МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ

(підпис)

01 червня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформатика»

на тему: «Мобільний додаток для підтримки осіб, які перебувають на прифронтових територіях»

здобувача групи ІН-02 Давиденка Олександра Анатолійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Олександр ДАВИДЕНКО

(підпис)

Керівник ст. викл., канд. фіз.-мат. наук, доц.

Оксана ШОВКОПЛЯС _____

(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 «Комп'ютерні науки», освітньо-професійної програми «Інформатика»
здобувача групи ІН-02 Давиденка Олександра Анатолійовича

1. Тема роботи: «Мобільний додаток для підтримки осіб, які перебувають на прифронтових територіях»
затверджую наказом по СумДУ від *«22» квітня 2024 р. № 0414-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 1 червня 2024 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми, конкурентів, цільової аудиторії, актуальність розробки, постановка й формування завдань дослідження. 2) Огляд та вибір програмних засобів для інформаційних систем. 3) Створення мобільного додатку для підтримки осіб, що перебувають на прифронтових територіях. 4) Аналіз отриманих результатів
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.
Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми, конкурентів, цільової аудиторії, актуальність розробки, постановка й формування завдань дослідження*		
2	*Огляд та вибір програмних засобів для інформаційних систем.*		
3	*Розроблення інформаційної системи для управління взаємовідносинами з клієнтами для підприємства електропостачання*		
4	*Аналіз отриманих результатів*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 47 стор., 18 рис., 1 табл., 2 додатки, 20 використаних джерел.

Обґрунтування актуальності теми роботи – тема кваліфікаційної роботи є актуальною, оскільки Україна переживає складний період через війну і це створює великі труднощі для всього населення. Розробка Android-додатку, котрий допоможе цивільному населенню отримати необхідну допомогу, є вкрай важлива.

Об'єкт дослідження – процес взаємодії з користувачами, що перебувають на прифронтових територіях.

Предмет дослідження – методи розробки андроїд-додатків для допомоги цивільному населенню.

Мета роботи – розробка програмного забезпечення для взаємодії та взаємодопомоги між користувачами, зокрема жителів прифронтових територій, за допомогою мов програмування та фреймворків.

Методи дослідження – мова програмування Kotlin, моделювання логіки за допомогою технології Firebase, розробка інтерфейсу для взаємодії з системою в Android Studio.

Результати – Розроблено андроїд-додаток, який надає можливість користувачам реєструватись або авторизуватись, залишати оголошення про потребу у допомозі. В додатку доступний довідник з порадами щодо дій у екстремальних ситуаціях, які можуть виникнути в умовах війни.

МОБІЛЬНИЙ ДОДАТОК, ОГОЛОШЕННЯ ПРО ДОПОМОГУ
ANDROID, FIREBASE, KOTLIN

ЗМІСТ

ВСТУП

Актуальність. Тема кваліфікаційної роботи є актуальною, оскільки Україна переживає складний період через війну і це створює великі труднощі для всього населення. Розробка Android-додатку, котрий допоможе цивільному населенню отримати необхідну допомогу, є вкрай важлива.

Об’єкт дослідження – процес взаємодії з користувачами, що перебувають на прифронтових територіях.

Предмет дослідження – методи розробки андроїд-додатків для допомоги цивільному населенню.

Гіпотеза. Завдяки створенню Android-додатку, який матиме зрозумілий інтерфейс, цивільне населення, зокрема на прифронтових територіях, буде простіше отримати допомогу.

Новизна. Описане у даній роботі програмне рішення дозволить досягти більшої ефективності у отриманні цивільному населенню необхідної допомоги, а також отримати корисні поради щодо дій під час обстрілів, повітряних тривог та евакуації.

Апробація матеріалів роботи. Основні результати роботи оприлюднені та обговорені на міжнародній науково-технічній конференції студентів та молодих вчених «Інформатика, математика, автоматика» (IMA – 2023).

Структура. Дана робота складається зі вступу, аналітичного огляду, постановки задачі, вибір засобів та мов програмування для реалізації поставленої задачі, опису та розробці програмного забезпечення, висновків, списку використаних джерел та додатків.

Зв'язок роботи з науковою темою. Кваліфікаційна робота виконана на кафедрі комп’ютерних наук та пов’язана з виконанням науково-дослідної роботи

№ 0118U006971 «Методи, математичні моделі та інформаційні технології аналізу і синтезу інфокомунікаційних систем» (2018-2023).

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Аналіз предметної області

Число користувачів смартфонів у всьому світі досягло 6,7 мільярдів. В Україні, станом на 2022 рік, ця частка становила понад 40 мільйонів. За даними, опублікованими компанією «DeviceAtlas», можна спостерігати що, найпопулярнішою операційною системою для телефонів став Android 76%. Слідом за ним, другу сходинку займає IOS з 20%. Кругову діаграму співвідношень різних ОС, можна побачити на рисунку нижче (рис. 1.1) [1].

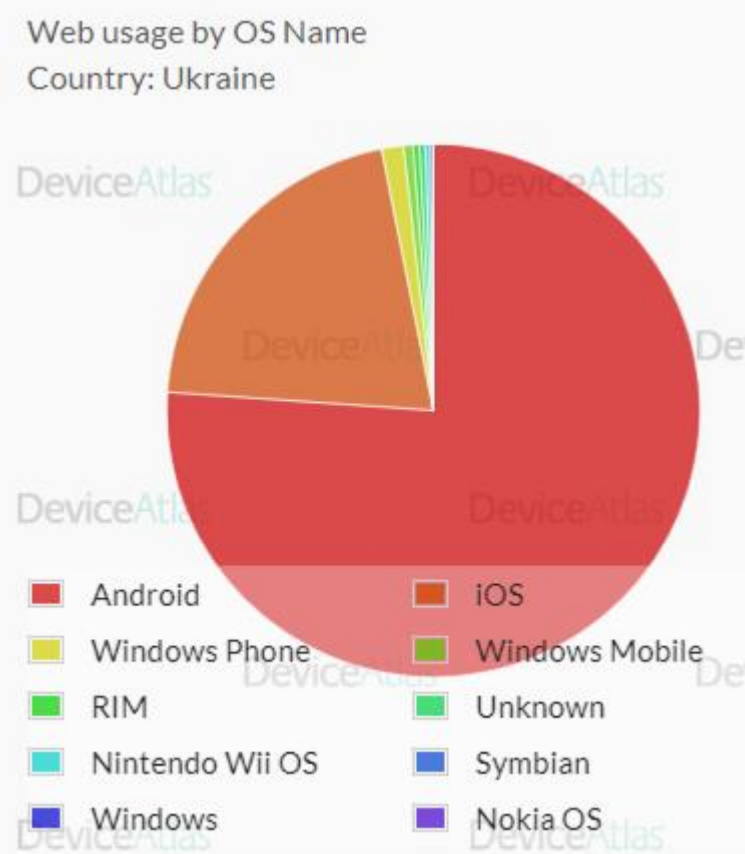


Рисунок 1.1 – Співвідношення користувачів різних ОС для смартфонів в Україні

ОС Android — це мобільна операційна система на основі ядра Linux, вперше розроблена компанією з Кремнієвої долини під назвою Android Inc. У 2007 році початок співпраці з Google. Розроблюється та підтримується альянсом Open Handset Alliance (OHA)[2].

Мобільний додаток — це програмний забезпечення, розроблене спеціально для використання на невеликих, бездротових обчислювальних пристроях, таких як смартфони та планшети та інших мобільних пристроях. Мобільні програми створюються з використанням різноманітних мов програмування та фреймворків, і їх можна завантажити та встановити з магазинів програм, таких як Apple App Store або Google Play[3].

1.2 Огляд методологій і технологій розробки Android-додатків

Створення мобільного додатку для операційної системи Android, котрий допоможе у взаємодії між населенням прифронтових територій та волонтерами, є основною метою. Ця система забезпечить зручний та швидкий спосіб обміну даними за для ефективної співпраці між користувачами.

Інтегроване середовище розробки (IDE) - це програмне забезпечення, яке надає комплексні інструменти та функції для полегшення розробки програмного забезпечення. Воно слугує центральною платформою для програмістів для ефективного написання, тестування та налагодження коду. IDE призначені для спрощення процесу розробки шляхом об'єднання різних інструментів і функцій в одному інтерфейсі, що полегшує розробникам управління проектами та ефективну співпрацю [4].

Для розробки додатків для Android основним середовищем розробки є Android Studio, яке є офіційним середовищем розробки від Google. Android Studio створена на основі IntelliJ IDEA та адаптована спеціально для розробки під Android. Android Studio постійно оновлюється і вдосконалюється, що робить її кращим IDE для більшості розробників Android завдяки потужному набору функцій, офіційній підтримці та частим оновленням, які відповідають останнім практикам і технологіям розробки Android [5].

Kotlin — це сучасна мова програмування, яка пропонує поєднання функціональної та об'єктно-орієнтованої парадигм програмування. Вона є статично типізованою, що означає, що типи змінних відомі під час компіляції. Це значно підвищує продуктивність та надійність мови. Основною метою розробки Kotlin було створення стислої, виразної та безпечної мови. Це досягається завдяки різноманітним особливостям мови, які зменшують кількість шаблонного коду, запобігають поширеним помилкам програмування та забезпечують безпеку [6].

1.3 Аналіз конкурентів-аналогів

З початком повномасштабного вторгнення в Україну, з’явилося багато нових додатків для допомоги населенню. А вже існуючі, додали новий корисний функціонал. Зявились додатки котрі допомагають с першою до медичною допомогою чи звернутись за консультацією лікарів. Серед таких можна назвати: «TacticMedAid», «Helsi», «Перша мобільна допомога – МФЧХ та ПП» та інші. Додатки які допомагають фіксувати пересування ворожих військ: «Bachu.info», «ТиХто», чат-бот у Telegram «STOP Russian War».

Додаток «Дія», який був створений для зручного користування електронними документами, додав новий фунціонал що також пов'язаний з повномасштабним вторгненням. Серед таких сервісів, можна назвати «Допомога армії», де можна зробити внесок до різних благодійних фондів (рис. 1.2).

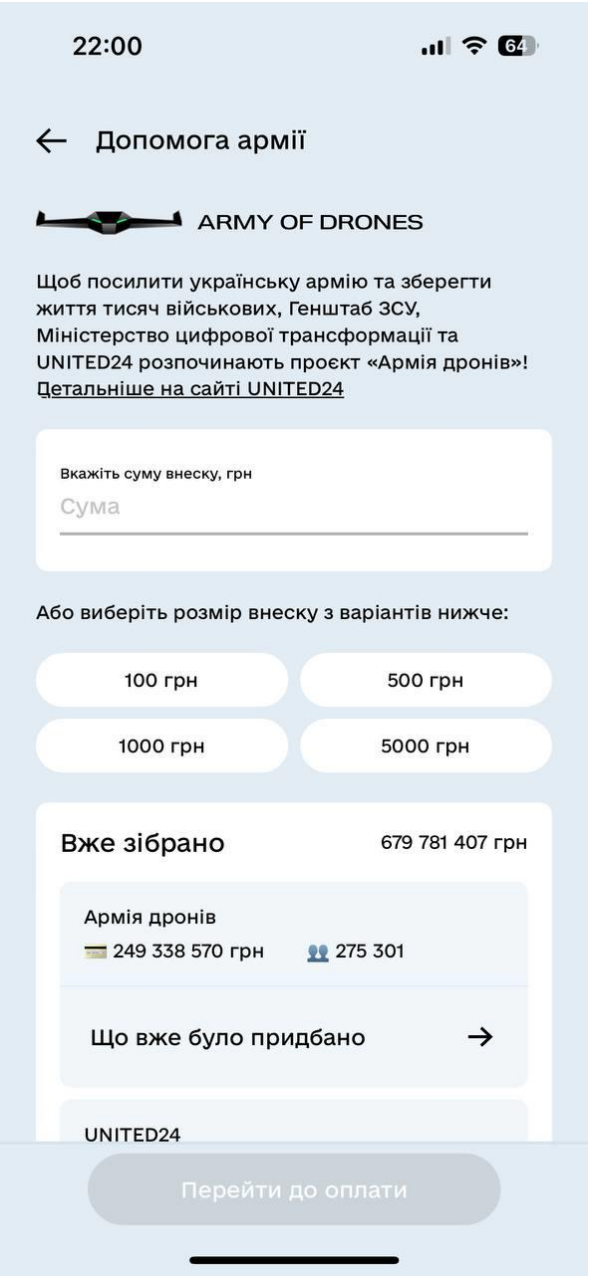
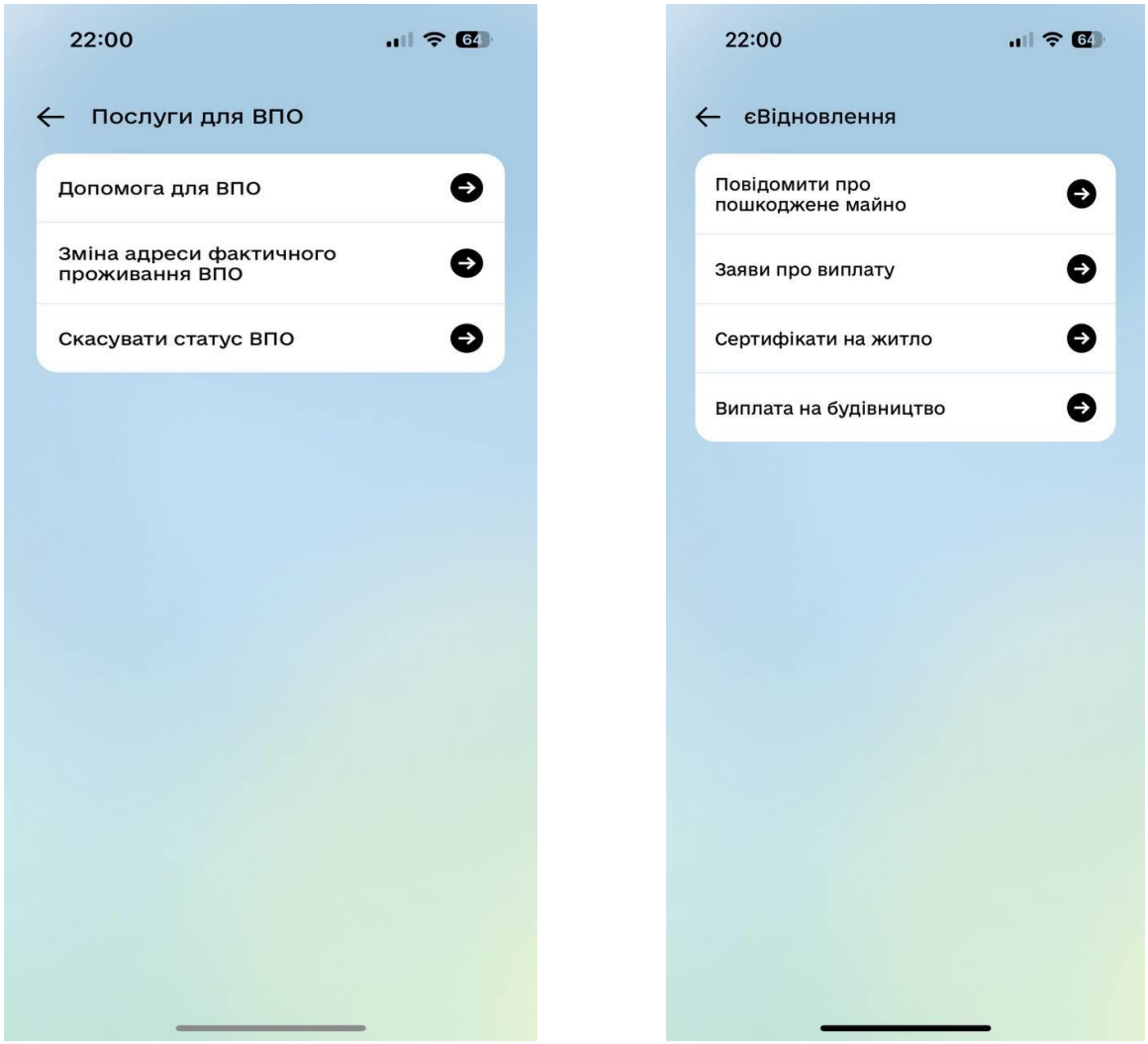


Рисунок 1.2 – Сервіс для грошової допомоги армії в застосунку «Дія»

Також сервіси для допомоги внутрішньо переміщеним особам(ВПО), та «Відновлення», якщо ваш будинок, майно зазнало пошкоджень від бойових дій, та дій агресора (рис. 1.3).



а б

Рисунок 1.1 – Сервіс для допомоги ВПО в застосунку «Дія» (а), сервіс для отримання допомоги в разі пошкодження майна (б)

Два схожих за своєю метою додатка «Trybefy» та «Допомога». Їх основна суть полягає в допомозі українськими українцям знайти співвітчизників за кордоном, знайти житло, влаштуватись на роботу. Перший додаток це не база оголошень чи вакансій, а, скоріш, соціальна мережа. Тож, що за допомогою Tribefy можна знайти т людей поруч із спільними інтересами, соціалізуватися на новому місці та налагодити контакти [7]. Нажаль, на даний момент доступ до додатку призупинений на невідомий термін. «Допомога» функціонує і на даний момент. Інсталювавши та

зареєструвавшись у додатку нас зустрічає головне меню (рис. 1.4), де ми можемо побачити оголошення щодо надання чи отримання допомоги.

Рисунок 1.4 – Головна сторінка застосунку «Допомога»

При створенні оголошення можна обрати пошук допомогу чи надання. Обравши та перейшовши до створення, нас зустрічає багато різних функцій, де можна вказати додаткову інформацію (рис. 1.5), наприклад:

– Тип оголошення (житло, робота, інше);

– Інформація щодо кількості людей, тварин;
– Назва та опис оголошення;
– Контактні дані.

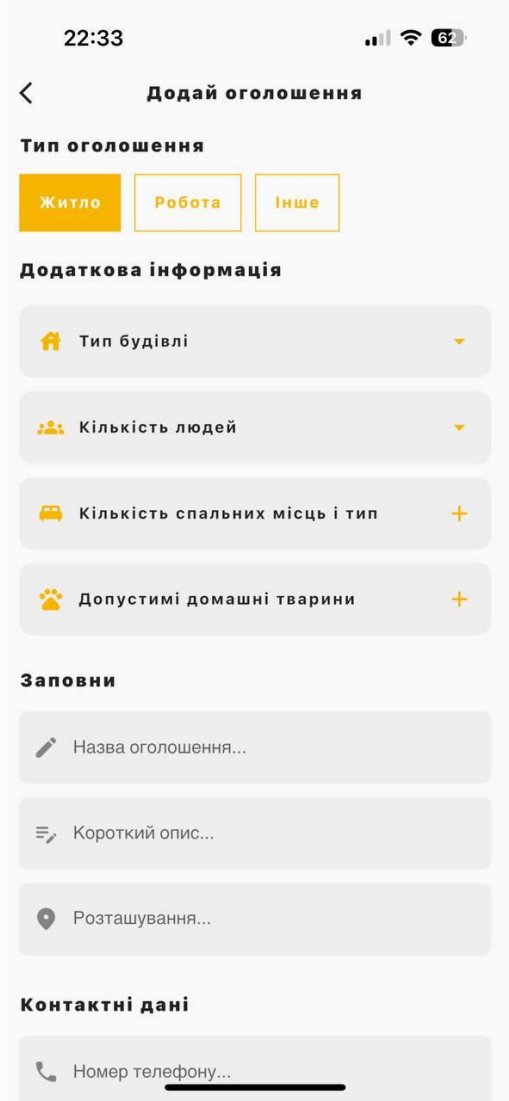


Рисунок 1.5 – Створення оголошення в застосунку «Допомога»

Додаток гарно структурований та має приємний дизайн. На мою думку існує проблема з модерацією, тому потрібно передбачити це при виконанні роботи.

Веб-додаток «HELPLIST.io», більш популярний ніж попередній додаток. Його мета також полягає у встановленні зв’язку між людиною якій потрібна допомога та волонтером. Сайт має простіший дизайн та функціонал, відсутність створення особистого кабінет, на відмінну від попереднього застосунку (рис. 1.6).

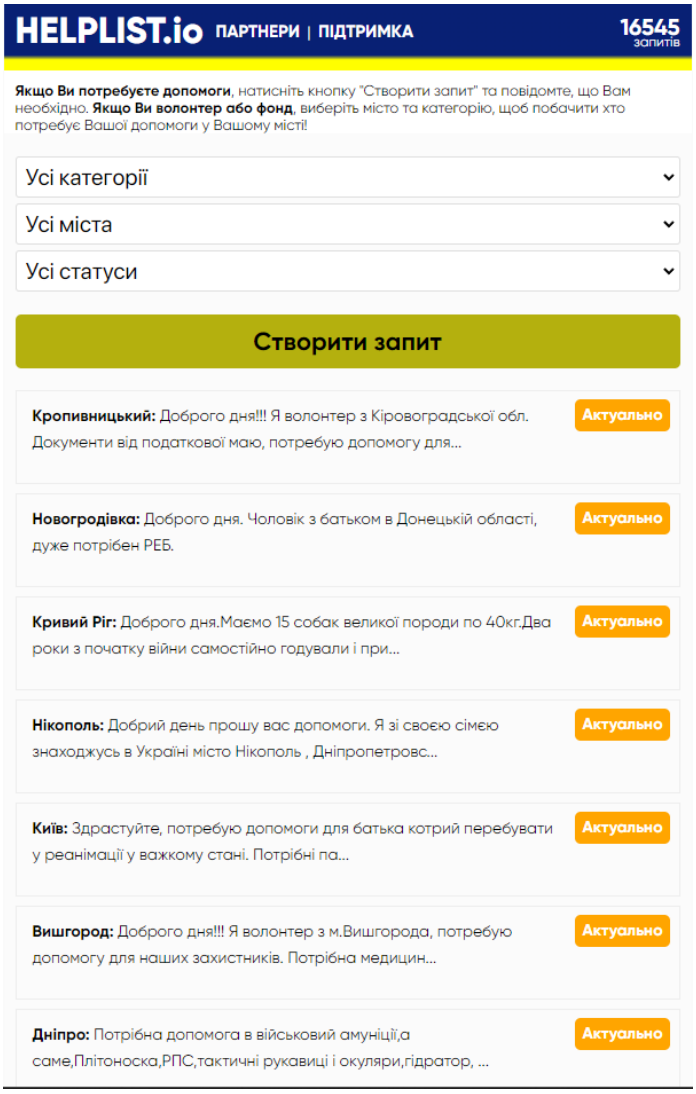


Рисунок 1.6 – Головна сторінка веб-додатку «HELPLIST.io»

При натисканні кнопки «Створити запит», з’являється вікно створення оголошення (рис. 1.7). Категорії і поля структуровані, інтерфейс зрозумілий. Важливим недоліком є, відсутність можливості прикріплення фото для додаткового підтвердження у правдивості оголошення.

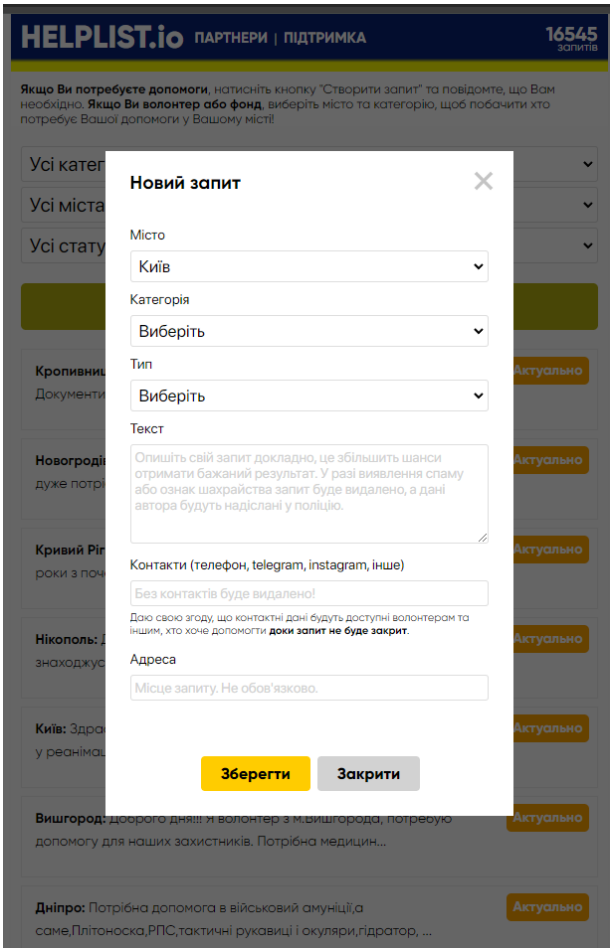


Рисунок 1.7 – Створення оголошення у веб-додатку «HELPLIST.io»

Аналізуючи додатки, які створені конкурентами, ми змогли скласти детальну порівняльну таблицю (таблиця 1.1), в якій представлено основні переваги та недоліки кожного з розглянутих аналогів. У порівняно функціонал, зручність використання, інтерфейс, технічні характеристики та інші важливі аспекти кожного додатку. Результати цього аналізу дозволили визначити сильні та слабкі сторони кожного продукту, що є надзвичайно корисним для розуміння конкурентного середовища та виявлення можливостей для покращення власного продукту. Використовуючи цю інформацію, ми можемо більш ефективно планувати розвиток нашого додатку, враховуючи успішні практики конкурентів і уникаючи їх помилок.

Таблиця 1.1 Аналіз конкурентів

Додаток	Переваги	Недоліки
Trybefy	Соціальна мережа для українців за кордоном. Допомагає знайти людей з спільними інтересами. Можливість соціалізації на новому місці.	Призупинений доступ. Відсутність бази даних оголошень або вакансій.
Допомога	Легкий у використанні інтерфейс. Можливість створення оголошень щодо надання чи отримання допомоги. Додаткові функції для уточнення інформації.	Проблеми з модерацією оголошень. Обмеженість функціоналу порівняно з соціальними мережами.
HELPLIST.io	Простий дизайн та зрозумілий інтерфейс. Швидке створення запитів на допомогу. Відсутність потреби у створенні особистого кабінету.	Немає можливості прикріплювати фото. Обмежений функціонал порівняно з мобільними додатками.
Дія	Безкоштовне завантаження та використання. Велика кількість сервісів, включаючи цифрові документи, підтримку армії, допомогу ВПО. Підтримка цифрового підпису.	Проблеми з реєстрацією та входом в систему. Технічні збої та помилки. Обмежена підтримка на інших мовах, крім української.

Порівняння додатків показує, що кожен з них має свої сильні сторони та обмеження. Trybefy є чудовим інструментом для соціалізації українців за кордоном, але його недоступність наразі зменшує його практичність. «Допомога» добре підходить для створення оголошень про допомогу, але проблеми з модерацією можуть вплинути на достовірність інформації. HELPLIST.io є простим та зрозумілим веб-додатком для швидкого отримання допомоги, але бракує функціоналу, такого як прикріплення фото. Дія виділяється багатофункціональністю і надає великий спектр послуг, проте має технічні проблеми та обмеження мовної підтримки. Ці додатки можуть бути корисними в різних ситуаціях, але всі вони потребують подальшого покращення, особливо в аспекті надійності та доступності.

1.4 Постановка задачі

Метою роботи є розроблення мобільного додатка для підтримки осіб, що перебувають на прифронтових територіях.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

- провести огляд та вибрати інструментів розробки;
- проєктування структури андроїд-додатку;
- розробити алгоритм роботи веб-додатка;
- розробити інтерфейс веб-додатка, що буде зручним для користувача;
- розробка логіки серверної частини веб-системи.

Даний мобільний додаток буде сприяти покращенню якості життя осіб, що перебувають на прифронтових територіях, шляхом забезпечення швидкого доступу до важливих ресурсів та інформації.

2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ

2.1 Основні концепції розробки мобільного додатку

Проаналізуємо особливості та засоби розробки мобільного додатку.

Для створення Android додатків є декілька популярних мов програмування та фреймворків:

Java – традиційна мова програмування для розробки Android-додатків. Вона має велику спільноту та підтримується офіційною документацією Android [8].

Kotlin - це новітня мова програмування, яка також підтримується офіційно для розробки Android додатків. Вона має багато сучасних функцій та простіше у використанні, ніж Java. Вона має короткий та зрозумілий синтаксис, простий у використанні, а також надійну підтримку з боку Google [9].

C++ з використанням Android NDK. Якщо вам потрібна висока продуктивність або ви вже маєте існуючий код на C++, ви можете використовувати Android NDK (Native Development Kit) для розробки додатків [10].

React Native - Це фреймворк, розроблений Facebook, який дозволяє створювати мобільні додатки за допомогою JavaScript та React. Використовуючи React Native, розробники мають можливість створювати як Android, так і iOS додатки [11].

Flutter - це фреймворк від Google, який дозволяє створювати мобільні додатки для Android та iOS з єдиним кодом бази. Він використовує мову програмування Dart [12].

Існує кілька інтегрованих середовищ розробки (IDE), які можна використовувати для написання Android додатків. Ось деякі з них:

1. Android Studio: Це офіційне середовище розробки для Android, розроблене на базі IntelliJ IDEA. Android Studio має вбудовані засоби для

редагування коду, відлагодження, дизайну імітації різних пристроїв, а також засоби для розгортання додатків на пристроях [13].

2. IntelliJ IDEA: Це потужне середовище розробки, яке також підтримує розробку Android додатків за допомогою відповідного плагіну. IntelliJ IDEA відкрите для розширень і надає багато корисних функцій для розробки програмного забезпечення [14].

3. Eclipse: Хоча це не так популярний вибір для розробки Android додатків, Eclipse все ще може бути використаний з Android Development Tools (ADT) плагіном для розробки Android додатків. Проте, рекомендується використовувати Android Studio або IntelliJ IDEA, оскільки вони отримують більше підтримки від Google [15].

4. Visual Studio Code: Це легкий та потужний текстовий редактор, який можна розширити за допомогою різноманітних розширень. Додавши плагіни для розробки Android, такі як Kotlin Language, ви можете створювати Android додатки у Visual Studio Code [16].

Кожне з цих середовищ має свої переваги та недоліки, і вибір залежить від особистого вподобання та потреб проекту. У більшості випадків Android розробники використовують Android Studio або IntelliJ IDEA через їхню повну підтримку Android розробки та інтеграцію з іншими інструментами Google.

2.2 Аналіз обраної мови та середовища програмування

Для розробки мобільного додатку під платформу Android було вирішено використовувати середовище розробки Android Studio та мову програмування Kotlin.

Використання Kotlin для розробки Android додатка має кілька очевидних переваг, що роблять його привабливим вибором для висококваліфікованих програмістів. Перш за все, сучасний синтаксис Kotlin дозволяє розробникам писати чистий та зрозумілий код, що зменшує ймовірність помилок і полегшує його розуміння. Крім того, Kotlin відзначається високою безпекою, завдяки

вбудованим функціям, таким як нульова безпека, що допомагає уникнути типових проблем з нульовими посиланнями та іншими багами, пов'язаними з ними [17].

Сумісність Kotlin з Java дає можливість використовувати існуючий Java код у проекті Kotlin, забезпечуючи плавний перехід на нову мову та забезпечуючи сумісність з існуючим кодом [18].

Офіційна підтримка Kotlin від Google підтверджує його статус як доречний вибір для Android розробки, гарантуючи постійне оновлення та підтримку з боку компанії [17].

Незважаючи на ці переваги, варто враховувати, що використання нової мови може вимагати часу на оволодіння її особливостями та синтаксисом. Також можуть виникнути проблеми зі сумісністю зі сторонніми бібліотеками або потреба в розв'язанні потенційних проблем зі сумісністю між версіями Kotlin та Java [9].

У підсумку, Kotlin є потужним інструментом для розробки Android додатків, який може значно полегшити та прискорити процес розробки, але варто уважно зважити його переваги та недоліки перед прийняттям рішення про використання цієї мови програмування.

Android Studio є найпопулярнішим та офіційним інтегрованим середовищем розробки для платформи Android. Варто використовувати Android Studio з кількох причин. По-перше, це офіційне інтегроване середовище розробки (IDE) для платформи Android, розроблене самою компанією Google. Воно надає повну підтримку для всіх функцій та оновлень, пов'язаних з розробкою для Android. Друга перевага - широкий вибір інструментів та плагінів, які спрощують і прискорюють процес розробки. Android Studio має вбудовані інструменти для створення інтерфейсу користувача, відлагодження, тестування, оптимізації та багато інших функцій, що робить розробку додатків більш ефективною (рис. 2.1). Також, Android Studio інтегрується з багатьма сервісами Google, такими як Firebase, Google Maps та Google Cloud Platform, що полегшує розробку та

інтеграцію з екосистемою Google. Проте, є і деякі недоліки. Android Studio може вимагати значних ресурсів обладнання, особливо при роботі з великими проектами. Іноді програма може запускатися повільно або виявляти нестабільність, що може вплинути на продуктивність. Також, для новачків Android Studio може здатися складним у налаштуванні та використанні через велику кількість доступних функцій [19].

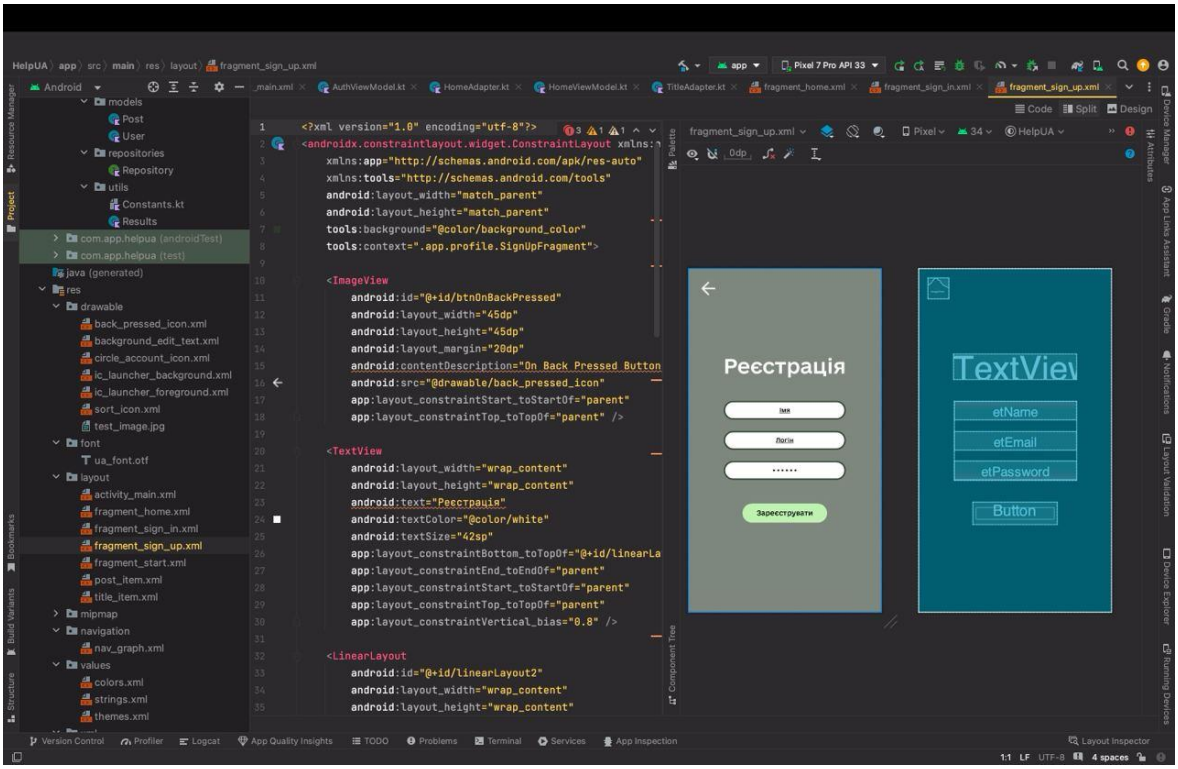


Рисунок 2.1 – Середовище розробки додатку

Також при створенні будуть використані такі технології:

1. Single Activity: Цей підхід використовується для створення додатків з однією активністю. Він спрощує навігацію між екранами, зменшує використання ресурсів та сприяє збереженню стану додатку. Підхід Single Activity дозволяє легше керувати фрагментами та покращує продуктивність розробки, зменшуючи витрати пам'яті і часу на переходи між екранами.

2. MVVM (Model-View-ViewModel): Цей шаблон проектування розділяє логіку додатку на три основні компоненти: модель (Model), представлення (View) і модель представлення (ViewModel). Модель відповідає за бізнес-логіку та

роботу з даними, представлення відображає інформацію користувачу, а модель представлення обробляє запити користувача та управляє відображенням даних. Використання MVVM спрощує розробку, полегшує тестування та зберігає стан екранів.

3. Jetpack Navigation (фрагменти): Ця бібліотека надає зручний спосіб для навігації між фрагментами в Android-додатках. Вона дозволяє визначати маршрути навігації у вигляді графа та використовувати їх для переходів між екранами. Jetpack Navigation спрощує створення та керування навігацією, зменшуючи кількість необхідного коду.
4. Koin (DI): Це легкий DI-контейнер для управління залежностями в Android-додатках. Koin використовує простий DSL-синтаксис для конфігурації залежностей та дозволяє здійснювати ін'єкцію залежностей без необхідності вручного налаштування DI-контейнера.
5. Coroutines та Flows: Ці механізми асинхронного програмування в Kotlin дозволяють ефективно працювати з фоновими операціями та реактивним програмуванням. Coroutines забезпечують легкий і ефективний спосіб виконання асинхронних операцій, в той час як Flows дозволяють працювати з потоками даних у вигляді послідовностей.
6. Clean Architecture (в пакетах): Цей архітектурний підхід дозволяє розділити додаток на логічні компоненти для полегшення тестування, розуміння коду та збереження його легкості. Clean Architecture допомагає зменшити залежності між різними компонентами системи та забезпечує їхню ізоляцію від змін.
7. ViewBinding: Цей механізм дозволяє отримати доступ до елементів інтерфейсу користувача безпосередньо в коді без необхідності використання методу findViewById. Використання ViewBinding спрощує розробку, полегшує читання та розуміння коду та зменшує ризик помилок.
8. Room (ORM для SQLite): Ця бібліотека надає зручний і простий спосіб роботи з базами даних SQLite на платформі Android. Room дозволяє використовувати об'єктно-реляційне відображення (ORM) для зберігання та отримання даних,

що дозволяє полегшити роботу з базами даних та зберігати код більш чистим та зрозумілим.

9. Firebase Tools (Database, Storage та інше): Firebase надає широкий спектр інструментів для розробки мобільних додатків, включаючи базу даних, сховище файлів, аутентифікацію користувачів та інші сервіси. Використання Firebase дозволяє прискорити розробку, забезпечити швидку Dependency Injection: Підхід, що дозволяє управляти залежностями між компонентами програми, спрощуючи тестування та підтримку коду.

Ці технології допомагають спростити розробку Android-додатків. Вони сприяють зменшенню складності коду та полегшенню його розуміння, ефективній навігації між екранами, управлінню залежностями між компонентами, забезпеченню асинхронної та неблокуючої роботи з фоновими операціями, зменшенню ризику помилок та підвищенню продуктивності розробки, а також інтеграції з хмарними сервісами для збереження та обробки даних.

2.3 Проєктування та структура додатку

Додаток спроєктовано з декількох основних компонентів, які взаємодіють між собою для забезпечення функціональності. Нижче наведено опис структури додатку та принципів його роботи, використовуючи зображену на рисунку схему.

1. Запуск програми: При запуску програми починається ініціалізація основних компонентів додатку.
2. Application: Цей компонент відповідає за загальну конфігурацію додатку, включаючи налаштування контексту та глобальних залежностей. Тут також відбувається ініціалізація необхідних сервісів та менеджерів.

3. MainActivity: Це основна активність додатку, яка виступає контейнером для фрагментів. Вона забезпечує навігацію та взаємодію між різними екранами додатку.
4. Fragments (Екрани): Фрагменти представляють різні екрани додатку. Кожен фрагмент відповідає за конкретну функціональність, наприклад, відображення профілю користувача або створення нового поста.
5. FireBaseRepository: Репозиторій відповідає за обробку даних та їх взаємодію з Firebase. Він виступає посередником між джерелами даних і компонентами додатку, що потребують ці дані.
6. AuthDataSource та DataBaseDataSource:
   – AuthDataSource: Цей компонент відповідає за обробку даних аутентифікації користувачів. Він взаємодіє з Firebase для виконання завдань, пов'язаних із входом, реєстрацією та іншими діями, пов'язаними з аутентифікацією.
   – DataBaseDataSource: Цей компонент відповідає за доступ до бази даних, що зберігається на сервері. Він забезпечує збереження, оновлення та отримання даних, таких як інформація про користувачів, їх профілі, пости та інші дані, пов'язані з функціональністю соціальної мережі.
7. Сервер: Сервер забезпечує зберігання даних, що надходять від додатку. Дані зберігаються у форматі JSON, що дозволяє легко їх обробляти та використовувати для різних потреб додатку. Сервер взаємодіє з Firebase, забезпечуючи надійне збереження та обробку даних.

На рисунку нижче представлена схема взаємодії між цими компонентами.

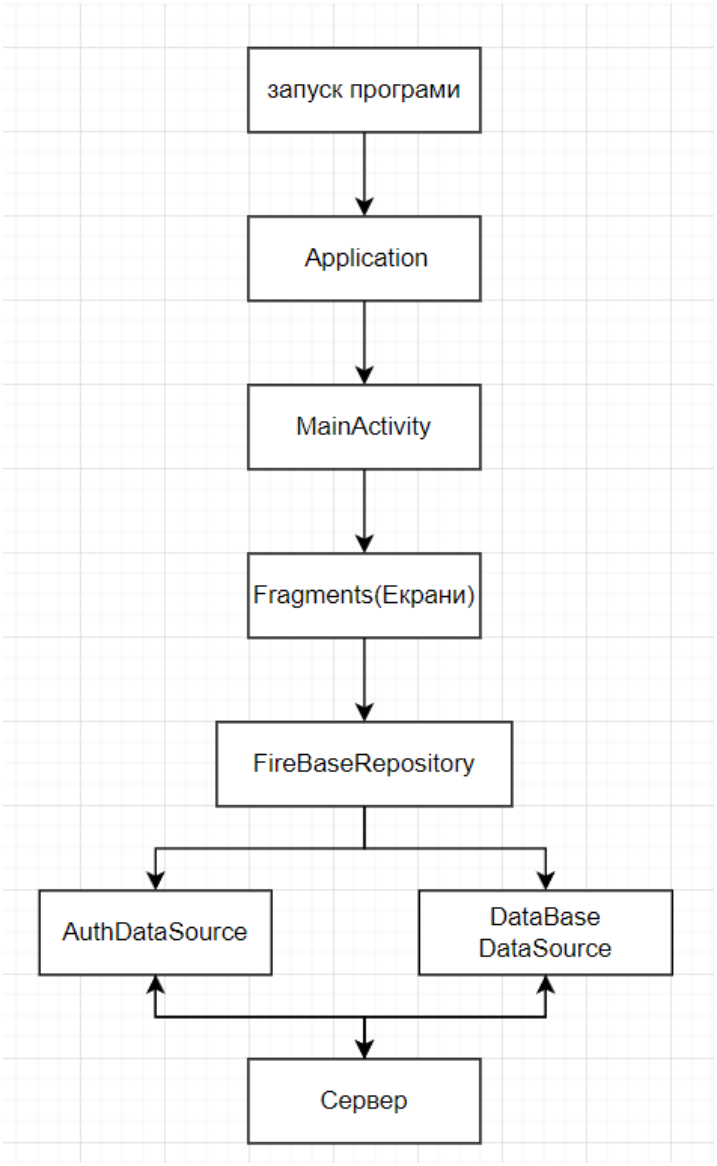


Рисунок 2.2 – Схема структури додатку

Таким чином, структура додатку організована так, щоб забезпечити ефективне зберігання, обробку та відображення даних, використовуючи віддалений сервер для збереження інформації. Це дозволяє оптимізувати роботу додатку і ефективно використовувати обмежені ресурси мобільного пристрою.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Налаштування бібліотек

Для досягнення надійності та стабільності роботи системи вимагається встановити необхідні програмні бібліотеки. Серед найважливіших бібліотек, які використані в реалізації, можна виділити:

- AndroidX (core, appcompat, material, constraintlayout, recyclerview, navigation, fragment)
- Firebase (analytics, auth, database, bom)
- Kotlin (serialization)
- Koin (для DI)
- CircleImageView, Picasso
- Тестування (JUnit, Espresso)

Ці бібліотеки відіграють критичну роль у розробці нашого додатку. AndroidX забезпечує базові компоненти та підтримку сучасного дизайну, Firebase - потужні інструменти для аналітики, аутентифікації та зберігання даних користувачів. Kotlin прискорює розробку завдяки своїм удосконаленим можливостям серіалізації. Koin полегшує впровадження залежностей. Інші бібліотеки додають додаткові можливості, такі як обробка зображень і керування фрагментами. Для забезпечення якості ми використовуємо JUnit і Espresso для автоматизованого тестування. Загалом, ці бібліотеки створюють потужне середовище для швидкої, надійної та функціональної розробки нашого додатку. (Додаток Б)

3.2 Розробка домашнього екрану

Використовуючи засоби та можливості інтегрованого середовища розробки Android Studio, було реалізовано інтерфейс Android-додатку.

На рисунку 3. 1 зображено головну сторінку додатка. На ній розміщено оголошення користувачів щодо допомоги. Як можна помітити, оголошення

різного кольору, що свідчить про актуальність та різні типи допомоги: медична, фінансова, психологічна.

Рисунок 3.1 – Головна сторінка з оголошеннями

Код файлу, відображає головний екран додатка, включаючи список постів та заголовок, та забезпечує їх оновлення при отриманні нових даних з ViewModel (**Error! Reference source not found.** – HomeFragment.kt).

3.3 Розробка екрану авторизації

Для авторизації та створення профілю, було розроблено окремий екран (**Error! Reference source not found.** – AuthDataSource.kt). На ньому розміщено дві основні кнопки: «Увійти» та «Реєстрація» (Рис 3.2). Якщо у вас є аккаунт то при вводі коректного логіну та пароля та натисканні на відповідну кнопку вас авторизує в систему. Якщо аккаунт відсутній, то можна його створити увівши необхідні дані. Сторінки реєстрації та авторизації зображено на рисунках 3.3 та 3.4 відповідно.

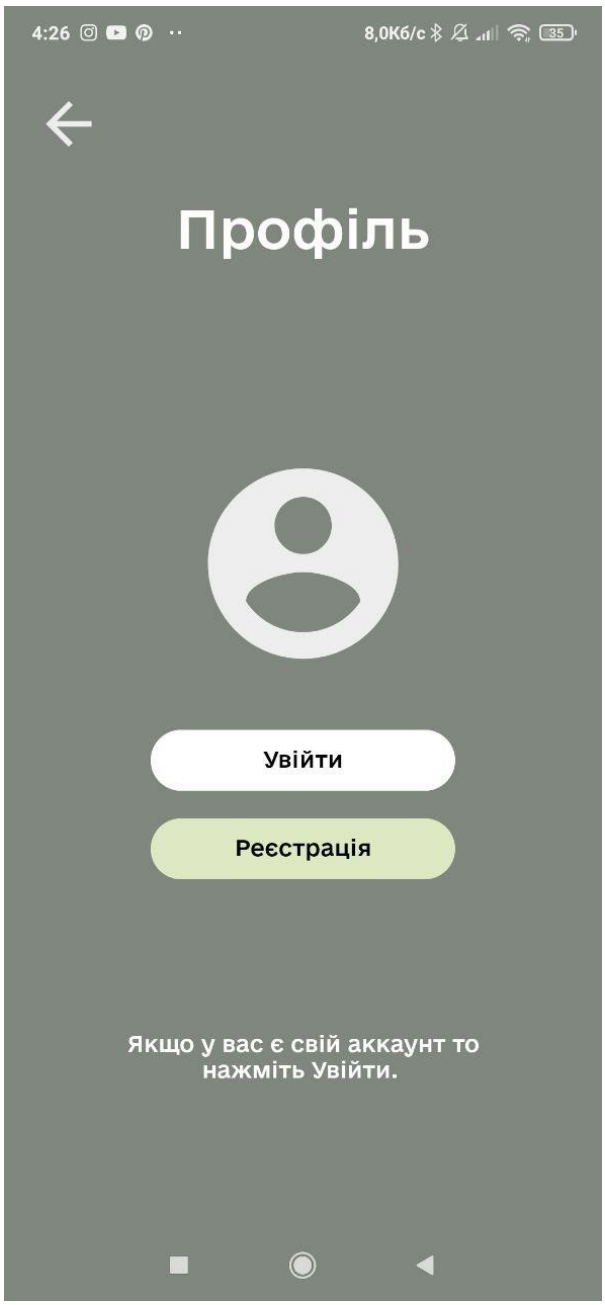
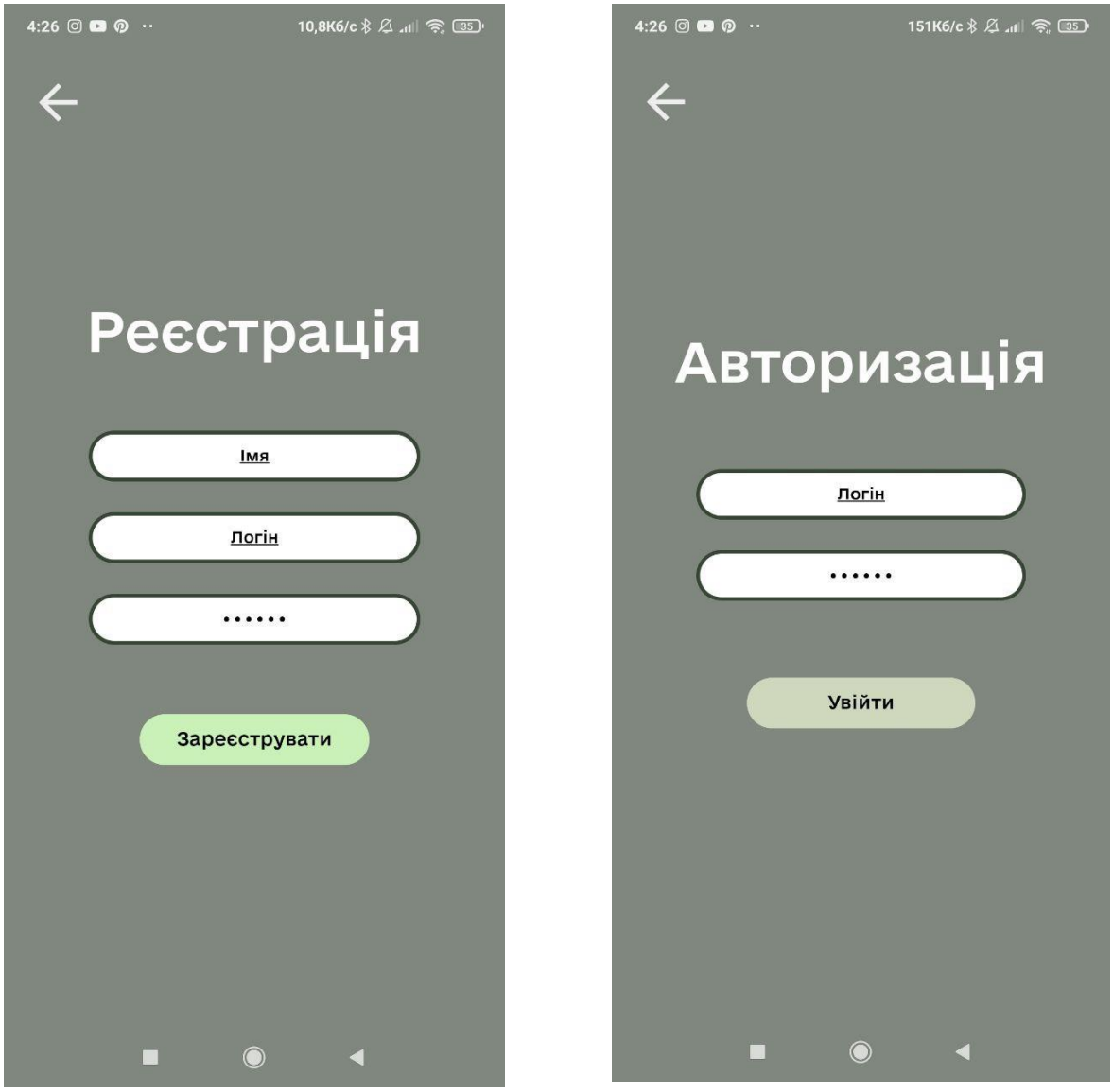


Рисунок 3.2 – Вхід до додатку



а б

Рисунок 3.3 – Реєстрація нового користувача(а), авторизація нового користувача(б)

3.4 Розробка екрану створення постів

Якщо користувач авторизований, то він має можливість створити пост (оголошення), щодо необхідної допомоги, натиснувши кнопку «Створити». Вікно створення оголошення(рис. 3.5), має декілька полів: заголовок, текст, зображення та тип необхідної допомоги. Всі ці речі є найбільш необхідними для створення посту, для ефективної взаємодії між користувачами. Файл

«CreatePostFragment» відповідає за відображення інтерфейсу створення нового поста та взаємодію користувача з ним (**Error! Reference source not found.** – CreatePostFragment.kt). Файл «CreatePostViewModel» відповідає за логіку створення поста та взаємодію з репозиторієм даних (**Error! Reference source not found.** – CreatePostViewModel.kt).

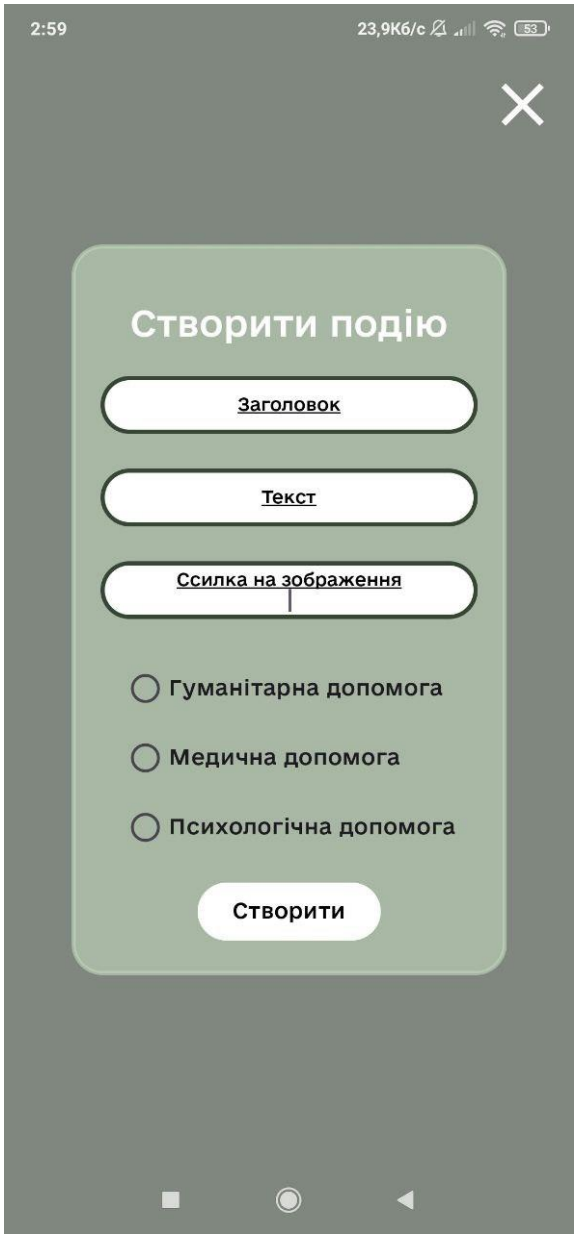


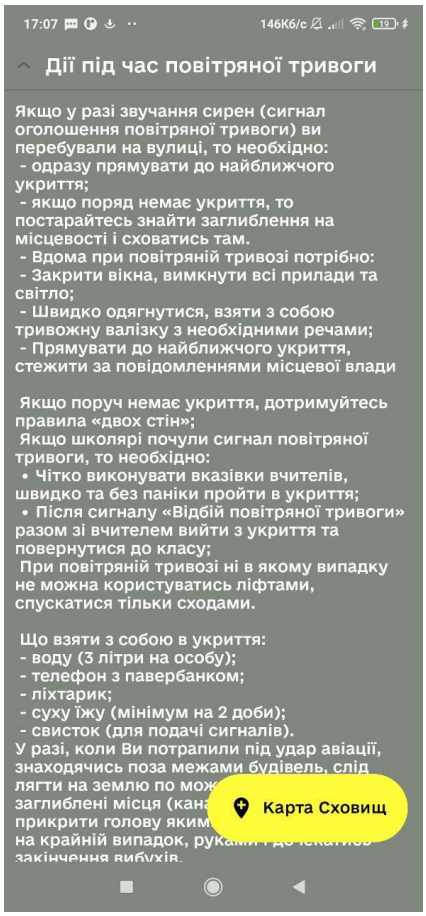
Рисунок 3.4 – Екран створення оголошення

Таким чином, «CreatePostFragment» відповідає за взаємодію з користувачем і відображення UI, тоді як «CreatePostViewModel» відповідає за

обробку даних та бізнес-логіку, забезпечуючи асинхронну взаємодію з репозиторієм даних. Разом вони дозволяють реалізувати функціонал створення нового поста в додатку, підтримуючи чисту архітектуру та розділення обов'язків.

3.5 Розробка довідника

У мобільному додатку для підтримки осіб на прифронтових територіях наявність довідника з інформацією про дії в критичних ситуаціях та мапи укриттів є надзвичайно важливою (Додаток А - SecurityPolicyFragment.kt). У кризових ситуаціях, таких як артилерійські обстріли чи авіаудари, швидкий доступ до правильної інформації може врятувати життя. Довідник надає чіткі інструкції, які допомагають користувачам швидко зорієнтуватися і прийняти правильні рішення під час стресу (рис. 3.5). Це знижує рівень паніки, дозволяючи людям зосередитися на виконанні необхідних дій.



а б

Рисунок 3.5 – Довідник дій та порад для критичних ситуацій(а), розгорнутий варіант(б)

Мапа укриттів також є критичною складовою додатку. Вона дозволяє людям швидко знайти найближче безпечне місце у разі небезпеки, що особливо важливо, коли час на пошук обмежений. Актуальна інформація про стан та доступність укриттів допомагає ефективно планувати свої дії. Натиснувши кнопку «Карта сховищ», відкриється карта актуальних укриттів (рис. 3.6).

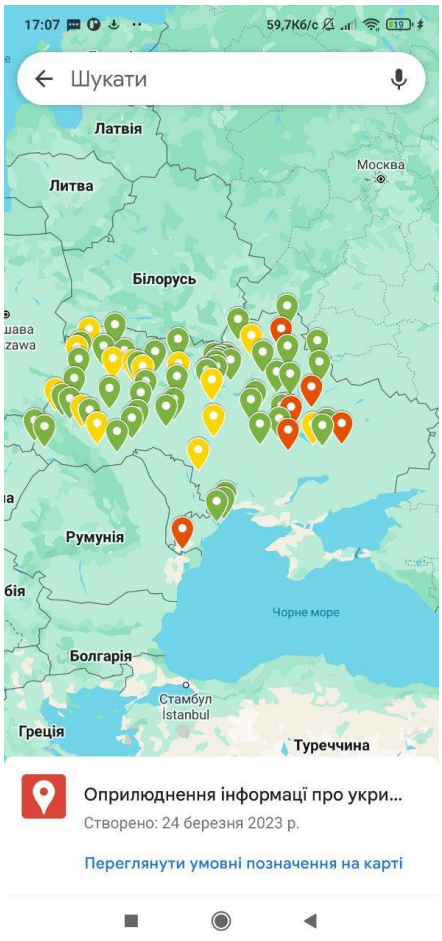


Рисунок 3.6 – Карта укриттів в Україні

Інтеграція цих функцій підвищує загальний рівень підготовки населення до надзвичайних ситуацій, надаючи не лише інструкції для негайного реагування, але й поради щодо підготовки до потенційних загроз. Це сприяє

формуванню культури готовності та знижує негативні наслідки від можливих інцидентів, забезпечуючи безпеку і збереження життя людей у прифронтових зонах.

3.6 Налаштування взаємодії з базою даних

Для створення бази даних були визначені правила доступу до контенту. Встановлено такі налаштування, що всі користувачі можуть читати дані, але записувати їх дозволено тільки авторизованим користувачам (рис. 3.7).

Рисунок 3.7 – Правила доступу до бази даних

Для здійснення реєстрації чи авторизації користувачів використовується електронна пошта та пароль (рис. 3.8).

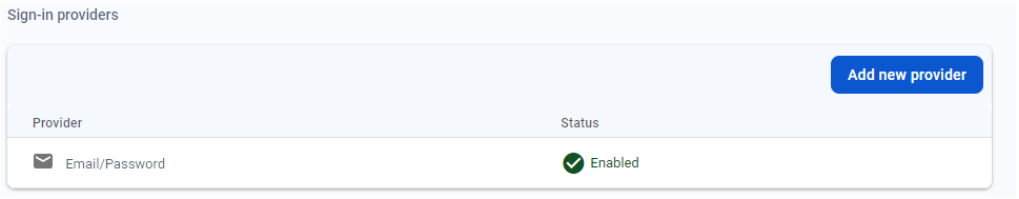


Рисунок 3.8 – Реалізація аутентифікації користувачів

Успішна аутентифікація надає доступ до особистих даних користувача та дозволяє здійснювати записи в базу даних, за умови наявності відповідних прав доступу.

Такий підхід до створення бази даних та забезпечення аутентифікації гарантує безпеку та конфіденційність даних, оскільки лише авторизовані користувачі можуть змінювати або додавати записи в систему.

Для зберігання даних користувачів застосовується віддалений сервер, який використовує функціонал Firebase. Це рішення дозволяє зберігати дані не на локальній пам'яті пристрою, а на сервері, що забезпечує їх віддалене зберігання. Такий підхід сприяє оптимізації роботи додатку і дозволяє ефективніше використовувати обмежені ресурси смартфона.

Для роботи з базою даних Firebase реалізовано інтерфейс Repository.Firebase, який містить основні потоки даних, необхідні для взаємодії з базою даних (**Error! Reference source not found.** – Repository.Firebase.kt). Отримані дані зберігаються у відповідних об'єктах, які відображають структуру бази даних (рис. 3.9). Це можуть бути, наприклад, інформація про користувачів, пости, типи оголошень та інші дані, пов'язані з функціоналом додатку.

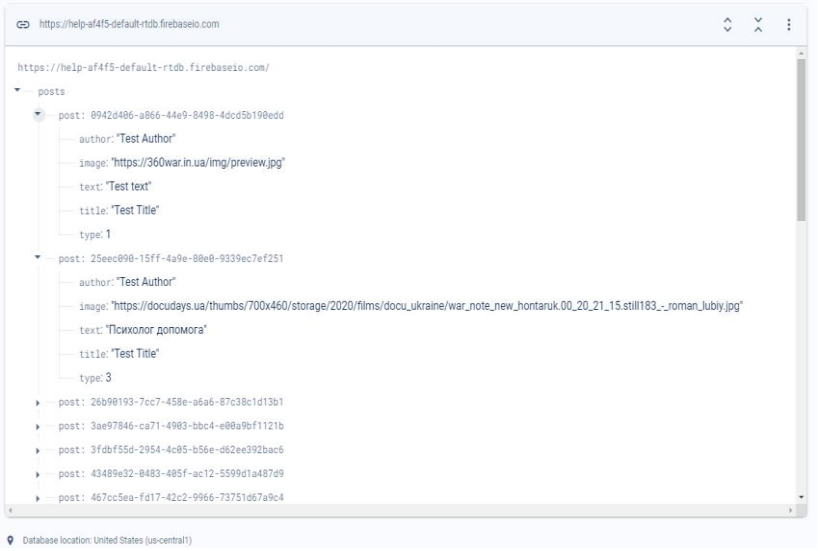


Рисунок 3.9 – Збережені дані

ВИСНОВКИ

У результаті виконання кваліфікаційної бакалаврської роботи розроблено мобільний додаток для підтримки осіб, які перебувають на прифронтових територіях та виконано усі поставлені завдання.

1) Проведено аналіз та літературний огляд сучасних джерел за тематикою розробки веб-додатків із метою формування постановки задачі для подальшої реалізації.
2) Проведено моніторинг аналогів даної системи задля виокремлення необхідних факторів.
3) Розглянуто та обрано інструменти розробки задля досягнення поставленої мети.
4) Спроєктовано дизайн сайту та створено карту системи.
5) Налаштований розподіл за ролями та доступ до різних функцій додатку.
6) Розроблено клієнт-орієнтований інтерфейс для зручного користування додатком.

Розроблена інформаційна система не є ідеальною та має напрямки для подальших удосконалень. До таких відноситься створення ролі для модерування постів, можливість залишати коментарі під оголошеннями.

Детальну інформацію про програмну реалізацію можливо переглянути в моєму GitHub репозиторії, перейшовши за посиланням.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Частка ринку Android та iOS: статистика 2022 року URL: https://root-nation.com/ua/news-ua/it-news-ua/ua-android-ios-statistika-2022/ (дата звернення: 03.05.2024).
2. What is Android? - Definition from Techopedia. URL: https://www.techopedia.com/definition/5415/android (дата звернення: 03.05.2024).
3. What is a mobile app (mobile application)? – TechTarget Definition. URL: https://www.techtarget.com/whatis/definition/mobile-app (дата звернення: 03.05.2024).
4. What Is an IDE? | Codecademy. URL: https://www.codecademy.com/article/what-is-an-ide (дата звернення: 03.05.2024).
5. Smyth Neil. Android Studio Iguana Essentials - Kotlin Edition: Developing Android Apps Using Android Studio 2023.2.1 and Kotlin. Payload Media, Inc. 53–72 p.
6. Brett Neutrron. Brett Neutrron - Kotlin Programming Fundamentals_ A Concise Guidebook.
7. Мобільні додатки, що допоможуть під час війни — Суспільне Новини. URL: https://suspilne.media/239406-mobilni-dodatki-voennogo-stanu-zastosunki-so-budut-korisnimi-pid-cas-vijni/ (дата звернення: 03.05.2024).
8. Zaid Kamil. My First Mobile App for Students. BPB Publications, 2024. 22–24 p.
9. Peter Späth. Pro Android with Kotlin: Developing Modern Mobile Apps with Kotlin and Jetpack. Apress, 2023. 4–6 p.
10. Get started with the NDK | Android NDK | Android Developers. URL: https://developer.android.com/ndk/guides (дата звернення: 04.05.2024).
11. Kris Hermans. Mastering React Native: A Comprehensive Guide to Learn React Native. Cybellium Ltd, 2024. 4–23 p.
12. Flutter - Dart API docs. URL: https://api.flutter.dev/ (дата звернення: 05.05.2024).
13. Android Studio Jellyfish | 2023.3.1 | Android Developers. URL: https://developer.android.com/studio/releases (дата звернення: 05.05.2024).
14. Features - IntelliJ IDEA. URL: https://www.jetbrains.com/idea/features/ (дата звернення: 05.05.2024).
15. Bernhard Steppan. Getting Started With Java Using Eclipse mastering the Language and the Development Platform. Elektor, 2023. 25–35 p.
16. Get Started with Visual Studio Code. URL: https://code.visualstudio.com/learn (дата звернення: 05.05.2024).

17. Pankaj Kumar. Building Android Projects with Kotlin: Use Android SDK, Jetpack, Material Design, and JUnit to Build Android and JVM Apps That Are Secure and Modular. BPB Publications, 2023. 21–40 p.
18. Brett Neutrron. Kotlin Programming Fundamentals_ A Concise Guidebook. Independently Published , 2024. 11–17 p.
19. Neil Smyth. Android Studio Flamingo Essentials - Java Edition: Developing Android Apps Using Android Studio 2022.2.1 and Java. Independently Published, 2023. 35–42 p.
20. GitHub - Dav7723/HelpUA. URL: https://github.com/Dav7723/HelpUA/tree/main (дата звернення: 20.05.2024).

ДОДАТОК А
МОДЕЛІ БАЗИ ДАНИХ

A1 HomeFragment.kt

```
package com.app.helpua.app.home

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.lifecycle.lifecycleScope
import androidx.navigation.fragment.findNavController
import androidx.recyclerview.widget.ConcatAdapter
import androidx.recyclerview.widget.LinearLayoutManager
import com.app.helpua.databinding.FragmentHomeBinding
import com.app.helpua.domain.utils.Results
import kotlinx.coroutines.launch
import
org.koin.androidx.viewmodel.ext.android.activityViewModel


class HomeFragment : Fragment() {

    private lateinit var binding: FragmentHomeBinding
    private val viewModel by
activityViewModel<HomeViewModel>()
    private val listAdapter = HomeAdapter()
    private val titleAdapter by lazy {
        TitleAdapter {

findNavController().navigate(HomeFragmentDirections.actionHome
FragmentToStartFragment())
        }
    }
    private val mergeAdapter by lazy {
ConcatAdapter(titleAdapter, listAdapter) }


    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {

        binding = FragmentHomeBinding.inflate(layoutInflater)

        binding.root.layoutManager =
LinearLayoutManager(context)
        binding.root.adapter = mergeAdapter
```

```
        viewModel.getIsAuth()
        subcribeToFlow()
        viewModel.subcribeToPosts()
        return binding.root
    }


    private fun subcribeToFlow() = with(viewModel) {
        lifecycleScope.launch {
            posts.collect {
                listAdapter.submitList(it)
            }
        }

        lifecycleScope.launch {
            user.collect {
                when (it) {
                    is Results.Success -> {

titleAdapter.submitList(mutableListOf(TitleModel(it.data)))
                    }

                    is Results.Error -> {

titleAdapter.submitList(mutableListOf(TitleModel(null)))
                    }
                }
            }
        }
    }
}
```

A2 AuthDataSource.kt

```
package com.app.helpua.data.datasources.firebase

import android.util.Log
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.UserProfileChangeRequest
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase
import com.app.helpua.data.datasources.DataSource
import com.app.helpua.data.datasources.utils.toUser
import com.app.helpua.domain.models.User
import com.app.helpua.domain.utils.DEFAULT_EXCEPTION
import com.app.helpua.domain.utils.Results

import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.tasks.await
import kotlinx.coroutines.withContext
```

```
class AuthDataSource : DataSource.Auth {

    private var auth: FirebaseAuth = Firebase.auth

    override suspend fun createUser(
        password: String,
        email: String,
        displayName: String
    ): Results<User> = withContext(Dispatchers.IO) {

        return@withContext try {
            val authResult =
                auth.createUserWithEmailAndPassword(email,
password).await()
            authResult.user?.updateProfile(

UserProfileChangeRequest.Builder().setDisplayName(displayName)
.build()
            )?.await()

            val user = authResult.user

            if (user != null) {
                Log.d("User:", user.toUser().toString())
                Results.Success(user.toUser())
            } else {
                Results.Error(Exception("User is Null"))
            }
        } catch (e: Exception) {
            Results.Error(e)
        }
    }

    override suspend fun signIn(
        password: String,
        email: String
    ): Results<User> {

        return try {
            val authResult = withContext(Dispatchers.IO) {
                auth.signInWithEmailAndPassword(email,
password).await()
            }.user

            when (authResult) {
                null -> Results.Error(Exception("User is
null"))
                else -> Results.Success(authResult.toUser())
            }

        } catch (e: Exception) {
```

```
            Results.Error(e)
        }
    }

    override suspend fun getCurrentUser(): Results<User> =
withContext(Dispatchers.IO) {

        val currentUser = auth.currentUser

        return@withContext if (currentUser != null) {
            Results.Success(currentUser.toUser())
        } else {
            Results.Error(Exception(DEFAULT_EXCEPTION))
        }

    }

    override suspend fun signOut(): Results<User?> =
withContext(Dispatchers.IO) {
        auth.signOut()

        return@withContext if (auth.currentUser == null) {
            Results.Success(null)
        } else {
            Results.Error(Exception("Sign Out Error"))
        }
    }
```

A3 CreatePostFragment.kt

```
package com.app.helpua.app.create

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.navArgs
import com.app.helpua.app.GUMMANITAR_TYPE
import com.app.helpua.app.MEDICAL_TYPE
import com.app.helpua.app.PSYHIC_TYPE
import com.app.helpua.databinding.FragmentCreatePostBinding
import
org.koin.androidx.viewmodel.ext.android.activityViewModel

class CreatePostFragment : Fragment() {

    private var _binding: FragmentCreatePostBinding? = null
    private val binding get() = _binding!!

    private val args by navArgs<CreatePostFragmentArgs>()
```

```
    private val viewModel by
activityViewModel<CreatePostViewModel>()

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding =
FragmentCreatePostBinding.inflate(layoutInflater)
        bindView()
        return binding.root
    }

    private fun getType(): Int = with(binding) {
        return when {
            rbGumanitar.isChecked -> GUMMANITAR_TYPE
            rbMedical.isChecked -> MEDICAL_TYPE
            rbPsyhical.isChecked -> PSYHIC_TYPE
            else -> 0
        }
    }

    private fun bindView() = with(binding) {
        ivClose.setOnClickListener {

requireActivity().onBackPressedDispatcher.onBackPressed()
        }

        btCreate.setOnClickListener {
            viewModel.createPost(
                userName = args.userName,
                title = etTitle.text.toString(),
                text = etText.text.toString(),
                image = etImageUrl.text.toString(),
                type = getType()
            )

requireActivity().onBackPressedDispatcher.onBackPressed()
        }
    }
}
```

A4 CreatePostViewModel.kt

```
package com.app.helpua.app.create

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.app.helpua.domain.models.Post
```

```
import com.app.helpua.domain.repositories.Repository
import kotlinx.coroutines.flow.MutableSharedFlow
import kotlinx.coroutines.flow.asSharedFlow
import kotlinx.coroutines.launch

class CreatePostViewModel(
    private val repository: Repository.Firebase
) : ViewModel() {

    private var _action = MutableSharedFlow<Boolean>(replay =
1)
    val action = _action.asSharedFlow()

    var selectedType = 0

    fun createPost(
        userName: String,
        title: String,
        text: String,
        image: String,
        type: Int
    ) {
        viewModelScope.launch {
            repository.writePost(
                Post(author = userName, title = title, text =
text, image = image, type = type)
            )
            _action.tryEmit(true)
        }
    }
}
```

A4 Repository.Firebase.kt

```
package com.app.helpua.domain.repositories

import com.app.helpua.domain.models.Post
import com.app.helpua.domain.models.User
import com.app.helpua.domain.utils.Results
import kotlinx.coroutines.flow.Flow

interface Repository {

    interface Firebase {
        suspend fun getCurrentUser(): Results<User>

        suspend fun signIn(password: String, email: String):
Results<User>

        suspend fun createUser(password: String, email:
String, displayName: String): Results<User>
```

```
            suspend fun signOut(): Results<User?>
            suspend fun writePost(post: Post)

            suspend fun readPosts(): Results<List<Post>>

            suspend fun subscribeToPosts(): Flow<List<Post>>

        }
    }
```

A4 SecurityPolicyFragment.kt

```
package com.app.helpua.app

import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.SimpleExpandableListAdapter
import androidx.fragment.app.Fragment
import com.app.helpua.R
import com.app.helpua.databinding.FragmentSecurityPolicyBinding


class SecurityPolicyFragment : Fragment() {

    private var _binding: FragmentSecurityPolicyBinding? = null
    private val binding get() = _binding!!

    private val groups = mutableListOf<Group>()

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding =
FragmentSecurityPolicyBinding.inflate(layoutInflater)

        createGroups()
        initView()

        return binding.root
    }

    override fun onDestroy() {
        super.onDestroy()

        _binding = null
    }
```

```
    private fun createChildList(groups: List<Group>):
List<List<Map<String, String>>> {
        val result: MutableList<List<Map<String, String>>> =
ArrayList()
        for (g in groups) {
            val secList: MutableList<Map<String, String>> =
ArrayList()
            for (c in g.children) {
                val child: MutableMap<String, String> = HashMap()
                child["Sub Item"] = c
                secList.add(child)
            }
            result.add(secList)
        }
        return result
    }

    private fun createGroupList(groups: List<Group>):
List<Map<String, String>>? {
        val result: MutableList<Map<String, String>> =
java.util.ArrayList()
        for (g in groups) {
            val m: MutableMap<String, String> =
java.util.HashMap()
            m["Group Item"] = g.name
            result.add(m)
        }
        return result
    }

    private fun initView() = with(binding) {
        val expListAdapter = SimpleExpandableListAdapter(
            requireActivity(),
            createGroupList(groups),
            R.layout.group_row,
            arrayOf("Group Item"),
            intArrayOf(R.id.row_name),
            createChildList(groups),
            R.layout.child_row,
            arrayOf("Sub Item"),
            intArrayOf(R.id.grp_child)
        )
        exList.apply {
            setAdapter(expListAdapter)

        }

        ivBack.setOnClickListener {
```

```
requireActivity().onBackPressedDispatcher.onBackPressed()
        }

        btMap.setOnClickListener {
            startMap()
        }
    }

    private fun startMap() {
        val videoIntent = Intent(Intent.ACTION_VIEW)
        videoIntent.data =

Uri.parse("https://www.google.com/maps/d/u/0/viewer?mid=1sELKMuMig
bzEzwUhzGcREs0Dic4OkQM&femb=1&ll=51.13625943057532%2C32.8902452918
2605&z=6")

        startActivity(videoIntent)
    }

    private fun createGroups() {
        val webG = Group("Дії під час повітряної тривоги")
        val mapG = Group("Поради щодо дій під час евакуації")
        val telG = Group("Як діяти населенню в місцях бойових
дій")
        val child4 = Group("Контакти екстрених служб")

        webG.addChild(getString(R.string.child_1) +
getString(R.string.child_3_1))
        mapG.addChild(
            "Якщо ви вдома під час евакуації:\n" +
                    "- виконуйте інструкції відповідних служб;\n"
+
                    "- будьте готові до того, що вам доведеться
рухатися повільно, особливо якщо людей багато;\n" +
                    "- не повертайтеся за речами, що залишилися в
будинку;\n" +
                    "- уникайте паніки та допомагайте іншим,
особливо дітям, людям з обмеженими можливостями та літнім
людям.\n\n" +
                    "Перед тим, як залишити житло, необхідно:\n" +
                    "- зачинити вікна;\n" +
                    "- вимкнути газ, воду та електрику;\n" +
                    "- забрати продукти з холодильника.\n" +
                    "- запасні ключі від квартири необхідно здати
представнику житлово-експлуатаційної організації.\n\n" +
                    "Громадяни повинні мати при собі:\n" +
                    "- паспорт;\n" +
                    "- військовий квиток;\n" +
                    "- документ про освіту;\n" +
                    "- трудову книжку або пенсійне посвідчення;\n"
+
```

```
                    "- свідоцтво про народження;\n" +
                    "- гроші і цінності;\n" +
                    "- продукти харчування і воду на 3 доби;\n" +
                    "- необхідний одяг і взуття загальною вагою не
більш як 50 кілограмів на кожного члена сім’ї.\n" +
                    "Дітям дошкільного віку вкладається у кишеню
або пришивається до одягу записка, де зазначається прізвище, ім’я
та по батькові, домашня адреса, а також ім’я та по батькові матері
і батька.\n\n" +

                    "Після евакуації:\n" +
                    "зібратися на місці збору, що було зазначене в
плані дій;\n" +
                    "намагайтеся отримати актуальну інформацію
щодо події від відповідних служб та дотримуйтесь їхніх
інструкцій;\n" +
                    "будьте готові до того, що може знадобитися
переселення на тимчасове місце проживання;\n" +
                    "Завжди пам'ятайте, що безпека важливіша за
будь-які матеріальні речі та від цього залежить ваше життя й життя
інших людей."
        )
        telG.addChild(getString(R.string.child_3))
        child4.addChild(
            "101 - пожежна охорона;\n" +
                    "\n" +
                    "102 - поліція;\n" +
                    "\n" +
                    "103 - швидка допомога;\n" +
                    "\n" +
                    "104 - аварійна служба газу.\n" +
                    "\n" +
                    "104 – Аварійна служба газової мережі\n" +
                    "\n" +

                    "109 – Довідкова служба"
        )

        groups.apply {
            add(webG)
            add(mapG)
            add(telG)
            add(child4)
        }

    }
}
```

ДОДАТОК Б
ЛІСТИНГ БІБЛІОТЕК

```
dependencies {
    implementation ("de.hdodenhof:circleimageview:3.1.0")
    implementation ("com.squareup.picasso:picasso:2.71828")
    implementation
("androidx.recyclerview:recyclerview:1.3.0")

    implementation("androidx.navigation:navigation-fragment-
ktx:2.7.4")
    implementation("androidx.navigation:navigation-ui-
ktx:2.7.4")
    implementation("androidx.fragment:fragment-ktx:1.6.1")

    implementation("io.insert-koin:koin-android:3.5.0")
    implementation("io.insert-koin:koin-annotations:1.2.0")
    val koinVer = "3.5.0"
    implementation("io.insert-koin:koin-core:$koinVer")
    implementation("io.insert-koin:koin-android:$koinVer")


    implementation ("org.jetbrains.kotlinx:kotlinx-
serialization-json:1.3.0")
    implementation(platform("com.google.firebase:firebase-
bom:32.8.1"))
    implementation("androidx.core:core-ktx:1.12.0")
    implementation("androidx.appcompat:appcompat:1.6.1")

implementation("com.google.android.material:material:1.11.0")

implementation("androidx.constraintlayout:constraintlayout:2.1
.4")
    implementation("com.google.firebase:firebase-analytics")
    implementation("com.google.firebase:firebase-auth:22.3.1")
    implementation("com.google.firebase:firebase-database-
ktx:20.3.1")
    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")

androidTestImplementation("androidx.test.espresso:espresso-
core:3.5.1")
}
```