Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

Випускна кваліфікаційна робота бакалавра
«**Інформаційна система узгодження процесів електронної освіти (за зразком preply.com) (Front mobile)**»

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проєкту

№	П.І.Б.	Група
1	Чернявський Дмитро Вікторович	КН-П-202
2	Пересунько Ігор Вадимович	КН-П-202
3	Легкодух Максим Віталійович	КН-П-202

Автор звіту

_____ Чернявський Дмитро Вікторович

Одеса – 2024

АНОТАЦІЯ

Чернявський Д. В. Онлайн-платформа яка допомагає знайти репетиторів за бажаною мовою та замовити його послуги: Автореферат випускної кваліфікаційної роботи на здобуття бакалаврського рівня вищої освіти зі спеціальності «122 Комп'ютерні науки». - Одеса: ОТУШ, 2024

Робота присвячена аналізу, розробці та тестуванню онлайн-платформи придбання послуг репетиторів. Головною функцією якої є взаємодія репетиторів з клієнтами. Платформа поділяється на багато частин, кожна з яких має свою функціональність. Основними функціями цих частини являються: реєстрація, автентифікація, авторизація, функція надання послуг від репетиторів, а також придбання цих послуг користувачами платформи, функція пошуку, система відгуків, уподобань користувача, отримання інформаційних листів на електронну пошту від нашого сервісу.

Релевантність проекту підтверджується реалізацією сучасних функцій, які полегшують процес пошуку та замовлення послуг репетитора. Зручність використання сервісу забезпечується різними способами авторизації, включаючи можливість входу через соціальні мережі. Основний акцент було зроблено на створенні ефективної системи пошуку. Ми доклали максимум зусиль, щоб користувач міг фільтрувати та сортувати результати пошуку без обмежень, що дозволяє знайти найвідповіднішого репетитора в умовах невизначеності.

Актуальність сервісу також базується на впровадженні надійної системи безпеки для захисту, збереження та цілісності персональних даних. В цю систему входить шифрування вхідних та вихідних даних, розробка авторизаційних та інших фільтрів на серверній частині, перевірка на коректність переданих даних та багато інших технологій.

Об’єктом дослідження у рамках даної роботи є алгоритм підрахунку загальної оцінки послуг репетитора за кількома різними категоріями оцінювання. У якості задач дослідження було встановлено наступне:

- Проаналізувати різні способи реалізації взаємодії клієнтської та серверної частини за допомогою SPA (Single Page Application).
- Реалізувати отримання вхідних даних від користувача на серверній стороні сервісу.
- Розробити програмний інтерфейс (API) для безпечного отримання вхідних даних від клієнтської частини платформи за допомогою протоколу HTTP.
- Реалізувати максимально продуктивний алгоритм з урахуванням всіх складностей, застосувавши передові технології для доступу до бази даних, такі як Entity Framework Core.
- Впровадити у серверну та клієнтську частину програмні засоби захисту для забезпечення збереження та цілісності користувацьких даних.

Методи дослідження, що використовувалися для досягнення поставлених задач, включають методи тестування програмного забезпечення, гнучкі методології розробки, системний аналіз, моделювання архітектури за допомогою принципів SOLID та теорії баз даних.

Особливу увагу було приділено організації роботи команди за допомогою Agile-методології, що дозволило гнучко розподілити задачі, своєчасно їх виконувати та отримувати зворотний зв’язок від ментора для вдосконалення коду та виправлення помилок. Також ми застосували методологію екстремального програмування (XP), що допомогло кожному учаснику команди краще розуміти загальну картину проекту та ефективно вирішувати задачі, уникати затримок і підвищувати якість коду. Це забезпечило безперервну інтеграцію, раннє виявлення дефектів, підвищення продуктивності та більш швидкі цикли випуску продукту.

Для розробки програмного забезпечення команда також використовувала розподілену систему контролю версій Git та веб-сервіс, який її реалізує - GitHub. Під час використання цієї системи та розробки ми також практикували альтернативну модель розгалуження Gitflow. При складанні звіту використано 14 посилань на електронні джерела.

ЗМІСТ

ВСТУП

Сучасне життя неможливо уявити без інформаційних технологій та цифрового розвитку суспільства. Кожного дня все більше аспектів нашого життя переходить до цифрового простору, включаючи освіту. Спираючись на стратегію розвитку інформаційних технологій Міністерства цифрової трансформації України, наша команда створила сервіс, який представляє собою інформаційну систему узгодження процесів електронної освіти.

Наша команда вірить, що інвестиції у технології, які полегшують та роблять життя людей комфортнішим, мають бути найважливішою ціллю сучасних розробників програмного забезпечення. Через це наш сервіс надає безліч можливостей для поліпшення навчального процесу без зайвих зусиль та пропонує популярні послуги з електронної освіти для здобуття знань у будь-якому місці та у зручний час.

Також платформа допомагає викладачам і навчальним закладам приваблювати студентів, застосовуючи максимально прості та вигідні умови для розвитку освіти в Україні та за її межами. В той же час, мільйони людей, котрі хочуть отримати якісну освіту, мають можливість скористатися найкращими інструментами для реалізації своїх навчальних планів. Які б цілі не мали користувачі нашої платформи, будучи викладачами або студентами, сервіс допоможе їм із вирішенням широкого спектру освітніх питань.

Далі у цьому документі розглянуто аналіз системи, проектування архітектури, саму розробку, тестування та майбутню підтримку цифрової платформи для електронної освіти. Також будуть розглянуті проблеми, спірні питання, вирішення та їх реалізація, а також проведений аналіз командної роботи.

Вичерпне технічне завдання до проекту представлено у наступному розділі під назвою “Технічне завдання до проекту”. У рамках даної роботи розглядається серверна частина платформи, а також елементи клієнтської сторони мобільного додатку. Висновки показують результати, які були отримані при тестуванні даної частини проекту, загальні практичні та логічні підсумки є сукупністю звітів усіх учасників проекту.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЕКТУ

ІНФОРМАЦІЙНА СИСТЕМА УЗГОДЖЕННЯ ПРОЦЕСІВ ЕЛЕКТРОННОЇ ОСВІТИ

(за прикладом сервісу Preply.com)

Загальний опис

Програмне забезпечення (ПЗ) призначене для створення розподіленої системи узгодження взаємодії користувачів у контексті навчального процесу. Головна функція ПЗ полягає в узгодженні інтересів користувача, який надає послуги навчання, та користувача, що споживає ці послуги. Додаткова функція ПЗ має дозволити залишати відгуки та оцінки щодо усіх видів користувачів.

Призначення:

ПЗ орієнтоване на різні види узгодження користувачів: за мовою викладання, за вартістю послуг, за рейтингом тощо. ПЗ має бути універсальним, щоб дозволити змінити цільове призначення для різних видів послуг.

Вимоги до функціоналу. Обов'язкова частина

Реєстрація користувачів

- Користувач ПЗ має змогу зареєструватись за допомогою авторизаційних даних (логіну/паролю).
- Під час реєстрації обов'язково вказується електронна пошта, яка підлягає верифікації.
- Користувач обирає роль: споживач або надавач послуг. Одна особа може зареєструватись і як споживач, і як надавач послуг.
- Під час реєстрації користувач вказує своє ім'я, яке буде використовуватись у листуванні.

Особистий кабінет надавача послуг

- Можливість розміщення пропозицій з навчання.
- Встановлення обмежень щодо рейтингу споживачів.
- Управління розкладом занять.

Особистий кабінет споживача послуг

- Пошук пропозицій за різними критеріями.
- Перегляд відгуків та рейтингу надавачів послуг.
- Управління власним розкладом навчання.

Авторизація та особистий кабінет користувача

- За фактом успішної авторизації користувачу надається доступ до особистого кабінету. Це окремий інформаційний ресурс (сторінка), що являє собою основне робоче місце користувача.

Зворотній зв’язок

- Користувач має змогу встановити власну оцінку або відгук для спожитого ресурсу чи його власника (надавача).

Засоби пошуку

Стартовий інтерфейс ПЗ (головна сторінка або активність) має за мету популяризацію системи. На ньому має бути забезпечено відображення популярних показників як-то загальна кількість користувачів та інформаційних одиниць, вибірка з найбільш популярного контенту та нових дописів, зовнішня реклама. Також головна сторінка має містити засоби авторизації та реєстрації або посилання на них.

При реєстрації обов'язково зазначається засіб оперативного контактування (ЗОК) - телефон та електронна пошта. На тестовому етапі, за умови ускладнення замовлення послуг телефонії чи СМС, можливе використання лише електронної пошти. Засіб оперативного контактування при реєстрації підлягає верифікації через відповідь на тестовий виклик чи надісланий лист, або введення коду, пересланого через повідомлення довільного типу. Рекомендується використовувати ЗОК у якості авторизаційного логіну користувача.

За умови що користувач забув пароль, ЗОК має бути використаний для його відновлення. Схема відновлення паролю жорстко не регламентується. Також передбачається наявність у системі користувачів з додатковими правами: модераторів та адміністраторів. Реєстрація кореневого адміністратора системи має бути реалізоване при інсталяції системи, наступним користувачам змінюють права наявні адміністратори.

Вимоги до функціоналу. Опціональна частина

Модерація

Для забезпечення дотримання загальноприйнятих норм спілкування, а також з метою недопущення публікування контенту, який містить заборонені компоненти (ненормативні висловлювання, заклики до протиправних дії, приниження чи дискримінація, тощо), усі інформаційні одиниці, а також створені користувачами метадані (маркери) мають в обов'язковому порядку пройти погодження (модерацію). Право надати погодження належить кореневому адміністратору системи, а також модераторам. При погодженні інформаційна одиниця має зберігати ідентифікатор користувача, який надав погодження.

Архітектура

ПЗ має бути спроектоване за розподіленою клієнт-серверною архітектурою з чітко відокремленими розподіленими вузлами системи:

- **Backend** - серверне програмне забезпечення, що забезпечує збереження та оброблення поточних даних, взаємодію з базою даних. Надає програмний інтерфейс взаємодії (API) з іншими вузлами.
- **Web App** - додаток, доступний з веб-простору (сайт), що забезпечує візуальний інтерфейс користувачів усіх категорій
- **Mobile App** - додаток, що встановлюється на мобільних пристроях (смартфон, планшет). Дана частина виконується за наявності у команді проекту трьох і більше програмістів.
-

Усі модулі ПЗ повинні обов'язково включати до свого складу засоби

- тестування функціональності та модульні тести,
- розгортання на новому пристрої (інсталяції) із встановленням початкових таблиць у БД та кореневого користувача-адміністратора

Технології

Зберігання вихідних кодів ПЗ повинно здійснюватись за допомогою системи контроля версій (вибір конкретної системи узгоджується групою проекту).

Структура файлів та каталогів повинна відбивати архітектуру ПЗ, на зразок:

- Backend - .NET технології із хмарним розміщенням (Azure)
- Web - JS, HTML, CSS,або JS фреймворки, зокрема React, Angular
- Mobile - Java, або Kotlin, або Flutter

Команда проєкту

Для виконання поставлених завдань передбачається робота команди розробників окремо за кожним з модулів проекту.
Мінімальна команда проекту - два учасники, у такому складі один учасник виконує модуль Backend та програмну частину Web, другий - модель
Рекомендована команда проекту - три учасники, у такій команді кожен з учасників відповідає за одну з частин: Mobile, Backend, Web. Дозволяється включення до команди більшої кількості учасників, у такому разі модуль Mobile App набуває обов'язкової реалізації.

Звітність

За результатами виконання завдань із створення ПЗ кожен з учасників команди проекту складає власний (індивідуальний) звіт, що складається з практичної частини та автореферату. Практична частина є прямим результатом роботи і подається у вигляді посилання на репозиторій (чи інший URL) або у вигляді архіву на довільному носієві даних. Вихідний код повинен оформлятися у відповідності до загальноприйнятих стандартів, на зразок стандартів GNU http://www.gnu.org/prep/standards/, у тому числі з додержанням вимог до коментування та документування коду.
Автореферат містить описову частину, у якій розкриваються основні концепти проекту, деталізуються завдання до виконавця, встановлюється актуальність роботи, її об'єкт та предмет, обґрунтовується вибір технологій та відзначається шлях виконання та ступінь досягнення поставлених задач, посилання на джерела інформації.

РОЗДІЛ 1
ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

Інформаційна система електронної освіти призначена для забезпечення ефективної комунікації між студентами та викладачами, надання доступу до навчальних матеріалів, управління навчальними процесами та оцінювання. Система включає серверну та клієнтську частини, які взаємодіють для забезпечення всіх необхідних функціональних можливостей. У відповідності до технічного завдання (ТЗ) система була спроектована за клієнт-серверною розподіленою архітектурою. У складі системи було умовно відокремлено наступні архітектурні частини:

- Mobile;
- Frontend;
- Backend;

Клієнтська частина (Mobile): Реалізована для мобільних пристроїв з використанням мови програмування Java та середовища розробки Android Studio. Використання бібліотек Retrofit та OkHttp для здійснення мережевих запитів. Використання бібліотеки Glide для завантаження зображень. Частина “Mobile” представляє собою набір готових екранів з урахуванням кожного елементу користувацького інтерфейсу. На деяких екранах сервісу були реалізовані анімації з урахуванням продуктивності мобільного пристрою та швидкого відгуку від дій користувача для забезпечення максимально комфортного користування додатком.

З метою покращення безпеки, звертаючи увагу на той факт, що мобільні пристрої досить часто підключаються до незахищених мереж передавання даних, інформаційний обмін між частинами «Backend» та «Frontend» захищений попереднім шифруванням сучасним криптоалгоритмом AES-256. Це убезпечує роботу системи у відкритих мережах, а також спрощує задачі автентифікації та доступу до контенту з обмеженим доступом без наявності ключів розшифрування даних, навіть за умови дискредитації інших модулів системи

APK - https://drive.google.com/file/d/1nv5DEJGTh1U3A-ztuaIj1z5-0XSX3zBu/view?usp=sharing

РОЗДІЛ 2
РЕАЛІЗАЦІЯ МОБІЛЬНОГО ЗАСТОСУНКУ

Мобільний додаток розроблений з використанням архітектурного паттерну MVVM (Model-View-ViewModel), який забезпечує розділення відповідальності між різними компонентами додатку. Це дозволяє покращити тестованість та підтримуваність коду. Мобільний застосунок, розроблений у рамках даного проекту, є ключовим компонентом системи електронної освіти. Метою цього застосунку є забезпечення зручного інтерфейсу для користувачів, який дозволяє легко реєструватися, авторизуватися, керувати своїм профілем, взаємодіяти з викладачами та студентами, а також здійснювати інші освітні операції.

Клієнтська частина інформаційної системи реалізована з використанням наступних технологій:

- Операційна система: Android, гарантується підтримка систем з версіями API від 23 до 31 (поточна на час початку створення проєкту).
- Засоби розроблення: мова програмування Java (OpenJDK 64-Bit Server VM by Oracle Corporation) та середовище Android Studio
- Бібліотеки: Retrofit, OkHttp, Glide, Android Material Design

Розробка мобільної частини включала в себе наступні ключові елементи:

1. Екран авторизації та реєстрації - перший екран, який бачить користувач при відкритті додатка. Він містить поля для вводу логіну та паролю, а також кнопки для реєстрації нового користувача і відновлення паролю. Реалізовані функції "запам'ятати мене" та "забули пароль".
2. Екран відновлення паролю - дозволяє користувачу ввести свою електронну адресу для відправлення інструкцій з відновлення паролю.
3. Інтерфейс реєстрації нового користувача - включає поля для вводу необхідної інформації (ім'я, електронна пошта, пароль) та кнопку для завершення реєстрації.
4. Головний екран додатка - після успішної авторизації користувач потрапляє на головний екран, де розміщені основні функції додатка. Він може містити навігацію по додатку, доступ до налаштувань профілю та інші ключові функції.

5. Анімації та переходи - для покращення користувацького досвіду були реалізовані плавні анімації при переході між екранами, а також анімації при взаємодії з елементами інтерфейсу, що створює більш інтуїтивний і приємний інтерфейс.

Початок роботи користувача із додатком супроводжується його реєстрацією як учасника інформаційної системи.

```
public class RegisterRequest {
    private String email;
    private String login;
    private String password;
    private String realName;
    private String repeatPassword;

    public RegisterRequest(String email, String login, String password, String
realName, String repeatPassword) {
        this.email = email;
        this.login = login;
        this.password = password;
        this.realName = realName;
        this.repeatPassword = repeatPassword;
    }
}
```

Реєстрація користувача

Клас RegisterRequest використовується для передачі даних про реєстрацію користувача на сервер. Він містить поля для електронної пошти, логіну, пароля, імені та підтвердження пароля.

При реєстрації користувач обов'язково надає відомості про електронну пошту, через яку буде здійснюватись оповіщення користувача щодо актуальних питань, а також про зміну активності у його особистому просторі. Попередньо до реєстрації введені дані проходять попередню валідацію за допомогою засобів регулярних виразів

```
Pattern pass =
Pattern.compile("(?=.*[0-9])(?=.*[!@#$%^&*])(?=.*[a-z])(?=.*[A-Z])[0-9a-zA-Z!@#$%^
&*]{6,}");
```

```
Pattern name = Pattern.compile("[a-zA-Z]|[а-яА-я]|[а-яА-я]{2,}");
Pattern email = Pattern.compile("^([a-z0-9_\\.-]+)@([a-z0-9_\\.-]+)\\.([a-z\\.]{2,6})$",
Pattern.CASE_INSENSITIVE);
```

Валідація введених даних

Авторизація користувача здійснюється за допомогою пошти та паролю, введенних на етапі реєстрації.

```
public class LoginRequest {
    private String email;
    private String passwordhash;

    public LoginRequest(String email, String passwordhash) {
        this.email = email;
        this.passwordhash = passwordhash;  }}
```

Авторизація користувача

Клас LoginRequest використовується для передачі даних про авторизацію користувача на сервер. Він містить поля для електронної пошти та хешованого пароля.

При реєстрації чи авторизації здійснюється комунікація з частиною «Backend» у результаті чого авторизація підтверджується або спростовується.

```
public interface ApiService   {

        @POST("Home/RegisterUserJson")
        Call<RegisterResponse> registerUser(@Body RegisterRequest
registerRequest);

    @GET("Home/Login")
    Call<LoginResponse> loginUser(@Query("email") String email,
@Query("passwordhash") String passwordhash);
}
```

API для авторизації та реєстрації користувача

Інтерфейс ApiService визначає методи для здійснення мережевих запитів до серверу. Метод registerUser використовується для реєстрації нового користувача, а метод loginUser для авторизації користувача.

Дані, що передаються між зазначеними частинами, шифруються за допомогою алгоритму AES-256 (режим CBC) та перетворюються у транспортне кодування Base64. Це убезпечує дані при передаванні. Для використання у різних частинах системи засоби шифрування реалізовані у вигляді окремого класу

```
public static String encrypt(String strToEncrypt) {
   try {
      byte[] iv = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
      IvParameterSpec ivspec = new IvParameterSpec(iv);
      SecretKeyFactory factory =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
      KeySpec spec = new PBEKeySpec(SECRET_KEY.toCharArray(),
SALT.getBytes(), 65536, 256);
      SecretKey tmp = factory.generateSecret(spec);
      SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");
      Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
      cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivspec);  return
Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes(Stan
dardCharsets.UTF_8)));
   } catch (Exception e) {
      System.out.println("Error while encrypting: " + e.toString());  }
   return null; }
@RequiresApi(api = Build.VERSION_CODES.O)
public static String decrypt(String strToDecrypt) {
   try {
      byte[] iv = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
      IvParameterSpec ivspec = new IvParameterSpec(iv);
      SecretKeyFactory factory =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
      KeySpec spec = new PBEKeySpec(SECRET_KEY.toCharArray(),
SALT.getBytes(), 65536, 256);
      SecretKey tmp = factory.generateSecret(spec);
      SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");
      Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
      cipher.init(Cipher.DECRYPT_MODE, secretKey, ivspec);
      return new
String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
   } catch (Exception e) {
      System.out.println("Error while decrypting: " + e.toString());}  return null; }
```

Шифрування даних

Екран відновлення паролю дозволяє користувачу ввести свою електронну адресу для отримання інструкцій з відновлення паролю. Цей екран складається з текстового поля для введення електронної адреси та кнопки для надсилання запиту на відновлення паролю

```
public class ForgotPasswordActivity extends AppCompatActivity {

    private EditText emailInput;
    private Button sendButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_forgot_password);
        emailInput = findViewById(R.id.emailInput);
        sendButton = findViewById(R.id.sendButton);
        sendButton.setOnClickListener(v -> {
            String email = emailInput.getText().toString().trim();
            if (!email.isEmpty()) {
                sendPasswordResetEmail(email);
            } else {
                Toast.makeText(ForgotPasswordActivity.this, "Будь ласка, введіть
вашу електронну адресу", Toast.LENGTH_SHORT).show();
            }
        });
    }
    private void sendPasswordResetEmail(String email) {
        ApiService apiService = RetrofitClient.getApiService();
        Call<Void> call = apiService.sendPasswordResetEmail(email);
        call.enqueue(new Callback<Void>() {
            @Override
            public void onResponse(Call<Void> call, Response<Void> response) {
                if (response.isSuccessful()) {
                    Toast.makeText(ForgotPasswordActivity.this, "Інструкції для
відновлення паролю надіслано на вашу електронну адресу",
Toast.LENGTH_LONG).show();
                } else {
                    Toast.makeText(ForgotPasswordActivity.this, "Помилка при
надсиланні інструкцій. Спробуйте знову", Toast.LENGTH_LONG).show();
                }
            }
```

```
        @Override
        public void onFailure(Call<Void> call, Throwable t) {
            Toast.makeText(ForgotPasswordActivity.this, "Сталася помилка: " +
t.getMessage(), Toast.LENGTH_LONG).show();
        }
    });
  }
}
```

Код відновлення паролю

Клас ForgotPasswordActivity відповідає за логіку екрану відновлення паролю. Він отримує введену користувачем електронну адресу та надсилає запит на сервер для відновлення паролю через API. Відповіді сервера обробляються, і користувач отримує відповідні повідомлення про успішне або невдале надсилання інструкцій.

Метод sendPasswordResetEmail використовує Retrofit для надсилання запиту на сервер та обробляє відповіді сервера і відображає відповідні повідомлення користувачу.

Інтерфейс ApiService визначає метод sendPasswordResetEmail, який надсилає POST-запит для відновлення паролю. Цей метод використовується в ForgotPasswordActivity для взаємодії з сервером.

```
public interface ApiService {
  @POST("Home/ForgotPassword")
  Call<Void> sendPasswordResetEmail(@Query("email") String email);}
```

Інтерфейс ApiService, який визначає sendPasswordResetEmail

Після успішної авторизації користувач потрапляє на головний екран, де розміщені основні функції додатка

.

Головний екран додатка реалізований за допомогою BottomNavigationView, що дозволяє користувачам перемикатися між різними фрагментами. Він містить навігаційне меню для доступу до різних розділів додатка, таких як профіль, налаштування та інше. Користувач може перемикатися між різними розділам, а також мати доступ до налаштувань профілю та можливості виходу з облікового запису.

```
package com.example.myapplication;
import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentTransaction;
import com.google.android.material.bottomnavigation.BottomNavigationView;
public class HomeActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_home);
      BottomNavigationView bottomNavigationView =
findViewById(R.id.bottom_navigation);
      bottomNavigationView.setOnNavigationItemSelectedListener(item -> {
         Fragment selectedFragment = null;
         int itemId = item.getItemId();
         if (itemId == R.id.navigation_search) {
            selectedFragment = new SearchFragment();
         } else if (itemId == R.id.navigation_messages) {
            selectedFragment = new MessagesFragment();
         } else if (itemId == R.id.navigation_schedule) {
            selectedFragment = new ScheduleFragment();
         } else if (itemId == R.id.navigation_vocab) {
            selectedFragment = new VocabFragment();
         } else if (itemId == R.id.navigation_profile) {
            selectedFragment = new ProfileFragment();
         }

         if (selectedFragment != null) {
            FragmentTransaction transaction =
getSupportFragmentManager().beginTransaction();
            transaction.replace(R.id.fragment_container, selectedFragment);
            transaction.commit();
         }
```

```
        return true;
    });
    if (savedInstanceState == null) {
        bottomNavigationView.setSelectedItemId(R.id.navigation_search); } }}
```

Код головного екрану

Реалізація профілю в мобільному застосунку включає можливість редагування особистої інформації користувача, завантаження та відображення фотографії профілю. Це реалізовано у фрагменті EditProfileFragment, який дозволяє користувачу змінювати своє ім'я та фотографію профілю, а також зберігати ці зміни.

```
package com.example.myapplication;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import com.bumptech.glide.Glide;
public class EditProfileFragment extends Fragment {
    private static final int PICK_IMAGE_REQUEST = 1;
    private ImageView profileImageEdit;
    private EditText profileNameEdit;
    private Button saveProfileButton;
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_edit_profile, container,
false);
```

```
        profileImageEdit = view.findViewById(R.id.profile_image_edit);
        profileNameEdit = view.findViewById(R.id.edit_profile_name);
        saveProfileButton = view.findViewById(R.id.save_profile_button);

        // Відкрити вибір зображення при натисканні на текст "Змінити фото"
        view.findViewById(R.id.change_photo_text).setOnClickListener(v ->
openImageChooser());

            saveProfileButton.setOnClickListener(v -> saveProfile()); return view;}
    private void openImageChooser() {
        Intent intent = new Intent();
        intent.setType("image/*");
        intent.setAction(Intent.ACTION_GET_CONTENT);
        startActivityForResult(Intent.createChooser(intent, "Select Picture"),
PICK_IMAGE_REQUEST);
    }
    @Override
    public void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == PICK_IMAGE_REQUEST && resultCode ==
getActivity().RESULT_OK && data != null && data.getData() != null) {
            Uri imageUri = data.getData();
            Glide.with(this).load(imageUri).into(profileImageEdit);
        }
    }
    private void saveProfile() {
        Toast.makeText(getActivity(), "Профіль збережено",
Toast.LENGTH_SHORT).show();
    }
}
```

EditProfileFragment.java

Цей фрагмент дозволяє користувачу редагувати свій профіль. Він включає можливість змінити фотографію профілю та ім'я. При натисканні на текст "Змінити фото" відкривається вибір зображення з галереї. Після вибору зображення воно відображається у вигляді. При натисканні на кнопку "Зберегти" зберігаються зміни (на даний момент це лише показ повідомлення "Профіль збережено").

Для взаємодії з сервером у проекті використовується бібліотека Retrofit.

```
package com.example.myapplication.network;
import okhttp3.OkHttpClient;
import okhttp3.logging.HttpLoggingInterceptor;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;
import com.example.myapplication.api.ApiService;
public class RetrofitClient {

   private static final String BASE_URL =
"https://e-educator.azurewebsites.net/";
   private static Retrofit retrofit = null;
   public static ApiService getApiService() {
      if (retrofit == null) {
         HttpLoggingInterceptor logging = new HttpLoggingInterceptor();
         logging.setLevel(HttpLoggingInterceptor.Level.BODY);
         OkHttpClient client = new OkHttpClient.Builder()
               .addInterceptor(logging)
               .build();
         retrofit = new Retrofit.Builder()
               .baseUrl(BASE_URL)
               .client(client)
               .addConverterFactory(GsonConverterFactory.create())
               .build();      }
      return retrofit.create(ApiService.class);  }}
```

RetrofitClient.java

Клас RetrofitClient відповідає за налаштування та створення клієнта Retrofit, який використовується для виконання HTTP-запитів до серверу. Клас RetrofitClient забезпечує створення та налаштування об'єкта Retrofit лише один раз (використовуючи паттерн сінглтон), що запобігає зайвому створенню об'єктів і покращує продуктивність.

РОЗДІЛ 3
РЕЗУЛЬТАТИ ВИПРОБУВАНЬ

Результатом компіляції програмного коду клієнтської частини являється інсталяційний (APK) файл додатку, що може бути встановлений на смартфон або планшет з операційною системою «Android». Головна серія випробувань, звіт з якої наведено далі, проведена на смартфоні «Samsung Galaxy S23 Plus» з операційною системою «Android 14».

Початок роботи нового користувача з додатком починається з реєстрації користувача у системі. Під час реєстрації здійснюється перевірка засобів комунікації - на електронну пошту надсилається код підтвердження, який має бути введений для завершення етапу реєстрації.

Якщо користувач раніше проходив реєстрацію, то він може увійти до системи за допомогою персональних даних, зазначених при реєстрації. Зовнішній вигляд екранів реєстрації та входу наведено на рис. 1.

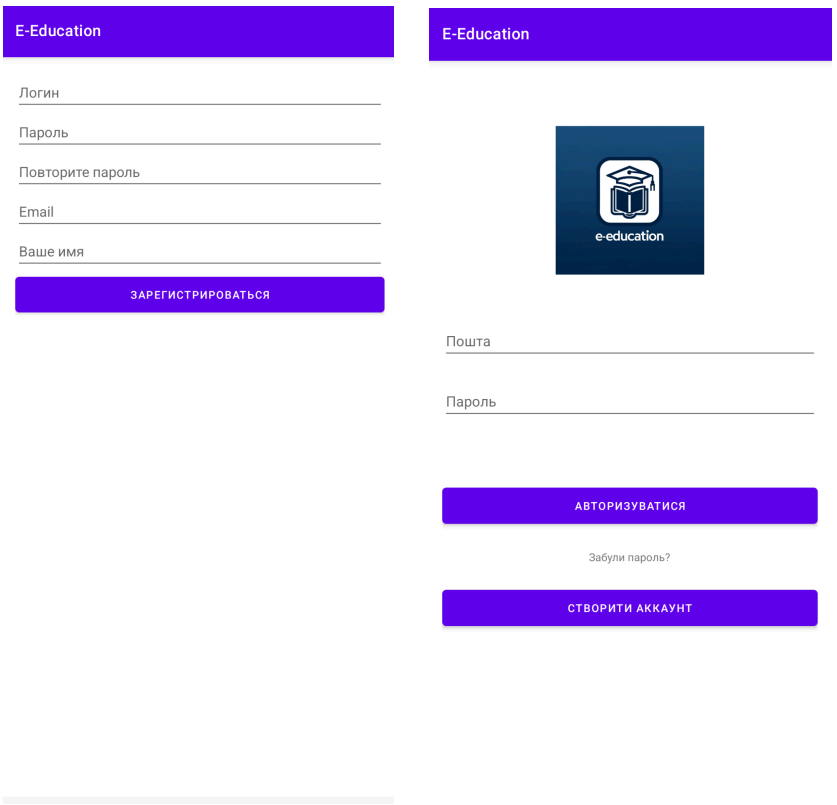

Рисунок 1 - Реєстрація користувача та вхід

Після входу до системи користувач отримує доступ до основних функцій системи. Зовнішній вигляд екранів наведено на рисунках 2.1-3.

Екран повідомлень забезпечує користувачів можливістю переглядати та надсилати повідомлення. Він включає в себе пошуковий рядок для швидкого пошуку повідомлень, а також вкладки для відображення всіх повідомлень або лише архівованих.

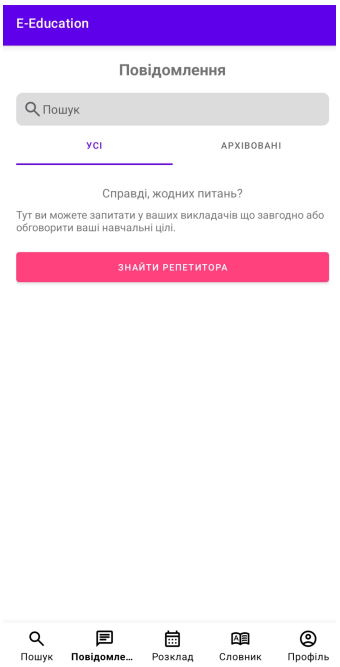

Рисунок 2.1 - Основні функції системи

Користувачі можуть легко переключатися між вкладками, щоб знайти потрібне повідомлення. На головному екрані повідомлень також відображається запрошення до пошуку репетитора, якщо немає активних повідомлень.

Екран профілю забезпечує користувачів можливістю переглядати та редагувати особисту інформацію. Він включає в себе фото профілю та ім'я користувача, кнопку для редагування профілю, а також різні налаштування, допомогу та додаткові опції. Користувачі можуть легко редагувати свої особисті дані за допомогою кнопки "Редагування профілю".

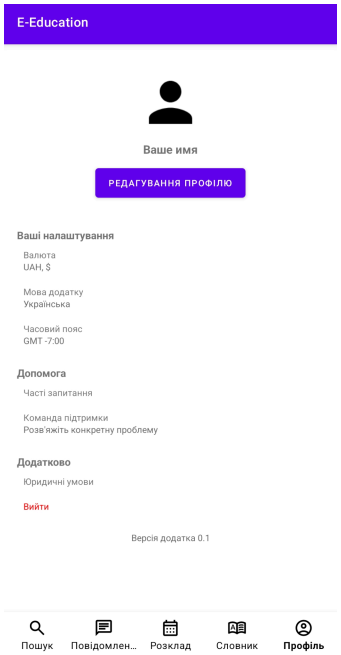

Рисунок 2.3 - Основні функції системи

Налаштування дозволяють користувачам вибирати бажану валюту, мову додатка та встановлювати часовий пояс, мову додатка та встановлювати часовий пояс. Розділ допомоги забезпечує доступ до частих запитань та команди підтримки для вирішення можливих проблем.

Екран розкладу забезпечує користувачів можливістю переглядати свої заплановані уроки та заняття. Він включає в себе основний текст, який інформує користувача про його розклад, а також кнопку для пошуку репетитора.

Рисунок 2.3 - Основні функції системи

Користувачі можуть легко переглядати свій розклад, щоб бути в курсі запланованих уроків. Якщо користувач не має запланованих уроків, він бачить відповідне повідомлення та має можливість скористатися кнопкою для пошуку репетитора. Це дозволяє швидко знайти потрібного викладача та запланувати нові заняття.

ВИСНОВКИ

Проведено огляд існуючих інформаційних систем, що забезпечують взаємодію викладачів та студентів у контексті електронної освіти. Виявлено, що важливу роль у таких системах відіграють функції реєстрації та авторизації користувачів, можливості редагування профілю, управління розкладом занять та обміну повідомленнями. Саме ці функції були визначені як ключові для реалізації у розробленій системі.

Визначено та впроваджено сучасні технології, що дозволяють забезпечити безпечну та ефективну роботу системи. Для клієнтської частини було обрано платформу Android з використанням мови програмування Java та Kotlin. Серед основних бібліотек, що були використані, варто виділити Retrofit для роботи з мережевими запитами, OkHttp для обробки HTTP-запитів, Gson для серіалізації та десеріалізації даних, а також Glide для завантаження та відображення зображень.

На серверній частині системи використано платформу ASP.NET Core для створення RESTful API та Entity Framework Core для доступу до бази даних. Для зберігання даних обрано Azure Cosmos DB, що забезпечує високу надійність та масштабованість системи. Особливу увагу приділено безпеці передачі даних, для чого застосовано шифрування за алгоритмом AES-256.

Система побудована за розподіленою клієнт-серверною архітектурою та складається з трьох основних компонентів:

- **Клієнтська частина (мобільний додаток)**: доступна на платформі Android.
- **Серверна частина**: розміщена на платформі Azure.
- **База даних**: реалізована на Azure Cosmos DB.

Система забезпечує можливість реєстрації та авторизації користувачів, редагування профілю, управління розкладом занять, обмін повідомленнями між викладачами та студентами, а також пошук репетиторів.

Систему було випробувано шляхом інсталяції на фізичні пристрої та проведення тестових сеансів комунікації. Результати випробувань підтвердили загальну працездатність створеної системи та відповідність її заявленим вимогам.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Android Studio. URL: https://developer.android.com/studio
2. Java Development Kit (JDK). URL: https://www.oracle.com/java/technologies/javase-jdk11-downloads.html
3. Kotlin Programming Language. URL: https://kotlinlang.org/
4. Retrofit. URL: https://square.github.io/retrofit/
5. OkHttp. URL: https://square.github.io/okhttp/
6. Gson. URL: https://github.com/google/gson
7. Glide. URL: https://bumptech.github.io/glide/
8. Azure Cosmos DB. URL: https://azure.microsoft.com/services/cosmos-db/
9. AES-256. URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf
10. Стратегія розвитку інформаційного суспільства в Україні. Розпорядження Кабінету Міністрів України від 15 травня 2013 р. № 386-р. // Урядовий портал. URL: https://www.kmu.gov.ua/npas/246420577
11. Стратегія кібербезпеки України. Указ Президента України від 14 травня 2021 року // Урядовий кур'єр від 28.08.2021 — № 165. URL: https://zakon.rada.gov.ua/laws/show/447/2021/print
12. Решетило В.П., Федотова Ю.В. Невизначеність та ризик: співвідношення понять та специфіка прийняття рішень // Проблеми системного підходу в економіці, випуск № 3(77)-2, 2020. с. 149. URL: http://psae-jrnl.nau.in.ua/journal/3_77_2_2020_ukr/23.pdf
13. Олександр Смичніков. Параноя або сучасні реалії: як зберегти конфіденційність у мережі. - ФЬЮЧЕР МЕДІА. URL: https://mind.ua/openmind/20183329-paranoya-abo-suchasni-realiyi-yak-zberegti-konfidencijnist-u-merezhi (дата звернення 30.05.2022)
14. Булгакова О.С. Методи та системи штучного інтелекту: теорія та практика. Навчальний посібник / Булгакова О.С., Зосімов В.В., Поздєєв В.О. - Херсон. - 2020. - 356 с. ISBN: 978-966-289-364-9