Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра інформатики

Курсова робота

освітній ступінь – бакалавр

на тему: «Розробка віртуального співрозмовника для практики розмовних навичок німецької мови»

Виконала: студентка 3-го
року навчання,
Спеціальності
122 «Комп’ютерні науки»

Прочна Софья

Керівник Тригуб О. С.

Доцент кандидат наук

«» травня 2024 р.

Київ – 2024

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри
інформатики

Гороховський С. С.

_____ (підпис)

„____”_____2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту_____ факультету _____ _____курсу

ТЕМА _____

Зміст ТЧ до курсової роботи:

Анотація

Вступ

1 Розділ 1. Дослідження та аналіз предметної області

2 Розділ 2 Проектування та розробка застосунку

Висновки

Список літератури

Дата видачі „___” _____ 2024 р.

Керівник _____ (підпис)

Завдання отримав_____ (підпис)

Календарний план виконання роботи

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на дипломну роботу.	20.11.2024	
2.	Огляд літератури за темою роботи.	1.12.2024	
3.	Дослідження та аналіз аналогів	16.12.2024	
3.	Складання плану курсової роботи	22.12.2024	
4.	Вибір інструментарію для роботи	15.01.2024	
5.	Написання текстової частини	15.02.2024	
6.	Проміжний контроль виконання роботи	16.03.2024	
7.	Проектування практичної частини роботи	20.03.2024	
8.	Створення слайдів для доповіді	26.04.2024	
8.	Оформлення її згідно з вимогами й подання на відгук науковому керівнику	02.05.2024	
10.	Корегування роботи за результатами попереднього захисту.	1.05.2024	
11.	Остаточне оформлення пояснювальної роботи та слайдів.	3.05.2024	
12.	Захист курсової роботи (проекту)	22.05.2024	

Науковий керівник Тригуб Олександр Семенович
Виконавець курсової роботи Прочна Софья

ЗМІСТ

Анотація

Робота присвячена проектуванню додатку для практики розмовних навичок під час вивчення німецької мови. В рамках дослідження пояснюється вибір фреймворку Django для створення Backend частини. Особливості та причини використання фреймворку React Native для створення Frontend частини. Поетапно розглядається процес створення навчального застосунку. Описуються особливості та складності розробки за допомогою описаного вище інструментарію.

Вступ

В сучасному динамічному світі, де знання іноземної мови – є необхідним, проте реальна інтеракція з носіями мови обмежена або складна. Віртуальний співрозмовник міг би надати необхідну інтерактивну практику та сприяти ефективнішому вивченню мови.

Глобалізація відкрила перед нами двері вільного пересування, роботи на закордонний ринок, подорожей, спілкування з людьми з різних куточків світу та споживання іноземного контенту, проте це все вимагає від нас знання розмовної іноземної мови. Один зі способів вивчення мови – є спілкування онлайн з носієм, але це підходить не всім, по-перше - це досить дорогий вид навчання, а по-друге вимагає чітко виділеного часу в графіку і по-третє є вірогідність помилитись з вибором носія. Рішення запропоноване в даній роботі – це онлайн переписка зі штучним інтелектом, яка імітує діалог зі справжнім носієм мови. Це рішення є набагато дешевшим для користувача та зручним, оскільки не потребує певно виділеного часу, діалог може бути поновлений в будь-який час з того ж місця, де був зупинений.

Мета цієї роботи полягає в створенні віртуального помічника для вивчення німецької мови. Створення інтуїтивно зрозумілого застосунку для практики розмовних навичок, збільшення словникового запису та закріплення знань за допомогою тестів.

Щоб виконати поставлене завдання, були виокремленні наступні підзавдання:

- створення кабінету користувача для збереження прогресу;
- розробка модуля для практики розмовних навичок з використанням

готового інструментарію штучного інтелекту;

- інтеграція системи збільшення словникового запасу;
- розробка модулю для проведення тестів, які допоможуть закріпити знання та відслідковувати прогрес у вивченні мови;
- забезпечення синхронізації даних користувача для можливості використання застосунку з різних пристроїв;
- створення інтуїтивно зрозумілого та унікального дизайну.

Розділ 1. Дослідження та аналіз предметної області

1.1. **Duolingo**

Duolingo – це один з найпопулярніших навчальних застосунків серед молоді. Він пропонує вивчення іноземної мови в ігровій формі, де користувачі навчаються через серію коротких уроків, які включають переклад слів, вправи на вибір правильного варіанта відповіді та формування речень. Цей додаток пропонує широкий вибір мов, як рідної так і мови для вивчення. Особливість Duolingo полягає в їхній системі мотивації, де користувачі заробляють бали та переходять на нові рівні. Виробники застосунку пропонують нам класичний формат вивчення мови, без використання штучного інтелекту та ведення діалогу.

1.2. **Rosetta Stone**

Rosetta Stone – це унікальний додаток, який пропонує вивчення мови через повне занурення, це досягається відсутністю перекладу слів і завдань на рідну мову. Завдяки інтуїтивно зрозумілому інтерфейсу та візуальних елементів, користувачі здобувають мовні навички, спираючись на асоціації та впізнавання. Проте такий підхід більше підходить для учнів мов з досвідом, оскільки новачків він може налякати і відштовхнути.

1.3. **Tandem**

Tandem – це інноваційний застосунок для мовного обміну, який з'єднує користувачів з носіями мови по всьому світу. Це забезпечує практичний досвід у використанні мови в натуральних діалогах і справжньому спілкуванні. Користувачі можуть партнера для спілкування, відповідно до своїх інтересів та цілей, що додає персоналізації навчанню. Tandem підтримує всі види зв’язку, а саме: текстові повідомлення, аудіо та відео дзвінки. Цей додаток на відміну від попередніх є платним. Недоліком може

бити планування дзвінка, щоб зв’язатись з партнером треба підібрати обом зручний час. А також підбір носія мови, що вимагає певного часу.

1.4. **Busuu**

Busuu – це додаток, який охоплює повний спектр мовного навчання з уроками, які включають читання, письмо, слухання та говоріння. Унікальна особливість Busuu полягає у соціальній платформі, де користувачі можуть отримувати відгуки від носіїв мови щодо вправ і завдань, які вони виконують. Це дозволяє отримати реальний зворотний зв'язок і покращити якість навчання.

1.5. **Mondly**

Mondly – це інноваційний застосунок, що використовує штучний інтелект для створення персоналізованого досвіду навчання. Штучний інтелект у Mondly адаптується до рівня володіння мовою користувача і стилю навчання, пропонуючи оптимізовані вправи та діалоги. Цей додаток також має чат-бот, який дозволяє користувачам практикувати розмовні навички в контрольованих, але реалістичних розмовних сценаріях.

1.6. **Replika**

Replika – не є спеціалізованим інструментом для вивчення мов вона представляє собою унікального супутника зі штучним інтелектом, з яким можна вести глибокі та значущі бесіди. Використовуючи такий додаток може бути корисним для вивчення мови, оскільки це забезпечує практику у побудові речень та підтриманні діалогу. Проте, оскільки він спрямований не лише на вивчення, то він не має повної функціоналу начального додатку.

1.7. **SimSimi**

SimSimi – це розмовна програма із вбудованим штучним інтелектом. Особливість додатку в тому, що він вміє будувати емоційно забарвлені діалоги, тому вони схожі на людську мову. Він може використовуватися для неформальної практики мови, пропонуючи натуральні відповіді на текстові повідомлення користувачів. Цей застосунок допомагає покращувати розуміння і використання мовних ідіом та виразів.

1.8. **Висновки до розділу 1**

Сучасний ринок пропонує широкий ринок рішень для вивчення іноземних мов, кожне з яких має свою унікальну методологію та особливості. Такі застосунки як Duolingo, Rosetta Stone, і Busuu надають користувачам структуровані уроки, що охоплюють граматику, лексику та фразеологію, використовуючи методи повторення та гейміфікації для підвищення мотивації. На відміну від них, соціальна платформа як Tandem забезпечують інтерактивну практику через безпосереднє спілкування з носіями мови, що допомагає вдосконалювати мовні навички в онлайн форматі. Застосунки, що використовують штучний інтелект, такі як Mondly та Replika, підвищують індивідуальну адаптацію навчального процесу імітують реальні розмовні сценарії і надавати розмові емоційного забарвлення. Проте немає оптимального рішення, яке об’єднує практику розмовних навичок та закріплення знань за допомогою тестів та тренування словнику. Саме це і стало визначальними у формуванні напрямків подальшої роботи над розробкою нового застосунку в рамках даної дослідницької роботи.

Розділ 2. Проектування та розробка застосунку

2.1. Опис етапів розробки

Розробка застосунку складається з багатьох етапів, набутки після кожного з них стають фундаментом для початку наступного етапу. До ключових етапів розробки навчального застосунку, який розглядається в цьому проєкті, відноситься:

- Планування і аналіз області

  Спочатку потрібно провести аналіз аналогічних застосунків, оцінити їх переваги та недоліки, це було розглянуто в попередньому розділі. Це дозволяє побудувати концепцію, сформувати мету та прокласти шлях розвитку.

- Розробка дизайну застосунку

  Далі розробка застосунку концентрується на проєктувані прототипу та створені макету. Це стає можливим, оскільки після першого етапу вже є основний ідейний каркас та необхідний функціонал, також виокремлюється пріоритетність блоків розробки. Створення візуального дизайну має концентруватись на основних принципах UX/UI дизайну. Він має бути інтуїтивно зрозумілим, когнітивно привабливим, забезпечувати легку навігацію та високий рівень користувацького досвіду.

- Розробка

  Маючи основні візуальні компоненти, запланований функціонал, відсортований за рівнем важливості розробка переходить до етапу кодування. На цьому етапі веб-застосунків і мобільних додатків є кілька підходів: проектування спочатку серверної частини, а потім

навпаки, проектування в зворотньому порядку, а також паралельне проектування. При розробці віртуального співрозмовника для вивчення німецької мови було обрано останній варіант – паралельне проектування серверної та клієнтської частини. Backend, тобто серверна частина спроектовані на Python орієнтованому фреймворці – Django, a frontend був написаний за допомогою мультиплатформеного фреймворку React Native, який дозволяє на одній кодовій базі створити веб додаток і мобільний застосунок, який підходить, як для IOS так і для Android платформи. На цьому етапі також відбувається корекція плану, оскільки розробка стикається з певними складнощами реалізації. В контексті даного проекту ніяких суттєвих змін плану не відбулося, весь запланований на самому початку функціонал було реалізовано, дизайн прототип також не зазнав змін.

- Тестування

  Критично важливий етап, оскільки він вимагає надвисокого рівня уваги. Тут важливо відслідкувати помилки та недоліки. Критично протестувати всі функціональні блоки, кнопки та процеси, аби нічого не пропустити. Також важливо провести тестування на вразливості та витривалість, оскільки в разі масштабного розгортання застосунку, він має бути готовим для використання певною кількістю користувачів. Сервер має приймати певну кількість запитів в хвилину, а розробники мають відстежувати статистику роботи серверів, для того, щоб за потреби збільшити потужність сервера, або принаймні зберегти його від зупинки.

Під час типової розробки за етапом тестування йде етап розгортання, тобто розміщення на виробничих серверах, налаштування середовища і забезпечення постійної підтримки і оновлень. На цьому етапі до застосунку

отримують доступ справжні користувачі. Потім додаток має оновлюватись та підтримуватись, тобто продукт має зберігати потрібну потужність серверів та пропонувати максимально можливий рівень відмовостійкості. Також є можливість продовжити роботу над розвитком застосунку за допомогою оновлень, які користувачі зможуть отримати за успішної реалізації.

Отже, розробка застосунку в рамках даного дослідження складалась з чотирьох основних етапів, а саме: аналіз області та проектування, за ним створення візуального прототипу, далі розробка і завершальним етапом стало тестування роботи застосунку.

2.2. Постановка задачі та опис предметної області

Застосунок орієнтований на вивчення німецької мови, як іноземної. Він використовує систему для стеження за прогресом користувачів, їхнім рівнем, вивченим лексиконом, а також дозволяє користувачам взаємодіяти через чат.

2.2.1 Модель даних

Під час аналізу предметної області було виділено такі сутності:

- User: сутність, яка зберігає основну інформацію про користувачів. Використовується при вході і реєстрації.
- UserProgress: сутність для фіксації прогресу користувача, вивчених ним слів, пройдених рівнів та питань в них, а також діалогу закріпленого за користувачем. Вона створюється автоматично під час реєстрації та оновл.ється під час проходження питань та слів.
- Level: зберігає можливі рівні володіння мовою, які використовуються саме в цій іноземній мові. Ідентифікують складність тем та питань

для вивчення.

- Topic: сутність, яка описує теми створені для навчання та спілкування. Вони поділені на рівні, мають назву, опис і рівень. Приклад тем може бути: "Сім'я", "Подорожі", тощо.
- TopicProgress: сутність для відстеження прогресу користувача по конкретній темі. Вона допомагає запам’ятовувати, де користувач закінчив вивчення конкретної теми і з якого місця необхідно продовжити.
- Question: сутність створена для питанньь, що використовуються в тестах, варіантів відповіді та правильної відповіді.
- Lexemes: сутність, яка зберігає слова, ідіоми та фрази, які вивчаються користувачами та їх переклад. Вона використовується для створення масиву варіантів відповідей на питання, а також для створення карток для вивчення словника.
- Message: сутність, яка фіксує повідомлення, які відправляються в чаті між користувачем та штучним інтелектом для збереження діалогу.

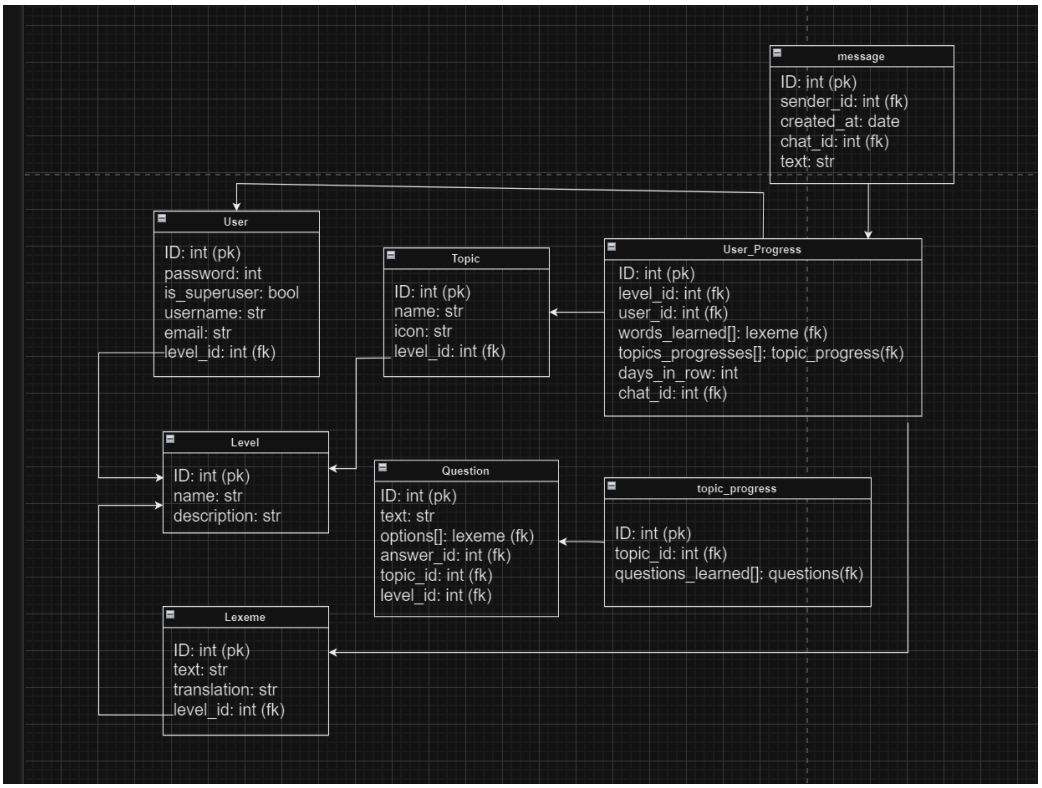

Рисунок 2.2.1.1 – Модель даних застосунку

2.2.2 Можливості користувача

Користувач – це єдина роль описана у даному застосунку, її отримує будь-яка зареєстрована людина. Користування додатком без авторизації неможлива, оскільки вона дозволяє зберегти прогрес користувача, вивченні ним питання і слова, щоб наступного разу він міг повернутись на те саме місце, де закінчив. У ході дослідження для користувача були виділені наступні можливості:

- Переглядати інформацію про власні здобутки в навчанні – загальний прогрес, кількість вивчених слів та пройдених тестів, прогрес по кожній темі.
- Отримувати інформацію про питання та варіанти відповіді, а також обирати правильну.
- Отримувати інформацію про лексеми запропоновані до вивчення та

помічати їх як опрацьовані.

- Переглядати інформацію про всі наявні теми, закріпленні на навчальному рівні, який відповідає рівню користувача.
- Відправляти повідомлення, які зберігаються для подальшого використання для потреби.
- Реєструвати акаунт, в якому вказується електронна пошта, рівень володіння мовою, ім’я та пароль.
- Авторизуватись в акаунт використовуючи ім’я користувача та зафіксований за ним пароль. А також виходити з акаунту. і

2.3. Розробка концептуального дизайну інтерфейсу

Для забезпечення чіткого бачення продукту, прорахування потрібного часу на розробку та виявлення потенційних проблем і питань, наступним етапом розробки було створення концептуального дизайну інтерфейсу. Дизайн сучасного застосунку в першу чергу має відповідати сучасним стандартам UI/UX, а також бути інтуїтивно зрозумілим користувачу та конкурентоспроможним на ринку аналогічних додатків. Основоположними принципами дизайну стали зрозумілість, консистенція та естетична привабливість. Всі елементи інтерфейсу розроблялися з метою мінімізації когнітивного навантаження на користувача, щоб спростити процес навчання та зробити його більш приємним та ефективним.

Інструментом для створення макету застосунку було обрано онлайн-сервіс Figma, оскільки він дозволяє зробити макет інтерактивним, налаштувати розміри елементів і шрифтів відповідно обраному екрану, підібрати потрібний шрифт та колір і такі елементи, як градієнти, розмиття та затемнення. Основна дошка ідей була організована з використанням різних

дизайн-концептів навчальних застосунків зібраних у соціальній мережі Pinterest.

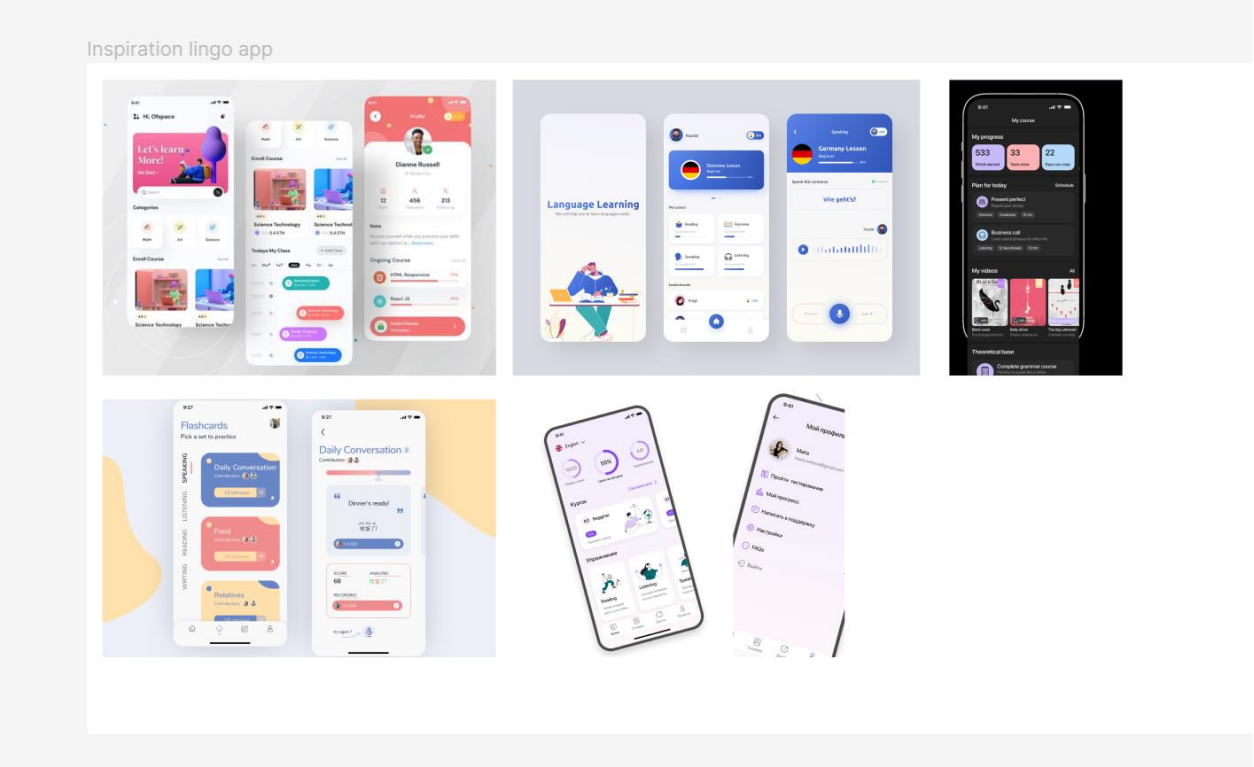

Рисунок 2.3.1 – Дошка ідей для створення дизайну застосунку

Домінуючими кольорами для проєкту були обрані білий, світло-рожевий, світло-жовтий та темно-фіолетовий, вони не створюють візуального шуму та сприяють концентрації на матеріалі. Емоджі були інтегровані для надання додатку емоційного забарвлення, аби зробити інтерфейс більш дружнім та сучасним.

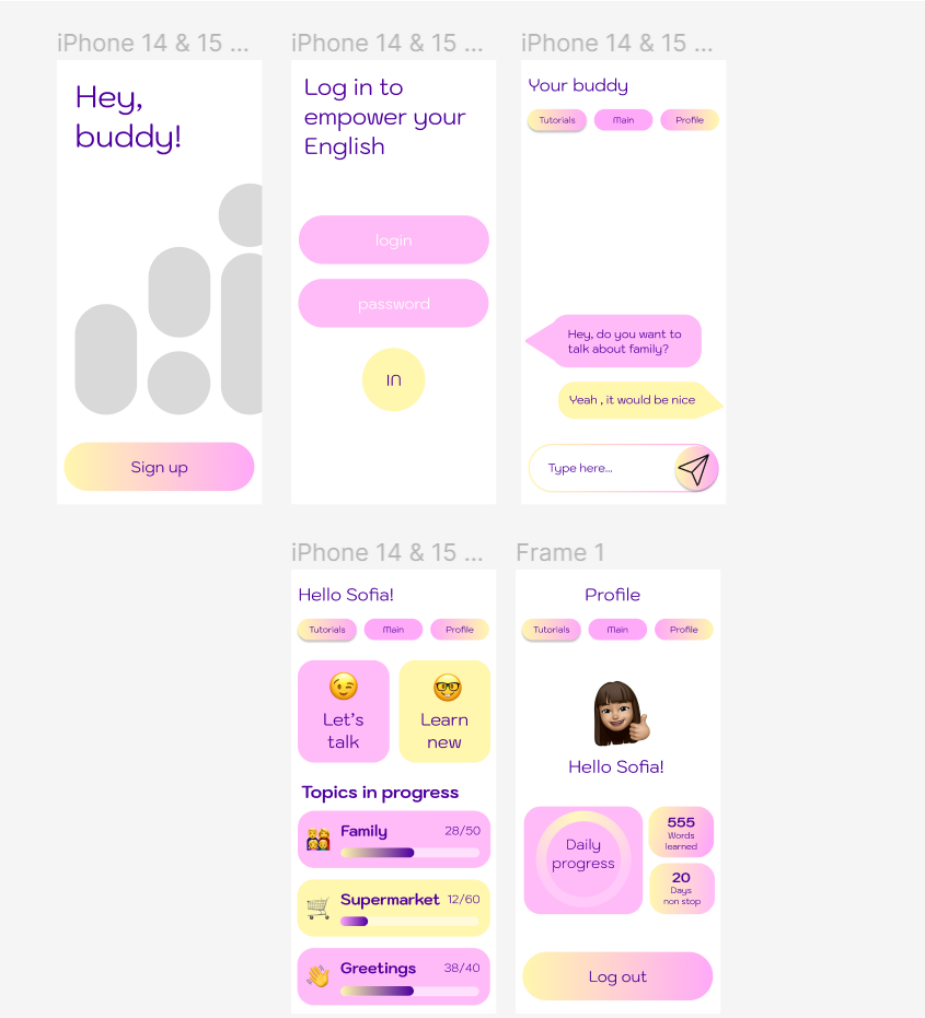

Рисунок 2.3.2 – Дизайн-макет застосунку

Розробка концептуального дизайну інтерфейсу стала ключовим етапом у процесі створення застосунку, цей підхід дозволив окреслити об'єм необхідної роботи, врахувати всі недоліки до стадії програмування, а також врахувати користувацькі вподобання.

2.4. Розробка серверної частини

2.4.1 Визначення архітектури сервера та вибір стеку технологій

Під час розробки серверної частини застосунку слід враховувати загальноприйняті важелі якісно спроектованого backend. Застосунок має відповідати наступним критеріям:

- легка масштабованість, для того щоб обробляти велике навантаження

та обсяг даних без зниження продуктивності;

- висока надійність, щоб користувачі могли покладатися на застосунок і не втрачати дані;
- безпека інформації, аби захищати дані користувачів від несанкціонованого доступу;
- максимальна ефективність та економне використання ресурсів;
- зрозумілість та зручність у використанні як для розробників, так і для користувачів.

Для визначення найкращого технологічного стеку для розробки backend частини проєкту було розглянуто чотири популярні технології: Node.js, Django, Go та Rust. Кожна з них в контексті даної роботи мала певні переваги та недоліки. Наприклад, Node.js підтримує асинхронність, підтримує безліч бібліотек для розробки, є досить популярною, тому має велику спільноту та серйозну навчальну базу з різними статтями і туторіалими. В цей же час Django, який є не менш популярним сереж розробників, пропонує високий рівень абстракції, вбудовані захистами від багатьох атак, що підвищує рівень безпеки, проте має відносно нижчу продуктивність порівняно з Node.js та Go при обробці одночасних запитів через синхронну природу Python. Go в свою чергу дає можливість легкої масштабованості та має вбудовану підтримку для створення мікросервісів та високу продуктивність. До недоліків можна віднести обмежену кількість бібліотек порівняно з Node.js та Python, оскільки Go це більш сучасна мова програмування. Ще один варіант – це Rust, до його переваг відносяться високий рівень безпеки та продуктивності, але одночасно він є менш зручний для швидкої розробки веб-застосунків та має обмежену кількість веб-фреймворків. З огляду на всі ці факти було обрано Django.

Не менш важливою частиною технологічного стеку є база даних. Під час проектування застосунку вибір стояв між реляційною і не реляційною базою даних, вони відрізняються за своєю структурою, способами зберігання даних та підходами до обробки запитів.

Реляційні бази даних зазвичай організовані у формі таблиць, де кожен рядок представляє окремий запис, а кожен стовпчик - поле з певним типом даних. містять рядки та стовпці. Реляційні бази даних використовують строго визначені схеми для опису структури даних. Між таблицями можна встановлювати зв'язки за допомогою первинних та зовнішніх ключів. Це дозволяє ефективно зберігати та отримувати складні дані.

Нереляційні бази даних, натомість, не використовують жорсткої реляційної структури. Вони можуть зберігати не тільки різні типи даних, а й дані у різних форматах, таких як JSON, граф, ключ-значення тощо. Це робить їх більш гнучкими та масштабованими, але може ускладнити доступ до даних та підтримку складних зв'язків.

Обидва типи баз даних мають свої переваги та недоліки, вибір між ними залежить в контексті цієї роботи залежав від вимог до масштабованості, структури даних, складності запитів та потребою в надійній, потужній і гнучкій системі підтримки, тому було обрано реляційний тип баз даних і PostgreSQL для роботи з ними.

2.4.2 REST

REST – це архітектурний підхід до розробки веб-сервісів і не тільки. Розділення між frontend, розробленим з використанням React Native, та backend на базі Django ідеально ілюструє модель клієнт-сервер REST. Це розділення дозволяє розвиток додатку незалежно, підвищуючи модульність і масштабованість додатку. Django REST Framework спонукає до

дотримання принципів REST, один з них - використання HTTP-методів , а саме: GET, POST, PUT, DELETE, що стандартизує спосіб взаємодії клієнтів з сервером. Це робить API передбачуваним і легшим для використання. Ще один принцип - кожен запит від клієнта до сервера повинен містити всю необхідну інформацію для його обробки. Сервер не зберігає стан сеансу між запитами. Наступний REST принцип - кожен ресурс ідентифікується унікальним адресним покажчиком (URL). А також, REST використовує формати даних, такі як JSON або XML, для обміну даними між клієнтом і сервером.

Отже, використання REST під час розробки дозволило створити гнучку систему API, яку легко адаптувати та оптимізувати. Такий підхід до розробки дозволяє клієнтським додаткам отримувати, створювати, оновлювати та видаляти дані на сервері, забезпечуючи максимальну продуктивність та ефективність взаємодії.

2.4.3 Django

Після аналізу варіантів, описаного вище, для розробки backend було обрано фреймворк Django - високорівневий фреймворк, написаний на Python. Його головна привабливість в очах розробників криється в філософії «батарейок», яка передбачає наявність багатьох вбудованих рішень для шаблонних задач веб-розробки, що значно пришвидшує процес розробки та зменшує кількість коду, необхідного для створення застосунку.

Основні переваги цього фреймворку можна описати наступним переліком:

- Django дозволяє розробникам зосередитися на безпосередньому написанні застосунку без потреби зайвого занурення в деталі, наприклад налаштування протоколу HTTP, обробка запитів та відповідей тощо;

- він надає вбудовану підтримку для запобігання типовим атакам, таким як SQL ін'єкціями, міжсайтовий скриптинг, підробка міжсайтових запитів та накладання кліків;
- розвинена спільнота, оскільки це один з найпопулярніших веб-фреймворків, Django має активну спільноту розробників і обширну документацію, що робить легким знаходження рішень та отримання допомоги за потреби;
- потужна ORM (Object-Relational Mapping) система, що дозволяє розробникам ефективно взаємодіяти з базою даних, використовуючи напряму код на Python, і завдяки цьому уникати побудови складних SQL запитів;
- пакетний менеджер pip, який не тільки пропонує безліч зручних бібліотек, а й робить управління залежностями легким та ефективним.

Таким чином, зважаючи на всі перераховані вищі чинники і аргументи вибір Django для написання серверної частини коду цього застосунку є оптимальним. В процесі розробки цей факт багато разів підтвердився його високою продуктивністю, безпекою, підтримкою великої спільноти, здатністю до швидкого розвитку проектів та легкої інтеграції з різними сервісами та базами даних.

2.4.4 PostgreSQL

PostgreSQL є системою управління реляційними базами даних, вона відповідає вимогам цього проєкту. Першим важливим фактором є широкий спектр стандартних і розширених функцій SQL, а саме складні запити, відкладені операції, загальні таблиці виразів та вкладені запити. По-друге PostgreSQL є універсальною для роботи з різноманітними типами даних, адже вона підтримує не тільки стандартні типи даних, а й також JSON,

XML та інші формати. Третьою перевагою є легка масштабованість, оскільки ця система управління підтримує як вертикальне, так і горизонтальне масштабування, що дозволяє управління великими даними, робить її гнучкою і оптимізованою. Не менш важливим фактором є високий рівень безпеки, а саме аутентифікація, шифрування даних, управління доступом та захист від SQL-ін'єкцій.

Отже, PostgreSQL виявилася зручною і легкою в керуванні системою управління баз даних. Вона відповідає всім сучасним вимогам до реляційних баз даних є легко масштабованою, підтримує різні типи даних, забезпечує високий рівень захисту даних. Окрім того, активна спільнота розробників і постійне оновлення забезпечують високу адаптивність та достатню кількість інформації.

2.5. Розробка клієнтської частини

2.5.1 React Native

Віртуальний співрозмовник – застосунок, який за концепцією має бути доступний в будь-якому місці, в будь-який час. Ідея зупинитись на рівні розробці лише веб-додатку без підтримки на мобільних пристроях була відкинута на самому початку проектування. Користувач повинен мати віртуального співрозмовника прямо в телефоні, без потреби кожен раз вмикати ноутбук, завантажувати браузер і тільки потім отримувати доступ до платформи. Враховуючи це, для розробки багатоплатформеного додатку, який підтримує веб застосунок та мобільну версію, як для IOS так і для Android. Ось основні причини, через які фреймворк React Native став вибором для цього проекту:

- він дозволяє розробляти додатки, які за допомогою одної кодової бази можуть працювати одночасно на різних платформах;

- фреймворк підтримує оновлення в реальному часі, що є корисним і зручним не тільки під час розробки, дозволяє вносити правки в код і бачити результати одразу, а й після релізу, оскільки випускати оновлення безпосередньо через мережу, що дозволяє швидко вносити зміни в додатки без необхідності повторної публікації через магазини додатків;
- компоненти React Native компілюються в нативний код, що забезпечує високу продуктивність і відмінну відгук додатків;
- використання JavaScript та багата екосистема компонентів та бібліотек, React Native спрощує та прискорює процес розробки.

Загалом React Native виявився вдалим вибором, надаючи розробникам потужний інструмент для створення мультиплатформеного додатку для вивчення німецької мови. Цей фреймворк дозволив зберегти багато часу, оскільки з одної кодової бази створено і мобільний застосунок і веб додаток.

2.5.2 Expo

Expo – це зручна платформа, яка забезпечує розробників всіма необхідними інструментами для створення мобільних додатків. Платформа дозволяє програмувати додатки для Android та iOS без безпосередньої взаємодії з нативними компонентами операційної системи, що робить процес створення доступним та простим.

До переваг Expo відноситься також легкий та швидкий початок, оскільки всі необхідні бібліотеки та залежності вже налаштовані в середовищі. Також ця платформа пропонує доступ до великого набору API. які включають функціонал камери, системи оповіщень, сенсорів та багато іншого без необхідності використання нативного коду. Одна з

найважливіших переваг Expo – це оновлення в реальному часі, з одного боку це пришвидшує розробку, оскільки всі зміни можна одразу тестувати, а з іншого боку дозволяє розробникам надсилати оновлення безпосередньо на мобільні пристрої користувачів без необхідності повторної публікації застосунків у магазинах додатків. Expo надає обладнання для збірки в хмарі, оскільки генерує готові до використання бінарні файли для iOS та Android без встановлення додаткових SDK на локальні машини розробників.

Expo Go – застосунок для тестування, який достатньо просто встановити цей додаток на мобільний пристрій та під’єднати ноутбук та телефон до однієї мережі, потім потрібно запустити проєкт за допомогою команди npx expo start і сканувати QR-код. Після цього продукт просто з’являється на мобільному пристрої з повним функціоналом.

Отже, Expo – це ідеальний вибір в контексті цього проєкту, завдяки своїй здатності швидко інтегруватись з різноманітними API, забезпечувати легке тестування і розповсюдження оновлень, а також ефективне використання ресурсів за допомогою хмарних збірок, Expo робить процес розробки більш простим та доступним. Також ця платформа легко компілюється з React Native.

2.5.3 Компоненти інтерфейсу користувача

Архітектура клієнтської частини складається з наступних екранів:

- екрани входу та реєстрації, вони надають форми для авторизації користувачів або реєстрації нового облікового запису. Ці екрани обробляють введення користувачів (ім'я користувача, пароль) і передають їх на серверну частину;
- екран профілю дозволяє користувачам переглядати свій прогрес, який

відображає загальний прогрес на курсі, кількість вивчених тестів та загальну кількість пройдених слів;

- головне меню, куди користувачві потрапляють після входу, де вони можуть перейти до різних частин додатку, наприклад, розпочати бесіду зі штучним інтелектом, переглянути список тем або отримати доступ до навчальних матеріалів;

- екран з тестами, на якому відображено тематичне питання і варіанти відповіді та кнопкою перевірити, аби дізнатись відповідь обрана правильна чи не правилна;

- екран для вивчення слів, на якому відображається слово англійською мовою з одної сторони і його німецький переклад з іншої, а також кнопка перейти до наступного слова;

- екран бесіди, де користувачі взаємодіють з віртуальним співрозмовником. Він включає текстове поле для введення повідомлень і область відображення історії бесіди.

Для керування даними між компонентами в додатку використовується система управління станами. Цей підхід включає використання вбудованих хуків React, таких як useState та useContext. Ці хуки дозволяють компонентам самостійно управляти своїм станом без залучення зовнішніх бібліотек чи засобів. useState використовується для оновлення та слідкування за змінами стану в межах окремого компонента, тоді як useContext дозволяє доступ до глобальних даних без необхідності прокидання props на кожному рівні ієрархії.

2.5.4 Навігація

В застосунку використовується один з найпопулярніших компонентів навігації - Stack Navigator, який знаходиться в бібліотеці React Navigation.

Навігатор дозволяє організовувати переходи між різними компонентами, дозволяє легко організувати різні екрани у формі стеку, де кожен новий екран відображається на вершині попереднього, утворюючи логічну послідовність переходів у застосунку. Він управляє переходами в стилі останній прийшов, перший пішов. Також компонент дозволяє формувати заголовки для кожного екрану. Щоб очікування між переходами і завантаження між сторінками було зрозумілим використовується індикатор завантаження реалізований в компоненті ActivityIndicator.

```
<Stack.Navigator>
    <Stack.Screen name="Login" component={LoginScreen} />
    <Stack.Screen name="Main" component={MainScreen} />
    <Stack.Screen name="Questions to Learn" component={TopicQuestionsLearn} />
    <Stack.Screen name="Messenger" component={MessengerScreen} />
    <Stack.Screen name="Sign Up" component={SignupScreen} />
    <Stack.Screen name="Profile" component={ProfileScreen} />
    <Stack.Screen name="Tutorial" component={TutorialScreen} />
</Stack.Navigator>
```

Рисунок 2.5.4.1 – Стек навігації

Отже, навігація – це необхідний компонент в мобільному додатку, вона дозволяє користувачу інтуїтивно пересуватись між екранами забезпечуючи швидкий доступ до основних функцій. Якісно запрограмована навігація покращує досвід користувачів, оскільки інструменти навігації не тільки підвищують зручність, але й сприяють залученню та утриманню уваги користувачів.

2.5.5 Фонові завдання

Ще одним важливим компонентом додатку є фонові завдання, які виконуються паралельно до основних задач застосунку, але не блокують інтерфейс користувача, оновлення даних або запланованих завдань, що виконуються навіть тоді, коли додаток не використовується. Для виконання таких завдань використовується бібліотека react-native-background-fetch.

Вона дозволяє встановлювати інтервали, це дуже корисно для завдань, які потребують регулярної взаємодії з сервером або обробки даних в фоновому режимі. Також ця бібліотека може автоматично управляти енергоспоживанням та визначенням оптимального часу для виконання завдань.

Отже, бібліотека react-native-background-fetch необхідна аби забезпечити високий рівень ефективності, покращити досвід користувача неперервним доступом до навчального застосунку.

2.6. Взаємодії серверної та клієнтської частини

Розробка API (Application Programming Interface) – один з ключових аспектів для забезпечення взаємодії між сервером і клієнтською частиною. API можна описати як міст, який об’єднує функціональну реалізацію з інтерфейсом користувача.

На початку цієї частини роботи важливо чітко визначити специфікації API, а саме точки доступу (endpoints), формати даних, та методами HTTP, які будуть використовуватися. Це включає опис кожного запиту та відповіді, що очікується отримати. Для гарантії безпеки даних та функціональностей, API повинні включати механізми аутентифікації та авторизації. В цьому проєкті це відбувається за допомогою токенів для забезпечення доступу користувача до тої інформації, до якої він має право доступитися. Під час розробки точок доступу використовуються HTTP методи, де GET відповідає за отримання даних, POST для створення нових записів, PUT або PATCH для редагування існуючих записів та DELETE для їх видалення. Це допомагає створювати API інтуїтивно зрозумілим та забезпечувати дотримання стандартів.

Ще один важливий блок – це обробка помилок в API, це виражається

отриманням статусу запиту з номером та коротким описом, з певного загальноприйнятного списку статусів відповіді веб запитів. Це створено для коректної обробки помилок сервера, некоректних запитів та помилок в даних на вхід або на вихід.

Application Programming Interface – це критично важливий елемент під час розробки будь-якого додатку. Саме він забезпечує гармонійну взаємодію клієнтської та серверної частини, полегшує процес розробки і покращує досвід користувача.

2.5.6 Postman

Postman – це інструмент для тестування API, він є одним з найбільш популярним і зручним для перевірки точок доступу. Саме це програмне забезпечення було використано під час тестування ендпойнтів, оскільки він дозволяє відправляти запити прямо на сервер, підібравши потрібний HTTP метод (GET, POST, PUT, PATCH), написавши коректне посил ання і натиснувши кнопку відправити. Postman виводить відповідь, а також статус виконання запиту, наприклад 200 – Успіх. Також в нього є можливість передати тіло запиту, не лише як параметри запиту, а й типи даних у форматі JSON для прикладу. Як додатковий функціонал, Postman має вбудований генератор випадкових значень, достатньо просто налаштувати це прямо в тілі запиту. Такими даними можуть бути електронна пошта та ім’я користувача для тестування запиту реєстрації. Це значно пришвидшує розробку і економить багато сил і часу розробників. Ще один корисний функціонал – це налаштування заголовків (headers) запиту, куди наприклад примусово можна записати токен для входу, для тестування запитів, в яких потрібна авторизація користувача. Також важливо відмітити зручність налаштування кукі, структурування запитів для проєкту, їх можна розбити на папки і кожен запис окремо підписати для повної прозорості. Це зручно

при розробці в команді, оскільки проєктом в Postman можна поділитися, просто скопіювавши посилання.

Отже, Postman є необхідним для розробників інструментом, який економить для тестування ендпойнтів проєкта, завдяки автоматизації тестів, має зрозумілий інтерфейс та сильну документацію і спільноту.

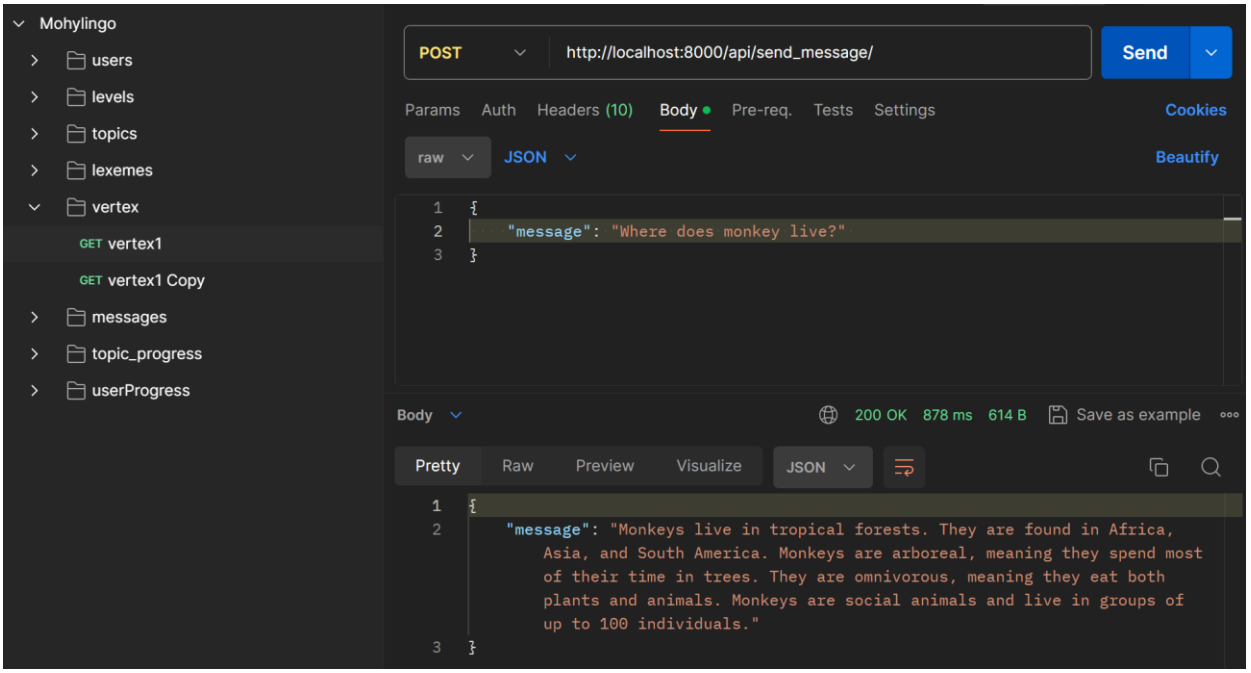

Рисунок 2.6.1.1 – Приклад побудови запиту в Postman

2.5.7 Ngrok

Оскільки метою цього проєкту було створення мобільного додатку, то запуск для тестування відбувався у декілька етапів:

- запуск backend серверу на ноутбуці, де зберігався проєкт;
- запуск frontend серверу також на ноутбуці;
- запуск застосунку за допомогою Expo Go на мобільному пристрої.

Оскільки мобільний застосунок був на іншому пристрої то тестування API через локальний хост, то було прийнято рішення використовувати Ngrok.

Ngrok – це інструмент, який дозволяє розробникам та іншим користувачам

експонувати локальні сервери через безпечні тунелі до глобального інтернету. Він допомагає облегшити тестування застосунків, що працюють на локальних машинах, без необхідності розгортання на віддалених серверах.

Ngrok не вимагає складних налаштувань і може бути запущений однією командою. Він використовує шифровані тунелі для забезпечення конфіденційності всіх переданих даних, що підсилює рівень безпеки даних. Ngrok працює на різних операційних системах, наприклад Windows, MacOS та Linux. Це середовище спрощує діагностику проблем під час розробки, оскільки має локальний веб-інтерфейс, який дозволяє переглядати історію трафіку, вхідні та вихідні запити, а також детальну інформацію про трафік.

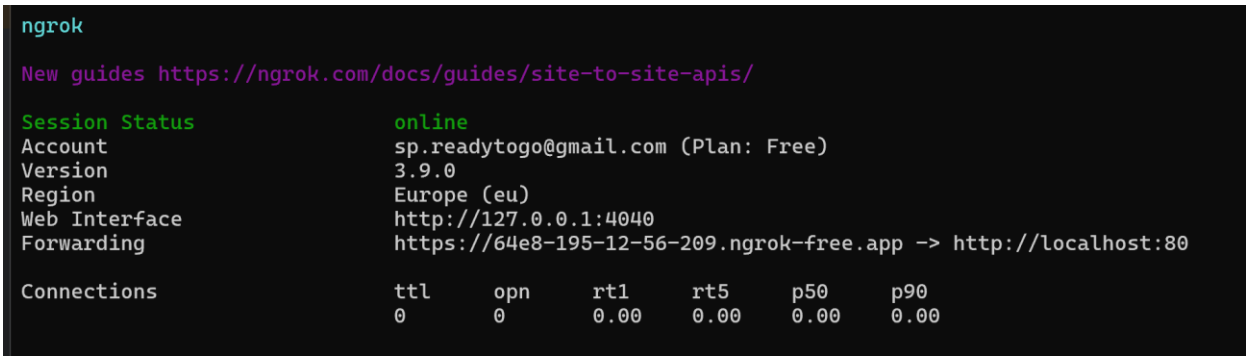

Рисунок 2.6.2.1 – Запущений Ngrok для localhost

Отже, Ngrok забезпечує надійний міст між локальною розробкою та зовнішнім тестуванням, роблячи процес розробки більш гнучким та доступним.

2.6.3 Мережева взаємодія

Наступний фундаментальний аспект розробки мобільного застосунку – це мережева взаємодія. Вона дозволяє додаткам комунікувати з серверами для обміну даними, обробки запитів і отримання оновлень.

Для управління асинхронними запитими в проєкті використовується React Query, розроблений TanStack. Він дозволяє ефективно керувати

закешовними даними, синхронізацією, оновленням та відновленням даних, цк значно спрощує процеси, пов’язані з взаємодією між клієнтом та сервером. React Query зручний, оскільки бібліотека підтримує синхронізацію стану даних між компонентами, що використовують однакові запити. Ще одна перевага – це автоматичне кешування відповіді від сервера, що зменшуючи кількість необхідних мережевих запитів. Також React Query дає компонентам можливість запитувати свіжі дані в фоновому режимі, що дозволяє користувачам бачити оновлену інформацію без перезавантаження сторінок або втручання в інтерфейс користувача. Бібліотека від TanStack надає готові хуки, такі як useQuery та useMutation, для керування станами запитів (завантаження, оновлення, помилка), що дозволяє легко інтегрувати логіку управління даними, економлячи час розробників на створення складних конструкцій. У випадку помилки запиту, бібліотека автоматично повторює запити, забезпечуючи високу надійність взаємодії.

Основним інструментом для HTTP запитів виступає Axios. У контексті Django - потужного Python веб-фреймворка для серверної розробки, Axios може виступати як ключовий компонент клієнтської частини для асинхронного звернення до сервера, оскільки він дозволяє клієнтській частині виконувати GET, POST, PUT, DELETE запити до цих API, що управляють даними.

Інтеграція Django та Axios відкривають перед розробниками широке вікно можливостей, ефективно управляти мережевими запитами на стороні клієнта. Цей інструмент забезпечує гнучкість, швидкість та масштабованість в розробці.

Отже, мережева взаємодія забезпечує безперервний потік даних між клієнтськими пристроями та серверами. При розробці навчального

застосунку були використані React Query та Axios бібліотеки, які допомагають у керуванні станом запитів, обробки помилок та синхронізації даних. Вони дозволяють підвищити стабільність роботи застосунку і залученість користувачів.

2.7. Штучний інтелект

Vertex AI – це набір інструментів та сервісів машинного навчання від Google. Він дозволяє розробникам легко побудувати, розгорнути та масштабувати моделі машинного навчання. Віртуальний співрозмовник для вивчення німецької мови використовує Vertex AI для генерації тексту через модель TextGenerationModel.from_pretrained, яка здатна автоматично генерувати відповіді на повідомлення користувачів у чаті. Розробка цього функціоналу додана аби підвищити інтерактивність та залученість користувачів, забезпечуючи більш розумний діалог з застосунком.

Vertex AI тут було поєднано з передовою моделлю ШІ, такою як Palm2. Ця модель штучного інтелекту може аналізувати вхідні повідомлення та генерувати відповіді, які здаються природними та інформативними. Для використання цієї текстової моделі було інтегровано API, яке дозволяє додатку взаємодіяти з моделлю машинного навчання. Це забезпечує безперервний потік даних між клієнтською частиною додатка та обчислювальними ресурсами в хмарі. Моделі, такі як Palm2, оптимізовані для високої продуктивності, що дозволяє їм швидко обробляти великі об'єми даних.

Використання Vertex AI та моделі штучного інтелекту Palm2 в розробці додатків значно підвищило можливості та функціональність застосунку, особливо в контексті обробки природної мови та автоматизованої взаємодії з користувачами. Тепер розмова з користувачем дозволяє імітувати справжній діалог, що довзоляє вдосконалювати їх досвід і забезпечувати

високу якість сервісу. Масштабованість, ефективність та висока продуктивність, які забезпечуються за допомогою Vertex AI та Palm2, забезпечують розробників стабільно якісним результатом.

2.8. Висновок до розділу 2

Розділ 2 детально описує ключові аспекти проектування та розробки мобільного застосунку для вивчення німецької мови, включаючи інтерфейс користувача, серверну частину, взаємодію з клієнтською частиною, мережеву взаємодію, а також інтеграцію з сучасними інструментами та технологіями штучного інтелекту.

Ретельний аналіз існуючих аналогів дозволив ідентифікувати сильні та слабкі сторони конкурентів. Це забезпечило цінні уроки та ідеї для покращення власного застосунку, включаючи вдосконалення функціональності та користувацького інтерфейсу.

Ми почали розглядати процес створення з розробки візуального прототипу. Розробка концептуального дизайну, створення макетів основних екранів та елементів навігації. Важливість чіткого UI/UX дизайну та його вплив на загальне сприйняття та зручність використання додатку підкреслюється через інструмент Figma.

Далі перейшли до аналізу створення серверної частини. Вибір технологічного стеку, включаючи такі мови програмування як Python (Django), та бази даних як PostgreSQL. Розробка архітектури сервера, API, інтеграція зі сторонніми сервісами, гарантування безпеки даних відіграють вирішальну роль у надійності та продуктивності додатку.

Після цього розглянули використання React Native для клієнтської частини, що дозволяє розробляти мобільні застосунки, що ефективно працюють на iOS та Android з одного кодової бази. Оптимізація продуктивності,

інтеграція з різними API та бібліотеками, та впровадження функціональності реального часу - все це підкреслює гнучкість та потужність React Native.

Наступним пунктом для аналізу стало використання API для ефективного обміну даними між бекендом і фронтендом. Розгляд безпеки, керування станом даних, використання таких інструментів як Axios та React Query підкреслює якісну та швидку мережеву взаємодію.

І звичайно ми розглянули застосування передових технологій, таких як Vertex AI та моделі Palm2, для підвищення інтерактивності та персоналізації застосунків. Інтеграція штучного інтелекту для автоматизації відповідей у чатах та покращення інтерфейсу з користувачами показує актуальність і сучасність даного проєкту.

Усі ці етапи та підпункти вказують на комплексний підхід до розробки сучасних мобільних застосунків, що забезпечують високу продуктивність, безпеку, користувацьку привабливість та інноваційність.

Висновки

Проект розробки віртуального співрозмовника для практики розмовних навичок німецької мови включає два ключові розділи: аналіз існуючих аналогів та детальне проектування та розробка застосунку. Ці розділи разом формують комплексний підхід до створення застосунку, починаючи з вивчення існуючих рішень на ринку до практичної реалізації ідей включаючи інтерфейс користувача, клієнтську частину на React Native, серверну частину на Django, інтеграцію з Vertex AI та Palm2 для штучного інтелекту. Ключові аспекти, такі як чіткий UI/UX, висока продуктивність, безпечна мережева взаємодія та передові технології ШІ, гарантують забезпечення високоякісного інтерактивного досвіду користувачів, підкреслюючи потенціал проекту стати ефективним інструментом для навчання мови та вирішення викликів у підвищенні розмовних навичок.

Список використаних джерел

1. 11 best apps to learn German [Електронний ресурс] // iamexpat – 2024. – Режим доступу до ресурсу: https://www.iamexpat.de/education/education-news/11-best-apps-learn-german.
2. Top 10 Examples of React Native Apps in 2024 [Електронний ресурс]// brainhub by – 2024 – Режим доступу до ресурсу: https://brainhub.eu/library/react-native-apps
3. React Native Hooks & How To Use useState and useEffect [Електронний ресурс] / Gilshaan Jabbar– 2020 – Режим доступу до ресурсу: https://gilshaan.medium.com/react-native-hooks-how-to-use-usestate-and-useeffect-3a10fd3e760c
4. Getting started with Django [Електронний ресурс] / djangoproject by Django Software Foundation – 2014 – Режим доступу до ресурсу: https://www.djangoproject.com/start/
5. A Comprehensive Guide to Django Views: Understanding Types and Use Cases [Електронний ресурс] //Satya Repala – 2023 – Режим доступу до ресурсу: https://medium.com/@satyarepala/a-comprehensive-guide-to-django-views-understanding
6. Setting up the development environment/ reactnative– 2024 – Режим доступу до ресурсу: https://reactnative.dev/docs/environment-setup
7. Use libraries [Електронний ресурс] // Expo – 2019 – Режим доступу до ресурсу: https://docs.expo.dev/workflow/using-libraries/
8. What is ngrok? How to use ngrok tunneling? [Електронний ресурс]. / pubnub – 2021 - Режим доступу до ресурсу: https://www.pubnub.com/guides/what-is-ngrok/
9. React Query: документація [Електронний ресурс]. Режим доступу до ресурсу: https://tanstack.com/query/latest/docs/framework/react/overview
10. Integrate ReactJS and Django using Django-Rest Framework and Axios [Електронний ресурс] / 2023 – Режим доступу до ресурсу: https://medium.com/@umarsunny2011/integrate-reactjs-and-django-using-django-rest-framework-and-axios-166223e0860b
11. Vertex AI PaLM API: Qwik Start [Електронний ресурс] // Google – 2023 – Режим доступу до ресурсу: https://www.cloudskillsboost.google/focuses/71944?parent=catalog
12. PostgreSQL Tutorial [Електронний ресурс]. // w3schools– 2020 – Режим доступу до ресурсу: https://www.w3schools.com/postgresql/index.php
13. Get started with the Gemini API: Python // Google – 2023 – Режим доступу до ресурсу: https://ai.google.dev/gemini-api/docs/get-started/python 319-46562-3_29
14. Model API for Gemini in Vertex AI [Електронний ресурс]. // Google – 2023 – Режим доступу до ресурсу: https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/gemini
15. Guide: What is Google Gemini API and How to Use it? //
16. Steven Ang Cheong Seng – 2019 – Режим доступу до ресурсу: https://apidog.com/blog/google-gemini-api/