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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка застосунку для вивчення японської мови з використанням мови Python та фреймворків Django, Angular»

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

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

_____ Андрій СОКОЛОВСЬКИЙ
(підпис)

Виконав: здобувач вищої освіти групи ПД-42

Андрій СОКОЛОВСЬКИЙ

Керівник: Вікторія ЖЕБКА
д.т.н., професор

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Соколовському Андрію Вадимовичу

1. Тема кваліфікаційної роботи: «Розробка застосунку для вивчення японської мови з використанням мови Python та фреймворків Django, Angular»
керівник кваліфікаційної роботи д.т.н., професор Вікторія ЖЕБКА,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про методи розробки веб-застосунків, опис роботи веб-додатків, документація фреймворків Django та Angular.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд застосунку для вивчення японської мови та аналіз існуючих аналогів, визначення вимог до програмного забезпечення.
2. Огляд та аналіз засобів реалізації застосунку для вивчення японської мови.
3. Проєктування архітектури застосунку для вивчення японської мови.

4. Програмна реалізація та тестування застосунку для вивчення японської мови.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма розгортання.
5. Діаграма варіантів використання
6. Діаграма класів
7. Карта сайту
8. Екранні форми.
9. Демонстрація роботи застосунку
10. Апробація результатів дослідження

6. Дата видачі завдання «28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-12.03.2024	
3	Огляд та аналіз застосунку для вивчення японської мови	13.03-15.03.2024	
4	Огляд та аналіз засобів реалізації застосунку для вивчення японської мови	16.03-21.03.2024	
5	Проєктування архітектури застосунку для вивчення японської мови	22.03-03.04.2024	
6	Програмна реалізація та тестування застосунку для вивчення японської мови	04.04-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Андрій СОКОЛОВСЬКИЙ
(підпис)

Керівник
кваліфікаційної роботи _____ Вікторія ЖЕБКА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 48 стор., 1 табл., 21 рис., 11 джерел.

Мета роботи – зменшення часу на вивчення японської мови шляхом розробки застосунку з використанням мови Python та фреймворків Django, Angular.

Об'єкт дослідження – процес вивчення японської мови.

Предмет дослідження – програмне забезпечення для вивчення японської мови.

Короткий зміст роботи: В роботі проаналізовано предметну галузь та методи реалізації застосунку для вивчення японської мови. Проведено огляд вже існуючих засобів для вивчення японської мови: LingoDeer, Duolingo, Tofugu. Розроблено вебсайт та програмно реалізовані основні функціональні можливості, такі як: авторизація, бібліотека курсів та словникових колод, система їх пошуку та фільтрації, сторінка проходження курсу, чат-бот асистент, аналіз речень, системи повторення слів та швидкий інтерфейс. Проведено модульне тестування застосунку. В роботі використано фреймворки Django та Angular для розробки бекенду та фронтенду відповідно, OpenAI API для реалізації функціональності чат-бота та аналізу речень, PostgreSQL для зберігання даних.

Сферою використання застосунку є організація самостійної роботи користувачів в процесі вивчення японської мови.

КЛЮЧОВІ СЛОВА: BACK-END, FRONT-END, PYTHON, DJANGO, ANGULAR, RESTful API, OPENAI API, ВЕБДОДАТОК.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – Програмне забезпечення

IDE – Integrated Development Environment

SRS – Spaced Repetition System

API – Application Programming Interface

SPA – Single Page Application

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

SQL – Structured Query Language

ORM – Object-Relational Mapping

DI – Dependency Injection

WSGI – Web Server Gateway Interface

ASGI – Asynchronous Server Gateway Interface

JSON – JavaScript Object Notation

ВСТУП

Актуальність: вивчення іноземних мов, зокрема японської, є важливим аспектом сучасної освіти, що допомагає встановлювати культурні та професійні зв'язки у глобалізованому світі. З розвитком технологій можливості для самостійного вивчення мов зростають, а засоби цифрової освіти стають все більш доступними та ефективними. Це дозволяє людям отримувати знання в зручний час і у комфортних умовах, використовуючи сучасні інструменти та ресурси. Однак одним з найзручніших способів організації навчання іноземній мові є вебсайт. Він надає можливість вивчати мову в індивідуальному темпі, контролюючи прогрес та повторюючи матеріал при необхідності.

Вебсайт постає як всебічний пакет усіх необхідних інструментів для вивчення японської мови в одному місці. На ньому можуть бути представлені курси, модулі, уроки та навчальні матеріали, що забезпечують комплексний підхід до вивчення. Цифрові інструменти, такі як вправи, SRS-система та чат-бот асистент [1], допомагають студентам зміцнити свої знання та покращити навички. Більш того, веб-платформа дозволяє користувачам легко проходити різні теми та уроки, обираючи ті, які найбільше відповідають їхнім потребам і рівню знань. Завдяки цьому, вивчення японської мови стає більш цікавим, гнучким та ефективним процесом.

Об’єктом дослідження є процес вивчення японської мови.

Предметом дослідження є програмне забезпечення для вивчення японської мови.

Метою роботи є зменшення часу на вивчення японської мови шляхом розробки застосунку з використанням мови Python та фреймворків Django, Angular.

Методи дослідження: першим кроком було проаналізовано існуючі рішення для вивчення японської мови для формулювання початкових функціональних та нефункціональних вимог до свого програмного

забезпечення. Далі було проведено огляд стеку технологій реалізації застосунку, що будуть відповідати вимогам до програмного забезпечення, і було обрано фреймворк Angular для клієнтської складової вебсайту, Django для серверної частини, PostgreSQL як об’єктно-реляційну систему бази даних, GitHub для системи контролю версій, OpenAI API для реалізації чат-бот асистенту [2] та Figma для прототипування інтерфейсу користувача. Дослідження реалізації застосунку проводилося під час безпосередньої розробки.

Наукова новизна роботи визначається наступними функціональними складовими: вбудована SRS-система; чат-бот асистент з вивчення японської мови; інструмент аналізу речень з подальшим поясненням їх складових. Вебсайт постає як всебічний пакет усіх найбільш затребуваних інструментів в одному місці.

Практична значущість результатів полягає в можливості використання реалізованого застосунку для ефективного навчання японській мові на більшості пристроїв.

1 АНАЛІЗ ЗАСТОСУНКУ ДЛЯ ВИВЧЕННЯ ЯПОНСЬКОЇ МОВИ

1.1 Вебсайт для вивчення японської мови

Вебсайт для вивчення японської мови — це спеціалізований онлайн ресурс, який надає доступ до різноманітних навчальних матеріалів і інструментів, спеціально розроблених для вивчення японської мови. Такі вебсайти допомагають користувачам опановувати японську через інтерактивні курси, відеоуроки, аудіоматеріали, та практичні вправи. Основна мета вебсайту для вивчення японської полягає у забезпеченні ефективного та доступного способу навчання, що дозволяє користувачам вивчати мову незалежно від їхнього місцезнаходження чи часових обмежень.

Основні компоненти вебсайту для вивчення японської мови включають:

– Мовні курси: структуровані за рівнями складності та тематиками, включають граматику, лексику, та культурні аспекти японської мови.

– Мультимедійні матеріали: використання відео, аудіо, та текстових діалогів для занурення у японське мовне середовище.

– Інтерактивні інструменти: ігри, квізи, і чат-боти для практики розмовних навичок і закріплення матеріалу.

– Системи оцінювання: онлайн тести та самоперевірки для відстеження прогресу у вивченні японської мови.

– Персоналізація навчання: можливість адаптувати курси та вправи до індивідуальних потреб та інтересів користувача.

Основні цілі вебсайту для вивчення японської мови:

– покращення розуміння мови;

– навчання нових слів і фраз;

– поглиблення граматичних знань;

– розвиток навичок спілкування;

– занурення у японську культуру;

– підготовка до іспитів;

– оцінка прогресу.

Оцінимо переваги та недоліки вебсайту як засобу для вивчення японської мови.

Переваги:

– Доступність: вебсайти забезпечують можливість вивчати японську з будь-якої точки світу, де є доступ до інтернету. Це ідеально підходить для людей, які живуть у віддалених або сільських районах.

– Гнучкість: користувачі можуть навчатися у власному темпі та вибирати час для занять, що є неможливим у традиційних класних умовах.

– Різноманіття матеріалів: велика кількість ресурсів, таких як відео, аудіо записи, текстові матеріали та інтерактивні завдання, допомагають краще засвоїти мову.

– Індивідуалізація: багато вебсайтів пропонують персоналізовані шляхи навчання, які адаптуються до рівня знань та потреб користувача.

– Вартість: зазвичай навчання на вебсайтах коштує дешевше, ніж участь у традиційних курсах або заняттях з репетитором.

Недоліки:

– Відсутність особистого контакту: навчання онлайн може ускладнювати реальне спілкування з викладачами та іншими студентами, що може вплинути на мотивацію та залученість.

– Технічні проблеми: потреба у стабільному інтернет-з'єднанні та сучасному обладнанні може стати перешкодою для деяких користувачів.

– Самодисципліна: самостійне навчання вимагає високого рівня самодисципліни та організації, що може бути складно для деяких студентів.

– Якість контенту: не всі вебсайти забезпечують високоякісний освітній контент, і користувачам часто потрібно витрачати час на пошук надійних джерел.

– Обмеження практичного спілкування: вивчення японської мови онлайн може обмежувати практичне спілкування з носіями мови, що є ключовим

для розвитку розмовних навичок.

1.2 Специфіка вивчення японської мови.

Вивчення японської мови має ряд особливостей, які роблять процес вивчення відмінним від вивчення інших іноземних мов. Розглянемо їх.

Система писемності:

– Використовуються три різні системи писемності: хірагана, катакана та кандзі.

– Хірагана та катакана складаються з 46 базових символів кожна, а кандзі має кілька тисяч символів китайського походження.

– Для досягнення базового рівня володіння необхідно знати мінімум 1000 кандзі.

Граматична структура:

– Відмінності в порядку слів: підмет, додаток, присудок (Subject-Object-Verb).

– Частки: маленькі слова, які позначають відносини між частинами речення.

– Суфікси: додаються до дієслів та прикметників для відмінювання, ввічливості та часу.

Фонетика:

– Звуки, відсутні в українській мові, наприклад, звуки つ (tsu) та し (shi).

– Інтонація відіграє важливу роль у розрізненні значень слів.

Культурні особливості:

– Розвинена система форм ввічливості, від розмовної до дуже офіційної.

– Важливість групової гармонії та непряме вираження думок.

1.3 Цільова аудиторія

Застосунок для вивчення японської мови орієнтований на широке коло користувачів, які відрізняються за своїми навчальними потребами та цілями. До основних категорій користувачів належать:

– Студенти мовних спеціальностей - студенти університетів та коледжів, які вивчають японську мову як основну або додаткову спеціальність. Їх мета - поглиблене вивчення японської мови для підготовки до іспитів та професійної діяльності.

– Ентузіасти та самонавчальники - люди, які вивчають японську мову для особистих або професійних цілей без доступу до формальної освіти. Їх мета - розширити знання японської мови та покращити навички спілкування.

– Особи, що готуються до іспитів JLPT (Japanese Language Proficiency Test) - студенти та професіонали, які прагнуть скласти іспити JLPT різних рівнів для отримання сертифіката володіння японською мовою. Їх мета - підвищити рівень володіння мовою для успішного складання іспитів.

1.4 Дослідження існуючих аналогів

На сьогоднішній день одними з найпопулярніших вебсайтів для вивчення японської мови є Duolingo, Tofugu та LingoDeer. Розглянемо їх детальніше.

1.4.1 Duolingo

Duolingo — це популярний онлайн-застосунок для вивчення іноземних мов, який дозволяє користувачам освоювати нові мови у форматі гри. Застосунок вирішує проблему доступу до якісного та інтерактивного навчання, надаючи користувачам ефективний спосіб вивчати мови безкоштовно. Його підхід полягає у поєднанні коротких уроків з мотиваційною системою балів та досягнень, що підтримує інтерес до навчання.

Основні функції Duolingo включають:

– Інтерактивні уроки: курси розбиті на маленькі, зручні для сприйняття уроки.

– Тестування навичок: система оцінювання знань на кожному етапі навчання.

– Вправи на практику: завдання на переклад, вимову та розуміння мови.

– Щоденні виклики: мотивація утримувати щоденну серію вправ.

– Візуальні події: ігрові елементи, такі як віртуальні монети, рівні, і досягнення.

– Персоналізація навчання: адаптація курсів під особистий рівень та темп користувача.

– Мовні сертифікати: можливість складання тесту для отримання сертифіката про знання мови.

– Багатомовна підтримка: доступність курсів на багатьох мовах.

– Мобільність: додаток доступний для iOS та Android, що дозволяє навчатися в дорозі.

– Форуми спільноти: можливість обговорення мовних питань із іншими користувачами.

Переваги:

– Інтерактивний підхід до навчання: уроки організовані у форматі коротких вправ із використанням системи балів та досягнень, що мотивує користувачів до постійного навчання.

– Гейміфікація: мотиваційна система, включаючи бали, нагороди та щоденні серії, робить процес навчання схожим на гру, підтримуючи інтерес користувачів.

– Доступність на різних платформах: Duolingo доступний на мобільних та настільних платформах, забезпечуючи навчання в будь-який час та з будь-якого пристрою.

Недоліки:

– Обмежений набір уроків для середнього та просунутого рівня: Duolingo здебільшого орієнтований на початковий рівень знань і не має достатньо

матеріалів для середнього та просунутого рівня.

– Недостатня глибина вивчення граматики: уроки граматики переважно базові та не охоплюють складніші аспекти граматики японської мови.

– Відсутність індивідуалізації навчальних планів: уроки та вправи не адаптуються під індивідуальні потреби кожного користувача.

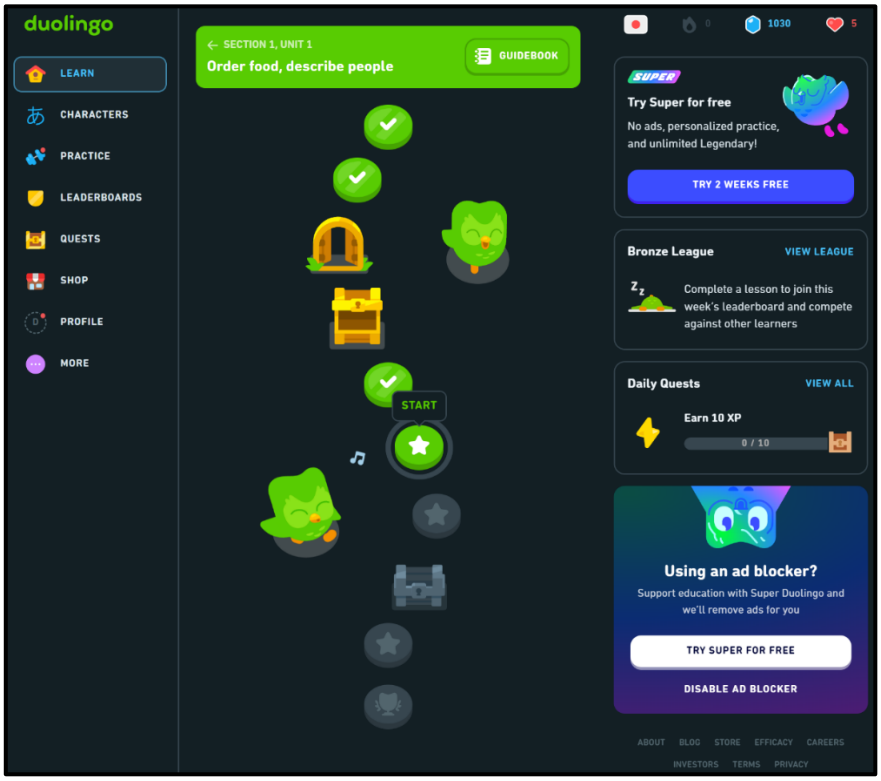


Рис. 1.1 Приклад сторінки уроків та вправ Duolingo

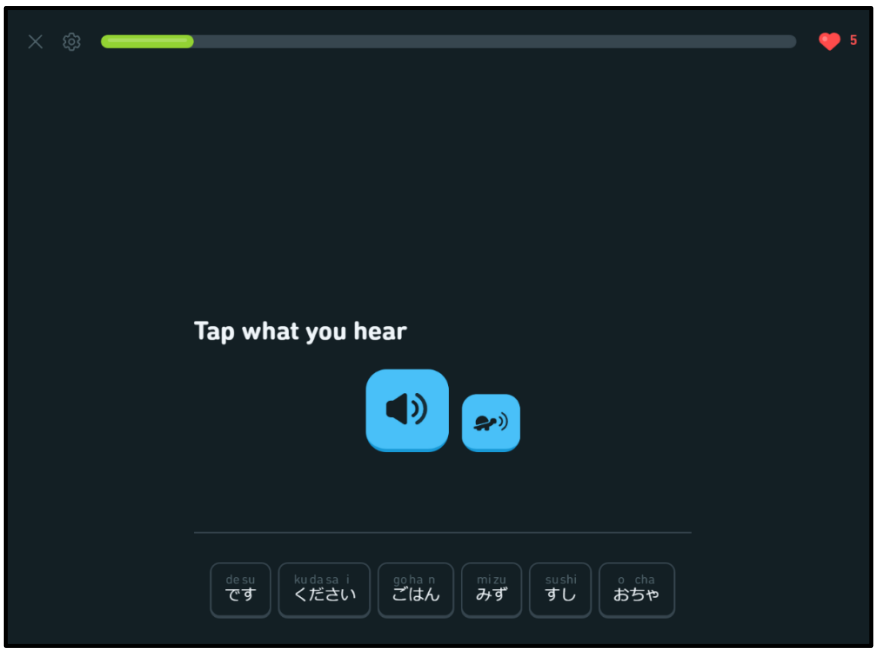


Рис. 1.2 Приклад сторінки граматичних вправ Duolingo

1.4.2 Tofugu

Tofugu — це ресурс для глибокого вивчення японської мови та культури, який пропонує детальні статті, поради та онлайн-матеріали для ентузіастів японської. Він зосереджений на забезпеченні користувачів глибоким розумінням японської мови, зокрема складнощів і викликів, які не завжди покриваються в традиційних мовних курсах. Основна стратегія Tofugu полягає в наданні якісних, обґрунтованих і практично корисних матеріалів, які сприяють залученості та мотивації у навчанні.

Основні функції Tofugu включають:

– Детальні публікації: статті, які покривають широкий спектр тем від граматики до культурних особливостей.

– Ресурси для навчання: посібники та інструкції по вивченню японських ієрогліфів, граматики та розмовних фраз.

– Поради щодо навчання: керівництва та стратегії для ефективного вивчення японської мови.

– Культурні інсайти: глибоке занурення у японську культуру, традиції та сучасне життя.

– Відеоматеріали: навчальні відео, які допомагають краще зрозуміти нюанси вимови та мовлення.

– Подкасти: аудіоматеріали для покращення слухового сприйняття та розуміння японської мови на слух.

– Спільнота та форуми: можливість обговорення питань, обміну досвідом та пошуку відповідей на запитання.

Переваги:

– Глибоке занурення у мову та культуру: Tofugu надає не тільки мовні уроки, але й детальні описи культурних аспектів, що дозволяє користувачам отримати більш комплексне розуміння Японії.

– Високоякісний контент: статті та інші матеріали на Tofugu відзначаються високим стандартом написання та глибиною дослідження.

– Різноманітні навчальні ресурси: від статей до відео та подкастів,

Tofugu пропонує широкий спектр ресурсів для всіх типів учнів.

– Спільнота однодумців: форуми та коментарі на сайті дають можливість спілкування з іншими студентами та експертами.

– Практичні поради та стратегії: посібники та рекомендації з ефективного вивчення японської, що підходять як новачкам, так і досвідченим учням.

Недоліки:

– Складність матеріалів: деякі матеріали можуть бути занадто складними для повних початківців, що вимагає попереднього рівня знань для повного розуміння.

– Обмежена інтерактивність: в порівнянні з іншими мовними платформами, Tofugu має менше інтерактивних вправ, що може ускладнити процес вивчення для деяких учнів.

– Відсутність структурованих курсів: Tofugu не пропонує традиційних мовних курсів з послідовними рівнями, що може ускладнити відстеження прогресу.

– Потреба в самостійній мотивації: без ігрових елементів та структурованої системи навчання, користувачам необхідно самостійно підтримувати власну мотивацію.

– Обмеженість мобільного доступу: хоча сайт можна переглядати на мобільних пристроях, він не має спеціалізованого мобільного додатку, що може зменшити зручність використання в дорозі.

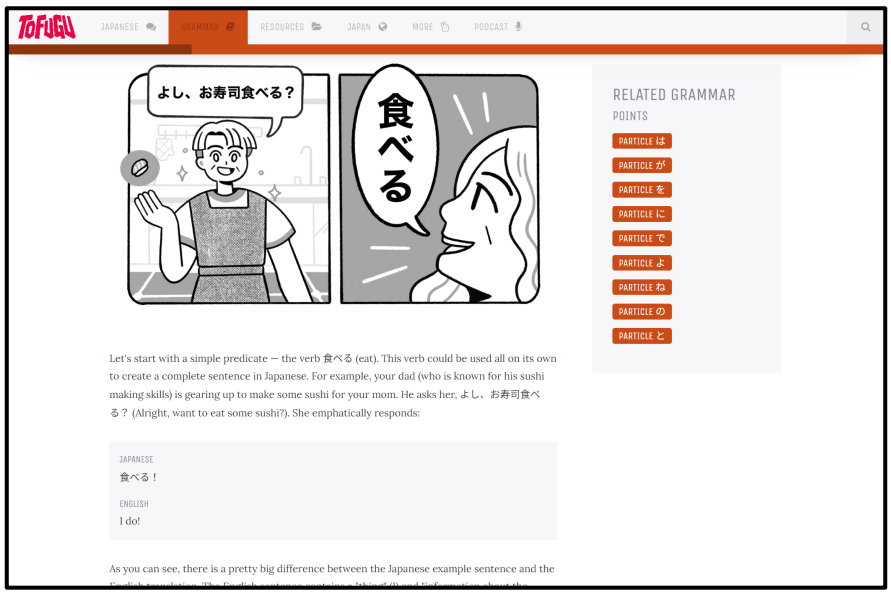


Рис. 1.3 Приклад сторінки уроку Tofugu

1.4.3 LingoDeer

LingoDeer — це спеціалізований онлайн-застосунок для вивчення східноазіатських мов, включаючи японську. Він вирішує проблему браку ефективних ресурсів для вивчення японської мови на всіх рівнях, надаючи користувачам комплексний підхід до навчання. Основна мета LingoDeer — забезпечити користувачів інструментами для освоєння японської мови від початкового до просунутого рівня.

LingoDeer пропонує користувачам комплексний набір функцій для ефективного вивчення японської мови:

– Структура уроків: уроки організовані в тематичні модулі, кожен з яких охоплює різні аспекти японської мови, такі як граматика, словниковий запас, аудіювання та розмовна практика.

– Тематика уроків: базові, середні та просунуті теми, включаючи «Привітання», «Транспорт», «Їжа», «Культура», «Подорожі» тощо.

– Граматичні уроки: короткі пояснення граматичних правил у кожному модулі.

– Граматичні вправи: інтерактивні вправи для засвоєння граматичних конструкцій.

– Вправи на запам'ятовування слів: інтерактивні вправи для вивчення

нових слів та фраз.

– Персоналізований словник: користувачі можуть переглядати вивчені слова у своєму персональному словнику.

– Розмовні вправи: можливість практикувати розмовні навички з диктофоном та вибором правильного варіанту відповіді.

– Прослуховування фраз: аудіовправи з вимовою та перекладом поширених японських фраз.

– Бали та досягнення: кожен урок оцінюється за допомогою системи балів та нагород.

– Щоденні серії: підтримання щоденної серії уроків для отримання нагород.

Переваги:

– Комплексний підхід до навчання: уроки та вправи охоплюють граматику, словниковий запас, аудіювання та розмовну практику.

– Граматичні уроки з різним рівнем складності: граматичні уроки організовані за рівнем складності, що дозволяє користувачам розвивати свої знання поступово.

– Інтерактивні вправи на запам'ятовування словникового запасу: вправи на запам'ятовування слів та фраз використовують метод інтервального повторення (SRS), що сприяє ефективному запам'ятовуванню.

Недоліки:

– Обмежений доступ до матеріалів без підписки: безкоштовна версія має обмежений доступ до матеріалів, а користувачі потребують підписки для повного доступу.

– Відсутність вправ на вільне спілкування: практика розмовних навичок обмежується лише короткими вправами та не включає вільне спілкування.

– Недостатній розвиток навичок письма: практика письма обмежується лише короткими текстами та не включає більш комплексні вправи.

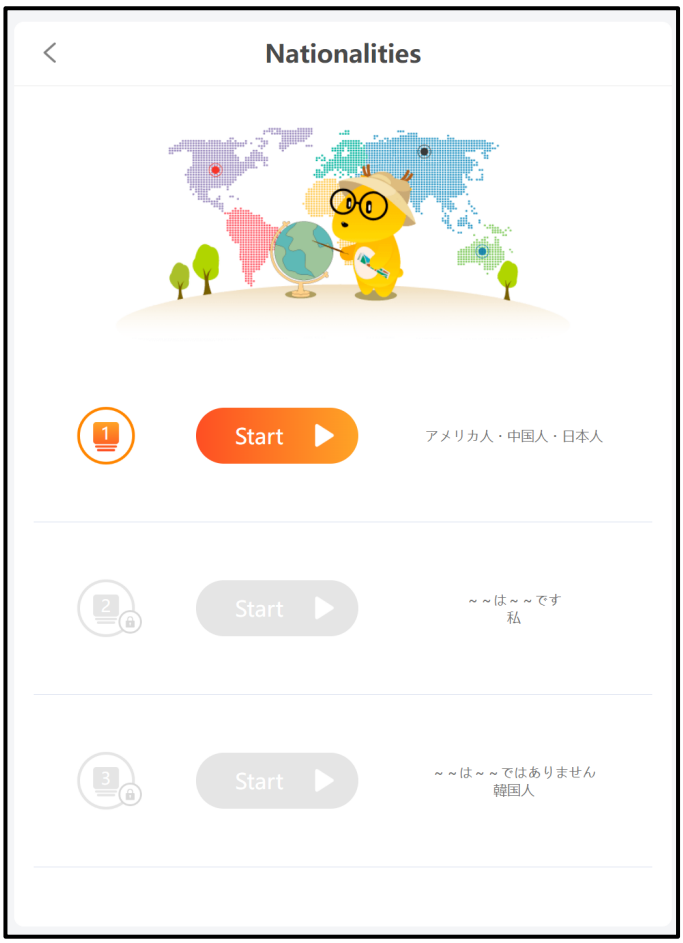


Рис. 1.4 Приклад сторінки уроків та вправ LingoDeer

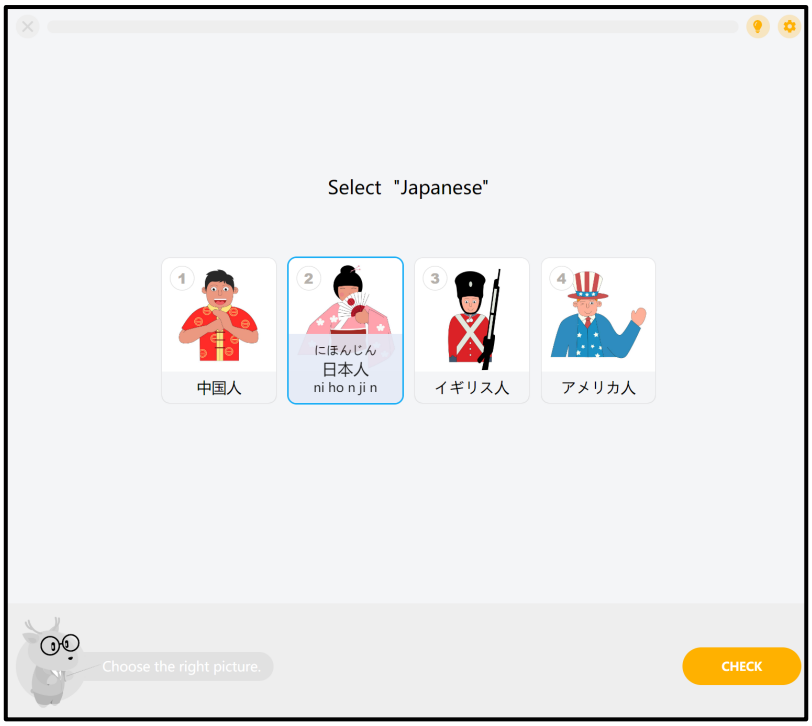


Рис. 1.5 Приклад сторінки граматичних вправ LingoDeer

В таблиці 1.1 зведено результати аналізу існуючих застосунків для вивчення японської мови та їх порівняння.

Таблиця 1.1

Зведена таблиця аналізу існуючих застосунків для вивчення японської мови та їх порівняння

Показник	**Duolingo**	**Tofugu**	**LingoDeer**
Уроки та вправи	Короткі уроки з темами «Привітання», «Їжа» тощо	Статті та посібники з різних аспектів японської мови та культури	Уроки організовані в тематичні модулі різних рівнів складності
Граматика	Базові уроки з граматики	Детальні пояснення граматичних правил і використання	Короткі уроки та вправи різних рівнів складності
Словниковий запас	Картки для запам'ятовування нових слів	Високоякісні статті та рекомендації для розширення лексики	Інтерактивні вправи з методом інтервального повторення (SRS)
Розмовна практика	Відсутня	Рекомендації для практики з носіями мови	Аудіовправи з вимовою та перекладом поширених фраз
Мотиваційна система	Бали, досягнення, щоденні серії	Відсутня	Бали, досягнення, щоденні серії, тести та ігри
Прогрес та статистика	Відстеження прогресу, статистика	Відстеження прогресу через особисті замітки та самооцінку	Відстеження прогресу, персоналізовані рекомендації
Доступність на платформах	Web, iOS, Android	Web	Web, iOS, Android

Продовження таблиці 1.1

Зведена таблиця аналізу існуючих застосунків для вивчення японської мови та їх порівняння

Показник	**Duolingo**	**Tofugu**	**LingoDeer**
Персоналізовані навчальні плани	Відсутні	Відсутні	Адаптуються під цілі та рівень знань користувача
Інтерфейс	Інтуїтивний та простий у використанні	Інтуїтивний, з багатим змістом та детальною інформацією	Інтуїтивний інтерфейс з багатомовною підтримкою
Обмеження без підписки	Повний доступ до базової версії	Повний безкоштовний доступ	Обмежений доступ до матеріалів
Цільова аудиторія	Початківці та особи, що готуються до JLPT N5-N4	Студенти мовних спеціальностей, ентузіасти, особи, що готуються до JLPT N5-N1	Студенти мовних спеціальностей, ентузіасти, особи, що готуються до JLPT
Особливості	Інтерактивний підхід, гейміфікація, безкоштовний доступ	Глибокий підхід до вивчення мови та культури, наголос на практичність	Комплексний підхід, граматичні уроки різного рівня, персоналізовані плани
Недоліки	Відсутність індивідуалізації планів, проблеми з вимовою	Може бути важким для повних початківців	Обмежений доступ до матеріалів без підписки, відсутність вправ на вільне спілкування

1.5 Визначення вимог до застосунку

Проаналізувавши сутність застосунку, його специфіку, цільову аудиторію та аналогів, було визначено наступні вимоги до застосунку для вивчення японської мови:

– Авторизація: реєстрація нових користувачів, вхід та вихід із системи.

– Бібліотека курсів та словникових колод: зберігання та управління курсами й колодами слів користувачів.

– Система пошуку та фільтрації курсів: пошук курсів за назвою, фільтрація результатів за рівнем JLPT та тривалістю курсу.

– Сторінка проходження курсу: навігація по курсу, читання та взаємодія з його вмістом.

– Чат-бот асистент на основі GPT-моделі OpenAI API: відповіді на питання користувача щодо уроків та тем курсу.

– Аналіз речень на основі GPT-моделі OpenAI API: розбиття речень на граматичні блоки та надання пояснень і посилань на відповідні матеріали.

– Інтервальна система повторення (SRS): можливість проходити сесії для запам'ятовування слів та фраз із використанням методу повторення інтервалів.

– Швидкий та інтуїтивний інтерфейс користувача: завантаження сторінок менш ніж за дві секунди. Інтерфейс повинен бути чітким, адаптивним та зручним у використанні.

2 ЗАСОБИ РЕАЛІЗАЦІЇ

2.1 Середовище розробки

Інтегроване середовище розробки (IDE) – це комплексне програмне забезпечення, що надає програмістам різноманітні інструменти для розробки програмного коду. Інтегровані середовища розробки зазвичай включають текстовий редактор, інструменти для автоматизації збірки коду, дебагер та, часто, інтерфейс для системи контролю версій. Це допомагає розробникам ефективно писати, тестувати і відлагоджувати свої програми в одному середовищі.

2.1.1 Visual Studio Code

Visual Studio Code (VS Code) — це передовий текстовий редактор від Microsoft, який широко використовується програмістами для розробки програмного забезпечення. Хоча VS Code сам по собі не є повноцінним інтегрованим середовищем розробки (IDE), його можна ефективно використовувати як IDE завдяки гнучкості та численним розширенням, які можуть значно розширити його функціональність.

VS Code підтримує багато мов програмування "з коробки", включаючи JavaScript, TypeScript, Python, PHP, C++, та інші. Це досягається через використання розширень, які можна легко встановити та налаштувати, адаптуючи редактор до специфічних вимог розробки.

Однією з ключових переваг VS Code є його система плагінів. Користувачі можуть встановлювати розширення для майже всього — від підтримки додаткових мов програмування і фреймворків до інтеграції з іншими інструментами та сервісами, такими як контейнери Docker, системи управління версіями (наприклад, Git) і бази даних.

Хоча VS Code не є IDE в класичному розумінні цього слова, його можливості та широка адаптованість роблять його відмінним вибором для

розробників, які шукають легкий, але потужний інструмент для кодування.

2.1.2 PyCharm

PyCharm — це потужне інтегроване середовище розробки (IDE), створене компанією JetBrains, спеціально призначене для мови програмування Python. PyCharm пропонує широкий набір інструментів для професійної розробки на Python і вважається одним із найкращих IDE для роботи з цією мовою.

PyCharm повністю підтримує розробку на Python, включаючи Python 3, а також велику кількість бібліотек Python, таких як Django, Flask, Google App Engine, Pyramid і інші. PyCharm автоматично визначає потреби проекту і налаштовує відповідні інтерпретатори та тести, забезпечуючи оптимальне середовище для розробки.

PyCharm інтегрується з HTML, CSS і JavaScript, забезпечуючи повноцінні можливості для веб-розробки. Це включає підтримку сучасних фреймворків, таких як AngularJS, Node.js, і інші, що робить PyCharm ідеальним вибором для веб-розробників, які використовують Python.

PyCharm надає потужні інструменти для профілювання програм Python та відлагодження коду. Це дозволяє розробникам ефективно ідентифікувати та вирішувати проблеми у своєму коді, оптимізувати продуктивність та забезпечувати надійність програм.

2.2 Мови програмування

Мова програмування — це формалізована мова, призначена для комунікації інструкцій до машини, зокрема комп'ютера. Вона використовується для створення програм, які виконують алгоритми. Більшість мов програмування складається з інструкцій для комп'ютерів. Вони дозволяють розробникам ефективно створювати програми, виражаючи розрахунки та управління даними.

2.2.1 Python

Python — це високорівнева, інтерпретована мова програмування, яка здобула широке визнання завдяки своїй читабельності, гнучкості та відносній простоті вивчення. Створена Гвідо ван Россумом у 1991 році, Python підтримує кілька парадигм програмування, включаючи об'єктно-орієнтоване, імперативне, процедурне та, у певній мірі, функціональне програмування [3].

Python відомий своїм чистим синтаксисом, що забезпечує високий рівень читабельності коду. Це робить мову легкою для вивчення новачками та сприяє кращій співпраці між розробниками на великих проектах.

Python застосовується у різних доменах, від веб-розробки до наукових досліджень, data-science, штучного інтелекту, автоматизації та багатьох інших. Мова здатна ефективно працювати як у простих скриптах, так і в складних підприємницьких системах [4].

Велика кількість бібліотек та фреймворків, доступних для Python, значно розширює можливості мови. Для веб-розробки популярні фреймворки, такі як Django і Flask, які надають потужні інструменти для створення веб-сайтів.

2.2.2 TypeScript

TypeScript є відкритою мовою програмування, яку розробила компанія Microsoft у 2012 році. Вона є надбудовою над JavaScript, що додає строгу типізацію та додаткові можливості мови, що спрямовані на підвищення продуктивності розробки та покращення масштабування великих програмних додатків.

TypeScript дозволяє явно вказувати типи змінних, параметрів та значень, що повертаються функціями. Це допомагає уникнути помилок, пов'язаних з неправильними типами даних, що часто зустрічаються у JavaScript.

TypeScript транскомпілюється в JavaScript, що означає, що будь-який код, написаний на TypeScript, може виконуватися в будь-якому браузері, на будь-якому сервері чи в будь-якому іншому середовищі, що підтримує JavaScript.

TypeScript є вибором для багатьох сучасних веб-фреймворків та бібліотек, таких як Angular, Vue.js (використання TypeScript заохочується у Vue 3) та React

(через використання TypeScript у React компонентах та хуках). Це дозволяє створювати більш надійні та легко підтримувані веб-додатки.

2.3 Веб-фреймворки

Веб-фреймворк - це програмний фреймворк, призначений для підтримки розробки веб-додатків, включаючи веб-сервіси, веб-ресурси та веб-API. Веб-фреймворки надають стандартний спосіб будувати та розгортати веб-додатки в Інтернеті

2.3.1 Django

Django є високорівневим Python веб-фреймворком, який сприяє швидкій розробці чистих та практичних веб-додатків. Створений у 2005 році, Django був розроблений з метою зробити процес розробки складних баз даних та орієнтованих веб-додатків якомога простішим та швидшим. Це відомо своєю філософією "batteries included" — велика кількість стандартних функцій для веб-розробки вже вбудовані в сам фреймворк [5].

Django використовує принцип DRY (Don’t Repeat Yourself), що дозволяє програмістам писати менше коду з вищою повторюваністю, що значно прискорює процес розробки.

Одна з ключових особливостей Django — потужна, налаштовувана адміністративна панель, яка автоматично генерується з моделей. Це дозволяє адміністраторам легко керувати даними додатка.

Django забезпечує високий рівень безпеки за замовчуванням. Він допомагає розробникам уникнути багатьох поширених помилок безпеки, таких як SQL ін'єкції, cross-site scripting, cross-site request forgery та clickjacking. Його користувацька система надає спосіб реалізації аутентифікації та авторизації.

2.3.2 Angular

Angular є потужним фронтенд-фреймворком, розробленим командою Google і спільнотою розробників. Це один із трьох найпопулярніших фреймворків

для розробки веб-додатків. Angular відомий своєю здатністю спрощувати розробку динамічних, односторінкових веб-додатків (SPA).

Angular використовує компонентно-орієнтовану архітектуру, де кожна частина інтерфейсу користувача (UI) будується як компонент. Компоненти в Angular містять HTML-шаблони і класи TypeScript, які керують цими шаблонами, та вони можуть легко взаємодіяти між собою [6].

Angular розроблений з використанням TypeScript, який надає строгу типізацію та об'єктно-орієнтовані можливості. Це допомагає створювати більш організований та легкий для підтримки код.

Angular використовує систему впровадження залежностей (DI), що дозволяє додаткам бути більш гнучкими та легшими в тестуванні. DI робить компоненти менш залежними один від одного, а також полегшує управління спільними ресурсами, такими як сервіси.

2.4 Система управління базою даних

PostgreSQL, часто називаний просто Postgres, є однією з найпотужніших відкритих систем управління базами даних (СУБД). Це об'єктно-реляційна СУБД, яка є відомою своєю надійністю, гнучкістю та підтримкою великої кількості даних.

PostgreSQL гарантує повну відповідність ACID (Atomicity, Consistency, Isolation, Durability), що робить її надійною системою для обробки транзакцій, критичних з точки зору вимог до надійності та безпеки даних.

PostgreSQL підтримує розширений набір функцій SQL, включаючи складні запити, зовнішні ключі, тригери, відображення, інтеграцію з користувацькими функціями та типами даних. Це дозволяє розробникам ефективно управляти даними та розширювати функціональність за необхідності.

PostgreSQL пропонує потужну підтримку JSON, що дозволяє зберігати і обробляти JSON-дані безпосередньо в базі. Це робить її ідеальною для веб-додатків, які використовують JSON для обміну даними.

2.5 Система контролю версій

GitHub — це веб-платформа для хостингу коду, яка базується на системі контролю версій Git. GitHub швидко став однією з найбільш впливових платформ у світі розробки програмного забезпечення, забезпечуючи інструменти для спільної роботи, контролю версій та коду розподіленого доступу.

Центральний елемент GitHub — репозиторії, де зберігається код. Користувачі можуть створювати публічні чи приватні репозиторії для своїх проектів, куди входить управління версіями за допомогою Git [7].

GitHub дозволяє користувачам "форкати" (створювати копії) чужих репозиторіїв, вносити зміни та пропонувати ці зміни авторам оригінальних репозиторіїв через пул-реквести. Це ключова можливість для співпраці над відкритими проектами.

2.6 Інструменти проєктування інтерфейсу користувача

Figma є веб-базованим інструментом для інтерфейсного дизайну, який дозволяє командам створювати, співпрацювати та ділитися дизайном інтерфейсів та прототипами. Заснований у 2012 році Діланом Філдом та Еваном Воллесом, Figma стала популярною серед дизайнерів завдяки своїй доступності, гнучкості та багатофункціональності.

Однією з ключових особливостей Figma є її повна веб-інтеграція, яка дозволяє користувачам працювати з будь-якого комп'ютера без необхідності завантаження додатків. Це забезпечує легкий доступ та співпрацю між учасниками проекту з різних географічних точок.

Figma включає інструменти для створення інтерактивних прототипів без необхідності стороннього програмного забезпечення. Це дозволяє дизайнерам швидко створювати прототипи з анімацією переходів та мікровзаємодіями, і тестувати їх прямо в середовищі Figma.

Figma підтримує створення та управління дизайн-системами, дозволяючи дизайнерам зберігати стилі, компоненти та інші ресурси в легкодоступних бібліотеках, що спрощує узгодженість і повторне використання в багатьох проектах.

3 ПРОЄКТУВАННЯ

3.1 Проєктування архітектури застосунку

Додаток реалізований за моделлю клієнт-сервер, що дозволяє розділяти обробку даних між сервером і клієнтом. Це розділення ролей сприяє оптимізації виконання завдань та покращує масштабованість та ефективність системи.

Клієнтська частина (фронтенд) забезпечує взаємодію з користувачем через графічний інтерфейс, розроблений за допомогою фреймворку Angular. Фронтенд відправляє запити до сервера через HTTP протокол, отримує відповіді та динамічно оновлює вміст сторінок без перезавантаження, використовуючи технології AJAX та JSON. Це створює плавний та зручний користувацький досвід, схожий на роботу з настільними додатками.

Серверна частина (бекенд), розроблена на основі фреймворку Django, виконує основну логіку обробки даних. Вона приймає запити від клієнта, обробляє їх згідно з бізнес-логікою додатку, звертається до бази даних PostgreSQL для зберігання або вилучення інформації, і надсилає необхідні дані назад клієнту. Бекенд також керує аутентифікацією та авторизацією користувачів, забезпеченням безпеки транзакцій і даних.

Сервер також інтегрований з різними сторонніми сервісами через API. Наприклад, використовується OpenAI API для реалізації функцій чат-бота [8] та аналізу речень, що дозволяє забезпечити додаткові освітні функції всередині платформи. Це включає автоматичний розбір речень на граматичні складові та надання користувачам роз'яснень або посилань для глибшого вивчення матеріалу.

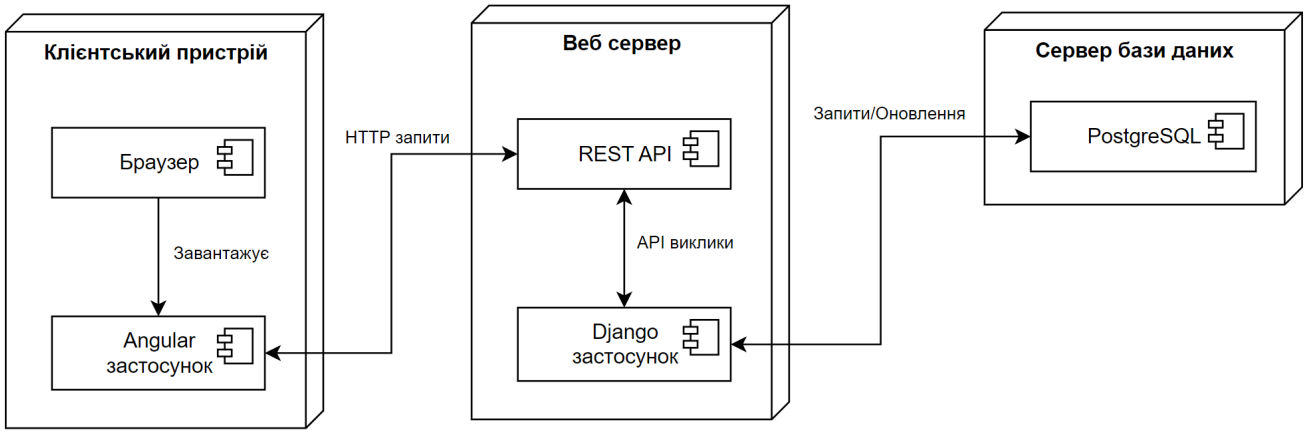


Рис. 3.1 Діаграма розгортання клієнт-серверної моделі

3.2 Архітектура клієнтської частини

Структура Angular проєкту організована у три основні частини: компоненти, сторінки та сервіси. Цей підхід забезпечує чітке розділення відповідальностей між різними частинами програми, що сприяє кращій організації коду та полегшує його розуміння, тестування та обслуговування [9].

Компоненти є будівельними блоками Angular-додатків. Вони включають логіку управління даними користувацького інтерфейсу та шаблон, який визначає, як виглядає інтерфейс. Кожен компонент зазвичай має власний файл HTML для шаблону, файл CSS для стилізації та TypeScript-файл для логіки. Організація компонентів у проєкті включає різноманітні елементи інтерфейсу, такі як списки, кнопки, панелі тощо, що можуть бути повторно використані в різних частинах додатку.

Сторінки складаються з груп компонентів та формують окремі відображення, які користувач бачить у веб-додатку. Наприклад, сторінка пошуку може включати компоненти для пошукової стрічки, фільтрів та результатів пошуку. Сторінки дозволяють агрегувати функціонал, необхідний для певного користувацького досвіду, і спрощують навігацію та взаємодію в рамках додатку.

Сервіси в Angular використовуються для виокремлення бізнес-логіки з компонентів, що дозволяє зробити код більш чистим та легким для тестування.

Сервіси можуть включати взаємодію з сервером, обробку даних та інші повторно використовувані функції. Вони вводяться у компоненти через механізм впровадження залежностей, що забезпечує низьку зв'язність.

Така організація проекту на Angular сприяє модульності та повторному використанню коду. Розділення логіки додатку на компоненти, сторінки та сервіси допомагає утримувати код організованим та чистим, а також полегшує розширення та обслуговування додатку. Використання компонентно-орієнтованого підходу дозволяє розробникам легко змінювати або оновлювати окремі частини інтерфейсу без впливу на інші, що є важливим для великих та складних додатків.

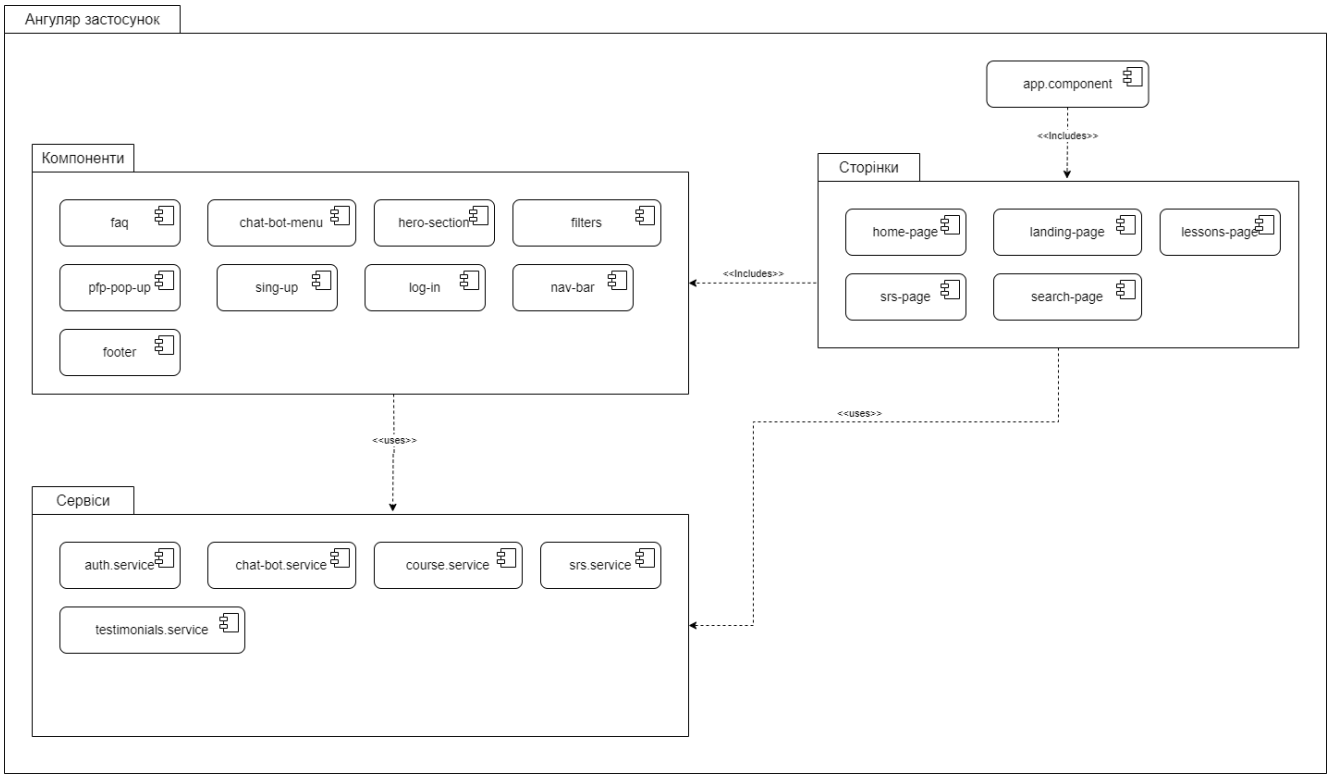


Рис. 3.2 Діаграма пакетів застосунку Angular

3.3 Архітектура серверної частини

Проєкти Django вирізняються шаруватою структурою. На верхньому рівні знаходиться шар самого проекту, який включає налаштування та конфігурації, що охоплюють весь проект. Нижче розміщений шар додатків проекту, де кожен додаток виконує певні функції та має чітко визначені завдання. Кожен Django

додаток має 5 основних файлів: моделі, відображення, адміністрування, серіалізатори, URL адреси. Розглянемо їх детальніше:

Models (Моделі): Визначають структуру бази даних. Моделі є класами Python, які Django використовує для створення схеми бази даних у PostgreSQL. Це дозволяє проекту ефективно управляти даними через об'єктно-орієнтований підхід.

Views (Відображення): Обробляють запити користувачів та відповідають на них, взаємодіючи з моделями для отримання та модифікації даних. Види логіки додатків є місцем, де визначається, які дані відправляються на клієнтську частину або отримуються від неї.

Admin (Адміністрування): Django надає готовий адміністративний інтерфейс, який дозволяє легко управляти даними додатків через веб-інтерфейс. Цей інтерфейс автоматично генерується з моделей і є великою перевагою для швидкого адміністрування.

Serializers (Серіалізатори): використовуються для перетворення даних моделей у формати, зручні для передачі через мережу (наприклад, JSON), та навпаки, для обробки даних, отриманих від клієнта.

URLs (URL адреси): визначають схему маршрутизації в додатку, спрямовуючи запити користувачів до відповідних обробників у вигляді відображень.

Перевагами даного виду архітектури проєкту є:

– Модульність: шарувата архітектура та чітке відокремлення залежностей між файлами та компонентами дозволяють легко масштабувати проект та розширювати його функціональність. Модулі можуть розроблятися та тестуватися незалежно один від одного.

– Легкість управління: завдяки інтеграції з PostgreSQL та структурованому підходу до адміністрування баз даних, проект може ефективно опрацьовувати великі обсяги даних, забезпечуючи високу продуктивність і надійність.

– Швидка розробка: Django пропонує багатий набір вбудованих

інструментів для розробки, що значно спрощує процес створення та запуску веб-додатків.

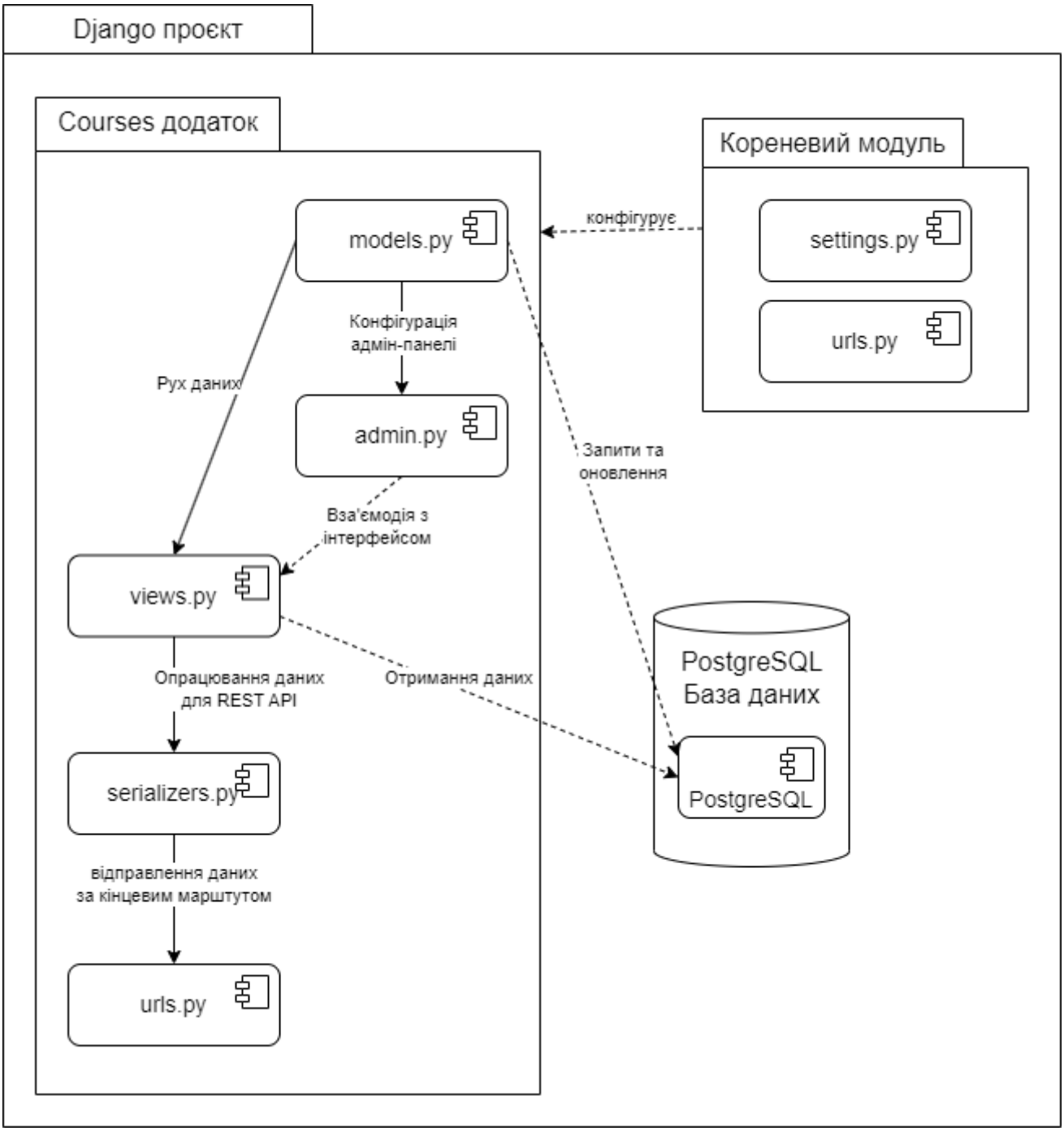


Рис. 3.3 Діаграма пакетів архітектури Django проєкту на прикладі одного Django додатку

3.4 Архітектура бази даних

Архітектура бази даних проєкту заснована на реляційній моделі, що використовує систему управління базами даних PostgreSQL. Основні принципи роботи та структури реляційних баз даних дозволяють ефективно організувати, зберігати та керувати великими обсягами даних.

База даних складається з таблиць, які містять дані про об'єкти або сутності проекту. Кожна таблиця має колонки та рядки:

– Колонки (поля) визначають типи даних, які можуть бути збережені (наприклад, рядки, числа, дати). У Django ці поля описуються у моделях як атрибути класів.

– Рядки: кожен рядок в таблиці представляє один або екземпляр сутності, наприклад, один користувач, один курс, тощо.

SQL (Structured Query Language) використовується для управління даними. Він дозволяє: створювати (CREATE) нові записи, читати (READ) існуючі дані, оновлювати (UPDATE) дані та видаляти (DELETE) дані. У Django ці операції зазвичай керуються через ORM (Object-Relational Mapping), що дозволяє взаємодіяти з базою даних через вищий рівень абстракції — об'єкти Python.

Реляційні бази даних дозволяють встановлювати відносини між таблицями:

– Один до одного (One-to-One): коли один рядок в одній таблиці відповідає одному рядку в іншій таблиці.

– Один до багатьох (One-to-Many): найбільш поширений тип відносин, де один рядок в одній таблиці може бути пов'язаний з багатьма рядками в іншій таблиці.

– Багато до багатьох (Many-to-Many): використовується коли багато рядків у одній таблиці можуть бути пов'язані з багатьма рядками в іншій таблиці.

Серед переваг реляційної моделі бази даних можна виділити наступні:

– Цілісність даних: можливість визначення обмежень і правил гарантує

точність і надійність даних.

– Гнучкість запитів: SQL дозволяє виконувати складні запити для аналізу і звітності.

– Безпека: контроль доступу та авторизація на рівні бази даних забезпечують захист інформації.

Ця архітектура бази даних є оптимальною для реалізації широкомасштабних веб-додатків, оскільки вона поєднує високу продуктивність з масштабованістю та гнучкістю в управлінні даними.

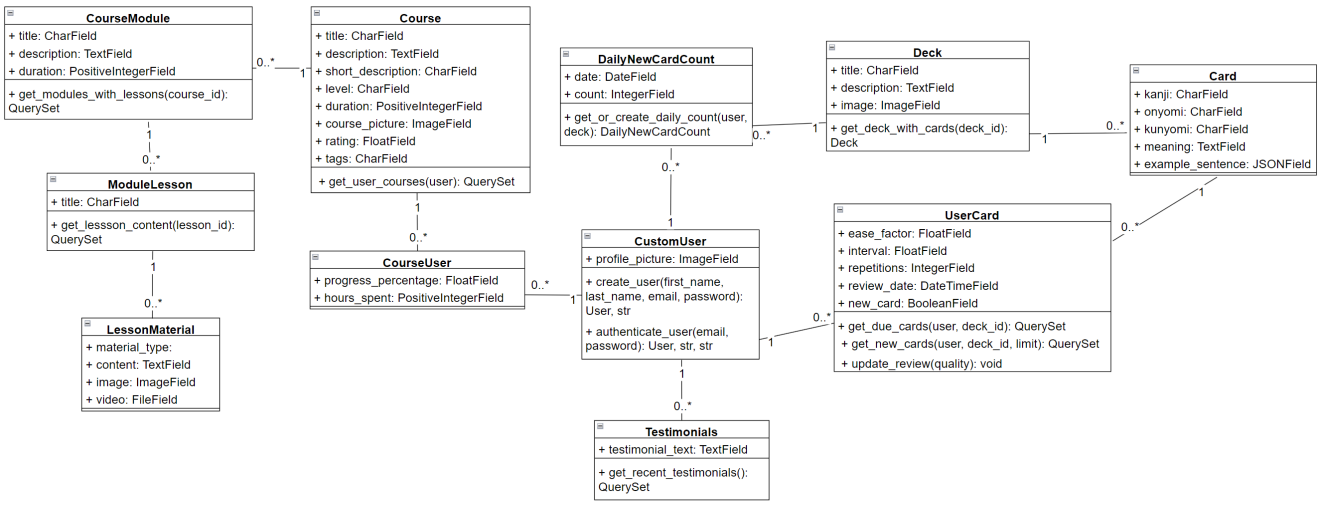


Рис. 3.4 Діаграма класів бази даних

Для зображення взаємодії користувача з системою створимо діаграму варіантів використання. На ній зліва зображується актор (в даному випадку – користувач) і система, яка відповідає на певні запроси користувача.

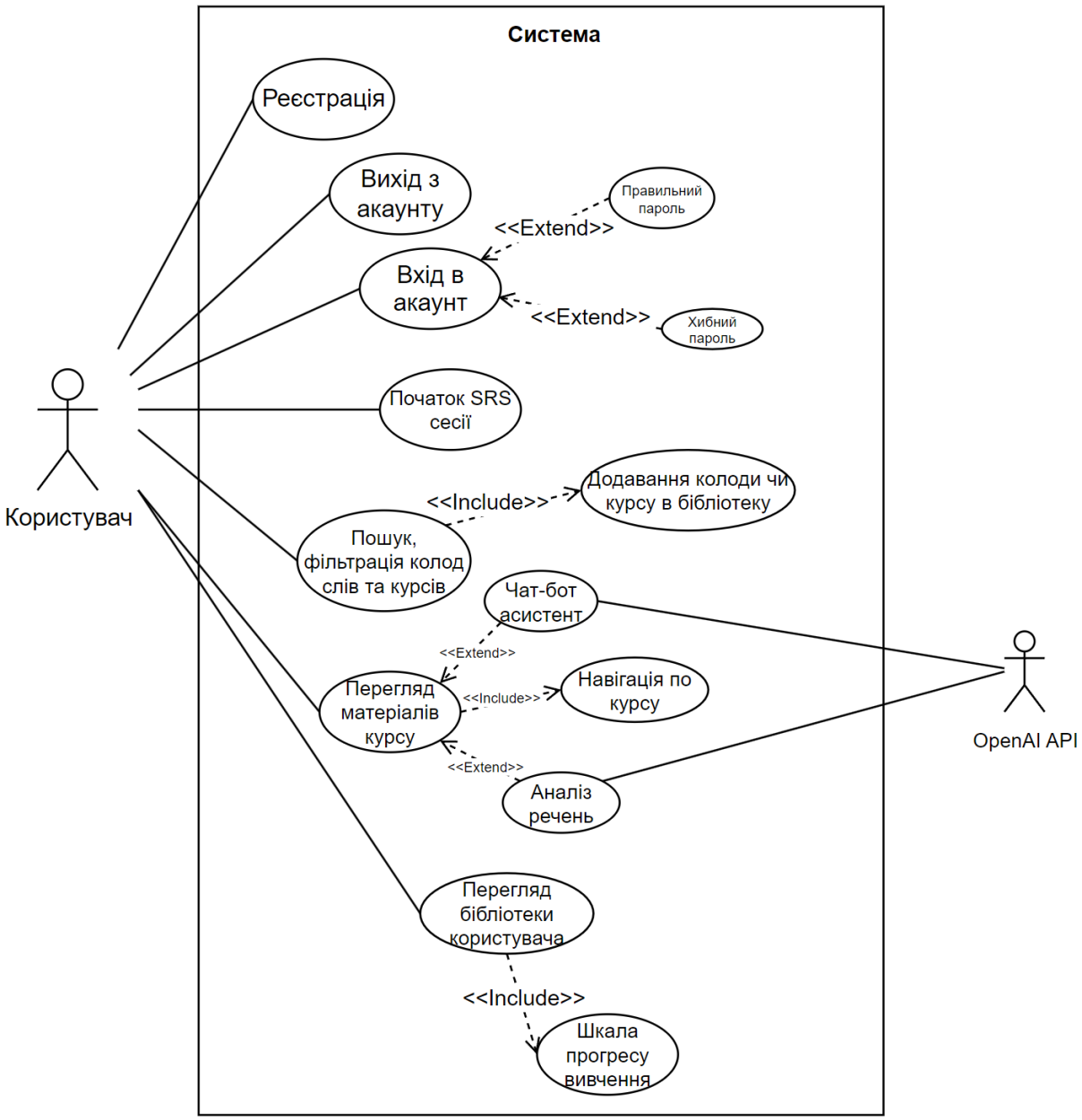


Рис. 3.4 Діаграма варіантів використання

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

4.1 Дизайн інтерфесу

Figma — це потужний інструмент для дизайну інтерфейсів користувача та прототипування, який дозволяє створювати інтерактивні макети і співпрацювати в режимі реального часу. Нижче описано основні кроки, використані для прототипування UI за допомогою Figma.

1. Створення нового проекту:

– Зареєструватися та увійти у Figma: відвідати Figma, створити обліковий запис або увійти, якщо обліковий запис вже є.

– Створити новий файл: на головній панелі інструментів вибрати кнопку "New File" для створення нового проекту.

2. Додавання фреймів. Фрейми у Figma слугують контейнерами для елементів дизайну і можуть представляти окремі екрани або компоненти інтерфейсу.

– Додати фрейм: на лівій панелі інструментів вибрати інструмент "Frame" (або натиснути клавішу F), потім намалювати фрейм на робочій області.

– Вибрати розмір фрейму: у правій панелі можна вибрати розмір фрейму з попередньо встановлених опцій (наприклад, для мобільних пристроїв, планшетів або десктопів) або задати власні розміри.

3. Додавання елементів UI:

– Використовувати інструменти "Rectangle", "Line", "Ellipse" та інші для створення основних форм та елементів інтерфейсу.

– Додати текст: використати інструмент "Text" (або натиснути клавішу T) для додавання текстових блоків.

– Вставити іконки та зображення: завантажити іконки та зображення з локального комп'ютера або використати вбудовані ресурси

Figma.

4. Створення компонентів. Компоненти у Figma дозволяють створювати повторювані елементи інтерфейсу, які можна легко змінювати та оновлювати.

– Створити компонент: вибрати групу елементів, які потрібно перетворити на компонент, натиснути правою кнопкою миші та вибрати "Create Component" (або натиснути клавіші Ctrl+Alt+K).

– Використовувати компоненти: перетягувати створені компоненти з бібліотеки компонентів на робочу область для повторного використання.

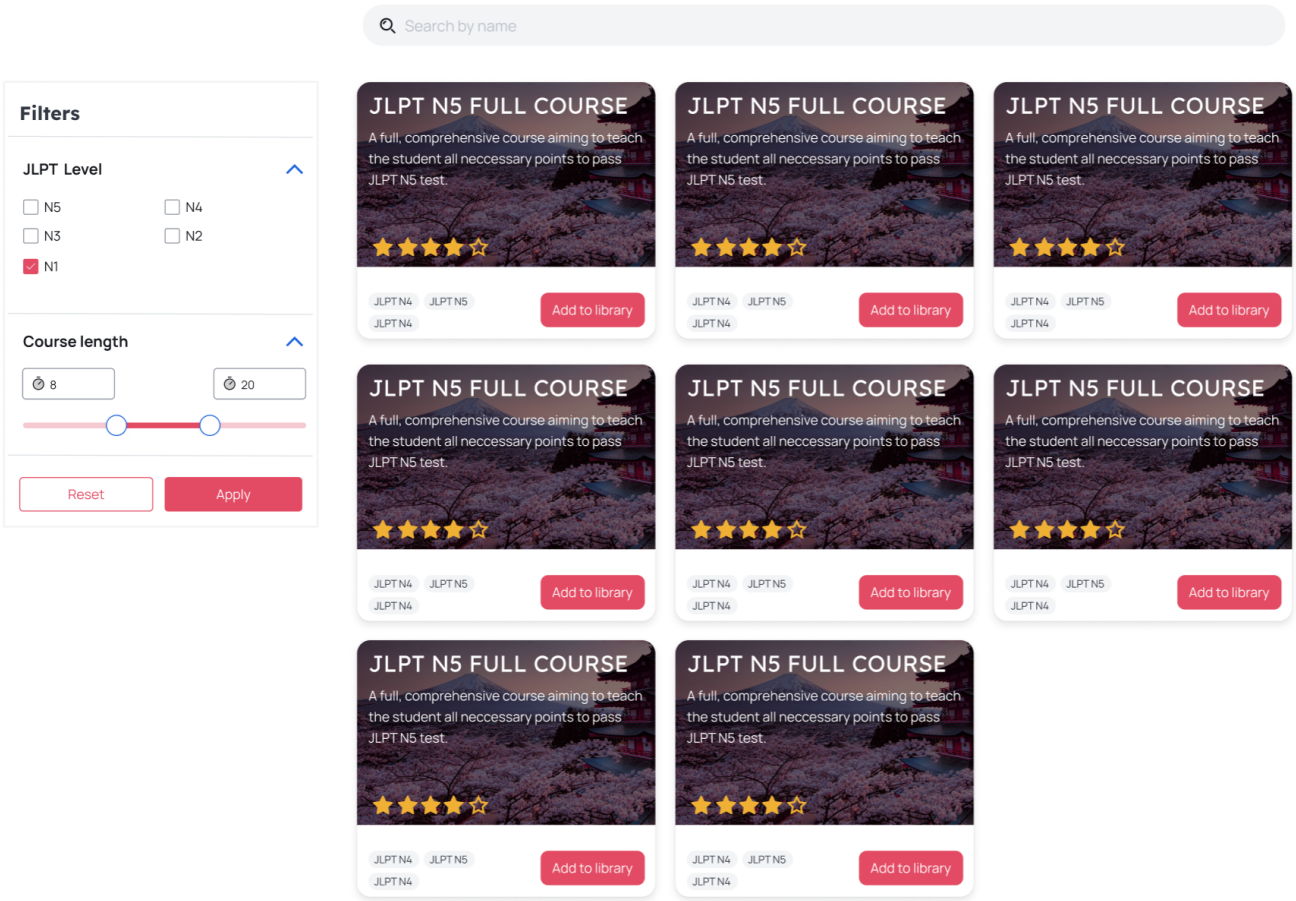


Рис. 4.1 Екранна форма створеного прототипу сторінки пошуку

Рис. 4.2 Екранна форма створеного прототипу сторінки запам’ятовування слів

4.2 Реалізація клієнтської частини

1. Для ініціалізації проекту Angular, використовуючи Angular CLI, потрібно створити новий проект. Виконати це можна з допомогою наступної команди у терміналі:

– ng new study-japanese

Після ініціалізації проєкту Angular за допомогою команди ng new, створюється стандартна структура каталогів і файлів, яка є основою для розробки Angular додатків. У списку наведено опис основних каталогів і файлів:

– Каталог /src/ є центральним місцем розробки, де знаходиться весь вихідний код додатка. Він містить компоненти, модулі, сервіси, стилі та інші аспекти додатка, які використовуються для створення та управління функціональністю.

– У каталозі /assets/ розміщуються статичні ресурси, такі як зображення, шрифти та інші медіа файли. Цей каталог важливий для організації та зберігання ресурсів, які використовуються в додатку, але не є частиною вихідного коду.

– Каталог /node_modules/ зберігає бібліотеки та пакети, які були завантажені через npm. Цей каталог є важливим для забезпечення функціонування

залежностей додатка, і зазвичай до нього не потрібно вносити зміни вручну.

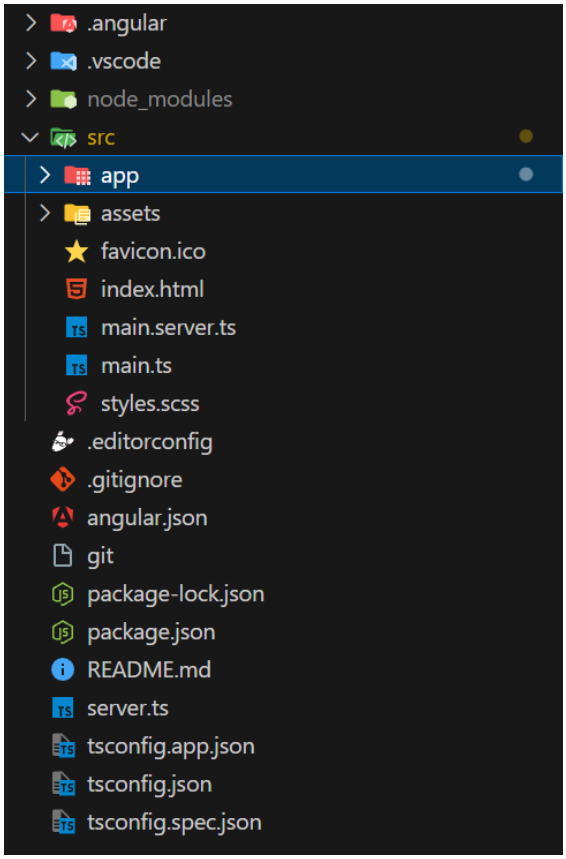


Рис. 4.1 Приклад структури Angular проєкту після ініціалізації

2. Другим кроком у реалізації клієнтської складової проєкту є створення компонентів. Для створення компонентів у проекті Angular використовується Angular CLI, який спрощує процес додавання нових компонентів. Компоненти є основними будівельними блоками будь-якого Angular додатка та включають шаблони HTML, стилі CSS та класи TypeScript для логіки [10].

Команда для створення нового компонента виглядає наступним чином:

– ng generate component sign-up

Ця команда створює новий каталог з назвою компонента в каталозі /src/app/, де розміщуються чотири основні файли:

– Файл TypeScript (component-name.component.ts), який містить клас компонента, декоратори та логіку компонента.

– Файл шаблону HTML (component-name.component.html), що визначає структуру візуальної частини компонента.

– Файл стилів CSS (component-name.component.css), який містить стилі для компонента.

– Файл специфікацій для тестування (component-name.component.spec.ts), що використовується для написання тестів для компонента.

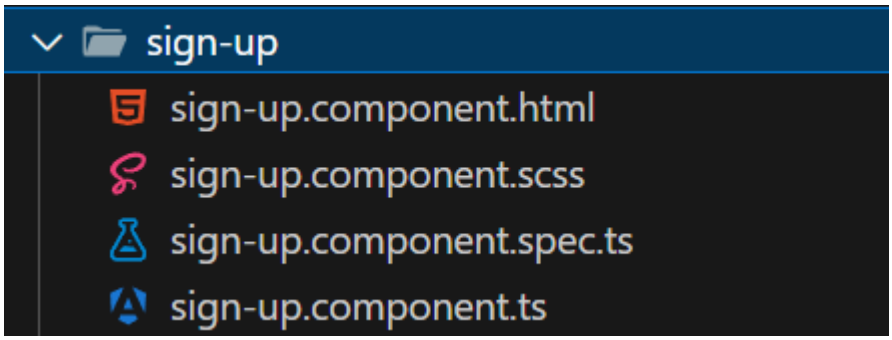


Рис. 4.2 Приклад створеного компоненту

3. Третім кроком створення клієнтської частини є запровадження сервісів. Для створення сервісів у проекті Angular використовується Angular CLI, що полегшує роботу з бізнес-логікою та зв'язками даних між компонентами додатка. Сервіси в Angular відіграють ключову роль у спільному використанні методів, даних та функціональності.

Команда для генерації нового сервісу виглядає наступним чином:

– ng generate service auth

Ця команда створює два основні файли в каталозі /src/app/:

– Файл TypeScript (service-name.service.ts), який містить клас сервісу з необхідними методами та властивостями.

– Файл специфікацій для тестування (service-name.service.spec.ts), що використовується для написання тестів для сервісу.

Сервіси в Angular часто використовуються для управління даними, взаємодії з сервером через HTTP запити, зберігання стану додатку, та інших завдань, які вимагають спільного використання функціональності між різними компонентами. Для інтеграції сервісу в компоненти Angular, використовується механізм впровадження залежностей (dependency injection): сервіс імпортується в

компонент, де він має бути використаний; сервіс впроваджується у конструктор компонента, що дозволяє Angular управляти життєвим циклом сервісу і надавати його екземпляр компоненту.

4. Після реалізації компонентів, сервісів та їх наповнення, четвертим кроком є налагодження маршрутів застосунку. Управління маршрутами в Angular виконується за допомогою вбудованого модуля маршрутизації, який дозволяє організувати навігацію між різними компонентами в односторінковому додатку (SPA). В Angular версії 17 управління маршрутами здійснюється з використанням RouterModule, який дозволяє визначити шляхи та асоціювати їх з конкретними компонентами.

В модулі маршрутизації (app.routes.ts), що створюється автоматично при ініціалізації Angular проєкту, можна визначити маршрути та їх відповідності з компонентами. Приклад конфігурації маршрутів:

```
import { Routes } from '@angular/router';
export const routes: Routes = [
    {'path': 'home', component: SrsPageComponent},
    {'path': 'search-courses', component: SearchPageComponent},
];
```

Для використання маршрутизації у компонентах Angular використовуються спеціальні директиви, такі як <router-outlet> та <router-link>. <router-outlet> визначає місце на сторінці, де будуть відображатися компоненти в залежності від активного маршруту. <router-link> дозволяє створювати посилання для навігації між маршрутами.

5. Після реалізації усіх компонентів, сервісів та налаштування маршрутів, останнім кроком є запуск проєкту. Запуск Angular додатку дозволяє перевірити функціональність та інтерфейс перед релізом додатку. Angular CLI надає простий спосіб запустити додаток, що допомагає розробникам швидко тестувати зміни у реальному часі.

Для запуску Angular додатку використовується команда:

- ng serve

Ця команда компілює додаток в режимі розробки і запускає веб-сервер, доступний за адресою http://localhost:4200. Веб-сервер слідкує за змінами у файлах джерельного коду та автоматично перекомпілює додаток та перезавантажує вікно браузера, коли зміни вносяться в код.

Нище наведено екранні форми декількох сторінок клієнтської частини вебсайту.

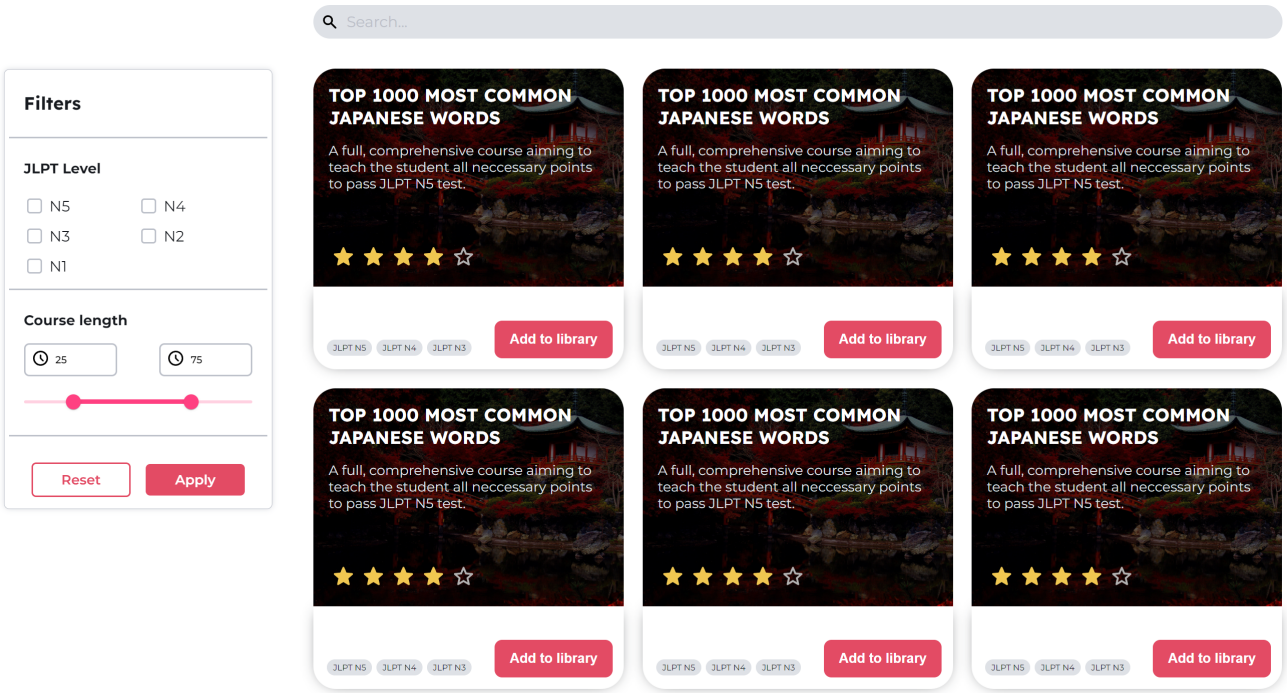


Рис. 4.3 Екранна форма реалізації сторінки пошуку

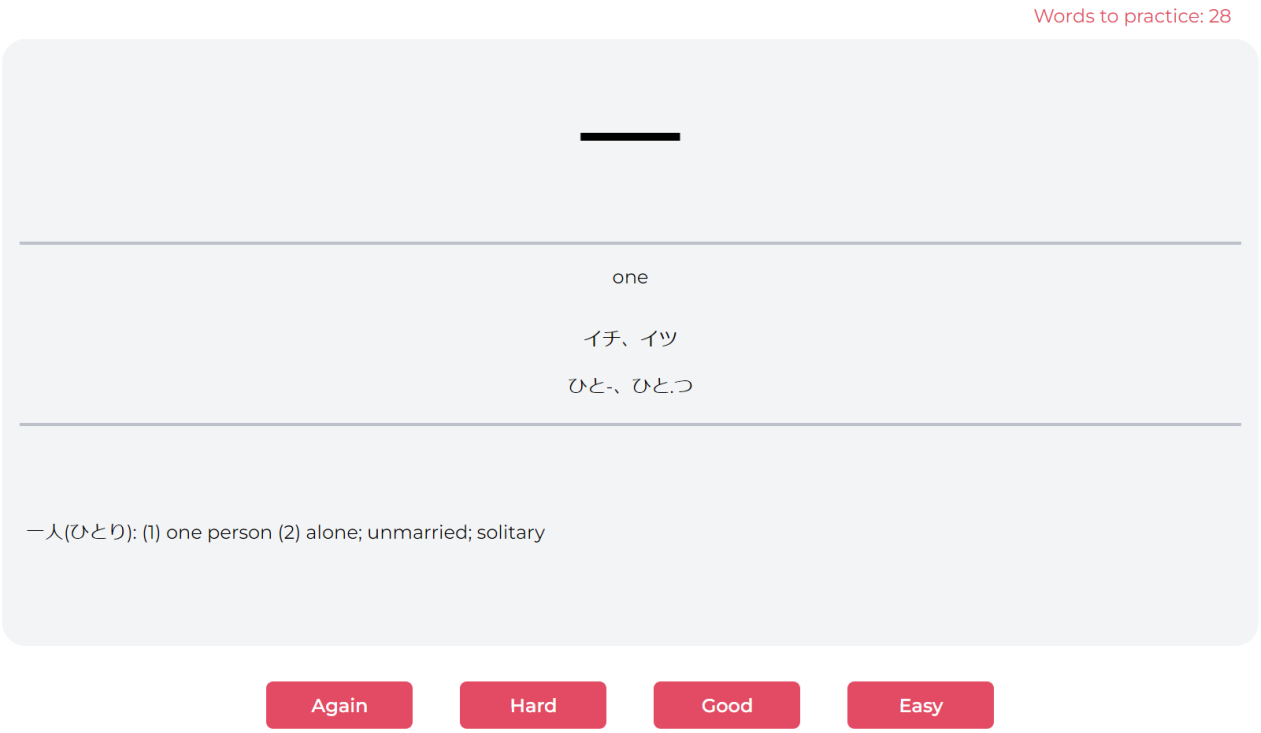


Рис. 4.4 Екранна форма реалізації сторінки запам’ятовування слів

4.3 Реалізація серверної частини

1. Першим кроком створення серверної частини є ініціалізація Django проєкту. Це робиться за допомогою наступної команди:

– django-admin startproject study-japanese

Ця команда створює новий каталог study-japanese, який містить основні файли і структуру для Django проекту:

– manage.py: Скрипт для керування проектом (запуск сервера, міграції бази даних, створення додатків тощо).

– my_django_project/: Основний каталог проекту, який містить:

– init.py: Позначає каталог як пакет Python.

– settings.py: Файл конфігурації проекту.

– urls.py: Файл маршрутизації для кореневого проекту.

– wsgi.py: Файл для розгортання проекту за допомогою WSGI-серверів.

– asgi.py: Файл для розгортання проекту за допомогою ASGI-серверів

(для підтримки WebSockets і інших протоколів).

2. Другим кроком є створення Django додатків. У Django проєкті додатки використовуються для організації коду за функціональними модулями, що полегшує управління та масштабування проекту. Щоб створити новий додаток, треба написати наступну команду з кореневого каталогу проекту (де знаходиться файл manage.py) [11]:

– python manage.py startapp spaced_rep_system

Після виконання цієї команди створюється новий каталог my_app, що містить наступні файли та каталоги:

– admin.py: Файл для налаштування адміністративного інтерфейсу Django.

– apps.py: Конфігураційний файл додатка.

– models.py: Файл для визначення моделей бази даних.

– tests.py: Файл для написання тестів для додатка.

– views.py: Файл для визначення логіки обробки запитів та формування відповідей.

– migrations/: Каталог для зберігання міграцій бази даних, які відстежують зміни в моделях.

Додатково було створено файли serializers.py та urls.py. Щоб додаток почав працювати у проекті, необхідно додати його до списку встановлених додатків у файлі конфігурації settings.py:

```
INSTALLED_APPS = [
    'my_app',
]
```

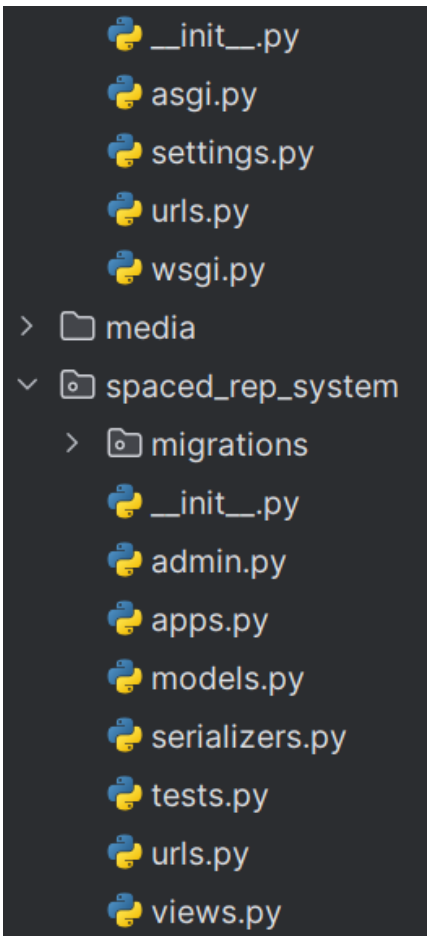


Рис. 4.5 Приклад створеної структури Django проєкту та додатку

3. Третій крок - це реалізація файлу models.py. Моделі в Django визначають структуру бази даних, включаючи поля та відносини між ними. Реалізуємо модель Deck (колода слів) для додатку spaced_rep_system:

```
class Deck(models.Model):
    title = models.CharField(max_length=255)
    description = models.TextField(blank=True, null=True)
    image = models.ImageField(upload_to='deck_images/', blank=True, null=True)
```

В даному випадку ми визначили, що модель Deck матиме поле title строкового типу з обмеженням в 255 символів, поле description, але вже без обмеження в довжині, і поле image, що зберігає зображення колоди слів.

Після визначення моделей необхідно створити та застосувати міграції, щоб синхронізувати структуру бази даних з визначеннями моделей:

– python manage.py makemigrations my_app
– python manage.py migrate

Data Output | Messages | Notifications

	id [PK] bigint	title character varying (255)	description text	image character varying (100)
1	1	ALL 2,136 JOUYOU KANJI	The deck contains all the 2,136 Jouyou kanji characters with their meaning, reading and example sentenc...	deck_images/Himeji.jpg
2	2	[JLPT N5] ALL KANJI	All JLPTN5 Kanji here	deck_images/Fuji.jpg

Рис. 4.6 Створена таблиця Deck в базі даних PostgreSQL

4. Четвертим кроком є налагодження адміністративної панелі. Адміністративний інтерфейс Django дозволяє легко керувати моделями та даними через веб-інтерфейс. Для налаштування адміністративного інтерфейсу, необхідно зареєструвати моделі у файлі admin.py:

```
class DeckAdmin(admin.ModelAdmin):
    list_display = ('title', 'description')
    search_fields = ('title', 'description')
admin.site.register(Deck, DeckAdmin)
```

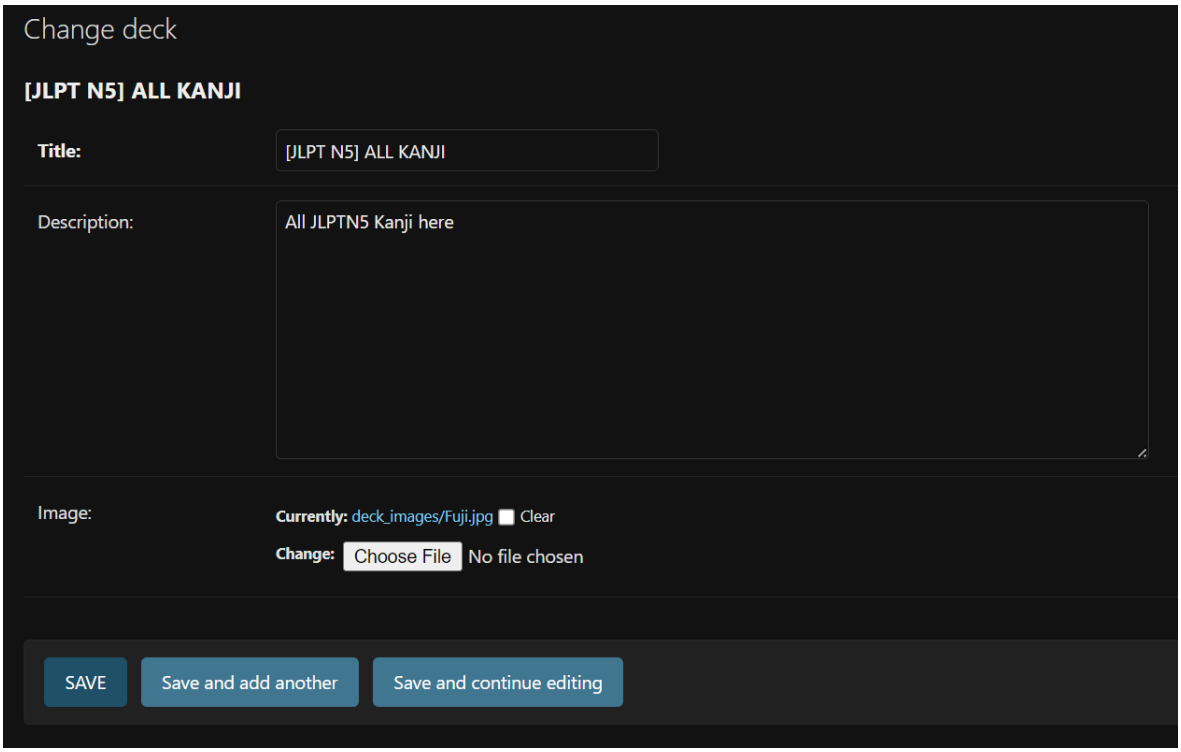


Рис. 4.7 Зареєстровану модель зручно редагувати через адмін-інтерфейс

5. П’ятий крок – це реалізація відображень. Відображення (views) у Django відповідають за обробку запитів користувачів та формування відповідей. Відображення можуть повертати HTML-сторінки, JSON-відповіді або перенаправляти користувачів на інші сторінки. Реалізуємо відображення для додатку spaced_red_system:

```
class DeckCardsAPIView(APIView):
    def get(self, request, deck_id):
        try:
            deck = Deck.objects.get(id=deck_id)
            cards = Card.objects.filter(deck=deck)
            serializer = CardSerializer(cards, many=True)
            return Response(serializer.data)
        except Deck.DoesNotExist:
            return Response({"error": "Deck not found"}, status=404)
```

В даному випадку реалізовано відображення, що повертає колоду слів за певним id з бази даних до клієнта, який буде відповідати за відображення переданих даних.

6. Шостим кроком є реалізація серіалізаторів. Як було видно з минулого кроку, відображення використовувало серіалізатор CardSerializer перед поверненням відповіді. Серіалізатори у Django REST Framework використовуються для перетворення комплексних типів даних, таких як запити до бази даних і екземпляри моделей, у формат, що може бути легко перетворений у JSON, XML або інші типи контенту. Серіалізатори також дозволяють перевіряти дані та перетворювати їх назад у складні типи.

```
class CardSerializer(serializers.ModelSerializer):
    class Meta:
        model = Card
        fields = ['id', 'kanji', 'onyomi', 'kunyomi', 'meaning', 'example_sentences']
```

7. Сьомий крок – це налаштування маршрутизації. Маршрутизація у

Django забезпечується за допомогою URL-конфігурацій, які спрямовують HTTP-запити до відповідних відображень. Кожний додаток може мати власний файл urls.py, де визначаються маршрути для цього додатку. Створимо маршрут для відображення DeckCardsAPIView:

```
from django.urls import path
urlpatterns = [
path('decks/<int:deck_id>/cards/', DeckCardsAPIView.as_view(), name='deck-cards'),]
```

8. Останнім кроком є запуск проекту: після налаштування всіх компонентів проекту Django, необхідно запустити сервер розробки для перевірки функціональності та тестування додатка.

Щоб запустити сервер розробки, використовуйте наступну команду у терміналі з кореневого каталогу проекту (де знаходиться файл manage.py):

– python manage.py runserver

Ця команда запускає вбудований у Django сервер розробки, який доступний за адресою http://localhost:8000.

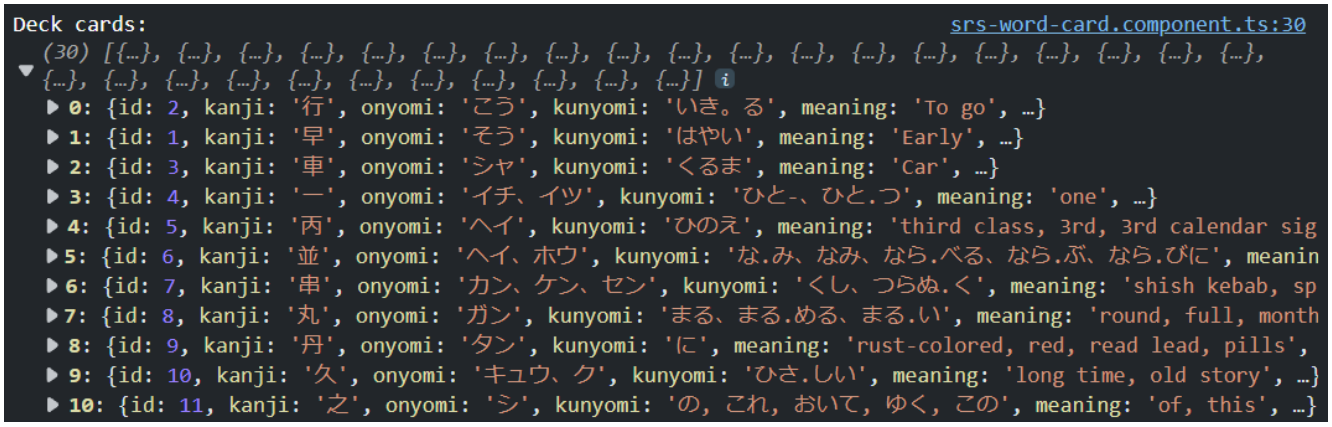


Рис. 4.8 Дані успішно передано до клієнтської частини

4.4 Завантаження проєкту на GitHub

Для того, щоб додати Django та Angular проект до існуючого репозиторію на GitHub, необхідно виконати кілька кроків. Вони включають ініціалізацію Git у

проекті, додавання файлів до репозиторію та завантаження змін до GitHub.

Для початку треба відкрити термінал у кореневому каталозі Django або Angular проєкту. Далі виконати команду для ініціалізації Git-репозиторію:

– git init

Далі додаємо URL існуючого репозиторію GitHub як віддалений репозиторій:

– https://github.com/AndriiSokolovskyi/Study-Japanese

Далі треба додати всі файли проєкту до індексу Git:

– git add .

Наступним кроком створюємо коміт з повідомленням, яке описує зміни, внесені у проєкт:

– git commit -m "Added SRS system"

Останнім кроком є завантаження змін до віддаленого репозиторію на GitHub:

– git push -u origin master

4.5 Тестування застосунку

Протестуємо серверну частину. Виконаємо перевірку правильності роботи моделей. Тестування моделей у Django є важливим етапом для забезпечення правильності та стабільності роботи додатку. Було використано модульне тестування, яке дозволяє перевіряти окремі компоненти системи ізольовано від інших.

Переваги модульного тестування:

– Ізольованість: Модульні тести перевіряють лише одну функцію чи метод, що полегшує виявлення джерела помилок.

– Автоматизація: Тести можна запускати автоматично, що прискорює процес перевірки змін у коді.

– Покращення якості коду: Регулярне написання тестів сприяє створенню більш надійного та зрозумілого коду.

– Запобігання регресії: Тести допомагають переконатися, що нові зміни не спричиняють неочікуваних помилок у вже працюючому коді.

Опис тестів, які було виконано. Наступні тести перевіряють базові операції створення об'єктів моделей Deck, Card, UserCard та DailyNewCardCount.

```
User = get_user_model()
class SRSModelTests(TestCase):
    def setUp(self):
        self.user = User.objects.create_user(username='testuser', password='testpassword')
        self.deck = Deck.objects.create(title='Test Deck')
        self.card = Card.objects.create(deck=self.deck, kanji='行', meaning='To go')
        self.user_card = UserCard.objects.create(user=self.user, card=self.card)
        self.daily_new_card_count = DailyNewCardCount.objects.create(user=self.user,
date=date.today(), count=20)
    def test_create_deck(self):
        self.assertEqual(self.deck.title, 'Test Deck')
    def test_create_card(self):
        self.assertEqual(self.card.kanji, '行')
    def test_create_user_card(self):
        self.assertEqual(self.user_card.user, self.user)
        self.assertEqual(self.user_card.card, self.card)
    def test_create_daily_new_card_count(self):
        self.assertEqual(self.daily_new_card_count.user, self.user)
        self.assertEqual(self.daily_new_card_count.date, date.today())
        self.assertEqual(self.daily_new_card_count.count, 20)
```

Пояснення тестів:

– setUp: Цей метод запускається перед кожним тестом і створює необхідні об'єкти для тестування.

– test_create_deck: Перевіряє, чи правильно створено об'єкт колоди.

– test_create_card: Перевіряє, чи правильно створено об'єкт карти.

– test_create_user_card: Перевіряє, чи правильно створено об'єкт, який пов'язує користувача з картою.

– test_create_daily_new_card_count: Перевіряє, чи правильно створено об'єкт підрахунку нових карт на день.

Для запуску тестів використовується наступна команда:

– python manage.py test spaced_rep_system

```
Found 4 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
....
----------------------------------------------------------------------
Ran 4 tests in 0.834s

OK
Destroying test database for alias 'default'...
```

Рис. 4.9 Успішне проходження усіх тестів

Як видно з зображення, тести завершилися зі статусом «ОК», що означає, що система успішно пройшла усі тести моделей Deck, Card, UserCard та DailyNewCardCount. Отже, система працює справно.

ВИСНОВКИ

Результатами виконаної роботи стало зменшення часу на вивчення японської мови шляхом розробки застосунку з використанням мови Python та фреймворків Django і Angular.

Було виконано наступні задачі:

– Проаналізовано предметну область: визначено сутність сайту для вивчення японської мови та специфіку процесу навчання; охарактеризована цільова аудиторія та проаналізовано існуючі засоби з вирішення поставленої задачі, виведено їх переваги та недоліки; виходячи з проведеного аналізу предметної області було визначено вимоги до програмного забезпечення, а саме: реалізація авторизації, бібліотеки курсів та словникових колод, системи пошуку та фільтрації, сторінки проходження курсу, чат-бот асистенту, аналізу речень, системи повторення слів та швидкого інтерфейсу.

– Обрано засоби реалізації: VS Code та PyCharm як IDE, TypeScript та Python як мови програмування, Angular та Django як фреймворки для розробки веб-застосунків, PostgreSQL як система управління базою даних, GitHub як система контролю версій та Figma як інструмент прототипування інтерфейсу користувача.

– Зроблено прототип робочої системи з використанням діаграми класів, пакетів та розгортання.

– Реалізовано систему, використовуючи обрані технології: зроблено прототип UI, налаштовано та створено проєкт Angular, налаштовано та створено проєкт Django, завантажено проєкт до GitHub’у та протестовано моделі на коректність роботи.

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

1. Соколовський А.В. Розробка застосунку для вивчення японської мови з використанням мови Python та фреймворків Django, Angular/Жебка В.В.,

Соколовський А.В.// Матеріали всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в ІКТ». Збірник тез. 24.04.2024, м. Київ. К.:ДУІКТ, С. 105-107.

2. Соколовський А.В. Використання OpenAI API для реалізації боту-помічника з вивчення японської мови/Жебка В.В., Соколовський А.В.// Матеріали всеукраїнської науково-практичної конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. 15.05.2024, м. Київ. К.:ДУІКТ, С. 36-37.

ПЕРЕЛІК ПОСИЛАНЬ

1. Sarrion E. Learning a language. ChatGPT for beginners. Berkeley, CA, 2023. P. 183–210. URL: https://doi.org/10.1007/978-1-4842-9804-6_11 (date of access: 14.05.2024).

2. Tingiris S., Kinsella B. Exploring GPT-3: an unofficial first look at the general-purpose language processing API from openai. Packt Publishing, Limited, 2021. 296 p.

3. 3.12.3 documentation. Python documentation. URL: https://docs.python.org/3/ (date of access: 28.03.2024).

4. Python for web development / U. Patkar et al. International journal of computer science and mobile computing. 2022. Vol. 11, no. 4. P. 36–48. URL: https://doi.org/10.47760/ijcsmc.2022.v11i04.006 (date of access: 29.03.2024).

5. Django web development framework: powering the modern web / S. Chen et al. American journal of trade and policy. 2020. Vol. 7, no. 3. P. 99–106. URL: https://doi.org/10.18034/ajtp.v7i3.675 (date of access: 29.03.2024).

6. Interpretation and analysis of angular framework / G. Geetha et al. 2022 international conference on power, energy, control and transmission systems (ICPECTS), Chennai, India, 8–9 December 2022. 2022. URL: https://doi.org/10.1109/icpects56089.2022.10047474 (date of access: 29.03.2024).

7. Uzayr S. B. Introduction to Git and GitHub. Mastering GitHub Pages. Boca Raton, 2022. P. 1–52. URL: https://doi.org/10.1201/9781003242055-1 (date of access: 14.05.2024).

8. Exploring chatgpt capabilities and limitations: a survey / A. Koubaa et al. IEEE access. 2023. P. 1. URL: https://doi.org/10.1109/access.2023.3326474 (date of access: 07.05.2024).

9. Garcia V. H. Introduction to angular framework. Getting started with angular. Berkeley, CA, 2023. P. 1–11. URL: https://doi.org/10.1007/978-1-4842-9206-8_1 (date of access: 14.05.2024).

10. Angular documentation. Angular. URL: https://angular.io/docs (date of access: 01.04.2024).

11. Django documentation | Django documentation. Django Project. URL: https://docs.djangoproject.com/en/5.0/ (date of access: 10.04.2024).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка застосунку для вивчення японської мови з використанням мови Python та фреймворків Django, Angular

Виконав студент 4 курсу
групи ПД-42
Соколовський Андрій Вадимович
Керівник роботи
Д.т.н, професор, завідувач кафедри ТЦР Жебка Вікторія Вікторівна

Київ – 2024

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – зменшення часу на вивчення японської мови шляхом розробки застосунку з використанням мови Python та фреймворків Django, Angular.
- **Об’єкт дослідження** - процес вивчення японської мови.
- **Предмет дослідження** - програмне забезпечення для вивчення японської мови.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Огляд специфіки вивчення японскьої мови, аналіз цільової аудиторії та існуючих засобів, визначення вимог до застосунку з вивчення японської мови.
2. Огляд та аналіз засобів реалізації застосунку з вивчення японської мови.
3. Проєктування архітектури застосунку з вивчення японської мови.
4. Програмна реалізація застосунку з вивчення японської мови.
5. Тестування застосунку.

3

АНАЛІЗ АНАЛОГІВ

Показник	Tofugu	Duolingo	LingoDeer	Study Japanese
Платформа	Веб-сайт	Веб-сайт, мобільний додаток	Веб-сайт, мобільний додаток	Веб-сайт
Бібліотека матеріалів користувача	-	+	+	+
Тип уроків	Статті, блоги, відео	Інтерактивні вправи	Інтерактивні вправи	Курси
Система SRS (spaced repetition system)	-	+	-	+
Чат-бот асистент	-	+	-	+
Інструмент аналізу речень	-	-	-	+
Система пошуку та фільтрації навчальних матеріалів	+	-	-	+
Оцінка прогресу проходження курсу	-	+	+	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні:

1. Реєстрація нових користувачів, вхід та вихід із системи.
2. Зберігання та перегляд курсів і колод слів.
3. Система пошуку та фільтрації курсів.
4. Сторінка проходження курсу, що включає навігацію по курсу та читання його вмісту.
5. Чат-бот асистент з вивчення японської мови
6. Інструмент аналізу речень
7. Інтервальна система повторення.

Нефункціональні:

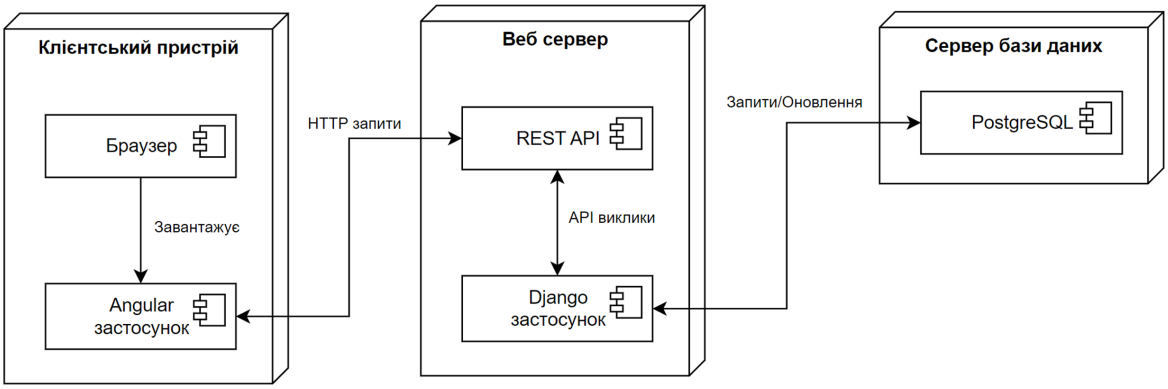
1. Завантаження сторінки менш ніж за 2 секунди
2. Адаптивне верстання
3. Шифрування паролю користувача
4. Сумісність з браузерами Chrome, Edge, Firefox та Safari

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

6

ДІАГРАМА РОЗГОРТАННЯ

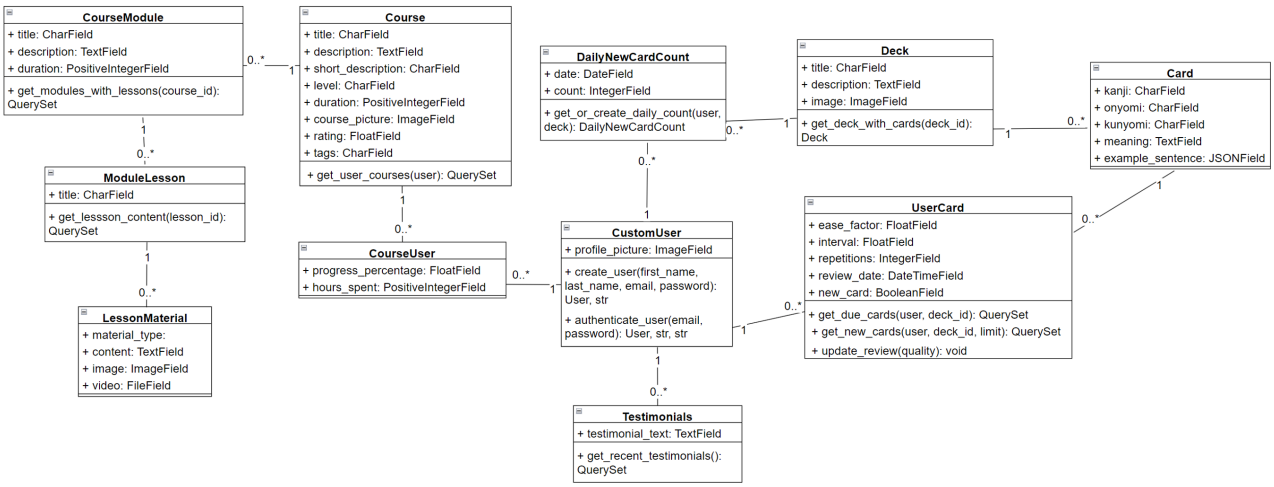


7

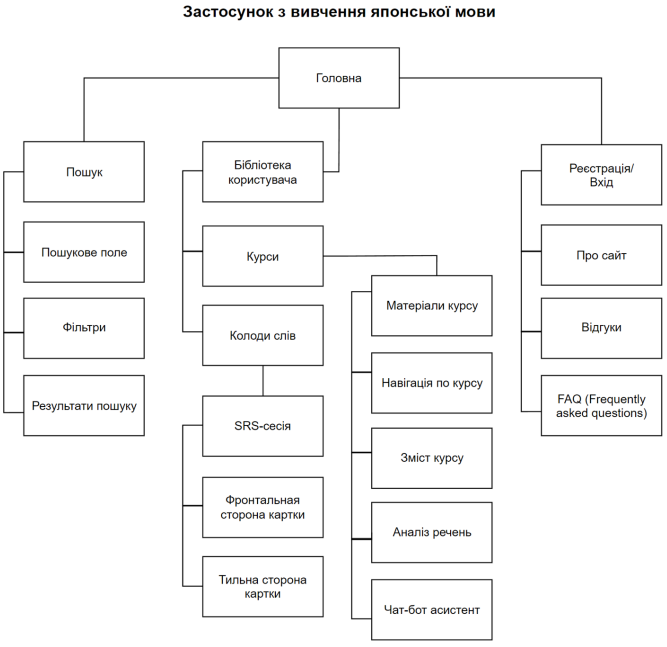
ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

8

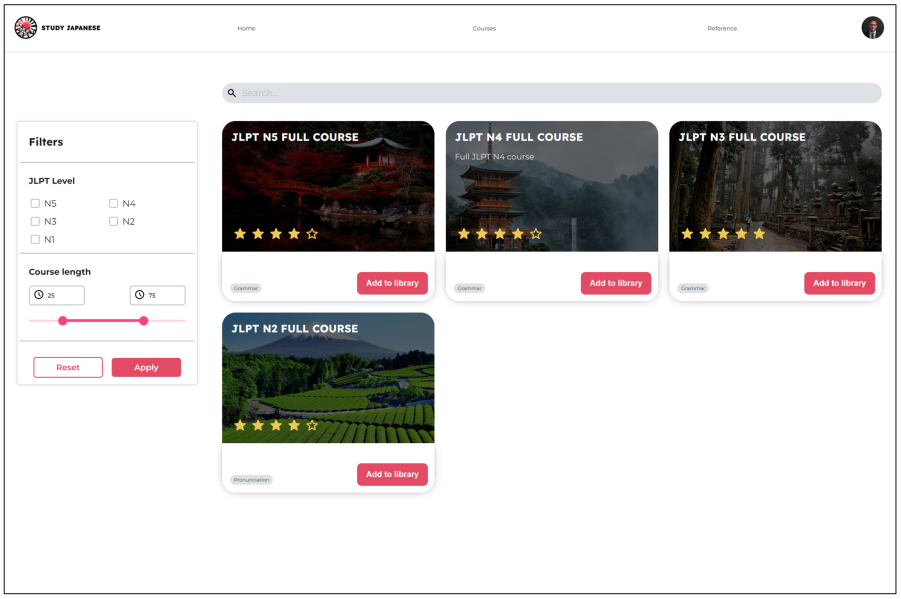
ДІАГРАМА КЛАСІВ



КАРТА САЙТУ



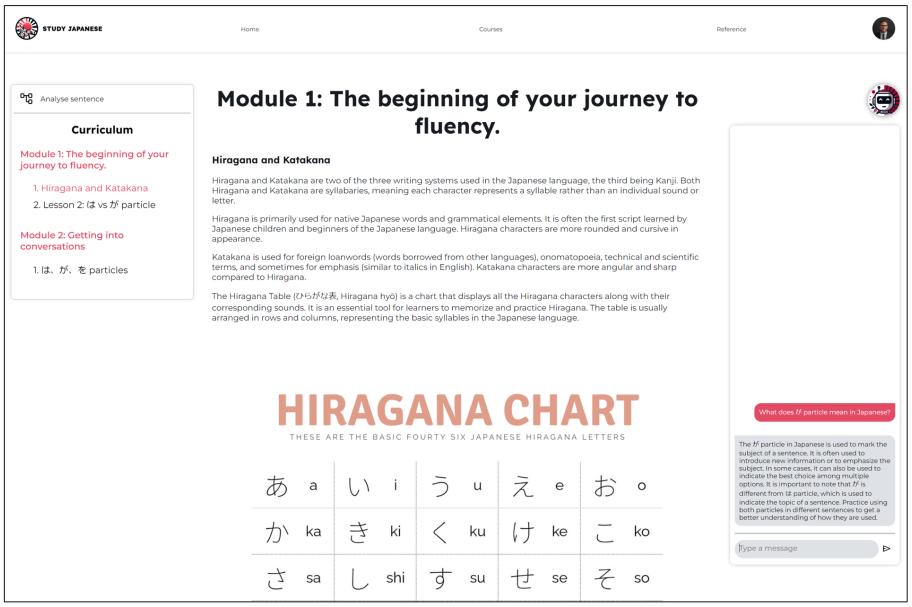
ЕКРАННІ ФОРМИ



Пошук колод слів та курсів

11

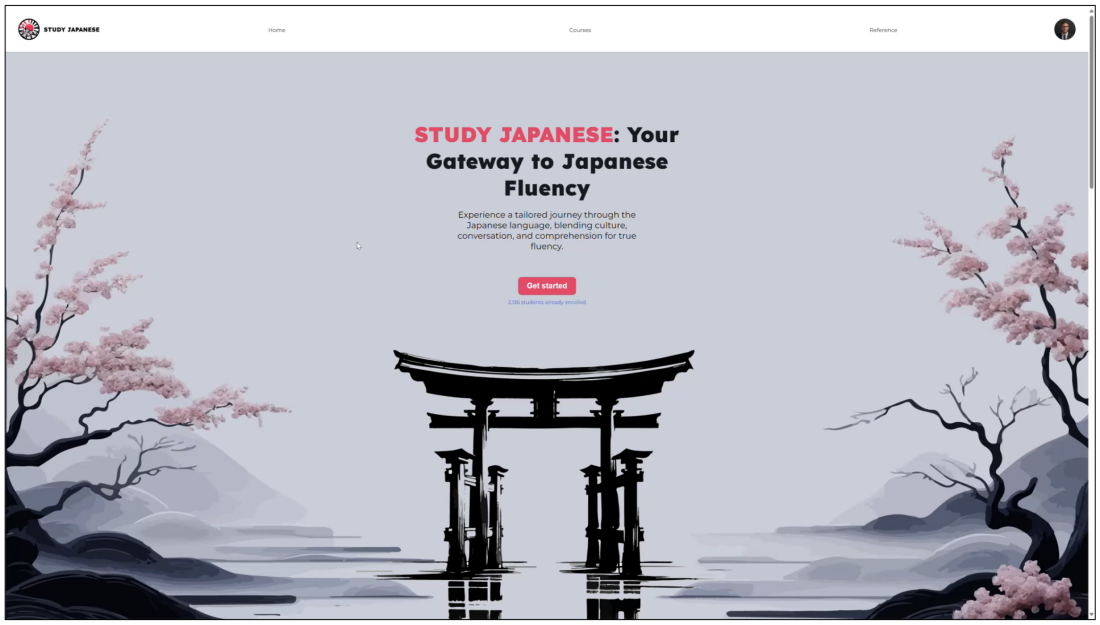
ЕКРАННІ ФОРМИ



Матеріали курсу

12

ДЕМОНСТРАЦІЯ РОБОТИ ЗАСТОСУНКУ



13

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Соколовський А.В. Розробка застосунку для вивчення японської мови з використанням мови Python та фреймворків Django, Angular/Жебка В.В., Соколовський А.В.// Матеріали всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в ІКТ». Збірник тез. 24.04.2024, м. Київ. К.:ДУІКТ – Стор. 105-107.
2. Соколовський А.В. Використання OpenAI API для реалізації боту-помічника з вивчення японської мови/Жебка В.В., Соколовський А.В.// Матеріали всеукраїнської науково-практичної конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. 15.05.2024, м. Київ. К.:ДУІКТ - Стор. 36-37.

14

ВИСНОВКИ

1. Проведено огляд специфіки вивчення японскьої мови, аналіз цільової аудиторії та існуючих засобів, а саме: Tofugu, Duolingo та LingoDeer; визначено вимоги до застосунку для вивчення японської мови.
2. Проведено огляд та аналіз засобів реалізації застосунку для вивчення японської мови, і обрано Python, TypeScript, Angular, VS Code, PyCharm, Django, Figma, GitHub та PostgreSQL.
3. Зпроєктовано архітектуру застосунку для вивчення японської мови за допомогою UML діаграм, зокрема діаграми класів, варіантів використання та розгортання.
4. Реалізовано застосунок для вивчення японської мови.
5. Протестовано застосунок, що визначило його справність.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

Вихідний код файлу Angular

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-search-bar',
  standalone: true,
  imports: [],
  templateUrl: './search-bar.component.html',
  styleUrl: './search-bar.component.scss'
})
export class SearchBarComponent {

}

<form class="search-bar">
    <button class="search-button">
        <img src="../../../assets/images/icons/search.svg"
alt="search-scope-icon">
    </button>
    <input type="text" class="search-input"
placeholder="Search...">
</form>

import { CommonModule } from '@angular/common';
import { Component } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { MatSliderModule } from
'@angular/material/slider'
import { NgModel } from '@angular/forms';

@Component({
  selector: 'app-filters',
  standalone: true,
  imports: [FormsModule, CommonModule,
MatSliderModule],
  templateUrl: './filters.component.html',
  styleUrl: './filters.component.scss'
})
export class FiltersComponent {
  leftValue = 25;
  rightValue = 75;
}

<form class="filters-container">
    <div class="form-section">
        <h2 class="filters-header">Filters</h2>
    </div>
    <div class="form-section">
        <h3 class="section-header">JLPT Level</h3>
        <div class="options">
            <label class="option"><input type="checkbox"
name="jlpt-level" value="n5">N5</label>
            <label class="option"><input type="checkbox"
name="jlpt-level" value="n4">N4</label>
            <label class="option"><input type="checkbox"
name="jlpt-level" value="n3">N3</label>
            <label class="option"><input type="checkbox"
name="jlpt-level" value="n2">N2</label>
            <label class="option"><input type="checkbox"
name="jlpt-level" value="n1">N1</label>
        </div>
    </div>
    <div class="form-section">
        <h3 class="section-header">Course length</h3>
        <div class="hours-input-container">
            <div class="input-box">
                <input type="number" class="hours-text"
name="leftInput" [(ngModel)]="leftValue">
            </div>
            <div class="input-box">
                <input type="number" class="hours-text"
name="rightInput" [(ngModel)]="rightValue">
            </div>
        </div>
        <div class="range-slider">
          <mat-slider color="accent" min="5"
max="100">
            <input matSliderStartThumb
name="leftThumb" [(ngModel)]="leftValue">
            <input matSliderEndThumb
name="rightThumb" [(ngModel)]="rightValue">
          </mat-slider>
        </div>
    </div>
    <div class="form-section">
        <div class="apply-buttons-container">
            <button class="reset button">Reset</button>
            <button class="apply button">Apply</button>
        </div>
    </div>
</form>

import { Component } from '@angular/core';

@Component({
  selector: 'app-course-cards-list',
  standalone: true,
  imports: [],
  templateUrl: './course-cards-list.component.html',
  styleUrl: './course-cards-list.component.scss'
})
export class CourseCardsListComponent {

}

<div class="courses-container">
    <div class="course-card">
        <div class="course-header"  style="background-
image:
url('../../../assets/images/courses/RedTrees.jpg');">
            <h2 class="course-name">TOP 1000 MOST
COMMON JAPANESE WORDS</h2>
            <p class="short-description">
```

```
            A full, comprehensive course aiming to teach
the student all neccessary points to pass JLPT N5 test.
          </p>
          <div class="rating">
            <img
src="../../../assets/images/icons/star_filled.svg"
class="star filled" alt="image of a drawn filled star">
            <img
src="../../../assets/images/icons/star_filled.svg"
class="star filled" alt="image of a drawn filled star">
            <img
src="../../../assets/images/icons/star_filled.svg"
class="star filled" alt="image of a drawn filled star">
            <img
src="../../../assets/images/icons/star_filled.svg"
class="star filled" alt="image of a drawn filled star">
            <img
src="../../../assets/images/icons/star.svg" class="star
non-filled" alt="image of a drawn non-filled star">
          </div>
        </div>
        <div class="add-container">
          <div class="tags">
            <span class="jlpt-tag">JLPT N5</span>
            <span class="jlpt-tag">JLPT N4</span>
            <span class="jlpt-tag">JLPT N3</span>
          </div>
          <button class="add-to-library-btn">Add to
library</button>
        </div>
      </div>

</div>
```

Вихідний код файлу Django

```
from django.contrib import admin
from .models import Deck, Card, UserCard,
DailyNewCardCount

admin.site.register(DailyNewCardCount)

class DeckAdmin(admin.ModelAdmin):
    list_display = ('title', 'description')
    search_fields = ('title', 'description')

admin.site.register(Deck, DeckAdmin)

def reset_all_user_cards(modeladmin, request,
queryset):
    queryset.update(
        ease_factor=2.5,
        interval=1,
        repetitions=0,
        review_date=None,
        new_card=True
    )
reset_all_user_cards.short_description = "Reset all
fields of selected user cards to default"

@admin.register(Card)
class UserCardAdmin(admin.ModelAdmin):
    list_display = ['id', 'deck', 'kanji', 'onyomi', 'kunyomi',
'meaning', 'example_sentences']

@admin.register(UserCard)
class UserCardAdmin(admin.ModelAdmin):
    list_display = ['id', 'user', 'card', 'ease_factor',
'interval', 'repetitions', 'review_date', 'new_card']
  actions = [reset_all_user_cards]

from django.db import models
from django.conf import settings

class Deck(models.Model):
    title = models.CharField(max_length=255)
    description = models.TextField(blank=True,
null=True)
    image =
models.ImageField(upload_to='deck_images/',
blank=True, null=True)

    def __str__(self):
        return self.title
class Card(models.Model):
    deck = models.ForeignKey(Deck,
related_name='cards', on_delete=models.CASCADE)
    kanji = models.CharField(max_length=10)
    onyomi = models.CharField(max_length=50,
blank=True, null=True)
    kunyomi = models.CharField(max_length=50,
blank=True, null=True)
    meaning = models.TextField()
    example_sentences = models.JSONField(default=list,
blank=True)

    def __str__(self):
        return self.kanji

class UserCard(models.Model):
    user =
models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE)
    card = models.ForeignKey(Card,
on_delete=models.CASCADE)
    ease_factor = models.FloatField(default=2.5)
    interval = models.FloatField(default=1)
    repetitions = models.IntegerField(default=0)
    review_date = models.DateTimeField(null=True,
blank=True)
    new_card = models.BooleanField(default=True)
```

```
    class Meta:
        unique_together = ('user', 'card')

    def __str__(self):
        return f"{self.user.username} - {self.card.kanji}"

class DailyNewCardCount(models.Model):
    user =
models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE)
    deck = models.ForeignKey(Deck,
on_delete=models.CASCADE, null=True)
    date = models.DateField()
    count = models.IntegerField(default=0)

    class Meta:
        unique_together = ('user', 'deck', 'date')

    def __str__(self):
     return f"{self.user.username} - {self.deck.title} -
{self.date} - {self.count} new words"

from rest_framework import serializers
from .models import UserCard, Card
class CardSerializer(serializers.ModelSerializer):
    class Meta:
        model = Card
        fields = ['id', 'kanji', 'onyomi', 'kunyomi', 'meaning',
'example_sentences']

class UserCardSerializer(serializers.ModelSerializer):
    card_details = CardSerializer(source='card',
read_only=True)
    review_date =
serializers.DateTimeField(format="%Y-%m-
%dT%H:%M:%S", input_formats=['%Y-%m-
%dT%H:%M:%S', 'iso-8601'])

    class Meta:
        model = UserCard
     fields = ['id', 'card_details', 'ease_factor', 'interval',
'repetitions', 'review_date']

from .views import review_card, fetch_daily_cards,
DeckCardsAPIView, DeckListView
from django.urls import path

urlpatterns = [
    path('review-card/', review_card, name='review-
card'),
    path('decks/<int:deck_id>/cards/',
DeckCardsAPIView.as_view(), name='deck-cards'),
    path('decks/<int:deck_id>/daily-cards/',
fetch_daily_cards, name='daily-cards'),
    path('decks/', DeckListView.as_view(), name='deck-
list'),
]

from datetime import timedelta
from rest_framework import serializers
from django.utils import timezone
from django.utils.timezone import now
from rest_framework.views import APIView
from rest_framework.response import Response
from .models import UserCard, Deck, Card,
DailyNewCardCount
from .serializers import UserCardSerializer,
CardSerializer
from rest_framework.decorators import api_view,
permission_classes
from rest_framework.permissions import
IsAuthenticated
from datetime import date
from django.db.models import F, Count, Q


class DeckCardsAPIView(APIView):
    def get(self, request, deck_id):
        try:
            deck = Deck.objects.get(id=deck_id)
            cards = Card.objects.filter(deck=deck)
            serializer = CardSerializer(cards, many=True)
            return Response(serializer.data)
        except Deck.DoesNotExist:
            return Response({"error": "Deck not found"},
status=404)

@api_view(['GET'])
@permission_classes([IsAuthenticated])
def fetch_daily_cards(request, deck_id):
    user = request.user
    deck = Deck.objects.get(id=deck_id)
    today = date.today()

    daily_count, created =
DailyNewCardCount.objects.get_or_create(
        user=user,
        deck=deck,
        date=today,
        defaults={'count': 20}
    )

    due_cards = UserCard.objects.filter(
        user=user,
        card__deck_id=deck_id,
        review_date__lte=timezone.now(),
        new_card=False
    ).select_related('card').order_by('review_date')

    new_cards_limit = max(0, daily_count.count)

    new_cards = UserCard.objects.filter(
        user=user,
        card__deck_id=deck_id,
        new_card=True
    ).select_related('card').order_by('?')[:new_cards_limi
t] if new_cards_limit > 0 else UserCard.objects.none()

    due_cards_serializer = UserCardSerializer(due_cards,
many=True)
    new_cards_serializer =
UserCardSerializer(new_cards, many=True)

    return Response({
        'due_cards': due_cards_serializer.data,
```

```
        'new_cards': new_cards_serializer.data
    })



@api_view(['POST'])
@permission_classes([IsAuthenticated])
def review_card(request):
    user_card_id = request.data.get('user_card_id')
    quality = int(request.data.get('quality'))
    user_card = UserCard.objects.get(id=user_card_id,
user=request.user)

    if quality == 1:
        user_card.ease_factor *= 0.85
        new_interval_minutes = 1
    elif quality == 2:
        user_card.ease_factor *= 0.9
        new_interval_minutes = 10
    elif quality == 3:
        new_interval_days = 1 if user_card.repetitions ==
0 else max(1, user_card.interval *
user_card.ease_factor)
        user_card.ease_factor += 0.1
    elif quality == 4:
        new_interval_days = 3 if user_card.repetitions ==
0 else max(3, user_card.interval *
user_card.ease_factor * 1.15)
        user_card.ease_factor += 0.15

    user_card.ease_factor = max(1.3,
user_card.ease_factor)

    if quality in [1, 2]:
        user_card.interval = new_interval_minutes / 1440
        user_card.review_date = now() +
timedelta(minutes=new_interval_minutes)
    else:
        user_card.interval = new_interval_days
        user_card.review_date = now() +
timedelta(days=new_interval_days)

    if user_card.new_card:
        today = date.today()
        deck_id = user_card.card.deck_id
        daily_count =
DailyNewCardCount.objects.filter(user=request.user,
date=today, deck_id=deck_id).first()
        if daily_count:
            daily_count.count -= 1
            daily_count.save()

    user_card.new_card = False
    user_card.repetitions += 1 if quality >= 3 else 0
    user_card.save()

    return Response({
        'status': 'success',
        'next_review_date':
user_card.review_date.isoformat(),
        'interval_days': user_card.interval,
        'ease_factor': user_card.ease_factor,
        'repetitions': user_card.repetitions
    })

class DeckListView(APIView):
    permission_classes = [IsAuthenticated]

    def get(self, request):
        today = timezone.now().date()
        decks = Deck.objects.annotate(
            reviewWordsCount=Count('cards__usercard',
filter=Q(cards__usercard__user=request.user,
cards__usercard__review_date__lte=timezone.now())),
            newWordsCount=Count('cards__usercard',
filter=Q(cards__usercard__user=request.user,
cards__usercard__new_card=True))
        )
        serializer = DeckSerializer(decks, many=True)
        return Response(serializer.data)

class DeckSerializer(serializers.ModelSerializer):
    imageUrl = serializers.ImageField(source='image')
    reviewWordsCount = serializers.IntegerField()
    newWordsCount = serializers.IntegerField()

    class Meta:
        model = Deck
      fields = ('id', 'title', 'imageUrl', 'reviewWordsCount',
'newWordsCount')
```