Міністерство освіти і науки України
Державний університет «Одеська Політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра системного програмного забезпечення

Синюк Іван Миколайович
студент групи АС-161

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

« Мобільне застосування для вивчення англійської мови »

Спеціальність:
121 – Інженерія програмного забезпечення

Освітня програма:
Інженерія програмного забезпечення

Керівник:
Роговський Вадим Томович
канд. техн. наук, доцент

Одеса – 2021

ЗМІСТ

Міністерство освіти і науки України
Державний університет «Одеська Політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра системного програмного забезпечення

Рівень вищої освіти: другий (магістерський)

Спеціальність: 121 – Інженерія програмного забезпечення

Спеціалізація: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Любченко В.В.
«___»_____20__р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Синюка Івана Миколайовича, група АС-161

1. Тема роботи: *Мобільне застосування для вивчення англійської мови*

Керівник роботи: Роговський Вадим Томович, *канд. техн. наук, доцент*

Затверджені наказом ректора від «25» жовтня 2021р. № 374-в

2. Зміст роботи: підстави для розробки, опис предметної області, формалізація вимог, проектування системи, програмна реалізація, тестування

3. Перелік ілюстративного матеріалу:

Згідно зі слайдами презентації

4. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Дата видачі завдання: «___»_____20__р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	Підстави для розробки	10.09.2021 – 15.09.2021	вик.
2	Опис предметної області	16.09.2021 – 23.09.2021	вик.
3	Формалізація вимог	25.09.2021 – 12.10.2021	вик.
4	Проектування програмної системи	21.10.2021 – 01.11.2021	вик.
5	Програмна реалізація та тестування	02.11.2021 – 21.11.2021	вик.
6	Оформлення пояснювальної записки та графічного матеріалу	22.11.2021 – 29.11.2021	вик.

Здобувач вищої освіти _____ *І.М. Синюк*

Керівник роботи _____ *В.Т. Роговський*

АНОТАЦІЯ

Метою дипломної роботи є покращення рівня володіння англійською мовою шляхом розробки спеціалізованого мобільного застосування. Мовою розробки є JavaScript, а саме її фреймворк для створення мобільних застосувань React Native. Основною перевагою використання React Native – це кросплатформена розробка, тобто можливість використовувати застосування як на Android так і IOS. Як результат роботи виконана програмна реалізація ПЗ для посилення рівня іноземної мови у користувачів. Забезпечує можливість проходженню тестів з іноземної мови, створення фото профілю. Також в мобільному застосуванні існує система рівнів володіння англійською мовою(від Початківця до Експерта), що буде заохочувати користувачів вивчати англійську мову.

Ключові слова: JavaScript, Android, IOS, збереження даних, інтерактив, навчання, спілкування.

ABSTRACT

The purpose of the qualification work is to increase the level of foreign language proficiency by creating a mobile application. The development language is JavaScript, namely its framework for creating mobile applications React Native. The main advantage of using React Native is cross-platform development, ie the ability to use applications on both Android and iOS. As a result of the work the software implementation of the software for strengthening the level of foreign language for users was performed. Provides the ability to exchange messages, take tests in a foreign language.

Keywords: JavaScript, Android, IOS, data storage, interactivity, learning, communication.

ВСТУП

Метою дипломної роботи є покращення рівня володіння англійською мовою шляхом розробки спеціалізованого мобільного застосування. Придбання і розвиток необхідних практичних умінь і навичок відповідно до вимог до рівня підготовки випускника згідно до затвердженої тематики дипломного дослідження.

Головними завданнями роботи є:

- Аналіз існуючих аналогів;
- Діаграма варіантів використання;
- Сценарії варіантів використання;
- Нефункціональні вимоги;
- Під час проектування системи було проведено:
    - Проектування архітектури програмного продукту;
    - Проектування діаграми програмних класів;

Сьогодні англійська мова є однією з найпопулярніших мов світу. Приблизно 1,5 млрд. людей володіє англійською мовою, і десь 1 млрд. вивчає її. Англійська охопила всі сфери життєдіяльності людини: наука, засоби масової інформації, навчання, робота і дозвілля.

Що ми отримаємо вивчивши англійську мову?

-Перспективна робота;

-Можливість йти в ногу з часом;

-Освіта за кордоном;

-Подорожі;

-Комунікація з людьми будь-якої країни;

-Нові знайомства[1].

У першому розділі роботи проводиться аналіз можливості застосування системи.

Другий розділ містить опис технічного завдання на розробку програмної системи.

У третьому розділі було сформовано планування програмної системи.

Четвертий розділ містить проектування клієнтської частини мобільного застосування для вивчення англійської мови.

У п’ятому розділі наведена програмна реалізація системи.

Шостий розділ містить посібник користувача.

Сьомий розділ містить тестування розробленої системи.

РОЗДІЛ 1 АНАЛІЗ МОЖЛИВОСТІ ЗАСТОСУВАННЯ СИСТЕМИ

Розглянемо деякі проблеми, з якими зустрічаються користувач при вивчені англійської мови.

Нерозуміння англійської мови на слух.

Щоб спілкуватися англійською, потрібно не тільки вміти говорити, а й розуміти, що вам відповідають. Нерозуміння на слух - дуже поширена проблема. Навіть люди, які мають великий запас слів і знають граматику, стикаються з нею. Проблема полягає в тому, що у людини немає досвіду аудіювання (розпізнавання мови). Зіткнувшись з легкої промовою, людина не розуміє сенсу сказаного, так як не може розпізнати окремі слова. Адже вони зливаються воєдино. Рішення проблеми: Розпізнавання англійської мови на слух - це навичка, над яким необхідно працювати. Для того щоб розвинути його, вам потрібно слухати якомога більше англійської мови. В **EasyLEng** ви можете слухати лекції англійською.

Постійне забування англійських слів.

Вам знайома ситуація, коли ви не можете згадати потрібне слово, хоча ви точно знаєте, що вчили його? Ви стовідсотково стикалися з такою проблемою. Відбувається це тому, що люди звикли зубрити слова. Якщо слова вчити таким способом, ви зможете запам'ятати лише деякі слова. Інші або підуть в пасивний запас, або забудуться. В результаті вам доведеться витратити багато часу на вивчення нових слів. Рішення проблеми: вчити слова потрібно правильно. Потрібно відразу ж використовувати їх у своїй промові. Для цього треба скласти декілька речень з цим словом. В **EasyLEng** ви можете відразу написати в загальний чат речення з цим словом та вам обов’язково хтось відповість.

Нездатність думати англійською мовою.

Ще однією проблемою є постійний переклад того, що ви хочете сказати, з рідної мови на англійську. Через це ви говорите повільно і постійно замислюєтеся, як побудувати речення. Адже вам спочатку потрібно придумати рідною мовою те, що ви хочете сказати, потім згадати потрібне слово англійською і правильно побудувати речення. Рішення проблеми: щоб навчитися думати на мові, потрібно кожен шматочок теорії, кожне слово відпрацьовувати до автоматизму. Тоді у вашій голові відразу будуть виникати англійські слова і складатися в правильні пропозиції.[2].

EasyLEng – мобільне інтерактивне застосування, за допомогою якого всі вищезгадуванні проблеми можливо вирішити.

Критичний аналіз програмного забезпечення для мобільного застосування з вивченням іноземної мови.

Вивчення англійської мови - на перший погляд просте завдання. Але якщо уважно проаналізувати всі проблеми та складнощі, то можна прийти до висновку, що вивчення іноземної мови - складна і детальна робота, в якій необхідна структурна праця та бажання людини. Далі представлені огляд сучасних мобільних Web/Anroid/IOS застосувань для вивчення іноземної мови.

ПЗ LinguaLeo. Один із секретів успіху цієї програми для вивчення англійської мови - ігрова форма навчання. Ваш власний симпатяга-левеня жадає фрикадельок, отримати які можна тільки при проходженні уроків.

Цільова аудиторія – школярі, діти.

Переваги: інтуїтивний інтерфейс; відкритий код продукту; універсальність багатьох умов; зберігання особистих даних: контактів, списків і т.д., не залежить від підключення до інтернету.

Недоліки: працює на операційній системі для смартфонів Android; відсутня гнучка настройка відображення завдань; БД зберігатися тільки на смартфоні.

Висновок

Вивчення англійської мови є необхідною складовою частиною сучасної людини. Тому зараз існує дуже багато застосувань, які вирішують ту чи іншу проблему.

ПЗ EdWords. Один із секретів успіху цієї програми для вивчення англійської мови - ігрова форма навчання.

Переваги: відкритий код продукту; універсальність багатьох умов; зберігання особистих даних: контактів, списків і т.д., не залежить від підключення до інтернету.

Недоліки: працює на операційній системі для смартфонів Android; відсутня гнучка настройка відображення завдань; БД зберігатися тільки на смартфоні.

Висновок

Вивчення англійської мови є необхідною складовою частиною сучасної людини. Тому зараз існує дуже багато застосувань, які вирішують ту чи іншу проблему.

EasyLEng – необхідний продукт в умовах вивчення іноземної мови, розрахований на школярів, студентів, дітей дошкільного віку.

В таблиці 1.1 представлені критерії порівняння чотирьох описаних вище систем планувальників.

Таблиця 1.1 – Критерії оцінки ПЗ для систем з вивченням іноземних мов.

Критерії \ ПЗ	LinguaLeo	Ed Words	Easy ten	**EasyLEng**
1. Android-доступ	+	+	+	+
2. Безкоштовність	-	-	-	+
3. Кросплатформеність	+	-	+	+
4. Розрахований на багато користувачів	-	+	+	+
5. Захищеність ПЗ	-	+	+	-
6. Локалізація	+	-	-	+

Продовження таблиці 1.1

7. Складність в освоєнні програми	+	-	-	-
8. Синхронізація з іншими програмами	-	+	+	+

Виходячи з даних в таблиці, ми бачимо, що мій програмний продукт буде безкоштовним, кросплатформеним, простий інтерфейс, синхронізацію з іншими програмами, та локалізований під різних користувачів з різними вадами зору та слуху. Також є можливість обмінюватись особистою інформацією між користувачами. Недоліком же є відсутність незахищеність ПЗ.

РОЗДІЛ 2 ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ “EasyLEng”

2.1 Загальні відомості

Повне найменування ПЗ*: Розробка мобільного застосування для вивчення англійської мови*.

Умовне позначення програмного забезпечення*:* «**EasyLEng**»

Планові строки початку та закінчення роботи зі створенню системи: представлені в таблиці 2.1 та на діаграмі Ганта (рисунок 2.2).

Таблиця 2.1 – Планові строки початку та закінчення роботи зі створенню системи

Назва етапу	Початок	Днів	Кінець
Підстави для розробки	10.09.2021	5	15.09.2021
Опис предметної області	16.09.2021	7	23.09.2021
Формалізація вимог	25.09.2021	18	12.10.2021
Проектування програмної системи	21.10.2021	11	01.11.2021
Програмна реалізація та тестування	02.11.2021	19	21.11.2021
Оформлення пояснювальної записки та графічного матеріалу	22.11.2021	7	29.11.2021

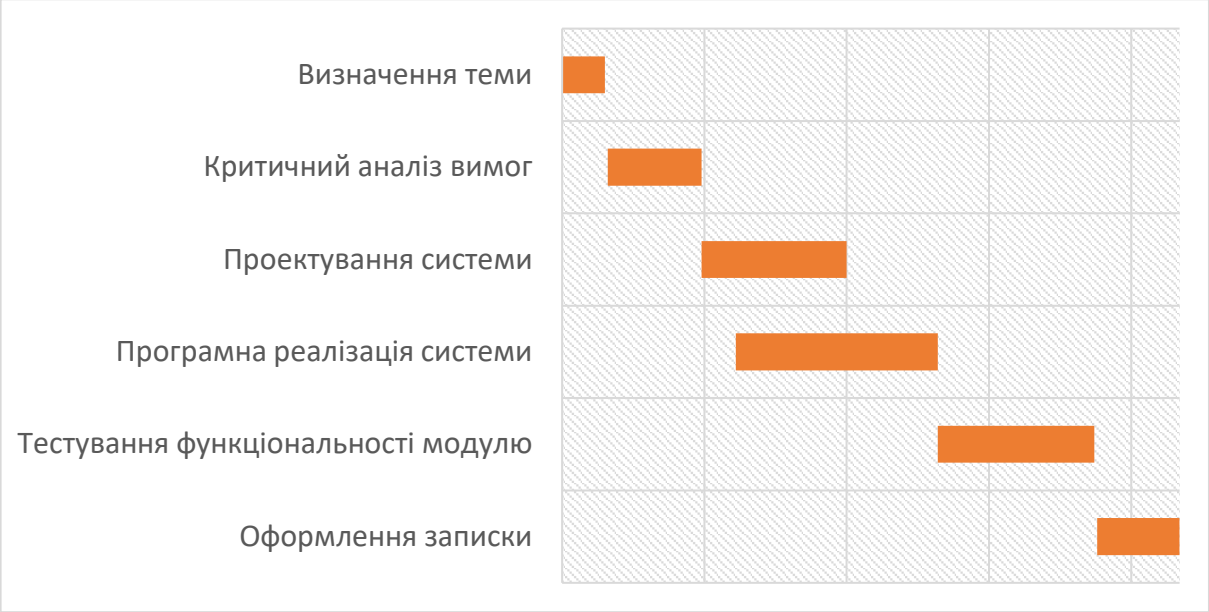

Рисунок 2.2 – Діаграма Ганта процесу розробки ПЗ «**EasyLEng**»

Порядок оформлення та пред'явлення замовнику результатів робіт за створенням системи. Результати роботи будуть представлені згідно з вимогами до виконанню дипломних робіт бакалаврів. Результат проектування системи «**EasyLEng**» буде представлений у вигляді мобільного програмного застосування.

2.2 Призначення та мети створення системи

Призначення системи: Програмне забезпечення «**EasyLEng**» призначений для вивчення англійської мови.

Мета створення системи: надання можливості безкоштовного використання та збільшення кількості функцій подібних ПЗ. Програмний продукт буде виконувати наступні задачі:

- Можливість проходити тести з англійської мови
- Система рейтингів та конкуренції між користувачами
- Можливість завантажувати фото в профіль

2.3 Вимоги до системи

Згідно до нотації Rational Unified Process (RUP), програмне забезпечення має вимоги – умови або можливості, якими програмне забезпечення має відповідати.

Стандарт IEEE Standard Glossary of Software Engineering Terminology [посилання] має такі вимоги для програмного забезпечення:

- умови або можливості, які необхідні користувачу для вирішення проблем або досягнення мети;

- умови або можливості, якими програмне забезпечення або її компоненти, щоб виконати контракт або задовольнити стандартам, специфікаціям або іншим формальним документом;

- документоване відображення умов або можливостей для пунктів 1 та 2.

Для опису вимог до програмного забезпечення «**EasyLEng**» буде використано специфікація RUP та модель FURPS+ . Назва утворена від перших букв слів:

- Functionality – функціональні вимоги: властивості, можливості, безпека. Вони є основними, по цим вимогам будуються діаграми варіантів використання (відомі як Use-Case Diagram);
- Usability – вимоги до зручності використання: людський фактор, послідовність, документація;
- Reliability – вимоги до надійності: частота можливих збоїв, відмовостійкість, відновлюваність, передбачення стійкості;
- Performance – вимоги до продуктивності: час відгуку, використання ресурсів, ефективність, потужність, масштабованість;
- Supportability – вимоги до підтримки: можливість підтримки, гнучкість, можливість модифікацій, модульність, розширюваність, локалізація.

Функціональні вимоги

Функціональні вимоги записуються за допомогою таких правил:

- програмне забезпечення повинно дозволяти користувачеві створити свій власний акаунт;

- програмне забезпечення повинно дозволяти користувачеві проходити тести з англійської мови;

Іншим способом є діаграма варіантів використання (Use-Case Diagram). Use-Case Diagram – це сценарна техніка опису взаємодії, який допомагає описати як вимоги користувача, так і вимоги до взаємодії систем та опису взаємодії людей та компаній у реальному житті. У розробці ПЗ цю техніку часто застосовують для проектування і опису взаємодії користувача і системи, тому назва Use-Case часто сприймається як синонім вимоги людини-користувача до вирішення певної задачі в системі.

На рисунку 2.3 зображено Use-Case діаграму, яка у загальному вигляді відображає функціональні вимоги до системи «**EasyLEng**».

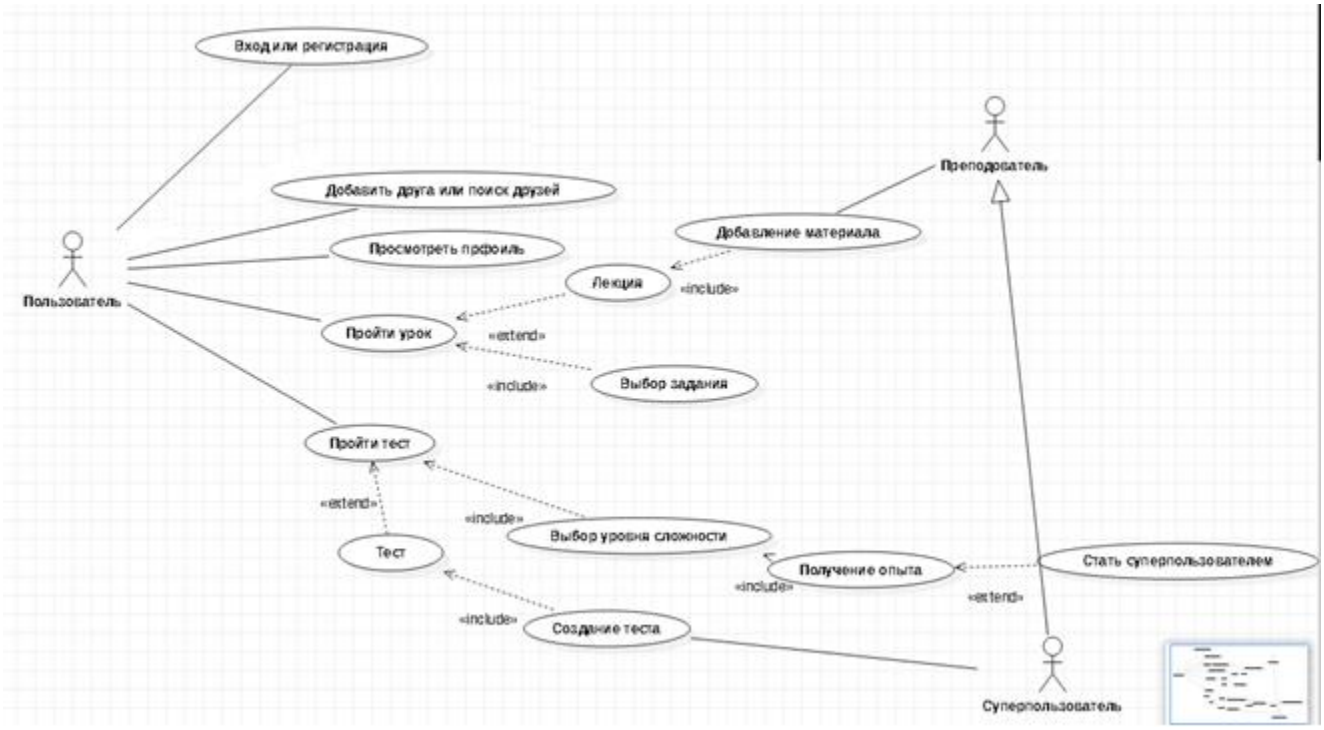

Рисунок 2.3 – Use-Case діаграма програмного забезпечення «**EasyLEng**»

В таблиці 2.2 описані дані прецеденти та їх короткий опис для більшого розуміння концепції програмного продукту.

Таблиця 2.2 – Опис прецедентів.

Прецедент	**Короткий опис**
Аунтефікація	Цей прецедент викликається актором «Користувач» який авторизується в системі, та дозволяє йому використовувати застосування
Додати друга	Цей прецедент викликається актором «Користувач». Користувач додає до своїх контактів друга
Пройти тест	Прецедент викликається актором «Користувач». Дозволяє користувачеві пройти тест.
Пошук даних	Прецедент викликається актором «Студент». Після заповнення даних в систему, дозволяє здійснювати процес пошуку по системі. Має відношення << include >> до прецеденту «За ПІБ», «По кафедрі», «По групі», «За номером телефона».

Нефункціональні вимоги

Згідно з стандарту ISO 9241-11 термін Usability означає міру якості користувацького досвіду, набутого при взаємодії з продуктом або системою, наприклад, веб-сайтом, програмним додатком і т.д.

- естетика та логічність призначеного для користувача інтерфейсу. Програмне забезпечення, яке орієнтоване на роботу з розробниками, повинен мати зовнішній вигляд ділового додатка. Оформлення має бути виконано в стриманому стилі і строгій колірній гаммі. Є частина інтерфейсу, в якій будуть розташовані діаграма класів та код, з якими буде проведена робота.
- захист від людського фактора. Вчинення помилок – це природньо. В цьому разі програмне забезпечення повинна бути максимально підготовлена до виявлення помилок та їх виправлення.
- довідкова інформація в ПЗ. Система повинна супроводжуватися довідковою документацію для розробників і містити інформацію про правила додавання

Під час формування нефункціональних вимог були визначені такі сценарії якості для розроблювальної системи.

Надійність:

- введення студентом дані, які необхідно зберегти, система зберігає у 90% випадку;
- при неможливості взаємодії з даними, система перейде на інші ресурси у 65%.

Зручність використання:

- надання підтримки та допомоги розробником протягом 6-ти годин від відправки запиту;
- кількість натискань пальцем, що потрібні для відкриття необхідного екрану, складає не більше 4-ох;
- кількість натискань мишею, що потрібні для збереження даних користувача, складає не більше 5-ти;

Продуктивність:

- час завантаження та відображення даних користувача на екрані складає не більше 4-ох секунд;
- час збереження даних для студента складає не більше 2-ух секунд;

Безпека:

- система захищає дані користувача від неправомірного доступу сторонніх осіб у 80% випадків;
- система захищає дані користувача від порушення цілісності у 79% випадків.

2.4 Сценарії варіантів використання

На діаграмі варіантів використання 14 варіантів використання програмного забезпечення **EasyLEng**. Нижче – опис кожного варіанту використання розроблюваної системи.

Авторизація

1. Користувач відкриває ПЗ.
2. Користувач вводить свій номер телефону.
3. Система валідує номер на правильність вводу та відправляє на сервер
4. Сервер відправляє користувачеві код-підтвердження

5. Користувач вводить код
6. Система перенаправляє користувача на головну сторінку «Профіль»

Переглянути список користувачів

1. Користувач переходить в «Користувачі»
2. Система відправляє запит на сервер.
3. Сервер відправляє данні про всіх користувачів
4. Система виводить список всіх користувачів.

Пройти тест

1. Користувач переходить в «Тести»
2. Користувач вибирає доступний йому тест
3. Система перенаправляє користувача на сторінку с вибраним тестом
4. Користувач відповідає на кожен тест почергово
5. Система оброблює кожну відповідь
6. Система відправляє данні на сервер
7. Система відображає набрану оцінку

Пройти урок

1. Користувач переходить в «Матеріали»
2. Користувач вибирає необхідний йому матеріал
3. Система перенаправляє користувача на сторінку с вибраним метаріалом
4. Система відображає матеріал вибраного уроку
5. Користувач завершує урок
6. Система відправляє данні на сервер про завершення даного уроку
7. Система повертає користувача на початкову сторінку

Створити тест

1. Користувач переходить на сторінку створити тест
2. Система перевіряє чи являється даний користувач Експертом.
3. Система перенаправляє його на сторінку створення тестів

4. Користувач почергово придумує 10 тестів та відповіді до них.
5. Система валідує введені поля.
6. Система відправляє на сервер новий тест.

Стати Експертом

1. Користувач проходить тест на Експерта
2. Система перевіряє рівень користувача
3. При достатній кількості балів система переназначає користувача як Експерта.
4. Система відправляє данні на сервер.

Додання фотографії профілю

1. Користувач відкриває свої фотографії
2. Система запитує користувача про можливість додання нового фото
3. Користувач вибирає Додати нове фото профілю
4. Система запитує механізм додання нового фото
5. Користувач робить фото або завантажує з галереї
6. Система відправляє данні на сервер

РОЗДІЛ 3 ПЛАНУВАННЯ ПРОГРАМНОГО ПРОЄКТУ

3.1 Оцінка тривалості виконання проекту

Оцінювання тривалості розробки проекту виконано за допомогою методу UCP (Use-Case Points), який базується на основних варіантів використання системи та складається з 5-ти етапів.

Етап I. Оцінка складності інтерфейсів (UAW)

UAW є фактором, який вносить свій внесок в розмірі розроблюваного програмного забезпечення. UAW розраховується на підставі кількості та складності акторів для системи. Кожен з учасників системи може бути визначений і класифікований по 3 категоріям: простий, середній або складний - комплекс на основі типу актора. Кожна класифікація також має визначений вага, призначений. UAW це сума ваг для кожного з учасників[4].

Оскільки в проекті фігурують один актор: Студент, то згідно з типом актора були виставлені вагові коефіцієнти, які зображені на таблиці 3.1.

Таблиця 3.1 – Визначення вагових коефіцієнтів

Актор	Тип	Вага	Обґрунтування
Адміністратор	Складний	3	

Підрахуємо по формулі складність інтерфейсів:

$$UAW = (1 * 3) = 3 \quad (3.1)$$

Етап II. Оцінка масштабу системи на основі варіантів використання (UUCW)

UUCW є одним з факторів, які вносять свій внесок в розмірі розроблюваного програмного забезпечення. Він розраховується на підставі кількості та складності прецедентів для системи Кожна класифікація має визначений вага призначений.

Тип варіанту використання визначено за кількістю транзакцій. Для підрахунку UUCW буде використовуватися Use-Case діаграма, зображений на рис. 2.1. По цій діаграмі буде визначено тип прецеденту і відповідна вага. На таблиці 3.2 визначені типи варіантів.

Таблиця 3.2 – Визначення типів прецедентів за кількістю транзакцій

Назва прецеденту	Вага	Кількість транзакцій
Авторизація	10	4
Додати друга	5	1
Пройти тест	5	1
Пройти урок	5	1
Створити тест	10	4
Стати суперкористувачем	5	1
Змінити автар	5	1
Відправити фото\документ	5	1
Переглянути список користувачів	10	4

Підрахувавши кількість транзакцій у кожному прецеденті, отримаємо такий результат: UUCW = $10 * 3 + 7 * 4 = 30 + 28 = 78$

Показник UCP обчислюється за формулою:

$$UCP = (UAW + UUCW)\left(\frac{100-\%REUSE}{100}\right) \qquad (3.2)$$

Підставимо наші значення у формулу:

$$UCP = (3 + 78)\left(\frac{100 - 20}{100}\right)$$

Отримали результат: UCP = 81 * 0,8 = 65.3

Етап 3. Оцінка складності архітектури технічних факторів (TCF)

Для використання методу TCF буде використана таблиця 3.3, де опишемо технічні фактори, їхню вагу та оцінку.

Таблиця 3.3 – Технічні фактори

Фактор	Опис	Вага	Оцінка	Обґрунтування
T1	Розподіленість системи	2	2	Існують розподілені системи у вигляді розподілених елементів бази даних.
T2	Час відгуку	1	4	Для користувача важливо, щоб час відгуку був мінімальним.
T3	Ефективність кінцевого користувача	1	3	Можливість досягнення мети за короткий час роботи системи.
T4	Внутрішня складність обробки	1	0	Буде використано близько п’яти складних алгоритмів
T5	Фокус на повторному використанні коду	1	3	Система має повторне використання.
T6	Простота інсталяції	0,5	2	Встановлення програми не повинна бути дуже складним.
T7	Простота використання	0,5	3	Зручний та простий інтерфейс, не мають складних елементів меню.
T8	Портативність	2	1	Доступний тільки для смартфонів операційної системи Android.
T9	Простота змін	1	2	Зручність в масштабуванні.
T10	Паралельні обчислення	1	0	Є несуттєвим елементом системи.

Продовження таблиці 3.3

T11	Засоби захисту	1	3	Потрібне постійне підключення інтернету. Дані мають ризик бути зламані третю стороною так як вони зберігаються на хмарне сховище бази даних.
T12	Доступ до третьої сторони	1	0	Відкрите API.
T13	Потреби в спеціальному навчанні	1	0	Простий в освоєнні і інтуїтивно зрозумілий для кожного користувача.

Показник TF(Technical factor), який є сумою добутків коефіцієнтів на оцінку: TF = 23,5.

Оцінка TCF обчислюється за формулою:

$$TCF = 0{,}6 + (0{,}01 * TF) \qquad (3.3)$$

Отримали результат: TCF = 0,6 + (0,01 * 23,5) = 0,295

Етап IV. Оцінка зовнішніх факторів (EF)

Для підрахунку оцінки при розробці програмного забезпечення, скористаємося таблицею 3.4, де є зовнішні фактори, їх вага та оцінки.

Таблиця 3.4 – Зовнішні фактори

Фактор	Опис	Вага	Оцінка	Обґрунтування
E1	Знайомство з процесом розробки	1,5	3	Вивчення матеріалів і предметної області для вирішення труднощів розробки і досягнення поставлених цілей.

Продовження таблиці 3.4

E2	Досвід подібних проектів	0,5	2	Затребуваний, не був би зайвим.
E3	Досвід об’єктно-орієнтованої розробки	1	3	Бажаний.
E4	Досвідченість провідного аналітика	0,5	0	Необов'язковий елемент.
E5	Мотивація	1	3	Необхідна і наявна.
E6	Стабільність вимог	2	4	Низька ймовірність зміни.
E7	Часткова зайнятість працівників	-1	1	Повна зайнятість працівників.
E8	Складність мови програмування	-1	4	Додаткове вивчення необхідно для вдосконалення роботи.

Показник EF(Environmental Factor), що складається з суми добутків коефіцієнтів на оцінку: EF = 14,5

Оцінка зовнішнього фактору ECF обчислюється за формулою:

$$ECF = 1{,}4 + (-0{,}03 * EF) \quad (3.4)$$

Отримали результат: ECF = 1,4 + (-0,03 * 14,5) = 0,965

Етап V. Загальна оцінка.

Скоректовані оцінка UCP обчислюється за формулою:

$$\text{AUCP} = \text{U}CP * TCF * ECF \quad (3.5)$$

Звідси, AUCP = 86,4 * 0,295 * 0,965= 24,6

3.2 Управління ризиками проекту

Управління ризиками проекту - це процес реагування на події та зміни ризиків в процесі виконання проекту. Важливим є проведення моніторингу ризиків. Моніторинг ризиків включає контроль ризиків протягом всього життєвого циклу проекту. Події, які виникають під час виконання проекту і мають певний ефект, називаються ризиками. Негативні ризики повинні бути обов'язково визначити і ретельно опрацьовані.

Для ідентифікації ризиків було використано метод «контрольних списків». Контрольні списки являють собою переліки ризиків, складені на основі інформації і знань, які були накопичені в ході виконання колишніх аналогічних проектів. При аналізі контрольних списків були виділені наступні ризики:

- Нестача часу.
- Зміни в технології розробки.
- Переробка всього проекту.
- Апаратні і програмні помилки.

Для аналізу причин виникнення цих ризиків була використана діаграма Парето. Діаграма дозволяє визначити першочергові завдання, які необхідно вирішити для усунення виниклої проблеми. Діаграма Парето була обрана в якості можливого варіанту виявлення причин проблеми через декілька причин:

- Простота використання;
- Швидкість розрахунку і відсутність необхідності використовувати будь-яке додаткове програмне забезпечення;
- Мобільність використання;
- Комінукація з клієнтом;
- Наочність отриманих даних.

Для побудова діаграми Парето були використані вхідні дані зазначені в таблиці 3.5.

Таблиця 3.5 – Вхідні дані з ризиками для побудови діаграми Парето.

Причини	Частка кожної причини в загальному результаті, %	Сумарний вплив на проект
Нестача часу	60%	60%
Апаратні і програмні помилки	20%	80%
Переробка всього проекту	13%	93%
Зміни в технології розробки	7%	100%
Всього	100%	

На рисунку 3.1 можемо бачити побудовану діаграму Парето з використанням вхідних даних ризиків.

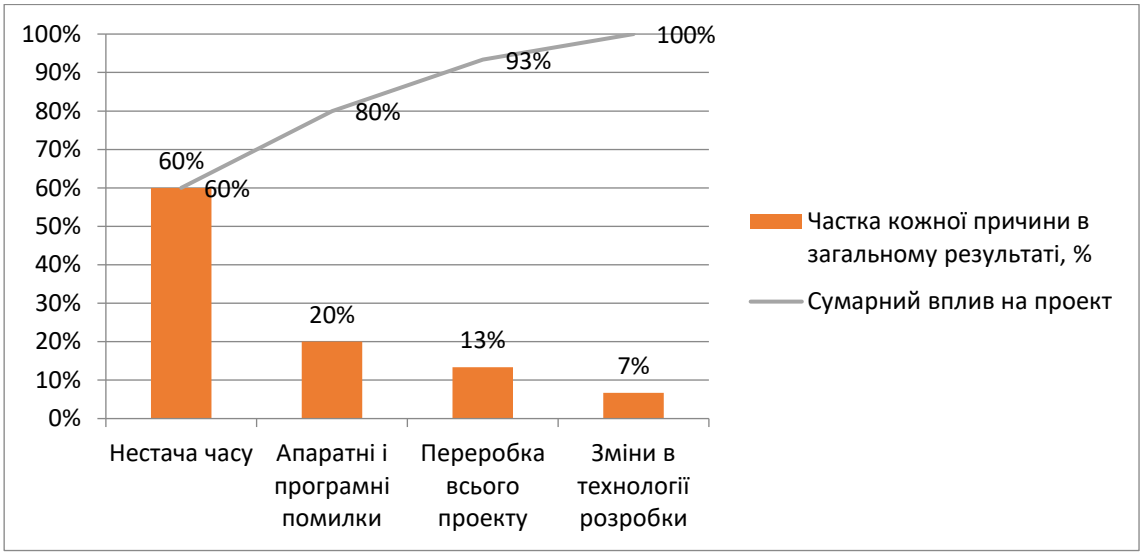

Рисунок 3.1 – Діаграма Парето з причин

Графік Парето показує вплив кожної причини і за законом 80/20 (сумарний вплив) можна виділити головні чинники, які вплинули на результат. У наведеному прикладі він викликаний 3 причинами:

- Нестача часу
- Апаратні і програмні помилки
- Переробка всього проекту

Після ідентифікації ризиків потрібно розробити план роботи з ними. Усього існує чотири типу роботи з ризиками:

1. Ухилення ризику
2. Передача ризику
3. Прийняття ризику
4. Зниження ризику

Під час виконання проекту будуть використані дві стратегії – передача та зниження, а саме:

1) щоб знизити ймовірність ризики «Нестача часу» рекомендується почати процес розробки раніше запланованого терміну, а завдання і плани розробників розподілити таким чином, щоб завдання не було б сфокусована на одному розробник;
2) щоб зменшити ризик «Апаратної і програмної помилки» необхідно більш детально ознайомитися з технікою і проводити профілактичні роботи з обладнанням на робочому місці, що стосується програмних помилок, то рекомендується частіше проводити аналітичну роботу з проектом і спостерігати успішність розробників;
3) щоб зменшити ймовірність ризику «часткової або повної переробки» необхідно постійно контактувати між групами планування і розробки для того, щоб при виявленні помилок заощадити перерозподіл часу на переробку.
4) щоб зменшити ймовірність ризику «зміни в технології розробки» необхідно мати можливість моніторингу змін інформації в галузях розробки, також потрібно постійно оновлювати власну базу знань новими матеріалами.

По методу «Картки Кроуфорда» були виявлені такі ризики:

а) нестача часу для завершення проекту;

б) відсутність техніки;

в) переробка великої частини проекту.

Для аналізу виникнення причин ризиків та наслідків була розроблена діаграма Ішікави. Завдяки даній діаграмі можна побачити основні причини появи негативних ризиків.

Щоб проаналізувати виникнення причин ризиків та причин була розроблена діаграма. Завдяки даному методу можна протестувати основні причини появи негативних ризиків.

Використовуючи діаграму Ішикави можна прослідкувати виникнення причин ризиків та наслідків. Завдяки даній діаграмі можна побачити основні причини появи негативних ризиків.

Існують три основні причини появи ризику «Нестача часу для завершення проекту»:

а) Освоєння нових технологій;

б) неправильне планування проекту;

в) відсутність тайм-менеджменту.

На рис. 3.2 зображена діаграма Ішікави для ризику «Нестача часу для завершення проекту»: шрифт

Рисунок 3.2 – Діаграма Ішікави для ризику «Нестача часу для завершення проекту»

На рис. 3.3 зображена діаграма Ішікави для ризику «Відсутність техніки»:

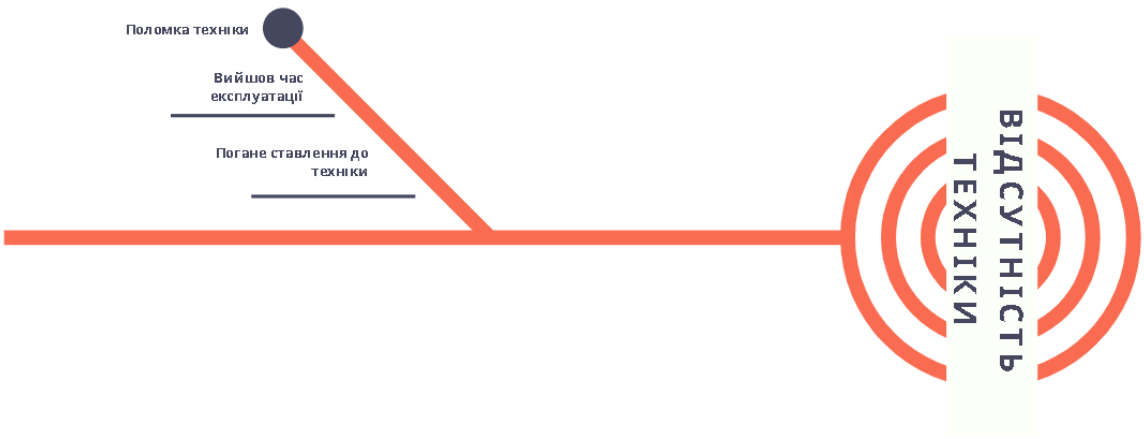

Рисунок 3.3 – Діаграма Ішікави для ризику «Відсутність техніки»

На діаграмі (рис. 3.3) видно, що єдиною причиною виникнення ризику «Відсутність техніки» - це її поломка.

На рис. 3.4 зображена діаграма Ішікави для ризику «Переробка великої частини проекту».

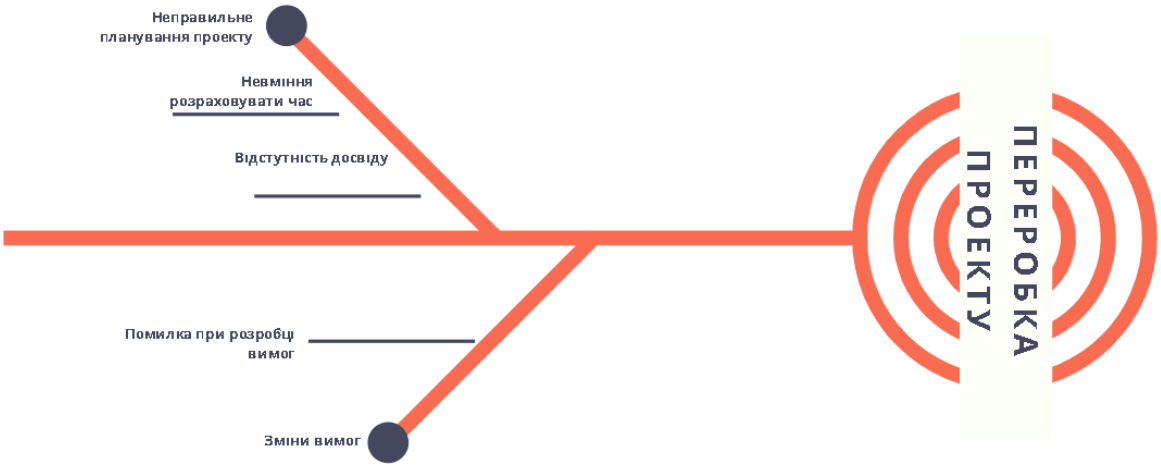

Рисунок 3.4 – Діаграма Ішікави для ризику «Переробка великої частини проекту»

На діаграмі (рис. 3.4) показано, що під час виконання проекту причинами виникнення ризику «Переробка великої частини проекту» можуть стати:

а) неправильне планування через відсутність досвіду у подібних проектах або невміння розраховувати час;

б) зміни вимог, так як були допущені помилки у розробці вимог та вони були пізно знайдені.

Після ідентифікації ризиків слід розробити план роботи з ними. Існують чотири основні стратегії роботи з ризиками:

а) ухилення від ризику;

б) передача ризику;

в) прийняття ризику;

г) зниження ризику.

Під час виконання мого проекту буде використана тільки одна стратегія – «зниження ризику», а саме:

а) для зменшення ймовірність виникнення ризику «нестача часу для завершення проекту» можна розпочати виконання проекту раніше та виділити проекту більше часу та більше потрати часу на вивчення технологій ;

б) для зменшення серйозність наслідків при виникненні другого ризику «відсутність техніки», потрібно обновити техніку та більше детальніше стежити за нею;

в) для збільшення надійності та відсутності наслідків при виникненні другого ризику «відсутність правильного використання», потрібно обновити техніку та більше детальніше стежити за нею.

3.3 План розробки

Стадії і етапи розробки, отримання робіт та терміни виконання показані в таблиці 3.6.

Таблиця 3.6 – Стадії та етапи розробки

№	Стадії розробки	Зміст робіт	Початок	Кінець
1	Аналіз розробки	Аналіз предметної області.	01.09.2021	07.02.2021
2	Пошук існуючих варіантів рішення	Перегляд аналогів предметної області, прийняття рішення про необхідність розробки.	07.09.2021	21.09.2021
3	Визначення вимог до ПЗ	Визначення вимог до програмного забезпечення.	22.09.2021	01.10.2021
4	Визначення технологій розробки	Прийняття рішень по вибору технологій і методів розробки.	02.10.2021	26.10.2021
5	Проектування	Моделювання, проектування діаграм, створення макетів інтерфейсів, структури баз даних.	26.10.2021	15.11.2021
6	Розробка прикладної програми	Моделювання, проектування діаграм, створення макетів інтерфейсів, структури баз даних.	26.10.2021	15.11.2021
7	Тестування	Тестування та налагодження системи.	15.11.2020	05.12.2021

Завдяки таблиці 3.6 ми можемо побачити весь цикл створення програмного продукту " *Мобільне застосування для вивчення англійської мови*.»

РОЗДІЛ 4 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ EasyLEng

Проектування[27] - це етап життєвого циклу розроблення програмних систем, наступний після інженерії вимог. Завданням цього етапу є перетворення побажань замовників системи, які ми подали як моделі вимог, у проектні рішення, що забезпечать здійснення згаданих побажань у формі відповідної системи програмування. Таким чином, під час проектування виконується трансформація простору вимог у простір проектних рішень. При цьому можна виділити процеси, котрі можна вважати відносно незалежними одне від одного і виконувати як послідовно, так і паралельно, окремими командами виконавців. Це такі процеси:

- концептуальне проектування полягає в уточненні розуміння й узгодження деталей вимог;

- архітектурне проектування полягає у визначенні головних структурних особливостей системи, яку будують;

- технічне проектування полягає у відображенні вимог середовища функціонування і розроблення системи та у визначенні всіх конструкцій як композицій компонент;

- функціональне проектування полягає у показі вимог середовища технічної системи і розроблення проекту у визначенні всіх конструкцій як компонентних застосувань.

4.1 Діаграма програмних класів

Після сформування архітектури програмної системи перейдемо до формування її структури у вигляді програмних класів, їх атрибутів та методів. Для цього була створена діаграма програмних класів (рис. 4.1).

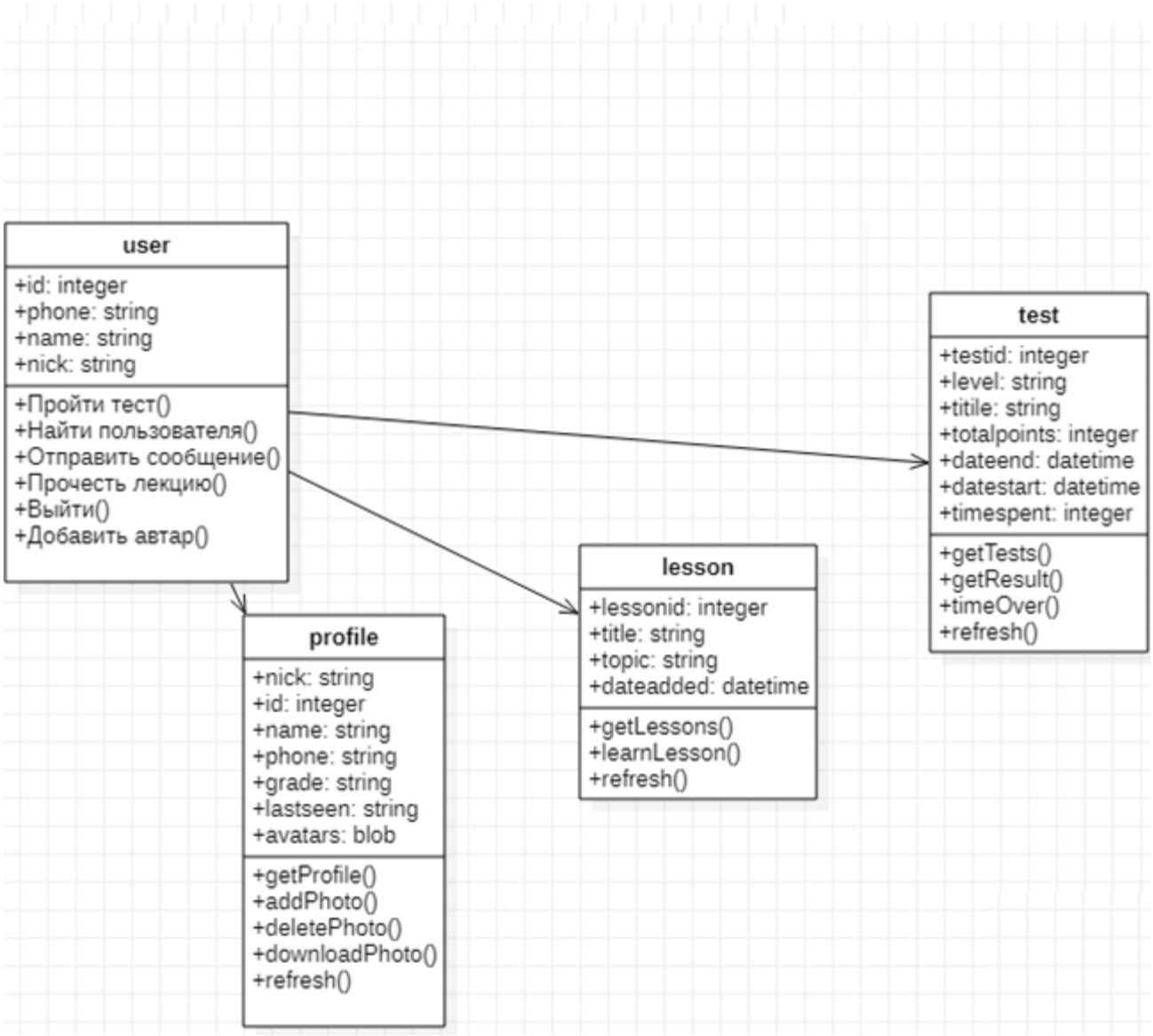

Рисунок 4.1 – Діаграма програмних класів

Клас User представляє собою сутність користувача. Він складається з чотирьох атрибутів: ім’я користувача, нік користувача, номер телфону та унікальний ідентифікатор. До цього класу входять шість методів які дають можливість написати повідомлення, пройти тест, пройти урок, авторизуватись, переглянути профіль, додати/оновити аватар.

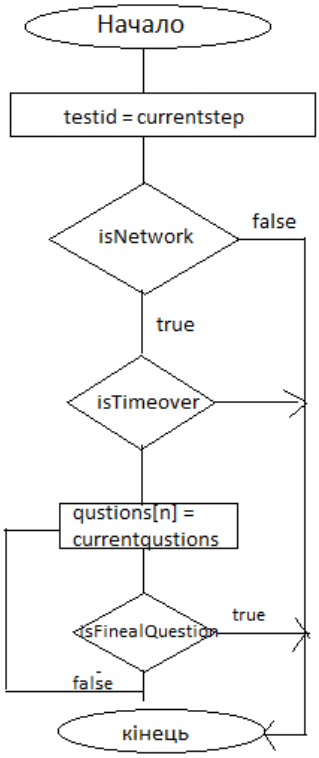

Рисунок 4.2 – Алгоритм методу Пройти тест

Клас Test представляє собою сутність тесту. Він складається з семи атрибутів: унікальний ідентифікатор тесту, рівень складності тесту, витрачений час, назва тесту, дата початку, дата кінця, максимальна кількість можливих балів. Алгоритм тесту (рис 4.2.).

До цього класу входять чотири методи getTests – отримати тест, getResult – отримати результат тесту, timeOver – час на проходження тесту закінчився, refresh – оновити список тестів.

Клас Profile представляє собою сутність профілю користувачів. Він складається з семи атрибутів: унікальний ідентифікатор профілю, нік, рівень володіння англійською, ім’я користувача, номер телефону, останній час активності та масив з фотографій аватарів. До цього класу входять п’ять методів getProfile – отримати дані профілю користувача, addPhoto – додати нове фото в профіль, deletePhoto - видалити обране фото з профілю, downloadPhoto – завантажити фото на локальний пристрій та refresh – оновити дані профілю.

Клас Lesson представляє собою сутність уроку. Він складається з чотирьох атрибутів таких як: lessonid – унікальний ідентифікатор уроку, title – назва уроку,

topic – тема уроку та dataadded – дата створення уроку. До цього класу входять три методи getLessons – отримати список доступних уроків, learnLesson – пройти урок та refresh – оновити дані про уроки.

4.2 Діаграма послідовності

Щоб продемонструвати взаємодію об’єктів для прикладу використаємо діаграму послідовності. Діаграми послідовностей використовуються для уточнення діаграм прецедентів, більш детального опису логіки сценаріїв використання. Це відмінний засіб документування проекту з точки зору сценаріїв використання.

Діаграми послідовностей зазвичай містять об'єкти, які взаємодіють в рамках сценарію, повідомлення, якими вони обмінюються, і які повертаються результати, пов'язані з повідомленнями. Втім, часто повертаються результати позначають лише в тому випадку, якщо це не очевидно з контексту.

Об'єкти позначаються прямокутниками з підкресленими іменами (щоб відрізнити їх від класів).

Повідомлення (виклики методів) - лініями зі стрілками.

Повертаються результати - пунктирними лініями зі стрілками.

результати - пунктирними лініями з стрілками.

Представимо діаграму послідовності для варіанту використання «Авторизація» (рис. 4.3).

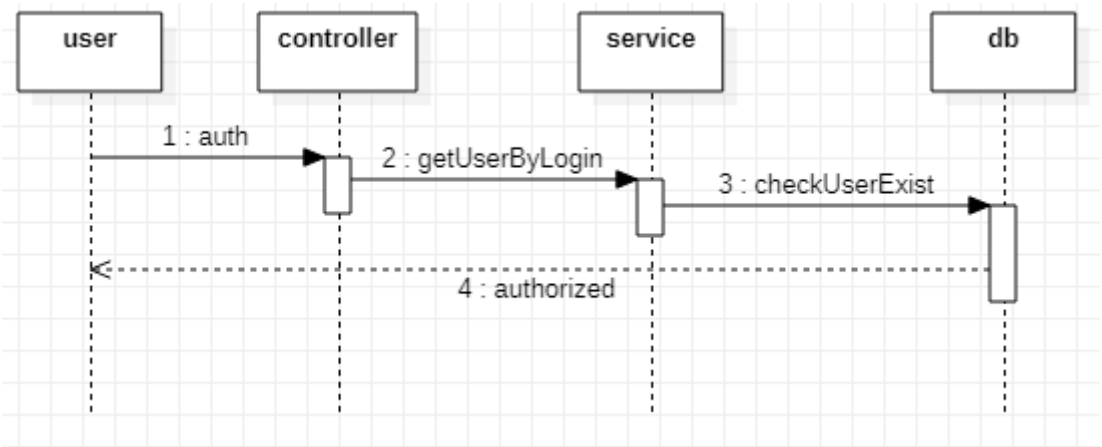

Рисунок 4.3 – Діаграма послідовності для варіанту використання «Авторизація»

Об’єкт User відправляє об’єкту Controller номер телефона, за допомогою метода login(). Controller після певної обробки даних, відправляє кінцеві дані до Service. Service повинен виконати пошук конкретного користувача за номером телефону користувача, проте для цього, об’єкту Service необхідно звернутися до бази даних DB, щоб отримати користувача, після отримання якого, буде виконаний безпосередньо сам пошук.

Представимо діаграму послідовності для варіанту використання «Відправка та відображення результату тесту» (рис. 4.4).

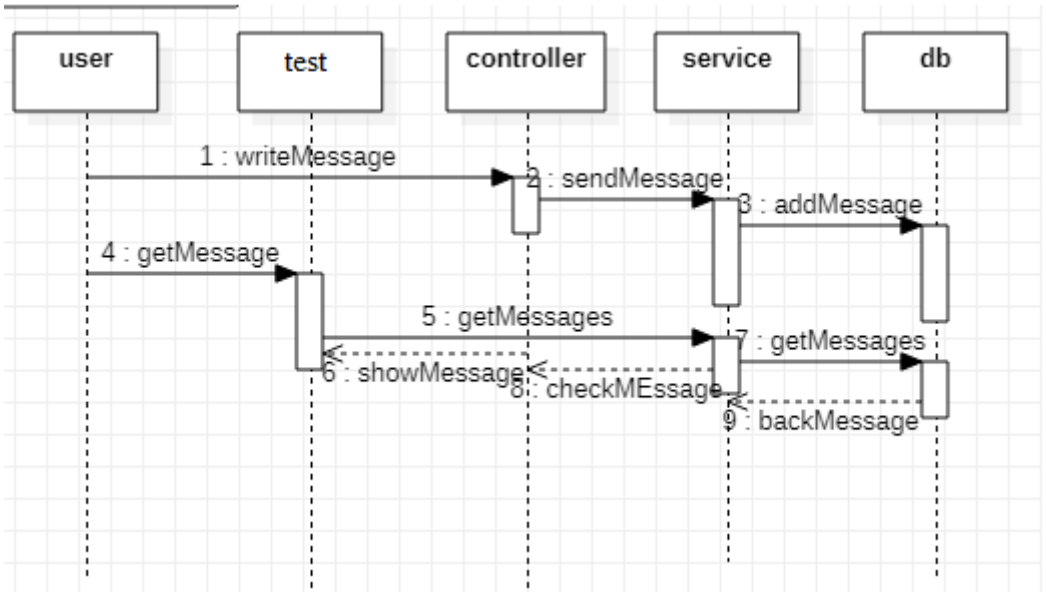

Рисунок 4.4 – Діаграма послідовності для варіанту використання «Відправка та відображення результату тесту»

Об’єкт User відправляє об’єкту Controller тест, за допомогою метода sendTest. Controller після певної обробки даних, відправляє кінцеві дані до Service. Service повинен занести ці данні до бази, проте для цього, об’єкту Service необхідно звернутися до бази даних DB, щоб занести їх. Якщо об’єкт User звертався до об’єкта Test, то йому відобразяться його результаті тесту та в подальшому користувач матиме можливість перепроходити тест.

4.3 Зберігання даних

Для реалізації бази даних для програми був обраний сервіс Firebase компанії Google. Основний сервіс Firebase - хмарна СУБД класу NoSQL, що дозволяє розробникам додатків зберігати і синхронізувати дані між декількома клієнтами. Підтримані особливості інтеграції з додатками під операційні системи Android і iOS, реалізовано API для додатків на JavaScript, Java, Objective-C і Node.js, також можливо працювати безпосередньо з базою даних в стилі REST з ряду JavaScript-фреймворків, включаючи AngularJS, React, Vue.js, Ember.js і Backbone.js. Передбачено API для шифрування даних.[5]

Хмарний сервіс Firebase був обраний через його особливості:

- Realtime оновлення;
- підтримка оффлайн режиму;
- гнучкість платформи;
- передбачено API для шифрування даних.

Всі дані в Firestore зберігаються в колекціях в NOSQL форматі.

Перша колекція з якою ми зустрічаємося це Users вона представляє собою сутність Користувач, яка використовується для того що би зберігати всю необхідну інформацію користувача, щоб надалі він міг використовувати при вході в систему (табл. 4.1).

Таблиця 4.1 – Опис колекції Users

Назва	Тип	Пояснення
user_id	integer	Унікальний ідентифікатор користувача
nick	string	Логін, що ідентифікує користувача
phone	string	Номер телефону користувача
name	string	ФІО користувача

Продовження таблиці 4.1

level	string	Рівень потрібний для проходження уроку
lesson	string	Зміст уроку
date	integer	Дата створення уроку в мілісекундах

Колекція Tests представляє сутність тестів, що містить в собі інформацію про питання тестів та відповіді на них (табл. 4.2).

Таблиця 4.2 – Опис колекції Tests

Назва	Тип	Пояснення
Test_id	integer	Унікальний ідентифікатор тесту
questions	array	Масив запитань в даного тесту
mark	integer	Максимально можлива кількість балів за тест
title	string	Тема тесту
answers	array	Масив відповідей на тест

Колекція Lessons представляє сутність уроків, яке зберігає інформацію про тему уроків та його зміст (табл. 4.3).

Таблиця 4.3 – Опис колекції Lessons

Назва	Тип	Пояснення
lessonsid	integer	Унікальний ідентифікатор уроку
topic	string	Тема уроку
title	string	Назва уроку

Продовження таблиці 4.3

level	string	Рівень потрібний для проходження уроку
lesson	string	Зміст уроку
date	integer	Дата створення уроку в мілісекундах

У даному підрозділі було показано чотири головні колекції, які відображають сутності системи, також в системі є допоміжні таблиці, в яких зберігається інформація про підручники та необхідні довідки для навчання.

РОЗДІЛ 5 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

5.1 Вибір програмних засобів

Після проектування необхідно уважно вибрати технологію, яка буде використовуватися при реалізації програмного продукту. Якщо вибрати технологію правильно, то це допоможе нам збільшити впевненість та працездатність проекту.

Реалізація дипломної роботи проводиться у середовищі програмування xCode та Android Studio, що має в своєму розпорядженні широкі можливості по створенню програм для смартфонів.

Середовище розробки адаптоване для виконання типових завдань, що вирішуються в процесі розробки застосунків для платформи Android. У тому числі у середовище включені засоби для спрощення тестування програм на сумісність з різними версіями платформи та інструменти для проектування застосунків, що працюють на пристроях з екранами різної роздільності (планшети, смартфони, ноутбуки, годинники, окуляри тощо). В Android Studio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання застосунків. Для прискорення розробки застосунків представлена колекція типових елементів інтерфейсу і візуальний редактор для їхнього компонування, що надає зручний попередній перегляд різних станів інтерфейсу за стосунку.

Спираючись на велику комплексну роботу, вибір був зроблений на користь React Native через те, що вона є мовою розподіленого програмування зі зручним інструментарієм. Вона чудово підходить для великих програмних забезпечень, оскільки React Native є сумісною з усіма попередніми версіями, а розробка програмного забезпечення може тривати декілька років.

Що таке React Native?

React Native — це фреймворк JavaScript для написання реальних мобільних додатків для iOS та Android. Він заснований на React, бібліотеці JavaScript від

Facebook для створення користувацьких інтерфейсів, але замість того, щоб орієнтуватися на браузер, він націлений на мобільні платформи. Іншими словами: веб-розробники тепер можуть писати мобільні додатки, які виглядають і відчувають себе справді «рідними», не вдаючись до бібліотеки JavaScript, яку ми вже знаємо і любимо. Крім того, оскільки більшість коду, який ви пишете, можна спільно використовувати між платформами, React Native полегшує одночасну розробку як для Android, так і для iOS.

Подібно до React для Інтернету, програми React Native написані з використанням суміші розмітки JavaScript і XML, відомої як JSX. Потім, під капотом, React Native «міст» викликає нативні API візуалізації в Objective-C (для iOS) або Java (для Android). Таким чином, ваша програма буде відображатися за допомогою реальних компонентів мобільного інтерфейсу, а не веб-переглядів, і буде виглядати і відчувати себе як будь-який інший мобільний додаток. React Native також надає інтерфейси JavaScript для API платформ, тому ваші програми React Native можуть отримати доступ до функцій платформи, таких як камера телефону або місцезнаходження користувача.

React Native наразі підтримує як iOS, так і Android, а також має потенціал для розширення на майбутні платформи. У цій книзі ми розглянемо як iOS, так і Android. Переважна більшість коду, який ми пишемо, буде кросплатформним. І так: ви дійсно можете використовувати React Native для створення готових для виробництва мобільних додатків! Деякі анекдоти: Facebook, Palantir і TaskRabbit вже використовують його у виробництві для додатків, призначених для користувачів.

Переваги React Native

Той факт, що React Native фактично виконує візуалізацію за допомогою стандартних API візуалізації своєї хост-платформи, дозволяє йому виділитися з більшості існуючих методів розробки кросплатформних додатків, таких як Cordova або Ionic. Існуючі методи написання мобільних додатків із використанням комбінацій JavaScript, HTML і CSS зазвичай відображаються за допомогою веб-

переглядів. Хоча цей підхід може працювати, він також має недоліки, особливо щодо продуктивності. Крім того, вони зазвичай не мають доступу до набору власних елементів інтерфейсу користувача хост-платформи. Коли ці фреймворки намагаються імітувати нативні елементи інтерфейсу користувача, результати зазвичай «відчуваються» лише трохи; Зворотна інженерія всіх дрібних деталей таких речей, як анімація, вимагає величезних зусиль, і вони можуть швидко застаріти.

На відміну від цього, React Native фактично переводить вашу розмітку в реальні, рідні елементи інтерфейсу користувача, використовуючи наявні засоби відтворення представлень на будь-якій платформі, з якою ви працюєте. Крім того, React працює окремо від основного потоку інтерфейсу користувача, тому ваша програма може підтримувати високу продуктивність без шкоди для можливостей. Цикл оновлення в React Native такий самий, як і в React: коли параметри або стан змінюються, React Native повторно відтворює представлення. Основна відмінність між React Native і React у браузері полягає в тому, що React Native робить це, використовуючи бібліотеки інтерфейсу користувача своєї хост-платформи, а не HTML і CSS розмітки.

Для розробників, які звикли працювати в Інтернеті за допомогою React, це означає, що ви можете писати мобільні програми з продуктивністю та зовнішнім виглядом рідної програми, використовуючи знайомі інструменти. React Native також є покращенням у порівнянні зі звичайною мобільною розробкою у двох інших областях: досвід розробника та потенціал кросплатформного розвитку.

Досвід розробника

Якщо ви коли-небудь розробляли для мобільних пристроїв раніше, ви можете бути здивовані, наскільки легко працювати з React Native. Команда React Native вклала у фреймворк потужні інструменти розробника та значущі повідомлення про помилки, тому робота з надійними інструментами є природною частиною вашого досвіду розробки.

Наприклад, оскільки React Native — це «просто» JavaScript, вам не потрібно перебудовувати свою програму, щоб побачити ваші зміни; замість цього ви можете натиснути Command+R, щоб оновити програму так само, як і будь-яку іншу веб-сторінку. Усі ці хвилини, витрачені на очікування створення вашої програми, можуть дійсно додатися, і навпаки, швидкий цикл ітерацій React Native виглядає як знахідка.

Крім того, React Native дозволяє скористатися перевагами інтелектуальних інструментів налагодження та звітів про помилки. Якщо вам подобається інструменти розробника Chrome або Safari (рис 1.1.), ви будете раді знати, що ви також можете використовувати їх для мобільної розробки. Аналогічно, ви можете використовувати будь-який текстовий редактор для редагування JavaScript: React Native не змушує вас працювати в Xcode для розробки для iOS або Android Studio для Android.

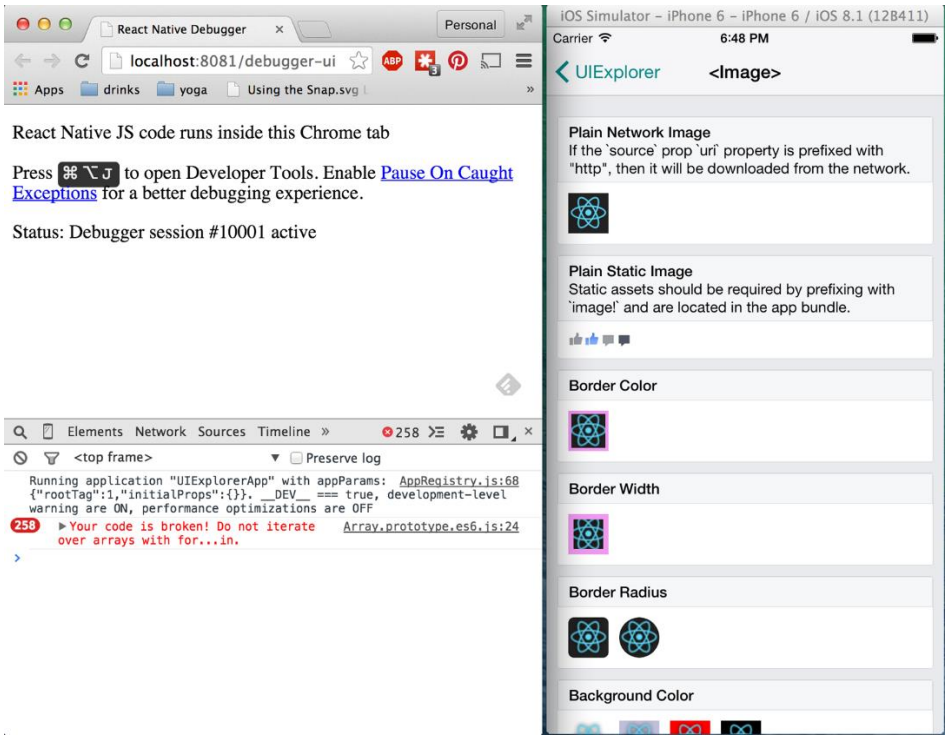

Рисунок 5.1 – «Використання налагоджувача Chrome»

Окрім щоденного покращення вашого досвіду розробки, React Native також може позитивно вплинути на цикл випуску продукту. Наприклад, Apple дозволяє

завантажувати зміни поведінки програми на основі JavaScript без необхідності додаткового циклу перевірки.

Усі ці невеликі переваги заощаджують вам і вашим колегам-розробникам час і енергію, дозволяючи зосередитися на більш цікавих частинах вашої роботи та бути більш продуктивними в цілому.

Повторне використання коду та обмін знаннями

Робота з React Native може різко скоротити ресурси, необхідні для створення мобільних додатків. Будь-який розробник, який знає, як писати код React, тепер може орієнтуватися на Інтернет, iOS та Android, і всі вони мають однакові навички. Усуваючи необхідність «розшарування» розробників на основі їх цільової платформи, React Native дозволяє вашій команді виконувати ітерації швидше та ефективніше ділитися знаннями та ресурсами.

Крім обміну знаннями, багатою частиною вашого коду також можна поділитися. Не весь код, який ви пишете, буде кросплатформним, і залежно від того, яка функціональність вам потрібна на конкретній платформі, іноді вам може знадобитися зануритися в Objective-C або Java. (На щастя, це не так вже й погано, і ми розглянемо, як працюють так звані нативні модулі, у розділі 7.) Але повторно використовувати код на різних платформах напрочуд легко з React Native. Наприклад, додаток Facebook Ads Manager для Android ділиться 87% своєї кодової бази з версією iOS, як зазначено в основній доповіді React Europe 2015. Останній додаток, який ми розглянемо в цій книзі, програма флеш-картки, має повне повторне використання коду між Android та iOS. Це важко перемогти!

Ризики та недоліки

Як і в будь-якому випадку, використання React Native не позбавлене недоліків, і чи підходить React Native для вашої команди, залежить від вашої індивідуальної ситуації.

Найбільшим ризиком, ймовірно, є зрілість React Native, оскільки проект ще відносно молодий. Підтримка iOS була випущена в березні 2015 року, а підтримка Android була випущена у вересні 2015 року. Документація, безумовно, має місце для вдосконалення, і вона продовжує розвиватися. Деякі функції на iOS і Android досі не підтримуються, і спільнота все ще знаходить найкращі методи. Хороша новина полягає в тому, що в переважній більшості випадків ви можете самостійно запровадити підтримку відсутніх API.

Оскільки React Native запроваджує ще один рівень у ваш проект, це також може зробити налагодження більш волосистим, особливо на перетині React і хост-платформи.

React Native ще молодий, і тут застосовуються звичайні застереження, пов’язані з роботою з новими технологіями. Але в цілому, я думаю, ви побачите, що переваги переважають ризики[25].

TypeScript

TypeScript — це мова програмування, розроблена та підтримувана Microsoft. Це суворий синтаксичний наднабір JavaScript і додає до мови необов'язковий статичний введення. TypeScript розроблено для розробки великих додатків і транскомпілює на JavaScript. Оскільки TypeScript є надмножиною JavaScript, існуючі програми JavaScript також є дійсними програмами TypeScript.

TypeScript можна використовувати для розробки програм JavaScript для виконання як на стороні клієнта, так і на стороні сервера (як з Node.js або Deno). Існує кілька варіантів транскомпіляції. Можна використовувати або перевірку TypeScript за замовчуванням, або компілятор Babel для перетворення TypeScript на JavaScript.

TypeScript підтримує файли визначення, які можуть містити інформацію про типи існуючих бібліотек JavaScript, подібно до того, як файли заголовків C++ можуть описувати структуру існуючих об’єктних файлів. Це дозволяє іншим програмам використовувати значення, визначені у файлах, так, ніби вони були статично типізованими об’єктами TypeScript. Існують сторонні заголовні файли

для популярних бібліотек, таких як jQuery, MongoDB і D3.js. Також доступні заголовки TypeScript для базових модулів Node.js, що дозволяє розробляти програми Node.js у TypeScript.

Сам компілятор TypeScript написаний на TypeScript і скомпільований на JavaScript. Він ліцензований за ліцензією Apache 2.0. TypeScript включено як першокласну мову програмування в Microsoft Visual Studio 2013 Update 2 і новіших версій, поряд із C# та іншими мовами Microsoft. Офіційне розширення дозволяє Visual Studio 2012 також підтримувати TypeScript. Андерс Хейлсберг, провідний архітектор C# і творець Delphi і Turbo Pascal, працював над розробкою TypeScript.

5.2 Архітектура програмної системи

Модель клієнт-сервер, або архітектура клієнт-сервер, являє собою розподілену програмну базу, яка розподіляє завдання між серверами і клієнтами, які або знаходяться в одній системі, або спілкуються через комп’ютерну мережу або Інтернет. Клієнт покладається на відправку запиту до іншої програми, щоб отримати доступ до служби, доступної сервером. Сервер запускає одну або кілька програм, які спільно використовують ресурси та розподіляють роботу між клієнтами.

Відношення клієнт-сервер спілкуються за схемою обміну повідомленнями запит-відповідь і повинні дотримуватися загального протоколу зв’язку, який формально визначає правила, мову та шаблони діалогів, які будуть використовуватися. Зв'язок клієнт-сервер зазвичай відповідає набору протоколів TCP/IP.

Протокол TCP підтримує з'єднання, поки клієнт і сервер не завершить обмін повідомленнями. Протокол TCP визначає найкращий спосіб розподілу даних програми на пакети, які можуть доставити мережі, передає пакети в мережу та отримує пакети з мережі, а також керує контролем потоків і повторною передачею

відкинутих або спотворених пакетів. IP - це протокол без з'єднання, в якому кожен пакет, що проходить через Інтернет, є незалежною одиницею даних, не пов'язаною з будь-якими іншими одиницями даних.

Клієнтські запити організовані та визначені пріоритетами в системі планування, яка допомагає серверам справлятися з отриманням запитів від багатьох різних клієнтів за короткий проміжок часу. Підхід клієнт-сервер дозволяє будь-якому комп’ютеру загального призначення розширити свої можливості, використовуючи спільні ресурси інших хостів. Популярні програми клієнт-сервер включають електронну пошту, всесвітню павутину та мережевий друк[26].

Для початку почнемо зі створення архітектури – сукупності рішень щодо організації програмної системи, а саме вибір структурних елементів, їх інтерфейсів та способів взаємодії між ними.

Отже мною була вибрана клієнт-серверна архітектура. Клієнт-серверна архітектура – обчислювальна або мережева архітектура, в якій завдання або мережева навантаження розподілені між постачальниками послуг, званими серверами, і замовниками послуг, званими клієнтами. Вибрана модель на даний час є найбільш розповсюдженою через ряд переваг, таких як:

а) Відсутність дублювання коду програми-сервера програмами-клієнтами.

б) Так як всі обчислення виконуються на сервері, то вимоги до комп'ютерів, на яких встановлено клієнт, знижуються.

в) Всі дані зберігаються на сервері, який, як правило, захищений набагато краще за більшість клієнтів. На сервері простіше організувати контроль повноважень, щоб вирішувати доступ до даних тільки клієнтам з відповідними правами доступу.

Також клієнт-серверна архітектура має свої недоліки:

а) Непрацездатність сервера може зробити непрацездатною всю обчислювальну мережу. Непрацездатним сервером слід вважати сервер, продуктивності якого не вистачає на обслуговування всіх клієнтів, а також сервер, що знаходиться на ремонті, профілактиці і т. п.

б) Підтримка роботи даної системи вимагає окремого фахівця - системного адміністратора.

в) Висока вартість обладнання.

Клієнт-серверна архітектура може бути двох видів: двох ланкова и трьох ланкова. Різниця між ними в тому, що в двох ланковій моделі взаємодія тільки між клієнтом і сервером, а в трьох ланковій з'являється ще один зв'язок між сервером і базою даних.

Дані користувача було вирішено зберігати у базі даних, для більш досконалої безпеки.

В даному проєкті у вигляді серверу та бази даних виступає Firebase.

Архітектура програмної системи представлена на рис. 5.2.

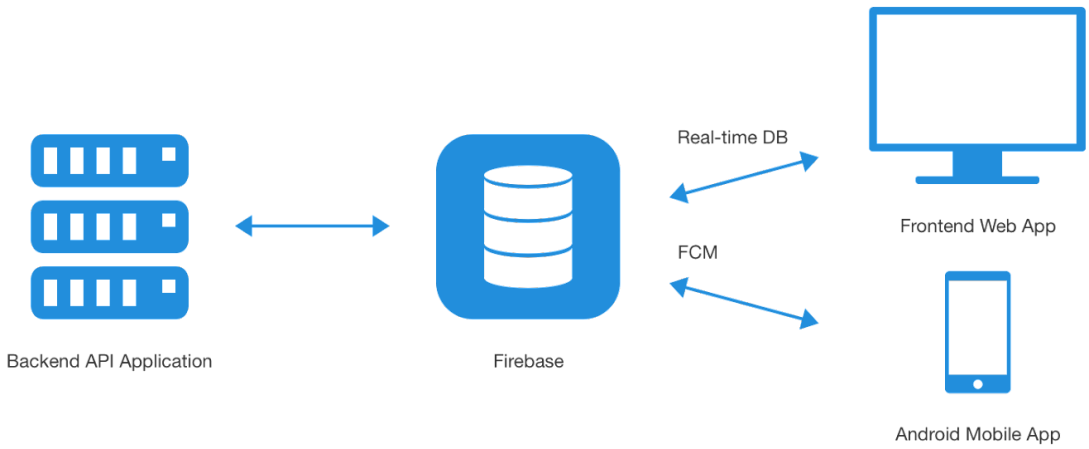

Рисунок 5.2 – Архітектура бази даних

РОЗДІЛ 6 ПОСІБНИК КОРИСТУВАЧА

6.1 Встановлення та системні вимоги

Для встановлення мобільного засобу на операційну систему Android вам потрібно перейти в PlayMarket та в пошуку знайти **EasyLEng** та завантажити його. Для встановлення мобільного засобу на операційну систему IOS вам потрібно перейти в AppStore та в пошуку знайти **EasyLEng** та завантажити його.

Для забезпечення працездатності програми необхідне виконання наступних мінімальних вимог до технічних параметрів комп'ютера:

- Android OS (API > 16) IOS (10.4)
- Оперативна пам'ять 1 ГБ (мінімум), 2 ГБ (рекомендується)
- Вільне місце на диску 68 МБ мінімум;

6.2 Основні режими та пункти меню

Програмна система вивчення англійської мови складається з декількох екранів: авторизація, головний, профіль, тести та налаштування. Кожному з екранів відведена конкретна задача.

Екран авторизації користувача є початковим, після проходження екрану авторизації користувачеві відкривається доступ до його профіля, завдань, налаштування і системи пошуку для вивчення англійської мови. Кожен користувач проходить авторизацію на своєму клієнті, щоб отримати доступ до основного функціоналу мобільного застосування. Під час авторизації користувачу потрібно заповнити необхідні поля. Перше – номер мобільного телефону. Друге – згода з політикою використання мобільного застосування для вивчення англійської мови.

Детальніше про екран авторизації можна спостерігати на (рис 6.1.).

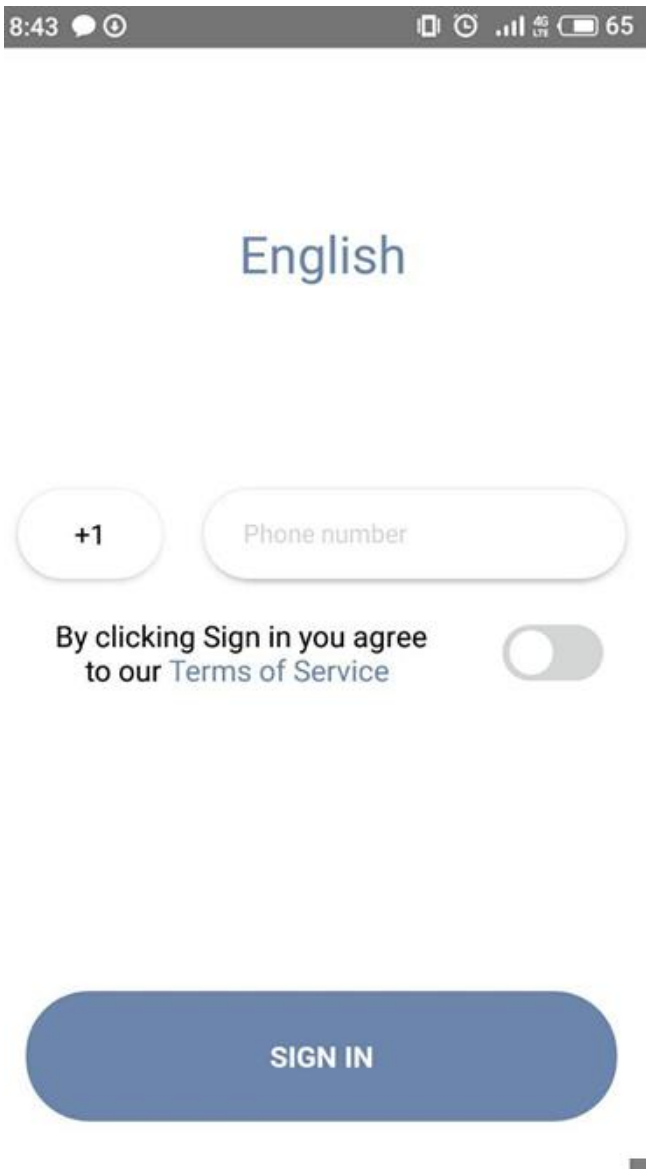

Рисунок 6.1 – Екран авторизації

Після успішного підтвердження авторизації клієнт з’єднується з сервером у разі, якщо користувача було знайдено в базі, мобільне застосування його відправляє на головний екран, де він має можливість використовувати весь фунціонал. Після успішної першої авторизації данні про вхід зберігаються на телефоні в спеціальному локальному сховищі від Firebase. Firebase Auth сам оперує стейтом авторизії користувача та його сесії.

Коли юзер наступний раз відкриє застосунок, то він одразу буде занавігований до головного екрану застосування.

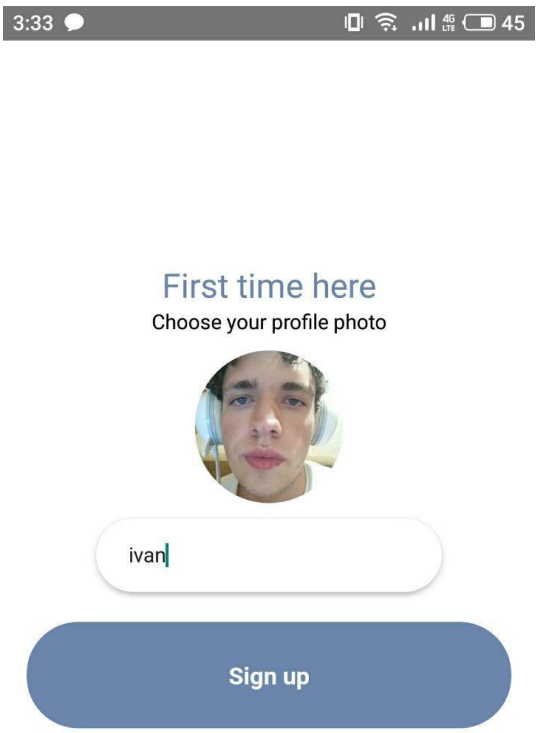

Рисунок 6.2 – Перший вхід

При першій авторизації користувачу потрібно заповнити додаткові персональні дані, а саме вести бажаний нікнейм, який буде відображатися для інших користувачів системи та буде мати глобальну видимість. Нікнейм потрібен бути унікальним. Саме по ньому ми можемо ідентифікувати користувача в системі.

Наступним кроком користувачу потрібно завантажити свою аватарку(власну фотографію). Це є обов'язковим кроком для успішної регестрації в системі. Якщо користувач не завантажить фотографію то він не зможе зареєструватись в системі, а як наслідок користуватись мобільним застосуванням для вивчення іноземної мови. Далі користувачу потрібно натиснути на кнопку зареєструватись. Після цього мобільне застосування виконає запит на сервер, в нашому випадку це Firebase DataBase і якщо ми отримає успішну відповідь то користувача буде перенесено на головний екран(профіль).

Після введення ніку та фотографії, користувач потрапляє на головний екран, де відображаються його персональні данні (див. рис. 6.3). З головного екрану ми можемо перейти до списку всіх користувачів, до тестів. На головному мобільного застосування, а саме екрану під назвою Профіль користувач має можливість побачити свої персональні данні які він ввів під час реєстрації в системі. По-перше це головна фотографія профілю, яка займає майже увесь екран. Друге – це дата останнього входу в систему(останньої активності користувача). Датою останньої активності може бути:

- останній пройдений тест;
- останнє редагування профілю;
- останній раз коли користувач натискав на будь яку кнопку в застосуванні.

Третє – це поле нікнейму, який юзер вказав під час реєстрації. Далі користувач має можливість переглянути декілька додаткових полів. Це номер мобільного телефону, за допомогою якого юзер зареєструвався в системі. Та рівень володіння англійською мовою. Значенням за умовчуванням є початковий рівень(Begginer).

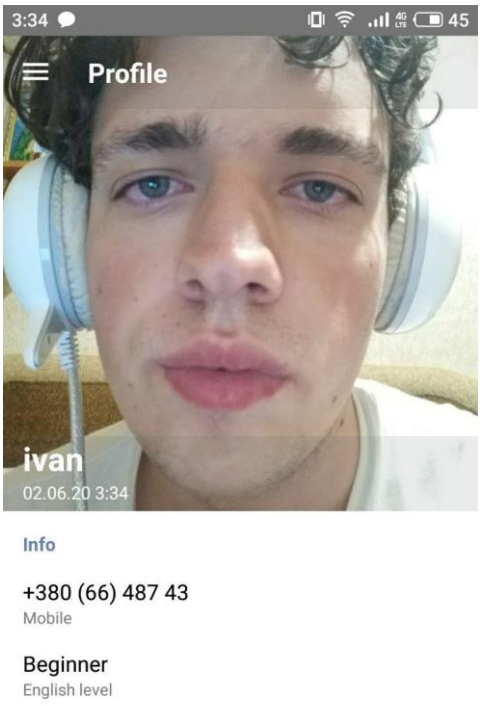

Рисунок 6.3 –Профіль

Натиснувши на кнопку у верхньому лівому куті ми можемо вивести бокове меню(«сайд-меню») (див. рис. 6.4). Також другим способом для переходу у бокове меню є проведення пальцем зліва направо(«свайп вправо»). В боковому меню користувач має можливість використовувати глобальну навігацію по всьому мобільному застосуванню для вивчення англійської мови.

Перший пунктом є екран Профіль(«Profile») – головний екран застосування, куди користувач попадає після входу в застосування.

Другий пункт – це екран для проходження тестів з англійської мови(«English»), де користувач має можливість проходити тести з англійської мови та покращувати свій рівень володіння та свої навички.

Третій пункт – це екран всіх користувачів системи(«Users»), де користувач може побачити весь список усіх зареєстрованих користувачів та їх рівень володіння англійською мовою.

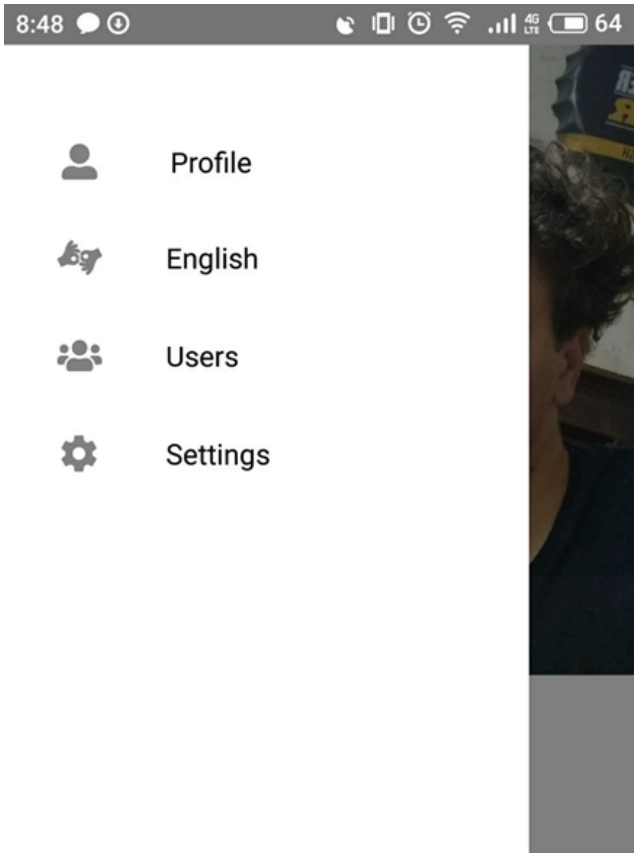

Рисунок 6.4 – Бокове меню

Натиснувши на пункт всіх користувачів («Users») юзер матиме можливість побачити список усіх користувачів(рис 6.5) . Також додатковою функцією на цьому екрані є можливість пошуку та фільтрації по всім користувачам. Користувачу потрібно ввести нікнейм бажаного юзера та мобільне застосування почне шукати потрібного користувача в системі. Також у користувача є можливість побачити рівень володіння англійською мовою всіх користувачів(рівень володіння англійською мовою на цьому екрані відображається справа). Наступною особливістю даного екрану є те що користувач має можливість побачити дату та час останньої активності інших користувачів в системі(дата відображається під нікнеймом). Також користувач може натиснути на будь-якого іншого користувача щоб перейти на екран профілю обраного юзера, де він матиме можливість більш детально ознайомитись з даними про обраного юзера та відкрити його фотографію.

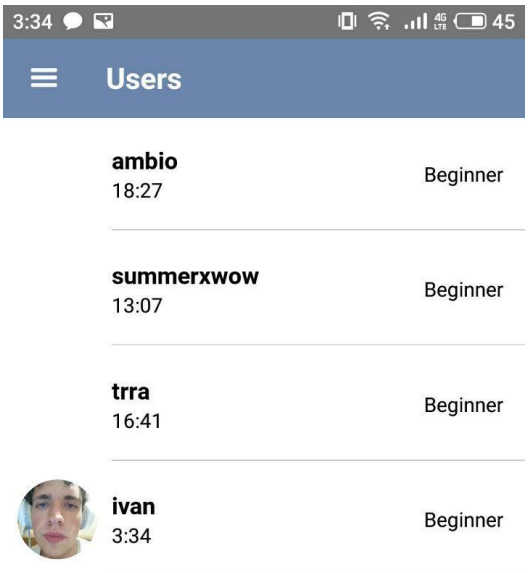

Рисунок 6.5 – Користувачі

Натиснувши на пункт проходження тестів з англійської мови («English») користувач перейде на екран тестів. Йому потрібно буде обрати тест який він хоче пройти за складністю. Щойно зареєстрованим користувачам надається можливість проходити тести тільки початкового рівня. Обравши необхідний тест користувач перейде на екран обраного тесту(рис 6.6.), де йому буде надано 300 секунд(5 хвилин) щоб пройти даний тест. Запитання будуть надаватися в випадковій послідовності один за одним. Користувач матиме три варіанти для відповіді та можливість натиснути на один з них. Після натискання на один з варіантів користувач ще матиме можливість переобрати варіант відповіді(це зроблено задля уникнення випадкового натискання на неправильний варіант). Щоб підтвердити свій вибір користувачу потрібно натиснути на кнопку наступне запитання («Next question») після чого користувачу прийде наступне запитання в довільному порядку. Варіанти відповідей також надаються в випадковому порядку(щоб запобігти однакових відповідей у кожному запитанні). Після завершення проходження тесту користувачу відкриється тимчасовим поп-ап з результатами проходження тесту.

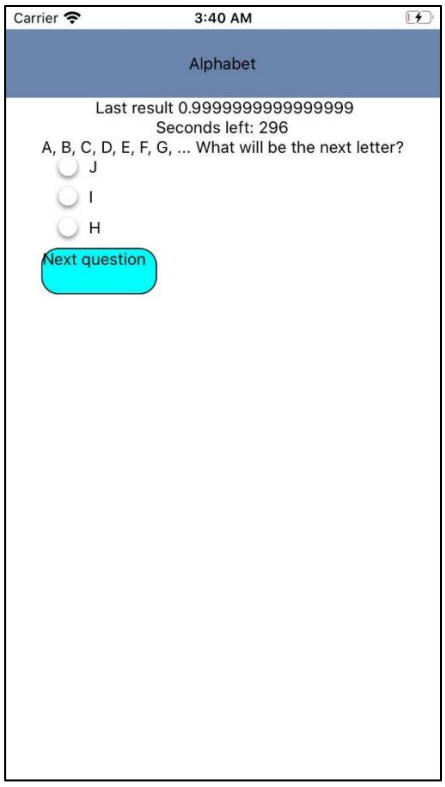

Рисунок 6.6 – Приклад проходження тесту

6.3 Аварійні ситуації та інформативні повідомлення

В **EasyLEng** існує два типи повідомлень: інформативні та аварійні. До інформативних повідомлень відносяться:

- Користувач успішно авторизувався в системі(рис 6.7.):

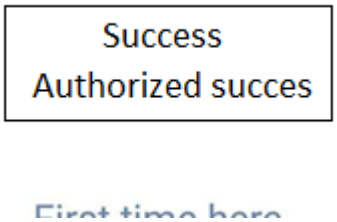

Рисунок 6.7 – Ви успішно авторизувались

До аварійних ситуацій в **EasyLEng** відносяться:

- Користувач не згоден з політикою мобільного засобу(рис 6.8.):

Рисунок 6.8 – Користувач не згоден з політикою **EasyLEng**

- Користувач ввів не коректний номер телефону(рис 6.9.):

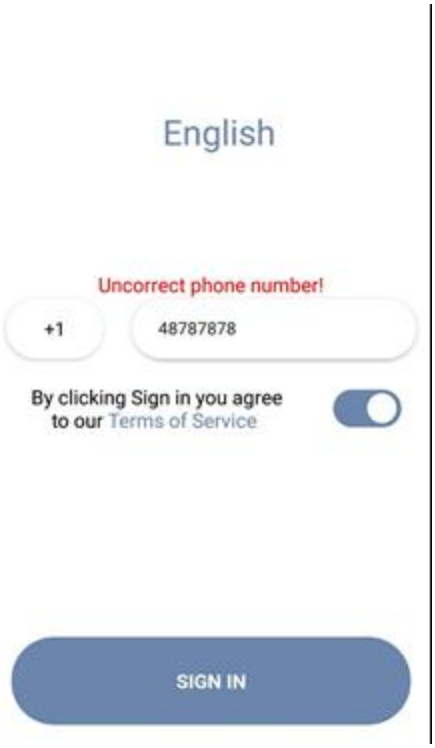

Рисунок 6.9 – Користувач ввів не коректний номер телефону

РОЗДІЛ 7 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

7.1 Сценарії тестування

На останньому етапі розробки програмного продукту, має бути проведено тестування для визначення і усунення можливих помилок.

Тестування програмного забезпечення - це одна з технік контролю якості, що включає в себе активності з планування робіт (Test Management), проектування тестів (Test Design), виконання тестування (Test Execution) і аналізу отриманих результатів (Test Analysis)..

Модульне тестування, або також відоме нам як юніт-тестування – це процес в програмуванні, що дозволяє перевірити на коректність окремі модулі вихідного коду програми. Ідея полягає в тому, щоб писати тести для кожної нетривіальною функції або методу.

Таблиця 7.1 - Сценарії тестування

№	Сценарій	Очікувана реакція	Мета проведення тесту
1	Створення облікового запису користувача	Обліковий запис користувача повинен бути створений	Перевірка функції створення облікового запису
2	Проходження тесту з англійської мови	Успішно проходиться тест	Перевірка функції проходження тесту
3	Додавання фотографії профілю	Фотографія профілю повинна бути додана до колекції	Перевірка як система додає нову фотографію

7.2 Результати тестування

Вимоги, які були встановлені в таблиці 7.1 були виконані повністю, за виключенням тесту номерів 3 та 4. На таблиці 7.2 приведені результати.

Таблиця 7.2 - Результати тестування

№	Сценарій	Заплановано дефектів	Виявлено дефектів	Результат дефектів	Час при «ручному» виявленні	Час при автоматичному виявленні
1	Створення облікового запису користувача	40%	15%	20%	15 секунд	3 секунди
2	Проходження тесту з англійської мови	86%	23%	38%	6 секунд	2 секунди
3	Додавання фотографії профілю	70%	20%	40%	13 секунд	8 секунд

7.3 Test-cases програмного забезпечення

Суть test-case (тестовий випадок) це тестовий артефакт, суть якого полягає у виконанні певної кількості дій. В таблиці 7.3 наведено приклад тест-кейсу для перевірки функції «Проходження тесту».

Таблиця 7.3 – Test-case «Проходження тесту»

Ідентифікатор	Користувач
Назва	Проходження тесту
Передумови	1. Користувач зареєстрований в системі. 2. Є підключення до інтернету.
Опис дій	1. Після реєстрації користувач проходить тест. 2. Натискає на кнопку завершити.

Продовження таблиці 7.3

Очікуваний результат 1	1.1. Вибирає відповідь. 1.2. Натискає на клавішу вибрати відповідь. 1.3. Обробка введених даних 2. Данні відправляються на сервер. 2.1 Данні оброблюються та заносяться в базу даних. 3. Результат тесту відображається на екрані. 3.1 Користувач може пройти тест.
Очікуваний результат 2	1.1. Користувач не завершує тест. 1.2. Відсутнє інтернет з’єднання. 1.3. Тест заноситься в локальну базу даних 1.4. При повторному підключенні до інтернету тест знову відправляються на сервер

Таблиця 7.3 – Test-case «Проходження тесту»

В таблиці 7.4 наведено приклад тест-кейсу для перевірки функції «Створення облікового записку користувача».

Таблиця 7.4 – Test-case «Створення облікового записку користувача»

Ідентифікатор	Користувач
Назва	Створення облікового записку користувача
Передумови	1. Користувач завантажив застосування. 2. Є підключення до інтернету.
Опис дій	1. Після відкривання застосування користувач потрапляє на екран авторизації. 2. Вводить данні(мобільний телефон, фотографію профілю.

Продовження таблиці 7.4

Очікуваний результат 1	1.1. Вибирає відповідь. 1.2. Натискає на клавішу вибрати відповідь. 1.3. Обробка введених даних 2. Данні відправляються на сервер. 2.1 Данні оброблюються та заносяться в базу даних. 3. Результат тесту відображається на екрані. 3.1 Користувач може пройти тест.
Очікуваний результат 2	1.2. Користувач не завершує тест. 1.2. Відсутнє інтернет з'єднання. 1.3. Тест заноситься в локальну базу даних 1.4. При повторному підключенні до інтернету тест знову відправляються на сервер

Цей розділ був присвячений тестуванню. Були сформовані сценарії тестування. Після проведення тестування отримали їх результати зі значенням виявлених дефектів.

ВИСНОВКИ

Кваліфікаційна робота була присвячена розробці програмного забезпечення для вивчення англійської мови яка має назву «Мобільне застосування для вивчення англійської мови».

Метою кваліфікаційної роботи було покращення рівня володіння англійською мовою. Я вважаю що мета даного програмного забезпечення була досягнена, так як ним користувались 12 людей, та опитування показало, що 8 користувачам сподобався інтерфейс. У 7(більшість) з 12 людей покращився рівень проходження початкового тесту в середньому на 20%(З 15 правильних відповідей до 18).

Робота почалася зі специфікації вимог до програмного забезпечення. З самого початку був проведений аналіз предметної області та пошук і вивчення існуючих програмних рішень. Далі було визначено основний функціонал системи у вигляді діаграми варіантів використання та їх опис.

Під час проектування була створена діаграма програмних класів, діаграми послідовності, прототипування зовнішнього вигляду системи та її реалізація.

Основний функціонал програмних модулів був реалізований на мові програмування TypeScript та фреймворку ReactNative разом з додатковими бібліотеками. Також були описані особливості та методи розробки мобільного застосування.

Також було добавлено посібник користувача для більш детального пояснення основних кроків для завантаження та користування застосунком.

Останнім етапом розробки стало її тестування, де були складені тестові випадки для декількох варіантів використання.

Розроблена система не є кінцевим результатом та може бути розширена за допомогою нового функціоналу . А саме потрібно додати наступні функціональні особливості:

- можливість додавати власні тести при досягненні рівня екперт в системі;

- підтримка застосунку для людей з обмеженими можливостями, шляхом добавлення підтримки проходження тесту використовуючи мікрофон та динамік телефону;
- додавання інших мов до системи, підтримка мультилокалізації.

Завдяки цьому функціоналу застосування може стати біль конкурентно-спроможним серед аналогів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація ReactNative [Електронний ресурс] / ReactNative documentation. – Режим доступу: www / URL : https://reactnative.dev/guides – 23.05.2020 р.

2. Документація React [Електронний ресурс] / React documentation. – Режим доступу: www / URL : https://react.dev/docs – 23.05.2020 р.

3. Документація Firebase [Електронний ресурс] / Python documentation. – Режим доступу: www / URL : https://firebase.org/3/ – 23.05.2020 р.

4. Документація Flask [Електронний ресурс] / Flask documentation. – Режим доступу: www / URL : https://flask.palletsprojects.com/en/1.1.x/ – 23.05.2020 р.

5. Документація PostgreSQL [Електронний ресурс] / PostgreSQL documentation. – Режим доступу: www / URL : https://www.postgresql.org/docs/ – 23.05.2020 р.

6. Sinuyk Information systems and technologies as an instrument for increase qualification// Матеріали науково-практичної інтернет-конференції «Економічна кібернетика: теорія, практика та напрямки розвитку», 28-29 листопада 2018, – Одеса: ОНПУ, 2018, – 156 с

7. Sinuyk Features of the development of social projects using information technology// Informatics and Mathematical Methods in Simulation Vol. 8 (2018), No. 3, pp.238-244

8. Синюк Вимоги до випускників IT-спеціальностей // Матеріали науково-практичної інтернет-конференції «Економічна кібернетика: теорія, практика та напрямки розвитку», 27-28 листопада 2019, – Одеса: ОНПУ, 2019, – 149 с

9. Журан О.А, Філатова Т.В., Модель формування сучасних компетенцій IT-фахівців// Informatics and Mathematical Methods in Simulation Vol. 9 (2019), No. 3, pp.195-202

10. Філатова Т. В., Використання інформаційних інтернет-технологій для підтримки прийняття рішень визначення актуальних професій // Матеріали науково-практичної інтернет-конференції «Економічна кібернетика: теорія,

практика та напрямки розвитку», 28-29 листопада 2017, – Одеса: ОНПУ, 2017, – 111 с

11. Філатова Т., Методологія представлення соціальних проектів ІТ-індустрії // Матеріали ІІІ Міжнародної конференції «Комп'ютерна алгебра та інформаційні технології», Одеса, 20–25 серпня 2018 р./Одеський національний університет імені І.І. Мечникова, – Одеса 2018. – 198с.

12. Брайченко. С,х` Інформаційні системи та технології як інструмент для аналізу актуальності існуючих професій // Матеріали IV Міжнародної науково-практичної конференції студентів, аспірантів та молодих науковців «Актуальні питання документознавства та інформаційної діяльності: теорії та інновації», Одеса, 22-23 березня 2018 р./Одеський національний політехнічний університет, – Одеса 2018. – 569с.

13. Бізнес-план: розробка, обґрунтування та аналіз. Навч. посібник. / Тарасюк Г.М., К.:Каравелла, 2004. – 344 с.

14. Управління проектами. / Тян Р.Б., Холод Б.І., Ткаченко В.А. К.: ЦНП, 2004. –224 с.

15. Виробничий (операційний) менеджмент. / Василенко В. О. – ПУЛ., 2004. –535с.

16. Операційний менеджмент: Навч. пос. / Михайловська О.В. – К.: Кондор, 2008. – 600 с.

17. Бундюк А.М. Управління проектами: організація, планування, автоматизація: навч. посібник / А.М. Бундюк. - Одеса: ОНПУ, 2014. - 264 с.

18. Катренко А.В. Управління ІТ-проектами. Кн. 1. Стандарти, моделі та методи управління проектами / А.В. Катренко; за ред. В.В. Пасічника. - Л.: Новий Світ-2000, 2011. - 550 с.

19. Кобиляцький, Л. С. Управління проектами: Навч. посібник. - К.: МАУП, 2002. - 200 с.

20. Конспект лекцій по курсу "Управління проектами інформатизації" для студентів спеціальності - 051 Економіка, спеціалізація Економічна кібернетика / Укладач: О.А. Журан. Одеса: ОНПУ, 2019. - 160 с. URL:

http://memos.library.opu.ua:8080/memos/jsp/materials.iface?mId=40161 (дата звернення: 29.05.2020).

21. Ноздріна Л. В., Ящик В. І., Полотай О. І. Управління проектами / Підручник. – К.: Центр учбової літератури, 2010. – 432 с. URL: http://www.immsp.kiev.ua/postgraduate/Biblioteka_trudy/UpravlinnyaProektamiNozdrina2010.pdf (дата звернення: 11.05.2020).

22. Управління проектами: Навч. посібник. / Л.Є. Довгань, Г.А. Мохонько, І.П Малик. – К.: КПІ ім. Ігоря Сікорського, 2017. – 420 с. URL: http://ela.kpi.ua/bitstream/123456789/19481/1/DMM_UP_2017.pdf (дата звернення: 09.04.2020).

23. Щукін Б.М. Аналіз інвестиційних проектів: Конспект лекцій. - К.: МАУП, 2002.-128с.

24. Данілов О. Д. Інвестування: Навчальний посібник для вузів / О.Д.

25. Стаття ReactNative [Електронний ресурс] Oreilly. – Режим доступу: www / URL : https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html – 02.06.2017 р.

26. Стаття ClientServer [Електронний ресурс] Omnisci. – Режим доступу: www / URL : https://www.omnisci.com/technical-glossary/client-server– 23.05.2019 р.

27. Dobrynin, Ye. V., Boltenkov, V. O. & Maksymov, M. V. “Information Technology for Automated Assessment of the Artillery Barrels Wear Based on SVM Classifier”. Applied Aspects of Information Technology. Publ. Nauka i Tekhnika. Odessa: Ukraine. 2020; Vol. 3 No. 3: 117–132. DOI: https://doi.org/10.15276/aait.03.2020.1

28. Sivokobylenko V. F., Nikiforov A. P. & Zhuravlov I. V. “ Detecting development scenarios of dynamic events in electric power networks smart-grid. Part 1 “Method”. Applied Aspects of Information Technology. Publ. Nauka i Tekhnika. Odessa: Ukraine. 2021; Vol. 4 No. 3: 219– 234. DOI: https://doi.org/10.15276/aait.03.2021.1

ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ

```
impor
t {
            CHAT_LOADING,
            CHAT_ERROR,
            CHAT_DATA,
            CHAT_UNSUBSCRIBE,
            FILE_LOADING,
            PROFILE_CHAT_DATA,
            PROFILE_CHAT_LOADING,
          } from '../actionTypes';
          import firestore from '@react-native-firebase/firestore';
          import storage from '@react-native-firebase/storage';

          export const getChat = (limit) => (dispatch) => {
            dispatch({
              type: CHAT_LOADING,
            });
            const unsubscribe = firestore()
              .collection('chat')
              .limit(limit)
              .orderBy('date', 'desc')
              .onSnapshot(
                (snapshotList) => {
                  const newdata = [];
                  snapshotList.forEach((item) =>
          newdata.push(item.data()));
                  dispatch({
                    type: CHAT_DATA,
                    payload: newdata,
                  });
                },
                (e) => console.log(e),
              );
            dispatch({
              type: CHAT_UNSUBSCRIBE,
              payload: unsubscribe,
            });
          };

          export const sendMessage = (message, nick, messageType, extra)
          => (
            dispatch,
          ) => {
            const date = Date.now();
            let data;
```

```
  switch (messageType) {
    case 'default':
      data = {
        nick,
        message,
        date,
        messageType: messageType,
        edited: false,
      };
      break;
    case 'reply':
      data = {
        nick,
        message,
        date,
        messageType,
        edited: false,
        replyMessage:
          extra.messageType === 'image'
            ? 'Photo'
            : extra.messageType === 'sticker'
            ? 'Sticker'
            : extra.message,
        replyNick: extra.nick,
        replyImage: extra.image ? extra.image : false,
      };
      break;
    case 'sticker':
      data = {
        nick,
        date,
        messageType,
        sticker: extra,
      };
      break;
    case 'image':
      const referenceStorage =
storage().ref(`chat/${date.toString()}`);
      const task = referenceStorage.putFile(extra);
      dispatch({
        type: FILE_LOADING,
        payload: true,
      });
      task.then(async () => {
        const image = await referenceStorage.getDownloadURL();
        data = {
          nick,
          date,
```

```
          messageType,
          image,
          message,
        };
        firestore()
          .collection('chat')
          .doc(date.toString())
          .set(data)
          .then(() => dispatch({type: FILE_LOADING, payload:
false}));
      });

      break;
    default:
      console.log('Somethinh wrong');
  }
  messageType !== 'image' &&
    firestore()
      .collection('chat')
      .doc(date.toString())
      .set(data)
      .then(() => console.log('send'));
};

export const deleteMessage = (id, messageType) => (dispatch) =>
{
  if (messageType === 'image') {
    const referenceStorage =
storage().ref(`chat/${id.toString()}`);
    referenceStorage.delete().then(() => {
      firestore()
        .collection('chat')
        .doc(id.toString())
        .delete()
        .then(() => console.log('okey'));
    });
  } else {
    firestore()
      .collection('chat')
      .doc(id.toString())
      .delete()
      .then(() => console.log('okey'));
  }
};

export const updateMessage = (id, message) => (dispatch) => {
  firestore()
    .collection('chat')
```

```
            .doc(id.toString())
            .update({
              message,
              edited: true,
            })
            .then(() => console.log('updated'));
        };

        export const fetchProfile = (nick) => (dispatch) => {
          dispatch({
            type: PROFILE_CHAT_LOADING,
          });
          firestore()
            .collection('users')
            .where('nick', '==', nick)
            .get()
            .then((querySnapshot) => {
              querySnapshot.forEach((snapshot) => {
                dispatch({
                  type: PROFILE_CHAT_DATA,
                  payload: snapshot.data(),
                });
              });
            });
        };
import React, {Component} from 'react';
                                          import {
                                            View,
                                            Text,
                                            FlatList,
                                            TextInput,
                                            TouchableOpacity,
                                            ImageBackground,
                                            Keyboard,
                                            Image,
                                            ActivityIndicator,
                                          } from 'react-native';
                                          import Header from '../../../components/Header';
                                          import {container} from '../../../resource/style';
                                          import {connect} from 'react-redux';
                                          import {
                                            getChat,
                                            sendMessage,
                                            deleteMessage,
                                            updateMessage,
                                          } from '../../../actions/chatActions';
                                          import Icon from 'react-native-vector-
                                          icons/MaterialCommunityIcons';
```

```
import IconAttachment from 'react-native-vector-
icons/Entypo';
import IconEdit from 'react-native-vector-
icons/MaterialIcons';
import MessageList from
'../../../components/MessageList';
import moment from 'moment';
import {
  backgroundImageStyle,
  chatContainerStyle,
  editMessageStyle,
  editTextContainer,
  editTitleStyle,
  iconEditStyle,
  iconSendImage,
  inputChatContainer,
  inputEditContainer,
  loadingImageStyle,
  modalBackgroundStyle,
  modalContainer,
  modalItemStyle,
  modalItemText,
  newDayStyle,
  replyImageStyle,
  sendBlockStyle,
  sendImageStyle,
  stickerContainer,
  stickerStyle,
  textInputChat,
} from '../../../resource/style/chat';
import {
  w,
  h,
  blueColor,
  remA,
  stickerArray,
  remH,
  remW,
} from '../../../resource/constants';
import ImagePicker from 'react-native-image-
picker';
import ImageResizer from 'react-native-image-
resizer';

class Chat extends Component {
  state = {
    message: '',
    modalVisible: false,
```

```
    x: 0,
    y: 0,
    modalItem: undefined,
    edit: false,
    itemsSelected: false,
    limit: 40,
    prevLimit: 0,
    reply: false,
    sticker: false,
    sourceImg: undefined,
    pathImg: undefined,
    el: undefined,
  };

  componentWillUnmount(): void {
    const {unsubscribe} = this.props;
    console.log('un')
    unsubscribe();
  }

  componentDidMount() {
    const {getChat} = this.props;
    getChat(this.state.limit);
  }

  _sendDefault() {
    const {message} = this.state;
    const {sendMessage, nick} = this.props;
    if (message.trim()) {
      let trimedmessage = message.trimEnd();
      trimedmessage = trimedmessage.trimStart();
      sendMessage(trimedmessage, nick, 'default');
      this.setState((prev) => ({...prev, message:
''}));
    }
  }

  _sendSticker(index) {
    const {sendMessage, nick} = this.props;
    sendMessage(null, nick, 'sticker', index);
  }

  _sendImage() {
    const {message, pathImg} = this.state;
    const {sendMessage, nick} = this.props;
    let trimedmessage = message.trimEnd();
    trimedmessage = trimedmessage.trimStart();
```

```
    sendMessage(trimedmessage, nick, 'image',
pathImg);
    this.setState((prev) => ({
      ...prev,
      message: '',
      sourceImg: undefined,
      pathImg: undefined,
    }));
  }

  _pickImage() {
    ImagePicker.showImagePicker((res) => {
      res.uri &&
        ImageResizer.createResizedImage(res.uri,
w, h, 'JPEG', 100, 0)
          .then((response) => {
            this.setState((prev) => ({
              ...prev,
              sourceImg: {uri: response.uri},
              pathImg: response.path,
            }));
          })
          .catch((err) => {
            console.log(err);
          });
    });
  }

  _showModal(event, item) {
    const {nick} = this.props;
    let el = 3;
    if (nick !== item.nick) {
      el = 2;
    } else if (item.messageType === 'sticker') {
      el = 2;
    }
    let xc = event.nativeEvent.pageX;
    let yc = event.nativeEvent.pageY;
    if (xc - 67 * remW > 0 && xc + 67 * remW < w -
30 * remW) {
      xc -= 67 * remW;
    } else if (xc + 67 * remW > w - 30 * remW) {
      xc -= 125 * remW;
    }
    if (yc - el * 30 * remH > 0 && yc + el * 30 *
remH < h - 30 * remH) {
      yc -= el * 30 * remH;
```

```
    } else if (yc + el * 30 * remH > h - 30 *
remH) {
      yc -= el * 60 * remH;
    }
    this.setState((prev) => ({
      ...prev,
      sourceImg: undefined,
      pathImg: undefined,
      modalVisible: true,
      x: xc,
      y: yc,
      reply: false,
      edit: false,
      modalItem: item,
      el,
    }));
  }

  _openDetails() {
    const {navigation} = this.props;
    const {nick} = this.state.modalItem;
    return (
      <TouchableOpacity
        style={modalItemStyle}
        onPress={() => {
          this.setState((prev) => ({...prev,
modalVisible: false}));
          navigation.navigate('ProfileDetails',
{nickDetails: nick});
        }}>
        <Icon size={26 * remA} name={'face-
profile'} color={'grey'} />
        <Text style={modalItemText}>Open
profile</Text>
      </TouchableOpacity>
    );
  }
  _generateModal() {
    const {modalItem} = this.state;
    const {messageType} = modalItem;
    const {nick} = this.props;

    switch (messageType) {
      case 'default':
        return (
          <View>
            {nick !== modalItem.nick &&
this._openDetails()}
```

```
            {this._generateReply()}
            {nick === modalItem.nick &&
this._generateEdit()}
            {nick === modalItem.nick &&
this._generateDelete()}
          </View>
        );
      case 'sticker':
        return (
          <View>
            {nick !== modalItem.nick &&
this._openDetails()}
            {this._generateReply()}
            {nick === modalItem.nick &&
this._generateDelete()}
          </View>
        );
      case 'image':
        return (
          <View>
            {nick !== modalItem.nick &&
this._openDetails()}
            {this._generateReply()}
            {nick === modalItem.nick &&
this._generateEdit()}
            {nick === modalItem.nick &&
this._generateDelete()}
          </View>
        );
      case 'reply':
        return (
          <View>
            {nick !== modalItem.nick &&
this._openDetails()}
            {this._generateReply()}
            {nick === modalItem.nick &&
this._generateEdit()}
            {nick === modalItem.nick &&
this._generateDelete()}
          </View>
        );
    }
  }

  _generateReply() {
    return (
      <TouchableOpacity
        style={modalItemStyle}
```

```
        onPress={() => {
          this.inputRef.blur();
          setTimeout(() => this.inputRef.focus(),
400);
          this.setState((prev) => ({
            ...prev,
            reply: true,
            modalVisible: false,
          }));
        }}>
        <Icon size={26 * remA} name={'reply'}
color={'grey'} />
        <Text style={modalItemText}>Reply</Text>
      </TouchableOpacity>
    );
  }

  _generateEdit() {
    const {message} = this.state.modalItem;
    return (
      <TouchableOpacity
        style={modalItemStyle}
        onPress={() => {
          this.inputRef.blur();
          setTimeout(() => this.inputRef.focus(),
400);
          this.setState((prev) => ({
            ...prev,
            edit: true,
            modalVisible: false,
            message,
          }));
        }}>
        <IconEdit size={26 * remA} name={'edit'}
color={'grey'} />
        <Text style={modalItemText}>Edit</Text>
      </TouchableOpacity>
    );
  }

  _generateDelete() {
    return (
      <TouchableOpacity
        style={modalItemStyle}
        onPress={() => this._deleteMessage()}>
        <Icon size={26 * remA} name={'trash-can-
outline'} color={'grey'} />
        <Text style={modalItemText}>Delete</Text>
```

```
      </TouchableOpacity>
    );
  }

  _deleteMessage() {
    const {deleteMessage} = this.props;
    const {date, messageType} =
this.state.modalItem;
    deleteMessage(date, messageType);
    this.setState((prev) => ({
      ...prev,
      modalVisible: false,
      x: 0,
      y: 0,
      modalItem: undefined,
    }));
  }

  _updateMessage() {
    const {updateMessage} = this.props;
    const {message, modalItem} = this.state;
    if (message.trim() && message.trim() !==
modalItem.message) {
      let trimedmessage = message.trimStart();
      trimedmessage = trimedmessage.trimEnd();
      updateMessage(modalItem.date,
trimedmessage);
    }
    this.setState((prev) => ({
      ...prev,
      edit: false,
      modalItem: undefined,
      message: '',
    }));
  }

  _sendReply() {
    const {message, modalItem} = this.state;
    const {sendMessage, nick} = this.props;
    if (message.trim()) {
      let trimedmessage = message.trimEnd();
      trimedmessage = trimedmessage.trimStart();
      sendMessage(trimedmessage, nick, 'reply',
modalItem);
      this.setState((prev) => ({
        ...prev,
        reply: false,
        modalVisible: false,
```

```
        edit: false,
        modalItem: undefined,
        message: '',
      }));
    }
  }

  _moreData() {
    const {unsubscribe, getChat, chatData} =
this.props;
    const {limit, prevLimit} = this.state;
    if (chatData.length === limit &&
chatData.length !== prevLimit) {
      unsubscribe();
      getChat(limit + 40);
      this.setState((prev) => ({
        ...prev,
        limit: limit + 40,
        prevLimit: prevLimit + 40,
      }));
    }
  }

  render() {
    const {
      navigation,
      error,
      loading,
      chatData,
      nick,
      fileLoading,
    } = this.props;
    const {
      message,
      modalVisible,
      x,
      y,
      edit,
      modalItem,
      itemsSelected,
      reply,
      sticker,
      sourceImg,
      el,
    } = this.state;
    return (
      <View style={container}>
        <Header
```

```
          navigate={() =>
navigation.toggleDrawer()}
          title={'Chat'}
          icon={'menu'}
        />
        <View style={chatContainerStyle}>
          <ImageBackground

source={require('../../../resource/image/assets/ba
ckground.jpg')}
            style={backgroundImageStyle}>
            <FlatList
              keyboardShouldPersistTaps={'always'}
              inverted={true}
              windowSize={16}
              onEndReached={(info:
{distanceFromEnd: number}) =>
                this._moreData()
              }
              onEndReachedThreshold={0.1}
              data={chatData}
              renderItem={({item, index}) => {
                let sametitle = true;
                if (index < chatData.length - 1) {
                  if (chatData[index + 1].nick ===
item.nick) {
                    sametitle = false;
                  }
                }
                let newdate = true;
                if (index < chatData.length - 1) {
                  if (
                    moment(chatData[index +
1].date).format('MMMM Do YYYY') ===
                    moment(item.date).format('MMMM
Do YYYY')
                  ) {
                    newdate = false;
                  }
                }
                return (
                  <View>
                    {newdate && (
                      <Text style={newDayStyle}>
                        {moment(item.date).year()
=== moment().year()
                          ?
moment(item.date).format('Do MMMM')
```

```
                        :
moment(item.date).format('Do MMMM, YYYY')}
                      </Text>
                    )}
                    <MessageList
                      item={item}
                      mynick={nick}
                      sametitle={sametitle}
                      showModal={(ev, it) =>
this._showModal(ev, it)}
                      showIncreasedPhoto={(el) =>
navigation.navigate('IncreasedPhoto', {nick:
el.nick, avatar: el.image, date: el.date})}
                      showSticker={() =>
                        this.setState((prev) =>
({...prev, sticker: true}))
                      }

itemsSelected={itemsSelected}
                    />
                  </View>
                );
              }}
              keyExtractor={(item) =>
item.date.toString()}
            />
          </ImageBackground>

          {(edit || reply) && (
            <View style={inputEditContainer}>
              <View style={iconEditStyle}>
                <IconEdit
                  size={26 * remA}
                  name={edit ? 'edit' : 'reply'}
                  color={blueColor}
                />
              </View>
              {modalItem.messageType === 'image'
&& (
                <Image
                  source={{uri: modalItem.image}}
                  style={replyImageStyle}
                />
              )}
              <View style={editTextContainer}>
                <Text style={editTitleStyle}>
                  {edit && 'Edit Message'}
                  {reply && modalItem.nick}
```

```
                </Text>
                <Text style={editMessageStyle}
numberOfLines={1}>
                  {modalItem.messageType ===
'sticker' ? (
                    <Text style={{color:
'#247d09'}}>Sticker</Text>
                  ) : modalItem.messageType ===
'image' ? (
                    <Text style={{color:
'#247d09'}}>Photo</Text>
                  ) : (
                    modalItem.message
                  )}
                </Text>
              </View>
              <View style={iconEditStyle}>
                <TouchableOpacity
                  onPress={() =>
                    this.setState((prev) => ({
                      ...prev,
                      edit: false,
                      message: edit ? '' :
message,
                      reply: false,
                      modalItem: undefined,
                    }))
                  }>
                  <Icon name={'close'} size={26 *
remA} color={'grey'} />
                </TouchableOpacity>
              </View>
            </View>
          )}
          <View style={inputChatContainer}>
            <View style={sendBlockStyle}>
              <TouchableOpacity
                onPress={() => {
                  Keyboard.dismiss();
                  this.setState((prev) => ({
                    ...prev,
                    sticker: !sticker,
                    sourceImg: undefined,
                    pathImg: undefined,
                  }));
                }}>
                <Icon
                  size={26 * remA}
```

```
                color={'gray'}
                name={sticker ? 'close' :
'sticker-emoji'}
              />
            </TouchableOpacity>
          </View>
          <TextInput
            ref={(input) => {
              this.inputRef = input;
            }}
            onFocus={() =>
              this.setState((prev) => ({...prev,
sticker: false}))
            }
            maxLength={4000}
            multiline={true}
            returnKeyType={'done'}
            style={textInputChat}
            placeholderTextColor={'gray'}
            placeholder={'Message'}
            value={message}
            blurOnSubmit={false}
            onChangeText={(val) =>
              this.setState((prev) => ({...prev,
message: val}))
            }
          />
          {message.length || edit || sourceImg ?
(
            <View
              style={[
                sendBlockStyle,
                {
                  paddingBottom: sourceImg ? 6 *
remH : 11 * remH,
                  paddingRight: sourceImg ? 6 *
remW : 0,
                },
              ]}>
              <TouchableOpacity
                onPress={() =>
                  edit
                    ? this._updateMessage()
                    : reply
                    ? this._sendReply()
                    : sourceImg
                    ? this._sendImage()
                    : this._sendDefault()
```

```
                    }>
                    {sourceImg ? (
                      <View>
                        <Image
style={sendImageStyle} source={sourceImg} />
                        <View style={iconSendImage}>
                          <Icon
                            size={26 * remA}
                            color={blueColor}
                            name={'send'}
                          />
                        </View>
                      </View>
                    ) : (
                      <Icon
                        size={26 * remA}
                        color={blueColor}
                        name={edit ? 'check-circle'
: 'send'}
                      />
                    )}
                  </TouchableOpacity>
                </View>
              ) : fileLoading ? (
                <View style={loadingImageStyle}>
                  <ActivityIndicator size={'large'}
color={blueColor} />
                </View>
              ) : (
                <View style={sendBlockStyle}>
                  <TouchableOpacity
                    onPress={() => {
                      this.setState((prev) =>
({...prev, reply: false}));
                      this._pickImage();
                    }}>
                    <IconAttachment
                      size={26 * remA}
                      color={'#30618a'}
                      name={'attachment'}
                    />
                  </TouchableOpacity>
                </View>
              )}
            </View>
            {sticker && (
              <View style={stickerContainer}>
                <FlatList
```

```
showsVerticalScrollIndicator={false}
                numColumns={5}
                data={stickerArray}
                renderItem={({item, index}) => (
                  <TouchableOpacity onPress={() =>
this._sendSticker(index)}>
                    <Image style={stickerStyle}
source={item} />
                  </TouchableOpacity>
                )}
                keyExtractor={(item) =>
item.toString()}
              />
            </View>
          )}
        </View>
        {modalVisible && (
          <TouchableOpacity
            style={modalBackgroundStyle}
            onPress={() =>
              this.setState((prev) => ({...prev,
modalVisible: false}))
            }>
            <View
              style={[
                modalContainer,
                {height: el * 60 * remH,
marginTop: y, marginLeft: x},
              ]}>
              {this._generateModal()}
            </View>
          </TouchableOpacity>
        )}
      </View>
    );
  }
}

const actionsCreators = {
  getChat,
  sendMessage,
  deleteMessage,
  updateMessage,
};

const mapState = (state) => {
```

```
  const {chatData, error, loading, unsubscribe,
fileLoading} = state.chat;
  const {nick} = state.profile.profileData;
  return {
    unsubscribe,
    nick,
    chatData,
    error,
    loading,
    fileLoading,
  };
};

export default connect(mapState,
actionsCreators)(Chat);
```