НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

КАФЕДРА АВТОМАТИЗОВАНИХ СИСТЕМ ОБРОБКИ ІНФОРМАЦІЇ ТА УПРАВЛІННЯ

**КУРСОВА РОБОТА**

з дисципліни «Сучасні технології програмування - 3. Технології створення програмних продуктів для мобільних платформ»

на тему: «Розробка соціальної мережі з елементами листування у вигляді чату»

Студента 3 курсу ІП-71 групи

напряму підготовки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

спеціальності\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| Члени комісії |  |  |  |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |
|  |  |  |  |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |

Київ - 2020 рік

Національний технічний університет України “КПІ”

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра автоматизованих систем обробки інформації і управління

Дисципліна «Сучасні технології програмування - 3. Технології створення програмних продуктів для мобільних платформ»

Спеціальність 121 "Інженерія програмного забезпечення"

Курс 3 Група ІП-71 Семестр 6

**ЗАВДАННЯ**

**на курсову роботу студента**

|  |
| --- |
| Фоміна Владислава Віталійовича |

(прізвище, ім’я, по батькові)

|  |  |
| --- | --- |
| 1. Тема роботи | Розробка соціальної мережі з елементами листування у вигляді чату |
|  | |
|  | |

|  |  |
| --- | --- |
| 2. Строк здачі студентом закінченої роботи | 15.05.2020 |

|  |  |
| --- | --- |
| 3. Вихідні дані до роботи | Макет програмного продукту, архітектура бази даних, архітектура бек-енду |
|  | |
|  | |
|  | |

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

|  |
| --- |
| Вибір ідеї, вибір інструментарію, створення інтерактивного прототипу застосунку, |
| архітектура застосунку, імплементація UI застосунку, імплементація бізнес-логіки користувача, |
| імплементація взаємодії з мережею, імплементація зберігання даних локально, |
| імплементація взаємодії з hardware пристрою, імплементація тестів |

|  |  |
| --- | --- |
| 5. Дата видачі завдання | 20.03.2020 |

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів курсової роботи | Термін виконання етапів роботи | Підписи керівника, студента |
| 1. | Вибір ідеї, вибір інструментарію | 21.03.2020 |  |
| 2. | Створення інтерактивного прототипу застосунку | 01.04.2020 |  |
| 3. | Архітектура застосунку | 15.04.2020 |  |
| 4. | Імплементація UI застосунку | 25.04.2020 |  |
| 6. | Імплементація бізнес-логіки користувача | 30.04.2020 |  |
| 5. | Імплементація взаємодії з мережею | 01.05.2020 |  |
| 7. | Імплементація зберігання даних локально | 03.05.2020 |  |
| 8. | Імплементація взаємодії з hardware пристрою | 10.05.2020 |  |
| 9. | Імплементація тестів | 14.05.2020 |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  |  |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

|  |  |  |  |
| --- | --- | --- | --- |
| Керівник |  |  | Іванова Л.А. |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

**АНОТАЦІЯ**

Курсова робота «Розробка соціальної мережі з елементами листування у вигляді чату» направлена на розробку програмного продукту, який надасть можливість користувачам комунікувати між собою. Для розробки було обрано наступні технології:

1. Інтерфейс користувача – Xamarin;
2. Серверна частина – ASP.NET Core 3.0;
3. База даних – SQL Server;
4. База даних на пристрої користувача – SQLite.

У пояснювальній записці буде описано процес розробки інтерфейсу користувача вищеописаного програмного продукту.

Результатами виконаної роботи буде програмний продукт у вигляді соціальної мережі з можливістю комунікації між користувачами у вигляді он-лайн чату.

**ЗМІСТ**

[**ВСТУП** 6](#_Toc41320266)

[**ОСНОВНА ЧАСТИНА** 7](#_Toc41320267)

[**1 ВИБІР ІДЕЇ ТА ІНСТРУМЕНТАРІЮ** 7](#_Toc41320268)

[**2 СТВОРЕННЯ ІНТЕРАКТИВНОГО ПРОТОТИПУ ЗАСТОСУНКУ** 8](#_Toc41320269)

[**3 ІМПЛЕМЕНТАЦІЯ UI ЗАСТОСУНКУ** 17](#_Toc41320270)

[**4 ІМПЛЕМЕНТАЦІЯ ВЗАЄМОДІЇ З МЕРЕЖЕЮ** 27](#_Toc41320271)

[**5 ІМПЛЕМЕНТАЦІЯ ЗБЕРІГАННЯ ДАНИХ ЛОКАЛЬНО** 28](#_Toc41320272)

[**6 ІМПЛЕМЕНТАЦІЯ ВЗАЄМОДІЇ З HARDWARE ПРИСТРОЮ** 29](#_Toc41320273)

[**ВИСНОВОК** 32](#_Toc41320274)

[**ПЕРЕЛІК ПОСИЛАНЬ** 33](#_Toc41320275)

[**ДОДАТКИ** 34](#_Toc41320276)

[**ДОДАТОК А** 34](#_Toc41320277)

[**ДОДАТОК Б** 35](#_Toc41320278)

[**ДОДАТОК В** 36](#_Toc41320279)

[**ДОДАТОК Г** 39](#_Toc41320280)

### **ВСТУП**

Обраний вектор розробки – розробка інтерфейсу користувача.

В процесі виконання даної курсової роботи були досліджені та використані новітні практики UI/UX для розробки максимально комфортного у користуванні інтерфейсу. Для проектування даного інтерфейсу було створено макет за допомогою стороннього програмного забезпечення під назвою Figma.

Отриманий результат проходив тестування двох видів:

1. Мануальне тестування;
2. Юніт тести з серверної сторони.

При розробці програмного коду, який забезпечує реєстрацію та аутентифікацію користувача було створено кастомний варіант шифрування паролю. Паролі в базі даних зберігаються у зашифрованому вигляді, що забезпечило безпеку особистих даних кожного з користувачів. Також кожному користувачу надається особистий токен авторизації, що сприяє підвищенню рівня безпеки.

### **ОСНОВНА ЧАСТИНА**

## **1 ВИБІР ІДЕЇ ТА ІНСТРУМЕНТАРІЮ**

У якості інструментів для розробки серверної частини застосунку було обрано ASP.NET Core 3.0 - вільне та відкрите програмне забезпечення каркаса вебзастосунків, з продуктивністю вищою ніж у ASP.NET, розроблена корпорацією Microsoft і співтовариством. Це модульна структура, яка працює як на повній платформі .NET Framework, так і на платформі .NET Core. Фреймворк являє собою повний перепис, який об'єднує раніше окремі ASP.NET MVC та ASP.NET Web API у єдину програмувальну модель.

Для бази даних було обрано продукт MS SQL - систему управління базами даних, яка розробляється корпорацією Microsoft. Як сервер даних виконує головну функцію по збереженню та наданню даних у відповідь на запити інших застосунків, які можуть виконуватися як на тому ж самому сервері, так і у мережі.

Для розробки фронт-енд частини застосунку обрано Xamarin - представляє платформу, яка націлена на створення кроссплатформенних додатків під Android, iOS і Windows 10. Xamarin дозволяє створювати одну едінcтвенную логіку додатка із застосуванням C # і .NET відразу для всіх трьох платформ - Android, iOS, UWP. Xamarin надає прямий доступ до нативним API кожної платформи.

У якості бази даних на пристрої користувачів було обрано SQLite - полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92. Сирцевий код SQLite поширюється як суспільне надбання, тобто може використовуватися без обмежень та безоплатно з будь-якою метою. Фінансову підтримку розробників SQLite здійснює спеціально створений консорціум, до якого входять такі компанії, як Adobe, Oracle, Mozilla, Nokia, Bentley і Bloomberg.

Для отримання повідомлень в режимі реального часу обрано SignalR - це безкоштовна бібліотека програмного забезпечення з відкритим кодом для Microsoft ASP.NET, яка дозволяє коду сервера надсилати асинхронні сповіщення веб-додаткам на стороні клієнта. Бібліотека включає компоненти JavaScript на стороні сервера та клієнта.



Рисунок 1. Використані під час розробки технології

## **2 СТВОРЕННЯ ІНТЕРАКТИВНОГО ПРОТОТИПУ ЗАСТОСУНКУ**

Для розробки прототипу застосунку було обрано інструмент Figma - векторний онлайн-сервіс розробки інтерфейсів та прототипування з можливістю організації спільної роботи, що розробляється однойменною компанією. Працює у двох форматах: у браузері та як клієнтський додаток на десктопі користувача. Зберігає онлайн-версії файлів, з якими працював користувач.

Спочатку було створено прототипи сторінок авторизації та реєстрації.



Рисунок 2. Сторінка авторизації

Зверху знаходиться логотип програмного застосунку – ZakrityPredmetChat. Нижче знаходиться два віконця для вводу свого логіну та паролю та кнопка «Sign in» для того, щоб успішно залогінитись в системі.

Якщо ж ми не маємо аккаунта в системі, то необхідно зареєструватися, натиснувши кнопку «Sign up», яка знаходиться у правому нижньому куті.

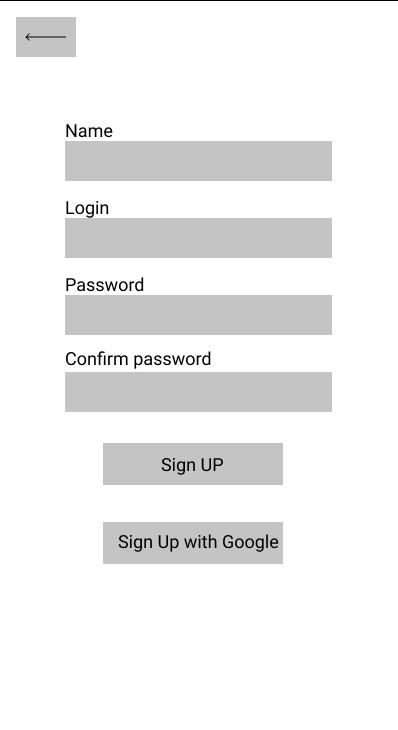


Рисунок 3. Сторінка реєстрації

Зверху зліва знаходиться кнопка для повернення на попередню сторінку у разі відміни дії. Нижче знаходяться поля для заповнення власною інформацією: ім’я (нікнейм), логін, пароль та підтвердження паролю. Коли всі ці поля будуть заповнені – необхідно натиснути кнопку «Sign up». Якщо ж необхідно зареєструватися за допомогою сервісу Google – слід натиснути кнопку «Sign up with Google».



Рисунок 4. Сторінка списку чатів

Зверху знаходиться навігаційне меню, за допомогою якого можна серфити між головними сторінками програми. Виділено 3 основні сторінки: список чатів, профіль користувача та налаштування.

Нижче знаходиться список чатів, у яких виділено фото співбесідника, його нікнейм, останнє надіслане смс та дату цього смс.

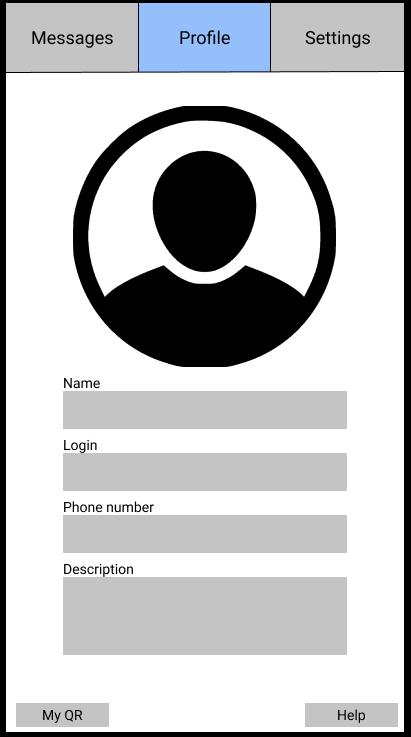


Рисунок 5. Профіль користувача

В центрі екрану знаходиться фото користувача. Нижче – список полів, які користувач заповнює. В нижньому лівому куті знаходиться кнопка переходу до особистого унікального QR-коду. Зправа – кнопка переходу на сторінку допомоги.

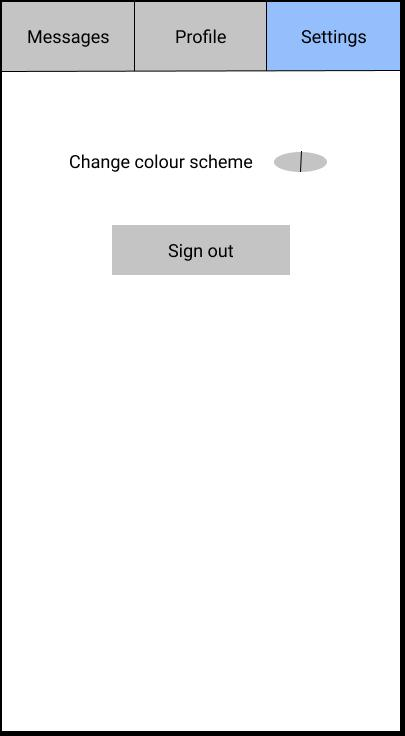


Рисунок 6. Сторінка налаштувань

На даній сторінці знаходиться кнопка зміни кольорової схеми з світлої на темну і навпаки. Нижче розташована кнопка виходу із аккаунта.

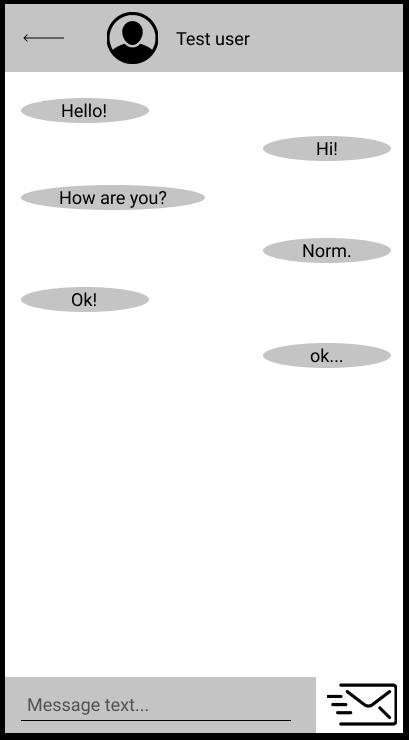


Рисунок 7. Сторінка листування

У верхній частині знаходиться кнопка повернення на попередню сторінку, фото та нікнейм користувача. Нижче розташований власне чат. Справа – повідомлення власника аккаунта. Зліва – повідомлення співбесідника.

У нижній частині сторінки розташоване місце для вводу смс та правіше – кнопка відправки повідомлення.

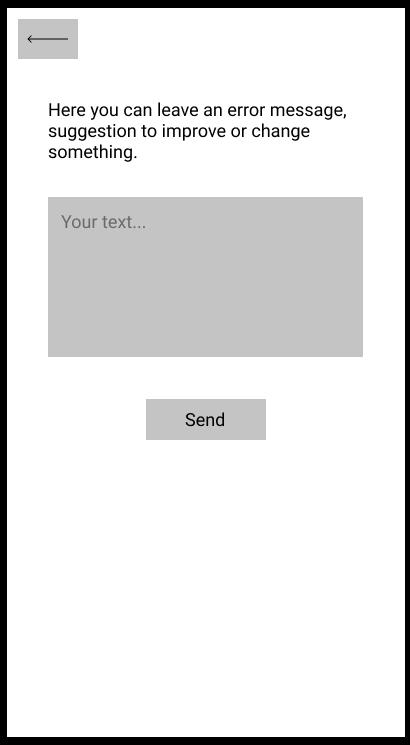


Рисунок 8. Сторінка саппорту

Ця сторінка створена для зв'язку з розробниками та надання їм необхідної інформації про баги, помилки та пропозиції по вдосконаленню.

Зверху розташована кнопка повернення на попередню сторінку.

Нижче текст про цю сторінку та віконце для вводу повідомлення розробникам. Щоб його відправити необхідно натиснути кнопку «Send».



Рисунок 9. Особистий QR-код

На цій сторінці відображений особистий QR-код користувача, який створений для полегшення пошуку необхідних людей у системі. Вистачить просто відсканувати цей код.

## **3 ІМПЛЕМЕНТАЦІЯ UI ЗАСТОСУНКУ**

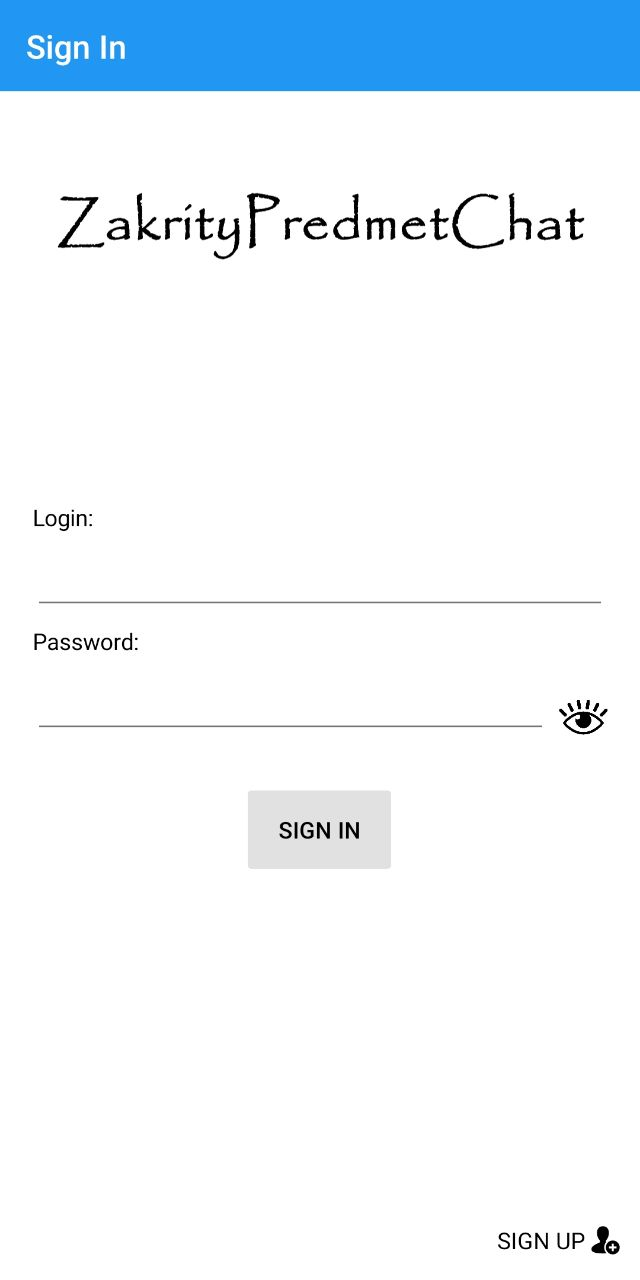


Рисунок 10. Сторінка авторизації

При заповненні обох полів необхідно натиснути кнопку авторизації, після чого запит буде відправлений на сервер та отримано токен доступу. Іконка ока дозволяє ввімкнути видимість паролю та вимкнути її при необхідності. У разі необхідності в реєстрації аккаунта необхідно натиснути на кнопку реєстрації, після чого буде створено Navigation page, на якому буде відображена сторінка реєстрації.

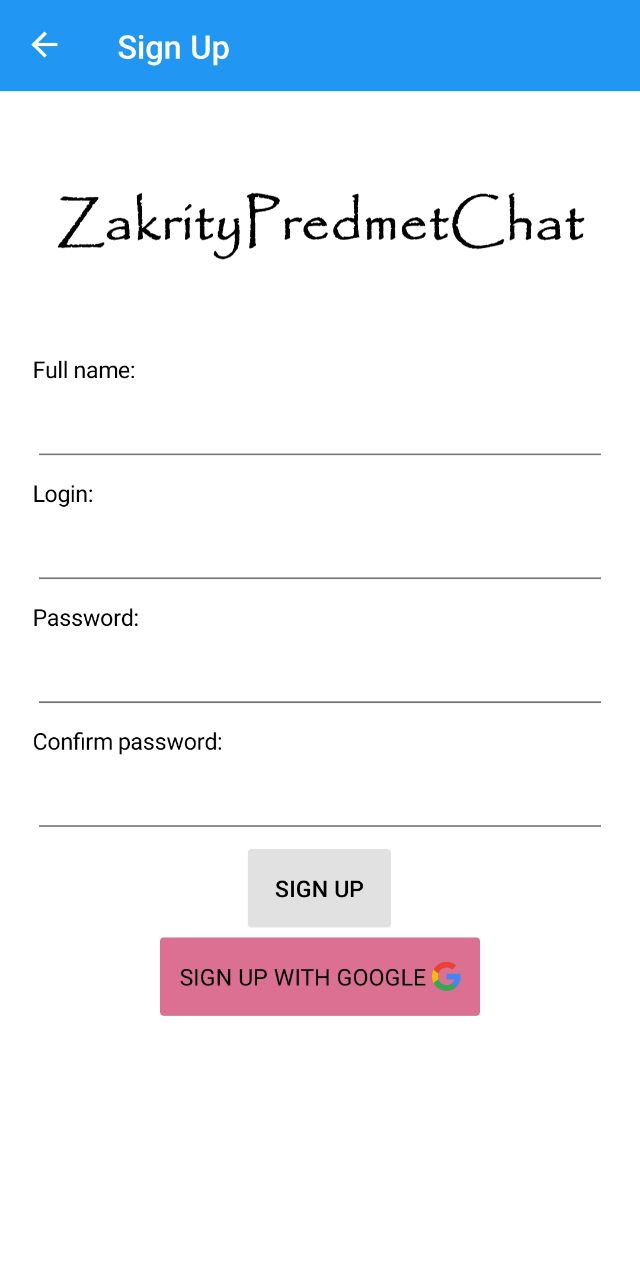


Рисунок 11. Сторінка реєстрації

Для реєстрації аккаунта необхідно заповнити всі ці поля та натиснути кнопку реєстрації. Існують обмеження на дані в полях:

* В полі нікнейму має бути як мінімум 1 символ;
* В полі логін має бути електронна пошта або номер телефону;
* Поле пароль повинне мати як мінімум 8 символів, які мають складатися з цифр, літер великого та малого регістру та спецсимволів.

Є можливість реєстрації за допомогою Google. Необхідно лиш натиснути відповідну кнопку.

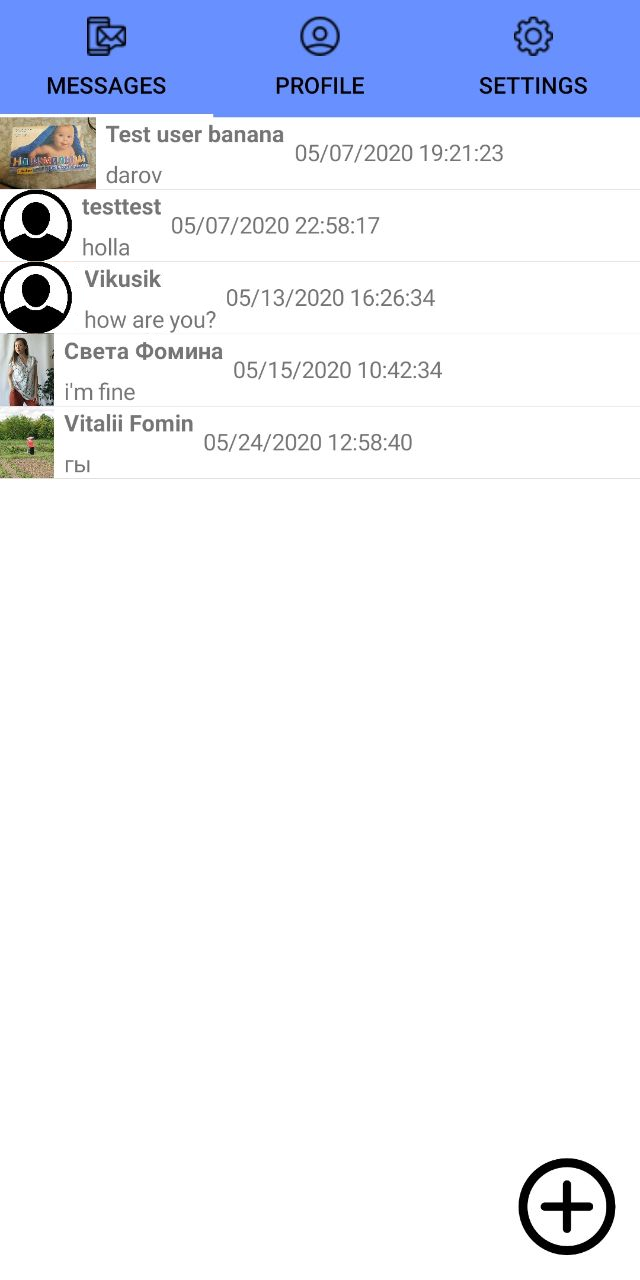


Рисунок 12. Сторінка списку чатів

Відображається актуальний список чатів. Для оновлення необхідно потягнути сторінку донизу. При натисканні на будь-який чат буде відкрито навігейшн пейдж з відображенням переписки з обраним користувачем. У разі натискання на плюсик буде відкрита сторінка пошуку нового користувача.

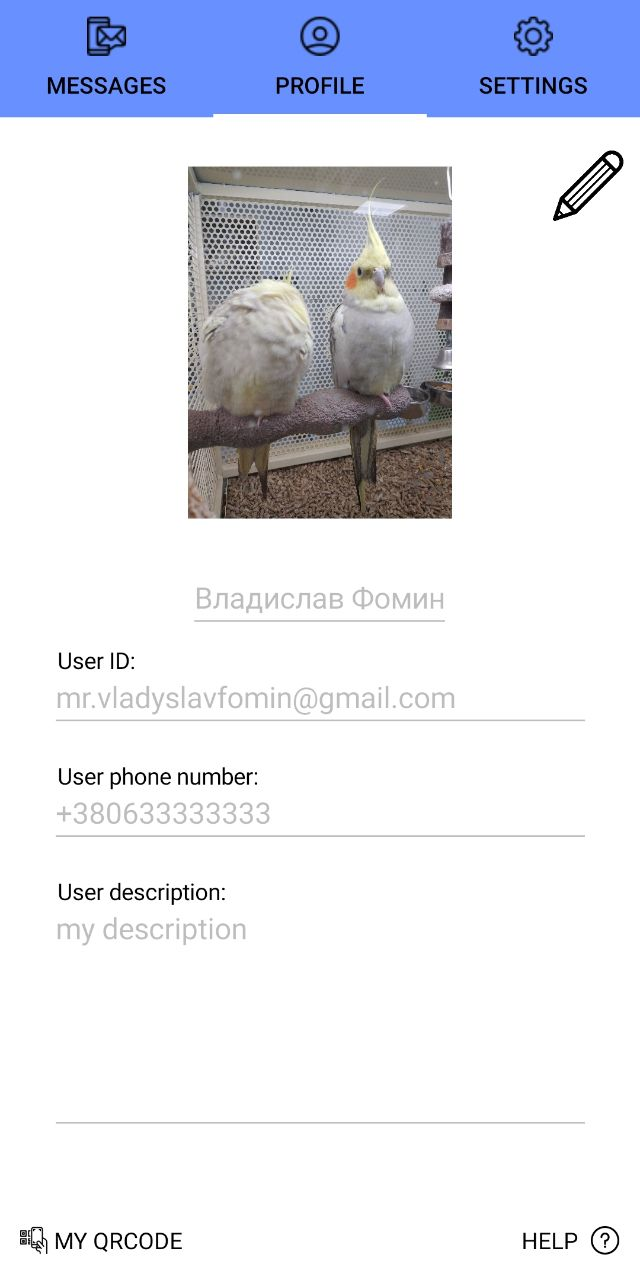


Рисунок 13. Сторінка профілю користувача

Сторінка профілю користувача дозволяє редагування даних про користувача методом натискання на іконку олівця. Тоді всі поля будуть доступні для редагування, а іконка фотографії зміниться на відповідну кнопку обирання нового фото.

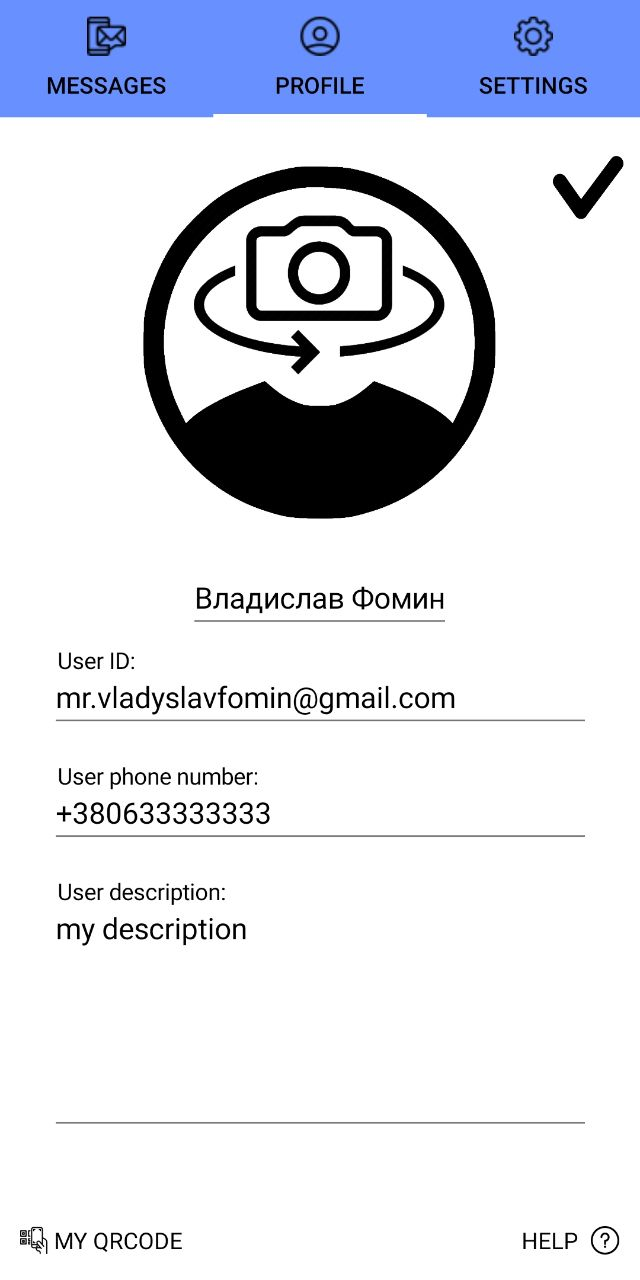


Рисунок 14. Сторінка профілю при редагуванні

При редагуванні даних користувача необхідно ввести дані та натиснути на кнопку збереження даних. При успішній валідації буде створено запит на сервер для заміни старих даних новими. Кнопка кьюар коду внизу зліва відобразить особистий кьюар код користувача. Кнопка допомоги внизу справа відкриє сторінку саппорту.

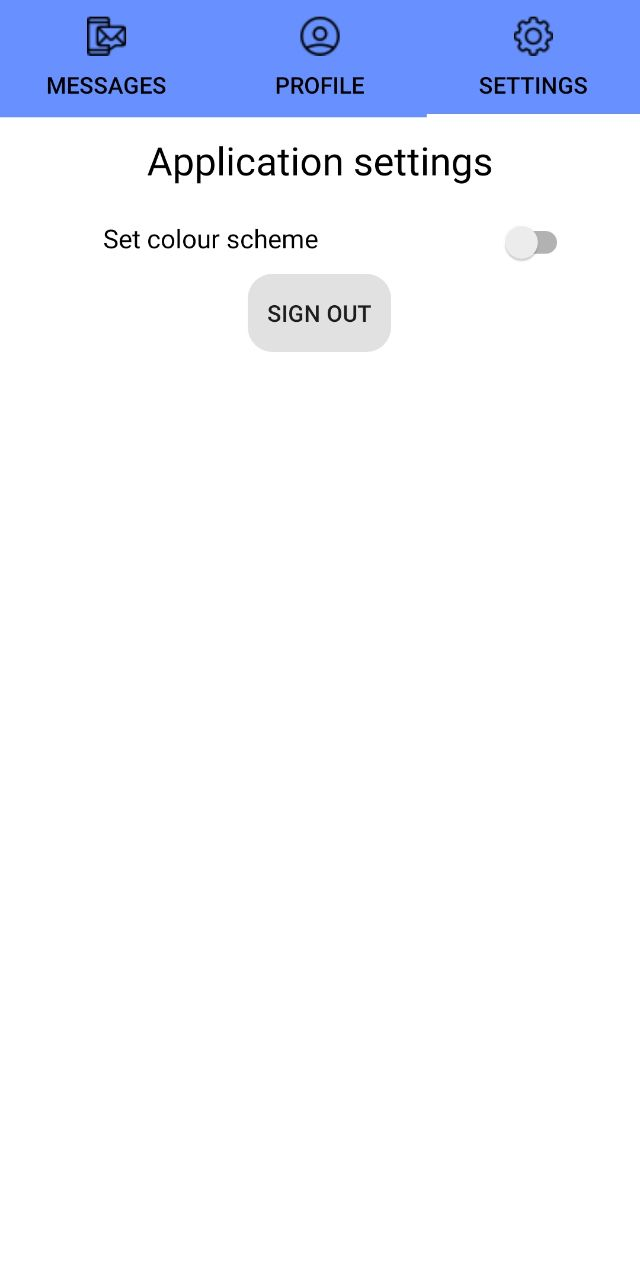


Рисунок 15. Сторінка налаштувань

Сторінка налаштувань дозволяє змінювати кольорову схему шляхом зміни позиції тумблера. Нижче розташована кнопка виходу з аккаунта. Ця дія змінить сторінку на сторінку авторизації.

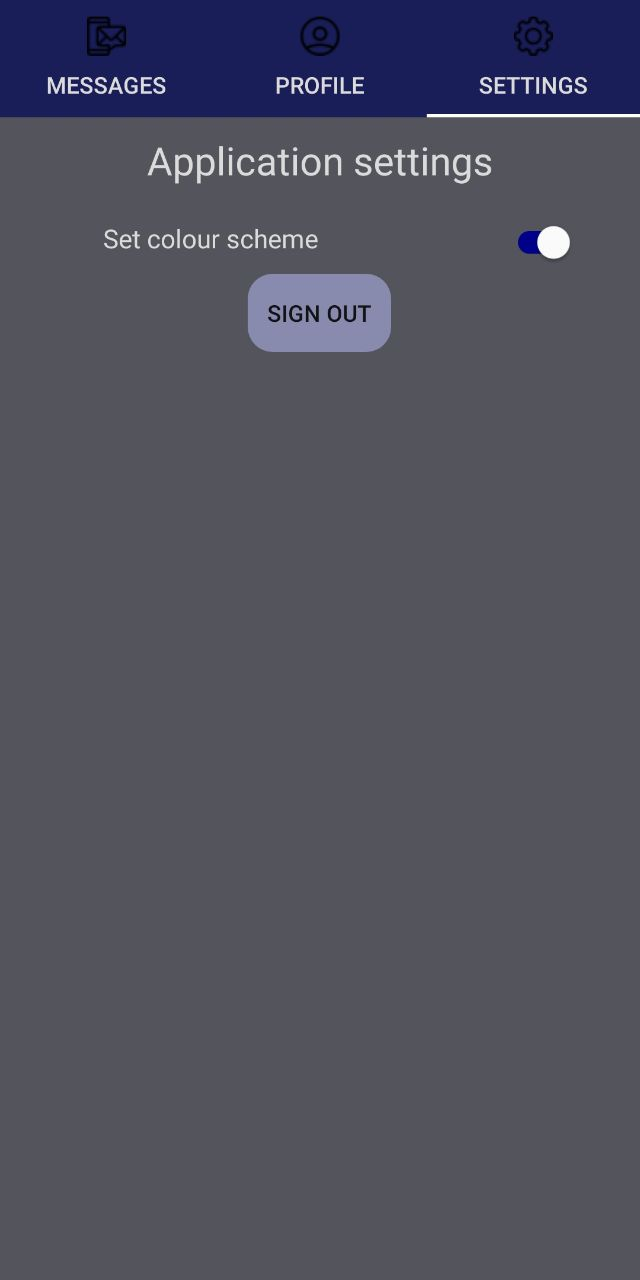


Рисунок 16. Сторінка налаштувань з темною схемою

Вище зображено вигляд темної схеми, яка вмикається та вимикається шляхом зміни положення тумблера.



Рисунок 17. Особистий QR-код

Сторінка особистого унікального кьюар коду генерує його з ідентифікатору користувача та відображає результат на екрані. За допомогою цього можна з легкістю знайти користувача в системі.

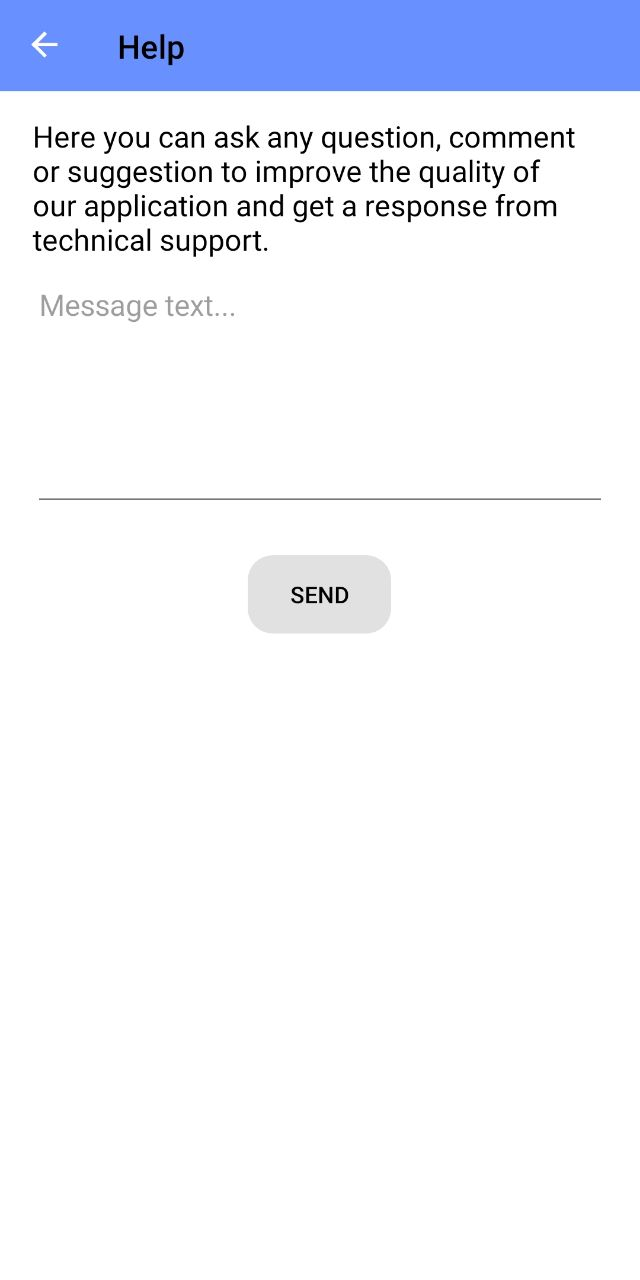


Рисунок 18. Сторінка саппорту

Сторінка саппорту реалізує зв’язок з розробниками. Вона дозволяє надіслати інфрмацію про баги, помилки та пропозиції по вдосконаленню програмного продукту шляхом введення в текстове поле та натисканням кнопки відправки.



Рисунок 19. Сторінка пошуку користувача

Сторінка пошуку користувача має два варіанти пошуку:

* Пошук за введеною інформацією. Для цього необхідно ввести логін або нікнейм користувача у текстове поле та натиснути кнопку пошуку.
* Пошук за кьюар кодом. Для цього необхідно натиснути на іконку пошуку за кьюаром та відсканувати кьюар користувача.

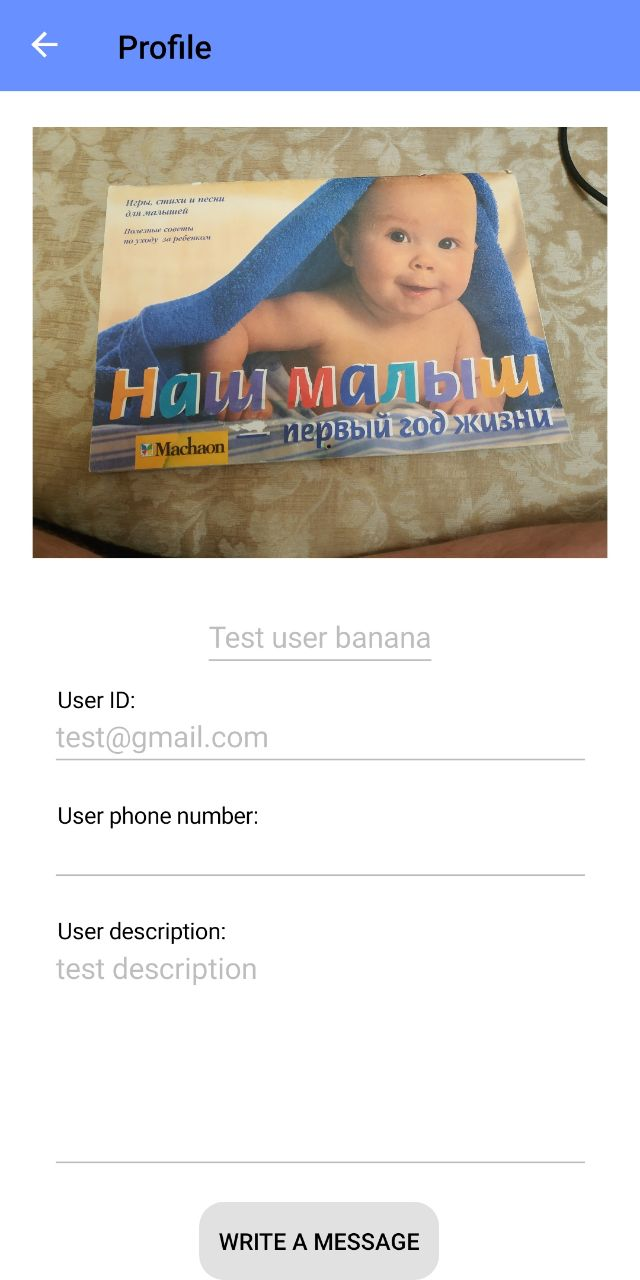


Рисунок 20. Знайдена сторінка користувача

У разі позитивного результату пошуку користувача буде відображено навігейшн пейдж з інформацією про знайденого користувача. Внизу буде кнопка для створення діалогу зі знайденим користувачем.

## **4 ІМПЛЕМЕНТАЦІЯ ВЗАЄМОДІЇ З МЕРЕЖЕЮ**

Для взаємодії з мережею використовується клас HttpProvider, за допомогою якого створюються запити до API сервера. У програмі перевіряється з’єднання з інтернет – мережею за допомогою вбудованого класу CrossConnectivity, який дозволяє рендерити в режимі реального часу.

Клас взаємодії з мережею описано в додатку А. Далі в додатку Б описано клас для взаємодії з API користувачів. За допомогою цього класу ми можемо залогінитись, отримати інформацію про користувачів, отримати список чатів та інше.

У додатку В показано клас взаємодії за допомогою функціоналу SignalR. Це дозволяє отримувати повідомлення у режимі реального часу та відправляти їх на сервер.

## **5 ІМПЛЕМЕНТАЦІЯ ЗБЕРІГАННЯ ДАНИХ ЛОКАЛЬНО**

Для зберігання даних локально була використана база даних SQLite. Було створено дві таблиці: Settings, ColourScheme.

public class Settings

{

public int Id { get; set; }

public int ColourSchemeId { get; set; }

public ColourScheme ColourScheme { get; set; }

}

Поле id – унікальний ідентифікатор налаштування.

Поле ColourSchemeId – унікальний ідентифікатор кольорової схеми.

Поле ColourScheme – саме об’єкт кольорової схеми.

public class ColourScheme

{

public int Id { get; set; }

public string SchemeType { get; set; }

public string HeaderColour { get; set; }

public string PageColour { get; set; }

public string TextColour { get; set; }

public string ButtonColour { get; set; }

}

Поле id – унікальний ідентифікатор кольорової схеми.

Поле SchemeType – тип схеми.

Поле HeaderColour – колір «шапки» сторінки.

Поле PageColour – колір бекграунду сторінки.

Поле TextColour – колір тексту.

Поле ButtonColour – колір кнопок.

Контекст бази даних описаний у додатку Г. При створенні БД таблиці відразу заповнюються початковими даними:

ColourScheme colourSchemeLight = new ColourScheme()

{

Id = 1,

SchemeType = "Light",

HeaderColour = "#6891ff",

PageColour = "#ffffff",

TextColour = "#000000",

ButtonColour = "#e1e1e1"

};

ColourScheme colourSchemeDark = new ColourScheme()

{

Id = 2,

SchemeType = "Dark",

HeaderColour = "#191d58",

PageColour = "#54555c",

TextColour = "#dbdbdb",

ButtonColour = "#888bae"

};

if (!ColourSchemes.Any())

{

ColourSchemes.AddRange(colourSchemeLight, colourSchemeDark);

}

Models.Settings settings = new Models.Settings()

{

Id = 1,

ColourScheme = colourSchemeLight

};

## **6 ІМПЛЕМЕНТАЦІЯ ВЗАЄМОДІЇ З HARDWARE ПРИСТРОЮ**

Для взаємодії з hardware використовуються вбудовані засоби взаємодії. Наприклад, для взаємодії з камерою для сканування QR-коду:

private async void btnScanQr\_Tapped(object sender, EventArgs e)

{

try

{

ZXingScannerPage scannerPage = new ZXingScannerPage();

NavigationPage.SetHasNavigationBar(scannerPage, false);

await Navigation.PushAsync(scannerPage);

scannerPage.OnScanResult += (result) =>

{

Device.BeginInvokeOnMainThread(async () =>

{

await Navigation.PopAsync();

if(!Int32.TryParse(result.Text, out int userId))

{

throw new Exception("Qr code is not user id!");

}

UserInfo userInfo = App.UserService.Info(userId, App.UserInfo.AccessToken);

if (userInfo != null)

{

ProfilePage profilePage = new ProfilePage(userInfo);

await Navigation.PushAsync(profilePage);

}

else

{

throw new Exception("userInfo is null!");

}

});

};

}

catch (Exception ex)

{

await DisplayAlert("Error!", "Unexpected error.", "OK");

}

}

Також, для взаємодії з системними повідомленнями використовується вбудований функціонал:

class NotificationHelper : INotification

{

private Context mContext;

private NotificationCompat.Builder mBuilder;

public static String NOTIFICATION\_CHANNEL\_ID = "10023";

public NotificationHelper()

{

mContext = global::Android.App.Application.Context;

}

public void CreateNotification(String title, String message)

{

try

{

var intent = new Intent(mContext, typeof(MainActivity));

intent.AddFlags(ActivityFlags.ClearTop);

intent.PutExtra(title, message);

var pendingIntent = PendingIntent.GetActivity(mContext, 0, intent, PendingIntentFlags.OneShot);

var alarmAttributes = new AudioAttributes.Builder()

.SetContentType(AudioContentType.Sonification)

.SetUsage(AudioUsageKind.Notification).Build();

mBuilder = new NotificationCompat.Builder(mContext);

mBuilder.SetSmallIcon(Resource.Drawable.settings);

mBuilder.SetContentTitle(title)

//.SetSound(sound)

.SetAutoCancel(true)

.SetContentTitle(title)

.SetContentText(message)

.SetChannelId(NOTIFICATION\_CHANNEL\_ID)

.SetPriority((int)NotificationPriority.High)

.SetVibrate(new long[0])

.SetDefaults((int)NotificationDefaults.Sound | (int)NotificationDefaults.Vibrate)

.SetVisibility((int)NotificationVisibility.Public)

.SetSmallIcon(Resource.Drawable.mobileDevAppLogo)

.SetContentIntent(pendingIntent);

NotificationManager notificationManager = mContext.GetSystemService(Context.NotificationService) as NotificationManager;

if (global::Android.OS.Build.VERSION.SdkInt >= global::Android.OS.BuildVersionCodes.O)

{

NotificationImportance importance = global::Android.App.NotificationImportance.High;

NotificationChannel notificationChannel = new NotificationChannel(NOTIFICATION\_CHANNEL\_ID, title, importance);

notificationChannel.EnableLights(true);

notificationChannel.EnableVibration(false);

notificationChannel.SetShowBadge(true);

notificationChannel.Importance = NotificationImportance.High;

notificationChannel.SetVibrationPattern(new long[] { 100, 200, 300, 400, 500, 400, 300, 200, 400 });

if (notificationManager != null)

{

mBuilder.SetChannelId(NOTIFICATION\_CHANNEL\_ID);

notificationManager.CreateNotificationChannel(notificationChannel);

}

}

notificationManager.Notify(0, mBuilder.Build());

}

catch (Exception ex)

{

}

}

}

### **ВИСНОВОК**

Першим кроком було створення макету графічного інтерфейсу користувача, використовуючи новітні техніки та рекомендації UI / UX для створення максимально комфортного для користування інтерфейсу. Це дозволило виділити основні сутності, які надалі були використані при проектуванні архітектури бази даних.

Наступним кроком була розробка WEB API серверної частини, яке відповідало за логіку програми. Усі маніпуляції з базою даних та між користувачами відбуваються за допомогою звертання до написаного API.

Паралельно з розробкою серверної частини проекту велася розробка графічного інтерфейсу користувача. Для аутентифікації та авторизації був обраний стандартний підхід з сторінкою для введення логіну та паролю, якщо такі існують, інакше ж необхідно перейти на сторінку реєстрації нового користувача. Пароль відразу ж шифрується за допомогою обраного алгоритму шифрування та передається на серверну частину у зашифрованому вигляді, що додає програмному продукту більшої безпеки для особистих даних користувача.

Чат був створений у режимі реального часу за допомогою механізму SignalR. Він дає змогу відображати зміни у базі даних без перезавантаження сторінки.

В результаті виконання курсової роботи ми отримали програмний продукт, який повністю відповідає заданим вимогам та очікуваним результатам.

### **ПЕРЕЛІК ПОСИЛАНЬ**

1. Мова програмування C# та платформи .Net та .NET Core. Електронний ресурс режим доступу: // https://codernet.ru/books/c\_sharp
2. Енциклопедія C# та платформи .Net. Електронний ресурс режим доступу: // https://metanit.com/sharp/tutorial/
3. MS SQL Server 2017. Електронний ресурс режим доступу: // https://metanit.com/sql/sqlserver/
4. Керівництво ASP.NET Core 3. Електронний ресурс режим доступу: // https://metanit.com/sharp/aspnet5/
5. Програмування на Xamarin Forms. Електронний ресурс режим доступу: // https://metanit.com/sharp/xamarin/
6. HTML5 та CSS3. Електронний ресурс режим доступу: // https://metanit.com/web/html5/

# **ДОДАТКИ**

## **ДОДАТОК А**

public class HttpProvider : IHttpProvider

{

private readonly HttpClient \_client;

public HttpProvider(HttpClient client)

{

\_client = client ?? throw new ArgumentNullException(nameof(client));

}

public TResult SendRequest<TResult>(HttpRequestMessage requestMessage)

{

TimeSpan currentWaitingTime;

for (int i = 0; i < Configuration.HttpRetryCount; i++)

{

try

{

HttpResponseMessage response = \_client.SendAsync(requestMessage).Result;

if (!response.IsSuccessStatusCode)

{

Thread.Sleep(Configuration.HttpWaitMs);

continue;

}

string responseStr = response.Content.ReadAsStringAsync().Result;

return JsonConvert.DeserializeObject<TResult>(responseStr);

}

catch (Exception ex)

{

var sw = Stopwatch.StartNew();

currentWaitingTime = TimeSpan.FromMilliseconds(Configuration.HttpWaitMs) - TimeSpan.FromMilliseconds(sw.ElapsedMilliseconds);

if(currentWaitingTime.TotalMilliseconds > 0)

Thread.Sleep(currentWaitingTime);

continue;

}

}

return default;

}

## **ДОДАТОК Б**

public class UserService : IUserService

{

private readonly IHttpProvider \_httpProvider;

public UserService(IHttpProvider httpProvider)

{

\_httpProvider = httpProvider ?? throw new ArgumentNullException(nameof(httpProvider));

}

public UserInfo Info(int userId, string userAuthToken)

{

var requestMessage = new HttpRequestMessage(

HttpMethod.Get, $"{Configuration.UserInfoIdGettingRoute}/{userId}");

requestMessage.Headers.Add(Configuration.AuthHeaderKey, userAuthToken);

return \_httpProvider.SendRequest<UserInfo>(requestMessage);

}

public UserInfo Info(string userSearchInfo, string userAuthToken)

{

var requestMessage = new HttpRequestMessage(

HttpMethod.Get, $"{Configuration.UserInfoNameGettingRoute}/{userSearchInfo}");

requestMessage.Headers.Add(Configuration.AuthHeaderKey, userAuthToken);

return \_httpProvider.SendRequest<UserInfo>(requestMessage);

}

public UserInfo Register(UserRegister newUser)

{

var requestMessage = new HttpRequestMessage(HttpMethod.Post, Configuration.UserInfoRegisterRoute);

requestMessage.AddStringContent(newUser);

return \_httpProvider.SendRequest<UserInfo>(requestMessage);

}

public UserInfo Login(UserLogin userLoginInfo)

{

var requestMessage = new HttpRequestMessage(HttpMethod.Post, Configuration.UserInfoLoginRoute);

requestMessage.AddStringContent(userLoginInfo);

return \_httpProvider.SendRequest<UserInfo>(requestMessage);

}

public UserInfo Edit(UserEdit userEditInfo, string userAuthToken)

{

var requestMessage = new HttpRequestMessage(HttpMethod.Post, Configuration.UserInfoEditRoute);

requestMessage.Headers.Add(Configuration.AuthHeaderKey, userAuthToken);

requestMessage.AddStringContent(userEditInfo);

return \_httpProvider.SendRequest<UserInfo>(requestMessage);

}

}

## **ДОДАТОК В**

public class ChatMessagingClient : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

private readonly HubConnection \_hubConnection;

private bool \_isBusy;

public bool IsBusy

{

get => \_isBusy;

set

{

if (\_isBusy != value)

{

\_isBusy = value;

OnPropertyChanged("IsBusy");

}

}

}

private bool \_isConnected;

public bool IsConnected

{

get => \_isConnected;

set

{

if (\_isConnected != value)

{

\_isConnected = value;

OnPropertyChanged("IsConnected");

}

}

}

public MessageInput Message { get; set; }

public ObservableCollection<MessageInfo> Messages { get; private set; }

public ObservableCollection<ErrorMessage> Errors { get; private set; }

public Command SendMessageCommand { get; private set; }

public ChatMessagingClient(string userAuthToken)

{

// создание подключения

\_hubConnection = new HubConnectionBuilder()

.WithUrl(Configuration.ChatLiveMessagingUrl, httpOptions =>

{

httpOptions.Headers.Add(Configuration.AuthHeaderKey, userAuthToken);

})

.Build();

\_hubConnection.Closed += async (error) =>

{

IsConnected = false;

await Task.Delay(5000);

Connect();

};

\_hubConnection.On<MessageInfo, ErrorMessage>("Receive", (message, error) =>

{

ReceiveMessage(message, error);

});

Messages = new ObservableCollection<MessageInfo>();

Errors = new ObservableCollection<ErrorMessage>();

IsConnected = false;

IsBusy = false;

SendMessageCommand = new Command(() => SendMessage(), () => IsConnected);

}

// подключение к чату

public async void Connect()

{

if (IsConnected)

return;

try

{

await \_hubConnection.StartAsync();

IsConnected = true;

}

catch (Exception ex)

{

Errors.Add(new ErrorMessage { Message = ex.Message });

}

}

// Отключение от чата

public async void Disconnect()

{

if (!IsConnected)

return;

try

{

await \_hubConnection.StopAsync();

IsConnected = false;

}

catch (Exception ex)

{

Errors.Add(new ErrorMessage { Message = ex.Message });

}

}

// Отправка сообщения

public async void SendMessage()

{

try

{

IsBusy = true;

await \_hubConnection.InvokeAsync("Send", Message);

}

catch (Exception ex)

{

Errors.Add(new ErrorMessage { Message = ex.Message });

}

finally

{

IsBusy = false;

}

}

// Добавление сообщения

private void ReceiveMessage(MessageInfo message, ErrorMessage error)

{

if(message != null)

Messages.Add(message);

if (error != null)

Errors.Add(error);

}

public void OnPropertyChanged(string prop = "")

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

}

## **ДОДАТОК Г**

public class AppDbContext : DbContext

{

public string dbName { get; private set; }

private string dbPath { get; set; }

public AppDbContext(string dbPath = null, string dbName = "testDb24627416725322332.db")

{

this.dbPath = dbPath ??

Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Personal), dbName);

Database.EnsureCreated();

InitializeData();

}

private void InitializeData()

{

ColourScheme colourSchemeLight = new ColourScheme()

{

Id = 1,

SchemeType = "Light",

HeaderColour = "#6891ff",

PageColour = "#ffffff",

TextColour = "#000000",

ButtonColour = "#e1e1e1"

};

ColourScheme colourSchemeDark = new ColourScheme()

{

Id = 2,

SchemeType = "Dark",

HeaderColour = "#191d58",

PageColour = "#54555c",

TextColour = "#dbdbdb",

ButtonColour = "#888bae"

};

if (!ColourSchemes.Any())

{

ColourSchemes.AddRange(colourSchemeLight, colourSchemeDark);

}

Models.Settings settings = new Models.Settings()

{

Id = 1,

ColourScheme = colourSchemeLight

};

if (!Settings.Any())

{

Settings.AddRange(settings);

}

SaveChanges();

}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlite($"Filename={dbPath}");

}

public bool AddUserIfNotExist(UserInfo user)

{

if(!userInfo.Any())

{

userInfo.Add(user);

SaveChanges();

return true;

}

else

{

return false;

}

}

public bool DeleteUserIfExist()

{

if (userInfo.Any())

{

userInfo.Remove(userInfo.FirstOrDefault());

SaveChanges();

return true;

}

else

{

return false;

}

}

public DbSet<Models.Settings> Settings { get; set; }

public DbSet<ColourScheme> ColourSchemes { get; set; }

public DbSet<UserInfo> userInfo { get; set; }

}