вмісту, створеного користувачами;
- можливість вивчати тільки лексику.

Висновки до розділу 1

В результаті аналізу області вивчення іноземних мов, можна зробити висновок, що вивчення іноземних мов є комплексним процесом і певні під процеси можуть бути автоматизовані за допомогою комп’ютерних технологій, що зробить цей процес набагато ефективнішим та продуктивнішим. А використання веб-систем у мережі інтернет відкриє ще більше можливостей для взаємодії всім учасникам процесу.

Дослідивши існуючи рішення в даній області було визначено, що наявні веб-системи найчастіше концентруються на вузьких темах вивчення іноземних мов, таких, як граматика, лексика, пряма комунікація тощо. Також всі сервіси є платними або частково платними.

РОЗДІЛ 2. ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ ВЕБ-СИСТЕМИ ДЛЯ ВИВЧЕННЯ ГРАМАТИКИ ТА ЛЕКСИКИ ІНОЗЕМНИХ МОВ НА ПРИКЛАДІ АНГЛІЙСЬКОЇ

2.1. Загальна постановка задачі

Задача полягає у розробці веб-системи для вивчення лексики та граматики іноземних мов. Тобто користувач має мати доступ до двох модулів системи – лексики та граматики. Користувач повинен зареєструватися для збереження закріплених за ним даних.

У модулі для вивчення граматики користувач повинен мати доступ до вибору уроку для перегляду теоретичного матеріалу уроку та подальшого тестування з метою контроля набутих знань. У кожного уроку є певна складність. Також користувач повинен мати можливість перегляду результатів минулих тестувань. Під час самого тестування користувач має вибрати один з варіантів відповіді на поставлені питання. Також у модулі для вивчення граматики користувачі мають мати можливість створювати свої уроки та тести до них.

У модулі для вивчення лексики користувач повинен мати можливість додати слова собі у словник і запустити повторення слів, під час якого користувачу треба буде вибрати правильний переклад слова.

2.2. Вибір інструментальних засобів та технологій

Для розробки веб-системи було обрано середовище розробки IntelliJ IDEA для розроблення серверної частини за допомогою мови програмування Java і для розроблення веб-частини за допомогою мови програмування JavaScript. Для роботи з базою даних обрано середовище pgAdmin.

При розробленні веб-системи будуть використані наступні технології:

- **Spring Framework** – універсальний фреймворк з відкритим вихідним кодом для Java-платформи. Технологія розробки серверної частини. Будуть використані наступні модулі цієї технології: Spring Boot для швидкої конфігурації системи та розгортання на контейнер Java

сервлетів; Spring MVC для реалізації шаблону Model-View-Controller; Spring IoC для реалізації принципу Inversion of control; Spring Data для доступу до бази даних; Spring AOP для використання аспектно-орієнтовного програмування; Spring Security для забезпечення безпеки веб-системи.

- **Apache Tomcat** – контейнер Java сервлетів з відкритим вихідним кодом. Технологія розробки серверної частини.
- **Hibernate** – технологія об’єктно-реляційного відображення. Буде використана для відображення об’єктів мови Java в записи таблиць реляційної бази даних і навпаки. Також ця технологія буде використана для реалізації підходу Code First для створення таблиць реляційної бали даних з класів мови Java.
- **PostgreSQL** – вільно-розповсюджена об'єктно-реляційна система управління базами даних. База даних веб-системи буде зберігатися в цій системі.
- **Thymeleaf** – механізм шаблонів Java, який може працювати в Інтернеті (на основі сервлетів), так і не в Інтернеті. Буде працювати на основі сервлетів. Технологія веб-частини.
- **jQuery** – набір функцій JavaScript, що фокусується на взаємодії JavaScript і HTML. Буде використана для зручної взаємодії з DOM і для створення AJAX запитів.
- **Apache Maven** – технологія автоматизації збірки, який використовується переважно для проектів Java.
- **Apache Log4j** - утиліта ведення журналу подій на основі Java.

2.3. Специфікація вимог до веб-системи

2.3.1. Вступ

Призначення даної веб-системи полягає в наданні можливості автоматизовано вивчати лексику і граматику іноземних мов на прикладі англійської.

2.3.2. Загальний опис

2.3.2.1. Класи користувачів та їх характеристики

Для реалізації системи створені три класи користувачів: адміністратор, звичайний користувач та анонімний користувач.

Від класу користувача залежать його можливості та функціонал.

Класи адміністратор та звичайний користувач мають бути попередньо авторизовані. Звичайний користувач має бути зареєстрований для можливості авторизації.

Анонімний користувач – клас, який за замовчуванням надається усім неавторизованим користувачам. З функціональних можливостей має право тільки на реєстрацію та авторизацію для отримання прав звичайного користувача або адміністратора.

Звичайний користувач – клас, який має повний доступ до функціоналу веб-системи, окрім додаткового функціоналу адміністратора.

Адміністратор – клас, який має повний доступ до функціоналу веб-системи і додаткового функціоналу адміністратора, який включає в себе можливість видалення уроків з модуля граматики та видалення користувачів.

Звичайний користувач та адміністратор мають можливість вийти з облікового запису і отримати права анонімного користувача.

2.3.2.2. Середовище функціонування

Дане програмне забезпечення розробляється як веб-система, тобто для коректної роботи на стороні користувача для його середовища достатньо будь-якого браузеру та доступу в інтернет. Вимоги до середовища функціонування сервера можна знайти у таблиці 2.1.

Таблиця 2.1.

Вимоги до середовища функціонування сервера

Оперативна пам’ять	2 ГБ вільної оперативної пам’яті або більше.
Центральний процесор	Процесор з тактовою частотою 1 ГГц або більше.
Пам’ять на диску	500 Мб вільного дискового простору і можливість розширення для збереження журналів подій.
Операційна система	Система з підтримкою JVM.
Встановлене програмне забезпечення	JDK 11, Maven, PostgreSQL, Tomcat
Додатково	Доступ до мережі Інтернет

2.3.3. Характеристики системи

2.3.3.1. Реєстрація

2.3.3.1.1. Опис і пріоритет

Пріоритет високий. Реєстрація користувача у системі.

2.3.3.1.2. Послідовності дія/відгук

Потрібно перейти на сторінку реєстрації. Користувачу буде виведено поля для вказання логіну, пароля та підтвердження пароля. Також буде виведено кнопки реєстрації та переходу на сторінку логіна. При натисканні на кнопку реєстрації відбувається валідація введених користувачем даних. При успішній реєстрації користувач переходить на головну сторінку.

2.3.3.1.3. Функціональні вимоги

REQ-1: Необхідність вводу логіну, пароля та підтвердження паролю;

REQ-2: Валідація логіну, пароля та підтвердження пароля;

REQ-3: При виникненні помилки валідації полів вказати користувачу на помилку;

REQ-4: При успішній валідації надати користувачу права звичайного користувача.

2.3.3.2. Авторизація

2.3.3.2.1. Опис і пріоритет

Пріоритет високий. Авторизація користувача у системі.

2.3.3.2.2. Послідовності дія/відгук

Потрібно перейти на сторінку логіну. Ця сторінка є сторінкою за замовчуванням для анонімних користувачів. Користувачу буде виведено поля для вводу логіну та паролю. Також буде виведено кнопки для авторизації та переходу на сторінку реєстрації. При натисканні на кнопку авторизації відбувається валідація введених користувачем даних. При успішній авторизації користувач переходить на головну сторінку.

2.3.3.2.3. Функціональні вимоги

REQ-1: Необхідність вводу логіну та пароля;

REQ-2: Пошук логіну та пароля у базі даних;

REQ-3: При виникненні помилки валідації полів вказати користувачу на помилку;

REQ-4: Якщо користувача знайдено у базі даних, надати користувачу права звичайного користувача.

2.3.3.3. Перегляд списку уроків граматики

2.2.3.3.1. Опис і пріоритет

Пріоритет високий. Вивід уроків граматики користувачу.

2.3.3.3.2. Послідовності дія/відгук

Потрібно перейти на сторінку уроків граматики. Ця сторінка є сторінкою за замовчуванням для звичайних користувачів та адміністраторів. Користувач буде мати можливість бачити список останніх уроків, їх складність та автора. При натисканні на урок користувач буде направлений на сторінку уроку. Також присутня кнопка створення уроку, при натисканні на яку користувач переходить на сторінку створення уроку.

2.3.3.3.3. Функціональні вимоги

REQ-1: Вивід списку уроків;

REQ-2: Кожен урок у списку відображає назву, опис, складність та автора уроку;

REQ-3: На одній сторінці має відображатися до 20 уроків;

REQ-4: Якщо уроків більше ніж 20, групувати уроки до 20 на сторінку;

REQ-5: Якщо уроків більше ніж 20, користувач може перейти на іншу сторінку з уроками;

REQ-6: Уроки, на які користувач вже переходив мають інакший колір від інших.

2.3.3.4. Сторінка уроку

2.3.3.4.1. Опис і пріоритет

Пріоритет високий. Вивід теоретичної частини уроку користувачу.

2.3.3.4.2. Послідовності дія/відгук

Потрібно перейти на сторінку уроку. Користувачу буде виведено назву уроку, його складність, автора та теоретична частина уроку. В кінці теоретичного матеріалу знаходиться кнопка тестування, якщо урок містить практичну частину. При натисканні на кнопку тестування користувач переходить на сторінку тестування уроку.

2.3.3.4.3. Функціональні вимоги

REQ-1: Вивід інформації уроку: назву, складності, автора та теоретичної;

REQ-2: Якщо урок має практичну частину, користувач може перейти до тестування уроку.

2.3.3.5. Тестування уроку

2.3.3.5.1. Опис і пріоритет

Пріоритет високий. Користувач має можливість пройти тестування, яке належить певному уроку.

2.3.3.5.2. Послідовності дія/відгук

Потрібно перейти на сторінку тестування уроку. Користувачу буде виведено список питань та варіантів відповідей до нього. В кінці списку питань знаходиться кнопка завершення тестування. При натисканні на кнопку відбувається валідація кількості відповідей. Якщо відповіді на всі питання були наданні, користувачу буде виведено результат тестування.

2.3.3.5.3. Функціональні вимоги

REQ-1: Вивід списку питань;

REQ-2: Вивід варіантів відповідей біля кожного питання;

REQ-3: Можливість вибрати одну правильну відповідь до кожного питання;

REQ-4: Вивід результатів.

2.3.3.6. Словник

2.3.3.6.1. Опис і пріоритет

Пріоритет високий. Користувач має можливість бачити список своїх слів, які він вивчає. Також користувач має можливість додати слово до свого словника.

2.3.3.6.2. Послідовності дія/відгук

Потрібно перейти на сторінку лексики. Користувачу буде виведено список слів та їх переклад. Біля кожної пари слово-переклад знаходиться кнопка видалення слова. При натисканні на цю кнопку слово видаляється зі словника користувача.

Також окремо є поля для вводу іноземного слова та перекладу і кнопка для додавання слова. При натисканні кнопки для додавання слова це слово додається до словника користувача. Також присутня кнопка для тренування лексики, при натисканні якої користувач переходить на сторінку тренування лексики.

2.3.3.6.3. Функціональні вимоги

REQ-1: Вивід списку пар слово-переклад користувача;

REQ-2: Можливість видалити слово зі словника користувача;

REQ-3: Можливість додати слово до словника користувача;

REQ-4: Можливість переходу на сторінку тренування лексики.

2.3.3.7. Тренування лексики

2.3.3.7.1. Опис і пріоритет

Пріоритет високий. Користувач має можливість повторити слова, які знаходяться у нього в словнику.

2.3.3.7.2. Послідовності дія/відгук

Потрібно перейти на сторінку тренування лексики. Користувачу буде виведено список до 10 слів з його словника і варіанти перекладу. Користувач вибирає варіант перекладу до кожного слова і правильна відповідь виділяється зеленим кольором. Зберігається момент повторення кожного слова і при наступному тренуванні слова, які були повторенні найближчим часом до часу тестування будуть мати найменший пріоритет при вибору слів для тестування.

2.3.3.7.3. Функціональні вимоги

REQ-1: Вивід списку до 10 слів та варіантів їх перекладу;

REQ-2: Виділення правильної відповіді зеленим кольором;

REQ-3: Збереження моменту повторення слова.

2.3.3.8. Створення уроку

2.3.3.8.1. Опис і пріоритет

Пріоритет середній. Користувач має можливість створити урок та тест до нього.

2.3.3.8.2. Послідовності дія/відгук

Потрібно перейти на сторінку створення уроку. Користувач має можливість ввести назву, опис, складність за 5-бальною шкалою та теоретичний матеріал. Ці поля є обов’язковими. Також користувач може опціонально додати тест до уроку.

При додаванні тесту користувач має додати від 1 до 10 питань та від 2 до 6 варіантів відповідей до нього. Присутня кнопка збереження уроку, при натисканні на яку відбувається валідація полів і урок зберігається в базу даних.

2.3.3.8.3. Функціональні вимоги

REQ-1: Необхідність вводу назви, опису, складності та теоретичного матеріалу уроку;

REQ-2: Можливість додати тест до уроку;

REQ-3: Можливість додати від 1 до 10 питань з від 2 до 6 варіантів відповідей до тесту;

REQ-4: Валідація усіх полів;

2.3.3.9. Функції адміністратор

2.3.3.9.1. Опис і пріоритет

Пріоритет середній. Адміністратор має можливість видалити тест або заблокувати користувача.

2.3.3.9.2. Послідовності дія/відгук

Потрібно перейти на сторінку граматики. Біля кожного уроку є кнопка видалення. При натисканні кнопки видалення урок видаляється з бази даних. Також у шапці веб-сайту є кнопка блокування користувача.

При натисканні на кнопку блокування користувача з’являється поле для вводу логіну користувача і кнопка підтвердження блокування, при натисканні якої, якщо користувач знайдений, він блокується.

2.3.3.9.3. Функціональні вимоги

REQ-1: Можливість видалення уроку адміністратором;

REQ-2: Можливість блокування користувача адміністратором;

REQ-3: Перевірка логіну користувача на існування перед блокуванням.

2.3.4. Вимоги зовнішніх інтерфейсів

2.3.4.1. Апаратні інтерфейси

Машина, на якій буде розгортатися серверна частина веб-системи має мати наступні характеристики:

- Центральний процесор з тактовою частотою 1 ГГц, або більше;
- 2 ГБ вільної оперативної пам’яті або більше;
- 2 ГБ вільного дискового простору або більше;

2.3.4.2. Програмні інтерфейси

Сервер взаємодіє з базою даних PostgreSQL, тому для роботи сервера вона має бути встановлена на машину, на якій знаходиться сервер, або на іншу, але при цьому її адреса має бути вказана в конфігураційних файлах сервера. Дана взаємодія відбувається по TCP порту 5432. Також для роботи сервера має бути встановлений контейнер сервлетів Tomcat. Tomcat використовує порт 8080.

2.3.4.3. Комунікаційні інтерфейси

Для роботи веб-системи вимагається доступ до мережі Інтернет. Комунікація з іншими компонентами системи може відбуватися в межах однієї машини (якщо вони знаходяться на одній машині), в межах локальної мережі або в межах мережі Інтернет, для цього інші компоненти також мають мати доступ до мережі Інтернет.

Висновки до розділу 2

Під час роботи над цим розділом було зроблено загальну постановку задачі розробки веб-системи для вивчення іноземних мов на прикладі англійської мови. Було зазначено, що головні задачі створюваної веб-системи це автоматизоване вивчення граматики та лексики іноземної мови. Також сформовано основні вимоги для коректного функціонування системи та функціональні характеристики.

Також можна відзначити, що для реалізації веб-системи для вивчення граматики та лексики іноземних мов було обрано наступні середовища розробки: IntelliJ IDEA та pjAdmin. В самій розробці будуть використані наступні технології: Spring, Thymeleaf, PostgreSQL.

В результаті створено специфікацію вимог до програмного забезпечення, яка і стане основою для проектування і реалізації веб-системи.

РОЗДІЛ 3. АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ВЕБ-СИСТЕМИ

3.1. Архітектура системи

Для реалізації програмного застосунку було обрано клієнт-серверну архітектуру. Дана модель архітектури найкраще підходить для реалізації програмного застосунку даного типу. Такий висновок можна зробити з того, що клієнт-серверна архітектура надає можливість створення кроссплатформеного програмного забезпечення.

Дана модель передбачає існування одного центрального сервера з яким взаємодіють клієнти. Загалом процес взаємодії клієнтів з сервером є наступним: клієнт відправляє запит серверу, сервер взаємодіє з базою даних, опрацьовує інформацію і надсилає клієнту відповідь. Клієнтів може бути декілька. Загалом структуру можна побачити на діаграмі розгортання, створеної для веб-системи, що наведена на рис. 3.1.

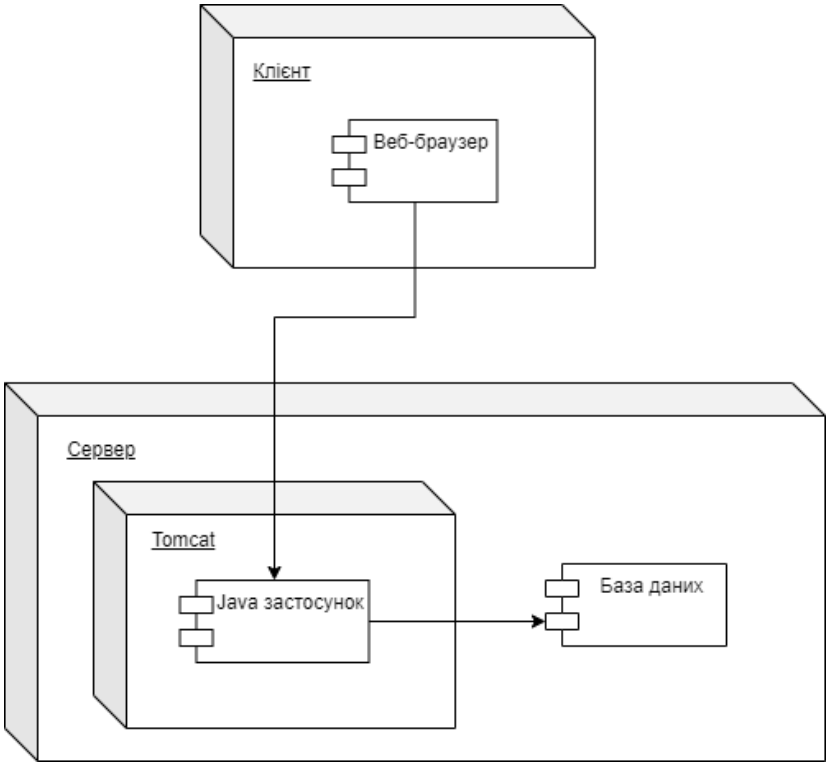

Рис. 3.1. Діаграма розгортання веб-системи.

Для забезпечення серверної кроссплатформеності обрано мову програмування Java. Веб-застосунки на даній мові компілюються у файли з розширенням war – Web Archive або Web Application Resource. Цей файл містить усі скомпільовані Java класи і усі ресурси, які необхідні для запуску застосунку.

Для компіляції Java класів, завантаження необхідних бібліотек, копіювання ресурсів і збірки всього у war файл використовується технологія збірки Maven. Дана технологія створює готовий war файл і розгортає його на контейнері сервлетів Tomcat.

Основою веб-системи стане модуль фреймворка Spring – Spring MVC. Цей модуль дозволяє застосувати шаблон проектування MVC для написання всіх частини веб-системи, що зробить процес розробки програмного забезпечення швидшим і відкриє доступ до паралельної розробки. Також це означає, що кожний об’єкт системи можна віднести до певного шару шаблону – контролера, моделі чи представлення.

Підхід до розробки серверної частини буде об’єктно-орієнтовним. Сама серверна частина має 3 шари: шар контролерів, який відповідає за HTTP запити та відповіді, шар сервісів, який відповідає за бізнес-логіку системи та репозиторіїв, який відповідає за взаємодію сервера з базою даних. Шар репозиторіїв реалізує шаблон проектування Repository. Тобто кожен клас-репозиторій повністю інкапсулює взаємодію з певною частино бази даних, а усі репозиторії повністю покривають всю взаємодію з базою даних.

Також використовується ще один модуль фреймворка Spring – Spring IoC. Головна задача цього модуля – зменшити кількість зв’язків у системі. Цей модуль є реалізацією шаблону Inversion of Control. На практиці це означає, що існує певний контейнер компонент, який відповідає за життєвий цикл цих компонент та їх зв’язки. Усі компоненти системи, будь то контролери, сервіси чи репозиторії або інші допоміжні компоненти, будуть належати цьому контейнеру.

Для вибору технології клієнтської частини, тобто технології забезпечення View частини у модулі Spring MVC було проаналізовано специфікацію вимог та створено діаграму прецедентів для розуміння випадків використання користувачами даної веб-системи. Діаграма прецедентів знаходить на рис. 3.2.

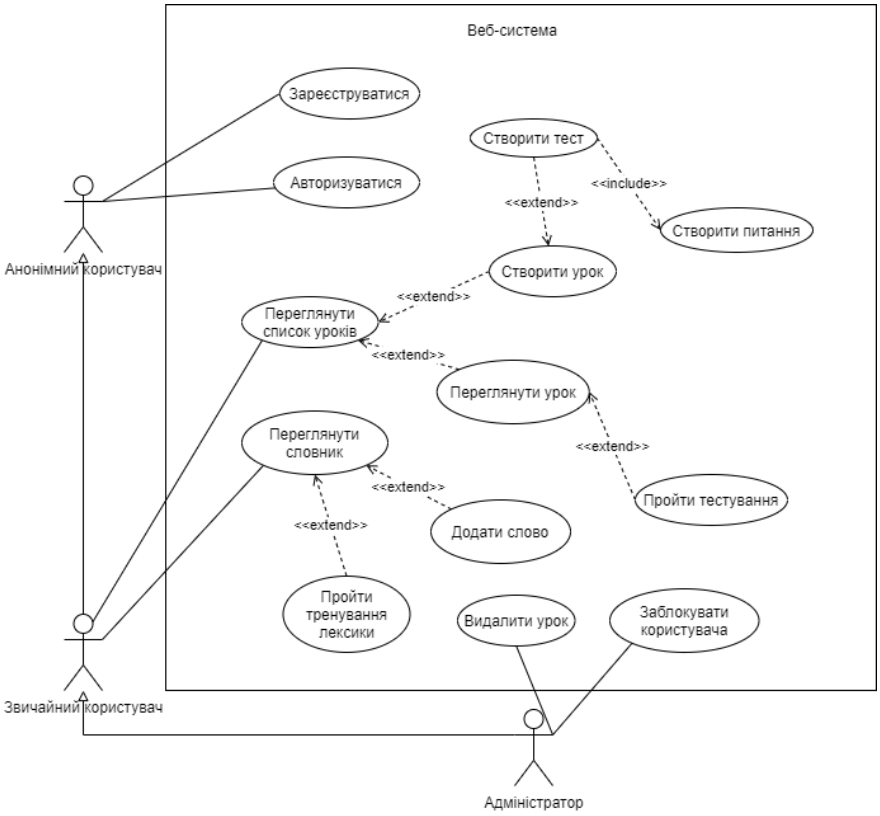

Рис. 3.2. Діаграма прецедентів веб-системи.

На основі проаналізованих та складених даних було обрано технологію Thymeleaf для створення клієнтського графічного інтерфейсу веб-системи. Thymeleaf – це механізм Java шаблонів, який може обробляти багато видів файлів, у цьому випадку це HTML, CSS і JavaScript файли. Ця технологія відповідає за шаблонізацію графічного інтерфейсу користувача. Тобто у кожної компоненти інтерфейсу є статичний шаблон, в який Thymeleaf динамічно впроваджує елементи логіки, при цьому не змінюючи сам прототип дизайну.

Для взаємодії клієнта з сервером використовується протокол HTTP і його методи POST та GET. Використовуються як синхронні запити так і асинхронні. Для асинхронних запитів використовується технологія AJAX. Ці технології також підходять для забезпечення умови роботи в мережі Інтернет для забезпечення кроссплатформеності.

Для забезпечення безпеки використовується технологія HTTP Basic access authentication. Клієнт передає свій логін та пароль при кожному запиті у повідомленні HTTP. А саме у його заголовку Authorization у вигляді логін:пароль.

Щодо інших використаних шаблонів та підходів проектування можна відмітити те, що Spring Framework у більшості своїх модулів використовує шаблон Proxy для надання або зміни певних особливостей коду користувацьких класів. Також модуль Spring IoC використовує шаблон Singleton, і всі компоненти в цьому випадку мають лише один екземпляр. Також використана бібліотека Lombok, яка реалізує шаблон Builder для створення певних об’єктів.

3.2. Проектування структури програмного забезпечення та її компонент

Проект має стандартну структуру для Maven веб-проекту. На верхньому рівні знаходяться дві папки: src і target. В папці scr вихідний код системи, а в папку target компілюється проект у вигляді war файлу для подальшого розгортання. Сама конфігурація Maven також знаходиться на верхньому рівні у файлі pom.xml. Головний код системи знаходиться за шляхом src/main/java, а допоміжні компоненти, такі як файли представлення і конфігурації за шляхом src/main/java. Також можна відмітити 4 основні Java пакети з яких складається застосунок: controller, service, repository, persistence. Перші три є пакетами для компонент трьох відповідно названих шарів. А останній відповідає за відображення об’єктів бази даних і за генерацію таблиць бази даних у підході Code First. Повну структуру проекту можна побачити на рис. 3.3.

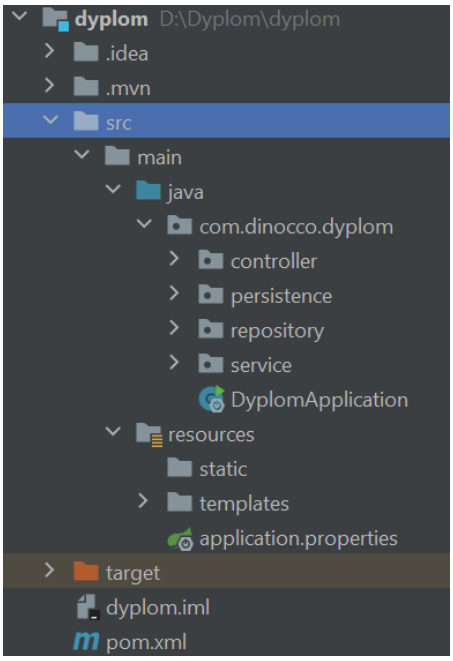

Рис. 3.3.Структура проекту.

Також були спроектовані такі компоненти:

- UserController – компонент, який відповідає за обробку запитів для роботи з користувачами системи.
- SecurityController – компонент, який відповідає за обробку запитів для реєстрації та авторизації.
- LessonController – компонент, який відповідає за обробку запитів для роботи з уроками.
- WordController – компонент, який відповідає за обробку запитів для роботи зі словами.
- QuestionController – компонент, який відповідає за обробку запитів для роботи зі питаннями тестування.
- OptionController – компонент, який відповідає за обробку запитів для роботи зі варіантами відповідей до питання.

3.3. Проектування бази даних

При розробці бази даних використано підхід Code First, а реалізовано його технологією Hibernate. Використання підходу Code First означає, що спочатку створюються Java-класи сутностей в Java застосунку, на основі яких

генеруються таблиці та зв’язки між ними у базі даних. Таким чином структура бази даних виходить дуже схожою на діаграму класів застосунку. Hibernate також використовується для відображення екземплярів Java класів в записи таблиць і навпаки.

Технологія Hibernate використовує шаблон проектування Proxy і динамічно створює похідний клас від класу репозиторію, при цьому автоматично відкриваючи і закриваючи з’єднання з базою даних, а також створюючи транзакцію при кожному зверненні до бази даних, навіть якщо це явно не вказано у коді.

Остаточна логічна модель бази даних зображена на рис. 3.4. Фізична модель зображена на рис. 3.5.

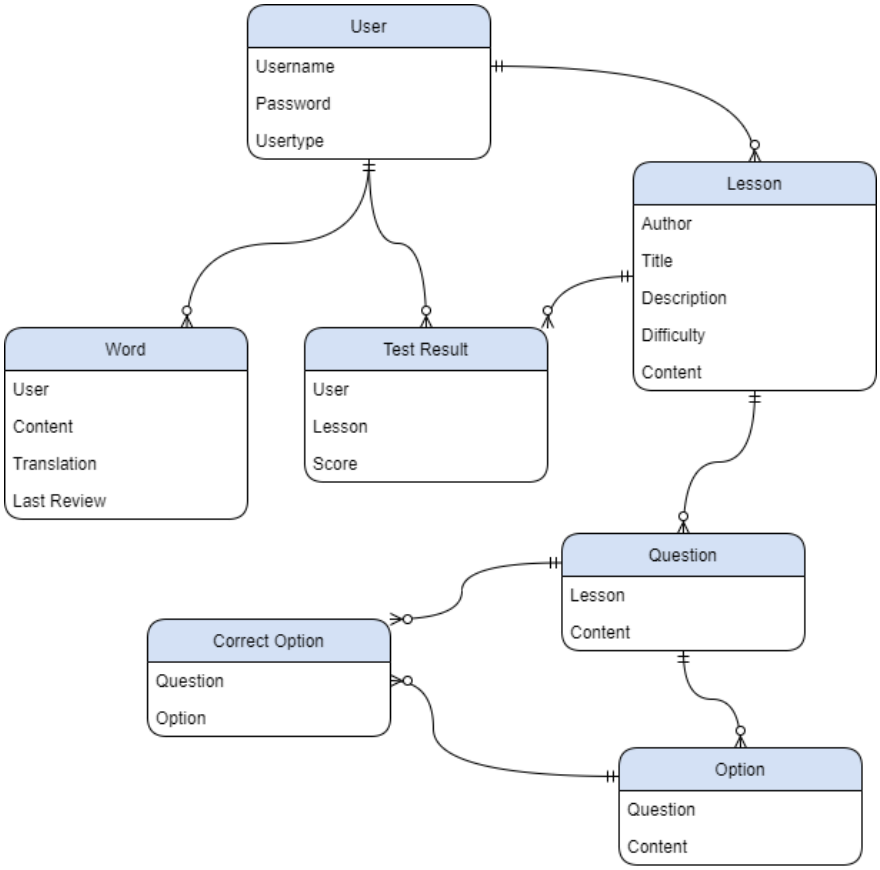

Рис. 3.4. Логічна модель бази даних.

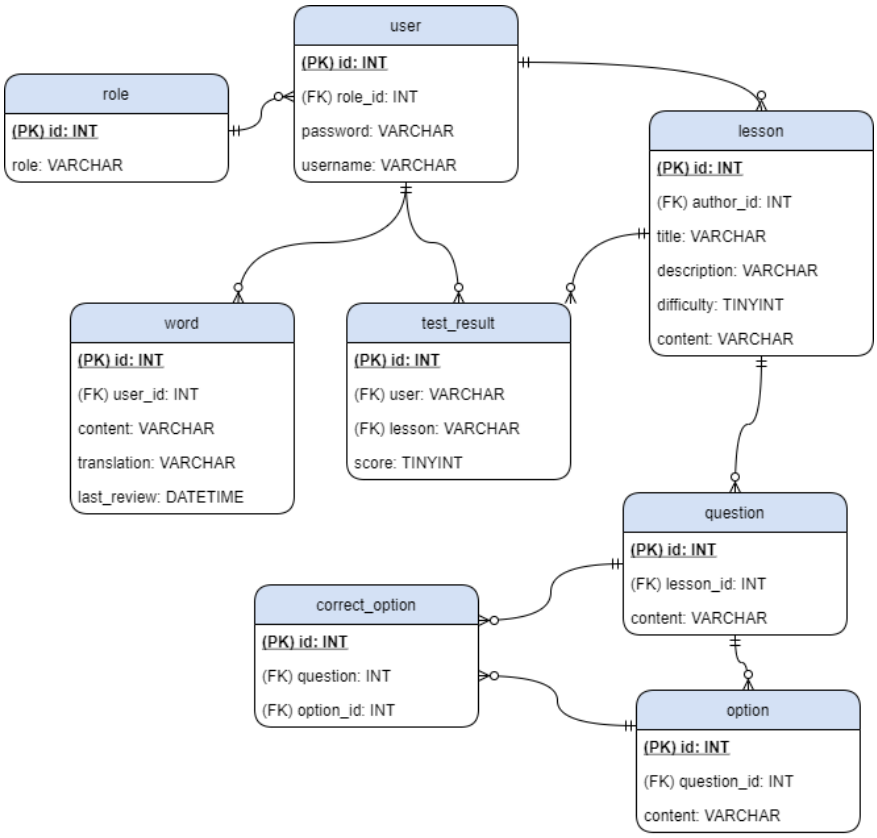

Рис. 3.5. Фізична модель бази даних.

Опис основних таблиць:

1) User – таблиця, яка містить інформацію про користувачів системи;
   - Username – логін користувача;
   - Password – пароль користувача;
   - Usertype – клас користувача.
2) Lesson – таблиця, яка містить інформацію про уроки граматики;
   - Author – користувач, який створив урок;
   - Title – назва уроку;
   - Description – опис уроку;
   - Difficulty – складність уроку по 5-бальній шкалі;
   - Content – теоретичний матеріал уроку.
3) Question – таблиця, яка містить інформацію про питання тесту;
   - Lesson – урок, якому належить питання;
   - Content – текст питання.

4) Option – таблиця, яка містить інформацію про варіанти відповідей на питання;
    - Question – питання, якому належить варіант відповіді;
    - Content – текст варіанту.
5) Correct Option – таблиця, яка містить інформацію про правильні варіанти відповідей на питання;
    - Question – питання, якому належить правильний варіант відповіді;
    - Option – правильний варіант відповіді.
6) Test Result – таблиця, яка містить інформацію про результати тестування користувачів;
    - User – користувач, якому належить результат тестування;
    - Lesson – урок, якому належить результат тестування;
    - Score – кількість правильних відповідей тестування.
7) Word – таблиця, яка містить інформацію про слова зі словників користувачів;
    - User – користувач, якому належить слово;
    - Content – текст слова на іноземній мові;
    - Translation – переклад слова;
    - Last Review – дата останнього повторення слова.

Висновки до розділу 3

В результаті проектування архітектури системи було обрано клієнт серверну архітектуру для розробки веб-системи. На цей вибір вплинула можливість охопити машини та пристрої будь-якого типу, будь то мобільний пристрій чи персональний комп’ютер. Для забезпечення кроссплатформеності серверного застосунку обрано мову програмування Java у зв’язці з технологіями фреймворку Spring. У ролі контейнера сервлетів було обрано сервер Tomcat. Для забезпечення представлення графічного інтерфейсу обрано технологію Thymeleaf, а для збереження даних обрано базу даних PostgreSQL. З’єднання між клієнтом та сервером буде відбуватися по протоколу HTTP та Basic access authentication для забезпечення безпеки.

Під час проектування серверного застосунку та його компонентів було вирішено використовувати 3-шарову архітектуру, у якій всі компоненти поділяються на відповідні 3 типи: компоненти взаємодії з HTTP запитами та відповідями, компоненти бізнес логіки та компоненти взаємодії з базою даних. Окремо розроблено контролери. Кожен контролер відповідає за свою частину. Список спроектованих контролерів: UserController, SecurityController, LessonController, WordController, QuestionController, OptionController.

Такод під час проектування бази даних було обрано використовувати технологію Code First для генерації таблиць бази даних та зв’язків між ними. Розроблені сутності бази даних і на їх основі логічну та фізичну модель бази даних.

Таким чином, проведений процес проектування веб-системи значно спрощує процес її розробки, а також зменшує витрачений на це час.

РОЗДІЛ 4. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-СИСТЕМИ

4.1. Процес реалізації веб-системи

Процес розроблення веб-системи відбувався в інтегрованому середовищі розробки IntelliJ IDEA як для серверної, так і для клієнтської частини, а база даних створена на сервері PostgreSQL. Також для тестування програмного інтерфейсу серверної частини був використаний Postman.

Спочатку була розроблена серверна частина веб-системи. Першим етапом в процесі розроблення серверної частини було створення класів-сутностей бази даних, їх обмежень та зв'язків на основі спроектованих в попередньому розділі моделей бази даних. Це було зроблено для їх подальшого перетворення у таблиці бази даних за допомогою технології для об'єктно реляційного відображення Hibernate як реалізації підходу Code First. Список створених на цьому етапі класів-сутностей можна побачити на рис. 4.1.

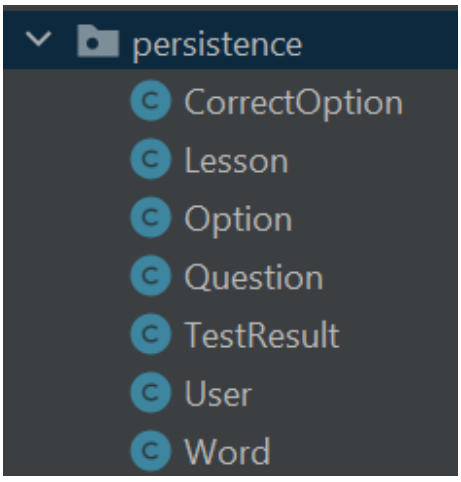

Рис. 4.1. Список класів-сутностей.

В процесі створення класів-сутностей були створені наступні класи: CorrectOption, Lesson, Option, Question, TestResult, User, Word. Для створення даних класів були використані тільки анотації Java Persistence API, реалізацією якого є технологія Hibernate. Даних підхід дає можливість змінити реалізацію Java Persistence API у майбутньому, при цьому не змінюючи код програми, а тільки конфігураційні файли. Код прикладу класу-сутності можна побачити у додатку А.

Далі було розроблено один з трьох основних шарів серверної частини веб-застосунку – шар репозиторіїв. Даний шар відповідає за маніпулювання раніше створеними класами-сутностями для роботи з відповідними даними в базі даних. Він побудований на шаблоні Repository і є його реалізацією. Для цієї реалізації використані можливості модуля Spring Data.

Даний модуль є обгорткою над Java Persistence API, яка значно спрощує взаємодію з класами-сутностями веб-системи. Також цей модуль запроваджує свій власний синтаксис, який побудований на назвах методів інтерфейсів, які він перетворює на об’єкти Java Persistence API, які в подальшому перетворюються на повноцінні SQL-запити. Spring Data використовує модель Java інтерфейсів. Для кожного створеного на попередньому етапі класу-сутності створюється свій окремий інтерфейс-репозиторій який наслідується від інтерфейсів Spring Data, в даному випадку від інтерфейсу JpaRepository.

В інтерфейсі-репозиторії наявні сигнатури методів, які відповідають за певну частину взаємодії з базою даних відповідної сутності. В процесі запуску застосунку, модуль Spring Data реалізує дані інтерфейси-репозиторії на основі їх сигнатури у код, який відповідає Java Persistence API, і також далі реалізації інтерфейсів-репозиторіїв поміщаються в контекст компонентів системи модуля Spring IoC для подальшого використання в інших шарах системи. Список створених інтерфейсів-репозиторіїв можна побачити на рис. 4.2.

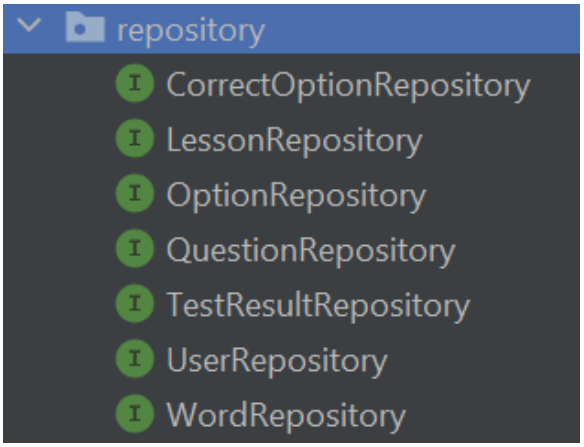

Рис. 4.2. Список інтерфейсів-репозиторіїв.

Наступним етапом було створено шар контролерів. Цей шар створений за допомогою модуля Spring MVC. Створено два види контролерів: контролери, які повертають HTML-сторінку і REST-контролери, які повертають JSON-об’єкт. Перші в свою чергу використовуються здебільшого для представлення інформації, а другі для її редагування. Контролер – це звичайний Java-клас, який має відповідну анотацію модуля Spring MVC. Кожен з його методів є кінцевою точкою в системі адрес застосунку. Код прикладу класу контролеру можна побачити у додатку Б. Розроблено наступні класи-контролери:

- WordController.java – контролер, який відповідає за обробку запитів, які стосуються роботи зі словами користувача;
- LessonController.java – контролер, який відповідає за обробку запитів, які стосуються роботи з уроками;
- PageController.java – контролер, який відповідає за обробку запитів, які у відповідь отримують статичну HTML-сторінку і їх обробка не використовує бізнес-логіку;
- CustomErrorController.java – контролер, який відповідає за обробку запитів, які завершилися помилкою.

Також розроблено наступні REST-контролери:

- LessonRestController.java – контролер, який відповідає за обробку REST-запитів, які стосуються роботи з уроками;
- UserRestController.java – контролер, який відповідає за обробку REST-запитів, які стосуються роботи з користувачами системи;
- WordRestController.java – контролер, який відповідає за обробку REST-запитів, які стосуються роботи зі словами користувача.
- конфігураційні файли.

Для створення структури HTTP-запитів та відповідей були розробленні додаткові класи. Для цього використано шаблон Data Transfer Object, і це означає, що дані класи не несуть функціонального навантаження, а створенні лише для передачі даних між процесами. При недотриманні очікуваної

структури запиту система автоматично повідомить про помилку у відповіді на запит з кодом 400. Також на цьому рівні розроблено систему валідації отриманих даних в запитах, яка не тільки структуру, але їх вміст на відповідність вимогам системи. Валідація запитів розроблена з використанням Bean Validation API. Кожен можливий запит у системі має бути провалідований. Процес розроблення системи валідації являє собою додавання анотацій Bean Validation API над полями структури Data Transfer Object класів, які відповідають за запити. Таким чином було введено обмеження на довжину, вміст, можливі значення полів тощо. Певні обмеження неможливо було виразити за допомогою існуючих анотацій Bean Validation API, тому були створені власні анотації та класи-валідатори до них. Це:

- Анотація CorrectOptionConstraint.java і відповідний клас-валідатор CorrectOptionConstraintValidator.java, який відповідає за те, щоб при додаванні питання завжди була надана одна правильна відповідь;
- Анотація PasswordsEqualConstraint.java і відповідний клас-валідатор PasswordsEqualConstraintValidator.java, який відповідає за те, щоб при реєстрації поля паролю та підтвердження паролю були ідентичними.

Також на цьому рівні був створений обробники виняткових ситуацій. Вони створений за допомогою модуля Spring AOP, який є реалізацією аспектно-орієнтованого програмування. Дані обробники являють собою клас з методами, кожен з яких відповідає за обробку певної виняткової ситуації з усіх рівнів системи, а також за відповідь сервера у випадку цієї виняткової ситуації. Модуль Spring AOP обгортає кожен з методів класів-контролерів усіма заданими в цьому класі обробниками виняткових ситуацій.

Завершальним етапом розроблення серверної частини було створення класів та інтерфейсів, які відповідають за бізнес-логіку застосунку. Всього було створено 5 класів-сервісів:

- LessonService.java – сервіс, який реалізує бізнес-логіку роботи застосунку з уроками;

- QuestionService.java – сервіс, який реалізує бізнес-логіку роботи застосунку з питаннями тестів;
- OptionService.java – сервіс, який реалізує бізнес-логіку роботи застосунку з варіантами відповідей до питань;
- WordService.java – сервіс, який реалізує бізнес-логіку роботи застосунку зі словами користувачі;
- UserService.java – сервіс, який реалізує бізнес-логіку роботи застосунку з користувачами системи.

Також створені дві виняткові ситуації: RestBadRequestException і InternalServerException.java. Перша використовується у випадку помилки користувача, а друга у випадку помилки сервера.

Створена конфігурація для збереження журналів роботи програми. Журнали застосунку виводяться у консоль, а також зберігаються на диск у директорію logs відносно розташування jar файлу. Журнали налаштовані на створення одного файлу кожен день з відповідною датою у назві файлу. Для цього використана технологія Logback.

Окремо створена конфігурація безпеки додатку. Для забезпечення безпеки використовується модуль Spring Security. Безпека застосунку налаштована наступним чином: сторінки реєстрації та авторизації доступні анонімним користувачам; усі інші сторінки системи доступні тільки авторизованим користувачам; кінцеві точки, шлях яких починається на шаблон /admin/** доступні тільки адміністраторам.

Після завершення розроблення серверної частини була розроблена веб-частина, яка відповідає за інтерфейс користувача. Для цього застосовано двигун шаблонів Thymeleaf. Ця технологія дозволила зробити сторінки динамічними, тобто такими, які відображають актуальні дані, які збережені в базі даних. Для розмітки шаблонів Thymeleaf було обрано мову розмітки гіпертексту HTML, а для створення стилів та інтерактивності вже звичні для такого типу застосунків мова стилю сторінок CSS і мова програмування

JavaScript. Для забезпечення зручного способу маніпуляції даними сторінок використано бібліотеку jQuery. Також використано технологію Bootstrap для спрощення роботи зі стилями і використання вже готових класів стилей. Для створення асинхроних запитів використано технологію AJAX.

Усі файли веб-частини застосунку знаходяться за стандартним для технології Thymeleaf шляхами: src/main/resources/static та src/main/resources/templates.

Загалом у веб-частині застосунку використовується 3 типи файлів:

- .html – шаблони сторінок Thymeleaf;
- .css – реалізація стилів для елементів сторінок;
- .js – скрипти мови JavaScript для забезпечення інтерактивності та AJAX-запитів.

4.2. Опис роботи з програмним продуктом

Після запуску застосунку він доступний за портом 8080, проте його можна змінити у конфігураційному файлі застосунку за шляхом src/main/resources/application.properties. Після переходу за веб-адресою сервера та відповідним портом користувачеві доступно дві сторінки: сторінка авторизації (рис. 4.3) та сторінка реєстрації (рис. 4.4).

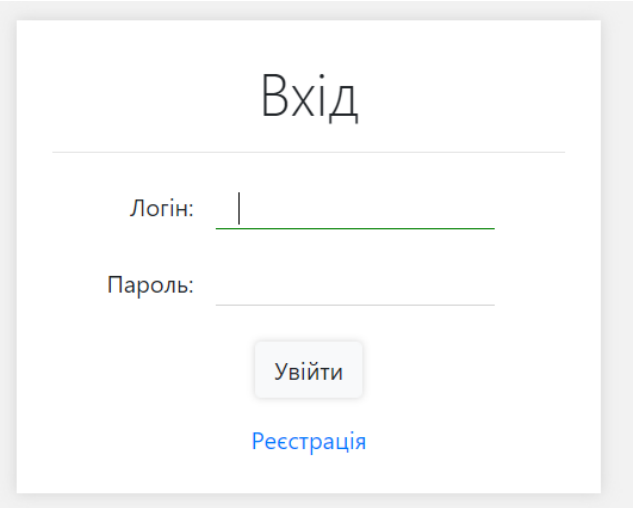

Рис. 4.3. Сторінка авторизації.

Якщо користувач не проходив процес реєстрації до цього, для доступу до функцій веб-систем він має його пройти. Для цього на сторінці реєстрації користувач має ввести логін, пароль та підтвердження паролю.

Логін має бути унікальним. Якщо при реєстрації виникають помилки, вони відображаються у лівому верхньому куті. Цей шаблон є універсальним і усі помилки користувача або системи відображаються у верхньому лівому куті кожної сторінки веб-системи.

Після успішної реєстрації користувач автоматично авторизований у системі і попадає на сторінку граматики. Якщо користувач вже проходив процес реєстрації, він має можливість авторизуватися у системі на сторінці авторизації за допомогою введених ним на етапі реєстрації логіну та паролю.

Після успішної авторизації користувач також попадає на сторінку граматики. Після авторизації усі сторінки веб-системи мають у верхньому правому куті блок з ім’ям авторизованого користувача та кнопкою для виходу із системи.

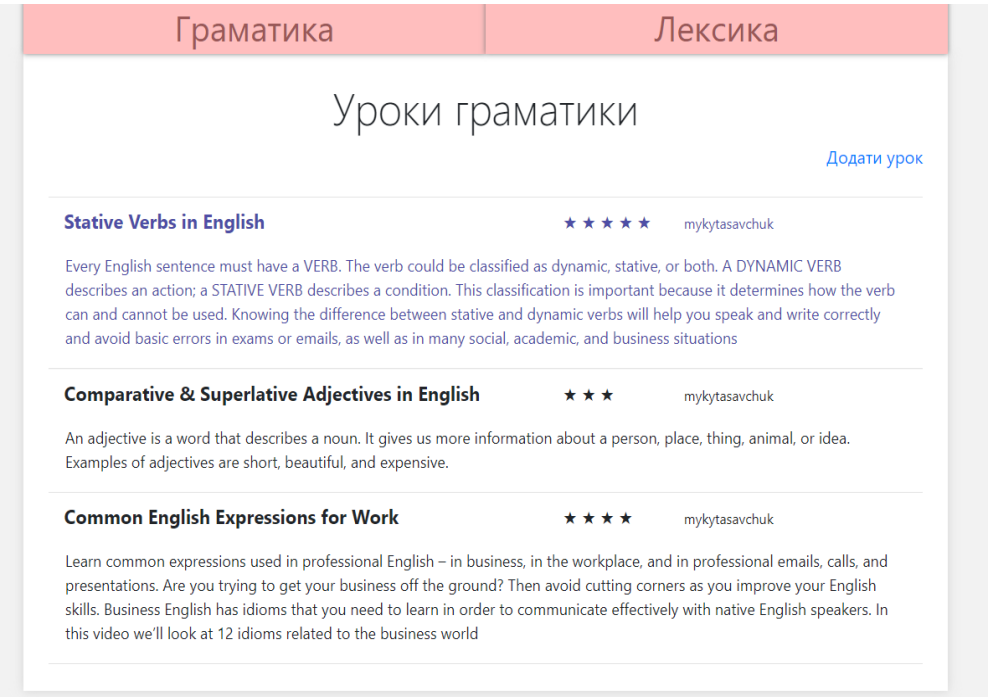

Рис. 4.4. Сторінка граматики.

Сторінка граматики містить усі доступні уроки граматики, а саме їх назви, складності, автори та описи. Якщо уроків багато, вмикається механізм сторінок. Кожна сторінка може містити до 15 уроків граматики.

Якщо користувач вже переходив за посилання певного уроку, він має колір синього відтінку у списку уроків.

А якщо користувач проходив тест до певного уроку, останній результат його тестування знаходиться біля автора уроку. Також доступне посилання у верхній частині сторінки для переходу на сторінку створення уроку.

Окремо можна виділити кнопки для переходу на сторінки граматики та лексики у верхній частині сторінки, які є шапкою веб-системи і доступні на всіх сторінках після авторизації.

Для переходу на сторінку конкретного уроку треба натиснути на нього. Загальний вигляд сторінки граматики можна побачити на рис. 4.5.

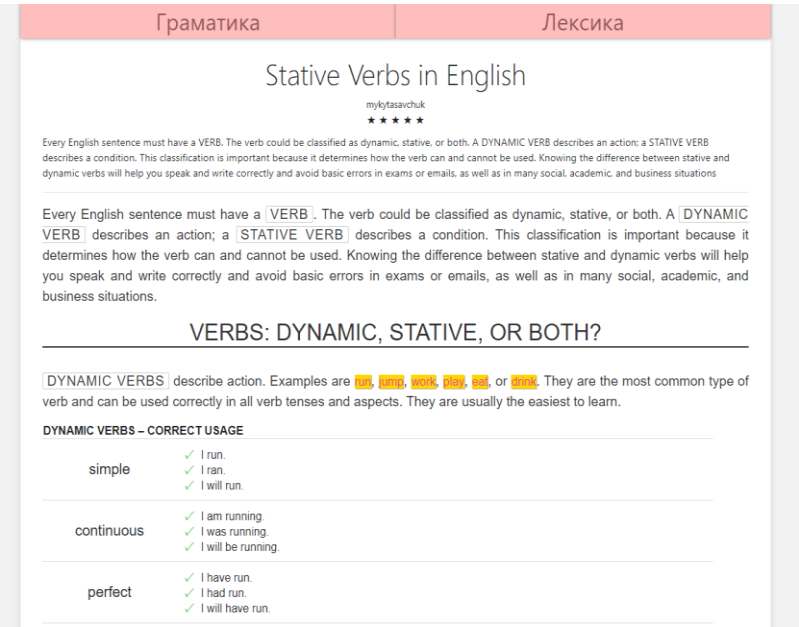

Рис. 4.5. Сторінка уроку.

Сторінка уроку містить назву уроку, його складність, автора, опис та сам текст уроку.

Якщо урок має тест, кнопка для переходу на нього знаходиться в кінці уроку, а якщо користувач вже проходив цей тест, кнопка для переходу на сторінку останнього результату тестування знаходиться поруч з кнопкою

переходу на тестування. Загальний вигляд сторінки уроку можна побачити на рис. 4.6.

Сторінка тесту містить уроку, всі питання тесту та варіанти відповідей до них. В кінці знаходиться кнопка завершення тестування, яка спрямовує користувача на сторінку з результатом тестування. Загальний вигляд сторінки тесту можна побачити на рис. 4.7.

Сторінка результату тестування містить назву уроку, кількість правильних відповідей та відсоток правильних відповідей.

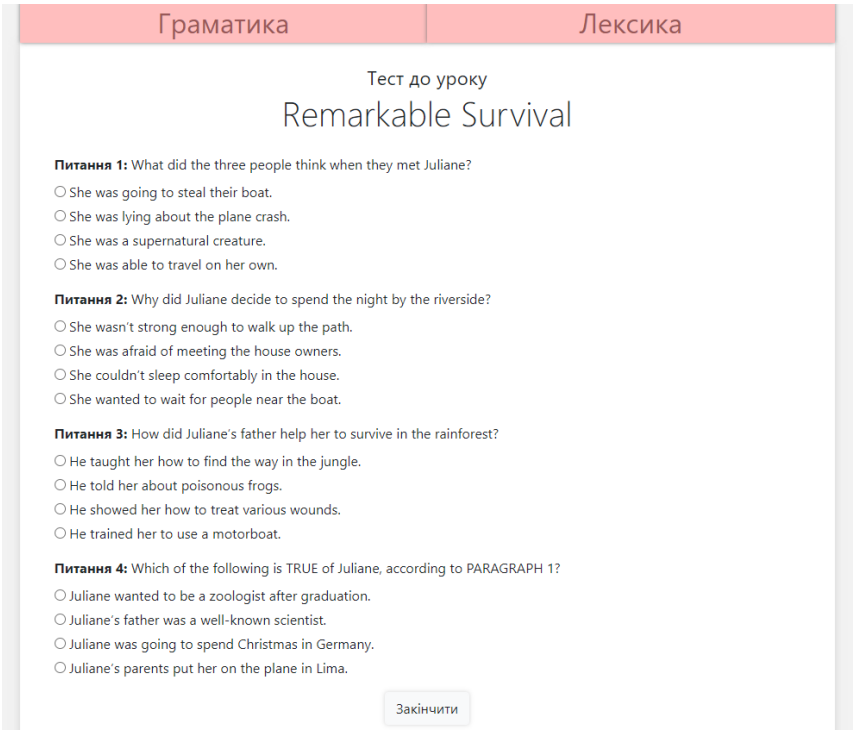

Рис. 4.6. Сторінка тесту.

Сторінка додавання уроку містить поля для вводу назви, опису, складності, тексту уроку, а також для вводу питань та варіантів до питань. Для додавання питання треба натиснути кнопку Додати питання.

Питання не є обов’язковими, і якщо питання не були додані, урок не буде мати тест. Максимальна кількість питань – 10. Кожне питання може містити від 2 до 6 варіантів відповіді. Також кожне питання має обов’язково мати один правильний варіант відповіді. Щоб позначити правильний варіант треба натиснути на радіо-кнопку поруч з ним.

Сторінка лексики містить інформацію про додані слова у словник, форму для додавання нового слова у словник, яка складається з поля для вводу слова, поля для вводу перекладу та кнопки для відправки запиту. Також поруч є кнопка для переходу на сторінку тестування. Якщо слів більше ніж 20, застосовується механізм сторінок. На кожній сторінці може бути до 20 слів.

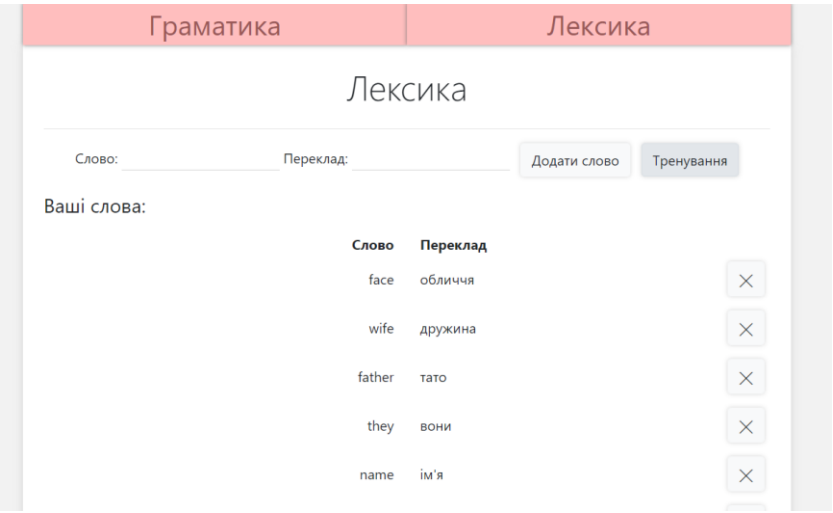

Рис. 4.7. Сторінка лексики.

Сторінка тренування містить слово на іноземній мові та 4 кнопки варіантів перекладу. Після натискання на один з варіантів перекладу правильний переклад буде мати зелений колір, а якщо був вибраний неправильний варіант перекладу, він буде мати червоний колір. Для тренування наступного слова треба натиснути на кнопку Наступне слово. Найбільший пріоритет мають слова, які були вибрані останній раз для тренування найдавніше або не вибрані ніколи.

4.3. Опис процесу тестування програмного забезпечення

Для відстеження коректної роботи веб-системи для вивчення граматики та лексики іноземних мов вона була покрита тестами. Саме завдяки цьому етапу можна знайти непередбачувані помилки системи та полегшити роботу мануальним тестерам.

4.3.1. Вступ

Для того, щоб система працювала ефективно вона була покрита тестовими випадками і автоматизованими тестами. Основною метою даного

етапу є створення тестів, які в майбутньому при додаванні нового функціоналу або при модифікації вже існуючого функціоналу будуть відстежувати коректну роботу усіх компонент системи, а також одразу розпізнавати пошкоджену функціональність.

4.3.2. Розроблення тестів

Серверну частину веб-системи було вирішено покрити функціональними тестами. Для цього було обрано технологію модульного тестування JUnit, а також технологію Mockito. За допомогою цієї технології створені тести для сервісів та контролерів. Самі тести знаходяться у директорії test, яка в свою чергу знаходиться на рівні з директорією main, яка містить головний код програми. Для запуску тестів використано технологію Maven. Перед тим, як зібрати проект до купи у jar файл, Maven після компіляції коду виконає усі тести, які знаходяться в директорії test. Якщо один з тестів виявиться неуспішним, Maven зупинить процес збірки застосунку.

4.3.3. Функціональне тестування

4.3.3.1. Результати тестування

Функціональне тестування покрило бізнес-логіку застосунку та його контролери. Завдяки цьому можна стверджувати про коректність роботи програмної системи. Всього було розроблено 14 тестових випадків. Вони описані детально у додатку Г.

4.3.3.2. Підсумки тестування

Під час розроблення програмного застосунку, для його мануального тестування використовувалася технологія Postman. Щодо автоматизованих тестів, то після розроблення кожної частини певного функціоналу системи, вона одразу покривалася функціональними тестами. Завдяки такому підходу можна було постійно бути впевненим в коректній роботі вже розробленого функціоналу та швидко відловлювати помилки.

Розподіл функціонального тестування можна побачити в таблиці 4.1.

Таблиця 4.1

Розподіл функціонального тестування

Варіанти використання	**Тестові випадки**
Сервіс користувача	2
Сервіс слова	2
Сервіс уроку	2
Сервіс варіанту відповіді	2
Сервіс питання	2
Контролер уроку	2
Контролер слова	2
Загалом	**14**

Так як у фінальному варіанті програми всі тести пройдено успішно, можна вважати функціональне тестування також успішним.

4.3.3.3. Критерії успіху/провалу проекту

Стверджувати про успішність розробленого програмного застосунку можна за наступними умовами:

Умови розроблення тестів:

- Усі варіанти використання програмного застосунку мають достатню кількість тестів;
- Кожен варіант використання покритий тестами;
- Усі тести посилаються на вимоги до програмного забезпечення.

Умови виконання тестів:

- Усі тести були виконані;
- Усі тести були успішно пройдені.

Так як усі згадані умови були виконанні в процесі тестування, можна сказати що програмний застосунок успішно розроблений.

Висновки до розділу 4

Під час роботи на даним розділом було розроблено усі елементи та компоненти веб-системи. Розроблення відбувалося в інтегрованому середовищі розробки IntelliJ IDEA як для серверної частини, так і для веб-частини системи. Розроблена база даних PostgreSQL. Основною технологією під час розроблення була Spring Framework та її модулі. Було використано шаблони проектування програмного забезпечення і підхід Code First.

Також можна виділити технологію Thymeleaf, яка стала основою веб-частини системи, а стилі використані у її шаблонах доступні при наявності з’єднання з мережею Інтернет.

Під час розроблення програмного забезпечення воно постійно покривалося тестами для забезпечення коректної роботи та виявлення помилок. Тестами була покрита уся функціональність програмної системи, що дає можливість стверджувати про правильну роботу системи. Всього було розроблено 14 тестових випадків, які були успішно пройдені на фінальному етапі розроблення.

Виходячи з цього, можна зробити висновок, що розроблення даного програмного забезпечення пройшло успішно.

РОЗДІЛ 5. ЕКОНОМІЧНА ЧАСТИНА

5.1. Економічна характеристика проектного рішення (програмного продукту)

Основна мета магістерської кваліфікаційної роботи є розроблення веб-системи для вивчення граматики та лексики іноземних мов на прикладі англійської. Такий вид системи дає можливість доєднатися до неї з любої точки світу через мережу Інтернет.

Користувачу надається багато можливостей для вивчення іноземної мови у веб-системі шляхом перегляду уроків та проходження тестування для визначення рівня закріплення матеріалу уроку, а також шляхом володіння віртуальним словником, запас якого може динамічно та інтерактивно змінюватися за бажанням користувача і бути повтореним за допомогою спеціального тренування лексики. Також користувачу надається можливість реалізувати власний досвід та творчі здібності шляхом написання своїх уроків у розділі граматики.

Для отримання повного доступу до функціоналу веб-систему користувач має бути зареєстрований та авторизований у систему. Дані усіх користувачів зберігаються на сервері та є у постійному доступі в режимі з’єднання через мережу Інтернет.

Головною перевагою даного програмного продукту є охоплення великої кількості класів користувачів, від вчителів та перекладачів до людей, які вивчають іноземну мову як хобі, а також до людей, професії яких не пов’язані з іноземними мовами, але вимагають знання певної мови. Також великою перевагою є абсолютна доступність веб-системи, адже для доступу до неї не накладено ніяких обмежень і потрібний тільки будь-який тип пристрою з виходом у мережу Інтернет та веб-браузером, а мінімальні технічні характеристики цих пристроїв незначні. Це дає змогу діяти на різних ринках.

З розвитком інформаційних технологій та мереж, завдяки повній та швидкій доступності до інформації з усього світу, люди у наш час переймають

певні особливості культур інших народів, а онлайн-спілкування з людьми з інших країн вже є звичним. Також освітня система має тенденцію до автоматизації процесу навчання. Усі ці чинники говорять про те, що програмний продукт є актуальним і повинний знайти багато покупців. За даний продукт з функціоналом такого об’єму покупці не будуть готові платити велику кількість коштів, проте, якщо зважати на кількість охоплення, а так як дана система розрахована на велику кількість класів користувачів і надає вирішення актуальний на даний момент проблем, система має можливість окупитися. Також окремо можна виділити такі характеристики системи, як мінімалістичний дизайн, відсутність реклами, інтуїтивно зрозумілий інтерфейс, адже завдяки ним реклама та просування даного програмного продукту мають бути легкими та ефективними.

Зважаючи на зазначені чинники, розроблення даного програмного забезпечення з економічної точки зору є більш ніж доцільним.

5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення товару

На ринку в даний момент існує велика кількість готових рішень для вивчення іноземних мов, як у вигляді веб-систем, так і у вигляді десктопних застосунків. Проте з розвитком сучасних технологій пік розвитку програмних продуктів для вивчення іноземних мов завжди збільшується і неможливо сказати, що він був досягнутий.

Адже не зважаючи на присутність на ринку великої кількості продуктів даного типу вони найчастіше не мають усього спектру необхідного функціоналу для комплексного вивчення іноземної мови. Але якщо проаналізувати темпи розвитку та кількість нових програмних продуктів у сфері вивчення іноземних мов, можна дійти висновку, що цей напрямок доволі перспективний і користується сьогодні великим попитом у суспільства .

Було проаналізовано декілька аналогічних продуктів на ринку для кращого розуміння ситуації:

1. Duolingo - це веб-сайт та мобільний додаток для вивчення іноземних мов. Надає можливість переглядати уроки з різноманітних тем іноземної мови та має окремий тренажер для тренування та закріплення матеріалу. Компанія використовує частково-безкоштовну модель: додаток та веб-сайт доступні безкоштовно, хоча Duolingo також пропонує преміум-послугу за окрему плату. Підтримує як десктопні пристрої, так і мобільні з операційними системами Android та iOS. Наявна внутрішня валюта, яка дає змогу налаштовувати персонажа або рівні бонусів.
2. LinguaLeo - це онлайн-платформа, що пропонує послугу з вивчення англійської мови для носіїв російської, португальської та бразильської мов. LinguaLeo персоналізує навчальну програму кожного користувача, щоб зробити вивчення англійської мови більш ефективним. Доступна версія для мобільних платформ. Також доступний до придбання за кошти користувача. преміум-профіль користувача для зняття обмежень з певних функцій.
3. Memrise – це веб-сайт та додаток для вивчення лексики іноземних мов. Сервіс працює за системою флеш-карт. У класичній формі це аркуш паперу, на якому на одній стороні написано слово чи фраза на вашій мові, а на іншій - переклад. Сервіс має певні обмеження на деякі функції, які можна зняти за кошти користувача. Також присутні функції які є строго платними.

Проаналізувавши існуючи альтернативи, можна стверджувати, що на даний момент існує потреба у програмних продуктах для вивчення іноземних мов, але наявні системи не задовольняють функціональних потреб користувачів, а також є надлишковими або вузькоспеціалізованими, що є критичним фактором у питанні створення продукту для вивчення іноземних мов.

5.3. Оцінювання та аналізування факторів зовнішнього та внутрішнього середовищ

Зовнішні фактори оцінюються в межах від -5 до 5. Чим ближча оцінка до значення -5, тим більший негативний вплив фактору на організацію. А чим ближча оцінка до значення 5, тим більший позитивний вплив фактору на організацію. Якщо ж оцінка фактору є близькою до 0, то фактор має нейтральний вплив на організацію.

Внутрішні фактори оцінюються в межах від 0 до 5. Чим ближча оцінка до значення 0, тим більше негативним є стан фактору. А чим ближча оцінка до значення 5, тим більше розвинутим є внутрішній фактор.

Для визначення рівня вагомості кожного фактору використовуються коефіцієнти, а сумою усіх вагомостей є одиниця. А для визначення зваженого рівня впливу факторів треба обчислити добуток рівня вагомості та впливу фактору у балах.

Оцінювання даних впливів можна бачити у табл. 5.1.

Таблиця 5.1

Результати експертних оцінок

Фактори	**Середня експертна оцінка, бали**	**Середня вагомість факторів**	**Зважений рівень впливу, бали**
1	2	3	4
Фактори зовнішнього середовища			
Споживачі	4	0,11	0,44
Постачальники	0	0,1	0
Конкуренти	-3	0,1	-0,3
Державні органи влади	-1	0,05	-0,05
Інфраструктура	0	0,06	0
Законодавчі акти	-1	0,1	-0,1
Профспілки, партії та інші громадські організації	0	0,05	0
Система економічних відносин в державі	0	0,06	0
Організації-сусіди	0	0,01	0
Міжнародні події	0	0,01	0
Міжнародне оточення	1	0,03	0,03
Науково-технічний прогрес	3	0,07	0,21
Політичні обставини	0	0,06	0
Соціально-культурні обставини	-1	0,05	-0,05
Рівень техніки та технологій	4	0,04	0,16
Особливості міжнародних економічних відносин	0	0,02	0
Стан економіки	-2	0,08	-0,16
Загальна сума	-	1	0,18
Фактори внутрішнього середовища			
Цілі	2	0,11	0,22
Структура	3	0,16	0,48
Завдання	2	0,07	0,14
Технологія	4	0,2	0,8
Працівники	1	0,21	0,21
Ресурси	2	0,25	0,5
Загальна сума	-	1	2,35

Після оцінювання факторів можна зробити висновок, що основними зовнішніми факторами є рівень техніки та технологій, науково-технологічний прогрес та споживачі.

Даний програмний застосунок охоплює усі класи людей, не зважаючи на їх вік, професію чи діяльність, і це дає змогу швидко поширювати його серед людей. Це можливо за допомогою особистих рекомендацій користувачів, а якість, доступність та легкість у використанні дають змогу бути впевненим, що програмний продукт буде швидко поширюватися і набирати популярність.

Окремо можна виділити зовнішні фактори рівень техніки та технологій, а також науково-технічний прогрес. У сфері систем вивчення іноземних мов, розвиток та використання нових технологій є вирішальним фактором для якісної оцінки системи і великим плюсом у порівнянні з конкурентами та аналогами.

Найбільш негативно впливає фактор конкурентів. За час використання технологій у сфері навчання з’явилося і продовжує з’являтися багато подібних продуктів для вивчення іноземних мов, тому найбільше уваги треба приділити унікальним характеристикам продукту у порівнянні з аналогами і загальній якості та актуальності зараз і в майбутньому.

Не зважаючи на це, після аналізу внутрішніх факторів, можна сказати що саме вони є домінуючими. Найбільш домінуючи: технології та структура. Якщо правильно вибрати технологічні інструменти, можна розробити систему, яка буде спроможна конкурувати з аналогами не тільки у функціональному плані, але і в плані стабільності та ефективності.

Структура, у свою чергу, є також важливим фактором, і при наявності правильно побудованої структури, коли усі її елементи логічно пов’язані між собою та утворюють зрозумілу загальну картину, можна говорити про якість програмного продукту.

Виходячи з усього вище згаданого можна дійти висновку, що існує потреба на такий тип продукту та є можливості виходу на ринок.

5.4. Формування стратегічних альтернатив

5.4.1. Стратегічні альтернативи, група №1

Можна виділити два критерія поділу альтернативних стратегій розвитку першої групи: існуючий продукт та новий. Також окремо виділяються супутні послуги.

Рис. 5.1. Стратегічні альтернативи, група №1

Критерій нового продукту – це створення абсолютно нового програмного забезпечення. Це означає що новий програмний продукт має вирішувати нові проблеми у своїй сфері застосування.

Критерій існуючого продукту – це удосконалення або певна модифікація програмного забезпечення, яке вже існує на ринку.

Якщо до удосконаленого програмного продукту додати супутні послуги, такі як супроводження або встановлення, в такому випадку критерій матиме назву стратегії розвитку існуючого продукту з супутніми послугами.

А якщо до нового програмного продукту додати супутні послуги, то критерій поділу альтернативних стратегій розвитку матиме назву стратегії нового продукту з супутніми послугами.

5.4.2. Стратегічні альтернативи, група №2

У другій групи стратегічних альтернатив можна виділити такі критерія поділу альтернативних стратегій розвитку: існуючий ринок та існуючий продукт, новий ринок та новий продукт.

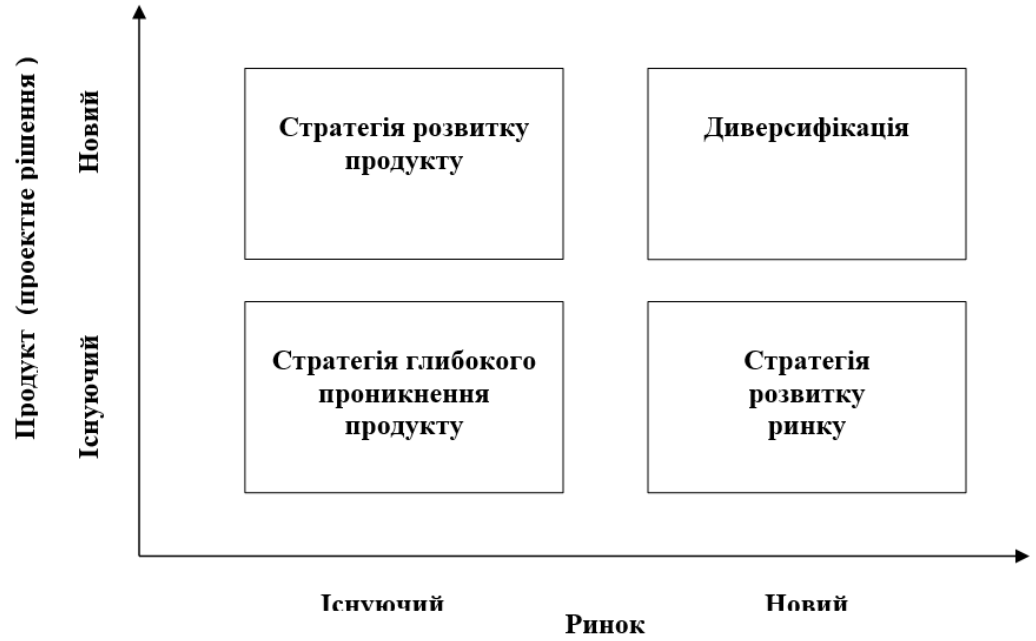

Рис. 5.2. Стратегічні альтернативи, група №2

Якщо йде пошук нових шляхів просування на вже існуючому ринку за допомогою вже існуючого проектного рішення, то така стратегічна альтернатива має назву стратегії глибокого проникнення продукту.

Ця стратегія має невеликі ризики, але при умові достатньої кількості ресурсів та потужностей. Проте ця стратегія передбачає зростання конкуренції і буде успішною тільки у випадку обмежених ресурсів та потужностей у конкурентів або у випадку зростання ринку.

Якщо стратегія полягає у використанні вже наявного продукту і його просуванні на новий ринок, вона має назву стратегії розвитку ринку. Дана стратегія є більш ризикованою ні стратегія глибокого проникнення продукту через непередбачуваність нового ринку.

Якщо ж стратегія полягає у створенні нового проектного рішення на вже існуючому ринку, то вона має назву стратегії розвитку продукту. Така стратегія

має також підвищений рівень ризику, але при певних умовах, таких як зменшення обсягів ринку, може бути доцільною.

Найбільш ризикованою є стратегія диверсифікації. Вона передбачає створення нового продукту на новому ринку.

5.4.3. Вибір стратегій

Після ознайомлення із стратегіями першої групи можна зробити висновок, що для веб-системи для вивчення граматики та лексики іноземних мов на прикладі англійської найбільше підходить стратегія розвитку нового продукту з супутніми послугами.

Такий тип стратегії був обраний через те, що веб-система вимагає постійного супроводження для реалізації покращень від користувачів, а також через розвиток технологій і методології підходів до вивчення іноземних мов, які мають якнайшвидше реалізовуватися у вигляді постійних модифікацій та доповнень продукту для можливості залишатися конкурентним та актуальним на ринку.

Щодо стратегій другої групи, найкращим варіантом є стратегія розвитку товару, так як ринок систем для вивчення іноземних мов вже не новий, але існуючі рішення не задовольняють усіх потреб користувачів. Дана стратегія надає можливість забезпечити популярність та попит на веб-систему для вивчення іноземних мов, не зважаючи на високий рівень ризику.

5.5. Бюджетування

Наступним етапом є бюджетування. Це є комплексно обґрунтована система, яка відповідає за розрахунок витрат. Ця система дає змогу проаналізувати витрати та розробити заходи для підвищення рентабельності. У результаті цього етапу буде визначено собівартість продукту та обґрунтовано доцільність вибору стратегії з економічної сторони.

Таблиця 5.2

Бюджет витрат матеріалів та комплектуючих виробів

Назва матеріалів та комплектуючих	**Марка, тип, модель**	**Фактична кількість, шт.**	**Ціна за одиницю, грн.**	**Разом, грн.**
Середовище бази даних	DataGrip	1	1500	1500
Середовище розроблення	IntelliJ IDEA	1	2400	2400
Разом:		2		3900

Таблиця 5.3

Бюджет витрат на оплату праці

Посада, спеціальність	**Кількість працівників, осіб**	**Час роботи, дні**	**Денна заробітна плата працівників, грн.**	**Сума витрат на оплату праці, грн.**
Основна заробітна плата				
Java Developer	1	60	1000	60000
Дизайнер	1	2	700	1400
Тестувальник	1	7	800	5600
Разом:	3	39	2500	67000

Таблиця 5.4

Бюджет обов’язкових відрахувань та податків

Посада, спеціальність	**Сума основної заробітної плати**	**Сума додаткової заробітної плати**	**Разом витрат на оплату праці**	**Сума податку з доходів фізичних осіб (18%), грн.**	**Сума військового збору, грн. (1,5%)**
1	**2**	**3**	**4**	**5**	**6**
Java Developer	60000	-	60000	10800	900
Дизайнер	1400	-	1400	252	21
Тестувальник	5600	-	5600	1008	84
Разом:	67000	-	67000	12060	1005

Таблиця 5.5

Бюджет загальновиробничих витрат

Статті витрат	**Сума, грн.**
Змінні загальновиробничі витрати, у т.ч.:	
- заробітна плата допоміжного персоналу;	4000
- нарахування на заробітну плату (ЄСВ 22%)	880
- витрати на МШП;	300
- витрати на електроенергію;	300
- витрати на ремонт;	-
- інші змінні витрати	-
Разом змінних витрат:	5480
Постійні загальновиробничі витрати, у т.ч.:	
- комунальні послуги;	400
- витрати на оренду;	1000
- інші постійні витрати (амортизація);	-
Разом постійних витрат:	
Разом загальновиробничих витрат:	1400

Таблиця 5.6

Бюджет адміністративних витрат та витрат на збут

Статті витрат	**Сума, грн.**
1	2
Адміністративні витрати, у т.ч.:	
- заробітна плата адміністративного персоналу;	2000
- нарахування на заробітну плату (ЄСВ 22%)	440
- витрати на МШП;	200
- витрати на відрядження;	-
- витрати на ремонт;	-
- витрати на паливно-мастильні матеріали;	-
- знос адміністративного обладнання;	-
- інші адміністративні витрати;	-
Разом адміністративних витрат:	2640
Витрати на збут, у т.ч.:	
- заробітна плата менеджерів зі збуту;	2000
- нарахування на заробітну плату (ЄСВ 22%)	440
- витрати на гарантійний ремонт;	-
- витрати на відрядження;	-
- витрати на гарантійне обслуговування;	-
- витрати на налагодження і експлуатацію;	-
- витрати на паливо-мастильні матеріали;	-
- витрати на рекламу;	3000
- інші витрати на збут;	-
Разом витрат на збут:	5440

Таблиця 5.7

Зведений кошторис витрат на розробку проектного рішення (продукту)

Статті витрат	**Одиниці виміру**	**Фактична кількість, шт.**	**Ціна одиниці, грн.**	**Разом, грн.**
Сировина і матеріали	шт			
Купівельні напівфабрикати та комплектуючі вироби	шт	4	-	3900
Зворотні відходи (вираховуються)	грн	-	-	-
Основна заробітна плата	грн	6	-	75000
Додаткова заробітна плата	грн	-	-	-
Відрахування на соціальне страхування (ЄСВ 22%)	грн	6	-	14828
Витрати на утримання й експлуатацію устаткування	грн	-	-	-
Загальновиробничі витрати, у т.ч.:				
- змінні;	грн	4	-	5480
- постійні;	грн	2	-	1400
Разом виробничих витрат:	грн	6	-	6880
Адміністративні витрати	грн	3	-	4104
Витрати на збут	грн	3	-	13270
Інші операційні витрати	грн	-	-	-
Разом виробничих і операційних витрат:	грн	34	-	124862

Для обчислення фінансових результатів потрібно визначити ціну програмного продукту. Для цього треба врахувати рентабельність виробництва 34% до виробничих та операційних витрат.

Ціна продукту буде наступною:

Ц = СБ * Р + СБ = 124862 * 0,34 + 124862 = 167315 грн

Таблиця 5.8

Бюджет фінансових результатів

Показники	**Сума, грн.**
1	2
Дохід від реалізації продукції	167315
Податок на додану вартість (0%)	0
Чистий дохід від реалізації продукції	167315
Собівартість реалізованої продукції	124862
Валовий прибуток	42453
Операційні витрати:	
- адміністративні витрати	2640
- витрати на збут	5440
- інші операційні витрати	0
Фінансовий результат від операційної діяльності	34373
Податок на прибуток (18%)	6187
Чистий прибуток (збиток)	28186

Висновки до розділу 5

На основі проведених обрахунків можна стверджувати, що обрана стратегія розвитку нового продукту з супутніми послугами, а також стратегія розвитку товару являються доцільними, так як чистий прибуток становить 28186 грн.

Програмний продукт, який розробляється, не є чимось новим на ринку, так як на ньому існує багато систем аналогів. З економічної точки зору та після проведення аналізу розрахованих значень можна зробити висновок, що програмний продукт є економічно вигідним і доцільним.

ВИСНОВКИ

В результаті виконання магістерської кваліфікаційної роботи було розроблено веб-систему для вивчення іноземних мов. А саме система дозволяє вивчати лексику та граматику іноземних мов на прикладі англійської.

Спочатку було описано предметну область вивчення іноземної мови і дано поняття про автоматизовані навчальні системи. Завдяки цим процесам було відзначено, що в процесі вивчення іноземної мови лексика та граматика найкраще підходять для автоматизації під час вивчення. А використання веб-системи дає додаткові можливості для взаємодії. Також були проаналізовані існуючі рішення аналоги.

Наступним етапом була загальна постановка задачі. Було з’ясовано що система повинна мати два модулі: граматика та лексика, основними функціями яких були уроки граматики та словник відповідно. Також було обрано інструментальні засоби та технології. Вирішено, що основою системи стане Spring Framework та його модулі. Завершальним пунктом цього етапу було створення специфікації вимог до веб-системи.

У третьому розділі була спроектована веб-система та її архітектура. Були створені діаграми розгортання та прецедентів. Також спроектована структура проекту та його основні компоненти. Окремо була спроектована база даних, а саму фізична та логічна її моделі. Був обраний підхід Code First для генерації таблиць та їх зв’язків.

Під час реалізації та тестування веб-системи спочатку був описаний процес реалізації веб-системи. Були створенні компоненти трьох шарів серверного застосунку, а саме: класи-сутності та інтерфейси-репозиторії на рівні з’єднання з базою даних, класи-сервіси на рівні бізнес-логіки та класи-контролери на рівні запитів та відповідей протоколу HTTP. Після створення компонент серверної частини були створені компоненти веб-частини. Створено шаблони сторінок Thymeleaf, CSS-стилі та скрипти мови JavaScript. Також описано процес взаємодії користувача з програмним продуктом. Після

завершення реалізації застосунку був проведений процес тестування програмного забезпечення. Описані варіанти використання та тестові випадки до них. Тестування програмного забезпечення пройшло успішно, а всі тести також були успішно пройдені.

В останньому розділі проаналізована економічна частина. Після її аналізу можна зробити висновок, що розроблене програмне забезпечення є економічно вигідним та може бути конкурентним на ринку. Основною перевагою даного програмного продукту можна виділити велику кількість класів користувачів та незалежність від платформи та пристрою користувача. Дані переваги дозволяють діяти на різних ринках. Також були обрані стратегії розвитку програмного продукту.

В результаті проведеної роботи її результат можна вважати успішним, так як всі етапи завершилися успішно, що говорить про технологічність, надійність та конкурентоспроможність системи.

СПИСОК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Граматика – [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0 (2021)
2. Лексика і лексикологія [Електронний ресурс] – Режим доступу: https://ukrainskamova.com/publ/chinnij_pravopis/leksika/leksika_i_leksikologija/5-1-0-44 (2021)
3. Автоматизована навчальна система – [Електронний ресурс] – Режим доступу: http://ito.vspu.net/ENK/KIt%20v%20osviti%20i%20nauk/slovnuk/ANS.htm (2021)
4. 7 Outstanding Language-Learning Apps and Websites – [Electronic Resource] – Web page: https://www.huffpost.com/entry/7-outstanding-language-le_b_6431448 (2021)
5. 12 Best Free Language Learning Websites of 2021 – [Electronic Resource] – Web page: https://www.lifewire.com/best-free-language-learning-websites-1357061 (2021)
6. How to Learn a New Language at Home, According to Language Experts – [Electronic Resource] – Web page: https://nymag.com/strategist/article/how-to-learn-languages-at-home.html (2021)
7. Language for life – [Electronic Resource] – Web page: https://www.babbel.com/ (2021)
8. Навчальна система – [Електронний ресурс] – Режим доступу: https://dbn.co.ua/blog/navchalna_sistema/2016-12-06-22698 (2021)
9. Система навчання: диференційний підхід. Реферат – [Електронний ресурс] – Режим доступу: https://osvita.ua/vnz/reports/pedagog/14110/ (2021)
10. КОМП'ЮТЕРНІ МЕТОДИ В СИСТЕМІ ОСВІТИ – [Електронний ресурс] – Режим доступу: https://studme.com.ua (2021)

11. Automation in education: how to streamline your educational processes– [Electronic Resource] – Web page: https://blog.airslate.com (2021)
12. Патерни проектування – [Електронний ресурс] – Режим доступу: https://refactoring.guru/uk/design-patterns (2021)
13. Фрімен Е. Патерни проєктування : Фабула, 2004. – 672 с.
14. Herbert Schildt - Java The Complete Reference, 10th Edition, 2017 – 1344 pages
15. Spring Framework Documentation – [Electronic Resource] – Web page: https://docs.spring.io/spring-framework/docs/current/reference/html (2021)

Додаток А. Код класу-сутності

```
package com.dinocco.dyplom.persistence;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.UniqueConstraint;

@Data
@Entity
@Table(uniqueConstraints={
        @UniqueConstraint(columnNames = {"author_id",
"title"})
})
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Lesson {

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

    @Column(nullable = false, length = 100)
    private String title;

    @Column(nullable = false, length = 1000)
    private String description;

    @Column(nullable = false)
    private Short difficulty;

    @Column(nullable = false, columnDefinition = "TEXT")
    private String content;

    @ManyToOne
    @JoinColumn(nullable = false)
    private User author;
}
```

Рис. А.1. Код класу-сутності

Додаток Б. Код класу-контролера

```
package com.dinocco.dyplom.controller;

import com.dinocco.dyplom.persistence.Lesson;
import com.dinocco.dyplom.service.LessonService;
import com.dinocco.dyplom.service.QuestionService;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
@RequiredArgsConstructor
public class LessonController {

    private final LessonService lessonService;
    private final QuestionService questionService;

    @GetMapping("/grammar")
    public String getLessons(@RequestParam(required = false)
Integer page, Model model) {
        model.addAttribute("lessonsView",
lessonService.getLessons(page));
        return "grammar";
    }

    @GetMapping("/lessons/{id}")
    public String getLesson(@PathVariable Long id, Model model) {
        Lesson lesson = lessonService.getLesson(id);
        model.addAttribute("lesson", lesson);
        model.addAttribute("hasTest",
questionService.hasLessonTest(id));
        model.addAttribute("triedTest",
lessonService.existsTestResultByLesson(lesson));
        return "lesson";
    }

    @GetMapping("/lessons/{id}/test")
    public String getTest(@PathVariable Long id, Model model) {
        Lesson lesson = lessonService.getLesson(id);
        model.addAttribute("lesson", lesson);
        model.addAttribute("questionViews",
questionService.getLessonQuestionViews(lesson));
        return "test";
    }

    @GetMapping("/lessons/{id}/test/result")
    public String getTestResult(@PathVariable Long id, Model
model) {
        Lesson lesson = lessonService.getLesson(id);
        model.addAttribute("lesson", lesson);
        model.addAttribute("testResult",
lessonService.getTestResult(lesson));
        model.addAttribute("maxScore",
lessonService.getMaxScore(lesson));
        return "test_result";
    }

}
```

Рис. Б.1. Код класу-контролера

Додаток В. Опис розроблених тестів

Таблиця В.1

LNG-1: Тестування функціоналу уроків

Summary: Перевіряє роботу логіки уроків
Tests:
1. Створення уроку 2. Видалення уроку

Резолюція тестових випадків - **Passed**

Таблиця В.2

LNG-2: Тестування функціоналу питань тестів

Summary: Перевіряє роботу логіки питань
Tests:
1. Створення питання 2. Чи має урок тест

Резолюція тестових випадків – **Passed**

Таблиця В.3

LNG-3: Тестування функціоналу варіантів відповідей на питання

Summary: Перевіряє роботу логіки варіантів
Tests:
1. Створення варіанту 2. Отримання варіанту

Резолюція тестових випадків – **Passed**

Таблиця В.4

LNG-4: Тестування функціоналу користувачів

Summary: Перевіряє роботу логіки користувачів
Tests:
1. Авторизація 2. Реєстрація

Резолюція тестових випадків - **Passed**

Таблиця В.5

LNG-5: Тестування функціоналу слів

Summary: Перевіряє роботу логіки слів
Tests:
1. Створення слова 2. Видалення слова

Резолюція тестових випадків - **Passed**

Додаток Г. Інструкція користувача

Г.1. Компоненти програмного забезпечення

Програмне забезпечення складається з двох основних компонентів: серверу та бази даних.

Сервер написаний на мові програмування Java. Дане програмне забезпечення є незалежним від платформи тому може працювати на всіх машинах та операційних системах з доступом до мережі Інтернет.

Мінімальні вимоги до сервера є наступними: 2 Гб оперативної пам’яті, частота процесора 1 Гц, 500 Мб вільного місця на диску.

У ролі бази даних виступає PostgreSQL, тому для роботи серверу потрібно встановити відповідне програмне забезпечення і створити базу даних із назвою dyplom. Для взаємодії з програмною системою користувачу потрібен будь-який браузер.

Таблиця Г.1

Файли, які необхідні для роботи

№1	Файл	Призначення	Належить проекту
1	dyplom-1.0.jar	Виконавчий файл	Веб-система для вивчення лексики та граматики іноземних мов на прикладі англійської

Г.2. Встановлення програмного забезпечення

Для встановлення програмного забезпечення потрібно скопіювати файл dyplom-1.0.jar на диск та створити директорію logs на рівні з ним. Так як використана мова програмування Java, для роботи програмного забезпечення потрібно встановити технологію Java Runtime Environment не нижче 11 версії.

Для запуску веб-системи потрібно у директорії з файлом dyplom-1.0.jar виконати команду java -jar dyplom-1.0.jar. Користувач, від імені якого відбувається запуск, має мати дозвіл на створення та редагування файлів в директорії logs.

Г.3. Базові функції програмного забезпечення

Основними функціями програмного забезпечення є: перегляд уроків, створення уроків, проходження тестів уроків, створення власного словника слів та тренування слів словника.

Перейшовши на сторінку веб-системи вперше, користувачу доступно дві функції: авторизація та реєстрація. При спробі перейти на будь-яку іншу сторінку веб-системи користувача буде переадресовано на сторінку авторизації.

Так як користувач вперше перейшов на сторінку веб-системи, він має зареєструватися. Після реєстрації користувача буде переадресовано на сторінку граматики.

На сторінці граматики користувач може бачити усі уроки доступні у системі. При натисканні на урок користувач перейде на сторінку уроку. Також на сторінці граматики є посилання для переходу на сторінку створення уроку. На сторінці уроку, якщо урок містить тестування, присутня кнопка переходу на тестування уроку.

Після завершення тестування користувачу буде виведений його результат. Коли користувач створює урок, він може вказати назву, опис, складність та текст уроку. Також користувач може додати питання до уроку, що автоматично буде означати, що урок має тест.

На кожній сторінці у верхній частині наявна шапка сайту, яка містить посилання на модулі граматики та лексики. Також справа кожної сторінки є ім’я користувача та кнопка для виходу.

На сторінці лексики користувачу доступний його словник, який містить усі слова додані користувачем. Також на цій сторінці користувач може додати нове слово або перейти до тренування. Тренування це процес повторення слів

наявних у словнику користувача. У цьому процесі користувачу задається слово та чотири варіанти перекладу. Слова, які були повторені останній раз найдавніше або взагалі ще не булі повторені, мають найвищий пріоритет при виборі слова для повторення.

Також, якщо увійти під користувачем, який має права адміністратора, наявні дві додаткові функції: видалення користувача та уроку. Для видалення користувача потрібно ввести його логін біля кнопки виходу. Для видалення уроку потрібно перейти на список уроків та натиснути на кнопку видалення біля певного уроку або натиснути на кнопку видалення на сторінці уроку.

Г.4. Аналіз помилок та можливих проблем

Найчастіше проблеми з даним програмним забезпеченням пов’язані з відсутністю доступу до мережі Інтернет. Для вирішення даного типу проблеми треба перевірити з’єднання з мережею Інтернет і у випадку відсутності доєднатися до неї. Також проблеми можуть бути пов’язані з браузером користувача. У цьому випадку треба перезавантажити сторінку веб-системи або сам браузер. Додатково можна доєднатися до системи з іншого браузера.

При виникненні інших типів помилок системою передбачено їх виведення в верхньому лівому куті. Також для пошуку причини помилки можна переглянути лог-файли.