МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ УПРАВЛІННЯ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____ Литвиненко О.Є.
«_____»_____2021 р.

ДИПЛОМНИЙ ПРОЄКТ
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ
«БАКАЛАВР»

Тема: «Програмний модуль батьківського контролю з функцією *GPS*-моніторингу»

Виконавець: _____ Таран Є.В.

Керівник: _____ Нечипорук В.В.

Нормоконтролер: _____ Тупота Є.В.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет Кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Спеціальність 123 «Комп'ютерна інженерія»
(шифр, найменування)

Освітньо-професійна програма «Системне програмування»
Форма навчання денна

ЗАТВЕРДЖУЮ
Завідувач кафедри

Литвиненко О.Є.
«_____»_____2021 р.

ЗАВДАННЯ
на виконання дипломної роботи (проєкту)
Тарану Євгену Вікторовичу
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи (проекту) «Програмний модуль батьківського контролю з функцією *GPS*-моніторингу»
затверджена наказом ректора від « 04 » лютого 2021 р. № 135/ст.
2. Термін виконання роботи (проєкту): з 17 травня 2021 р. по 20 червня 2021 р.
3. Вихідні дані до роботи (проєкту): програмна документація, ДСТУ, мова програмування *Java*, середовище розробки *Android Studio*
4. Зміст пояснювальної записки: аналіз побудови програмних модулів батьківського контролю, програмний модуль батьківського контролю з функцією *GPS*-моніторингу, програмне забезпечення модуля батьківського контролю з функцією *GPS*-моніторингу, інструкція користувача
5. Перелік обов'язкового графічного (ілюстративного) матеріалу:
1) структурна схема програмного модуля;
2) функціональна схема програмного модуля;
3) схема алгоритму перегляду змін точки місцезнаходження користувача;
4) схема алгоритму відправлення запиту на додавання в список друзів;
5) зовнішній вигляд основних вікон програмного модуля батьківського контролю з функцією *GPS*-моніторингу.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1.	Провести огляд літератури за темою дипломного проекту та аналіз існуючих систем.	17.05.2021 – 18.05.2021	
2.	Зробити вибір середовища програмування і компонентів програмної утиліти	18.05.2021 – 20.05.2021	
3.	Розробити структуру програмних засобів системи	21.05.2021 – 22.05.2021	
4.	Розробити програмні засоби. Провести відладку програмних засобів	23.05.2021 – 05.06.2021	
5.	Написати пояснювальну записку	06.05.2021 – 10.06.2021	
6.	Підготувати графічний та ілюстративний матеріал	11.06.2021 – 12.06.2021	
7.	Підготувати доповідь та презентацію	12.06.2021 – 13.06.2021	

7. Дата видачі завдання: «_17_» ___травня___ 2021 р.

Керівник дипломної роботи (проєкту) _____ Нечипорук В.В.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Таран Є.В.
(підпис студента)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Програмний модуль батьківського контролю з функцією *GPS*-моніторингу»: 68 сторінок, 19 рисунків, 15 літературних джерел, 2 таблиці, 1 додаток.

МІСЦЕЗНАХОДЖЕННЯ, *GPS*-ТРЕКІНГ, ДОДАТОК, ЗАПИТ, БАТЬКІВСЬКИЙ КОНТРОЛЬ, ГЕОЛОКАЦІЯ.

Об'єкт – *GPS*-моніторинг місцезнаходження людини.

Предмет – програмний модуль батьківського контролю з функцією *GPS*-моніторингу.

Мета дипломного проекту – розробити програмний модуль батьківського контролю з функцією *GPS*-моніторингу.

Метод проектування – мова програмування *Java*, інтегроване середовище розробки програмного забезпечення *Android Studio*.

Результати дипломного проектування рекомендується використовувати під час подорожей з дітьми, а також в моменти, коли батьки відсутні під час їх самостійного пересування, що знизить ймовірність виникнення ризику пропажі дітей.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

GPS – Global Positioning System (система глобального позиціювання)

LBS – Location-based service (служба, заснована на місцезнаходженні)

APK – Android Package

XML – Extensible Markup Language

SDK – Software Development Kit

JDK – Java Development Kit

ВСТУП

Основним завданням додатків батьківського контролю з використанням *GPS*-трекінгу є забезпечення безпеки для дитини під час різноманітних подорожей або в моменти, коли батьки відсутні в процесі їх самостійного пересування по вулицях міста. На жаль, на сьогоднішній день у батьків є багато причин для переживань пов'язаних з безпекою та благополуччям власної дитини, особливо, якщо підростаюче покоління робить тільки перші кроки до того, щоб самостійно пізнавати світ без допомоги у цьому рідних для нього людей. Лише думка про те, що власна дитина може загубитися в натовпі людей або пропасти безвісти може навести страх на любих батьків. Саме тому, щоб мінімізувати ймовірність виникнення ризику неочікуваної пропажі дітвори, використовують додатки батьківського контролю з функцією *GPS*.

Названі вище додатки в якості головного інструменту використовують *GPS*-маяки, які можуть відслідковувати місцезнаходження необхідної людини у будь-якій точці світу. Такі програмні засоби в якості даних використовують інформацію отриману зі супутників, завдяки чому похибка у знаходження точки локації засобу, що приєднаний до *GPS*, може складати всього від десяти до двадцяти метрів. Саме ця точність відстеження місцезнаходження пристроїв, а також можливість переглядати дані про зміни точки геолокації в онлайн режимі є критично важливими в питанні необхідності подібних додатків при роботі з дітьми, які можуть загубитися або потрапити в різного роду халепу.

Якщо розглядати детальніше процес отримання подібними додатками можливості оперувати даними про місцезнаходження дитини, то можна виділити певні етапи. Для початку роботи супутники формулюють координати місцезнаходження *GPS*-маяка, дані геолокації яких згодом за допомогою *SIM*-карти телефону дитини відправляють до спеціального додатку на телефон батьків.

Є певні додатки, які побудовані таким чином, що користування функціями глобальної навігаційної системи *GPS* не потрібне. Натомість подібні програмні

засоби використовують в якості трекерів для локації пристроїв вежі мобільних мереж (*LBS*). Частіше за все подібні можливості є суто опціональними та їх з легкістю можна виключити. Одним з недоліків подібних рішень розробників програмного забезпечення є те, що використання технології *LBS* має дуже високі похибки в питанні знаходження точки геолокації.

До можливих ситуацій в яких можна використати подібний додаток відносяться самі звичайні події в житті дитини, наприклад, похід в школу. Завдяки мобільному додатку батьківського контролю з функцією *GPS*-моніторингу батьки здатні контролювати пересування дітвори до навчального закладу. Завдяки подібним інструментам можна дізнатися де дитина находиться на даний момент часу, чи є вона на шляху додому. За допомогою певних функцій додатку є можливим переглядати в онлайн режимі процес її пересування.

Незамінною подібна програма є також в моменти різноманітних подорожей та просто виходів в громадські місця з дитиною, до яких можна віднести різного роду визначні пам’ятки міст, місця для екскурсій, магазини. В таких місцях з доволі великою вірогідністю можна побачити велике скупчення людей. В такому натовпі дитина може запросто загубитися або відстати, враховуючи той факт, що батьки постійно тримали її в полі зору та ретельно контролювали її переміщення, адже ніхто, навіть самі уважні люди, не є застрахованим від відволікань. В такому випадку на допомогу батькам може прийти мобільний додаток батьківського контролю з функцією *GPS*-моніторингу, який доволі швидко допоможе віднайти місцезнаходження дітвори.

На даний момент часу ринок програмних засобів пропонує різноманітні додатки для здійснення батьківського контролю, та не дивлячись на все різноманіття програмного продукту від недоліків вони не застраховані. Незалежно від того, який саме додаток або інструмент вирішив обрати користувач варто пам’ятати, що використане рішення повинно відповідати певним критеріям, які задовольнятимуть усі вимоги батьків, щодо захисту дітвори від різних факторів небезпеки та ніяк не вплине на зручність користування телефоном для самої дитини. До таких критеріїв можуть відноситися розширення

функціональних можливостей гаджета, а також оптимальна енергоємність додатку, яка не впливає на тривалість роботи сматрфону. Програми батьківського контролю застосовуються для вирішення різноманітних завдань і є найбільш ефективними у тих випадках, коли захищають ще незміцнілий дитячий розум від непризначеної для його психіки інформації повністю закриваючи до неї доступ, шляхом відслідковування активності, яку здійснює дитина на вулицях міста та переглядаючи місцезнаходження дітвори в даний момент часу. Широке використання таких програм – це надійний засіб надати необхідну безпеку організму, що ще розвивається.

Саме ці критерії і послужили для вибору теми даного дипломного проекту. Метою даного проекту є розробка програмного модуля батьківського контролю з функцією *GPS*-моніторингу.

В першому розділі описано сучасні програмні рішення та проаналізовано їх переваги і недоліки. На основі цього прийняті рішення щодо програмного модуля.

В другому розділі сформульовано функціональні вимоги програмного модуля, а також розроблено структуру та функціональну схему.

В третьому розділі програмно реалізовано алгоритми основних можливостей програмного модуля.

В четвертому розділі описано інтерфейс користувача, а також проведено тестування програмного модуля.

РОЗДІЛ 1
АНАЛІЗ ПРИНЦИПІВ ПОБУДОВИ ПРОГРАМНИХ МОДУЛІВ БАТЬКІВСЬКОГО КОНТРОЛЮ

1.1. Типові інструменти батьківського контролю на операційних системах мобільних пристроїв

За принципом впровадження інструменти для здійснення контролю над діями дитини можна розділити на дві великі групи. Першу утворюють різноманітні системні засоби, які надають розробники операційних систем і платформ для мобільних телефонів та планшетних комп'ютерів. Для прикладу наведемо відповідні інструменти двох найбільш популярних операційних систем:

– операційна система *Android*. В наш час мобільні операційні системи володіють дуже малою кількістю інструментів, що дозволяють вести спостереження за життєдіяльністю людини і є дієвими тільки лише для дуже малих дітей. Варто розуміти, що з деяким часом молодий користувач телефону стає все більш досвідченішим у використанні різноманітних веб-сервісів та додатків, тому його дії слід контролювати все більш досконалими методами. У випадку операційної системи *Android* батьки можуть налаштувати фільтри дозволеного контенту лише на рівні магазину *Google Play*. Для цього необхідно в налаштуваннях включити функцію «Батьківський контроль». Платформа дозволяє користувачу налаштовувати фільтри в залежності від віку дитини, тобто виставивши відповідний вік система підлаштовується під задані вказівки та починає відображати відповідний зміст сторінок. Проблема полягає в тому, що дитина, навчившись використовувати різноманітні браузери, може легко обійти задані заборони [1];

– операційна система *IOS*. Інструменти, що впроваджують батьківський контроль на телефонах даної операційної системи, не дозволяють стороннім продуктам брати участь в управлінні пристроєм. Це пояснюється тим фактом, що контроль за діями дитини на даній платформі здійснено досконало. На смартфонах під управлінням *IOS* ефективно працюють функції обмеження конфіденційності та контенту. Також широкого вжитку здобула функція екранного часу – це функція, що дозволяє в режимі реального часу отримувати доступ до звітів, в яких показано, як довго користувач проводить час в різноманітних продукціях компанії *Apple*. Перевагами даних функцій є те, що батькам надається широкі можливості в провадженні вікового цензу, блокуванні потенційно небезпечного контенту, обмеженні доступу до різноманітних програм, а також забороні проводити транзакції в магазині *Apple*. Недоліками подібних рішень розробників операційної системи є те, що батьки не можуть налаштувати необхідні обмеження дистанційно.

Друга група є ще більш різноманітною. Вхідні до неї інструменти постачаються за допомогою різноманітних додатків або програм, що обмежують відповідні функції гаджетів. Варто зазначити, що за допомогою подібних засобів можуть створюватися певні шаблони з дозволеними для дитини додатками та функціями, також є можливим повна заборона або виведення попереджень при здійснені спроби використання того, чи іншого програмного забезпечення. До цієї групи належать:

– веб-фільтри. В різноманітних програмах та телефонах подібний інструмент реалізується по-різному, але характерна риса подібної функції полягає в тому, що певний додаток має змогу використовувати дані з бази даних, в якій зберігаються небажані для дітей ресурси, і при спробі отримати до них доступ автоматично блокує всі спроби користувача;

– обмеження часу використання пристрою. Батьки можуть зазначити конкретні часові рамки під час яких робота телефону буде припинена або заблокована, а дитина зможе займатися відповідними справами не відволікаючись на використання різних додатків;

– обмеження використання додатків. Дана функція є більш спрощеною версією можливості обмеження часу використання пристрою, але в цьому випадку заборона використання поширюється не на весь телефон, а на певні його додатки. Також за допомогою цього інструменту здійснення контролю батьки можуть заборонити грати в ігри, які містять сцени насилля та обмежити доступ до певних програм в яких можна здійснювати різноманітні грошові операції [2];

– сигнал *SOS*. За допомогою додатків з подібною функцією дитина в разі екстреної необхідності може подати сигнал, що вона потрапила в біду. Також існують певні засоби, які при реалізації цього сигналу продемонструють де на даний момент знаходиться користувач;

– *GPS*-моніторинг. Додатки з подібною функцією дозволяють дізнатися не тільки де на даний момент перебуває дитина, але й подивитися історію її переміщень;

– контроль спілкування. Список подібних програм різноманітний. До можливостей, які надають подібні додатки, можна віднести такі: змога прослідкувати, як дитина поводить себе в соціальних мережах, перегляд СМС-повідомлень, перегляд списку нещодавніх дзвінків, а також прослуховування розмов.

1.2. Переваги і недоліки програм батьківського контролю

На даний момент часу ринок програмних засобів пропонує різноманітні додатки для здійснення батьківського контролю, та не дивлячись на все різноманіття програмного продукту від недоліків вони не застраховані. Незалежно від того, який саме додаток або інструмент вирішив обрати користувач варто пам’ятати, що використане рішення повинно відповідати певним критеріям, які задовольнять усі вимоги батьків, щодо захисту дітвори від різноманітних факторів небезпеки та ніяк не вплине на зручність користування телефоном для самої дитини. До таких критеріїв можуть відноситися розширення функціональних можливостей гаджета, а також оптимальна енергоємність

додатку, яка не впливає на тривалість роботи сматрфону. Існує доволі багато додатків, які відповідають виділеним вимогам, але варто розуміти, що кожен користувач по різному може оцінювати та віддавати переваги додаткам. Якщо один віддасть перевагу безлічі функцій, то другий – простоті у використанні.

1.2.1. Мобільний додаток *Find My Kids*

Додаток здійснює комплексний контроль усіх захоплень дитини, а саме від його місцезнаходження та маршруту (рис. 1.1) до кількості хвилин в різноманітних додатках, що знаходяться на телефоні користувача. Однією з особливостей програми є те, що додаток нічого не блокує, а натомість відсилає усі дані батькам, які в свою чергу обговорюють та приймають рішення на рахунок тих, чи інших обмежень.

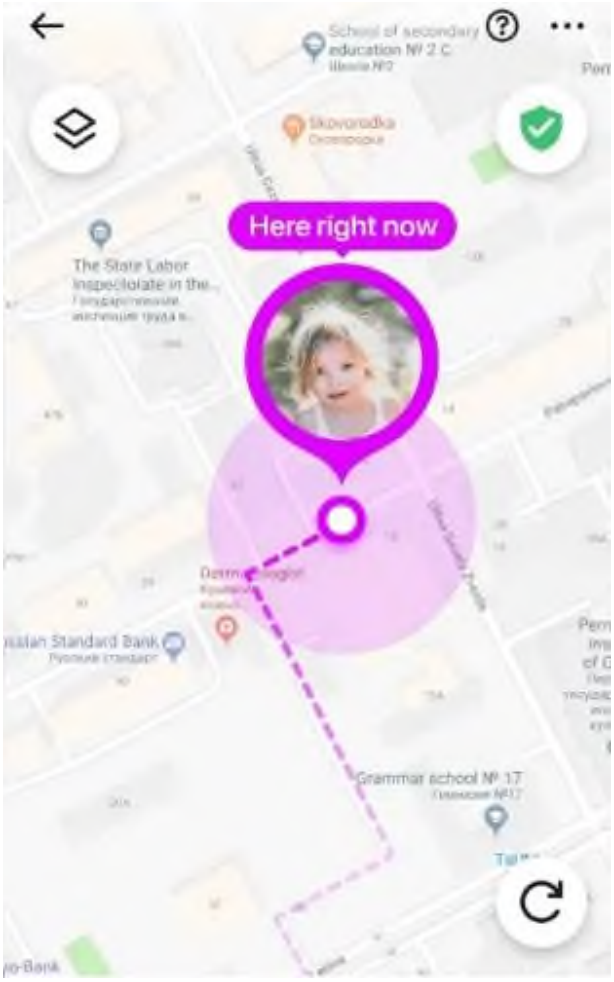

Рис. 1.1. Приклад роботи додатку *Find My Kids*

Даний програмний засіб дає змогу зрозуміти, чи не використовується смартфон в ті моменти часу, коли дітворі вже потрібно спати, а також дозволяє дізнатися батькам скільки дитина витрачає час в різноманітних соціальних мережах. За допомогою додатку батькам надана можливість отримати доступ до камери та мікрофону телефону, що допоможе дізнатися що саме на даний момент знаходиться біля дитини та що його оточує. Є можливість використати представлену програму не тільки на телефоні, але й на смарт-годиннику, що зробить її використання більш зручним.

До недоліків рішення можна віднести те, що досить велика кількість скарг та нарікань адресована до точності визначення локації телефону, що є досить критичним зауваженням для програм батьківського контролю з функцією *GPS*-моніторингу.

1.2.2. Мобільний додаток *Locategy*

Представлений мобільний додаток дає змогу користувачу визначити поточне положення дитини на карті та знаходження її в певній геолокації. Це є дуже корисною функцією, адже дає змогу дізнатися, якщо людина потрапила в небезпечний район на карті. Також програма має можливість відправляти повідомлення про знаходження людини в певній будівлі, наприклад, в школі або технікумі.

Окрім функцій *GPS*-моніторингу програмний засіб також включає в себе функції контролю іншими додатками, а саме контроль вмісту смартфону з доступною можливістю стирання усіх неналежних даних. Присутня можливість на певний час заблокувати телефон, що є корисною функцією під час проведення навчання або в нічну годину (рис. 1.2).

До недоліків мобільного додатку слід зазначити те, що для відстеження осіб на профіль користувача є можливим додати лише від трьох до п’яти людей, що є досить малою кількістю, якщо враховувати те, що з часом телефони у дітей за певних причин можуть змінюватися.

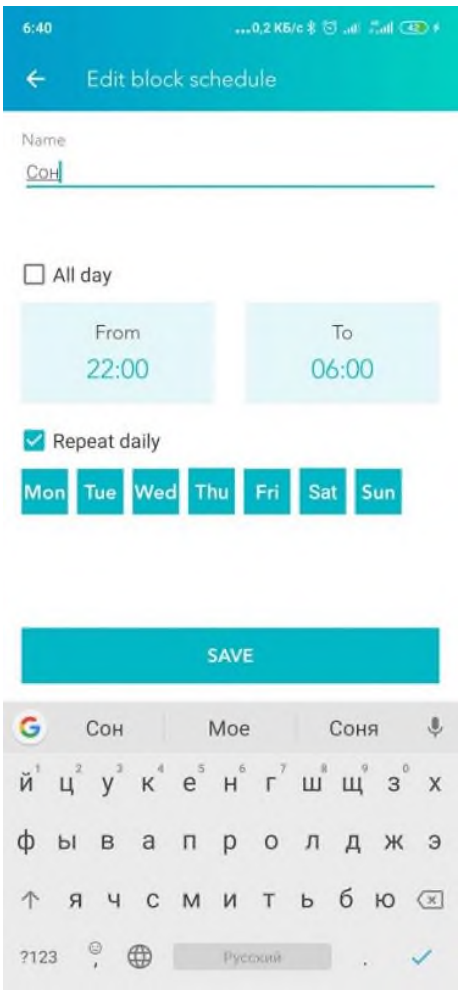

Рис. 1.2. Введення блокування телефону в нічний час

1.2.3. Мобільний додаток *KidControl*

Як і попередні програми запропонований додаток націлений на відслідковування поточної геолокації людини, але він має певні особливості, до яких належать зберігання історії переміщення за попередні два дні, налаштування небезпечних зон, пошук вкраденого телефону. Є можливою використання кнопки *SOS*, яка сповістить батьків про певну проблемну ситуацію. Для відображення місця знаходження людини здійснено можливість натиснення на іконку з фотографією, яка відповідає за певний контакт, після чого на карті у відповідному місці відобразиться точка. До додаткових функцій слід віднести відправлення сповіщення про досить малий рівень заряду на телефоні.

Точністю визначення геолокації слід завдячувати сервісу *LBS*. Це певний сервіс, який допомагає визначити місцезнаходження об’єкту за допомогою використання станцій операторів *GSM* стандарту. Виходячи з цього можна зробити висновок, що точність знаходження геолокації телефону є досить низькою, це приблизно сто або декілька тисяч метрів. Для того, щоб передача даних про місцезнаходження телефону через *LBS* все таки відбулося необхідне досить якісне Інтернет з’єднання, яке користувач програми не завжди може забезпечити. Якщо ж відбулося так, що Інтернет з’єднання на телефоні дитини відсутнє, тоді на карті батьківського телефону відобразиться іконка з символом антени.

З вищесказаного можна зробити висновок, що одним з недоліків додатку є досить поганої якості відображення місця локації смартфону, яке залежить від багатьох параметрів, які дитина не завжди в змозі контролювати. Також одним з мінусів використання даного програмного забезпечення є те, що в онлайн режимі перегляду локації об’єкту місцезнаходження не змінюється, для чого, щоб отримати останні дані потрібно перезавантажувати дану функцію.

1.2.4. Мобільний додаток *Parental Control*

До можливостей представленого додатку можна віднести як контроль екранного часу, тобто функція для відстеження та аналізування часу використання різноманітних засобів на смартфоні дитини, так і функцію *GPS*-трекеру, яка допомагає дізнатися місцезнаходження телефону користувача. Варто зазначити, що програмний засіб також містить в собі можливість впровадження веб-фільтів, які не дозволять особам, що користуються телефоном заходити на сайти з сумнівним контентом. Є можливість блокувати доступ до різного роду іграм та соціальним мережам.

Також додаток має інтеграцію з програмою *YouTube*, за допомогою якої батьки можуть контролювати які саме відео дивилася дитина (рис. 1.3) та блокувати доступ не тільки до самого відеоролику, але й до самого каналу.

Програма також виконує догляд за здоров’ям дитини, а саме за допомогою використання фронтальної камери телефону або планшету засіб здійснює функцію по збереженню зору дитини та в потрібний момент повідомить про доволі близьку відстань очей до екрану пристрою. Також автоматична перевірка навколишньої освітленості допоможе змінити яскравість екрану смартфону в потрібний момент часу.

До вищенаведених особливостей програми слід віднести також контроль спілкування, адже засіб дозволяє переглядати переписку дитини в таких системах обміну повідомленнями як *WhatsApp* та *Viber*, переглядати надіслані та отримані фотографії, тощо.

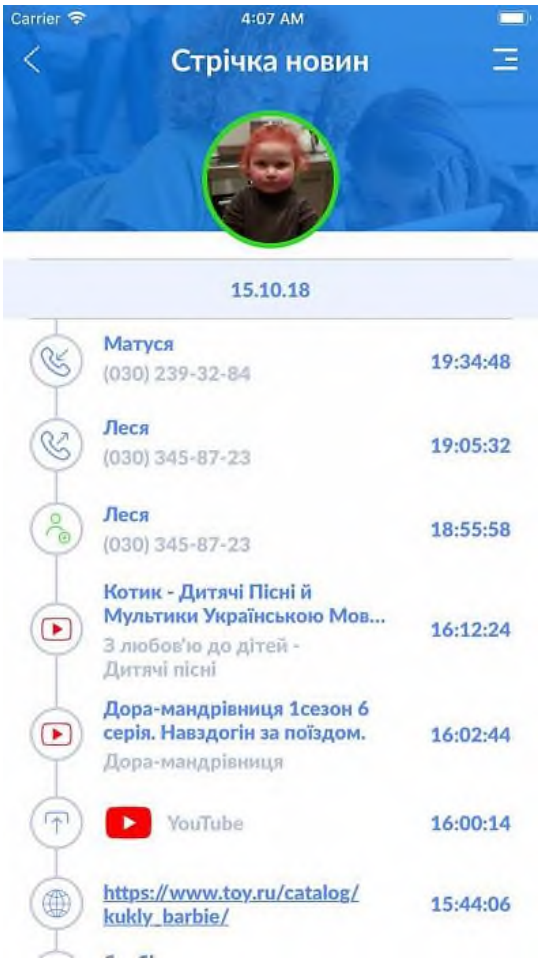

Рис. 1.3. Перегляд стрічки дій сторінки дитини

До недоліків слід віднести досить слабкий за можливостями *GPS*-трекер, адже в програмі абсолютно відсутня можливість слідкувати за

місцезнаходженням дитини при умові, якщо додаток є вимкненим на її телефоні, що робить неможливим знати в будь-який момент часу де саме вона знаходиться.

1.3. Постановка завдання проектування

Програми батьківського контролю застосовуються для вирішення різноманітних завдань і є найбільш ефективними у тих випадках, коли захищають ще незміцнілий дитячий розум від непризначеної для його психіки інформації повністю закриваючи до неї доступ, шляхом відслідковування активності, яку здійснює дитина на вулицях міста та переглядаючи місцезнаходження дітвори в даний момент часу. Широке використання таких програм – це надійний засіб надати необхідну безпеку організму, що ще розвивається.

Метою дипломного проекту є розробка програмного модуля батьківського контролю з функцією *GPS*-моніторингу.

Відповідно до проведеного аналізу переваг та недоліків сучасних програмних рішень в області додатків батьківського контролю та відповідно до мети проектування в даному проекті поставлено наступні завдання:

– проаналізувати сучасні програмні рішення та методи створення додатків батьківського контролю з використання *GPS*-трекінгу;

– реалізувати можливість роботи програмного модуля з серверною базою даних для оперування інформацією користувачів;

– розробити структуру програмного модуля відображення геолокації користувачів;

– реалізувати алгоритм додавання до профілю користувача аккаунтів людей для *GPS*-моніторнигу їх місцезнаходження;

– реалізувати алгоритм оновлення та перегляду змін в онлайн режимі точки місцезнаходження користувачів шляхом розробки програмного модуля;

– реалізувати в програмному модулі функцію слідкування за місцезнаходженням людини при умові, якщо додаток є вимкненим на її телефоні;

– протестувати роботу програмного модуля.

1.4. Висновки до розділу

1) За результатами аналізу існуючих програмних рішень сформовано вимоги до програмного модуля, які б забезпечували його універсальність, а саме:

– мати можливість створення та адміністрування користувачем профілів за допомогою введеної електронної почти та паролю;

– мати можливість перегляду та пошуку профілів користувачів, що є зареєстрованими в програмному модулі батьківського контролю з функцією *GPS*-моніторингу;

– мати можливість відправлення запиту на додавання профілю користувача у відповідний список для подальшої роботи з ним;

– мати можливість перегляду та пошуку необхідного отриманого запиту на додавання профілю користувача;

– мати можливість відхилити або прийняти запит від іншого користувача на додавання профілю у відповідний список для подальшої роботи з ним;

– мати можливість відслідковувати точку місцезнаходження на карті доданого профілю користувача;

– мати можливість користувачеві змінювати розмір карти на якій знаходиться точка місцезнаходження відповідного профілю;

– мати можливість переглядати в онлайн режимі зміни місцезнаходження доданого профілю користувача;

– мати можливість переглядати дані про момент часу в який профіль користувача, що відслідковується, знаходиться в конкретній точці геолокації;

– мати можливість співпраці з операційною системою *Android*.

2) Проаналізовано методи створення програмного модуля батьківського контролю з функцією *GPS*-моніторингу та обрано мову програмування *Java*, та інтегроване середовище розробки програмного забезпечення *Android Studio*.

РОЗДІЛ 2
ПРОГРАМНИЙ МОДУЛЬ БАТЬКІВСЬКОГО КОНТРОЛЮ З ФУНКЦІЄЮ *GPS*-МОНІТОРИНГУ

2.1. Функціональні вимоги

До функціональних вимог програмного модуля батьківського контролю з функцією *GPS*-моніторингу належать:

– можливість роботи програмного модуля з серверною базою даних для оперування інформацією користувачів;

– можливість створення користувачем нових профілів за допомогою введених даних, таких як електронна пошта, ім’я та фамілія, а також пароль;

– можливість адміністрування користувачем своїх створених раніше профілів, що є зареєстрованими в програмному додатку;

– можливість перегляду та пошуку необхідних профілів користувачів, що є зареєстрованими в програмному модулі батьківського контролю з функцією *GPS*-моніторингу;

– можливість відправлення запиту на додавання профілю користувача у відповідний список друзів для подальшої роботи з ним;

– можливість отримання сповіщень на телефон користувача про отримані від інших профілів запити, що його аккаунт бажають додати у відповідний список друзів;

– можливість перегляду та пошуку необхідного отриманого запиту від іншого користувача програмного модуля на додавання у список друзів;

– можливість відхилити або прийняти запит від іншого користувача на додавання профілю у відповідний список для подальшої роботи з ним;

– можливість відслідковувати точку місцезнаходження на карті світу доданого профілю користувача;

– можливість користувача програмного модуля змінювати отримане зображення карти на якій знаходиться точка місцезнаходження відповідного профілю;

– можливість переглядати в онлайн режимі зміни місцезнаходження доданого профілю користувача;

– можливість переглядати дані про момент часу в який профіль користувача, що відслідковується, знаходиться в конкретній точці геолокації;

– можливість взаємодіяти зі стандартними засобами операційної системи *Android* пов’язаними з картами, такими як прокладання маршруту від заданої користувачем адреси до точки геолокації профілю, що відслідковується, а також завантаження сторінки з відомостями та фотографіями про те, що знаходиться біля адреси профілю, та надання можливості дізнатися його координати.

2.2. Структура програмного модуля

На основі проведеного аналізу функціональних вимог програмного модуля батьківського контролю з функцією *GPS*-моніторингу побудовано структурну схему, яку умовно можна поділити на дві частини, а саме структурна схема серверного блоку (рис. 2.1) та структурна схема клієнтського блоку (рис. 2.2).

До структурної схеми серверного блоку входять наступні компоненти:

– база даних, елементами якої є сукупність особистих даних користувача, якими вона оперує;

– блок роботи з базою даних, за допомогою якого встановлюється зв'язок між серверною частиною програмного модуля та самою базою даних;

– блок оновлення індивідуальних ідентифікаторів користувачів. Отриманий в ході роботи блоку ідентифікатор визначає запис про дані користувача та у разі успішної авторизації стає ключем для доступу до служб програмного модуля;

– блок для збереження та відправлення даних про координати місцезнаходження користувача, а саме відомості про довготу та ширину його геолокації;

– блок зберігання особистих даних користувача, до яких належать відомості про електронні адреси, паролі та користувацькі ідентифікатори;

– блок оперування профілями, які є доступними користувачу в блоках клієнтської частини програмного модуля, до яких відносяться блок відображення та пошуку усіх профілів, яким користувач може надіслати запит на додавання їх до списку друзів, блок відображення усіх можливих профілів місцезнаходження, яких користувач може відслідкувати, блок відображення та пошуку усіх запитів на додавання до списку друзів отриманих від інших користувачів.

Рис. 2.1. Структурна схема серверного блоку програмного молуля

До структурної схеми клієнтського блоку входять наступні компоненти:

– база даних, елементами якої є сукупність особистих даних користувача, якими вона оперує;

– блок роботи з базою даних, за допомогою якого встановлюється зв'язок між клієнтською частиною програмного модуля та самою базою даних;

– інтерфейс користувача, який виконує функцію передачі інформації між користувачем та блоком роботи з базою даних;

– блок роботи з профілями користувачів, що надає можливість програмному модулю працювати та змінювати в ході роботи вже створені профілі або натомість створювати нові аккаунти для їх подальшого використання;

– блок відображення усіх можливих профілів місцезнаходження, яких користувач може відслідкувати. На самому початку роботи з програмним модулем батьківського контролю даний блок є пустим, а заповнити його можна використовуючи спеціальні блоки додавання нових профілів;

– блок відображення та пошуку усіх профілів, яким користувач може надіслати запит на додавання їх до списку друзів, що зробить можливим відслідковування їх місцезнаходження;

– блок відправлення запиту на додавання конкретного профілю до списку друзів. Після відправлення запиту він надійде до спеціального блоку, де інший користувач зможе його переглянути;

– блок відображення та пошуку усіх запитів на додавання до списку друзів отриманих від інших користувачів. Даний блок надає можливість переходу до блоків відхилення запиту та прийняття запиту;

– блок відхилення запитів забороняє користувачу, що надсилав запит на додавання до списку друзів, отримувати та оперувати даними місцезнаходження конкретного профілю;

– блок прийняття запитів дозволяє користувачу, що надсилав запит на додавання до списку друзів, подальше отримування та оперування даними місцезнаходження доданого профілю;

– блок *GPS*-моніторнигу профілю дозволяє користувачу отримувати та оперувати даними точки місцезнаходження доданого аккаунту.

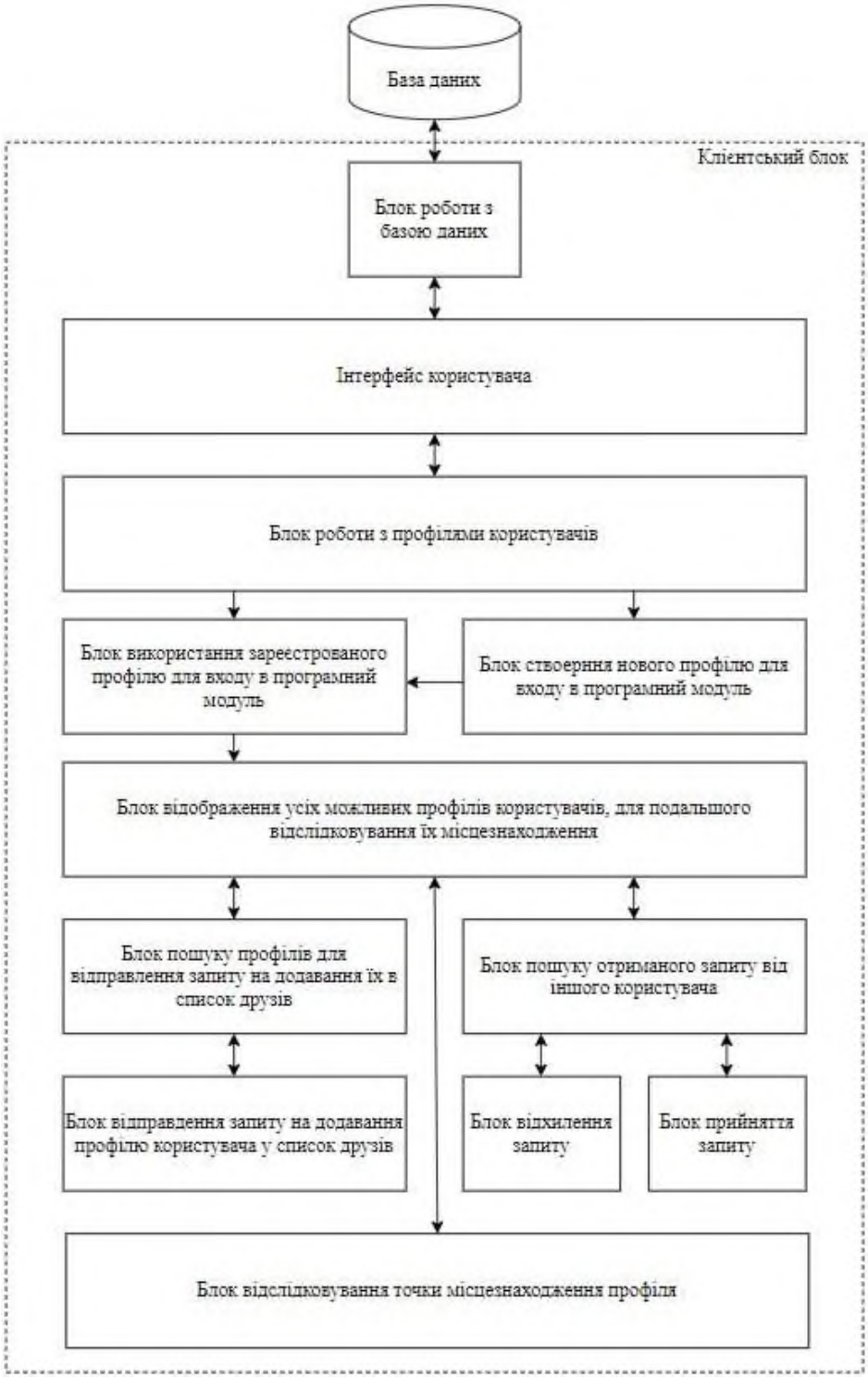

Рис. 2.2. Структурна схема клієнтського блоку програмного молуля

Загальна логічна послідовність виконання функцій програмного модуля батьківського контролю з функцією *GPS*-моніторингу наведена на діаграмі прецедентів (рис. 2.3).

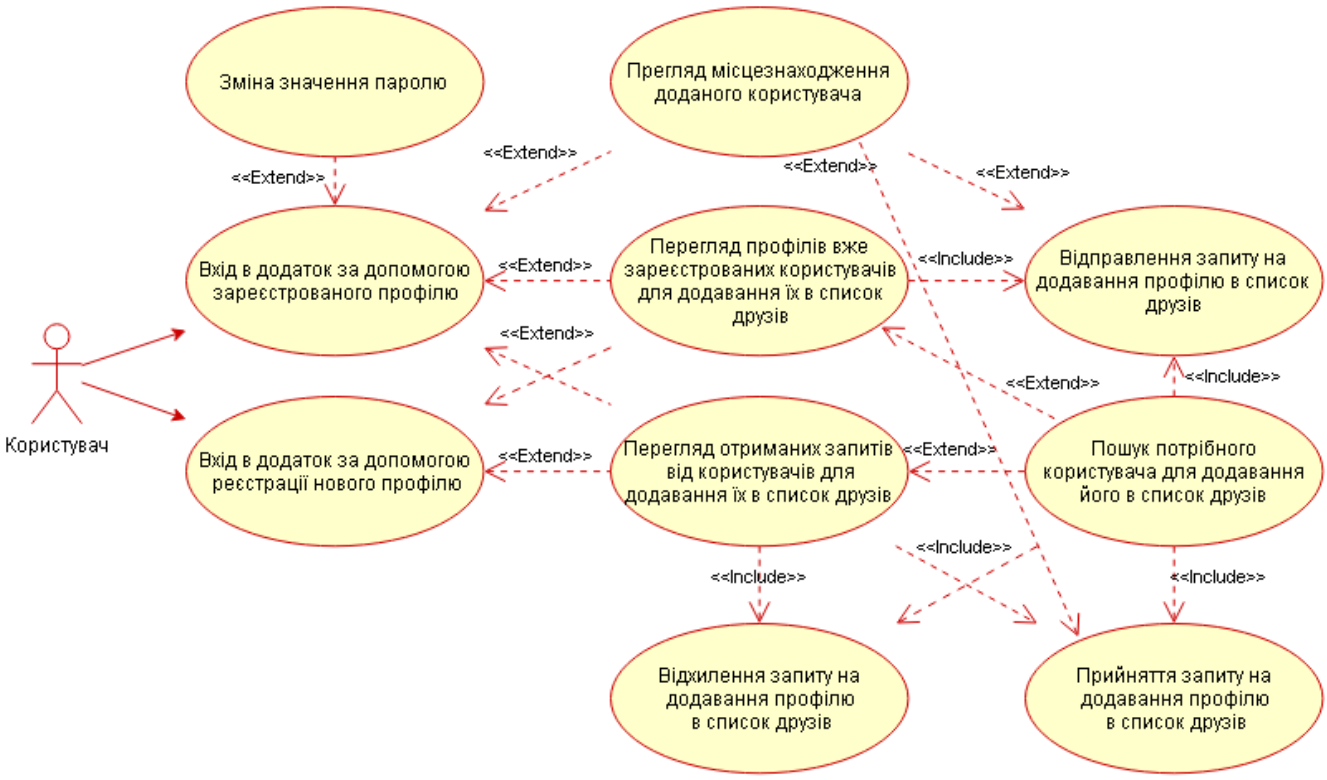

Рис.2.3. Діаграма прецедентів програмного молуля

Діаграма прецедентів починається з того, що користувач повинен ввійти в додаток використовуючи один з двох варіантів використання, а саме здійснити вхід в програмний модуль за допомогою зареєстрованого профілю або за допомогою реєстрації нового аккаунту користувача.

У випадку обрання прецеденту входу в систему за допомогою реєстрації нового профілю користувачу стають доступні такі розширення як: перегляд профілів вже зареєстрованих користувачів для їх додавання в список друзів та перегляд отриманих запитів від користувачів. Варто відмітити, що варіант використання перегляду місцезнаходження доданого профілю відсутній, адже після реєстрації користувач не має жодного аккаунту для *GPS*-моніторингу. Обидва доступні на даний момент варіанти використання поєднані відношеннями «розширення», бо є абсолютно опціональними, тобто користувач може просто їх не використовувати, але вони доповнюють базовий варіант використання.

У випадку, якщо користувач вирішив скористатись варіантом використання, що пропонує переглянути надіслані йому запити на додавання в список друзів, йому стають доступні прецеденти відхилення та прийняття запиту. Дані варіанти

мають тип відношення «включення», що говорить про те, що вони викликаються обовязкого, коли виконується вихідний сценарій використання. Також користувач має можливість виконати пошук потрібного йому профілю для подальшої роботи з ним.

Якщо ж обрано прецедент для перегляду профілів користувачів, що є зареєстрованими в програмному модулі для використання стає доступним варіант з відправленням запиту на додавання аккаунту в список друзів. Також, як і в варіанті з переглядом отриманих запитів користувач може провести спеціальний пошук на предмет наявності профілю, що його цікавить.

Після того, як відправлення запиту відбулося, а користувач, що його отримав, зміг дати згоду на використання даних про своє місцезнаходження стає можливим застосування варіанту з *GPS*-моніторингом.

У випадку обрання прецеденту входу в систему за допомогою зареєстрованого профілю, користувач має можливість скористатися усіма описаними вище варіантами використання. Також є доступним варіант зі зміною значення паролю у разі виникнення такої необхідності. Якщо в попередніх сесіях роботи з програмним модулем користувач зміг додати профіль до списку друзів, він має можливість використання прецеденту перегляду місцезнаходження конкретного доданого аккаунту.

2.3. Інформаційне забезпечення

Відповідно до вимог програмного забезпечення побудовано функціональну схему (рис. 2.4).

Відповідно до функціональної схеми програмного модуля кожен блок має свої вхідні та вихідні параметри.

1) Блок задання значень персональних даних профілю.

Вхідними даними блоку є функція *getUid(),* за допомогою якої стає можливим оперувати даними користувача, що намагається здійснити вхід до програмного модуля. В ході роботи даного блоку відбувається подія по перевірці

співпадіння вхідних даних з вже існуючими та вирішується до якого блоку надсилатиметься інформація, чи до блоку задання зареєстрованих персональних даних користувача, чи до блоку задання нових персональних даних.

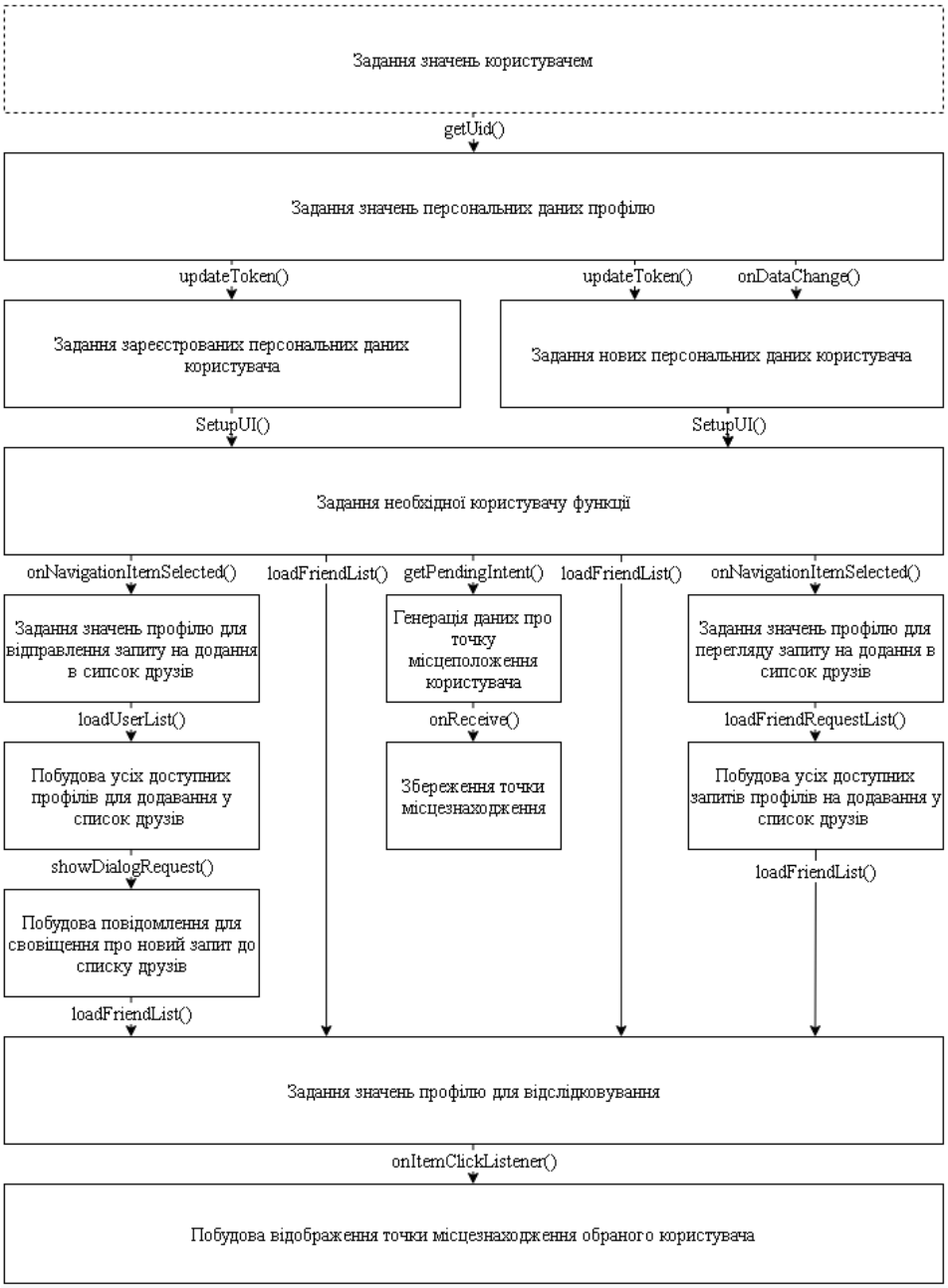

Рис. 2.4. Функціональна схема програмного модуля

2) Блок задання зареєстрованих персональних даних користувача.

Вхідними даними блоку є функція *updateToken()*, яка виконується в разі введення даних раніше зареєстрованого користувача програмного модуля [3]. В ході роботи блоку перевіряється надіслана раніше інформація та у разі збігу

відповідних даних прикріплюється спеціальний унікальний ідентифікатор користувача, що дає йому змогу у майбутньому скористатися сервісами програмного додатку.

3) Блок задання нових персональних даних користувача.

Вхідними даними блоку є функція *onDataChange()* та *updateToken()*, які дозволяють виконати завдання по генерації та збереженню для подальшого оперування вхідної інформації, а саме всередині блоку відбувається процес по генерації нового унікального ідентифікатору для користувача та передачі інформації до бази даних.

Результатом перелічених вище блоків є успішне виконання функції автентифікації, тобто встановлення належності користувачеві інформації в програмному модулі, в залежності від представленого ним ідентифікатору. По закінченню роботи цих блоків відбувається перехід до блоку задання необхідної користувачеві функції.

4) Блок задання необхідної користувачеві функції.

Вхідними даними представленого блоку є функція *setupUI()*, яка надає можливості, що дозволяють всередині блоку отримати доступ до основних можливостей програмного модуля, до яких відносяться блок генерації даних про точку місцезнаходження поточного користувача, блок задання значень профілю для відправлення запиту на додавання до списку друзів, блок задання значень профілю для перегляду запитів на додавання та блок задання значень профілю для відслідковування [4].

5) Блок генерації даних про точку місцезнаходження користувача.

Вхідними даними блоку генерації даних про точку місцезнаходження є функція *getPendingIntent()*, яка надає можливість блоку сформувати інформацію про дані геолокації користувача. Також відбувається перехід до спеціально створеного класу *MyLocationReceiver*, який дозволяє оперувати раніше сформованою інформацією про параметри локації.

6) Блок збереження точки місцезнаходження.

Вхідними даними блоку є функція *onReceive()*, яка робить можливим оперування над даними про локацію користувача, а саме функціональний блок отримує інформацію про сам профіль користувача, таку як код ідентифікатору, та відповідно до неї починає пересилання інформації до бази даних [5].

7) Блок задання значень профілю для відправлення запиту на додання в список друзів та блок задання значень профілю для перегляду запитів на додавання.

Вхідними даними блоків є функція *onNavigationItemSelected()*, яка надає змогу скористатися відповідними можливостями програмного модуля. Блок переглядає результат роботи задання необхідної користувачу функції та в залежності від вихідної інформації перенаправляє роботу додатка до потрібного класу або *FriendRequestActivity*, або *AllPeopleActivity* відповідно. Дані блоки призвані допомогти користувачу у пошуку потрібного профілю, геолокацію якого він зможе переглядати в майбутньому.

8) Блок побудови усіх доступних профілів для додавання їх у список друзів.

Вхідними даними є функція *loadUserList()*, яка дозволить функціональному блоку вести роботу із завантаженими з бази даних списком користувачів, які є можливими для додавання їх у список друзів. Всередині блоку також проводиться перевірка для знаходження власної адреси користувача та виділення її на фоні усіх інших. У разі, якщо один з запропонованих профілів зацікавив користувача він може натиснути на нього і в дію ввійде наступний блок побудови повідомлення для сповіщення про новий запит.

9) Блок побудови повідомлення для сповіщення про новий запит до списку друзів.

Вхідними даними блоку є функція *showDialogRequest()*, яка надасть функціональному блоку усі необхідні дані про користувачів, якім потрібно відправити запит, а також дозволить побудувати сповіщення з двома варіантами дій, а саме відправка повідомлення або закриття вікна. У разі відправлення повідомлення блок створює новий запис в базі даних з інформацією про двох

користувачів, а також викличе функцію *sendFriendRequest(),* яка і відповідає за надходження листа. Якщо ж обрано варіант з закриттям вікна, то хід програми повернеться до попереднього блоку.

10) Побудова усіх доступних запитів профілів на додавання їх у список друзів.

Вхідними даними є функція *loadFriendRequestList()*, яка дозволить функціональному блоку вести роботу з завантаженим списком профілів, запити яких надіслано користувачу. Всередині блоку відбувається завантаження та перегляд усіх запитів з можливістю їх прийняття та відхилення. У разі прийняття запиту в базі даних реєструється відповідний запис про додання профілів до списку друзів, а в разі відхилення запит видаляється з даного блоку.

11) Блок задання значень профілю для відслідковування.

Вхідними даними є функція *loadFriendList(),* яка дозволить функціональному блоку вести роботу із завантаженими з бази даних списком користувачів, які є доданими до списку друзів одне одного. Представлені в даному блоці профілі призначені вже для відслідковування точки їх місцезнаходження. Вміст даного блоку змінюється оперативно в залежності від результатів роботи попередньо представлених функціональних блоків.

12) Блок побудови відображення точки місцезнаходження обраного користувача.

Вхідними даними блоку є функція *onItemClickLestener()*, яка дозволяє взаємодіяти з результатами роботи блоку задання значень профілю для відслідковування [6]. В ході роботи блоку відбувається перехід до спеціального класу *TrackingActivity*, який будує зовнішній вигляд карти світу, а також використовує отримані з бази даних відомості про параметри локації профілю обраного раніше користувача. Також в наявності є оброблювачі подій, що пов’язані з призупиненням або повною зупинкою роботи програмного модулю, а також зміни даних геолокації.

2.4. Висновки до розділу

1) Запропоновано перелік вимог до програмного модуля батьківського контролю з функцією *GPS*-моніторингу:

– можливість роботи програмного модуля з серверною базою даних для оперування інформацією користувачів;

– можливість створення та адміністрування користувачем профілів за допомогою введених даних, таких як електронна пошта, ім’я та фамілія, а також пароль;

– можливість перегляду та пошуку необхідних профілів користувачів, що є зареєстрованими в програмному модулі батьківського контролю з функцією *GPS*-моніторингу;

– можливість відправлення запиту на додавання профілю користувача у відповідний список друзів для подальшої роботи з ним;

– можливість отримання сповіщень на телефон користувача про отримані від інших профілів запити, що його аккаунт бажають додати у відповідний список друзів;

– можливість перегляду та пошуку необхідного отриманого запиту від іншого користувача програмного модуля на додавання у список друзів;

– можливість відхилення або прийняття запитів від іншого користувача на додавання профілю у відповідний список для подальшої роботи з ним;

– можливість відслідковувати точку місцезнаходження на карті світу доданого профілю користувача;

– можливість змінювати користувачем програмного модуля отримане зображення карти на якій знаходиться точка місцезнаходження відповідного профілю (масштабування);

– можливість переглядати в онлайн режимі зміни місцезнаходження доданого профілю користувача;

– можливість переглядати дані про момент часу в який профіль користувача, що відслідковується, знаходиться в конкретній точці геолокації;

– можливість взаємодіяти зі стандартними засобами операційної системи *Android* пов’язаними з картами, такими як прокладання маршруту від заданої користувачем адреси до точки геолокації профілю, що відслідковується, а також завантаження сторінки з відомостями та фотографіями про те, що знаходиться біля місця перебування профілю, та його координати.

2) Розроблено структуру програмного модуля, яку умовно можна поділити на дві частини, а саме структурну схему серверного блоку та структурну схему клієнтського блоку.

До структурної схеми серверного блоку входять наступні компоненти:

– база даних, елементами якої є сукупність особистих даних користувача, якими вона оперує;

– блок роботи з базою даних, за допомогою якого встановлюється зв'язок між серверною частиною програмного модуля та самою базою даних;

– блок оновлення індивідуальних ідентифікаторів користувачів;

– блок для збереження та відправлення даних про координати місцезнаходження користувача;

– блок зберігання особистих даних користувача, до яких належать відомості про електронні адреси, паролі та користувацькі ідентифікатори;

– блок оперування профілями, які є доступними користувачу в блоках клієнтської частини програмного модуля.

До структурної схеми клієнтського блоку входять наступні компоненти:

– база даних, елементами якої є сукупність особистих даних користувача, якими вона оперує;

– блок роботи з базою даних, за допомогою якого встановлюється зв'язок між клієнтською частиною програмного модуля та самою базою даних;

– інтерфейс користувача, який виконує функцію передачі інформації між користувачем та блоком роботи з базою даних;

– блок роботи з профілями користувачів, що надає можливість програмному модулю працювати та змінювати в ході роботи вже створені профілі;

– блок відображення усіх можливих профілів місцезнаходження, яких користувач може відслідкувати;

– блок відображення та пошуку усіх профілів, яким користувач може надіслати запит на додавання їх до списку друзів;

– блок відправлення запиту на додавання конкретного профілю до списку друзів;

– блок відображення та пошуку усіх запитів на додавання до списку друзів отриманих від інших користувачів;

– блок відхилення запитів;

– блок прийняття запитів;

– блок *GPS*-моніторингу профілю, що дозволяє користувачу отримувати та оперувати даними точки місцезнаходження доданого аккаунту.

3) Побудовано функціональну схему програмного модуля, до якої належать такі компоненти:

– блок задання значень персональних даних профілю;

– блок задання зареєстрованих персональних даних користувача;

– блок задання нових персональних даних користувача;

– блок задання необхідної користувачеві функції;

– блок генерації даних про точку місцезнаходження користувача;

– блок збереження точки місцезнаходження;

– блок задання значень профілю для відправлення запиту на додання в список друзів;

– блок задання значень профілю для перегляду запитів на додавання;

– блок побудови усіх доступних профілів для додавання їх у список друзів;

– блок побудови повідомлення для сповіщення про новий запит до списку друзів;

– побудова усіх доступних запитів профілів на додавання їх у список друзів;

– блок задання значень профілю для відслідковування;

– блок побудови відображення точки місцезнаходження обраного користувача.

РОЗДІЛ 3
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ МОДУЛЯ БАТЬКІВСЬКОГО КОНТРОЛЮ З ФУНКЦІЄЮ GPS-МОНІТОРИНГУ

3.1. Системні вимоги

Програмний модуль батьківського контролю з функцією *GPS*-моніторингу представляється у вигляді файлу з розширенням *APK*. Для роботи програми необхідний телефон з такими мінімальними системними вимогами:

– операційна система *Android Lollipop* 5.0;

– 512 *МБ* оперативної пам'яті;

– процесор з двома ядрами та тактовою частотою 1200 *МГц*;

– можливість підключення до мережі;

– можливість підключення до *GPS*-локатору.

3.2. Алгоритми реалізації методів роботи модуля

Відповідно до поставлених завдань проектування програмного модуля для реалізації обрано можливість додавання до профілю користувача аккаунтів для їх подальшого *GPS* відслідковування, а також перегляд змін в онлайн режимі точки місцезнаходження користувачів.

1) Додавання до профілю користувача аккаунтів людей для їх подальшого *GPS* відслідковування.

У разі створення нового профілю користувача доступ до можливості відслідковування точки місцезнаходження інших людей відсутній, адже новостворений профіль не має жодного контакту в списку друзів.

Для того щоб отримати доступ до функції *GPS*-моніторингу достатньо додати один профіль до списку, для подальшої роботи з ним. Зробити це можна, скориставшись одним з двох варіантів:

– відправити запит іншому користувачу на додавання профілю в список друзів;

– прийняти запит на додавання профілю від іншого користувача в список друзів.

Для відображення реалізації методів відправлення запиту іншим користувачам на додавання профілю в список друзів розроблено відповідну схему алгоритму (рис. 3.1).

На самому початку роботи алгоритму методів відправлення запиту на додавання в список друзів, виконується функція *onCreate()*, яка є стандартною для створення різноманітних вікон при розробці мобільних додатків. Саме цей метод задає початкові значення параметрів при ініціалізації нових вікон [7]. На самому початку функції визначається використання елементу *toolbar* за допомогою якого стає можливим в майбутньому використати бокове меню та переходити до відповідних вікон програмного модуля [8]. Робиться це за допомогою класу *Toolbar* екземпляр якого відповідає за усі зміни панелі інструментів, що проводяться в програмі. Також потрібно вказати, що звичний для вікна *ActionBar* заміниться панеллю інструментів. Це робиться за допомогою функції *setSupportActionBar().*

```
Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
```

Для переходу до потрібних вікон програмного модуля, які містять необхідні користувачу функції, використовується спеціальна можливість програм написаних для мобільних додатків *NavigationView.* Вона є певною навігаційною секцією створеного додатку, яка виводить в потрібний момент на екран усі навігаційні опції програмного модуля, що більшість свого часу є прихованими і відкриваються після натиснення на спеціальний іконку, що знаходиться на створеній раніше панелі інструментів [9].

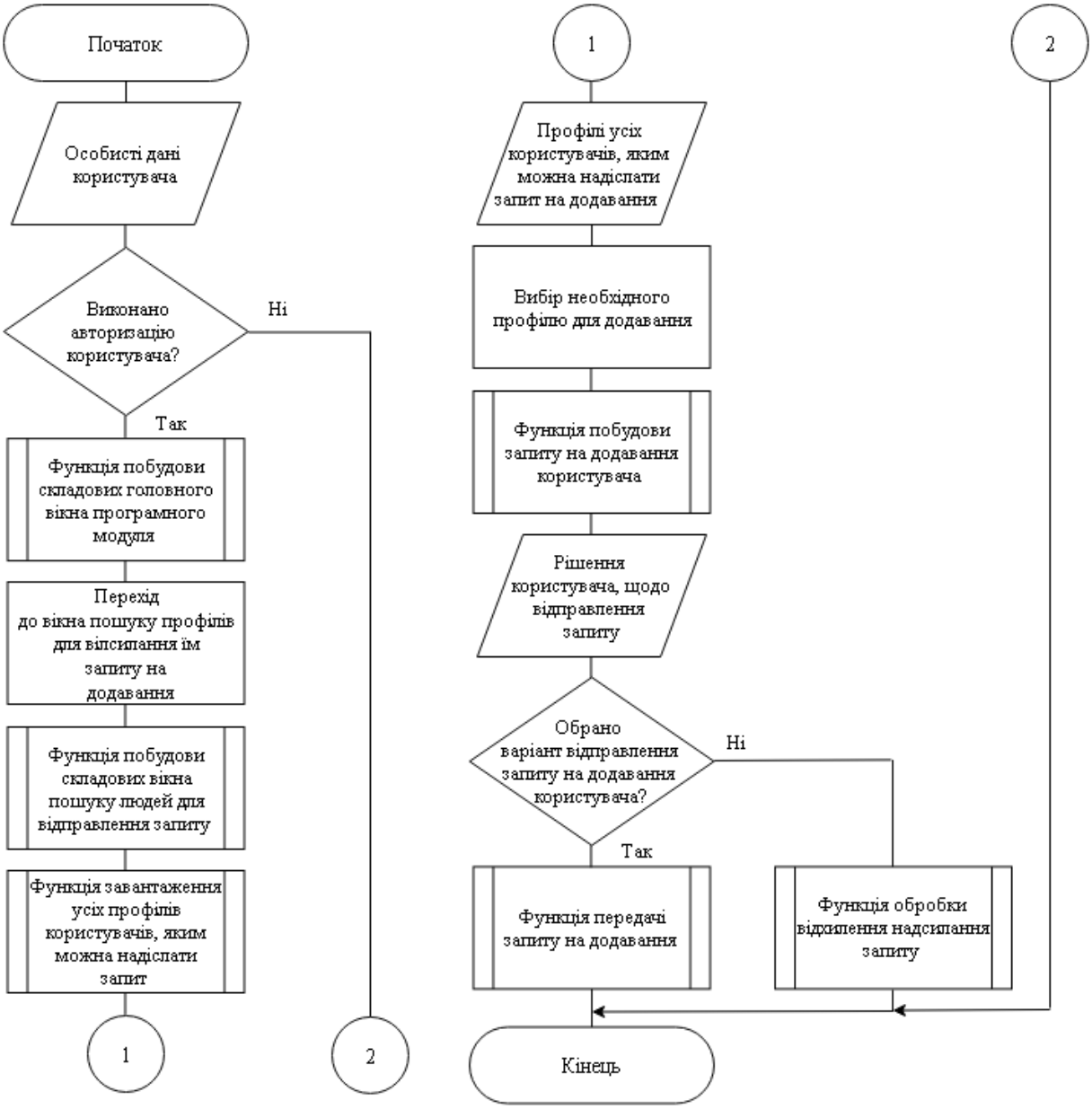

Рис. 3.1. Схема алгоритму відправлення запиту на додавання в список друзів

За вміст виведеного бокового меню відповідає екземпляр класу *DrawerLayout*, ініціалізація якого представлена нижче:

```
DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
drawer.addDrawerListener(toggle);
 toggle.syncState();
```

Для відображення привітання зареєстрованого користувача використовується змінна класу *View*, яка отримує доступ до електронної почти користувача за допомогою функції *findViewById()*. Вивід необхідного тексту виконується з використанням методу *setText()*:

```
View headerView = navigationView.getHeaderView(0);
TextView txt_user_logged = (TextView)
headerView.findViewById(R.id.txt_logged_email);
txt_user_logged.setText("Welcome back, " + Common.loggedUser.getEmail());
```

Також задана функція, яка оброблює подію закриття бокового меню за допомогою натиснення на кнопку «Назад» телефону. Це функція *onBackPressed()*, яка перевіряє чи є відкритим меню *isDrawerOpen()* та при натисненні на необхідну кнопку його закриває за допомогою методу *closeDrawer()*.

```
public void onBackPressed()
{DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
if(drawer.isDrawerOpen(GravityCompat.START))
{drawer.closeDrawer(GravityCompat.START); }
else{super.onBackPressed();}}
```

Для переходу на один з можливих варіантів вікон, що пропонує бокове меню використовується метод *onNavigationItemSelected()*, який за допомогою змінної-параметру отримає ідентифікатор обраного варіанту меню [10]. Даний ідентифікатор перевіряється на співпадіння одного з представлених значень умовного оператору *if*. Якщо ідентифікатор співпав зі значенням оператору, то відбувається перехід до відповідного вікна програмного модуля за допомогою використання функції *startActivity*, яка ініціалізує перехід до необхідного вікна.

```
int id = item.getItemId(); if (id == R.id.nav_find_people) {
startActivity(new Intent(HomeActivity.this,AllPeopleActivity.class));}
else if (id == R.id.nav_add_people) {
startActivity(new Intent(HomeActivity.this,FriendRequestActivity.class));
} else if (id == R.id.nav_sign_out) {
 System.exit(1);}
```

В свою чергу бокове меню знову стає скритим використовуючи метод *drawer.closeDrawer().*

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);

drawer.closeDrawer(GravityCompat.START); return true;

Для опису роботи алгоритму методів відправлення запиту на додавання в список друзів користувачу необхідно натиснути на варіант бокового меню «*Find People*». Представлене вікно зберігає усі можливі профілі користувачів, яким є можливість відправити запит. Функція створення даного вікна ініціалізує такі компоненти як: пошуковий рядок (*MaterialSearchBar*), список користувачів (*RecyclerView*) та сервіс для здійснення відправки повідомлень користувачам, використовуючи їх унікальні ідентифікатори (*FCMService*).

Рядок для пошуку оголошується за допомогою класу *MaterialSearchBar*, екземпляром якого є змінна *searchBar* [11]. Під час роботи програми при натисненні на рядок виведуться елементи, які відповідають за електронні адреси профілів користувачів. Їх кількість можна задати за допомогою функціх *setCardViewElevation().*

searchBar = (MaterialSearchBar) findViewById(R.id.material_search_bar);

searchBar.setCardViewElevation(10);

Також в програмні додано інструменти відслідковування зміни тексту в рядку для пошуку, які сформовано за допомогою спеціального інструменту мови *Java*, а саме слухачів змін. Слухачі змін – це спеціальні об’єкти, які отримують відомості про зміни під час виконання подій. Вони знаходяться в постійному очікування, поки не наступить відповідна подія. В даному випадку це реалізовано за допомогою методу *addTextChangeListener(new TextWatcher())*. В якості відслідковування змін функція пропонує встановити клас інтерфейсу *TextWatcher*, який містить в собі пусті методи, які необхідно заповнити програмісту. Методів в класі три: *beforeTextChanged(), onTextChanged()* та *afterTextChanged().* При зміні тексту вони викликаються автоматично. В програмі змінено метод *onTextChanged(CharSequence s, int start, int before, int count)*, який викликається

для повідомлення того, що в рядку *s*, починаючи з позиції *start* замінені символи новими *count* символами.

```
searchBar.addTextChangeListener(new TextWatcher() {
@Override
public void beforeTextChanged(CharSequence s, int start, int count, int after) {}
@Override
public void onTextChanged(CharSequence s, int start, int before, int count) {
List<String> suggest = new ArrayList<>(); for(String search:suggestList){
if(search.toLowerCase().contains(searchBar.getText().toLowerCase()))
suggest.add(search);} searchBar.setLastSuggestions(suggest); }
@Override
public void afterTextChanged(Editable s) {}});
```

У випадку, якщо користувач вирішив змінити рядок пошуку профілів почне працювати функція *setOnSearchActionListener(new MaterialSearchBar.OnSearchActionListener())*, яка відслідковує сигнал зміни стану рядку за допомогою вкладених методів *onSearchStateChanged(boolean enabled)* та *onSearchConfirmed(CharSequence text)*. В *onSearchStateChanged(boolean enabled)* перевіряється можливість доступу до рядку, і у разі його закриття відбудеться відновлення вікна зі списком усіх користувачів.

```
searchBar.setOnSearchActionListener(new
MaterialSearchBar.OnSearchActionListener() {
@Override
public void onSearchStateChanged(boolean enabled) {
if(!enabled) {if(adapter != null) {recycler_all_user.setAdapter(adapter);}}}
@Override
public void onSearchConfirmed(CharSequence text) {
startSearch(text.toString());}});
```

Метод *onSearchConfirmed()* надасть доступ до функції *startSearch(text.toString())*, яка окрім того, що надає можливість провести пошук необхідної адреси, також виведе усі можливі профілі, які користувач може

додати. Робиться це за допомогою ініціалізації запиту *query* введеного в пошук користувача:

Query query = FirebaseDatabase.getInstance()

.getReference(Common.USER_INFORMATION).orderByChild("name")

.startAt(text_search);

Де *text_search* – це введене в пошуковий рядок значення. Якщо потрібно прив'язати список користувачів до рядка для пошуку потрібно ввести спеціальний адаптер *Adapter*. У випадку програми використовується *FirebaseRecyclerAdapter*, який прив'язує запит *query* до рядка пошуку і відображає усі зміни в реальному часі, включаючи елементи, які додаються, видаляються або змінюються.

FirebaseRecyclerOptions<User> options = new FirebaseRecyclerOptions.Builder<User>().setQuery(query,User.class).build();

Далі потрібно створити об'єкт *FirebaseRecyclerAdapter*. На цей момент в програмі вже є підклас *ViewHolder* для відображення кожного елемента.

searchAdapter = new FirebaseRecyclerAdapter<User, UserViewHolder>(options)

Фунція *onCreateViewHolder()*, яка призначення для виведення усіх користувачів в рядку для пошуку:

public UserViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) { View itemView = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.layout_user,viewGroup, false); return new UserViewHolder(itemView);}

За виведення списку користувачів відповідає інструмент *RecyclerView.* Це певний компонент користувацького інтерфейсу, який дозволяє створювати списки, що можна прокрутити. В створеній змінній класу *RecyclerView* потрібно встановити *LayoutManager*, який відповідає за позиціювання компонентів *view*. Таким чином *LayoutManager* відображає створений список у відповідній формі. В даному випадку список відображається горизонтально, бо використана *LinearLayoutManager*.

recycler_all_user = (RecyclerView)findViewById(R.id.recycler_all_people);

RecyclerView.LayoutManager layoutManager = new LinearLayoutManager(this); recycler_all_user.setLayoutManager(layoutManager);

Після ініціалізації компонента *RecyclerView* дія переходить до виконання функції *loadUserList(). Як* і в методі *startSearch(),* який використовувався для відображення списку профілів в рядку для пошуку, функція *loadUserList()* також використовує ініціалізацію запиту *query.*

Query query = FirebaseDatabase.getInstance().getReference().child(Common.USER_INFORMATION);

В процесі приєднання списку користувачів до *RecyclerView* створюється спеціальний адаптер *Adapter*. В даному випадку використовується *FirebaseRecyclerAdapter*, який прив'язує запит *query* до *RecyclerView* і відображає усі зміни в реальному часі, включаючи записи профілів, які додаються, видаляються або змінюються.

FirebaseRecyclerOptions<User> options = new FirebaseRecyclerOptions.Builder<User>().setQuery(query,User.class).build();

Далі створюється об'єкт *FirebaseRecyclerAdapter*. На цей момент в програмі вже є створений підклас *ViewHolder* для відображення кожного елемента списку.

adapter = new FirebaseRecyclerAdapter<User, UserViewHolder>(options)

Дані, що виводяться в списку є профілями інших користувачів. Також в функції *onBindViewHolder()* присутній умовний оператор, який перевіряє чи є отримана електронна адреса, адресою самого користувача, що ввійшов в програмний модуль. В такому випадку програма виділяє даний профіль курсивом та дописує до цього рядка слово «*me*» для позначення власного аккаунту.

protected void onBindViewHolder(@NonNull UserViewHolder holder, int position, @NonNull User model) {

if(model.getEmail().equals(Common.loggedUser.getEmail())){

holder.txt_user_email.setText(new StringBuilder(model.getEmail()).append(" (me)")); holder.txt_user_email.setTypeface(holder.txt_user_email.getTypeface(), Typeface.ITALIC);}

else{holder.txt_user_email.setText(new StringBuilder(model.getEmail()));}

holder.setiRecyclerItemClickListener(new IRecyclerItemClickListener() {

@Override

public void onItemClickListener(View view, int position) {

showDialogRequest(model);}});}

Також функція містить вкладений метод *onItemClickListener()*, який викликає рядок *showDialogRequest(model)* у разі натиснення на один з запропонованих профілів користувачі.

Для відправлення на інший профіль запиту на додавання в список друзів користувачу потрібно підтвердити відправлення сповіщення. Саме сповіщення будується за допомогою екземпляру класу *AlertDialog.* Змінна отримує запис для заголовку (*setTitle()*), зміст самого сповіщення (*setMessage()*), а також відповідну іконку (*setIcon()*) [12].

AlertDialog.Builder alertDialog = new AlertDialog.Builder(this, R.style.MyRequestDialog); alertDialog.setTitle("Request Friend");

alertDialog.setMessage("Do you want to send request friend to "+model.getEmail()); alertDialog.setIcon(R.drawable.ic_baseline_account_circle_24);

Також створюються дві кнопки для відхилення та погодження надсилання запиту. Кнопки створюються за допомогою функцій *setNegativeButton()* та *setPositiveButton()*, до яких створюються відповідні обробники подій *onClick().*

Код кнопки відхилення надсилання запиту представлений нижче:

alertDialog.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {@Override

public void onClick(DialogInterface dialogInterface, int which) {

dialogInterface.dismiss();}});

При натисненні користувачем на кнопку з надписом «*Cancel*» створений раніше діалог зникне з екрану за допомогою методу *dismiss().*

У разі натиснення на кнопку погодження надсилання запиту створиться екземпляр класу *DatabaseReference*, яка є посиланням на конкретну змінну бази

даних та використовується для читання та запису даних списку друзів користувача.

DatabaseReference acceptList = FirebaseDatabase.getInstance()
.getReference(Common.USER_INFORMATION).child(Common.loggedUser.getUid())
.child(Common.ACCEPT_LIST);

Далі за допомогою змінної *acceptList* створюється обробник подій *onDataChange()*, який призначений для перевірки списку користувача на наявність в ньому конкретного ідентифікатору. Параметром функції *onDataChange()* є змінна класу *DataSnapshot*. Ця змінна зберігає інформацію, яка розташована в базі даних і якою можна оперувати за допомогою функції *getValue()*. Якщо метод *getValue()* повертає значення *null* це означає, що подібна інформація про користувача в базі відсутня. В такому випадку програма переходить до блоку коду умовного оператора *if*, в якому в якому викликається функція *sendFriendRequest()*, що призначення для відправлення запиту конкретному користувачу. У випадку, якщо *getValue()* повертає конкретне значення на екран виведеться повідомлення про те, що даний користувач вже є в списку друзів.

acceptList.orderByKey().equalTo(model.getUid()).addListenerForSingleValueEvent(new ValueEventListener() {
public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
if(dataSnapshot.getValue() == null){ sendFriendRequest(model);}
else
Toast.makeText(AllPeopleActivity.this, "You and "+model.getEmail()+" already are friend", Toast.LENGTH_SHORT).show();}});

Якщо користувач погодився на відправлення запиту для додавання іншого профілю в список друзів, дія програми переходить до функції *sendFriendRequest()*. На початку функції створюється ідентифікатор за допомогою методу *getInstance()*, який посилається на конкретну змінну бази даних. Потім за допомогою створеного екземпляру виконується обробник подій *onDataChange()* зі змінною-

параметром *dataSanpchot*. Спрацьовує умовний оператор *if*, який перевіряє створений ідентифікатор і в разі помилки викликає відповідне повідомлення.

if(dataSanpchot.getValue() == null){ Toast.makeText(AllPeopleActivity.this, "Token error", Toast.LENGTH_SHORT).show();}

У випадку ініціалізації коректного ідентифікатору створюється змінна класу *Request*, яка використовується для збереження хешованих даних запиту.

Request request = new Request();

Формується новий екземпляр класу *Map* з конструктором *HashMap()*, який зберігає задані значення за допомогою хеш-таблиці. Цей клас реалізує інтерфейс *Map* для зберігання даних у вигляді пар ключ/значення. В даному випадку і ключ, і значення мають тип *String*.

Map<String, String> dataSend = new HashMap<>();

Додавання елементів відбувається за допомогою методу *put(key, value)*, в аргументах якого задається ключ та значення. В програмному модулі даний метод використовується для того, щоб вказати ідентифікатор та електронну адресу відправника та отримувача запиту.

dataSend.put(Common.FROM_UID, Common.loggedUser.getUid());

dataSend.put(Common.FROM_NAME, Common.loggedUser.getEmail());

dataSend.put(Common.TO_UID, model.getUid());

dataSend.put(Common.TO_NAME, model.getEmail());

Далі визначену змінну *dataSend* надаємо в сеттер екземпляру *request* за допомогою функції *setData()*.

На самому початку роботи вікна пошуку профілів для відправлення запиту оголошено змінну класу *IFCMService*, яка використовується для здійснення відправки повідомлення користувачем, використовуючи спеціальні ідентифікатори. У випадку програми за відправку сповіщення відповідає функція *ifcmService.sendFriendRequestToUser(request)*, аргументом якої є визначена раніше змінна *request*. Для відправки сповіщення та його збереження використовується методи класу *compositeDisposable()*. Після відправлення запиту на вікно

виведеться повідомлення про успішно виконану дію за допомогою методу *Toast.makeText().*

compositeDisposable.add(ifcmService.sendFriendRequestToUser(request).
subscribeOn(Schedulers.io()).observeOn(AndroidSchedulers.mainThread()).subscribe(
new Consumer<MyResponse>() {

public void accept(MyResponse myResponse) throws Exception {

if(myResponse.success == 1)Toast.makeText(AllPeopleActivity.this, "Request sent!", Toast.LENGTH_SHORT).show();}}));

Результатом виконання алгоритму є перегляд усіх можливих профілів та відправлення у разі необхідності запиту на додавання до списку друзів.

Для відображення реалізації методів прийняття запиту на додавання профілю від іншого користувача в список друзів розроблено відповідну схему алгоритму (рис. 3.2).

Для опису роботи алгоритму користувачу необхідно натиснути на варіант бокового меню «*Request*». Вибір вікна в боковому меню опрацює функція *onNavigationItemSelected()* і розпочне роботу відповідного екрану за допомогою методу *startActivity().* Представлене вікно зберігає усі запити користувачів, які відправлені до даного профілю. В свою чергу функція створення вікна *onCreate()* ініціалізує компоненти пошукового рядка (*MaterialSearchBar*) та списку (*RecyclerView*). Дані компоненти ініціалізуються так само як і у вікні перегляду усіх користувачів для відправлення запиту, але їх функції роботи дещо змінені.

Таким чином у методі *startSearch()*, який призначений для виведення в пошуковому рядку усіх профілів користувачів, що надали запит, змінено ініціалізацію *query*, яка отримує посилання на сам запит:

Query query = FirebaseDatabase.getInstance().getReference().

child(Common.USER_INFORMATION).

child(Common.loggedUser.getUid()).

child(Common.FRIEND_REQUEST).

orderByChild("name").

startAt(search_value);

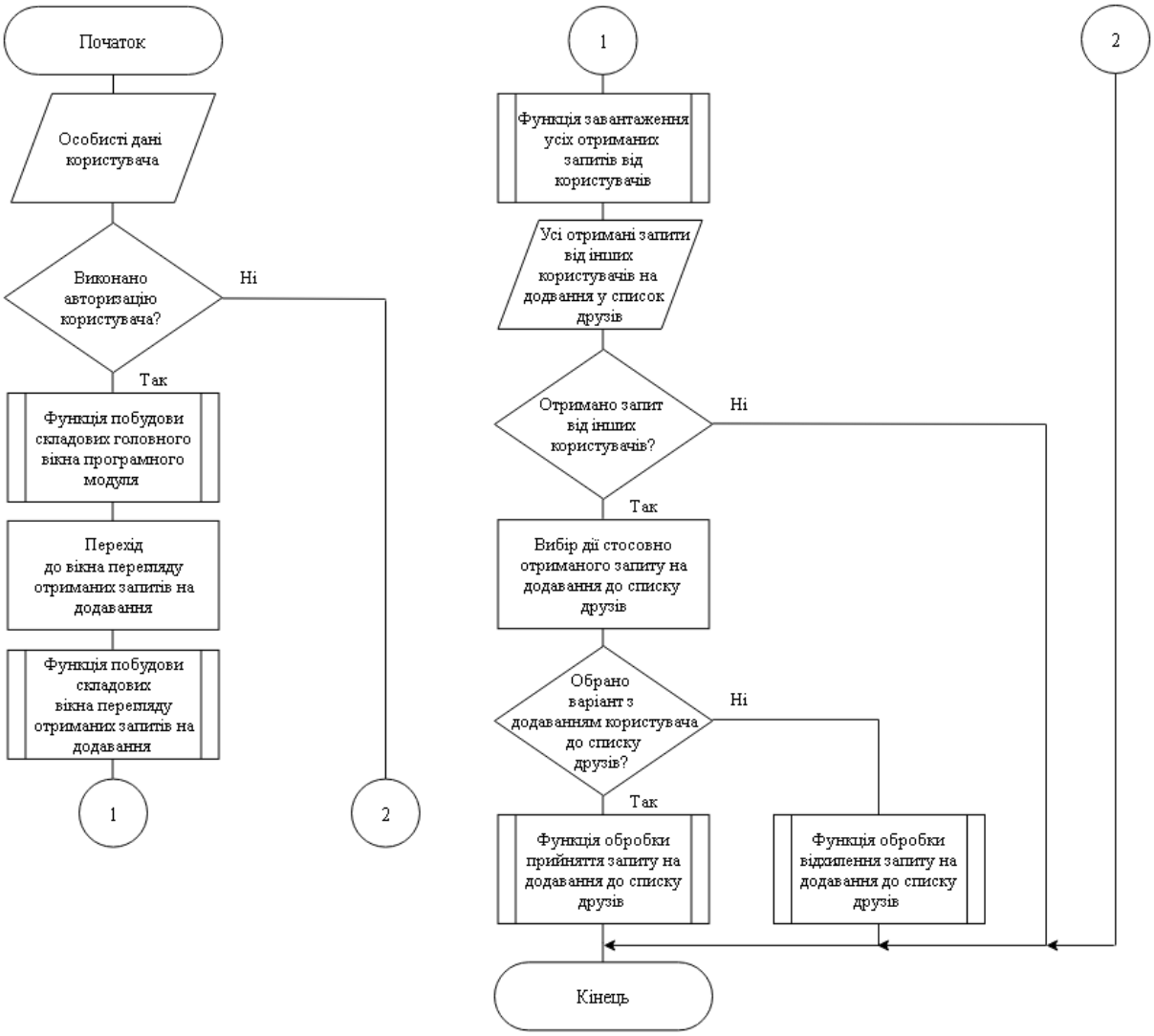

Рис. 3.2. Схема алгоритму прийняття запиту на додавання в список друзів

Більших змін зазнав метод завантаження *RecyclerView*. За вивід списку отриманих запитів відповідає функція *loadFriendRequestList()*. Як і в попередній реалізації подібного методу ініціалізується запит *query* та створюється спеціальний адаптер, який поєднує запит та *RecyclerView*, що дозволить відображати усі зміни вікна в реальному часі. Після чого створюється об'єкт *FirebaseRecyclerAdapter*. Далі в методі використовується функція *onBindViewHolder()*, яка призначена для завантаження до списку відповідних значень запитів. Окрім самого запису також завантажуються кнопки для його прийняття та відхилення, до яких створюються відповідні обробники подій.

Не зважаючи на яку кнопку натисне користувач задіється функція *deleteFriendRequest()*, яка повинна видалити запит з вікна, з єдиною різницею, що

у випадку відхилення запису виведеться повідомлення «*Remove!*». У випадку прийняття запиту спрацює метод *addUserToFriendContact()*, в якому створиться посилання на відповідний запис в базі даних та зміниться його значення, шляхом додавання туди відповідного користувача.

private void addUserToFriendContact(User model) {

DatabaseReference acceptList = FirebaseDatabase.getInstance().getReference(Common.USER_INFORMATION).child(model.getUid()).child(Common.ACCEPT_LIST);

acceptList.child(model.getUid()).setValue(Common.loggedUser); }

Результатом виконання алгоритму є перегляд усіх отриманих запитів та прийняття рішень щодо їх прийняття або відхилення.

2) Оновлення та перегляд змін в онлайн режимі точки місцезнаходження користувачів.

У випадку, якщо користувач додав до списку друзів необхідний профіль, він має доступ до функції *GPS*-моніторингу його точки місцеположення. Увесь список доданих профілів знаходиться в головному вікні програми. Для відображення реалізації методів перегляду точки місцезнаходження користувачів розроблено відповідну схему алгоритму (рис. 3.3).

Побудоване головне вікно програмного модуля складається з вже описаних компонент до яких відносяться список та пошуковий рядок. Обидва компонента ініціалізуються в функції *onCreate()* головного вікна програми і не мають особливих змін порівняно з реалізацією подібних компонент в інших вікнах. Варто лише відмітити, що подібні інструменти отримують та завантажують дані профілів з бази даних, які є доданими в список друзів користувача.

В ході роботи програми, при натисненні на профіль необхідного користувача, спрацює обробник подій *onItemClickListener()*, який призваний викликати відповідне вікно з картою світу для відслідковування місцезнаходження.

public void onItemClickListener(View view, int position) { Common.trackingUser = model; startActivity(new Intent(HomeActivity.this,TrackingActivity.class));}

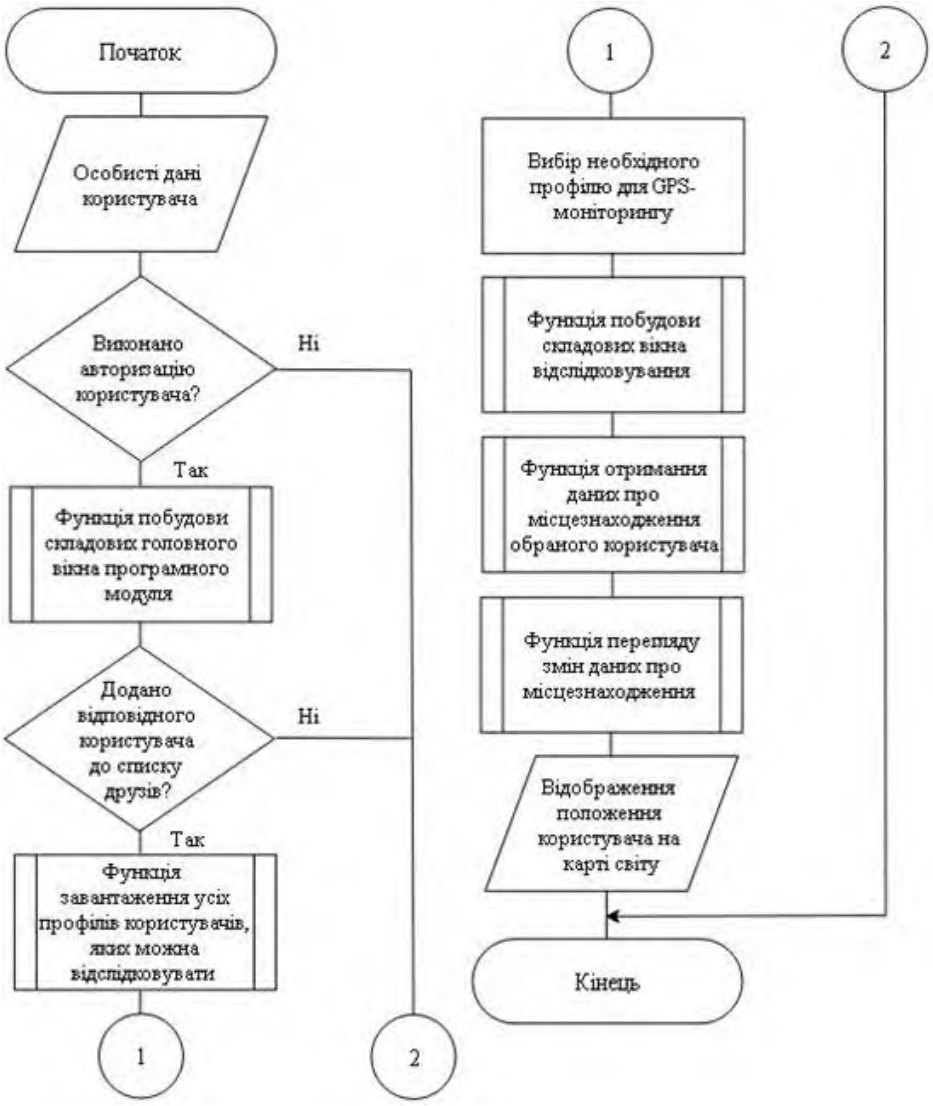

Рис. 3.3. Схема алгоритму та перегляду точки місцезнаходження користувачів

Під час створення вікна для відстеження місцезнаходження оголошується та визначається об’єкт класу *SupportMapFragment*, який призначений для розміщення карти світу в межах екрану. Функція *getSupportFragmentManager(). findFragmentById()* передає ідентифікатор ресурсу розмітки вікна, а метод *getMapAsync()* встановлює зворотній виклик для фрагменту [13].

SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager().

findFragmentById(R.id.map);

mapFragment.getMapAsync(this);

Встановлення зворотного зв’язку необхідне для програми, адже саме він вкаже, коли карта є готовою для відображення. Задамо використання карти світу сервісу *Google* за допомогою функції *onMapReady()*:

public void onMapReady(GoogleMap googleMap) {

mMap = googleMap; mMap.getUiSettings().setZoomControlsEnabled(true);}

Функція *setZoomControlsEnabled()* робить можливою користування інструментами масштабування.

Наприкінці створення вікна викликається функція *registerEventRealtime(),* яка призначена для створення обробника подій отримання та зміни даних локації. Для початку потрібно отримати дані про користувача, чию локацію потрібно відслідкувати. Ця інформація зберігаються в базі даних, тому отримати її можна в спеціальний екземпляр класу *DatabaseReference* за допомогою функції *FirebaseDatabase.getInstance()*, яка робить можливою отримати посилання на відповідні дані, в даному випадку точка локації користувача.

private void registerEventRealtime() {

trackingUserLocation = FirebaseDatabase.getInstance()

.getReference(Common.PUBLIC_LOCATION)

.child(Common.trackingUser.getUid());

trackingUserLocation.addValueEventListener(this);}

Створиться обробник подій *onDataChange()*, який має параметр класу *DataSnapshot*, що зберігає інформацію, яка розташована в базі даних. Отримати ці дані можна за допомогою функції *getValue().* В ході роботи умовного оператору *if*, що знаходиться в *onDataChange(),* створюється змінна *location*, яка зберігає дані про параметри локації, такі як довгота та ширина. Ця змінна згодом стане аргументом при створенні іншого екземпляру класу *LatLng*, що відповідає за місцезнаходження на карті світу маркеру. За допомогою функції *addMarker()* додається червона точка, якій можна задати інформаційне вікно з даними про електронну почту профілю (*getEmail()*) та моментом часу (*getTime()*). Також *animateCamera()* дозволить переміститись в точку знаходження червоного маркеру.

MyLocation location = dataSnapshot.getValue(MyLocation.class);

LatLng userMarker = new LatLng(location.getLatitude(),location.getLongitude());

mMap.addMarker(new MarkerOptions().position(userMarker)

.title(Common.trackingUser.getEmail())

.snippet(Common.getDateFormatted(Common.convertTimeStampToDate(location.getTime())));

mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(userMarker));

Результатом виконання алгоритму є перегляд точки місцезнаходження користувача.

3.3. Висновки до розділу

1) Визначено мінімальні системні вимоги програмного модуля батьківського контролю з функцією *GPS*-моніторингу. До них належать:

– операційна система *Android Lollipop* 5.0;

– 512 *МБ* оперативної пам'яті;

– процесор з двома ядрами та тактовою частотою 1200 *МГц*;

– можливість підключення до мережі;

– можливість підключення до *GPS*-локатору.

2) Програмно реалізовано алгоритми:

– алгоритм додавання до профілю користувача аккаунтів людей для їх подальшого *GPS* відслідковування;

– алгоритм оновлення та перегляд змін в онлайн режимі точки місцезнаходження користувачів.

РОЗДІЛ 4
ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1. Інструкція користувача програмного модуля

Програмний модуль батьківського контролю з функцією *GPS*-моніторингу представляється у вигляді файлу з розширенням *APK*. Для можливості запуску додатку смартфон користувач повинен мати операційну систему *Android* версії не нижче 5.0. Для запуску додатку потрібно натиснути на іконку відповідного програмного модуля в меню *Android*. У випадку запуску модуля користувач потрапляє до початкового вікна програми.

На цьому вікні користувач програмного модуля повинен обрати один з запропонованих варіантів входу, який дозволить йому почати роботу з головними функціями додатку. Користувач має можливість або зареєструвати новий профіль, або використати один з вже зареєстрованих.

У випадку, якщо користувач не має жодного профілю для роботи з програмним модулем для нього є доступним варіант реєстрації аккаунту як за допомогою довільного почтового сервісу, так і спеціальна кнопка, яка надасть можливість реєстрації профілю сервісу *Google*.

Для того, щоб зареєструвати аккаунт нового користувача використовуючи довільний почтовий сервіс потрібно обрати верхню кнопку на початковому вікні додатку. Після обрання потрібного варіанту відкриється нове вікно в якому пропонується ввести логін з яким ведеться подальша робота програми. Саме в цей момент програмний модуль переглядає усі зареєстровані на даний момент профілі і в разі невідповідності введених даних відкриється вікно створення нового аккаунту, що й описано далі в керівництві користувача.

Після натиснення на кнопку, яка відповідає за продовження роботи, відкриється вікно в якому є можливим введення інформації про ім’я та фамілію користувача, а також задання паролю за допомогою якого відбуваються усі подальші процеси входження користувача на свій профіль додатку (рис. 4.1).

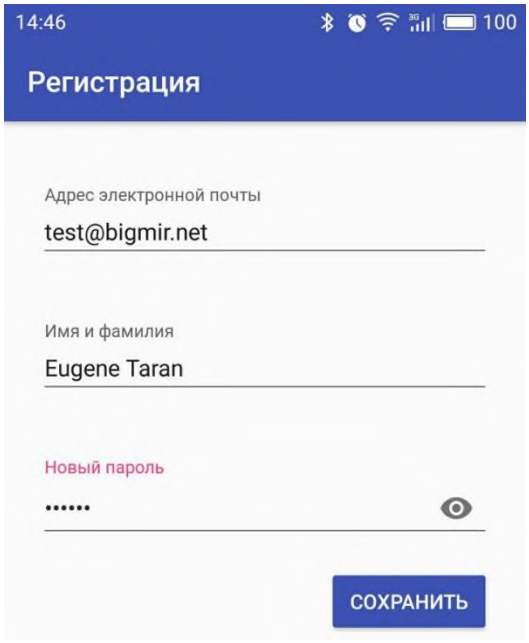

Рис. 4.1. Введення усієї необхідної інформації про профіль користувача

Введений пароль повинен складатися з літер та цифр, а також мати не менше ніж шість символів в рядку. У випадку, якщо користувач під час введення нового паролю не виконав обов’язкові умови програма повідомить про зроблену їм помилку повідомленням та запропонує повторне ведення даних. Також для полегшення процесу введення паролю є можливість перегляду введених даних у вигляді символів, а не точок. Для цього потрібно лише натиснути на піктограму з зображенням ока. Після натиснення на кнопку збереження даних користувача відкриється головне вікно програми, що дозволить користуватися усіма можливостями програмного модулю.

У випадку, якщо користувач має бажання зареєструвати новий профіль використовуючи сервіс *Google* обирається нижня кнопка, що знаходиться на початковому вікні додатку. Після обрання потрібного варіанту відкриється нове вікно в якому процес створення профілю є ідентичним до наведеного вище

варіанту створення аккаунту користувача за допомогою довільних почтових сервісів. Якщо ж користувач вже є зареєстрованим в додатку, то після введення необхідного логіну програма відкриє вікно (рис. 4.2), яке дозволяє користувачу ввести свій пароль та перейти на головне вікно модуля.

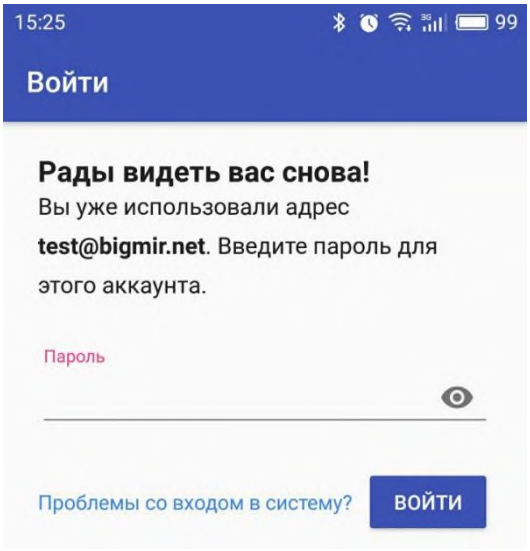

Рис. 4.2. Вікно, що дозволяє користувачу ввести зареєстрований пароль

Якщо користувач має бажання змінити пароль на вже створеному профілі, тоді йому потрібно у вікні введення паролю (див. рис. 4.2), яке з'являється одразу після підтвердження логіну, натиснути на кнопку, яка дасть зрозуміти програмному модулю, що користувач має певні проблеми з процесом входу до самої програми. Користувачу стане доступним вікно, яке запропонує ввести у відповідний рядок адрес почти, на яку відправиться лист з інструкціями по зміні паролю. На почту прийде лист в якому по натисненню на спеціальне посилання стане можливим змінити старий пароль на новий.

Головне вікно програми містить список тих профілів, які є додані користувачем. З цього вікна можна обрати необхідний профіль та розпочати відслідковування точки місцезнаходження на карті доданого профілю.

За допомогою головного вікна програми можна здійснювати переходи до інших функціональних вікон додатку. Робити це можна використовуючи бокове

меню (рис. 4.3), яке відкривається після натиснення на піктограму з зображенням трьох горизонтальних ліній, що знаходиться зліва вгорі екрану.

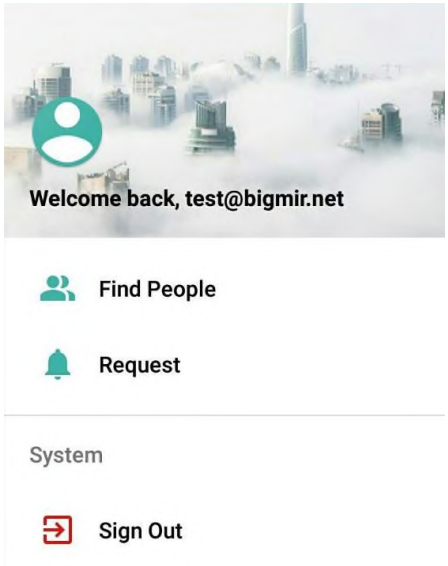

Рис. 4.3. Бокове меню програмного модуля

За допомогою бокового меню є можливим перейти до наступних вікон:

– вікно пошуку профілів користувачів, що є зареєстрованими в програмному модулі;

– вікно перегляду та пошуку необхідного отриманого запиту на додавання профілю користувача;

– вихід користувача з додатку.

Якщо користувач має бажання перейти до вікна пошуку профілів, що є зареєстрованими в програмному модулі йому потрібно натиснути на кнопку з надписом «*Find People*». Дане вікно використовується в разі необхідності відправити конкретному користувачу запит на додавання у список, що згодом можна побачити на головному вікні додатку. На початку роботи з вікном користувача програми зустріне перелік адрес тих профілів, які його можуть зацікавити для відправки запиту.

Після натиснення користувачем на потрібну йому адресу, що відповідає за профіль відповідної людини, на екрані з’явиться повідомлення з текстом запиту, на додавання адреси у список друзів (рис. 4.4).

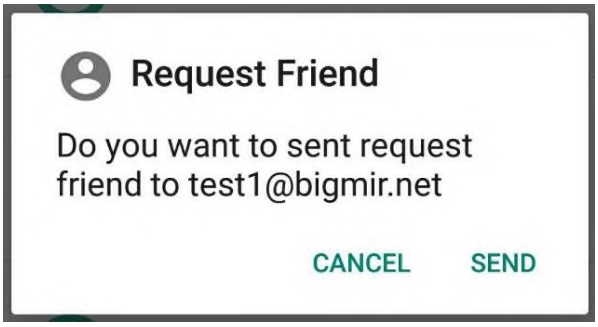

Рис. 4.4. Уточнююче повідомлення для додавання профілю другого користувача

Одразу після натиснення на кнопку відправки запиту, на телефон людини, якій було адресовано запрошення прийде оповіщення, яке призвано сповістити, що інший користувач хоче додати її у список друзів (рис. 4.5). Сповіщення містить в собі адресу користувача, що відправляє запит. Після такого повідомлення користувач може знайти відправлений йому запит у вікні «*Request*».

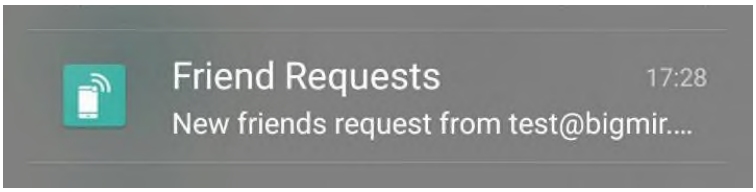

Рис. 4.5. Оповіщення про новий запит на додавання у список друзів

Усі отримані раніше запити зберігаються у спеціальному вікні, яке має назву «*Request*» і отримати доступ до якого можна за допомогою представленого раніше бокового меню (див. рис. 4.3). Після переходу до переглядання вмісту даного вікна користувача зустрінуть запити з адресами тих профілів, які мають бажання додати людину до списку друзів (рис. 4.6). Користувач має можливість взаємодіяти з надісланими йому запитами. Таким чином, отримуючи запит від невідомого йому профілю або не маючи жодного бажання додавати цей профіль у список друзів користувач може відмовити шляхом натиснення на кнопку з зображенням хрестика. У разі, якщо отриманий запит від іншого користувача є прийнятним для людини запит приймається шляхом натиснення на кнопку з зображенням прапорця.

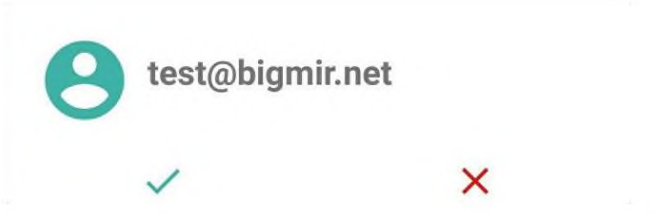

Рис. 4.6. Запит на додавання користувача до списку друзів

Одразу після прийняття запиту додавання профілю у список друзів користувач має можливість споглядати доданий їм аккаунт на головному вікні програмного модуля. З цього вікна вже можна починати вести роботу за переглядом місцезнаходження потрібного користувачеві профілю. Для цього потрібно лише, щоб людина, чиє місцезнаходження нас цікавить, мала ввімкнений *GPS* локатор, при цьому саме програмне забезпечення на її телефоні може бути вимкненим. Якщо користувач натисне на потрібний йому профіль людини почнеться завантаження карти світу і програма, яка отримала дані про місцезнаходження аккаунту, одразу переведе фокус камери на місце перебування людини, яке позначено червоним маркером (рис. 4.7).

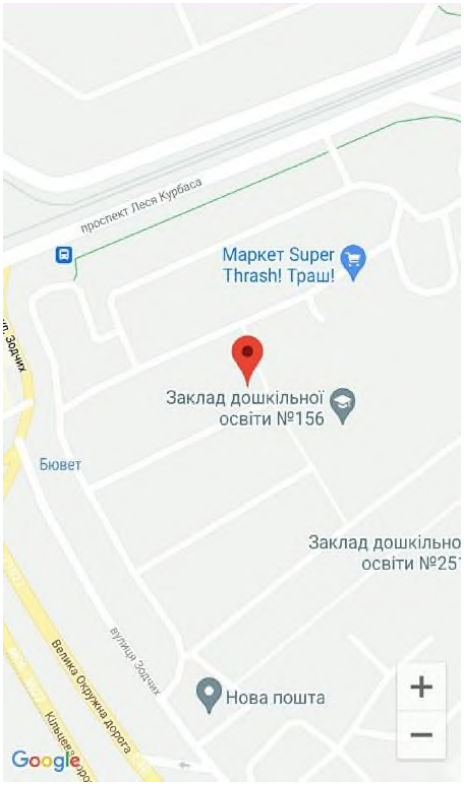

Рис. 4.7. Точка місцеперебування іншого профілю

У разі зміни місцезнаходження профілю людини, чию локацію відслідковує користувач, додаток зреагує на це і в результаті чого додасть на карту світу новий маркер. Таким чином додаток надає можливість переглядати в онлайн режимі зміни геолокації доданого профілю користувача (рис. 4.8).

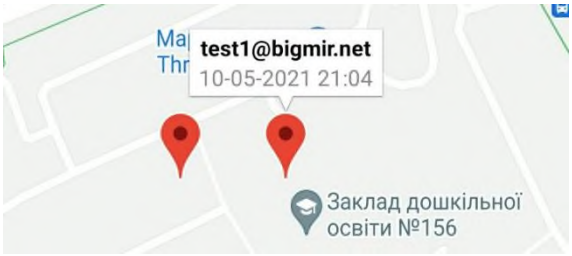

Рис. 4.8. Перегляд в онлайн режимі зміни геолокації доданого профілю користувача та вивід відповідного моменту часу

Натиснувши на виведений таким чином червоний маркер, що знаходиться на карті світу, користувач може дізнатися про дату та момент часу в який власник телефону знаходився у відповідній точці. Надалі користувач має можливість продовжувати роботу з виведеними червоними маркерами місцезнаходження профілів користувачів, шляхом взаємодії з ними сервісу *Google*. Таким чином, якщо натиснути на сам маркер, а потім на іконку з зображенням стрілки, що знаходиться знизу правої частини екрану, то можна прокласти маршрут від заданої користувачем адреси, до адреси знаходження самого маркеру (рис. 4.9).

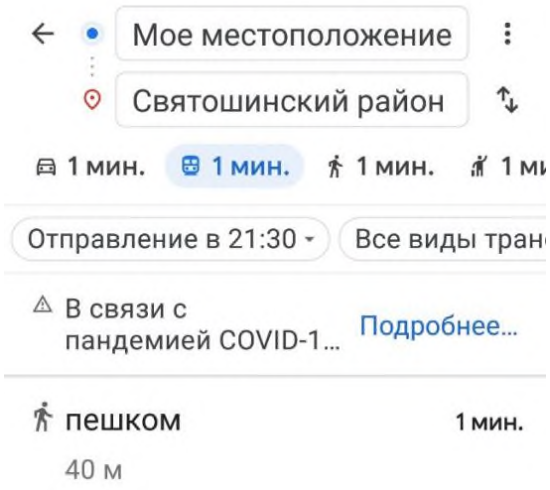

Рис. 4.9. Налаштування побудови маршрутів до місцезнаходження профілю

Якщо натиснути на іконку з зображенням мапи, то телефон завантажить сторінку з відомостями та фотографіями про те, що знаходиться біля адреси маркеру та надасть можливість дізнатися координати червоної точки.

Для виходу користувачеві з програмного модуля батьківського контролю з функцією *GPS*-моніторингу необхідно або натиснути на кнопку «*Sign Out*», що знаходиться в боковому меню головного вікна програми (див. рис. 4.3), або на кнопку смартфона, що відповідає за вихід з різноманітних додатків.

4.2. Тестування роботи програмного модуля

Для тестування програмного модуля батьківського контролю з функцією *GPS*-моніторингу використовується ручне тестування. В ньому застосовуються властивості програмного модуля для знаходження помилок у роботі. Процес та результати тестування представлені у вигляді тестових випадків.

Суть тестового випадку полягає в створенні та проходженні спеціального алгоритму тестування програми, який складається з певних кроків, умов та параметрів, що в свою чергу перевіряють можливість виникнення невизначеної ситуації в роботі функції. Кожен тестовий має містити наступні складові:

– визначення мети конкретного тестового випадку;

– певна умова, що передує виконанню основних дій випадку, яка повинна привести програмний продукт в необхідний стан, для початку перевірки;

– алгоритм дій, яким крок за кроком користується людина, що тестує програмний засіб;

– опис очікуваних наслідків, які повинні відбутися в результаті перевірки певної зазначеної функції додатку;

– статус випадку, який підтверджує або заперечує відповідність очікуваного результату до фактичного.

Тестування роботи програмного модуля проводиться над його певними можливостями, а саме: можливість відправлення запиту на додавання

користувачем профілю іншій людині, можливість відстеження місцезнаходження користувачем доданого їм в список друзів профілю.

4.2.1. Тестування можливості відправлення запиту на додавання користувачем профілю іншій людині в список друзів

Для тестування на помилки можливості відправлення запиту на додавання користувачем профілю іншій людині в список друзів використовується дві почтові адреси та паролі профілів, які є вже зарезервованими в програмі. В ході тесту один користувач шляхом використання спеціальних вікон програми повинен відправити запит на додавання в список друзів іншому, таким чином розглядається здатність програмного модуля відправляти запити на різні профілі для їх подальшого підтвердження або відхилення (табл. 4.1).

Таблиця 4.1

Тестовий випадок відправлення користувачем запиту на додавання профілю іншої людини в список друзів

Дія	Очікуваний результат	Фактичний результат
Передумова		
Відкрити програмний модуль батьківського контролю з функцією *GPS*-моніторингу та провести операцію входу в профіль	Відбувається перехід до головного вікна програми	Пройдено
Кроки тесту		
Відкрити бокове меню програми натиснувши на відповідну піктограму	Кнопка натискається та відкривається бокове меню програми	Пройдено

Закінчення таблиці 4.1

В боковому меню програми натиснути на кнопку «*Find People*»	Кнопка натискається та відкривається вікно, яке відповідає за пошук профілів для їх додавання в список	Пройдено
Натиснути на комірку з адресою потрібного профілю в списку можливих друзів	Комірка натискається та виводиться уточнююче повідомлення, яке запитує, чи дійсно користувач хоче відправити запит на додавання	Пройдено
Натиснути на кнопку «*Send*»	Кнопка натискається та виводиться повідомлення, що запит на додавання до списку друзів надіслано	Пройдено
Постумова		
Завершити роботу програмного модуля	Робота програмного модуля завершена	Пройдено

Висновком даного тестового випадку (див. табл. 4.1) є те, що можливість відправлення запиту на додавання користувачем профілю іншій людині в список друзів працює, таким чином це робить можливим подальше поєднання двох аккаунтів для роботи з *GPS*-трекером.

4.2.2. Тестування можливості відстеження місцезнаходження користувачем доданого їм в список друзів профілю

Для тестування на помилки можливості відстеження місцезнаходження користувачем доданого їм в список друзів профілю використовуються дві почтові адреси та паролі аккаунтів, які є вже зарезервованими в програмі. Передумовою

даного тестового випадку є те, що обидва профілі додали одне одного в список друзів. В ході тесту один користувач шляхом використання спеціальних вікон програми повинен отримати доступ до точки місцезнаходження іншого користувача, таким чином розглядається здатність програмного модуля здійснювати *GPS*-моніторинг (табл. 4.2).

Висновком даного тестового випадку є те, що можливість відстеження місцезнаходження користувачем доданого їм в список друзів профілю програмного модулю працює, таким чином це робить можливим проводити роботи з відслідковуванням точки геолокації користувачів додатку.

Таблиця 4.2

Тестовий випадок відстеження місцезнаходження користувачем доданого їм в список друзів профілю

Дія	Очікуваний результат	Фактичний результат
Передумова		
Відкрити програмний модуль батьківського контролю з функцією *GPS*-моніторингу. Провести операцію додавання до списку друзів необхідного для тестування профілю	Необхідний для тестування профіль додано до списку друзів	Пройдено
Кроки тесту		
Перейти до головного вікна програмного модуля	Головне вікно завантажилось і демонструє список доданих профілів місцеположення яких можна переглядати	Пройдено

Закінчення таблиці 4.2

На головному екрані додатку натиснути на необхідний для відслідковування профіль	Адреса необхідного профілю натискається та відкривається карта світу, де червоним маркером відображається положення потрібного профілю користувача	Пройдено
Постумова		
Завершити роботу програмного модуля	Робота програмного модуля завершена	Пройдено

4.3. Висновки до розділу

1) Описано інтерфейс користувача, який забезпечує передачу інформації між користувачем та компонентами обробки даних програмного модуля батьківського контролю з функцією *GPS*-моніторингу.

2) Проведено тестування програмного модуля. Для цього використовувалося ручне тестування, в якому застосовувалися властивості програмного модуля для знаходження помилок у роботі. Результати тестування представлені у вигляді позитивних тестових випадків.

За допомогою тест кейсів перевірені наступні можливості програми:

– можливість відправлення запиту на додавання користувачем профілю іншій людині;

– можливість відстеження місцезнаходження користувачем доданого їм в список друзів профілю.

ВИСНОВКИ

1) За результатами аналізу існуючих програмних рішень сформовано вимоги до розробленого в дипломному проекті програмного модуля, які б забезпечували його універсальність, а саме:

– мати можливість створення та адміністрування користувачем профілів за допомогою введеної електронної почти та паролю;

– мати можливість перегляду та пошуку профілів користувачів, що є зареєстрованими в програмному модулі батьківського контролю з функцією *GPS*-моніторингу;

– мати можливість відправлення запиту на додавання профілю користувача у відповідний список для подальшої роботи з ним;

– мати можливість перегляду та пошуку необхідного отриманого запиту на додавання профілю користувача;

– мати можливість відхилити або прийняти запит від іншого користувача на додавання профілю у відповідний список для подальшої роботи з ним;

– мати можливість відслідковувати точку місцезнаходження на карті доданого профілю користувача;

– мати можливість користувачеві змінювати розмір карти на якій знаходиться точка місцезнаходження відповідного профілю;

– мати можливість переглядати в онлайн режимі зміни місцезнаходження доданого профілю користувача;

– мати можливість переглядати дані про момент часу в який профіль користувача, що відслідковується, знаходиться в конкретній точці геолокації;

– мати можливість співпраці з операційною системою *Android*.

2) Проаналізовано методи створення програмного модуля батьківського контролю з функцією *GPS*-моніторингу та обрано мову програмування *Java*, та інтегроване середовище розробки програмного забезпечення *Android Studio*.

3) Запрограмовано перелік вимог до програмного модуля батьківського контролю з функцією *GPS*-моніторингу:

– можливість роботи програмного модуля з серверною базою даних для оперування інформацією користувачів;

– можливість створення користувачем нових профілів за допомогою введених даних, таких як електронна пошта, ім’я та фамілія, а також пароль;

– можливість адміністрування користувачем своїх створених раніше профілів, що є зареєстрованими в програмному додатку;

– можливість перегляду та пошуку необхідних профілів користувачів, що є зареєстрованими в програмному модулі батьківського контролю з функцією *GPS*-моніторингу;

– можливість відправлення запиту на додавання профілю користувача у відповідний список друзів для подальшої роботи з ним;

– можливість отримання сповіщень на телефон користувача про отримані від інших профілів запити, що його аккаунт бажають додати у відповідний список друзів;

– можливість перегляду та пошуку необхідного отриманого запиту від іншого користувача програмного модуля на додавання у список друзів;

– можливість відхилення або прийняття запитів від іншого користувача на додавання профілю у відповідний список для подальшої роботи з ним;

– можливість відслідковувати точку місцезнаходження на карті світу доданого профілю користувача;

– можливість змінювати користувачем програмного модуля отримане зображення карти на якій знаходиться точка місцезнаходження відповідного профілю (масштабування);

– можливість переглядати в онлайн режимі зміни місцезнаходження доданого профілю користувача;

– можливість переглядати дані про момент часу в який профіль користувача, що відслідковується, знаходиться в конкретній точці геолокації;

– можливість взаємодіяти зі стандартними засобами операційної системи *Android* пов’язаними з картами, такими як прокладання маршруту від заданої користувачем адреси до точки геолокації профілю, що відслідковується, а також завантаження сторінки з відомостями та фотографіями про те, що знаходиться біля місця перебування профілю, та його координати.

4) Розроблено структуру програмного модуля, яку умовно можна поділити на дві частини, а саме структурну схему серверного блоку та структурну схему клієнтського блоку.

До структурної схеми серверного блоку входять наступні компоненти:

– база даних, елементами якої є сукупність особистих даних користувача, якими вона оперує;

– блок роботи з базою даних, за допомогою якого встановлюється зв'язок між серверною частиною програмного модуля та самою базою даних;

– блок оновлення індивідуальних ідентифікаторів користувачів;

– блок для збереження та відправлення даних про координати місцезнаходження користувача;

– блок зберігання особистих даних користувача, до яких належать відомості про електронні адреси, паролі та користувацькі ідентифікатори;

– блок оперування профілями, які є доступними користувачу в блоках клієнтської частини програмного модуля.

До структурної схеми клієнтського блоку входять наступні компоненти:

– база даних, елементами якої є сукупність особистих даних користувача, якими вона оперує;

– блок роботи з базою даних, за допомогою якого встановлюється зв'язок між клієнтською частиною програмного модуля та самою базою даних;

– інтерфейс користувача, який виконує функцію передачі інформації між користувачем та блоком роботи з базою даних;

– блок роботи з профілями користувачів, що надає можливість програмному модулю працювати та змінювати в ході роботи вже створені профілі;

– блок відображення усіх можливих профілів місцезнаходження, яких користувач може відслідкувати;

– блок відображення та пошуку усіх профілів, яким користувач може надіслати запит на додавання їх до списку друзів;

– блок відправлення запиту на додавання конкретного профілю до списку друзів;

– блок відображення та пошуку усіх запитів на додавання до списку друзів отриманих від інших користувачів;

– блок відхилення запитів, що забороняє користувачу, який надсилав запит на додавання до списку друзів, оперувати даними місцезнаходження профілю;

– блок прийняття запитів, що дозволяє користувачу подальше отримування та оперування даними місцезнаходження доданого профілю;

– блок *GPS*-моніторингу профілю, що дозволяє користувачу отримувати та оперувати даними точки місцезнаходження доданого аккаунту.

5) Побудовано функціональну схему програмного модуля, до якої належать такі компоненти:

– блок задання значень персональних даних профілю;

– блок задання зареєстрованих персональних даних користувача;

– блок задання нових персональних даних користувача;

– блок задання необхідної користувачеві функції;

– блок генерації даних про точку місцезнаходження користувача;

– блок збереження точки місцезнаходження;

– блок задання значень профілю для відправлення запиту на додання в список друзів;

– блок задання значень профілю для перегляду запитів на додавання;

– блок побудови усіх доступних профілів для додавання їх у список друзів;

– блок побудови повідомлення для сповіщення про новий запит до списку друзів;

– побудова усіх доступних запитів профілів на додавання їх у список друзів;

– блок задання значень профілю для відслідковування;

– блок побудови відображення точки місцезнаходження обраного користувача.

6) Визначено мінімальні системні вимоги програмного модуля батьківського контролю з функцією *GPS*-моніторингу. До них належать:

– операційна система *Android Lollipop* 5.0;

– 512 *МБ* оперативної пам'яті;

– процесор з двома ядрами та тактовою частотою 1200 *МГц*;

– можливість підключення до мережі;

– можливість підключення до *GPS*-локатору.

7) Програмно реалізовано алгоритми:

– алгоритм додавання до профілю користувача аккаунтів людей для їх подальшого *GPS* відслідковування;

– алгоритм оновлення та перегляд змін в онлайн режимі точки місцезнаходження користувачів.

8) Описано інтерфейс користувача, який забезпечує передачу інформації між користувачем та компонентами обробки даних програмного модуля батьківського контролю з функцією *GPS*-моніторингу.

9) Проведено тестування програмного модуля. Результати тестування представлені у вигляді позитивних тестових випадків.

За допомогою тест кейсів перевірені наступні можливості програми:

– можливість відправлення запиту на додавання користувачем профілю іншій людині;

– можливість відстеження місцезнаходження користувачем доданого їм в список друзів профілю.

10) Практична цінність проекту полягає в наданні можливості користувачу використовувати програмний модуль під час подорожей з дітьми, а також в моменти, коли батьки відсутні під час їх самостійного пересування, що знизить ймовірність ризику пропажі дітей.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Дэвид Гриффитс, Дон Гриффитс. Head First. Программирование для Android. – Пер. с англ. – М.: «Питер», 2018. – 912 с.

2) Кристин Марсикано, Билл Филлипс. Android. Программирование для профессионалов. – Пер. с англ. – М.: «Питер», 2017. – 688 с.

3) Пол Дейтел. *Android* для разработчиков. – Пер. с англ. – М.: «Питер», 2016. – 512 с.

4) *Herbert Schildt. Java: A Beginner's Guide // McGraw-Hill Education*. – 2014. – 728 с.

5) *Bill Phillips, Brian Hardy. Android Programming: The Big Nerd Ranch Guide // Big Nerd Ranch Guides*. – 2019. – 624 с.

6) *Mark L. Murphy. Busy Coder's Guide to Android Development // CommonsWare*. – 2009. – 468 с.

7) *Greg Nudelman. Android Design Patterns: Interaction Design Solutions for Developers // Wiley*. – 2013. – 456 с.

8) Ян Дарвин. Сборник рецептов. – М.: «Диалектика». – 2018. –768 с.

9) Фрайман Зэев. Создание приложений для смартфонов и планшетов под ОС *Android*. – М.: «Ленанд». – 2019. – 504 с.

10) Эд Бурнет. Привет, *Android*!– М.: «Питер». – 2012. – 253 с.

11) Тереза Нейл. Мобильная разработка.– М.: «Питер». – 2013. – 208 с.

12) Джошуа Блох. Эффективное программирование. – М.: «Лори». – 2014. – 323 с.

13) Эккель Брюс. Философия Java. – М.: «Питер». – 2019. – 432 с.

14) ДСТУ 3008-95 Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.

15) Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: НАУ, 2017. – 63 с.

ДОДАТОК А

Лістинг коду головного вікна програмного модуля

Home.java

```
public class HomeActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener,
IFirebaseLoadDone {
        FirebaseRecyclerAdapter<User, UserViewHolder> adapter, searchAdapter;
        RecyclerView recycler_friend_list;
        IFirebaseLoadDone firebaseLoadDone;
        MaterialSearchBar searchBar;
        List<String> suggestList = new ArrayList<>();
        LocationRequest locationRequest;
        FusedLocationProviderClient fusedLocationProviderClient;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
           setContentView(R.layout.activity_home);
           Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
           setSupportActionBar(toolbar);
           FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
                fab.setOnClickListener((view) -> {
              startActivity(new Intent(HomeActivity.this,AllPeopleActivity.class));
        });
           DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
           ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
```

```
        drawer.addDrawerListener(toggle);
        toggle.syncState();
        NavigationView navigationView = (NavigationView)
findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);
        navigationView.setItemIconTintList(null);
        View headerView = navigationView.getHeaderView(0);
        TextView txt_user_logged = (TextView)
headerView.findViewById(R.id.txt_logged_email);
        txt_user_logged.setText("Welcome back, " +
Common.loggedUser.getEmail());
        searchBar = (MaterialSearchBar) findViewById(R.id.material_search_bar);
        searchBar.setCardViewElevation(10);
        searchBar.addTextChangeListener(new TextWatcher() {

    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
    }
    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
       List<String> suggest = new ArrayList<>();
       for (String search : suggestList) {
          if (search.toLowerCase().contains(searchBar.getText().toLowerCase()))
             suggest.add(search);
       }
       searchBar.setLastSuggestions(suggest);
    }

    @Override
    public void afterTextChanged(Editable s) {
```

```
        }
    });
    searchBar.setOnSearchActionListener(new
MaterialSearchBar.OnSearchActionListener() {
        @Override
        public void onSearchStateChanged(boolean enabled) {
            if (!enabled) {
                if (adapter != null) {
                    //if close search, restore defaults
                    recycler_friend_list.setAdapter(adapter);
                }
            }
        }
        @Override
        public void onSearchConfirmed(CharSequence text) {
            startSearch(text.toString());
        }
        @Override
        public void onButtonClicked(int buttonCode) {

        }
    });
    recycler_friend_list = (RecyclerView) findViewById(R.id.recycler_friend_list);
    recycler_friend_list.setHasFixedSize(true);
    RecyclerView.LayoutManager layoutManager = new LinearLayoutManager(this);
    recycler_friend_list.setLayoutManager(layoutManager);
    recycler_friend_list.addItemDecoration(new DividerItemDecoration(this,
((LinearLayoutManager) layoutManager).getOrientation()));
    updateLocation();
    firebaseLoadDone = this;
```

```
        loadFriendList();
        loadSearchData();
    }
    private void loadSearchData() {
        List<String> lstUserEmail = new ArrayList<>();
        DatabaseReference userList = FirebaseDatabase.getInstance()
                .getReference(Common.USER_INFORMATION)
                .child(Common.loggedUser.getUid())
                .child(Common.ACCEPT_LIST);
        userList.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                for(DataSnapshot userSnapShot:dataSnapshot.getChildren())
                {
                    User user = userSnapShot.getValue(User.class);
                    lstUserEmail.add(user.getEmail());
                }
                firebaseLoadDone.onFirebaseLoadUserNameDone(lstUserEmail);
            }
            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
                firebaseLoadDone.onFirebaseLoadFailed(databaseError.getMessage());
            }
        });

    }
    private void loadFriendList() {
        Query query = FirebaseDatabase.getInstance()
                .getReference(Common.USER_INFORMATION)
                .child(Common.loggedUser.getUid())
```

```
            .child(Common.ACCEPT_LIST);
        FirebaseRecyclerOptions<User> options = new
FirebaseRecyclerOptions.Builder<User>().setQuery(query,User.class).build();
        adapter = new FirebaseRecyclerAdapter<User, UserViewHolder>(options) {
            @Override
            protected void onBindViewHolder(@NonNull UserViewHolder holder, int
position, @NonNull User model) {
                holder.txt_user_email.setText(new StringBuilder(model.getEmail()));

                holder.setiRecyclerItemClickListener(new IRecyclerItemClickListener() {
                    @Override
                    public void onItemClickListener(View view, int position) {
                        //Show tracking
                        Common.trackingUser = model;
                        startActivity(new Intent(HomeActivity.this,TrackingActivity.class));
                    }
                });
            }
            @NonNull
            @Override
            public UserViewHolder onCreateViewHolder(@NonNull ViewGroup
viewGroup, int viewType) {
                View itemView =
LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.layout_user,viewGroup,
false);
                return new UserViewHolder(itemView);
            }
        };
        adapter.startListening();
        recycler_friend_list.setAdapter(adapter);
```

```
    }
    @Override
    protected void onStop() {
        if(adapter != null)
            adapter.stopListening();
        if(searchAdapter != null)
            searchAdapter.stopListening();
        super.onStop();
    }
    @Override
    protected void onResume() {
        super.onResume();
        if(adapter != null)
            adapter.startListening();
        if(searchAdapter != null)
            searchAdapter.startListening();
    }
    private void updateLocation() {
        buildLocationRequest();
        fusedLocationProviderClient =
LocationServices.getFusedLocationProviderClient(this);
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            return;
        }
```

```
        fusedLocationProviderClient.requestLocationUpdates(locationRequest,
getPendingIntent());
    }
    private PendingIntent getPendingIntent() {
        Intent intent = new Intent(HomeActivity.this,MyLocationReceiver.class);
        intent.setAction(MyLocationReceiver.ACTION);
        return
PendingIntent.getBroadcast(this,0,intent,PendingIntent.FLAG_UPDATE_CURRENT);
    }
    private void buildLocationRequest() {
        locationRequest = new LocationRequest();
        locationRequest.setSmallestDisplacement(10f);
        locationRequest.setFastestInterval(3000);
        locationRequest.setInterval(5000);
        locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
    }
    private void startSearch(String search_value) {
        Query query = FirebaseDatabase.getInstance()
                .getReference(Common.USER_INFORMATION)
                .child(Common.loggedUser.getUid())
                .child(Common.ACCEPT_LIST)
                .orderByChild("name")
                .startAt(search_value);

        FirebaseRecyclerOptions<User> options = new
FirebaseRecyclerOptions.Builder<User>().setQuery(query,User.class).build();
        searchAdapter = new FirebaseRecyclerAdapter<User, UserViewHolder>(options)
{
            @Override
```

```
        protected void onBindViewHolder(@NonNull UserViewHolder holder, int
position, @NonNull User model) {
            holder.txt_user_email.setText(new StringBuilder(model.getEmail()));
            holder.setiRecyclerItemClickListener(new IRecyclerItemClickListener() {
                @Override
                public void onItemClickListener(View view, int position) {
                    //Show tracking
                    Common.trackingUser = model;
                    startActivity(new Intent(HomeActivity.this,TrackingActivity.class));
                }
            });
        }
        @NonNull
        @Override
        public UserViewHolder onCreateViewHolder(@NonNull ViewGroup
viewGroup, int viewType) {
            View itemView =
LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.layout_user,viewGroup,
false);
            return new UserViewHolder(itemView);
        }
    };
    searchAdapter.startListening();
    recycler_friend_list.setAdapter(adapter);
    //recycler_friend_list.setAdapter(searchAdapter); }
  @Override
  public void onBackPressed()
  {
    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    if(drawer.isDrawerOpen(GravityCompat.START))
```

```
    {
        drawer.closeDrawer(GravityCompat.START);
    }else
    {
        super.onBackPressed();
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.activity_main_drawer, menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.action_settings) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}
@Override
public boolean onNavigationItemSelected(@NonNull MenuItem item)
{
    int id = item.getItemId();
    if (id == R.id.nav_find_people) {
        startActivity(new Intent(HomeActivity.this,AllPeopleActivity.class));
    }
    else if (id == R.id.nav_add_people) {
        startActivity(new Intent(HomeActivity.this,FriendRequestActivity.class));
```

```
        } else if (id == R.id.nav_sign_out) {
        }
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }
    @Override
    public void onFirebaseLoadUserNameDone(List<String> lstEmail) {
        searchBar.setLastSuggestions(lstEmail);
    }
    @Override
    public void onFirebaseLoadFailed(String message) {
        Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
    }
}
```