Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

СТВОРЕННЯ ВІРТУАЛЬНОГО РЕПЕТИТОРА ДЛЯ ПІДГОТОВКИ ДО МОВНИХ ІСПИТІВ

Текстова частина курсової роботи

за спеціальністю 122 «Комп'ютерні науки»

Керівник курсової роботи

доц. Тригуб О.С.

_____

(підпис)

«___»_____2024 р.

Виконав студент Письменний А.К.

_____

(підпис)

«___»_____2024 р.

Київ 2024

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ

Кафедра інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри інформатики

доцент, к.н

_____ Гороховський С.С

«___»_____2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту 3 курсу факультету інформатики

Письменному Антону Костянтиновичу

ТЕМА: Створення віртуального репетитора для підготовки до мовних іспитів

Зміст ТЧ до курсової роботи:

Зміст

Анотація

Вступ

1 Аналіз предметної області

2 Огляд обраних технологій

3 Проєктування додатку

4 Розробка додатку

Дата видачі «___»_____ 2024 р. Керівник _____

Завдання отримав _____

Тема: Створення віртуального репетитора для підготовки до мовних іспитів

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання завдання	Примітка
1	Ознайомлення з темою й завданням	24.10.2023	
2	Пошук та структурування теоретичної інформації на тему	24.02.2024	
3	Створення програмного продукту	10.04.2024	
4	Систематизація напрацювань, написання тестової частини роботи	10.5.2024	

Письменний А.К. _____

Тригуб О.С. _____

«___» _____ 2024 р.

Зміст

Анотація

Метою курсової роботи було створення віртуального тренера-репетитора в формі застосунку, що б готував користувачів до іспитів з англійської мови базового рівня (A1, A2), надаючи достатню різноманітність тестів і тем з можливістю перегляду попередньо пройдених матеріалів для роботи над помилками, а також підтримуючи вмотивованість користувачів за рахунок ґрунтовної статистики відвідуваності та успішності, регулярних нагадувань. На меті також було передбачення розширення даного функціоналу в майбутньому (складніші тести, більше різноманіття тем, додаткові типи завдань тощо).

В роботі проаналізовані наявні на ринку навчальні застосунки, їхні переваги й недоліки, сучасний інструментарій для створення Frontend і Backend складових таких проєктів, а також перспективи використання штучного інтелекту, а конкретніше – великих мовних моделей, для генерування й перевірки тестових завдань для досягнення їхньої максимальної різноманітності. В роботі описана архітектура та принцип роботи створеного застосунку.

Ключові слова: штучний інтелект, мобільний застосунок, клієнт-серверна взаємодія, велика мовна модель, комп’ютеризоване навчання.

Вступ

В сучасному світі англійська мова залишається основним засобом міжнародної комунікації. Результати сучасних опитувань показують, що в Україні проблема володіння населення цією мовою є актуальною: лише 1 відсоток опитаних вважає, що може вільно спілкуватися англійською, 7.5 відсотків – не володіють мовою вільно, хоч і можуть читати, писати й спілкуватися, а 44 відсотки зовсім не знають англійської мови [1]. Враховуючи високу ймовірність подальшої економічної інтеграції нашої країни в загальноєвропейські процеси, очевидна необхідність покращення цієї статистики.

Процес такого покращення вже відбувається, зокрема за рахунок середньої та вищої освіти. Так методичними рекомендаціями Міністерства освіти і науки України [2] закріплено мету забезпечити завершене формування в студентів потреби вивчення англійської мови з оволодінням нею як засобом спілкування, пізнання, самореалізації та соціальної адаптації. Природнім є припущення, що належне опанування англійської мови в закладах середньої освіти могло б полегшити таке оволодіння. Можливість такого припущення також підкріплюється результатами реєстрації осіб для проходження національного мультитесту (НМТ) 2024 року, де 95 тисяч осіб з 228 тисяч зареєстрованих будуть складати іспити з іноземних мов [3], серед яких наразі найпопулярнішою є англійська (51 відсоток) [4].

Для доповнення контексту варто додати також такі факти: 51 відсоток дітей вивчає англійську мову виключно в навчальному закладі, 22 відсотки її не вивчають, а 27 відсотків окрім навчального закладу також користуються послугами репетиторів [4].

Безперечно, допомогти з опануванням англійської мови в контексті потреб українських школярів могли б відповідні програмні застосунки, проте наявні на

ринку рішення скоріше допомагають з загальним опануванням англійської, а не з безпосередньо підготовкою до іспитів чи уроків на конкретні теми. До того ж, постає проблема обмеженої кількості наявних тестів, а також особливості вибору тем для деяких контрольних заходів. Уникнення цієї проблеми зазвичай полягає в використанні одразу багатьох ресурсів та постійному пошуку нових.

Враховуючи вищесказане, за мету даної роботи поставлено створення застосунку, що міг би дозволити безкоштовно покращувати рівень володіння англійською мовою користувачів, зокрема допомагаючи таким чином у підготовці до тестів чи опануванні шкільного матеріалу, виступаючи допоміжним засобом, а також забезпечити максимальну гнучкість й до певної міри невичерпність набору тестових питань та тем тестів, що надаються користувачу. До того ж, такий додаток має підготувати користувача до формату контрольних заходів, які на нього чекають. Завданнями цієї роботи є огляд доступних технологій для створення даного застосунку (клієнтської та серверної частини), аналіз наявних програмних рішень зі схожим функціоналом, дослідження перспективи інтеграції засобів штучного інтелекту (зокрема – великих мовних моделей) для досягнення згаданої невичерпності тестових завдань для користувача.

Робота складається з чотирьох розділів:

Перший розділ присвячений аналізу предметної області.

Другий розділ проводить огляд обраного інструментарію та технологій.

Третій розділ присвячений архітектурі додатку та його проєктуванню.

Четвертий розділ розглядає реалізацію даної архітектури в програмному коді.

1 Аналіз предметної області

1.1 Огляд подібних застосунків

На ринку наявна велика кількість застосунків, ніша яких перетинається з тою, яку б зайняв створюваний додаток. Для аналізу було обрано найпопулярніші рішення з платформи Google Play: “English Grammar Test”, “Exam Lift: English Practice”, “British Council EnglishScore”, “Duolingo” та інші.

Частина цих застосунків (як-от “Exam Lift: English Practice” чи “Mangoosh IELTS prep”) розроблена для підготовки користувача до конкретного екзамену. Розробляючи план з фіксованими питаннями та темами, яким він має рухатися, застосунок нагадує користувачу про необхідність практики.

Інші додатки (як-от “English Grammar Test”) пропонують більший контроль за рівнями та темами навчання, не нагадуючи користувачам про проходження тестів. Вибір же тестів і тем залишається обмеженим, можливості адаптувати його під потреби користувача немає, що виявилося основним недоліком рішень, представлених на ринку наразі.

“Duolingo” (найпопулярніший з розглянутих застосунків) незважаючи на відсутність орієнтованості на підготовку до контрольних заходів, пропонує механіку заохочення, яку доречно було б використати: користувачу демонструється кількість днів, яку він без пропусків відвідує додаток. До того ж, показовою є мультиплатформеність Duolingo, а також клієнт-серверна природа рішення, що дозволяє користувачам продовжувати практикуватися незалежно від їхнього місцезнаходження чи доступних девайсів.

Розглянуті застосунки в своїй більшості не показують користувачу час, витрачений на виконання тестів, а помилки демонструються безпосередньо на етапі проходження з детальним поясненням причини, що віддалює формат від того, з яким можна стикнутися під час контрольних заходів. Розглянуті рішення не

перебачають перегляд зданих робіт, тільки їхнє перепроходження. Доцільною була б опція ретельнішого аналізу та перегляду зроблених помилок.

1.2 Концепція

Враховуючи переваги й недоліки проаналізованих застосунків, можуть бути визначені проблеми, які мають бути адресовані в створюваному в роботі додатку:

а) Додаток має бути кросплатформенним, з системою акаунтів та усіма супутніми функціями для можливості безперешкодного перемикання між доступними користувачу на даний момент пристроями.

б) Під час вибору тесту, додаток має надавати користувачу можливість рівень тесту, а також надати можливість визначити власну тему, або ж обрати зі списку рекомендованих, після чого користувачу має бути наданий тест, що не буде або мінімально буде повторювати попередні.

в) Додаток має імітувати обстановку, в якій імовірніше за все опиниться користувач під час складання тесту: користувач має бачити час, який він витрачає під час проходження тесту, відповіді є закритими для нього до моменту завершення, візуальне оформлення тестів має передбачати не стільки естетику, скільки презентацію інформації в стандартному для тестувань вигляді.

г) Додаток має надавати можливість переглядати всі попередньо пройдені тести користувача для аналізу помилок. Має бути передбачена значна кількість таких тестів.

1.3 Прототипування

З урахуванням попередніх підрозділів даної роботи, можна визначити такі складові клієнтської частини додатку:

а) Екран аутентифікації (входу в акаунт, реєстрації, зміни паролю)

б) Домашній екран (загальна статистика, навігація до інших екранів)

в) Меню створення тестів

г) Екран тестування

д) Каталог пройдених тестів (з можливістю фільтрування)

Відповідно, серверна частина додатку має забезпечувати:

а) доступ до бази даних з інформацією про користувачів, про тести, які вони проходили, а також про підтримувані рівні й рекомендовані для проходження теми.

б) генерацію й надання згенерованих тестів кінцевому користувачу.

в) періодичне надходження сповіщень-нагадувань користувачу.

г) розсилку електронних листів для відновлення паролів

ґ) передачу форми відновлення паролю кінцевому користувачу.

1.4 Генерація тестів

Оскільки основною перевагою застосунку має бути до певної міри подолання проблеми вичерпності тестів, варто розглянути системи, що могли б ці тести готувати. Можливі декілька шляхів вирішення проблеми: збір власної бази тестів або використання таких баз, наявних у відкритих джерелах (фактично ідучи шляхом своїх попередників), синтез власної бази тестів (ще більш кропіткий та затратний процес) або спроба використання засобів штучного інтелекту (як-от великих мовних моделей) для динамічного створення тестових питань. Було обрано останній шлях подолання проблеми через його унікальність, що може приховувати нерозкритий потенціал і надати перевагу над рішеннями, які використали перші два згадані підходи.

Великі мовні моделі є підвидом фундаментальних моделей, в основі яких лежить слабокероване навчання на максимально різноманітних неструктурованих даних, що дозволяє досягти виняткової універсальності в порівнянні з іншими

моделями штучного інтелекту, які зазвичай передбачають навчання на певному тематичному наборі даних для вирішення конкретної задачі [5][6]. В основі таких моделей лежить трансформерна архітектура: технологія, якій по праву приписують стрімкий розвиток та інтеграцію штучного інтелекту в повсякденне життя. Завдяки механізму «уваги», ця архітектура дозволяє встановлювати контекст токена, його зв'язки з іншими токенами в вхідних даних. В контексті великих мовних моделей, такими токенами здебільшого виступають слова[7].

Основною задачею великих мовних моделей є генерація тексту, що полягає в ітеративному передбачені наступного токену. Важливим є також елемент невизначеності, що виникає під час вибору поточного токену. На міру цієї невизначеності впливають такі параметри, як topK (вибір одного з k найбільш популярних токенів), topP (вибір одного з такої групи найбільш популярних токенів, сума ймовірностей яких дорівнює p) та поняття температури (нормалізації результату роботи функції softmax). Нижче наведено математичне підґрунтя та демонстрацію ефекту температури.

На рисунку 1.1 бачимо формулу обрахунку ймовірності того, що наступним токеном буде токен з індексом i (функція softmax). За застосування цієї формули параметр температури, очевидно, не впливає на результат [8].

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_j^k e^{z_j}} \quad (1.1)$$

де z_i – значення на вихідному шарі на останній ітерації роботи трансформера

Нижче наведена аналогічна формула з використанням параметру температури [8].

$$\sigma(z)_i = \frac{e^{z_i/T}}{\sum_j^k e^{z_i/T}} \quad (1.2)$$

де T – температура моделі

Вплив температури на результат роботи моделі можна побачити нижче

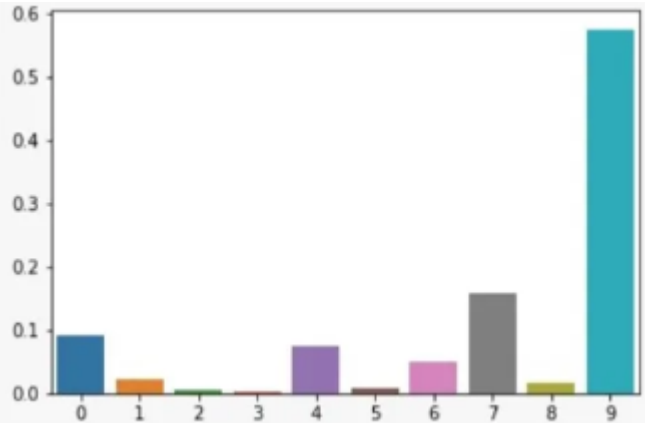

Рисунок 1.4 Діаграма розподілу ймовірностей токенів без температури [9]

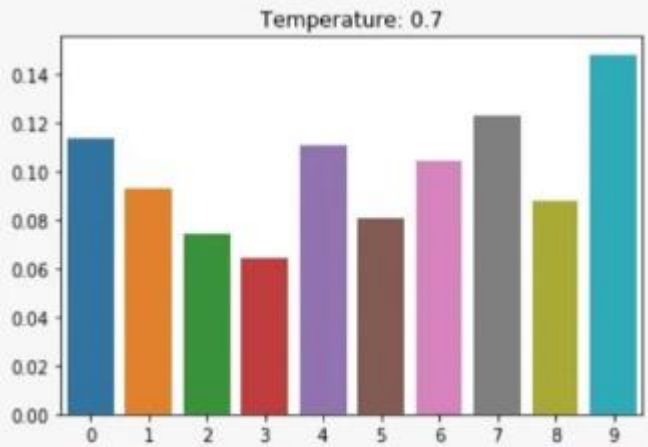

Рисунок 1.5 Діаграма розподілу ймовірностей токенів з температурою T=0.7 [9]

Для прикладу, за замовчуванням значення згаданих параметрів для мовної моделі PaLM такі: topK має значення 40 (тобто верхнє обмеження на кількість токенів, що мають шанс бути наступними – 40), topP має значення 0.95 (тобто обирається токен з такої кількості найбільш імовірних, що сумарна ймовірність рівна не більше ніж 0.95) [10]. Інші моделі також пропонують конфігурацію цих параметрів. Отже, використання великих мовних моделей дійсно може забезпечити генерування унікальних тестів, навіть в умовах ідентичності вхідних даних.

2 Огляд обраних технологій

2.1 Flutter: огляд і причини вибору

2.1.1 Коротка характеристика

Вимога кросплатформенності обмежує вибір фреймворків та мов для створення клієнтської частини майбутнього додатку. Використання двох окремих технологій для браузерної та мобільної версій застосунку збільшило б обсяги роботи, зменшило б можливості перевикористання надбань, хоч і могло б надати часткові переваги в адаптації під особливості кожної платформи. Було зроблено вибір на користь технологій, що пропонують збірку застосунків для різних типів девайсів на базі одного сирцевого коду.

Під час вибору такої технології, одним з основних критеріїв було обрано популярність рішення. Це може бути аргументоване очікуваною активністю форумів, швидкодією в контексті виправлення багів тощо. Згідно зі статистикою трендів Google, а також відкритими даними платформи StackOverflow, наразі найпопулярнішим рішенням на ринку є фреймворк Flutter [11][12].

Flutter – фреймворк для створення користувацького інтерфейсу, створений за підртимки компанії Google. Його розробка почалася в 2015 році [13], вперше ж пересічні розробники отримали доступ до технології в 2017. Кінцевий користувач фреймворку (розробник) використовує мову Dart для створення кросплатформенних додатків, описуючи з її допомогою як логіку, так і візуальну складову застосунку.

З використанням фреймворку Flutter було створено чимало відомих сучасних додатків, як-от Google Pay, Google Earth, Google Classroom, продукти таких гігантів ринку IT як Alibaba й Tencent [14].

2.1.2 Переваги й недоліки Flutter

З урахуванням думок програмістів, а також власного досвіду з фреймворком, можу виділити, окрім уже згаданих кросплатформенності та популярності, такі його переваги [15]:

а) Швидкий процес розробки – завдяки таким функціям, як швидкий скид (hot restart) і швидке перезавантаження (hot reload), більшість змін відображаються в додатку в момент їх збереження на пристрої, а навіть якщо ці зміни є більш фундаментальними, додаток можна миттєво (0.3-1 с) перезапустити

б) Швидкодія кінцевого застосунку – завдяки компіляції в рідний код, застосунки працюють зі швидкістю, що відповідає розробленим прив’язаними до платформи засобами

в) Краса за замовчуванням – віджети, пропоновані фреймворком, в своїй стандартній конфігурації мають привабливий дизайн, що вимагає хіба що незначних правок. До того ж, неабияким є різноманіття вбудованих віджетів: більшість задач представлення даних мають по декілька можливих реалізацій конкретними, спеціально створеними під ці задачі, компонентами інтерфейсу

г) Доступ до рідних функцій через програмний інтерфейс фреймворку (механізм буде розглянутий в наступному підрозділі)

ґ) Єдина мова опису складових логіки й представлення, що полегшує їхній взаємозв’язок (про інші переваги мови буде згадано в наступних підрозділах)

Користувачі згадують такі недоліки Flutter [15]:

а) Більший розмір додатків у порівнянні з тими, що були розроблені з використанням рідних засобів розробки (проблема будь-якого фреймворку)

б) Довга початкова збірка додатку

в) Необхідність вивчення нової мови для використання, що певною мірою підвищує поріг входження

г) менша кількість доступних бібліотек у порівнянні з конкурентами (Ionic, ReactNative тощо)

Враховуючи власний досвід, отриманий під час розробки з використанням фреймворку, можу сказати, що ні разу не опинився в ситуації відсутності потрібної для реалізації функціоналу бібліотеки, а також що завдяки своїй синтаксичній і семантичній схожості з іншими C-подібними мовами програмування, вивчення Dart складно вважати значним недоліком. Таким чином, фреймворк має більше переваг, ніж він має недоліків, що робить його оптимальним інструментом для розробки клієнстської частини застосунку.

2.1.3 Основні компоненти додатку на Flutter

На діаграмі представлені основні компоненти типового застосунку на базі фреймворку Flutter.

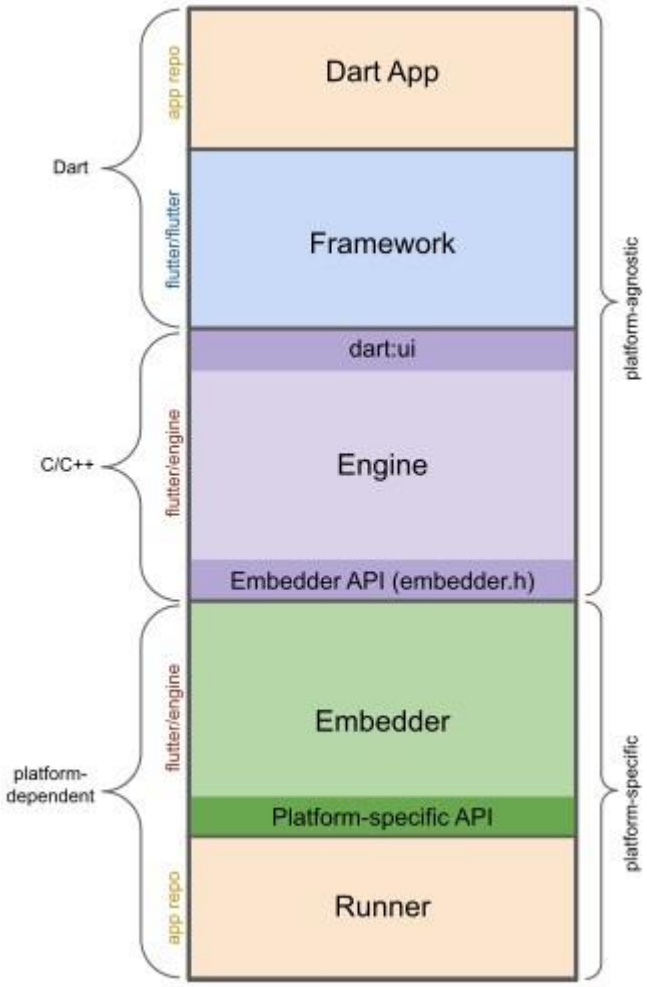

Рисунок 2.1 Архітектура додатків на Flutter [16]

Рівень додатку (Dart App) написаний безпосередньо розробником застосунку, він утилізує базові класи (в тому числі – класи базових віджетів) і функції, описані на рівні фреймворку (Framework). Однією з важливих функцій останнього є побудова дерева віджетів.

Проміжним етапом між унікальною для платформи логікою й вищими рівнями є рушій (Engine), написаний з використанням C і C++. Саме на цьому рівні дерево віджетів конвертується в растрову графіку, а також визначається низькорівнева реалізація частини компонентів Flutter (як-от текстова розмітка)

Рушій має доступ до вбудовувача (Embedder) через програмний інтерфейс (API), унікальний для кожної системи.

Вбудовувач визначає реалізацію багатьох функцій, що відрізняється залежно від платформи. Саме вбудовувач відповідає за координацію роботи пристроїв вводу й виводу, механізми збереження даних під час роботи додатку тощо. Також в його задачі входить координація механізмів збірки та запуску застосунків на кожній платформі.

2.1.4 Мова програмування Dart

Мова програмування Dart була створенна в 2013 році представниками компанії Google: Ларсом Баком і Каспером Лундом. В основі мови лежить об’єктно-орієнтована парадигма, якої Dart строго притримується: всі значення, які можуть набувати змінні (в тому числі – null), є об’єктами, а всі об’єкти, в свою чергу, є екземплярами класів.

Засновники мови приділили значну увагу перспективі її використання в контексті веб-технологій, через що одним із важливих компонентів набору інструментів розробки (SDK) є JavaScript компілятор [14]. Проте мова лише до певної міри схожа на JavaScript: однією з її суттєвих переваг є підтримка динамічної статичної, строгої й нестрогої типізації. Така гнучкість забезпечує належний рівень контроля, який розробник може використати за потреби. Корисним також є механізм null-безпеки (null-safety), який дозволяє контролювати не тільки тип змінної, а й можливість її невизначеності. Перечисленні особливості дозволяють назвати Dart більш структурованою мовою, ніж JavaScript.

2.2 Огляд Express.js і причини вибору

Під час вибору технології для створення серверної частини застосунку, в першу чергу розглядався потенціал до розширення функціоналу й експериментів, адже навіть маючи початковий план і розроблений концепт, розробники вимушені коригувати його, враховуючи нові ідеї, проблеми, що виникли на етапі імплементації попередньої версії тощо. До того ж, в пріоритеті стояла популярність технології з уже згаданих в попередніх розділах причин. Також враховувалася необхідність надавати користувачу форми зміни паролю та бажання мінімізувати кількість окремих додатків за рахунок використання в певних випадках технології server-side rendering (за якої сервером на запити відправляються готові сторінки, а не просто дані). Враховуючи вищесказане, було прийняте рішення використати фреймворк Express.js, що є надбудовою над середовищем Node.js.

Створений у 2010 році (за рік після появи Node.js) і будучи найпопулярнішим з усіх Node.js серверних фреймворків, Express.js доповнює подієорієнтований (event-based) підхід до управління веб-процесами простою маршутизацією, легкою побудовою ланцюгів обробки запитів і помилок та взаємозв’язків цих ланцюгів з обробниками маршрутів, а також забезпечує легке підключення шаблонних рушіїв як-от Twig, Pug та EJS, що могло б допомогти в нашій ситуації, коли не хотілося б ускладнювати механізм зміни паролю створенням додаткового окремого клієнтського застосунку [17][18]. Привабливим також є розмір кінцевого серверного застосунку: Express.js забезпечує повноцінне функціонування за відносно невеликого обсягу вихідного коду.

2.3 Огляд MongoDB. Причини вибору технології

Під час вибору технології керування базою даних, в першу чергу враховувалася гнучкість до позапланових змін, очікуваних в ході розробки застосунку. До того ж, обрана імплементація бази даних має бути достатньо захищеною й забезпечувати швидку клієнт-серверну взаємодію.

Ці вимоги повністю задовольняє MongoDB - документоорієнтована NoSQL система керування базами даних, що зберігає дані в форматі бінаризованих JSON-об’єктів (BSON)[19]. Безперечно, це пришвидшує операції з базою даних, особливо враховуючи вибір Express.js як серверного фреймворку, адже такий формат нівелює потребу в використанні шаблону DTO (класів-моделей сутностей з бази даних).

Колекції (альтернатива таблицям в SQL базах даних) не мають чітко визначеної структури, натомість клієнти (в нашому випадку – сервер на Express.js) визначають структуру записів, з якими вони бажають взаємодіяти. Це допомагає боротися з постійними міграціями після внесення змін у структуру таблиць бази даних.

Взаємодія з клієнтами відбувається через протокол рівня представлення MongoDB Wire[20], запити й відповіді якого переносить протокол транспортного рівня TCP. Власний специфічний протокол комунікації, а також власні бібліотеки взаємодії з базою даних (альтернатива ручним з’єднанням) роблять MongoDB за певної конфігурації достатньо захищеною.

Ще однією перевагою MongoDB є можливість використання MongoDB Atlas для розміщення бази даних у хмарі. Це допомогло б з майбутнім розширенням бази через ріст кількості й потреб користувачів (з урахуванням того, що в базі зберігатимуться всі тести клієнта, така опція нам знадобиться). До того ж, це підвищить стабільність роботи, адже за рахунок резервного копіювання, навіть в

разі відмови деяких серверів, доступ до даних не буде надовго втрачений [21]. В ході цієї роботи етап налаштування MongoDB Atlas не розглядається, проте наявність такої опції в майбутньому все одно є перевагою й причиною обрати MongoDB.

2.4 Вибір великої мовної моделі

Одним із ключових елементів застосунку є модуль взаємодії з мовною моделлю. Двома основними критеріями вибору самої моделі стали доступність (реальність достатньо довгих тестувань без суттєвих матеріальних витрат) і, безперечно, ефективність. В разі потрапляння моделі в перший критерій, проводилося її тестування. Через значні матеріальні вимоги було прийняте рішення не використовувати моделі OpenAI, незважаючи на їхню значну й перевірену ефективність. Були розглянуті безкоштовні моделі на платформі HuggingFace, як-от Flan-T5, T5, Gemma різних розмірів. Жодна з моделей переліку не показала задовільних результатів на тестових запитах. Такий результат можна було назвати очікуваним: кількість параметрів найбільшої моделі Gemma – 7 мільярдів (звідки й назва 7b), в той час як GPT-4, згідно з припущеннями, має близько 1.5 трильйонів параметрів [22][23].

В результаті було обрано модель Gemini 1.0 Pro. Дані щодо її точних розмірів залишаються спекулятивними, як і в випадку GPT-4. Згідно з матеріалами Google, Gemini не поступається, і навіть частково перевершує продуктивність GPT-4 [24]. Правдивість цієї інформації складно перевірити. Отримавши власний досвід з меншими й слабшими версіями великих мовних моделей від двох компаній (OpenAI ChatGPT-3 і Google Gemini 1.0), можу відмітити більшу кількість галюцинацій (відповідей, що протирічать або спотворюють факти) в останньої. Тим не менш, враховуючи доступність Gemini, статус флагмана компанії Google, а

також суттєво успішнішу генерацію тестів за компактні моделі з платформи HuggingFace, було прийняте рішення зупинитися саме на ній.

Через значні розміри обраної великої мовної моделі, було прийняте рішення проти розміщення її на сервері. Натомість використовується Vertex AI: платформа-складова Google Cloud, що дозволяє тренувати, модифікувати й використовувати віддалено моделі штучного інтелекту компанії Google [25]. Vertex AI пропонує просту й зрозумілу бібліотеку для Node.js, яка дозволяє швидко й безперешкодно налаштувати функціонал на боці сервера.

2.5 Вибір системи сповіщень

Деякі бібліотеки Flutter (як-от FlutterLocalNotifications) пропонують функціонал для створення сповіщень безпосередньо на девайсі з застосунком, проте такий підхід може викликати певні проблеми в разі накладання операційною системою обмежень на роботу додатку в фоновому режимі (що можна спостерігати, зокрема, на пристроях популярних в Україні Xiaomi, Huawei тощо)[26]. Відповідно, було прийняте рішення розмістити логіку надсилання сповіщень на сервері.

Для реалізації механізму відправки нагадувань було обрано платформу Firebase компанії Google, головними перевагами якої стали популярність (причини врахування цього критерію неодноразово згадувалися в роботі), наявність безкоштовного плану, а також легка інтеграція з серверним застосунком за рахунок бібліотеки для Node.js.

3 Проєктування додатку

3.1 Особливості архітектури додатків Flutter і їхній вплив на дизайн клієнтської частини

3.1.1 Шаблон проєктування Model-View-Controller

Складно уявити сучасний клієнтський застосунок, який би не намагався розділити відповідальність за функціонування додатку за певними принципами. Одним із стандартних підходів такого розподілу є шаблон Model-View-Controller (MVC).

Проглядаються три складові шаблону:

а) Модель – представлення структури даних.

б) Вигляд – представлення інтерфейсу користувача

в) Контролер – відповідає за обробку вводу користувача й ініціалізацію змін збережених даних.

В залежності від імплементації, вигляд може оновлюватися за рахунок змін моделі або ж за рахунок чіткої команди від контролера. [27]

3.1.2 Представлення MVC у додатках на Flutter

На архітектуру додатку, написаного з використанням фреймворку Flutter, впливають особливості останнього. Зокрема, унікальні для нього класи-обгортки Provider та ChangeNotifierProvider дозволяють відкрити більше можливостей для тестування віджетів, управління їхнім станом в залежності від стану даних тощо за рахунок поєднання й удосконалення з урахуванням специфіки Flutter таких шаблонів програмування, як Singleton і Publisher. Доступ до об’єктів класів, огорнутих класом Provider (далі – провайдери), з використанням контексту віджета (інформації про його положення в дереві віджетів [28]), в свою чергу, дозволяє

слідкувати за змінами стану цих об’єктів, що певною мірою реалізовує шаблон Listener. В поєднанні, ці два механізми суттєво спрощують взаємозв’язок між даними й представленням.

Легко бачити, що провайдери добре виконують роль контролерів у моделі MVC. Для подальшого вдосконалення такого представлення, можна додати також класи-сервіси, що б відповідали за клієнт-серверну взаємодію, або ж інші способи отримання даних і їхньої зміни, таким чином звужуючи відповідальність провайдерів до збереження стану моделей, їх надання засобам представлення (віджетам) і виклику потрібного сервісу в разі необхідності отримання нових або обробки старих даних.

Іноді виникає потреба в операціях, що взаємодіють з зовнішнім середовищем (з сервером чи сховищем додатку), але не з тими складовими поточного стану додатку, які впливають на дані безпосередньо. Такі класи, для прикладу, можуть мати локальні службові змінні, які більше ніде й ніяк не використовуються. Для цієї ролі відведено іншу категорію класів, які було названо менеджерами.

В деяких випадках, використання Provider виглядає недоречним, і перевага віддається статичним методам через потенційну необхідність доступу поза віджетами, а також через вищу продуктивність. Прикладами застосування таких статичних методів є валідація форм, приведення дати до певного формату для демонстрації користувачу, демонстрація сповіщень тощо.

Незважаючи на специфіку архітектури, шаблон проєктування Model-View-Controller в ній все одно можна спостерігати. Збереження й взаємодія з даними відбувається з використанням класів-моделей, за отримання нових відповідають сервіси, перетворення даних (часто в цілях наочнішого представлення)

виконують утиліти, доступ до даних на рівні представлення забезпечується провайдерами.

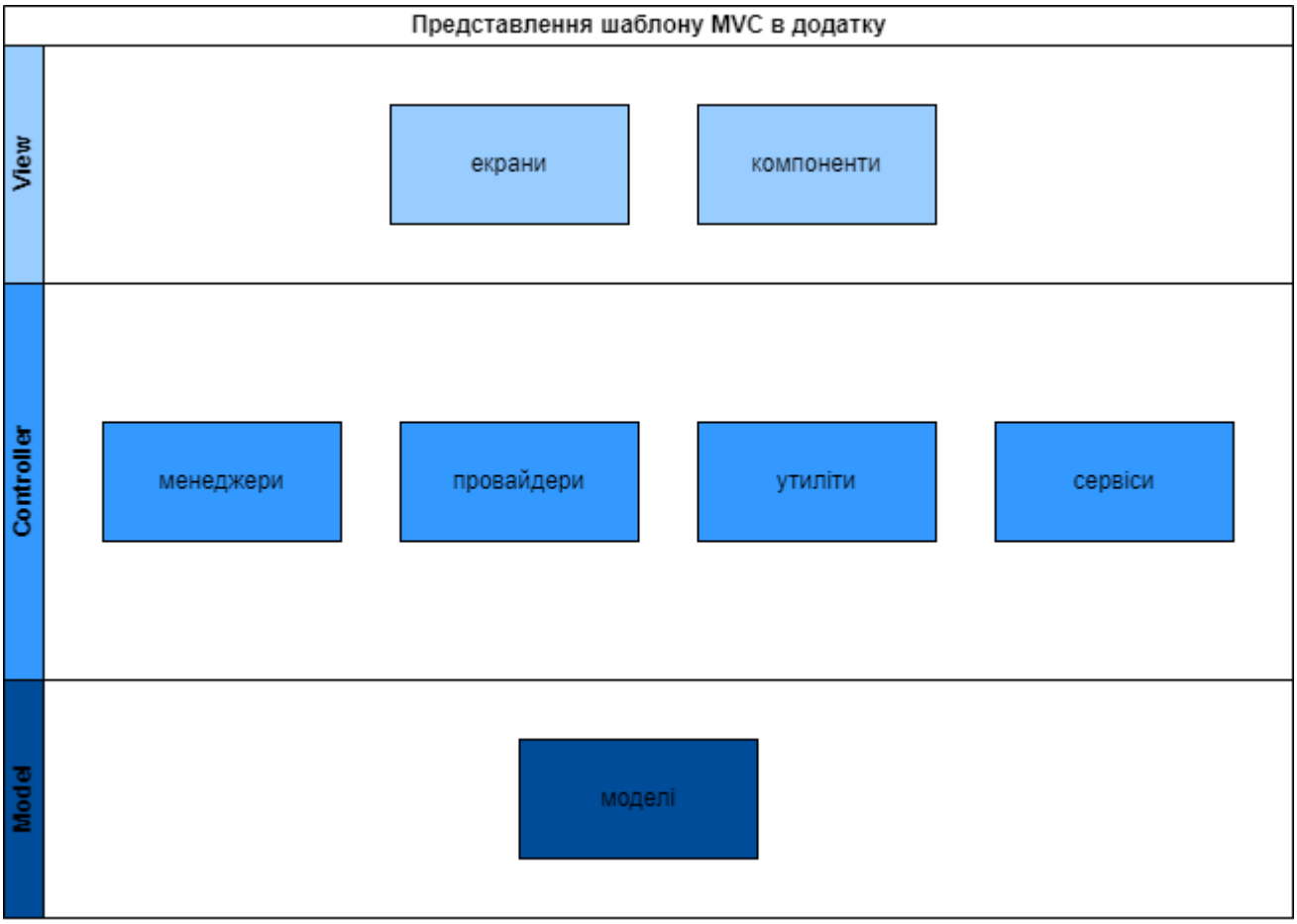

Рисунок 3.1 Представлення рис шаблону Model-View-Controller у додатку

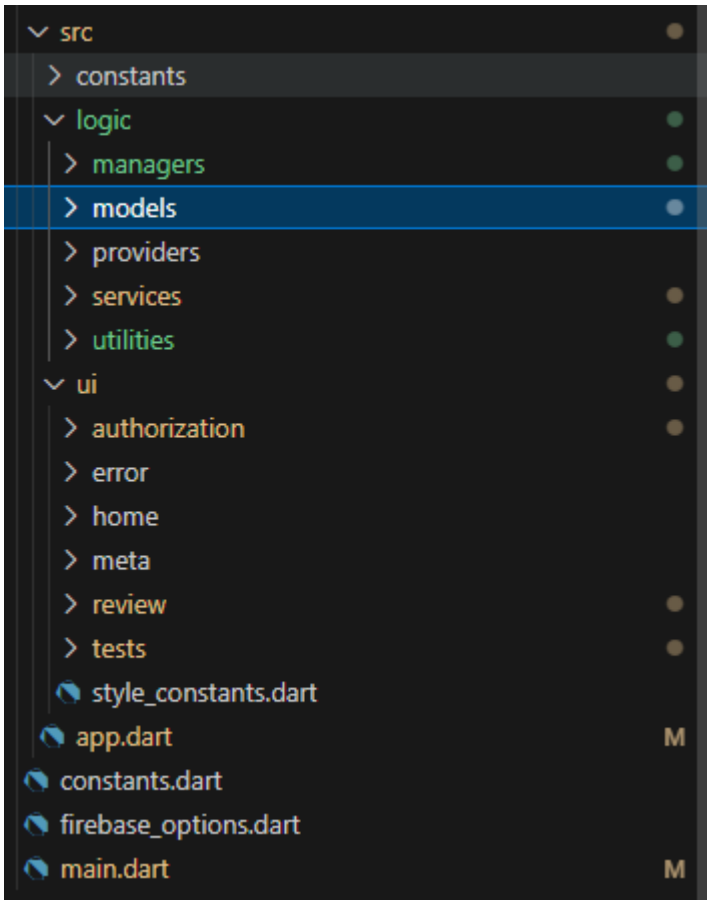

Рисунок 3.2 Відповідна організація коду додатку

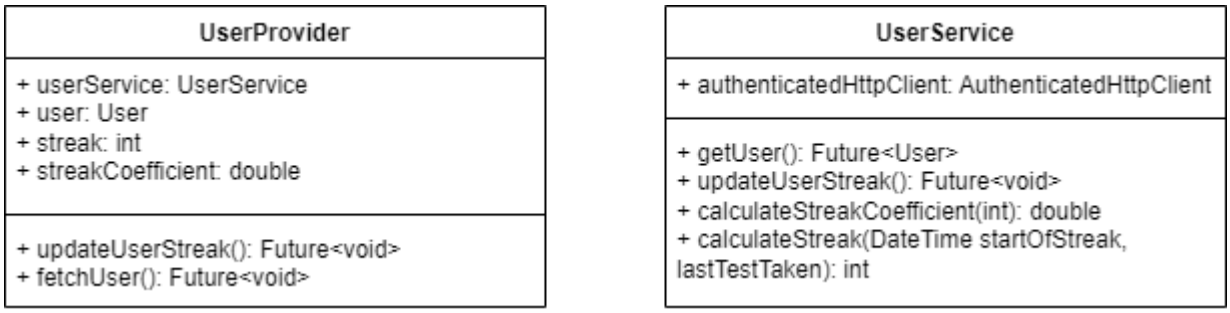

Рисунок 3.3 Демонстрація розподілу обов'язків між класами-провайдерами й класами-сервісами

3.1.3 Використання ієрархії класів для забезпечення гнучкості до змін

Вимагає уточнення також представлення типів питань у програмному коді. Безперечно, можна було б обмежитися одним універсальним класом Question, проте така конфігурація призвела б до великої кількості розгалуджень, потрібних щоразу, коли подальша робота програми залежить від типу питання. Це суттєво вплинуло б на легкість розширення функціоналу додатку в майбутньому (з додаванням нових типів питань, наприклад). Проблема полягала б не тільки в кількості необхідних змін для такого розширення, а й у наявності людського фактору в ході кожної такої зміни. Відповідно, було прийняте рішення відмовитися від такого підходу.

Натомість для забезпечення мінімальної кількості розгалуджень краще використати класову ієрархію. Абстрактний клас Question зберігатиме питання, коефіцієнт правильності відповіді, саму відповідь і пояснення до неї. Аби позбавитися розгалуджень в коді представлення, доцільно також додати до такого класу функції побудови віджетів відображення питання під час тесту й під час перегляду його результатів. Варто відмітити, що логіку побудови віджетів слід описати в окремих класах для кращого розподілу обов’язків, який ми все ж порушуємо. Таким чином, єдиним методом з розгалудженням буде конструктор об’єктів зі словників (JSON), адже в базі даних доведеться відображати типи питань як значення атрибуту «тип». Вважаю доцільним вибір чистішого коду, що краще притримується принципу DRY (Do not Repeat Yourself), на противагу чіткому дотриманню розподілу рівнів Model і View.

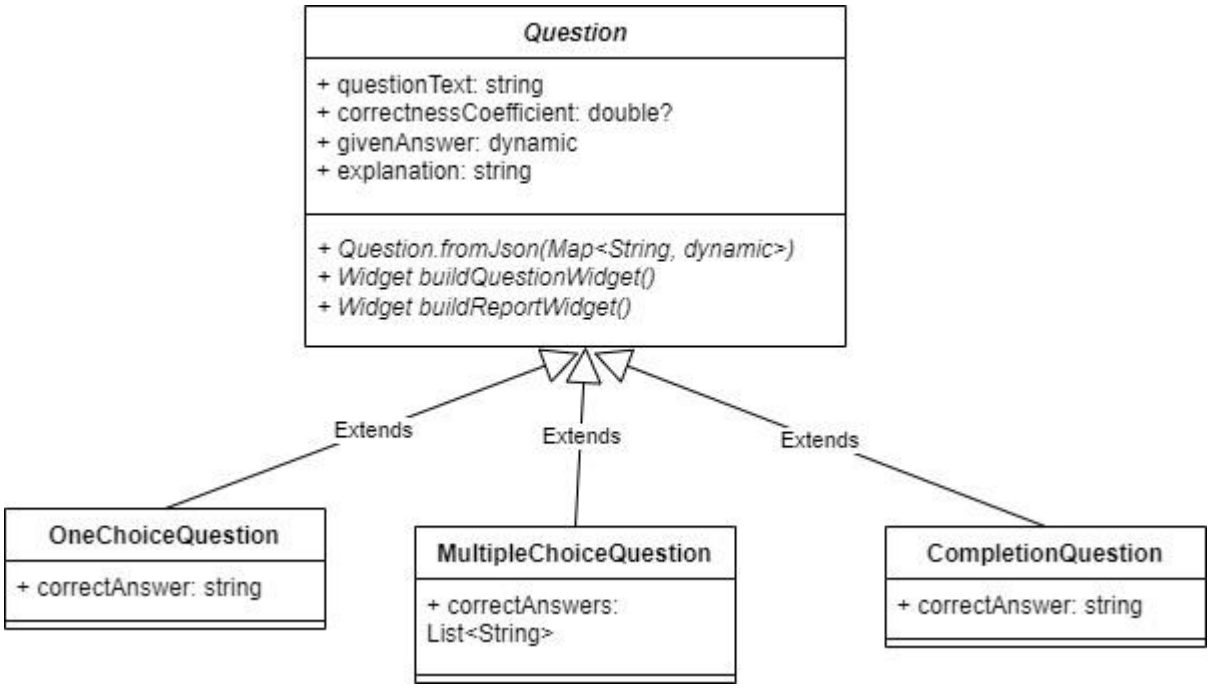

Рисунок 3.4 Орієнтовна демонстрація ієрархії класів-представлень тестових питань

3.2 Проєктування серверної частини застосунку

Під час проєктування серверної частини застосунку, потрібно пам’ятати його роль у трирівневій архітектурі програмного продукту [29]. Сервер має отримувати запити від клієнтської частини (рівня представлення), збирати необхідні для відповіді дані (або виконувати необхідні операції над ними), утилізуючи власні http-запити, або ж базу даних, після чого надсилати очікувану відповідь користувачу.

В випадку розроблюваного програмного продукту, за взаємодію з клієнтською частиною відповідатимуть маршрутизатори. Вони знаходитимуться на межі рівня сервісів і рівня представлення. Маршрутизатори в ході своєї роботи викликають сервіси, що представляють безпосередню логіку серверного рівня. Саме вони взаємодіють з даними через класи доступу до даних (DAO), які є посередниками між другим і третім рівнем архітектури застосунку в рамках

тришарової моделі. Потреби в використанні шаблону DTO немає через згадану специфіку мови JavaScript і системи керування базами даних MongoDB.

Варто також взяти до уваги необхідність аутентифікації вхідних запитів, яка буде необхідна на всіх шляхах обробки. Доцільно розробити окрему утиліту, яка, за рахунок інструментарію Express.js, буде відфільтровувати некоректні в контексті допуску запити перед їхнім потраплянням до функцій маршрутизаторів.

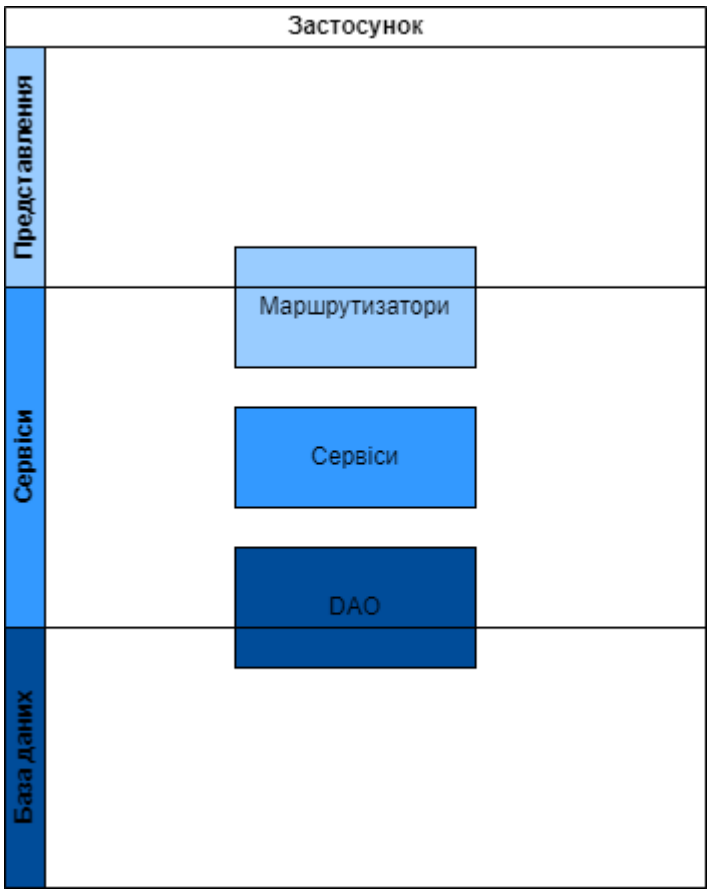

Рисунок 3.5 Серверна частина застосунку в контексті трирівневої архітектури

3.3 Проєктування бази даних

Важливо передбачити потенціал проєкту до розширення в структурі бази даних. Треба враховувати, що зміни на сервері доправити до кінцевого користувача

суттєво простіше, ніж зміни в додатку. Так, наприклад, в базі даних має знаходитися інформація про доступні для вибору рівні англійської мови для тестів, а також рекомендовані теми для них. До того ж, там має зберігатися остання обрана мова інтерфейсу користувача для відправки нагадувань. Певна інформація, що стосується статистики, також має зберігатися в базі даних, як-от кількість днів, які користувач відвідував додаток і проходив тести

У базі також мають зберігатися токени, за якими користувач зможе скидати пароль за потреби. Безперечно, знаходитися там вони мають лише певний час, що можна забезпечити можливістю MongoDB створювати короткострокові записи в базі даних, що видаляються через вказаний проміжок часу без втручання з боку клієнта.

Враховуючи акцент, зроблений на невичерпність набору питань, а також аби стимулювати користувачів проходити нові унікальні тести, було прийняте рішення не перевикористовувати тестові питання. Це дозволяє спростити представлення тестів у базі даних, а також їхню обробку на боці клієнта.

Незважаючи на те, що MongoDB не є реляційною базою даних, нижче була виконана спроба її схематичного зображення з використанням принципів зображення РБД. Типи даних записано в нетиповій для реляційної моделі формі через їхнє більше різноманіття. Ці типи даних відповідають позначенням, використовуваним у клієнтській і серверній частинах додатку.

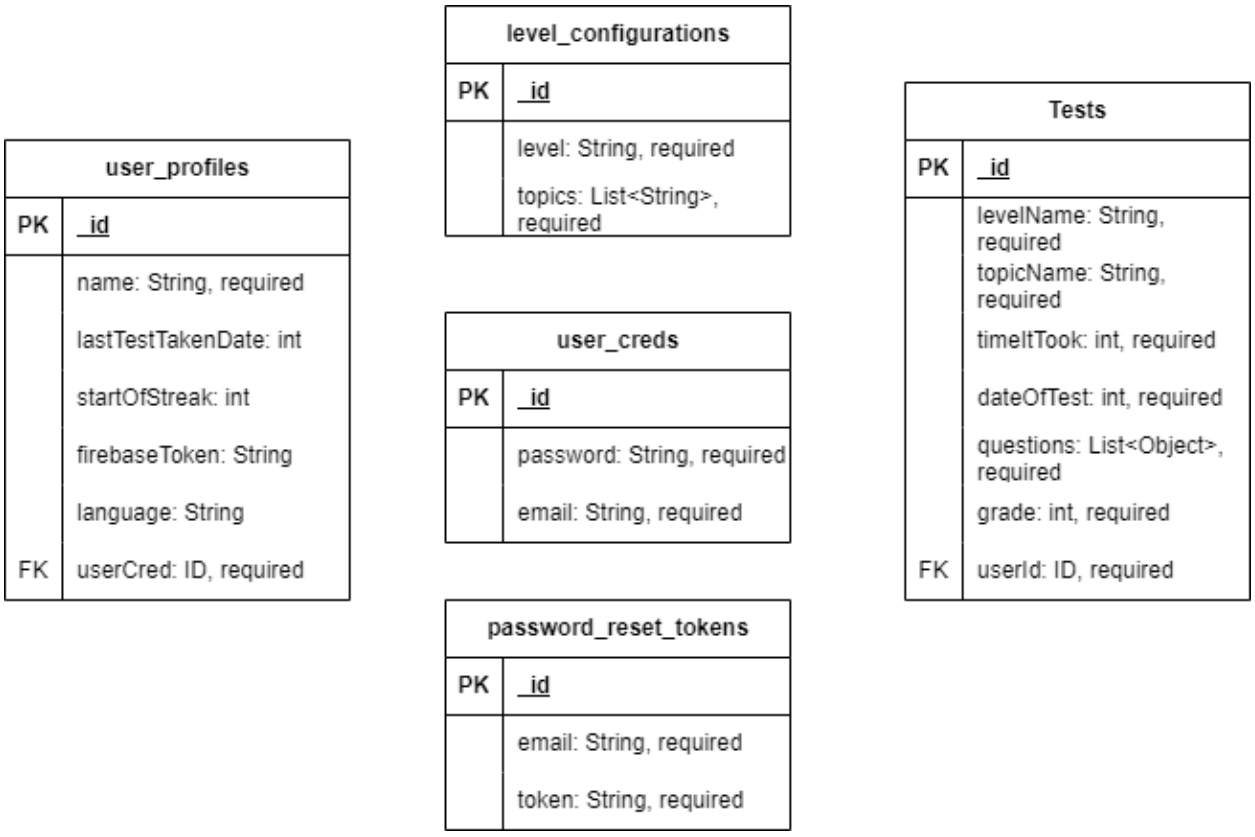

Рисунок 3.6 Схематичне представлення бази даних

4 Розробка додатку

4.1 Локалізація

Враховуючи потенційне майбутнє розширення аудиторії додатку та появу користувачів з інших країн, безперечно, варто додати механізм локалізації. Flutter пропонує власну бібліотеку, що могла б реалізовувати цей функціонал, - flutter_localizations. Незважаючи на переваги використання пакетів, створених розробниками самого фреймворку, варто врахувати деякі проблеми цієї бібліотеки:

а) файли локалізації зберігаються в форматі .arb (схожий на .json формат, що не допускає ієрархічне розташування ключів у файлі)

б) доступ до перекладів поза віджетами можливий тільки за наявності контексту деякого віджета

Щоб уникнути постійної передачі параметрів і довгих, складних для розуміння префіксів ключів перекладів, було прийняте рішення використати бібліотеку easy_localization, що пропонує доступ до перекладу в будь-якій частині додатку й збереження файлів з перекладами в форматі .json.

Наразі в додатку присутні українська й англійська мови. Мова інтерфейсу визначається автоматично, в залежності від обраної мови операційної системи.

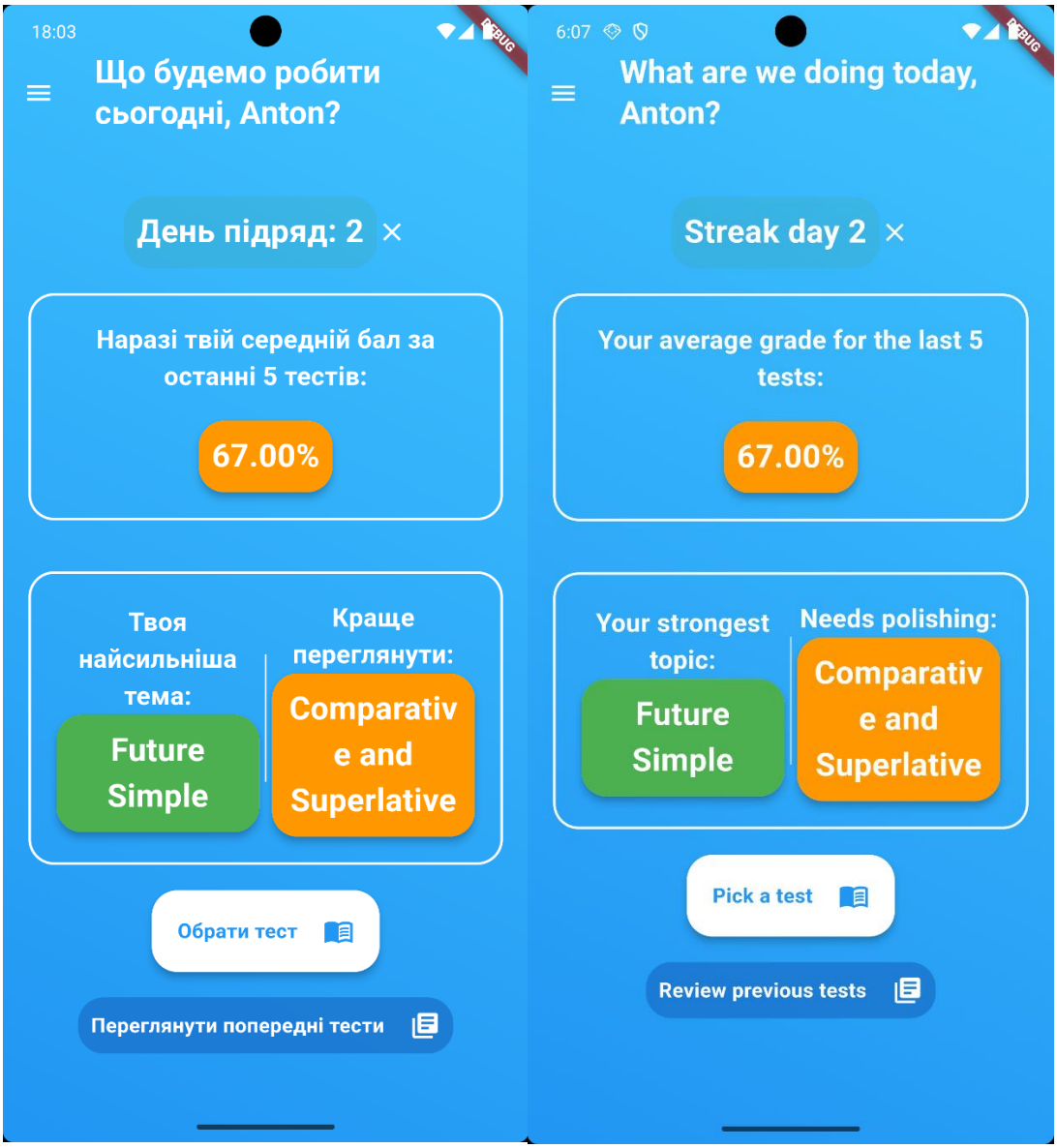

Рисунок 4.1 Головний екран додатку з різними обраними мовами операційної системи

4.2 Аутентифікація

4.2.1 Зберігання даних входу користувача

Для користувача передбачені можливості реєстрації, входу й скидання

паролю. Для повторної аутентифікації передбачається зберігання логіну та паролю на девайсі в зашифрованому вигляді з використанням пакету FlutterSecureStorage. Реалізація шифрування на кожній платформі специфічна: Android-девайси зберігають ці дані в зашифрованих SharedPreferences, iOS використовує технологію Keychain, а браузери – технологію WebCrypto для безпеки даних [30]. Пакет розрахований на збереження конфіденційної інформації, для чого він і використовується.

Користувач може за потреби вийти за акаунту, натиснувши кнопку в боковому меню. Функціонал бокового меню може бути з часом розширено.

4.2.2 Система оновлення паролю

В момент введення користувачем своєї поштової скриньки в відведеній для цього формі, після її перевірки на прив’язку до певного акаунту, сервер генерує токен, за яким користувач зможе скинути свій пароль, і на вказану поштову скриньку відправляється лист засобами Nodemailer з посиланням на форму введення нового паролю. Наразі для відправки листа використовується спеціально створений для навчальних цілей акаунт Google, але за потреби реально підключити такі сервіси як ProtonMail, щоб забезпечити вищу пропускну здатність сервісу скидання паролю, а також кращу ідентифікацію користувачами цих листів. Токен зберігається в базі даних протягом години разом із електронною адресою, за якою користувач ідентифікується в разі звернення. Токен видаляється після успішного оновлення паролю.

Форма скидання паролю реалізована з використанням шаблонізатора ejs. Дизайн віддалено повторює інтерфейс додатку. Бібліотеки для його створення не використовувалися.

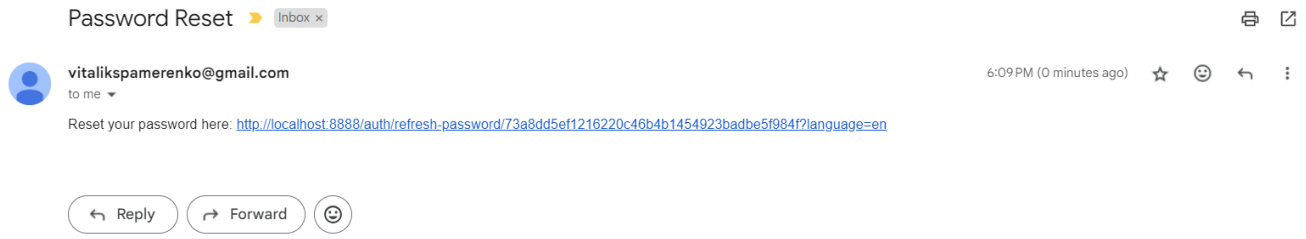

Рисунок 4.2 Лист-допуск до заміни пароля на пошті користувача

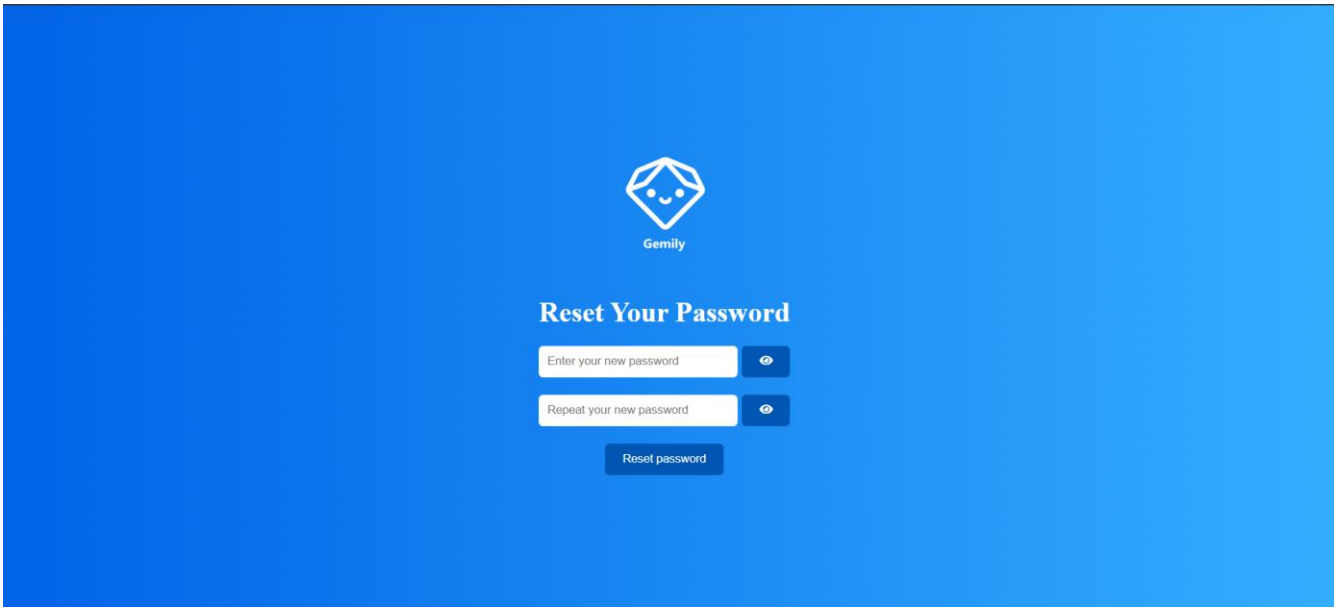

Рисунок 4.3 Форма заміни пароля в браузері

4.3 Головний екран

На головному екрані відображається статистика користувача: середня оцінка за останні п’ять тестів, а також теми з найвищим і найнижчим середніми балами за останні 10 тестів. Ці дані дозволяють користувачу зрозуміти, яку тему варто обрати для проходження наступного тесту. Головний екран застосунку можна бачити на рисунку 4.1.

4.4 Генерація тесту

На екрані конфігурації тесту можна обрати рівень англійської мови (доступні рівні зберігаються в базі даних), а також тему: рекомендовану (список рекомендованих тем зберігається в базі даних), або ж власну. Після підтвердження даних користувачем, вони передаються серверу, який в свою чергу вставивши ці дані в заздалегідь заготовлений текст, надсилає запит на сервери Vertex AI.

Текст запиту ретельно описує структуру тесту, що очікується у відповідь, а також деякі правила підбору завдань (зокрема - можливість відповісти без додаткового контексту, зображень тощо). З отриманої відповіді вилучається власне тест, описаний у форматі JSON, і передається назад клієнту, де ініціалізується TestController і поточний екран змінюється на екран проходження тесту.

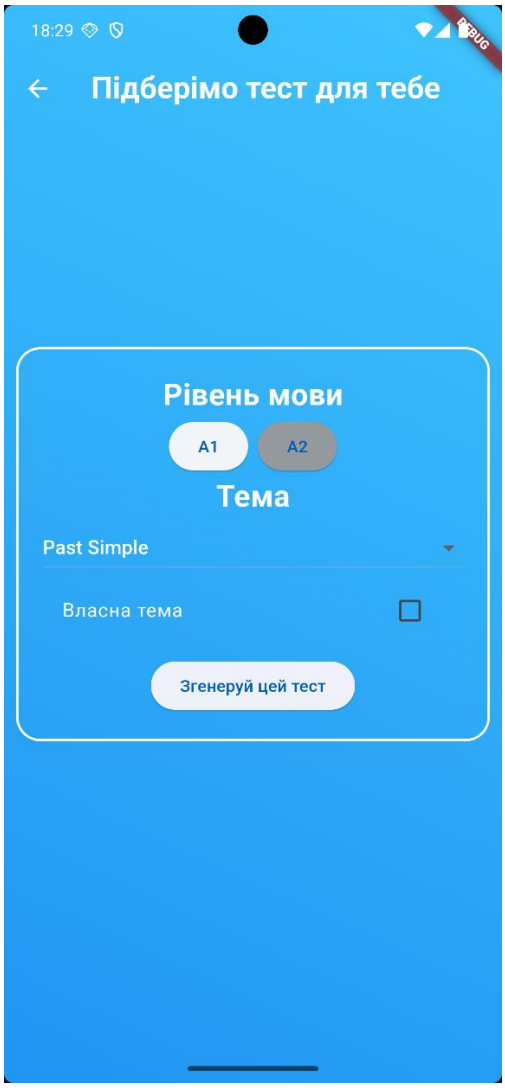

Рисунок 4.4 Меню генерації тесту

4.5 Реалізація процесу тестування

Екран проходження тесту відображає, окрім поточного питання, секундомір (що змушує користувача звертати увагу на час, який він витрачає на виконання завдань), а також номер питання. Після завершення виконання останнього тестового завдання, користувачу демонструється його бал у відсотковому відношенні. Пройдений тест зберігається в базу даних запитом до сервера.

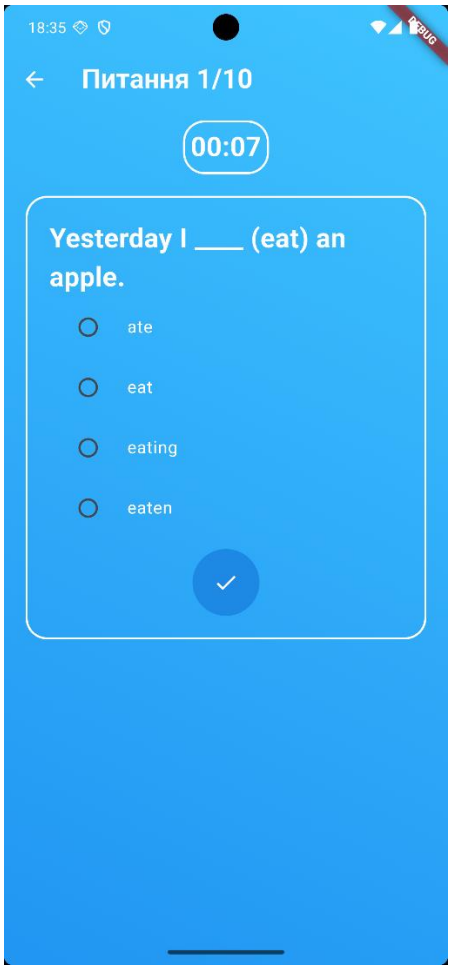

Рисунок 4.5 Процес тестування

4.6 Перегляд попередніх тестів

Користувач також має можливість переглянути попередні виконані тести, перейшовши на відповідний екран з головної сторінки. Для забезпечення стабільної роботи як клієнта, так і сервера, застосовується пагінація: тести відправляються користувачу порціями по 10 штук. Для перегляду наступної порції, користувач має опуститися вниз списку завантажених тестів. Підтримується пошук за темою тесту, а також сортування за часом, витраченим на проходження, датою проходження, балом, темою та рівнем (в порядку спадання або зростання).

Клікнувши по конкретному тесту, користувач може отримати більш детальну

інформацію про кожне питання, коефіцієнт правильності відповіді, а також – пояснення правильної відповіді.

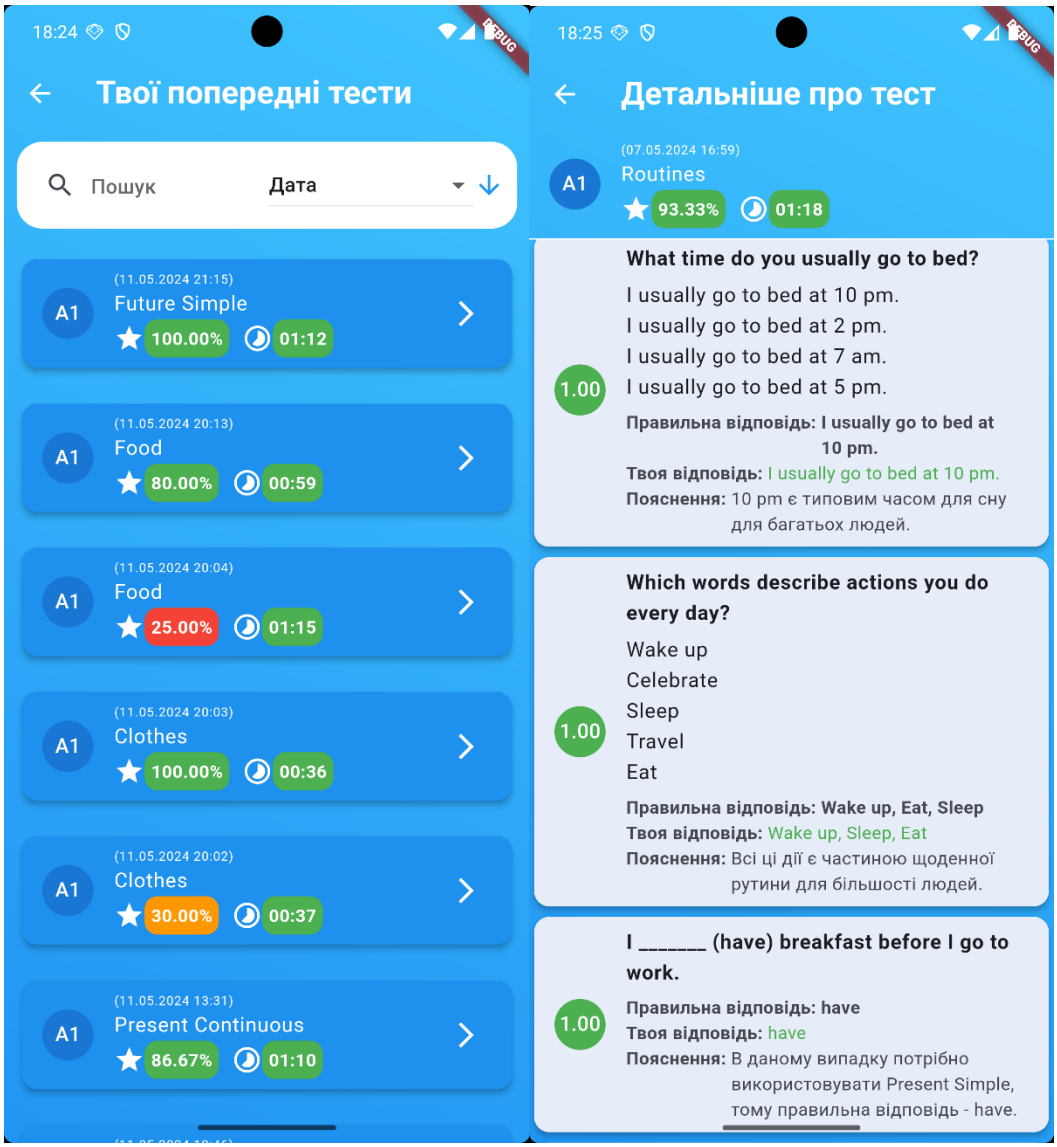

Рисунок 4.6 Перегляд результатів попередніх тестувань

4.7 Отримання сповіщень

На етапі входу користувача в акаунт, засобами бібліотеки Firebase отримується унікальний токен, який буде слугувати адресою цього користувача для відправки йому повідомлень. Цей токен зберігається в базі даних.

Для відправки повідомлень на боці сервера використано модулі firebase-admin, i18n і node-schedule. Firebase-admin дозволяє виконати надсилання скомпонованих повідомлень (бібліотека не вимагає виконання http-запитів напряму, що підвищує читабельність коду за рахунок приховання цього технічного аспекту в її методах), i18n потрібен для локалізації повідомлень (для цього використовується поточний мовний код користувача, що зберігається в його записі в базі даних), а node-schedule – дозволяє запускати процес розсилки повідомлень раз на певний час. В демонстраційних цілях наразі повідомлення відправляються всім користувачам раз на 1 хвилину, проте за потреби цю логіку можна або ускладнити, або просто змінити проміжок часу на більший. Другий варіант передбачає надсилання повідомлень навіть користувачам, що вже проходили тести за цей день, що має як свої переваги, так і недоліки. З одного боку, за використання такого підходу, користувачі згадають про додаток і необхідність регулярної практики більше за один раз на день. В випадку підготовки до екзаменів, декілька тестів, пройдених з певним інтервалом за день, краще, ніж один. З іншого боку, регулярність сповіщень може бути стимулом їх відімкнути. Тим не менш, автор схиляється до другого підходу.

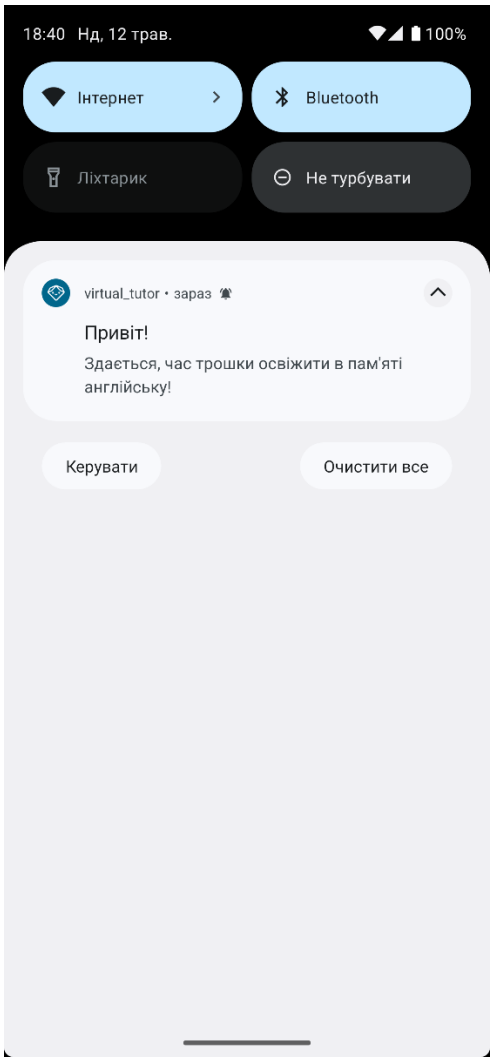

Рисунок 4.7 надіслане користувачу нагадування

4.8 Реалізація кросплатформеності

Незважаючи на те, що додаток в ході розробки не тестувався в браузері, на момент його першого запуску більшість складових інтерфейсу відображалися саме так, як вони мали б відображатися. Єдиною проблемою стали параметри, прив’язані до ширини екрану (здебільшого це стосувалося форм). Фактично єдині коригування, які були необхідні для того, щоб додаток безперешкодно працював у браузері, - це горизонтальні обмеження розмірів деяких полів.

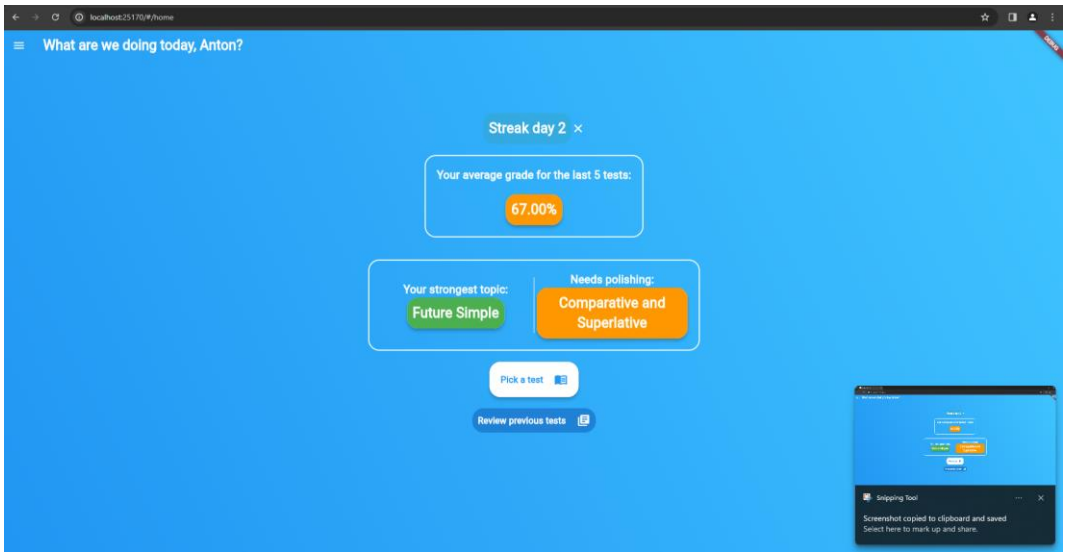

Рисунок 4.8 Головна сторінка застосунку в браузері

4.9 Вади та потенційні покращення продукту

4.9.1 Вади технології генерації тестів

Обраний підхід покращення роботи моделі за рахунок експериментів з уточнення запиту, зміною його формату тощо, а також за рахунок підбору експериментальним шляхом параметрів моделі (topK, topP, температура) дає задовільні, але не найкращі результати. Більшість проблем – з темами, пов’язаними з розвитком словникового запасу користувача. Через недостатнє розуміння знань пересічної людини, модель іноді генерує неочевидні питання, або ж навпаки надто ретельно допомагає користувачу, випадково розв’язуючи завдання за нього в процесі.

Можливим покращенням ситуації є дотренування моделі (fine-tuning) для виконання цього конкретного завдання. Інструментарій для цього процесу забезпечує використовуваний наразі Vertex AI [25]. За такої необхідності, проблемою буде знаходження або започаткування бази даних тестових завдань англійською мовою. Спроба поверхневого пошуку такої бази даних була зроблена

в ході створення застосунку, проте була неуспішною. Можлива автоматизація збору завдань з відкритих джерел (наприклад – з підручників з англійської мови) та створення таким чином власної бази даних як альтернативний шлях вирішення проблеми.

Ще одна можливість оптимізації процесу генерації тестів – розбиття процесу перевірки й процесу генерації тесту на два окремі запити до мовної моделі. В такому разі можлива більш м’яка перевірка тесту, що передбачає порівняння результату з очікуваним і надання об’єктивної оцінки в разі часткового збігу.

4.9.2 Розширення функціоналу додатку

Прикладами потенційних розширень функціоналу додатку є відображення статистики обраних користувачів (зі списку друзів, наприклад) для стимулювання конкуренції між клієнтами, додання функції консультації з великою мовною моделлю щодо граматичних питань в форматі чату, надання теорії до кожної з рекомендованих тем у вигляді тезового конспекту тощо.

Висновки

У даній роботі було розглянуто проблематику й потенційні шляхи подолання недоліків сучасних віртуальних репетиторів, зокрема завдяки використанню засобів штучного інтелекту. В цьому контексті досліджено способи використання великих мовних моделей у програмних продуктах та популярний наразі інструментарій створення кросплатформенних застосунків та серверів.

До того ж, у роботі розглянута проблема адаптації класичних шаблонів програмування під особливості нових інструментів розробки програмного забезпечення.

В ході виконання роботи було встановлено, що використання великих штучних моделей для генерації тестових завдань можливе, і навіть за відсутності дотренування для створення тестів на конкретні теми, результати на більшості завдань є задовільними, хоч для комерційного використання, безперечно, таке дотренування необхідне.

Результатом виконаної роботи є кросплатформенний застосунок, який дозволяє користувачу отримувати до певної міри унікальні тести з англійської мови, згенеровані засобами штучного інтелекту, слідкувати за власною успішністю, проводити роботу над помилками, а також отримувати нагадування задля забезпечення регулярної й успішної підготовки до мовних іспитів.

Покращення результату можна забезпечити наданням користувачу більшої кількості теоретичної інформації, стимулюванням конкуренції з іншими, а також – дотренуванням моделі для забезпечення кращих результатів.

Список використаних джерел

1. Оцінка громадянами ситуації в країні, довіра до соціальних інститутів, політиків, посадовців та громадських діячів, ставлення до окремих ініціатив органів влади (липень 2023р.) [Електронний ресурс] // Разумков центр – 2023 – Режим доступу до ресурсу: https://razumkov.org.ua/napriamky/sotsiologichni-doslidzhennia/otsinka-gromadianamy-sytuatsii-v-kraini-dovira-do-sotsialnykh-instytutiv-politykiv-posadovtsiv-ta-gromadskykh-diiachiv-stavlennia-do-okremykh-initsiatyv-organiv-vlady-lypen-2023r
2. МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ЩОДО ЗАБЕЗПЕЧЕННЯ ЯКІСНОГО ВИВЧЕННЯ, ВИКЛАДАННЯ ТА ВИКОРИСТАННЯ АНГЛІЙСЬКОЇ МОВИ У ЗАКЛАДАХ ВИЩОЇ ОСВІТИ УКРАЇНИ (наказ №898 від 25 липня 2023 р) / Міністерство освіти і науки України – Режим доступу до ресурсу: https://mon.gov.ua/ua/npa/metodichni-rekomendaciyi-shodo-zabezpechennya-yakisnogo-vivchennya-vikladannya-ta-vikoristannya-anglijskoyi-movi-u-zakladah-vishoyi-osviti-ukrayini
3. ПОТЕНЦІЙНІ УЧАСНИКИ НМТ СТВОРИЛИ ПОНАД 213 ТИСЯЧ ПЕРСОНАЛЬНИХ КАБІНЕТІВ [Електронний ресурс] // Український центр оцінювання якості освіти – 2024 – Режим доступу до ресурсу: https://testportal.gov.ua/potentsijni-uchasnyky-nmt-stvoryly-ponad-213-tysyach-personalnyh-kabinetiv/
4. РІВЕНЬ ВОЛОДІННЯ АНГЛІЙСЬКОЮ ТА ІНШИМИ ІНОЗЕМНИМИ МОВАМИ В УКРАЇНІ: РЕЗУЛЬТАТИ КІЛЬКІСНОГО СОЦІОЛОГІЧНОГО ДОСЛІДЖЕННЯ ПРОВЕДЕНОГО У ГРУДНІ 2022 – СІЧНІ 2023 [Електронний ресурс] / Максим Яшник // Київський міжнародний інститут соціології – 2023 – Режим доступу до ресурсу:

https://kiis.com.ua/?lang=ukr&cat=reports&id=1210&page=1&fbclid=IwAR104TqRYqK_chVY4Um4ttSrPRmz_J-v-wfhOV9oXIEHp--dz7PSMXZNbEE

5. What are LLMs [Електронний ресурс] // IBM – Режим доступу до ресурсу: https://www.ibm.com/topics/large-language-models
6. What are foundational models? [Електронний ресурс] // IBM – 2022 – Режим доступу до ресурсу: https://research.ibm.com/blog/what-are-foundation-models
7. Attention Is All You Need /Vaswani A., Shazeer N., Parmar N., J.U., L. J., A. N. G., L.K., I.P. // NIPS – 2017 – С. 3–4 – Режим доступу до ресурсу: https://arxiv.org/pdf/1706.03762
8. CONTEXTUAL TEMPERATURE FOR LANGUAGE MODELING // ICLR – 2020 – С. 1–2 – Режим доступу до ресурсу: https://openreview.net/pdf?id=H1x9004YPr
9. Softmax Temperature [Електронний ресурс] / Harshit Sharma // Medium – 2022 – Режим доступу до ресурсу: https://medium.com/@harshit158/softmax-temperature-5492e4007f71
10. PaLM for Text [Електронний ресурс] // Vertex AI – Режим доступу до ресурсу: https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/text
11. Популярність кросплатформенних фреймворків за трендами Google [Електронний ресурс] // Google Trends – Режим доступу до ресурсу: https://trends.google.com/trends/explore?cat=31&date=2016-03-17%202024-03-17&q=Flutter,React%20Native,Xamarin,MAUI,Ionic
12. Популярність кросплатформенних фреймворків за трендами StackOverflow [Електронний ресурс] // StackOverflow – Режим доступу до ресурсу: https://insights.stackoverflow.com/trends?tags=flutter%2Creact-native%2Cxamarin%2Cmaui%2Cionic-framework%2Ccordova

13. With Flutter, Google Aims Dart to Mobile App Cross-Development [Електронний ресурс] / Sergio De Simone // InfoQ – 2015 – Режим доступу до ресурсу: https://www.infoq.com/news/2015/12/flutter-dart-cross-platform/
14. Сучасні застосунки, створенні з використанням Flutter [Електронний ресурс] // Flutter – Режим доступу до ресурсу: https://flutter.dev/showcase
15. Flutter's Benefits and Drawbacks [Електронний ресурс] // Parth Bhanderi / Medium – 2023 – Режим доступу до ресурсу: https://medium.com/@parthbhanderi01/flutters-benefits-and-drawbacks-b268c1fe0f7f
16. Flutter architectural overview [Електронний ресурс] // Flutter – Режим доступу до ресурсу: https://docs.flutter.dev/resources/architectural-overview
17. 8 Advantages of Node.js Development for Startups in 2024 [Електронний ресурс] / Anna Dziuba // RelevantSoftware – 2024 – Режим доступу до ресурсу: https://relevant.software/blog/8-advantages-of-node-js-for-startups/#Why_Nodejs_and_How_it_Works
18. Express.js Fundamentals [Електронний ресурс] / Zulaikha Geer // Edureka – 2019 – Режим доступу до ресурсу: https://medium.com/edureka/expressjs-tutorial-795ad6e65ab3

JSON and BSON [Електронний ресурс] // MongoDB – Режим доступу до ресурсу: https://www.mongodb.com/json-and-bson

20. MongoDB Wire Protocol [Електронний ресурс] // MongoDB – Режим доступу до ресурсу: https://www.mongodb.com/docs/manual/reference/mongodb-wire-protocol/

Advantages of MongoDB [Електронний ресурс] // MongoDB – Режим доступу до ресурсу: https://www.mongodb.com/resources/compare/advantages-of-mongodb

22. Платформа HuggingFace [Електронний ресурс] – Режим доступу до ресурсу: https://huggingface.co/

23. GPT-4 architecture, datasets, costs and more leaked [Електронний ресурс] / Maximilian Schreiner // The Decoder – 2023 – Режим доступу до ресурсу: https://the-decoder.com/gpt-4-architecture-datasets-costs-and-more-leaked/

24. Meet the first version of Gemini— our most capable AI model [Електронний ресурс] // Deepmind – Режим доступу до ресурсу: https://deepmind.google/technologies/gemini/#gemini-1.0

25. Introduction to Vertex AI [Електронний ресурс] // Google Cloud – Режим доступу до ресурсу: https://cloud.google.com/vertex-ai/docs/start/introduction-unified-platform

26. flutter_local_notifications [Електронний ресурс] – Режим доступу до ресурсу: https://pub.dev/packages/flutter_local_notifications

MVC [Електронний ресурс] // MDN Web Docs Glossary – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Glossary/MVC

BuildContext class [Електронний ресурс] – Режим доступу до ресурсу: https://api.flutter.dev/flutter/widgets/BuildContext-class.html

29. What is three-tier architecture? [Електронний ресурс] // IBM – Режим доступу до ресурсу: https://www.ibm.com/topics/three-tier-architecture

30. flutter_secure_storage [Електронний ресурс] – Режим доступу до ресурсу: https://pub.dev/packages/flutter_secure_storage