Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

**Бакалаврська дипломна робота на тему:**

# «МОБІЛЬНИЙ ДОДАТОК ДЛЯ ЗАБЕЗПЕЧЕННЯ ОБМЕЖЕНОГО РЕЖИМУ ДОСТУПУ НА МОБІЛЬНОМУ ПРИСТРОЇ»

Виконав: студент 4 курсу, групи 2ІСТ-20б спеціальності 126 – «Інформаційні системи та технології»
_____Анастасія МОНАСТИРСЬКА
Керівник: к.т.н., доц. каф. САІТ
_____ Євгеній КРИЖАНОВСЬКИЙ
«31» _____05_____ 2024 р.
Рецензент: к.т.н., доц. каф. КН
_____Володимир ОЗЕРАНСЬКИЙ
«18» _____06_____ 2024 р.

**Допущено до захисту**

Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«05» ____06_____ 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність 126 – «Інформаційні системи та технології»
Освітньо-професійна програма – Прикладні інформаційні технології

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
"05" 05 2024 року

**ЗАВДАННЯ**
**НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ**
Монастирській Анастасії Юріївні

1. Тема роботи: «Мобільний додаток для забезпечення обмеженого режиму доступу на мобільному пристрої»
керівник роботи: Крижановський Є.М., к.т.н., доц. каф. САІТ
затверджені наказом ВНТУ від «11» 03 2024 року №80
2. Термін подання студентом роботи 31.05
3. Вихідні дані до роботи:
1) Мова програмування: Kotlin;
2) Платформа для розробки: Andoid;
3) Відображати лише ті додатки, які обрали батьки.
4. Зміст текстової частини:
1) Аналіз проблематики забезпечення обмеженого режиму доступу на мобільному пристрої.
2) Вибір оптимальних інформаційних технологій для розробки.
3) Розробка додатку для забезпечення обмеженого режиму доступу на мобільному пристрої.
5. Перелік ілюстративного матеріалу:
1) Діаграма use case;
2) Діаграма взаємодії;
3) Діаграма варіантів використання;
4) Діаграма класів;
5) Інтерфейс додатку.

6. Консультанти розділів роботи:

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання 11.05

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва та зміст етапу | Термін виконання | | Примітка |
|---|---|---|---|---|
| | | початок | закінчення | |
| 1 | Вступ | 11.03 | 15.05 | |
| 2 | Аналіз предметної області | 15.03 | 31.03 | |
| 3 | Вибір оптимальних інформаційних технологій | 01.04 | 15.04 | |
| 4 | Розроблення додатку для забезпечення обмеженого режиму доступу на мобільному пристрої | 16.04 | 15.05 | |
| 5 | Оформлення матеріалів до захисту БДР | 16.05 | 31.05 | |

Студент _____ Анастасія МОНАСТИРСЬКА

Керівник роботи _____ Євгеній КРИЖАНОВСЬКИЙ

# АНОТАЦІЯ

Бакалаврська дипломна робота складається з 55 сторінок формату А4, на яких є 27 рисунка, список використаних джерел містить 20 найменувань.

Метою дослідження є розробка мобільного додатку для забезпечення обмеженого режиму доступу на мобільному пристрої.

В роботі наведено розроблення мобільного додатку на платформі Android, за допомогою мови програмування Kotlin. Розроблено дизайн та функціонал мобільного додатку за допомогою мови розмітки XAML. Розроблена система забезпечує можливість керування режимом обмеженого доступу на мобільному пристрої.

Ключові слова: інформаційна система, мобільний додаток, дитячий режим Kotlin, Android Studio.

# ABSTRACT

The bachelor thesis consists of 50 pages of A4 format, on which there are 27 pictures, the list of used sources contains 11 names.

The purpose of the study is to develop a mobile application to provide a limited access mode on a mobile device.

The paper describes the development of a mobile application on the Android platform using the Kotlin programming language. The design and functionality of the mobile application was developed using the XAML markup language. The developed system provides the ability to control the restricted access mode on a mobile device.

Keywords: information system, mobile application, Kotlin child mode, Android Studio.

# ЗМІСТ

# ВСТУП

**Актуальність теми**. У сучасному світі мобільні пристрої відіграють важливу роль у повсякденному житті, надаючи користувачам можливість доступу до різноманітної інформації та сервісів. Однак разом зі зростанням функціональності мобільних додатків виникають нові виклики, пов'язані з безпекою та конфіденційністю даних. Однією з ключових проблем є забезпечення обмеженого режиму доступу на мобільних пристроях, що дозволяє захистити особисті дані користувачів від несанкціонованого доступу, а саме – випадкового використання коли пристрій передається дитині. За статистикою, кількість випадків здійснення покупок, ввімкнення функцій і надсилання повідомлень зростає щороку, що підтверджує актуальність розробки надійних рішень для гарантування безпеки і обмеження доступу.

**Мета і задачі дослідження.** Метою дослідження є розробка мобільного додатку для забезпечення обмеженого режиму доступу на мобільному пристрої.

Для досягнення поставленої мети необхідно розв’язати такі задачі:

— провести аналіз існуючих рішень щодо обмеження доступу на мобільних пристроях;

— вибрати оптимальні технології та платформи для розробки додатку;

— розробити мобільний додаток, що забезпечує обмежений режим доступу;

— провести тестування розробленого додатку для оцінки його ефективності та надійності.

**Об’єктом дослідження** Об’єктом дослідження є процес розробки мобільного додатку для забезпечення обмеженого режиму доступу на мобільному пристрої.

**Предметом дослідження** є методи та програмні засоби розробки мобільного додатку для забезпечення обмеженого режиму доступу на мобільному пристрої.

**Публікації**. За результатами виконання даної роботи опубліковано тези доповідей на тему: «Мобільний додаток для забезпечення обмеженого режиму доступу на мобільному пристрої» на Міжнародній науково-практичній інтернет-

конференції «Молодь в науці: дослідження, проблеми, перспективи (Вінниця, 2023-2024 рр.) [1].

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Аналіз предметної області

Додатки "батьківського контролю" — це програмні рішення, призначені для керування і контролю активностей дітей на мобільних пристроях та в інтернеті. Основні функції таких додатків включають:

1. Контроль доступу до додатків і контенту: Батьківські контрольні додатки дозволяють батькам обмежувати доступ дітей до певних додатків або типів контенту, таких як ігри або соціальні мережі. Це може включати встановлення часових обмежень на використання додатків або блокування окремих програм з метою контролю часу, який діти проводять за пристроєм.
2. Фільтрація веб-контенту: Деякі додатки батьківського контролю пропонують фільтрацію веб-контенту, що дозволяє обмежувати доступ до певних веб-сайтів або категорій контенту. Це може бути важливо для захисту дітей від небажаного або неналежного матеріалу в Інтернеті.
3. Моніторинг використання пристрою: Батьківські контрольні додатки надають інформацію про те, як і скільки часу витрачають діти на різних додатках та веб-сайтах. Це дозволяє батькам керувати часом, що витрачається на екрані, і встановлювати здорові звички використання технологій.
4. Моніторинг місцезнаходження і геозонування: Деякі додатки дозволяють батькам відстежувати місцезнаходження своїх дітей і створювати геозони, що сповіщають, коли дитина виходить за межі певної території. Це може бути корисним для забезпечення безпеки дітей під час їх переміщень.
5. Управління контактами і повідомленнями: Деякі додатки дозволяють батькам керувати контактами і повідомленнями, що надходять на пристрій дитини. Це може включати блокування небажаних контактів або контроль над тим, з ким дитина може обмінюватися повідомленнями.

Додатки батьківського контролю з'явилися як реакція на зростання використання мобільних технологій серед дітей і потребу батьків у засобах для ефективного контролю та захисту їхнього добробуту. Вони стали важливим інструментом для забезпечення безпеки та виховання дітей в цифровому віці. Розглянемо статистику (рис. 1.1 та 1.2) [1-3].

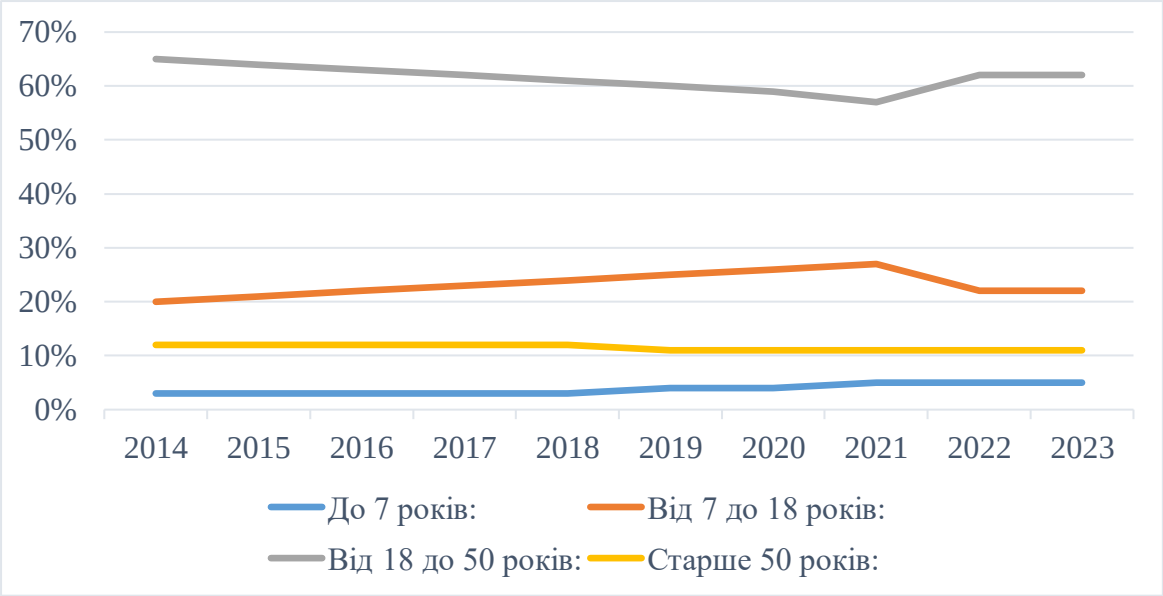


Рисунок 1.1 – Статистика скачування ігор за віком

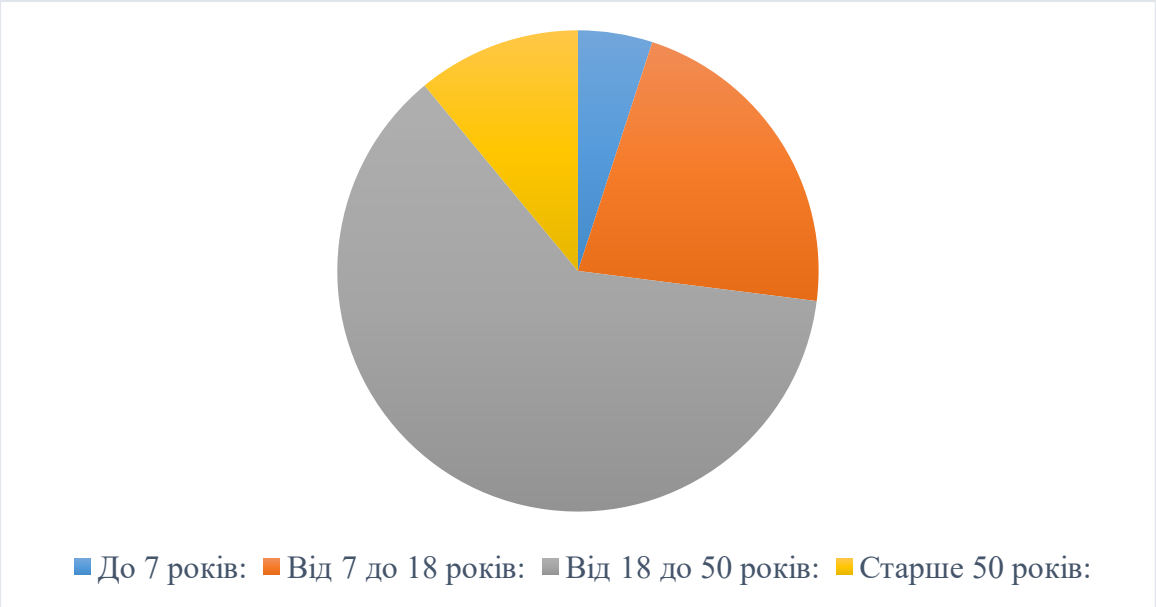


Рисунок 1.2 – Відношення вікових категорій

У середньому, до семи років діти не ходять до школи, вважаються неграмотними та користуються телефонами батьків. Тому статистика щодо використання мобільних пристроїв дітьми базується на даних, які вводяться у додатки, що запитують вік користувача. За наявними даними, спостерігається зростання відсотка завантаження додатків, у яких вказано вік користувача до семи років. Це вказує на те, що діти починають взаємодіяти з мобільними технологіями у все більш ранньому віці, що є важливою тенденцією для розуміння змін у сучасному суспільстві. Також слід враховувати, що більш молоді батьки можуть створювати електронні пошти на ім'я дитини для завантаження додатків. Проте ці випадки, як і наявність власного пристрою з більш раннього віку, слід відносити до винятків [3].

Варто також відзначити різку відмінність у кількості завантажених ігор між віковою категорією дітей, які мають власні пристрої (7-18 років), та дорослих людей (18-50 років). Дорослі люди у нашому столітті, як правило, вільно володіють своїми пристроями і часто передають телефон у руки дитини (до семи років). Тому наявність у такого користувача (дата народження з акаунту Play-маркету) дитячих ігор, призначених для малих дітей, не є дивною.

Це вказує на важливі соціальні й технологічні зміни, зокрема на те, як ранній доступ до технологій може впливати на розвиток дітей. Дослідження показують, що взаємодія з мобільними пристроями у ранньому віці може мати як позитивні, так і негативні наслідки. Позитивні аспекти включають розвиток когнітивних і моторних навичок, навчання через освітні додатки. Негативні ж аспекти можуть стосуватися впливу на фізичне здоров'я, зокрема на зір, а також ризиків, пов'язаних із надмірним використанням пристроїв і доступом до небажаного контенту [4].

Окрім цього, слід звернути увагу на літніх людей, які також можуть бути власниками мобільних пристроїв. Доволі часто вони використовують застарілі телефони, які не входять у дану статистику, або ж бояться використовувати сучасні пристрої через складність налаштувань. Часто літні люди дають свої пристрої в руки довіреній людині для коригування налаштувань або перевірки повідомлень, що може бути небезпечним для них. Це створює ризики щодо

конфіденційності їхніх даних та можливого несанкціонованого доступу до важливої інформації.

Таким чином, зростання віку, коли дитина вперше бере в руки телефон, і тенденції, пов'язані з цим, є важливими темами для подальших досліджень і аналізу. Водночас, необхідно приділяти увагу підтримці літніх людей у використанні сучасних технологій, забезпечуючи їм безпечний та зрозумілий доступ до мобільних пристроїв. Ці питання вимагають комплексного підходу та подальшого розвитку стратегій захисту користувачів усіх вікових категорій у цифровому середовищі.

Хоч і є достатньо ігор, розробники яких турбуються про наявність мір безпеки та намагаються зробити інтерфейс зрозумілим інтуїтивно, це не завжди працює. Для людей, які вперше беруть сенсорний телефон в руки, освоїтись в численній кількості іконок та позначок може бути надто складно [4].

Наприклад, літні люди часто стикаються з труднощами при переході від традиційних кнопкових телефонів до сенсорних пристроїв. Відсутність фізичних кнопок, змінений спосіб навігації та необхідність взаємодії з екраном за допомогою жестів можуть викликати у них значний стрес. Складність інтерфейсів, надмірна кількість функцій та непередбачуваність поведінки програм можуть стати причинами відмови від використання сучасних технологій.

Навіть у випадках, коли інтерфейси спрощуються, це не завжди гарантує успіх. Наприклад, збільшення розміру іконок або спрощення меню може не компенсувати відсутність попереднього досвіду взаємодії з сенсорними екранами. Діти можуть не розуміти, як виконати базові операції, такі як відповідь на дзвінок, відправлення текстових повідомлень, прибирання віконця повідомлення з екрану, або налаштування параметрів пристрою. Це створює бар'єри, які не завжди можуть бути подолані за допомогою інтуїтивно зрозумілих дизайнів.

Для дітей раннього віку ситуація теж не завжди проста. Хоча вони можуть швидко навчитися основам взаємодії зі смартфонами, спостерігаючи за батьками, випадкового виходу з гри або переходу у інший додаток при

помилковому натисненні на повідомлення може створити велику проблему, коли батько чи мати ведуть машину або займаються іншими справами, де неможна відволікатись.

Хоча зусилля розробників щодо забезпечення безпеки та інтуїтивно зрозумілих інтерфейсів є важливими, їх недостатньо для всіх користувачів. Необхідні додаткові заходи, такі як навчальні програми, персональна допомога та консультації для нових користувачів, на якій не завжди вистачає зусиль, часу та сил.

## 1.2 Огляд аналогів

На даний момент відсутні аналоги, які мать повністю схожий простий набір функцій, проте є доволі багато тих, хто вміщує частково де-які з них. Розберемо приклади.

Перший аналог являється вбудованою функцією Android, гостьовий режим. Головна особливість – він являє собою повну ізоляцію акаунту власника на термін, коли користувачем стає інша людина, але гостьовий режим при кожному використанні можна скинути і працювати як з окремим чистим акаунтом без додатків. Також він вимагає додаткових налаштувань, окремого завантаження додатків (рис. 1.3) [5].

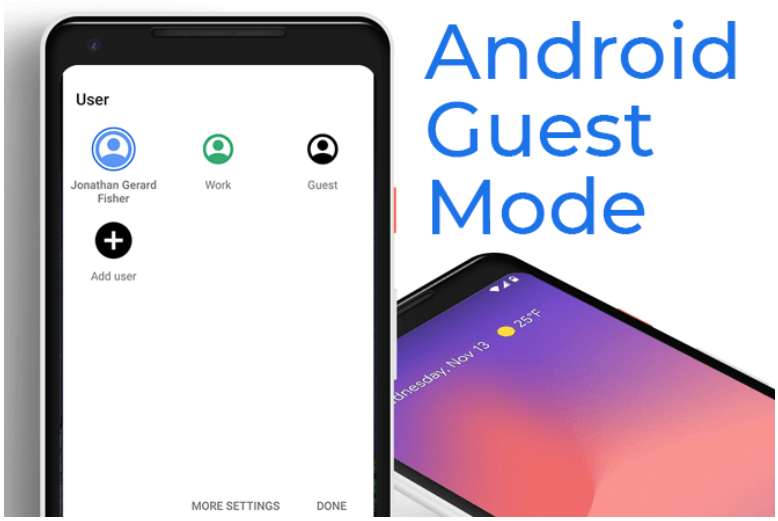


Рисунок 1.3 – Вбудований метод ізоляції акаунту

Частково, проблеми вирішує процес створення окремого акаунту, для дитини, але пристрій збереже всі функції, доступні до цього. Тобто, дитина матиме повний доступ до налаштувань, але батьки не мають змоги переглянути повідомлення або отримати дзвінок, а повернення до головного акаунту матиме ряд складнощів. По-перше, процес перемикання може займати до п’яти хвилин, а по-друге – дана можливість значно зменшує продуктивність пристрою.

Інший чудовий приклад – те, як додатки захищають функції купівлі послуг у дитячих іграх: наявність математичного прикладу для відповідної вікової категорії або культурного питання, працює так само добре, але не запобігає виходу з додатку (рис. 1.4).

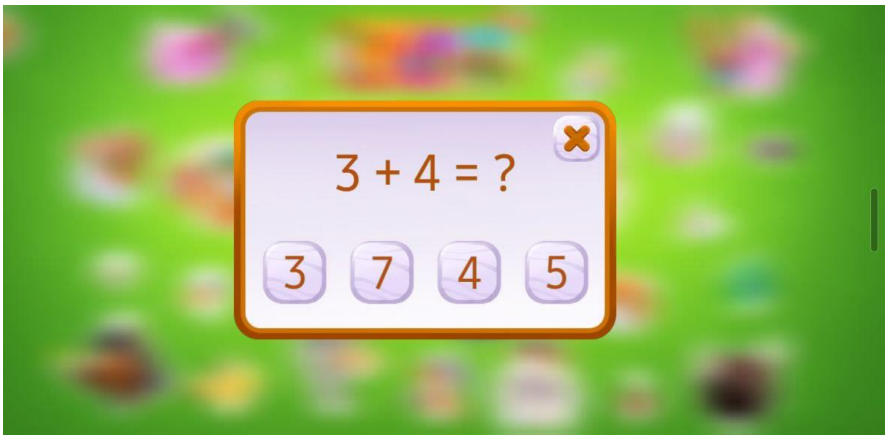


Рисунок 1.4 – Блокування можливості купівлі за реальні гроші в додатку

Для дітей шкільного віку таке блокування наврядчи спрацює, але додаток наведений як приклад призначений для 1-3 років, коли діти ще не вміють рахувати і читати [4-5].

Для батьків це не складна перешкода, проте надійна: не витрачатиметься пам'ять і ресурси телефону для відокремлення простору дитини та власника, але нема можливості зайти у частину додатку, де треба сплачувати реальними грошима або вийти з додатку. Також тка перешкода не завадить вчасно побачити повідомлення або прийняти телефонний дзвінок [6].

Єдина проблема подібної перешкоди – даний вид захисту є вбудованим в окремі додатки та часто саме у платній версії.

Наступний на черзі – звичайний «батьківський контроль». Основною перевагою даного методу обмеження являється те, що батьки неодмінно отримають повідомлення на споріднений пристрій або на пошту про кроки дитини. «Батьківський контроль» існує на багатьох рівнях і вікових зонах: для малих дітей – контроль закачаних ігор та відвіданих сайтів в інтернеті, переглянутих відео на ютубі; для більш дорослих – контроль банківських витрат, обмеження екранного часу та інше. Проте, це все розраховано на час, коли дитина вже досвідчена та має свій власний пристрій.

Варто відзначити, що «Батьківський контроль» ніяк не спрощує використання самого телефону для дитини та не обмежує використовувані функції.

## 1.3 Формування технічного завдання.

У рамках розробки мобільного додатку передбачено вирішення певних завдань та відповідних функціональних можливостей, що відповідають потребам цільової аудиторії і нефункціональним вимогам.

Метою цього проекту є створення засобу управління доступом дітей до додатків та телефоні, планшеті чи іншого пристрою, який використовує OS Android [6].

Основні функціональні можливості додатку передбачають управління самої можливості використання програм, на які не дає дозволу власник телефону.

Нефункціональні вимоги, такі як безпека, продуктивність і використовуваність, відіграють важливу роль у розробці цього додатку. Додаток повинен бути безпечним для використання дітьми, а доступ до деактивації додатку повинен мати лише основний користувач пристрою. Повинна забезпечуватися продуктивна робота без збоїв на різних пристроях, а також простий і зрозумілий інтерфейс для батьків та дітей.

Метою даної роботи є розробка мобільного додатку, який дозволить батькам обмежувати доступ до певних додатків і функцій телефону під час

використання їх дитиною або людиною, яка не має достатньо досвіду для повноцінного керування смартфоном, ізолюючи усі інші додатки під час сеансу використання пристрою. При цьому враховується, що це пристрій належить або адмініструється більш дорослою людиною [6].

Основною функцією додатку є запит пароля, коду, рішення задачі або іншого питання для виходу з обраних додатків. На користувацькому екрані додатку будуть відображені лише ті програми, які обрали батьки та надали доступ до їх використання.

Крім того, додаток буде здатен блокувати повідомлення, які було визначено батьками як небажані. При виході з додатку всі надані дозволи автоматично скасовуються і повертаються до початкового стану, забезпечуючи додатковий рівень контролю за використанням пристрою дитиною.

Додаток повинен містити підказки та простий інтерфейс.

## 1.4 Висновки

У цьому розділі було проведено аналіз проблематики предметної області, а саме проблеми використання батьківських пристроїв їх малими дітьми. Розглянуто основні можливості, переваги і недоліки існуючих інформаційних систем обмеження та контролю використання додатками та сформовано основні вимоги до додатку.

# 2 ВИБІР ОПТИМАЛЬНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

## 2.1 Використання можливостей мови програмування Kotlin

Вибір мов програмування для розробки додатків на платформі Android та створення їхнього інтерфейсу визначає результативність, продуктивність і зручність процесу розробки. На сьогоднішній день Kotlin і XAML вважаються одними з найперспективніших мов програмування для цієї мети, кожна з яких має свої унікальні особливості і переваги. Розумний вибір мови програмування сприяє оптимізації процесу розробки, забезпечуючи швидкість, безпеку та зручність роботи з програмним кодом.

Kotlin - це сучасна мова програмування, яка має численні особливості і переваги для розробки мобільних додатків на платформі Android [7]. Ось деякі з них:

1. Повна сумісність з Java: Kotlin ідеально поєднується з існуючим Java-кодом, що дозволяє розробникам поступово переходити на Kotlin, не переписуючи весь існуючий код з Java.
2. Безпека і надійність: Kotlin надає вбудовану підтримку нульової безпеки (null safety) та перевірку типів на етапі компіляції, що дозволяє уникнути багів, пов'язаних з неправильною роботою з пам'яттю.
3. Простота синтаксису: Kotlin має чистий і зрозумілий синтаксис, що полегшує розуміння коду і сприяє підвищенню продуктивності розробників.
4. Функціональна мова програмування: Kotlin підтримує функціональність, що дозволяє використовувати високорівневі абстракції і уникати зайвого коду.
5. Розширені можливості: Kotlin має багатий набір функцій, таких як розширення функціональності класів (extension functions), корутини (coroutines) для асинхронного програмування, а також підтримку DSL (domain-specific languages) для створення власних мовних конструкцій.

6. Інтероперабельність з Java і Android SDK: Kotlin легко інтегрується з існуючими Java-бібліотеками і Android SDK, що дозволяє використовувати всі можливості цих платформ без перешкод.

Kotlin має широкий спектр застосувань, які виходять за межі розробки мобільних додатків для платформи Android [7-8].

1. Розробка серверних додатків: Kotlin можна використовувати для створення серверних додатків завдяки його сумісності з популярними фреймворками, такими як Spring Boot, Ktor та інших. Мова надає всі необхідні інструменти для створення масштабованих і високопродуктивних серверних рішень.
2. Розробка веб-додатків: Kotlin можна використовувати для написання веб-додатків. Завдяки підтримці JavaScript, Kotlin дозволяє розробникам писати як серверну, так і клієнтську частину веб-додатків однією мовою, що значно спрощує процес розробки та підтримки.
3. Розробка багатоплатформних додатків: Kotlin/Multiplatform дозволяє використовувати одну кодову базу для розробки додатків, що працюють на різних платформах, включаючи Android, iOS, JVM і JavaScript. Це значно знижує витрати на розробку та підтримку додатків для різних платформ.
4. Наукові обчислення та аналіз даних: Kotlin можна використовувати для виконання наукових обчислень та аналізу даних завдяки його сумісності з існуючими бібліотеками на Java, такими як Apache Commons Math, а також з новими бібліотеками, розробленими спеціально для Kotlin.
5. Розробка програмного забезпечення для вбудованих систем: Завдяки своїй сумісності з JVM, Kotlin може бути використаний для розробки програмного забезпечення для вбудованих систем, що працюють на Java-базованих платформах.
6. Розширення існуючих Java-додатків: Оскільки Kotlin повністю сумісний з Java, його можна використовувати для додавання нових функціональностей до існуючих Java-додатків, що дозволяє поступово

впроваджувати Kotlin в проекти без необхідності повного переписування існуючого коду.

При розробці додатку для платформи Android і створенні його інтерфейсу, вибір мови програмування відіграє ключову роль у забезпеченні ефективності та зручності процесу розробки. Особливо важливі властивості мови Kotlin для розробки додатків та XAML для створення інтерфейсу. Kotlin є сучасною мовою програмування, спеціально створеною для платформи Android, яка поєднує в собі продуктивність Java з безпекою та чистотою функціональних мов. Її повна сумісність з Java дозволяє безперешкодно інтегрувати Kotlin-код з існуючим Java-кодом, спрощуючи процес переходу від Java до Kotlin. Безпека і надійність Kotlin, зокрема нульова безпека та перевірка типів на етапі компіляції, забезпечують відсутність типових помилок під час виконання програми. Простота синтаксису Kotlin полегшує розуміння коду і підвищує продуктивність розробників [8].

## 2.2 Використання мови розмітки XAML

XAML (Extensible Application Markup Language) - це декларативна мова розмітки на основі XML, розроблена компанією Microsoft для ініціалізації структурованих значень та об'єктів. Її використовують для створення користувацьких інтерфейсів (UI) на різних платформах та фреймворках Microsoft, таких як [9]:

- Windows Presentation Foundation (WPF): для створення десктопних програм Windows.
- Silverlight: для створення веб-додатків та мобільних програм.
- Windows UI Library (WinUI): для створення сучасних десктопних програм Windows.
- Universal Windows Platform (UWP): для створення кросплатформних додатків Windows.
- .NET Multi-platform App UI (.NET MAUI): для створення кросплатформних мобільних та десктопних програм.

XAML використовує декларативний підхід, де розробник описує, як має виглядати UI, а не як його програмно реалізувати. Це спрощує процес створення UI, роблячи його більш візуальним та інтуїтивно зрозумілим [10].

Переваги використання XAML:

1. Відокремлення дизайну від логіки: XAML дозволяє розділяти UI-розмітку від логіки бізнес-процесів, що покращує структуру та підтримку коду.
2. Підтримка інструментів: XAML добре підтримується різними інструментами розробки, такими як Visual Studio та Blend for Visual Studio, які надають потужні засоби для візуального дизайну інтерфейсів.
3. Уніфікований підхід: Використання XAML для різних платформ забезпечує уніфікований підхід до створення UI, що полегшує роботу розробникам, які працюють з декількома платформами.
4. Анімації та мультимедіа: XAML підтримує анімації, мультимедійний контент та складні візуальні ефекти, що дозволяє створювати багаті і динамічні користувацькі інтерфейси.
5. Можливість використання стилів та шаблонів: XAML дозволяє застосовувати стилі та шаблони для елементів UI, що сприяє уніфікованому зовнішньому вигляду та легкості в управлінні зовнішнім виглядом додатків.
6. Підтримка даних: Вбудована підтримка механізмів прив'язки даних (data binding) дозволяє ефективно працювати з джерелами даних та динамічно оновлювати UI відповідно до змін у даних.

XAML (Extensible Application Markup Language) з моменту свого появлення в складі Windows Presentation Foundation (WPF) у 2006 році стала важливим інструментом для створення користувацьких інтерфейсів на платформах Microsoft. З часом XAML зазнала значних змін і була впроваджена в різні платформи та фреймворки [10].

Історія розвитку XAML

- 2006: XAML вперше з'явився у складі Windows Presentation Foundation (WPF) як частина .NET Framework 3.0.

- 2007: З випуском Silverlight 1.0 XAML стала використовуватися для створення веб-додатків та мобільних програм.
- 2012: З випуском Windows 8 та Windows Phone 8 XAML стала основною мовою розмітки для Universal Windows Platform (UWP).
- 2018: З випуском .NET Core 3.0 XAML стала доступною на інших платформах, крім Windows.
- 2021: З випуском .NET 6 та .NET Multi-platform App UI (.NET MAUI) XAML стала використовуватися для створення кросплатформних мобільних та десктопних програм.

Ключові особливості XAML [11]:

1. Декларативний підхід: XAML описує, як має виглядати UI, а не як його програмно реалізувати. Це дозволяє розробникам зосередитися на візуальних аспектах інтерфейсу, не відволікаючись на деталізації коду.
2. Розширюваність: XAML можна розширювати за допомогою власних елементів та властивостей, що забезпечує високу гнучкість і можливість створення користувацьких компонентів.
3. Взаємодія з кодом: XAML тісно інтегрується з мовами програмування .NET, такими як C#, що дозволяє розробникам легко додавати динамічну поведінку до UI та створювати складні інтерфейси.
4. Підтримка шаблонів: XAML підтримує шаблони, які дозволяють повторно використовувати код UI, що значно спрощує розробку і зменшує кількість повторюваного коду.
5. Підтримка зв'язування даних: XAML підтримує зв'язування даних (data binding), що дозволяє синхронізувати UI з даними і забезпечує автоматичне оновлення інтерфейсу при зміні даних.

XAML є потужним інструментом для створення користувацьких інтерфейсів. Її декларативний підхід, розширюваність та інтеграція з мовами програмування .NET роблять її популярним вибором для розробників, які створюють програми для платформ Android, Apple та Microsoft. Завдяки своїм можливостям XAML сприяє більш продуктивній розробці, дозволяючи зосередитися на створенні якісних і зручних інтерфейсів для користувачів.

Проте, існують певні недоліки та проблеми при використанні XAML:

- Складність навчання: для новачків може бути складно освоїти XAML через її численні можливості та концепції.
- Великий обсяг коду: декларативний підхід іноді може призводити до великого обсягу розмітки, що ускладнює підтримку коду.
- Перформанс: у деяких випадках складні інтерфейси на XAML можуть мати проблеми з продуктивністю, особливо на старих або слабких пристроях.

Для створення інтерфейсу на платформі Android зазвичай використовують XML у поєднанні з Kotlin або Java. XML є стандартною мовою для опису UI на Android, що забезпечує гнучкість у створенні складних інтерфейсів та тісну інтеграцію з Android SDK. Однак, XML може вимагати великого обсягу коду для створення складних макетів і не завжди забезпечує сучасні можливості, які пропонують інші технології, такі як SwiftUI [11].

SwiftUI є сучасною мовою розмітки для розробки додатків на iOS, яка забезпечує декларативний підхід до опису інтерфейсу, що схоже на XAML. Вона тісно інтегрується з мовою програмування Swift і забезпечує простоту та сучасність у створенні інтерфейсів. Однак, SwiftUI працює лише на платформах Apple, що обмежує її використання для кросплатформних розробок. Крім того, SwiftUI є відносно новою технологією, тому має менше ресурсів та спільнот порівняно з більш зрілими технологіями.

HTML/CSS, у поєднанні з фреймворками, такими як React Native або Ionic, забезпечують можливість створення кросплатформних додатків. Ці технології використовують сучасні веб-технології для створення інтерфейсів, що дозволяє розробникам писати один код для різних платформ. HTML/CSS мають широку підтримку спільноти та великий обсяг ресурсів, проте можуть поступатися нативним рішенням за продуктивністю та ускладнювати процес налагодження додатків на різних платформах [11].

Flutter з використанням мови Dart є ще однією популярною технологією для кросплатформної розробки. Flutter забезпечує високу продуктивність, наближену до нативної, завдяки використанню власного рендерера. Він дозволяє

створювати додатки для Android, iOS та інших платформ, що забезпечує швидкий цикл розробки завдяки функції Hot Reload. Проте Flutter є відносно новою технологією, що має менше ресурсів та спільнот порівняно з більш зрілими технологіями, і може створювати великі файли додатків через включення власного рендерера [12].

У порівнянні з іншими мовами, такими як XML для Android, SwiftUI для iOS, HTML/CSS з фреймворками та Flutter з Dart, XAML виділяється своєю гнучкістю, інтеграцією з потужними інструментами .NET та можливістю розширення. Хоча XAML може мати певну складність у навчанні та можливі проблеми з продуктивністю на слабких пристроях, його переваги роблять його привабливим вибором для розробки користувацьких інтерфейсів на різних платформах.

## 2.3 Використання середовища розробки Android Studio

При розробці будь-якого програмного продукту, зокрема мобільних додатків, вибір середовища розробки впливає на продуктивність, ефективність та якість створеного продукту. Перед вибором середовища розробки для створення мобільного додатку на платформі Android варто ретельно проаналізувати різні аспекти, такі як зручність використання, підтримка потрібних технологій, можливості інтеграції, а також специфічні вимоги проекту.

Android Studio є офіційним середовищем розробки для платформи Android і має широкий набір інструментів для створення високоякісних мобільних додатків. Однією з його ключових переваг є підтримка мови програмування Kotlin, що забезпечує зручне та ефективне створення Android-додатків. Редактор коду Android Studio має спеціальні функції для підтримки Kotlin, такі як синтаксичне підсвічування, автозаповнення коду і рефакторинг, що сприяє прискоренню розробки. Це дозволяє розробникам писати безпечний та зрозумілий код для Android-додатків, зменшуючи кількість потенційних помилок і підвищуючи продуктивність [13].

Навпроти, XAML - це мова опису інтерфейсу користувача, широко використовується для створення інтерфейсів у платформі .NET, зокрема для додатків на платформі Windows. Хоча Android Studio не має вбудованої підтримки XAML, розробники можуть скористатися сторонніми бібліотеками, такими як Jetpack Compose, для використання XAML для створення інтерфейсів користувачів Android. Це дозволяє розробникам, знайомим з XAML, використовувати свої навички для створення добре структурованих інтерфейсів для Android-додатків [14].

При порівнянні різних середовищ розробки для створення Android-додатків можна розглядати кілька варіантів, кожен з яких має свої переваги та недоліки. Наприклад, Eclipse, IntelliJ IDEA та Visual Studio - кожен з цих інструментів має свої особливості і обмеження, і вибір серед них залежить від індивідуальних потреб та уподобань розробника. Незважаючи на це, Android Studio залишається одним з найбільш оптимальних інструментів для розробки Android-додатків, оскільки воно має всі необхідні компоненти та підтримку для ефективної розробки додатків для цієї платформи.

## 2.4 Технології блокування та надання доступу до додатків

### 2.4.1 Теоретична база

У мобільній операційній системі Android доступ до різних функцій та ресурсів пристрою, таких як камера, місцезнаходження, контакти, мікрофон і багато інших, регулюється системою дозволів. Ця система забезпечує користувачам контроль над тим, які додатки можуть отримувати доступ до їхніх особистих даних і функцій пристрою [15].

Механізм дозволів Android спроектований таким чином, щоб забезпечити безпеку і конфіденційність даних користувачів. Кожен тип дозволу (наприклад, доступ до камери чи місцезнаходження) пов'язаний з конкретною функцією або набором даних, до яких додаток може отримати доступ. При встановленні

додатку користувач зазвичай надає або відмовляє в доступі до цих дозволів. В разі відмови, додаток не може використовувати відповідний ресурс чи функцію.

Кожен запит на дозвіл підкріплюється обґрунтуванням використання даних або ресурсів, що дозволяє користувачам приймати обізнані рішення щодо надання доступу. Наприклад, додаток для соціальних мереж може потребувати доступу до камери для завантаження фотографій, або навігаційний додаток може потребувати доступу до місцезнаходження для навігації користувача. Користувачі можуть переглядати і змінювати дозволи для кожного додатку у налаштуваннях пристрою в будь-який час.

Такий підхід дозволяє забезпечити баланс між зручністю використання додатків і захистом конфіденційності особистих даних користувачів, що є важливим аспектом сучасних мобільних технологій [16].

Android Application Sandbox є важливим аспектом безпеки та стабільності операційної системи Android. Кожен додаток працює у власному віртуальному середовищі, яке відоме як "sandbox". Ця концепція ізоляції гарантує, що кожен додаток виконується в окремому обмеженому просторі, що ефективно запобігає можливості проникнення шкідливих програм в інші додатки або в саму операційну систему. В середовищі "sandbox" додатки не мають прямого доступу до системних ресурсів, таких як файли інших додатків або конфігураційні налаштування, які не є необхідними для їхньої роботи. Це підходить для забезпечення конфіденційності даних і захисту від вразливостей, що можуть бути експлуатовані зловмисниками. При цьому відокремлене виконання кожного додатка дозволяє забезпечити стабільність роботи операційної системи, оскільки проблеми в одному додатку не впливають на інші, що працюють в тому ж середовищі.

Permissions (дозволи) у мобільній операційній системі Android є важливим механізмом для контролю доступу додатків до системних ресурсів та приватних даних користувача. Кожен додаток, щоб отримати доступ до певних функцій або інформації, наприклад, камери, мікрофону, контактів, місцезнаходження чи інших системних ресурсів, повинен запитати від користувача відповідні дозволи [17].

Механізм permissions включає в себе деталізовану систему запитів та надання дозволів. Коли додаток вперше намагається отримати доступ до певного ресурсу, система Android показує користувачеві діалогове вікно, в якому зазначено, які ресурси або дані додаток хоче використовувати. Користувач може прийняти цей запит і надати необхідні дозволи, або ж відхилити його, що перешкодить додатку отримати доступ до вибраного ресурсу [17].

Надання дозволів залежить від впевненості користувача у необхідності цього доступу для нормальної роботи додатку. Користувачеві надається можливість детально оцінити, чи потрібно це дозвіл для функціонування програми, та відповідно до цього вирішити, чи дозволити додатку доступ до певного ресурсу чи інформації [18].

Після надання дозволу, додаток може безпечно використовувати відповідний ресурс або інформацію. Цей підхід дозволяє забезпечити прозорість та контроль за доступом до особистих даних користувача, що є критично важливим для забезпечення приватності і безпеки в мобільних додатках на Android.

Permissions Manager у мобільній операційній системі Android відіграє критичну роль у забезпеченні контролю та безпеки за дозволами, що надаються додаткам. Цей інструмент дозволяє користувачам переглядати та управляти дозволами для кожного встановленого додатка на пристрої.

За допомогою Permissions Manager, користувачі можуть легко налаштовувати, які системні функції та ресурси кожен додаток може використовувати. Це включає доступ до камери, мікрофону, контактів, місцезнаходження та інших чутливих даних. В меню "Налаштування" > "Додатки" > "Дозволи" користувач може переглянути всі встановлені додатки та їхні запити на дозволи.

Взаємодія з Permissions Manager дозволяє користувачам змінювати налаштування дозволів в будь-який час згідно з їхніми власними вимогами і зручності. Наприклад, користувач може надати доступ до камери одному додатку, а іншому заборонити цей доступ, щоб забезпечити приватність і безпеку своїх особистих даних.

Цей підхід є важливим аспектом в управлінні безпекою на Android, оскільки дозволяє зберігати контроль за тим, які дані можуть використовувати додатки і які функції можуть виконувати. Завдяки Permissions Manager, користувачі можуть знижувати ризик неправомірного використання особистих даних і зберігати контроль за своїм пристроєм.

Кожен дозвіл у Android може мати різні рівні, які визначають, який обсяг інформації може бути доступний для додатків. Наприклад, дозвіл на доступ до місцезнаходження може включати наступні рівні доступу [19]:

1. Загальне місцезнаходження: Додаток може отримати доступ лише до загального місцезнаходження користувача, такого як область або місто. Це може бути достатньо для багатьох додатків, які просто потребують інформацію про місце перебування користувача для надання місцевих послуг або контенту.
2. Точне місцезнаходження до метра: Цей рівень дозволу надає додатку доступ до деталізованого місцезнаходження користувача, що може бути використане для точної навігації, мапи або інших сервісів, де потрібна висока точність.

Різні рівні дозволів дозволяють користувачам забезпечувати безпеку своїх особистих даних, вибираючи, яку інформацію вони готові надати додаткам. Це також дозволяє забезпечити оптимальний баланс між функціональністю додатку і збереженням приватності.

Наприклад, користувач може дозволити додатку мапи отримувати точне місцезнаходження для навігації під час поїздок, тоді як іншим додаткам, наприклад, погоднім сервісам, може бути надано доступ лише до загального місцезнаходження для надання місцевої погодної інформації.

Android 11 та новіші версії впровадили концепцію тимчасових дозволів. Ця функція дозволяє користувачам надавати додаткам доступ до певних функцій або даних тимчасово. Наприклад, користувач може надати дозвіл на доступ до місцезнаходження для додатку лише на певний час, після якого дозвіл автоматично відкликається, коли додаток закривається або після встановленого

періоду неактивності. Це дозволяє зменшити ризик неповідомленого доступу додатків до особистих даних, коли вони більше не потребуються [18-19].

Далі, у версії Android 10 і вище, впроваджено запити дозволів "just-in-time". Це означає, що додатки запитують доступ до конкретних функцій або даних лише у момент, коли ці ресурси їм реально потрібні для роботи. Наприклад, додаток, який потребує доступу до камери лише для фотографування, не буде запитувати цей дозвіл під час встановлення, а лише в той момент, коли користувач вперше відкриє функціональність фотографування. Це дозволяє зменшити кількість зайвих запитів дозволів під час встановлення додатків, що підвищує зручність для користувачів і зменшує ризик надмірного збору даних.

Разом з тимчасовими дозволами та запитами "just-in-time", Android надає користувачам значний контроль над приватністю і безпекою їхніх особистих даних. Ці ініціативи сприяють покращенню інтерфейсу взаємодії між додатками і користувачами, забезпечуючи оптимальний баланс між функціональністю додатків і захистом особистих даних.

Play Protect і Google Play Store є ключовими компонентами мобільної платформи Android, спрямованими на забезпечення безпеки користувачів і захист від шкідливих програм [17-19].

Play Protect, інтегрована служба від Google, виконує низку функцій для захисту пристрою користувача. Основні можливості включають автоматичне сканування додатків на наявність шкідливого вмісту. Це важливо для запобігання інсталяції шкідливих програм, які можуть завдати шкоди пристрою або викрасти особисті дані. Після виявлення потенційно небезпечного програмного забезпечення Play Protect пропонує видалити його з пристрою користувача. Крім того, Play Protect автоматично перевіряє оновлення вже встановлених додатків, щоб забезпечити їхню актуальність і відповідність новітнім стандартам безпеки.

Google Play Store відіграє ключову роль у забезпеченні безпеки платформи, перш ніж додатки потраплять до кінцевих користувачів. Перед тим, як додатки дозволяється публікувати в магазині, вони проходять комплексну перевірку на відповідність стандартам безпеки. Цей процес включає аналіз коду додатків для

виявлення потенційно небезпечного або шкідливого вмісту. Якщо додаток не відповідає стандартам безпеки, він відхиляється або вимагається внесення змін перед публікацією. Такий підхід дозволяє зменшити ризики введення в обіг шкідливих програм і забезпечує захист користувачів під час використання додатків з Google Play Store.

### 2.4.2 Програмна база

Файл маніфесту (англ. manifest file) є важливим компонентом будь-якої Android програми. Це текстовий файл формату XML, який містить основну інформацію про додаток і визначає його параметри та конфігурацію. Маніфест не лише ідентифікує додаток системі Android, але й надає інструкції щодо його роботи, включаючи налаштування прав доступу, об'єм необхідних ресурсів і параметри середовища виконання [19].

Основні складові файлу маніфесту включають:

1. Ідентифікатор пакету (Package ID): Унікальний ідентифікатор, що визначає додаток на платформі Android. Він використовується системою для ідентифікації і управління додатками.
2. Версія додатку (Versioning): Інформація про версію програми, яка використовується для визначення, яка версія додатку є актуальною для оновлення та сумісності зі смартфоном.
3. Компоненти додатку (Application Components): Оголошення активностей (Activity), служб (Service), провайдерів вмісту (Content Provider) і приймачів (Broadcast Receiver), що входять до складу додатку. Кожен компонент має власні налаштування і параметри.
4. Дозволи (Permissions): Перелік дозволів, які додаток вимагає від користувача для доступу до різних функцій і ресурсів пристрою, таких як камера, мікрофон, мережа і т.д.
5. Конфігурація (Configuration): Налаштування, які впливають на роботу додатку, такі як мінімальна версія Android, розмір екрану, мовні налаштування тощо.

Користування файлом маніфесту в Android розпочинається з його створення у кореневій директорії проекту. Він має назву AndroidManifest.xml і автоматично генерується при створенні нового проекту в середовищі розробки Android Studio. Редагування маніфесту може виконуватися безпосередньо у текстовому редакторі або через спеціальний редактор маніфесту у середовищі Android Studio, що надає зручні інструменти для додавання і зміни параметрів.

Файл маніфесту є ключовим для правильної роботи додатку на платформі Android, оскільки він визначає всі необхідні конфігураційні дані, що дозволяють системі правильно ідентифікувати, управляти і взаємодіяти з програмою [20].

Основні елементи файлу маніфесту включають:

1. Елемент <manifest>: Це кореневий елемент, який містить загальні атрибути додатка, такі як пакет і версія.
2. Елемент <application>: Оголошує додаток і включає всі компоненти, що входять до його складу, такі як активності (activities), служби (services), провайдери вмісту (content providers) і приймачі (broadcast receivers).
3. Елемент <activity>: Визначає кожну окрему активність в додатку, яка представляє собою один екран з інтерфейсом користувача.
4. Елемент <service>: Вказує на наявність служби, яка може працювати в фоновому режимі незалежно від інтерфейсу користувача.
5. Елемент <provider>: Оголошує провайдера вмісту, який забезпечує доступ до даних додатку через інтерфейс Content Provider.
6. Елемент <receiver>: Визначає приймач, який обробляє повідомлення від системи чи інших додатків, такі як повідомлення про події або стан системи.
7. Елемент <intent-filter>: Використовується для визначення фільтра інтентів, які дозволяють компонентам додатку відповідати на спеціальні запити від інших додатків чи системи.

Основні компоненти для розробки та керування дозволами у мобільних додатках з використанням мови програмування Kotlin в середовищі Android Studio включають декілька ключових аспектів :

1. Запит дозволів: Для запиту дозволів у користувача використовується метод requestPermissions() з класу ActivityCompat. Цей метод приймає масив рядків, які визначають необхідні дозволи, та обробник результату, який спрацьовує після того, як користувач прийме або відхилить запит. Важливо звертатися до користувача за дозволом лише в тому випадку, якщо це необхідно для нормальної роботи додатку.
2. Перевірка дозволів: Для перевірки, чи має додаток певний дозвіл, використовується метод checkSelfPermission() з класу Context. Цей метод приймає рядок, який визначає дозвіл, і повертає константу PERMISSION_GRANTED або PERMISSION_DENIED. Важливо переконатися, що додаток не спробує отримати доступ до функцій або даних, на які він не має відповідного дозволу.
3. Реагування на результати запиту дозволів: У методі обробки результатів запиту дозволів (onRequestPermissionsResult()), потрібно перевіряти результат запиту і відповідно оновлювати користувацький інтерфейс або логіку додатку. Якщо користувач надав дозвіл, додаток може звернутися до відповідних функцій або даних. У випадку відмови дозволу, слід розглянути можливість запропонувати альтернативні дії або пояснити необхідність дозволу.
4. Використання бібліотеки AndroidX: AndroidX надає зручні бібліотеки, такі як androidx.core.content.ContextCompat та androidx.activity.result.ActivityResultLauncher, для спрощення роботи з дозволами. Ці бібліотеки містять методи, які спрощують процес запиту, перевірки та обробки результатів запиту дозволів, що робить код більш лаконічним і зрозумілим.

## 2.5 Висновки

В цьому розділі було здійснено опис та обґрунтування вибору оптимальних засобів та інформаційних технологій для вирішення поставленої задачі. А саме, обґрунтовано вибір мови програмування Kotlin та мови розмітки

XAML, розглянуті необхідні теоретичні засади щодо коректної роботи головної логіки додатку.

# 3 РОЗРОБЛЕННЯ ДОДАТОКУ ДЛЯ ЗАБЕЗПЕЧЕННЯ ОБМЕЖЕНОГО РЕЖИМУ ДОСТУПУ НА МОБІЛЬНОМУ ПРИСТРОЇ

## 3.1 Розроблення архітектури інформаційної системи

Архітектура інформаційної системи (ІС) – це високорівнева структурна концепція, яка визначає загальну структуру та поведінку інформаційної системи. Вона описує компоненти системи, їх функції, взаємозв'язки, а також правила та обмеження, які визначають, як ці компоненти взаємодіють один з одним. Основними елементами архітектури ІС є апаратне забезпечення, програмне забезпечення, бази даних, мережі та користувачі.

Зазвичай, зображення архітектури інформаційної системи за допомогою мови UML (Unified Modeling Language) є зручним з кількох причин:

1. UML є міжнародним стандартом для моделювання програмних систем. Це означає, що моделі, створені за допомогою UML, будуть зрозумілі всім, хто знайомий з цим стандартом.
2. UML надає різноманітні діаграми, які дозволяють візуально представити різні аспекти системи. Це допомагає зрозуміти складні структури та процеси, полегшуючи спілкування між розробниками, аналітиками та замовниками.
3. UML підходить для моделювання як програмних, так і апаратних компонентів системи, а також бізнес-процесів і вимог. Це робить UML універсальним інструментом для різних етапів розробки ІС.
4. UML дозволяє створювати моделі на різних рівнях абстракції, від загальних оглядів системи до детальних описів окремих компонентів. Це сприяє гнучкості та можливості адаптації моделі до потреб різних аудиторій.
5. Існує багато програмних інструментів для створення та аналізу UML-діаграм, що полегшує розробку, документування та підтримку інформаційних систем.

UML (Unified Modeling Language) – це стандартна мова для створення моделей програмного забезпечення. Вона була розроблена для допомоги у специфікації, візуалізації, конструюванні та документуванні компонентів систем. UML включає 14 різних типів діаграм, які можуть бути використані для представлення статичних структурувань, динамічних процесів і взаємодій у системі.

Використання UML для моделювання архітектури інформаційної системи дозволяє створити чітке та зрозуміле уявлення про систему, її компоненти та взаємозв'язки між ними, що є важливим для успішної розробки, впровадження та підтримки ІС.

Розглянемо першу з діаграм, яка надає розуміння варіантів використання додатку та його основні інтерфейси.Діаграма Use Case описує функціональні вимоги до системи через взаємодії з користувачами. Вона ілюструє, як різні типи користувачів взаємодіють із системою для виконання певних завдань або випадків використання .

У нашому випадку дорослий користувач є основним і єдиним користувачем, оскільки дитина використовує наявні додатки в системі без прямої взаємодії з функціями самого додатку. Додаток створює бар'єр для використання виключно дозволених додатків, тому на діаграмі зображений лише дорослий користувач. Згідно з рисунку 3.1, користувач, відкривши додаток, одразу потрапляє на головний екран, звідки можна перейти до налаштувань або запустити дозволений додаток. З екрану налаштувань користувач може налаштувати метод блокування або обрати додатки, які будуть виводитися на головний екран.

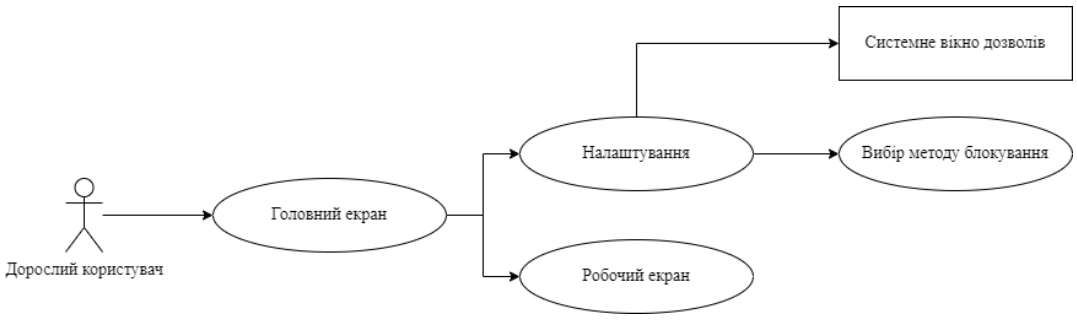


Рисунок 3.1 – Діаграма use case

Тепер зобразимо основний сценарій використання додатку та процеси, які можуть бути задіяні. На рис. 3.2 зображено sequence diagram для відображення порядку взаємодії між об'єктами в певному сценарії. Вони показують, як об'єкти взаємодіють один з одним через обмін повідомленнями або виклики методів.

Основними процесами є налаштування доступних додатків, налаштування паролів та процес використання додатку, перевірка введеного паролю для виходу або ні з додатку.

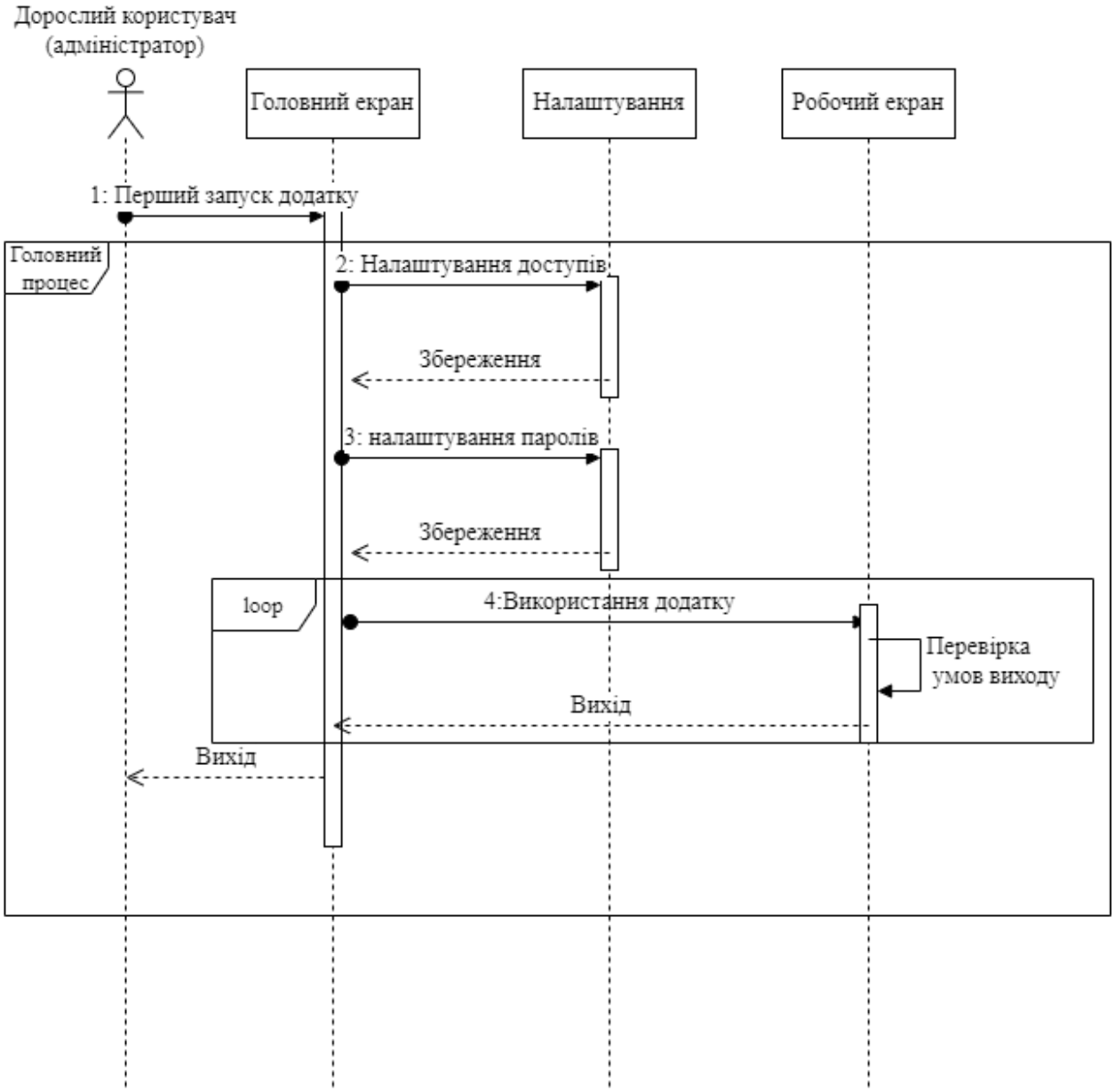


Рисунок 3.2 – Діаграма взаємодії

Діаграма станів — це тип діаграми, який використовується для представлення можливих станів об'єкта або системи протягом його життєвого циклу, а також переходів між цими станами, що відбуваються у відповідь на події або умови. Вона допомагає зрозуміти, як об'єкт реагує на зовнішні впливи та які дії виконуються при зміні стану. На рис. 3.3 зображено можливі стани використання додатку:

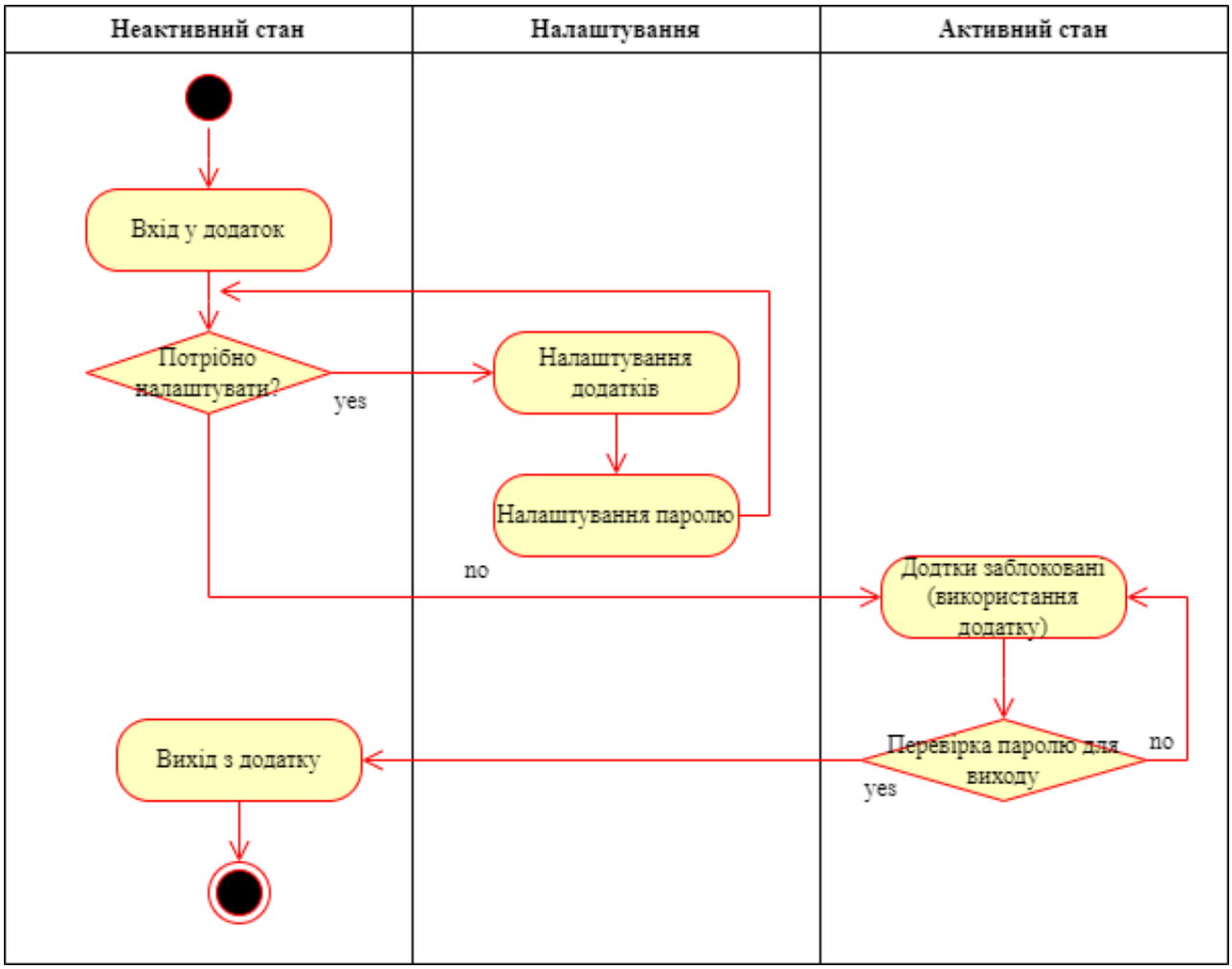


Рисунок 3.3 – Діаграма варіантів використання

У неактивному стані на екрані користувача відображається головна сторінка додатку. У майбутньому це також буде сторінка, де користувачі зможуть переглядати оновлення версій додатку. У стані налаштувань користувач має можливість змінити перелік додатків, які будуть доступні в активному стані, а також вибрати метод захисту. Варіанти захисту включають математичний

приклад, заздалегідь заданий пароль або вбудовані методи захисту телефону, такі як відбиток пальця, фейс-контроль, графічний ключ або пароль.

Саме через подібний функціонал у даному додатку буде доволі проста система класів, зображена на рис. 3.4.

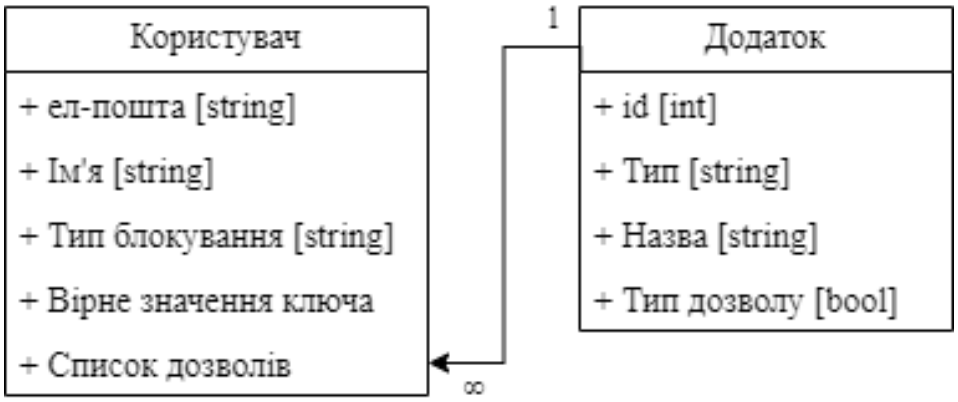


Рисунок 3.4 – Діаграма класів

Клас «користувач» необхідний для збереження даних про користувача для майбутнього відновлення використання або надсилання повідомлень про спробу виходу з додатку або зміну налаштувань. Також для скиду паролю або його відновлення буде використано саме електронну пошту та ім’я користувача.

Тип блокування визначає, що саме додаток буде використовувати для перевірки, а поле «вірне значення ключа» зберігає зашифрований правильний варіант для виходу з активного стану додатку.

Клас «додаток» відповідає за сортування всіх додатків на телефоні. Поле «тип» не дозволить надати дозвіл використання на системні додатки або додатки, що мають особливі дозволи (банківські додатки, контакти, телефонні повідомлення, електронна пошта та ін.)

Поле «назва» відповідає за інформацію додатку, яка зберігається на телефоні, а саме підпис під іконкою та іконка додатку.

«Тип дозволу» Встановлюється користувачем, а саме – значення “True” для додатку, який буде видно на активному екрані додатку та“False” – для заборони.

Окремо створено статичний клас для роботи з дозволами додатків: якщо доступ наявний, клас створює ярлик додатку на активному екрані.

## 3.2 Програмна реалізація мобільного додатку

### 3.2.1 Навігація додатку

Android Studio, як офіційне середовище розробки для Android, пропонує різні підходи та інструменти для реалізації навігації в додатках. Одним із основних способів реалізації навігації є використання Navigation Component — бібліотеки з набору інструментів Android Jetpack, яка значно спрощує реалізацію навігації між фрагментами та активностями в додатку.

Navigation Component забезпечує безпечну та гнучку навігацію за допомогою таких основних компонентів:

1. NavHostFragment — контейнер, який управляє навігацією між фрагментами. Він містить NavController і відповідає за заміну фрагментів у додатку.
2. NavController — об'єкт, який обробляє навігаційні команди. Він управляє навігацією у NavHostFragment і може запускати різні навігаційні дії, такі як переходи між фрагментами, обробка повернення назад та навігаційні переходи за допомогою навігаційних графів.
3. Navigation Graph (nav_graph) — XML файл, який визначає всі можливі маршрути і взаємозв'язки між фрагментами та активностями у додатку. Він дозволяє розробникам візуально проектувати навігаційні потоки.

Navigation Component має такі особливості:

- Спрощене управління переходами між фрагментами: Забезпечує інтуїтивно зрозумілий спосіб управління переходами, використовуючи NavController та навігаційні графи.
- Підтримка анімацій і транзакцій: Навігаційний компонент підтримує анімації при переходах між фрагментами, що покращує користувацький досвід.

- Інтеграція з Safe Args: Safe Args — це плагін для Gradle, який генерує типобезпечний код для передачі даних між фрагментами, зменшуючи ризик помилок, пов'язаних з неправильними типами даних.

Розглянемо й інші методи створення навігації додатку. Використання Intent для переходів між активностями є одним із найстаріших і найпростіших методів реалізації навігації в Android-додатках . Основною перевагою цього підходу є його простота. Для створення переходу між двома активностями достатньо створити новий об'єкт Intent, вказати цільову активність і викликати метод startActivity().

Такий підхід ідеально підходить для невеликих додатків, де кількість екранів обмежена, і навігаційні вимоги не є складними. Використання Intent дозволяє швидко і легко налаштувати навігацію без необхідності писати значний обсяг коду.

Однак, цей метод має свої недоліки. По-перше, при великій кількості переходів між активностями виникає складність в управлінні станом додатка. Відсутність вбудованої підтримки для фрагментів призводить до того, що розробникам доводиться створювати окремі активності для кожного екрану, що збільшує складність і обсяг коду.

Іншим поширеним методом навігації є використання Fragment Transactions для управління фрагментами. Цей підхід надає більшу гнучкість у реалізації складних переходів та анімацій. Розробники можуть точно контролювати стан кожного фрагмента, створювати анімації переходів, а також додавати фрагменти до back stack для підтримки навігації назад.

Перевагою цього підходу є можливість створення більш динамічних і багатих користувацьких інтерфейсів. Однак, цей метод також має свої недоліки. Основною складністю є необхідність написання значного обсягу коду для управління навігацією та обробки кнопки "Назад". У порівнянні з Navigation Component, використання Fragment Transactions вимагає більше зусиль для підтримки і розвитку коду.

На основі порівняння різних методів навігації можна зробити висновок, що для додатків, які не мають надмірно складних навігаційних шляхів, використання Navigation Component є найкращим вибором. Цей підхід забезпечує простоту у реалізації та підтримці навігації, дозволяє легко управляти переходами між фрагментами та активностями, підтримує анімації і транзакції, а також інтегрується з Safe Args для безпечної передачі даних. У порівнянні з використанням Intent та Fragment Transactions, Navigation Component надає більш структуроване та зручне рішення для розробників, спрощуючи процес створення та управління навігацією в Android-додатках.

Початковий етап включає створення XML-файлу, який буде використовуватися для визначення всіх можливих маршрутів і взаємозв'язків між фрагментами та активностями. Цей файл створюється в каталозі res/navigation і зазвичай називається nav_graph.xml.

Наступним кроком є додавання фрагментів, які будуть використовуватися в додатку, до навігаційного графа. Кожен фрагмент визначається як окремий елемент у XML-файлі і отримує унікальний ідентифікатор. Ці фрагменти представляють окремі екрани додатка.

Для початку спроектуємо всі майбутні фрагмент екрану та основні кнопки. Почнемо з домашнього екрану: він містить кнопки, що ведуть до блокування інших додатків та до налаштувань.

На рис 3.5 зображено найпершу сторінку, яку буде бачити уористувач під час використання присрою.

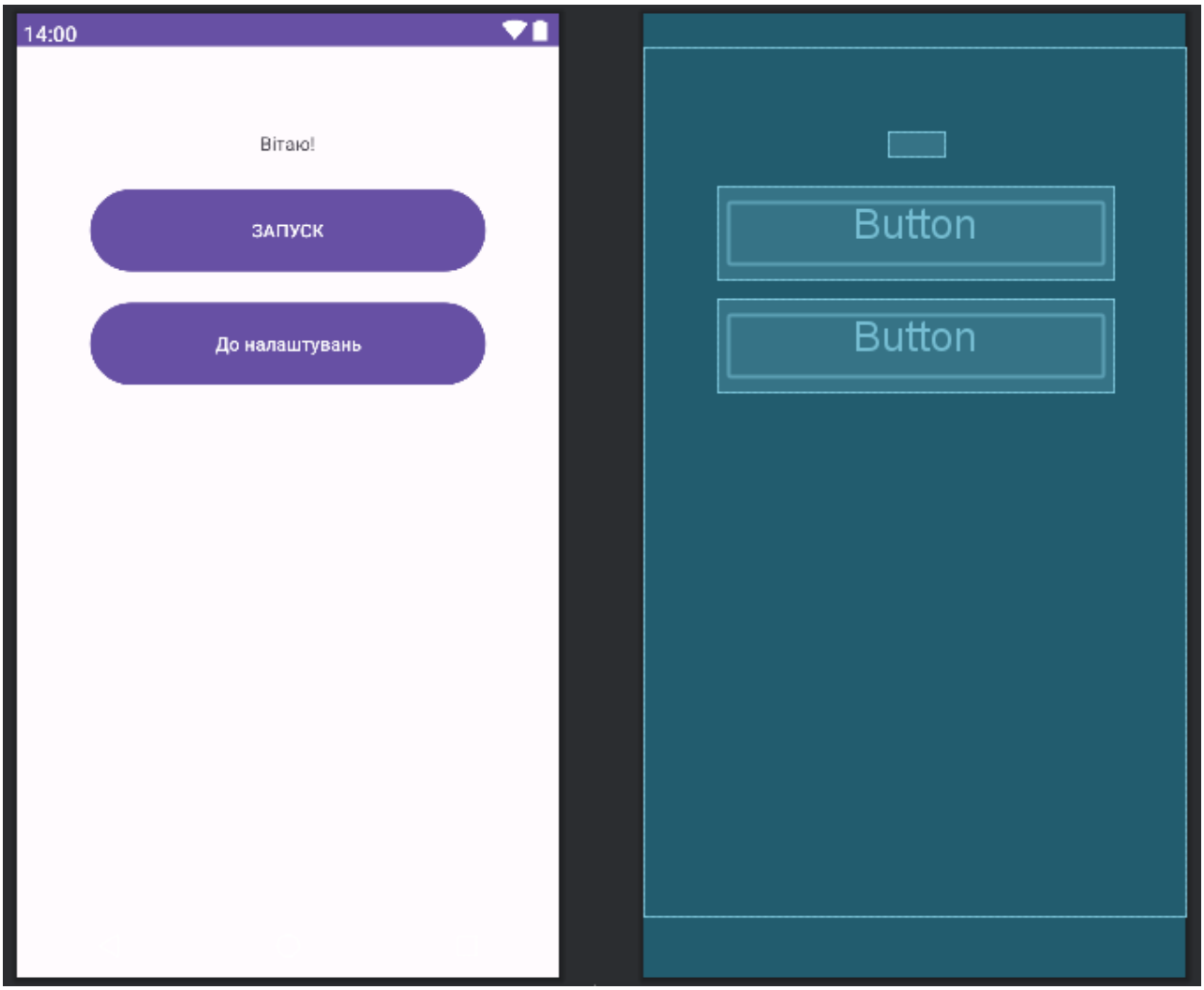


Рисунок 3.5 – Домашня сторінка, найперша сторінка

Сформуємо найпростішу – сторінку, де будуть відображатись майбутні ігри і додатки, яким надано доступ (рис. 3.6).

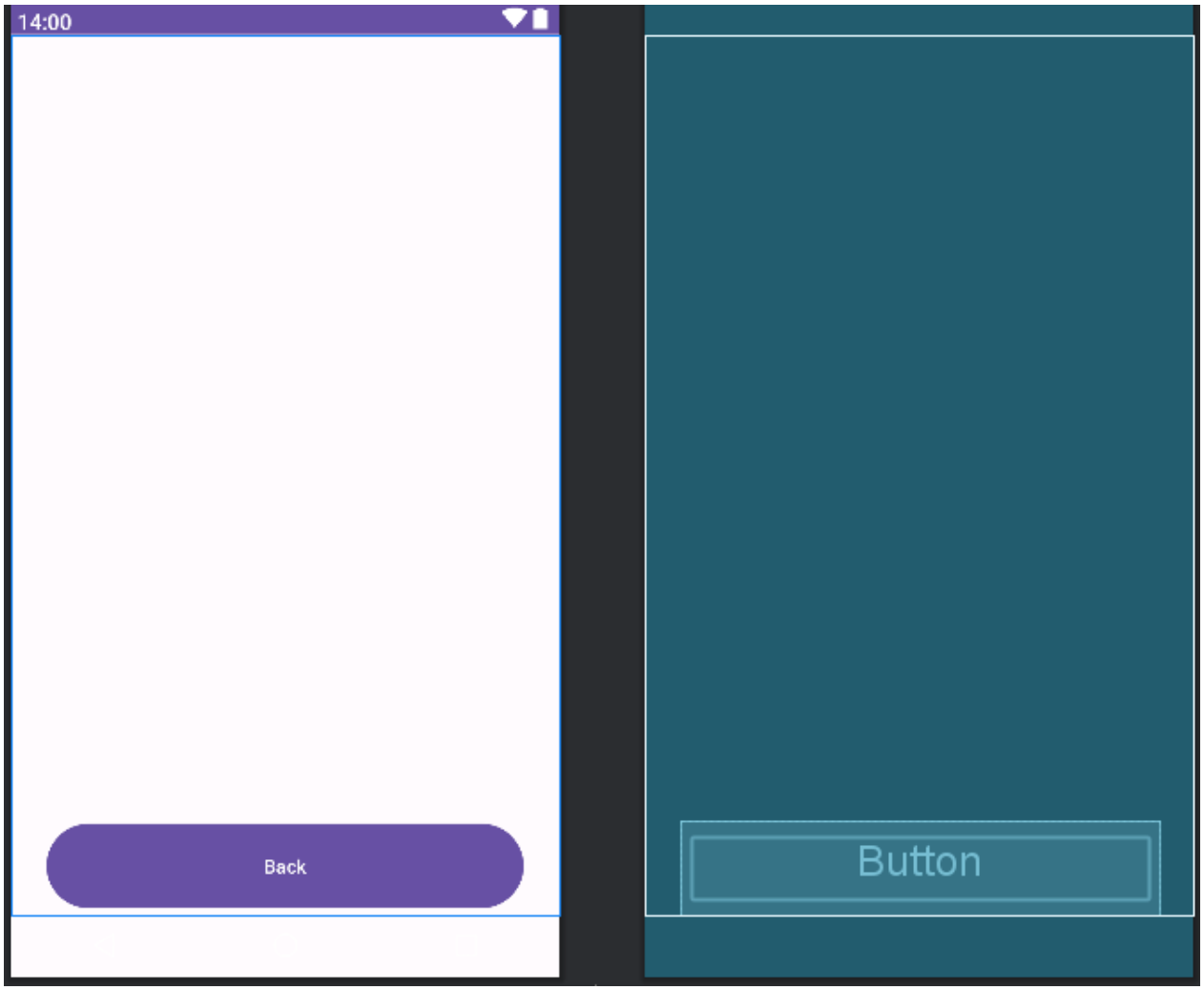


Рисунок 3.6 – Вікно, на якому відображатимуться додатки

Сторінка налаштувань матиме багато розширень та покращень у майбутньому (рис 3.7). Кнопка «Споріднений пристрій» міститиме можливість зміни адрес електронної пошти користувача, на який надсилатиметься повідомлення про спробу залишити додаток або вдалий вихід з нього.

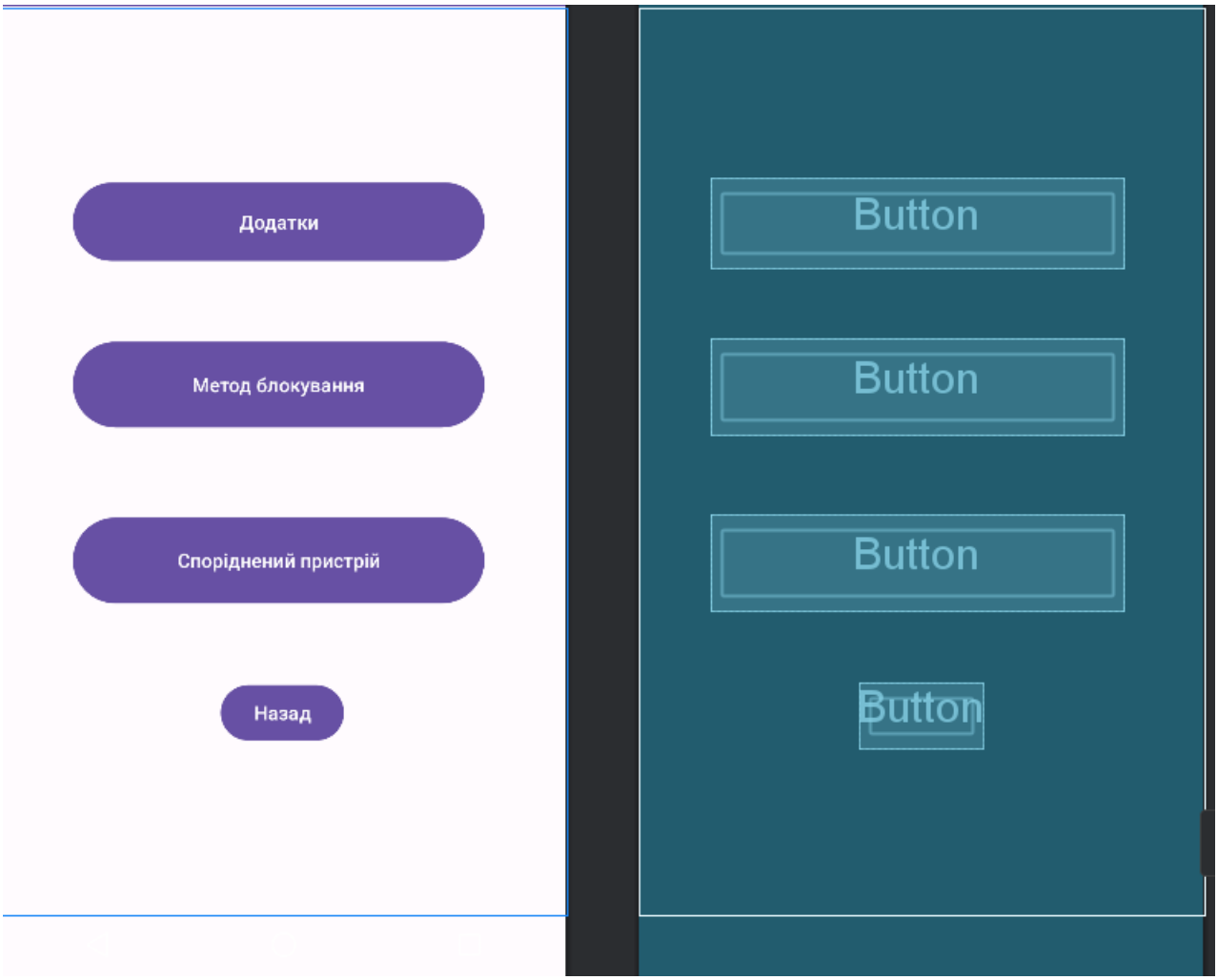


Рисунок 3.7 – Сторінка налаштування

При натисканні кропки «Метод блокування» відкривається вікно для зміни методу длокувань та самого паролю (рис. 3.8)

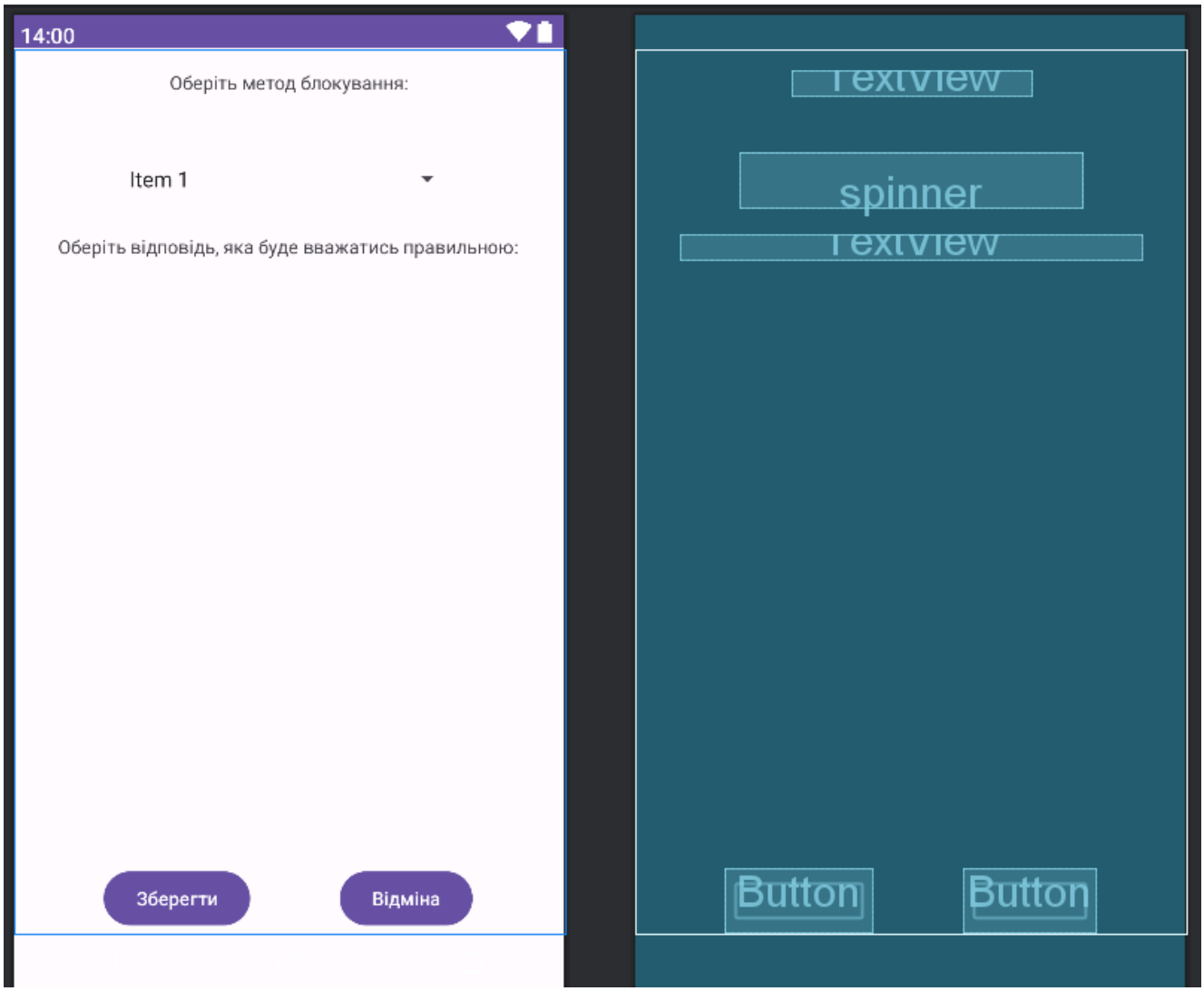


Рисунок 3.8 – Сторінка налаштування блокування

Далі налаштовуємо зв’язки між фрагментами, які будуть далі прив’язані до кнопок (рис 3.9)

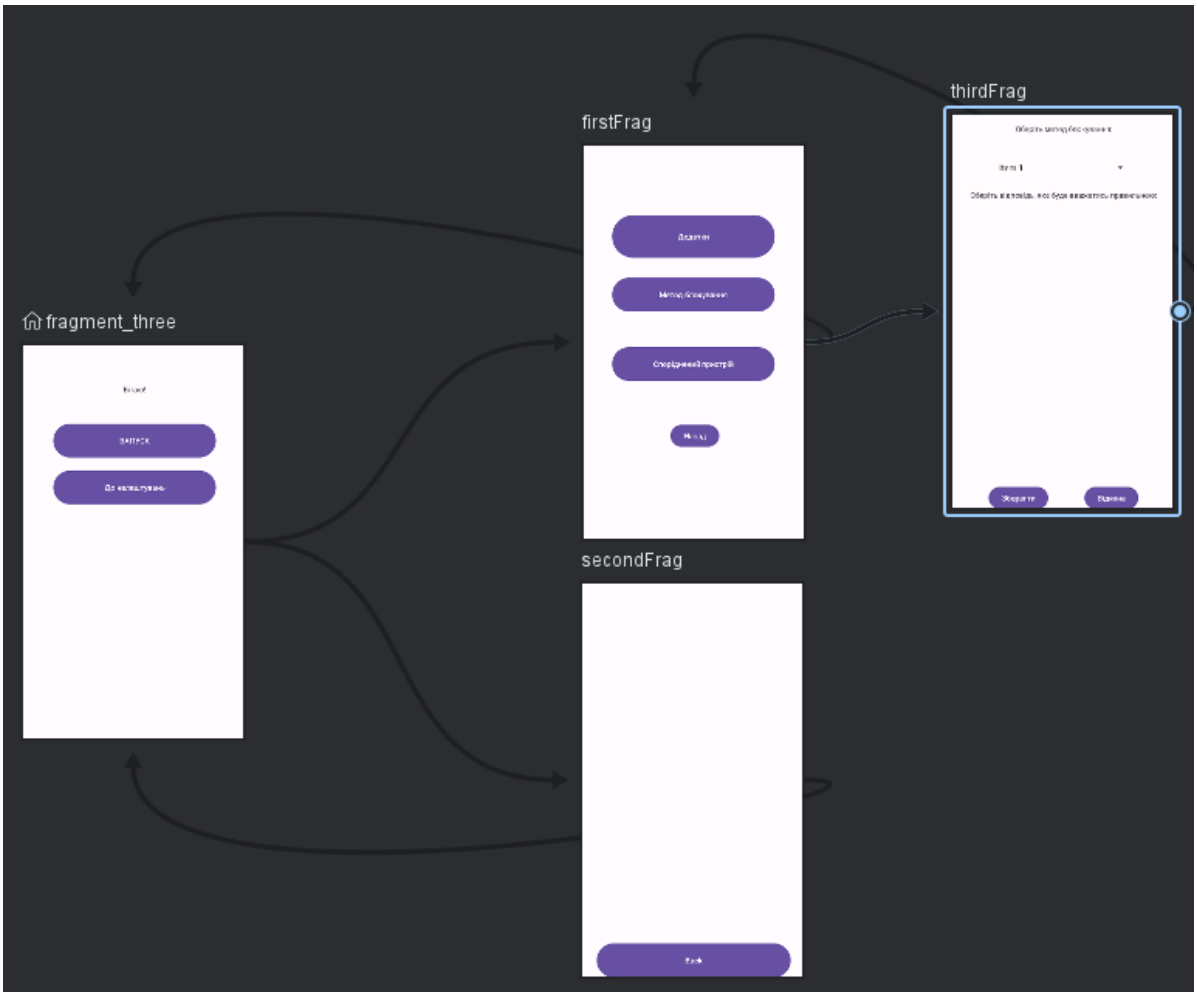


Рисунок 3.9 – Загальний вигляд навігації

Після додавання фрагментів необхідно визначити дії, які будуть виконуватися для переходу між фрагментами. Дії додаються як елементи в XML-файлі і вказують, який фрагмент є початковим і до якого фрагмента відбувається перехід. Дії також можуть включати анімації переходу та інші параметри.

На цьому етапі налаштовуємо параметри переходу між фрагментами. Це може включати передачу даних між фрагментами на більш пізніх етапах розробки за допомогою аргументів (arguments), а також визначення анімацій, які будуть використовуватися під час переходу.

Для того щоб навігаційний граф працював, додаємо NavHostFragment до макета головної активності. NavHostFragment є контейнером для фрагментів і відповідає за керування навігацією між ними. У цьому фрагменті потрібно вказати посилання на створений XML-файл навігаційного графа.

Завершальним етапом є тестування реалізованої навігації. Необхідно перевірити всі можливі маршрути переходів, роботу кнопки "Назад" та передачу даних між фрагментами. Це дозволяє переконатися, що навігація працює коректно і користувачі отримують очікуваний досвід.

На рис. 3.10 показано приклад програмного налаштування зв’язків між фрагментами.

```
package com.example.myapplication

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import com.example.myapplication.databinding.FragmentSecondBinding


class SecondFrag : Fragment() {

    lateinit var binding: FragmentSecondBinding
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        binding = FragmentSecondBinding.inflate(layoutInflater, container, attachToParent: false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        binding.back2.setOnClickListener { it: View!
            MAIN.navController.navigate(R.id.action_secondFrag_to_fragment_three)
        }


    }
}
```

Рисунок 3.10 – Частина написання звязку між фрагментами

### 3.2.2 Manifest

Для забезпечення коректної роботи додатку, який створює ярлики програм, необхідно внести такі зміни до файлу маніфесту (AndroidManifest.xml):

1. Додати дозвіл на INSTALL_SHORTCUT. Приклад коду зображено на рис. 3.11.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT" />
```

Рисунок 3.11 – Дозвіл на SHORTCUT

2. Додати дію для створення ярлика. Приклад коду зображено на рис. 3.12.

```xml
        <activity
            android:name=".MainActivity"
            android:exported="true">

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
                <action android:name="com.android.launcher.action.INSTALL_SHORTCUT" />
                <category android:name="android.intent.category.DEFAULT" />


            </intent-filter>
```

Рисунок 3.12 – Створення дії додання ярлика

3. Додати метадані для ярлика. Приклад коду зображено на рис. 3.12.

```xml
            </intent-filter>
            <meta-data
                android:name="android.launcher.ICON"
                android:resource="@drawable/ic_launcher" />
            <meta-data
                android:name="android.launcher.NAME"
                android:value="@string/app_name" />

        </activity>
    </application>
```

Рисунок 3.13 – Додання метаданих для ярлика

Один із ключових аспектів розробленого додатку полягає в тому, що він не запитує доступу до персональних даних, таких як камера, контакти чи галерея. Це було свідомо враховано на етапі проектування з метою забезпечення максимальної безпеки та конфіденційності користувачів. Оскільки функціонування додатку не потребує доступу до таких чутливих даних, користувачі можуть бути впевнені в тому, що їх особиста інформація залишиться захищеною. Таким чином, використання цього додатку є цілком безпечним, що робить його зручним інструментом для перенесення ярликів програм без будь-яких ризиків для приватності.

### 3.2.3 Клас GameShortcutHelper.kt

Для реалізації функціоналу створення ярликів ігор на екрані додатку, обираємо та використовуємо бібліотеки Activity Shortcuts API та PermissionChecker.

Activity Shortcuts API є однією з бібліотек, що входять до складу Android SDK. Ця бібліотека була представлена з версією Android 7.1 (API Level 25) і надає розробникам можливість створювати ярлики для активностей безпосередньо з їх додатків .

Перша версія, в якій було представлено Activity Shortcuts API це Android 7.1 (Nougat, API Level 25)**.** Вона дозволяла розробникам створювати три типи ярликів: статичні ярлики, динамічні ярлики та ярлики, що закріплюються.

- Статичні ярлики: Визначаються в маніфесті додатку і завжди доступні користувачам.
- Динамічні ярлики: Додаються та видаляються програмно під час виконання додатку.
- Ярлики, що закріплюються: Додаються користувачем до домашнього екрану, зберігаючи стійкість до змін додатку.

В Android 8.0 (Oreo, API Level 26) було покращено підтримку динамічних ярликів та додано нові методи для більш гнучкого керування ними.

Activity Shortcuts API продовжували вдосконалювати, додаючи нові можливості для покращення інтеграції ярликів у робочий процес користувача.

**Особливості Activity Shortcuts API**:

- Можливість швидкого доступу до основних функцій додатку.
- Підвищення зручності користування додатком шляхом надання швидких дій з домашнього екрану.
- Забезпечення гнучкості в управлінні ярликами, що дозволяє адаптуватися до потреб користувачів.

PermissionChecker є частиною бібліотеки підтримки Android (Android Support Library), що пізніше стала частиною AndroidX. Ця бібліотека спрощує процес перевірки та запиту дозволів, необхідних для роботи додатку, особливо після впровадження моделі дозволів у Android 6.0 (Marshmallow, API Level 23).

В Android 6.0 (Marshmallow, API Level 23) було введено нову модель дозволів, яка вимагала від додатків запитувати дозволи під час виконання, а не під час встановлення.

- Це було зроблено для того, щоб надати користувачам більше контролю над своїми даними.
- Додатки тепер повинні були перевіряти, чи надані необхідні дозволи, і, якщо ні, запитувати їх у користувача.

PermissionChecker була інтегрована до бібліотеки підтримки Android, щоб спростити процес перевірки дозволів.

Зміна структури Android Support Library до AndroidX забезпечила кращу модульність та управління версіями бібліотек. PermissionChecker стала частиною AndroidX і продовжила вдосконалюватися для підтримки нових версій Android.

Особливості PermissionChecker:

- Проста перевірка, чи надано необхідні дозволи.
- Легке запитування дозволів у користувача, коли це необхідно.
- Підтримка зворотної сумісності зі старими версіями Android, що робить бібліотеку універсальною для розробників.

На рис 3.14 наведено приклад коду для створення ярлика гри.

```
class GameShortcutHelper {

    companion object {

        private val activityShortcuts = ActivityShortcuts.getInstance(context)
        private val permissionChecker = PermissionChecker(context)

        fun createGameShortcut(gamePackageName: String, gameIcon: Drawable) {
            if (permissionChecker.checkSelfPermission(Manifest.permission.READ_EXTERNAL_STORAGE) == Perm
                // Перевіряємо, чи має гра доступ до особистих даних
                if (!hasGameAccessToPersonalData(gamePackageName)) {
                    // Створюємо ярлик
                    val shortcutInfo = ShortcutInfo.Builder(context, gamePackageName)
                        .setShortLabel(getGameName(gamePackageName))
                        .setIcon(gameIcon)
                        .setIntent(Intent(Intent.ACTION_VIEW, Uri.parse("package:$gamePackageName")))
                        .build()

                    activityShortcuts.addShortcut(shortcutInfo)
                }
            }
        }
```

Рисунок 3.14 – Використання бібліотек для створення ярликів ігор

### 3.2.4 Діалогові вікна

Діалогові вікна в програмуванні є спеціальними вікнами, які використовуються для взаємодії з користувачем. Вони використовуються для показу інформації, запитів на підтвердження або введення даних в контексті поточного додатка. Діалогові вікна можуть містити текстові повідомлення, кнопки дій, власні макети зі спеціалізованими елементами і багато іншого.

Основні типи діалогових вікон:

1. AlertDialog: Це один з найпоширеніших типів діалогових вікон в Android. Використовується для показу повідомлень з кнопками "OK" і "Cancel", вибору опцій або спливаючих підказок.
2. ProgressDialog: Використовується для показу прогресу завантаження або виконання довгих операцій з можливістю відміни.
3. DatePickerDialog і TimePickerDialog: Використовуються для вибору дати і часу відповідно.
4. BottomSheetDialog: Відображається знизу екрана і може містити власні макети з кнопками або списками вибору.

5. Custom Dialogs: Власні діалогові вікна, які створюються на основі власних XML-макетів і можуть містити будь-яку кількість елементів інтерфейсу користувача.

   Спроектуємо головні діалогові вікна (рис 315-3.16).

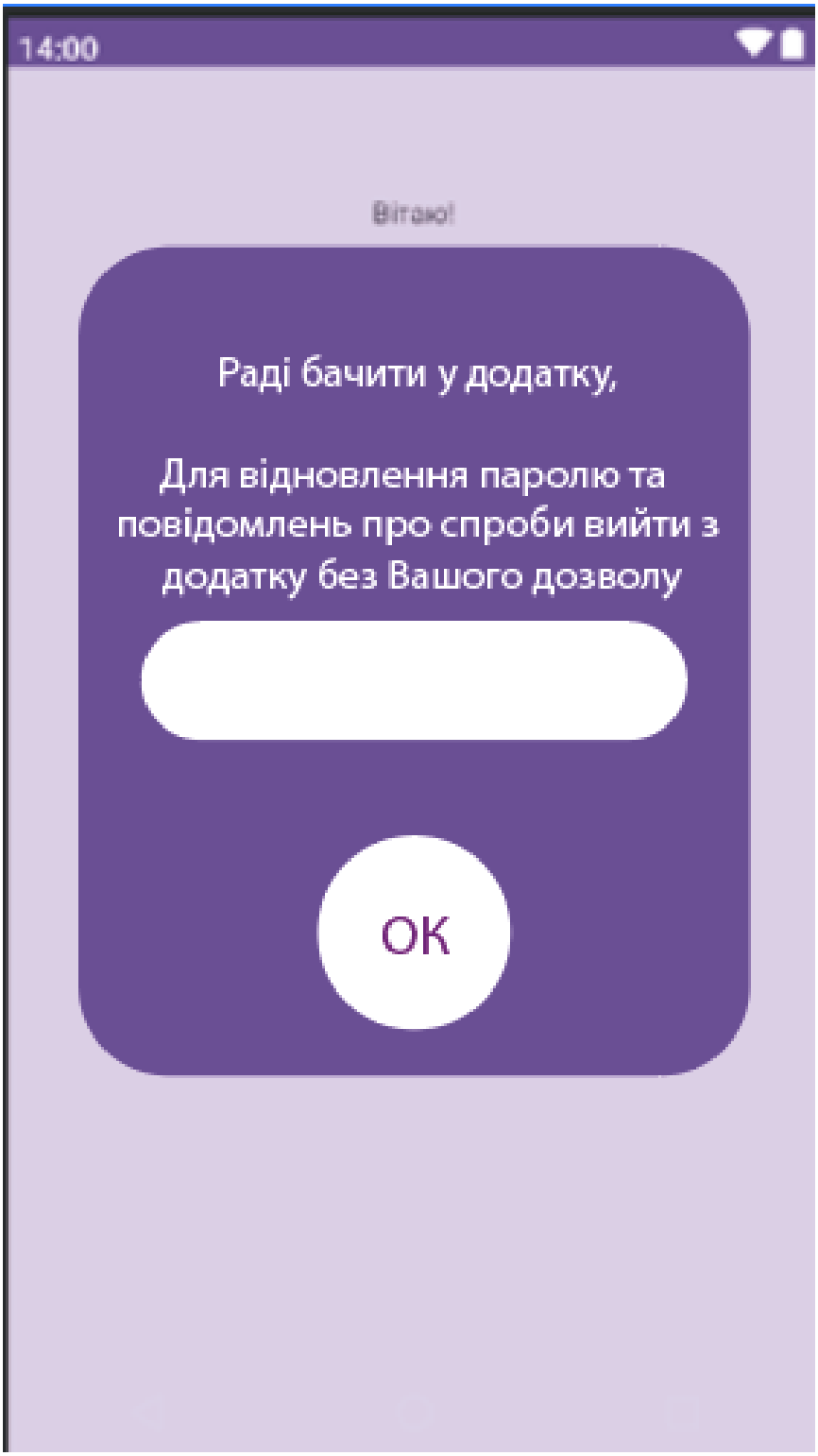


Рисунок 3.15 – Вітальне діалогове вікно

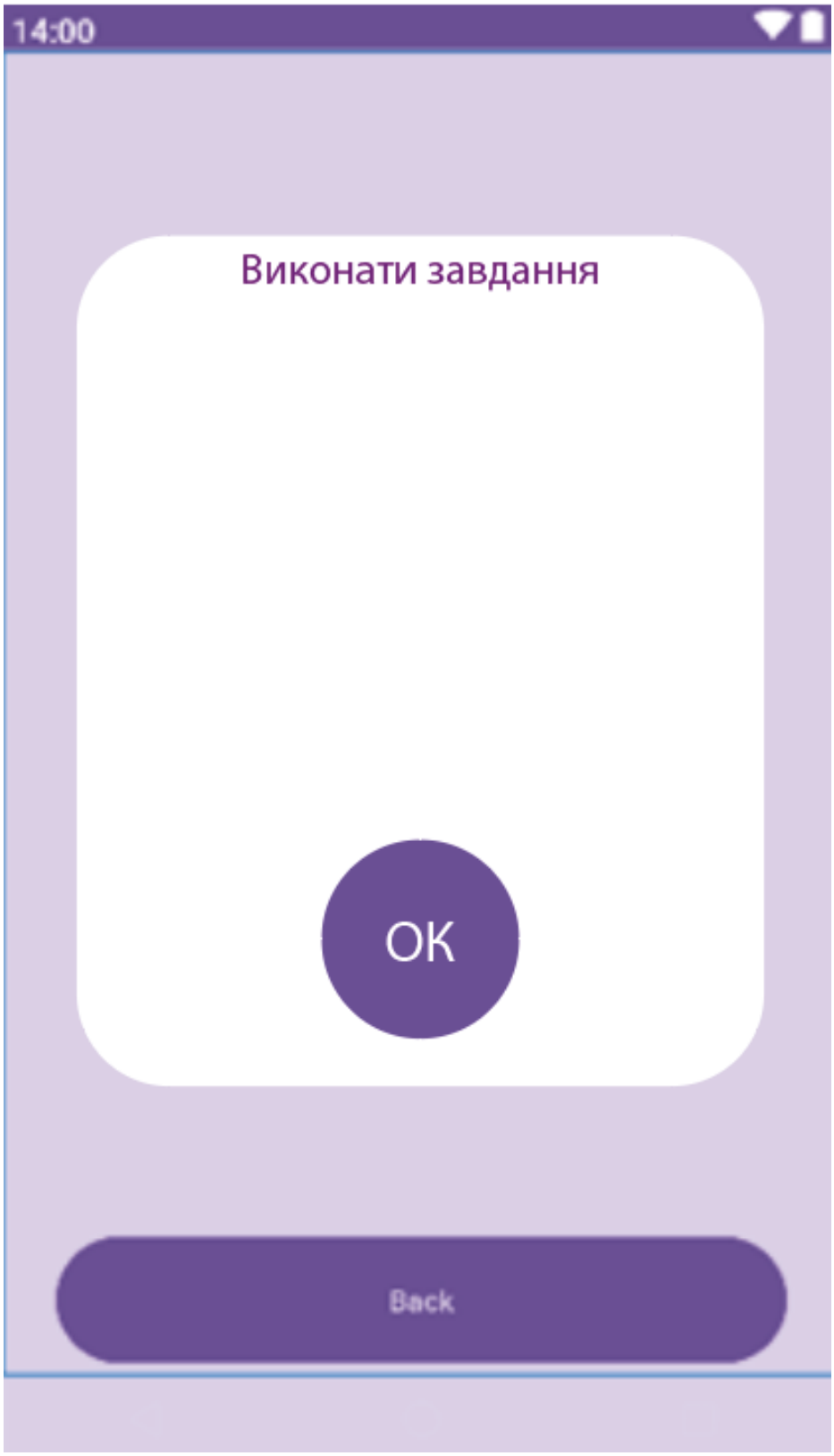


Рисунок 3.16 – Діалогове вікно для виходу з додатку

Обране завдання розміщуватиметься на білому фоні діалогового вікна. Кнопка та напис залежать від обраного методу блокування.

## 3.3 Тестування роботи системи

Розробимо дизайн ярлику додатку (рис.3.17).

Рисунок 3.17 – Оновлений ярлик додатку

Почнемо тестування додатку. При першому вході в додаток нас вітає діалогове вікно, яке запитує пошту користувача (рис. 3.18). Далі у всіподальші рази входу в додаток користувач буде бачити головне вікно додатку (рис. 3.19)

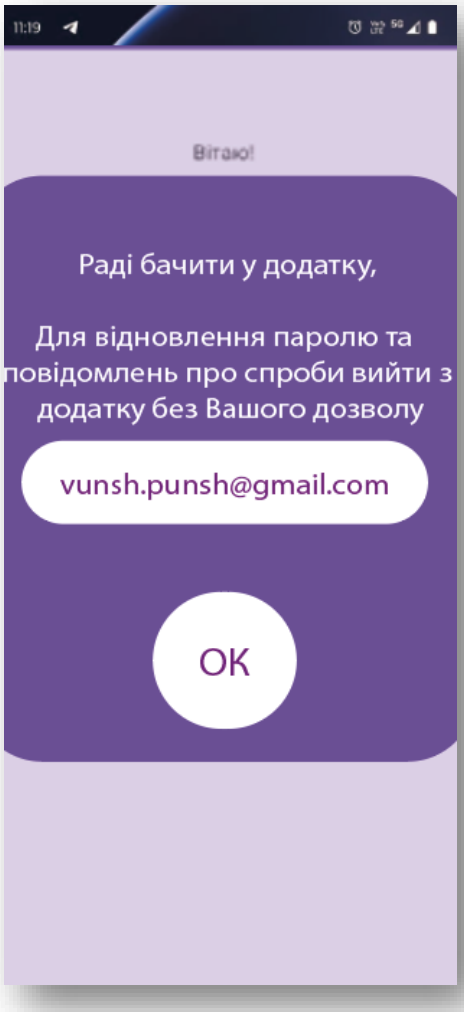


Рисунок 3.18 – Перший вхід в додаток

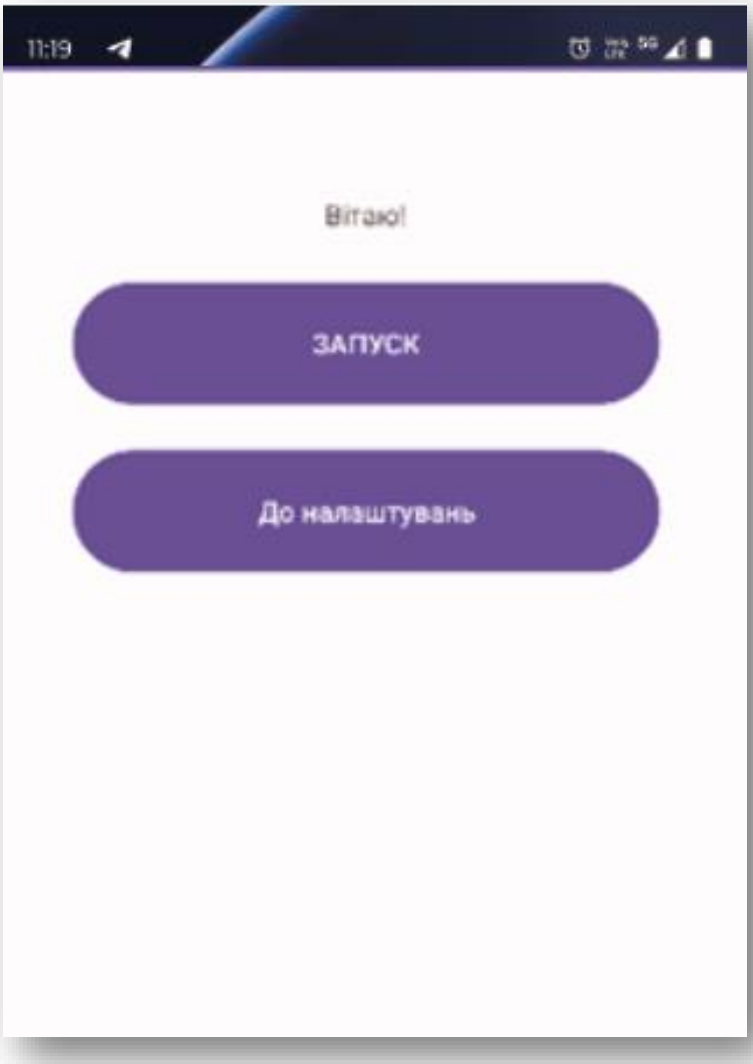


Рисунок 3.19 – Домашня сторінка

Натиснувши кнопку «Запуск» Користувач потрапить на активний екран, де згідно налаштуванням буде видно додатки, яким надано дозвіл (рис 3.20).


Рисунок 3.20 – Відображення додатків

При спробі вийти з додатку, виникатиме вікно, яке запитає пароль або ж дасть математичне завдання. Приклади перевірок на вихід з додатку зображено на рис 3.21-3.23.

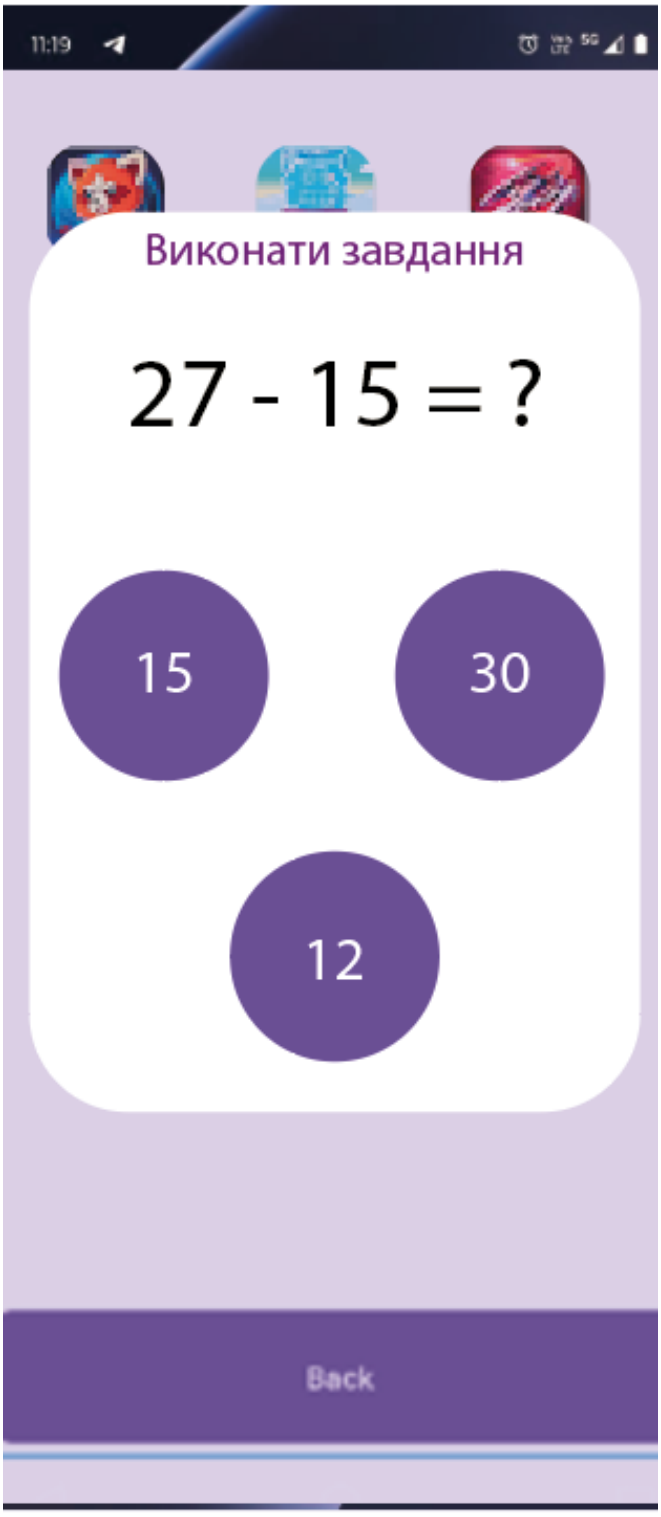


Рисунок 3.21 – Діалогове вікно для виходу з додатку.

Варіант блокування – математичний приклад

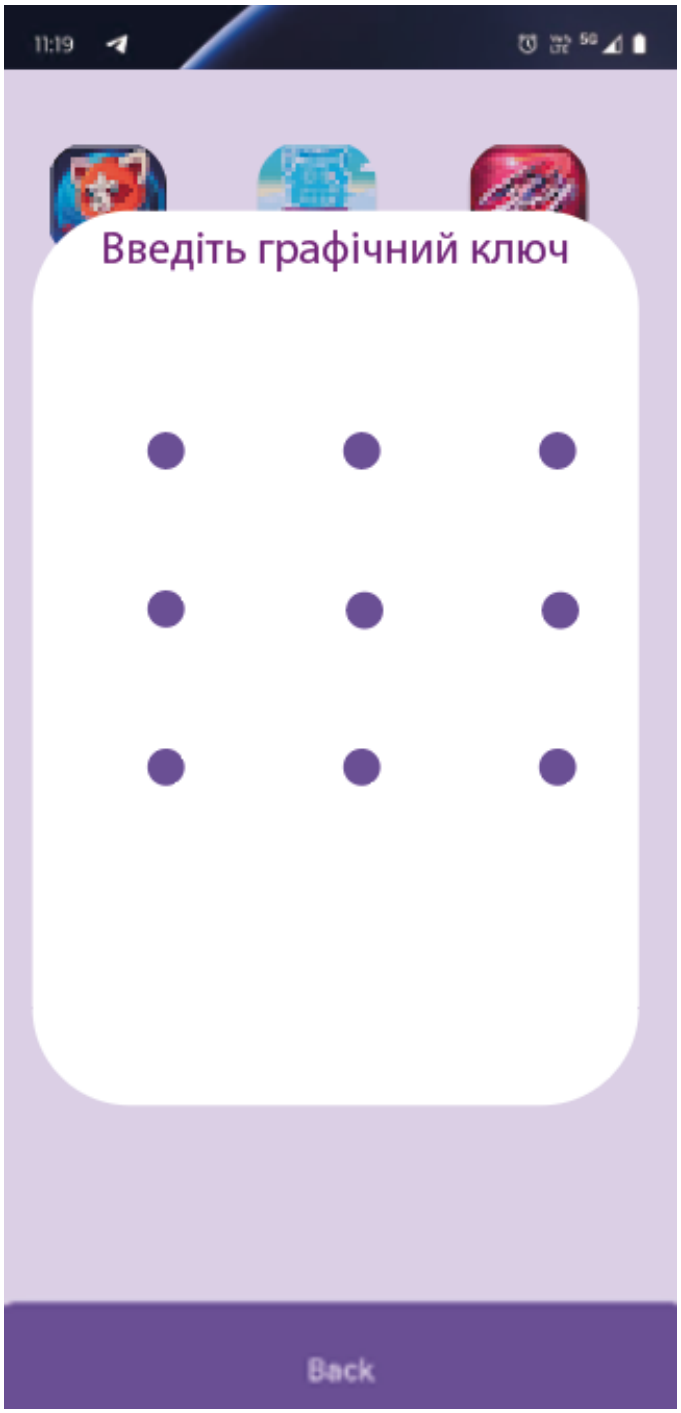


Рисунок 3.22 – Діалогове вікно для виходу з додатку

Варіант блокування – системний ключ

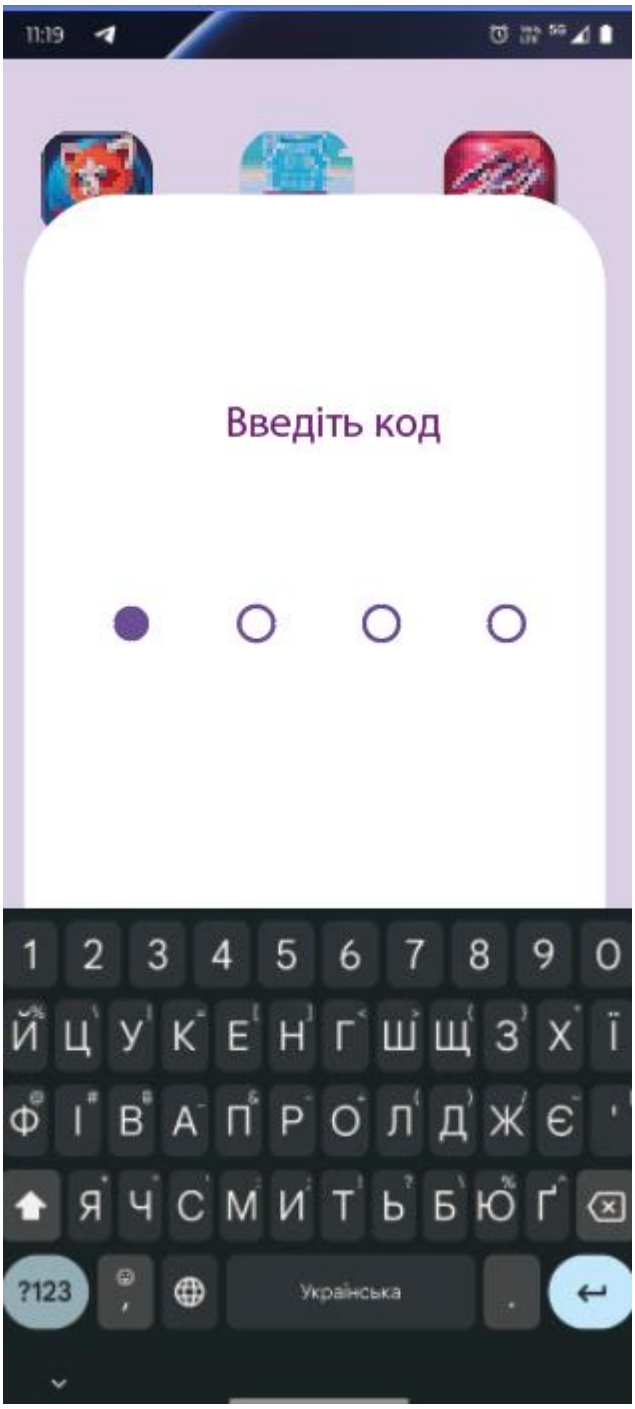


Рисунок 3.23 – Діалогове вікно для виходу з додатку

Варіант блокування – код

Для зміни базових налаштувань слід на головому екрані натиснути кнопку «налаштування». Інтерфейс налаштувань зображено на рис. 3.24.


Рисунок 3.24 – Сторінка налаштувань

Після вибору «Методу налаштування» буде відкрито екран для редагування методу блокування виходу з додатку (рис 3.25). Можна обрати один з трьох методів: Використання параметрів системи, використання пін-коду або пароля або ж простий математичний приклад, щоб не запам'ятовувати комбінацію.


Рисунок 3.25 – Вибір методу блокування виходу

## 3.4 Висновки

В даному розділі було розроблено додаток, який обмежує доступ дітей до інших додатків, які не дозволяють чіпати батьки. Було розроблено архітектуру інформаційної системи, UML діаграми інформаційної системи та окремих модулів, алгоритм роботи мобільного додатку, здійснено програмну реалізацію мобільного додатку. Продемонстровано роботу додатку в цілому.

# ВИСНОВКИ

В ході виконання бакалаврської дипломної роботи проведено аналіз предметної області, представлено суть проблеми та доведена актуальність необхідності обмежувати доступ недосвідчених користувачів до можливостей телефону.

Проведено аналіз проблематики предметної області, а саме зменшення віку, коли діти вперше отримують в руки телефон (навіть якщо він батьківський). Розглянуто основні можливості, переваги і недоліки існуючих інформаційних систем, які здійснюють контроль над використанням додатку.

Здійснено опис та обґрунтування вибору оптимальних засобів та інформаційних технологій для вирішення поставленої задачі. А саме, обґрунтовано вибір мови програмування Kotlin та програмної платформи Android Studio, за допомогою яких можливо забезпечити об’єднання мобільного додатку.

Розроблено мобільний додаток для забезпечення обмеженого режиму доступу на мобільному пристрої. Розроблено архітектуру інформаційної системи, UML діаграми інформаційної системи та окремих модулів, алгоритм роботи мобільного додатку, здійснено програмну реалізацію мобільного додатку.

Було продемонстровано функціонування додатку в цілому. Також проведено комплексне тестування функціональних можливостей додатку, перевірено його відповідність вимогам та поставленим задачам. На основі результатів тестування зроблено висновок, що додаток успішно виконує поставлені завдання. Додаток також працює ефективно, оскільки під час його використання, а також при його відключенні, ігрові додатки не демонстрували жодних затримок або помилок у роботі.

За результатами виконання даної роботи опубліковано тези доповідей на тему: «Мобільний додаток для забезпечення обмеженого режиму доступу на мобільному пристрої» на Міжнародній науково-практичній інтернет-

конференції «Молодь в науці: дослідження, проблеми, перспективи (Вінниця, 2023-2024 рр.) [1].

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Крижановський Є. М., Монастирська А. Ю. Мобільний додаток для забезпечення обмеженого режиму доступу на мобільному пристрої. Тези доповідей Міжнародної науково-практичної Інтернет-конференції студентів, аспірантів та молодих науковців. *Молодь в науці: дослідження, проблеми, перспективи (МН-2024)*, Вінниця, 20 травня 2024 р. 2024. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21652.
2. Дані моніторингу користувачів приладів, Statista. URL: https://www.statista.com/
3. Бліхар В. С. Особливості взаємозв'язків стилів сімейного виховання та батьківського ставлення зі страхами та тривожністю дітей. – 2024. URL: http://dspace.pnpu.edu.ua/handle/123456789/23063
4. Пахотіна Л. В. Вплив батьківського стилю виховання на майбутній сценарій життя дитини. – 2024.URL: https://dspace.znu.edu.ua/jspui/handle/12345/18975
5. . 44 Smartphone Addiction Statistics for 2022 [INFOGRAPHIC]. Accessed: Jan. 12, 2023. URL: https://www.slicktext.com/blog/2023/10/ smartphone-addictionstatistics/
6. Франчук В. М., Франчук Н. П. Використання Family Link батьками та дітьми //Комп'ютер в школі та сім'ї. – 2020. – №. 1. – С. 34-39. URL: http://dspace.puet.edu.ua/bitstream/123456789/10010/1/%2BCSF_01_20_00_RGB.pdf#page=34
7. Cunningham C., Zichermann G. Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps. O'Reilly Media, Incorporated, 2011.URL: https://doi.org/10.22214/ijraset.2019.2050
8. Guśpiel M. Mobile App Development Using Kotlin Multiplatform. 2024. URL: https://www.theseus.fi/handle/10024/852201.
9. Franchuk V., Franchuk N. Using Microsoft Family Safety for parents and children. 2022 International Conference on Innovative Solutions in Software Engineering (ICISSE), Vasyl Stefanyk Precarpathian National University,

Ivano-Frankivsk, Ukraine, Nov. 29-30, 2022, pp. 243-250. Accessed: Jan. 12, 2023. URL: https://lib.iitta.gov.ua/733847.

10. Налаштування дозволів. URL: https://support.google.com/android/answer/9431959.
11. Barzolevskaia A., Branca E., Stakhanova N. Measuring and Characterizing (mis) compliance of the Android permission system. IEEE Transactions on Software Engineering. 2024. URL: https://ieeexplore.ieee.org/abstract/document/10433080.
12. Компоненти користувацького інтерфейсу. URL: https://developer.android.com/design/ui.
13. Навігація для додатків. URL: https://developer.android.com/guide/navigation.
14. Paschke K., Matthis J., W ölfling K., Thome J. An app-based training for adolescents with problematic digital-media use and their parents (Res@t digital): protocol for a cluster-randomized clinical trial. Frontiers in Psychiatry. 2024. T. 14. C. 1245536. URL: https://www.frontiersin.org/journals/psychiatry/articles/10.3389/fpsyt.2023.1245536/full.
15. Карпенко М. І., Пуцик М. С., Гуйда О. Г., Василенко В. М. Розробка дизайну мобільного додатку для покращення якості освітнього процесу. Вчені записки Таврійського національного університету імені В.І. Вернадського. 2023. С. 18-25. URL: https://www.tech.vernadskyjournals.in.ua/journals/2023/4_2023/4_2023.pdf.
16. Білан Л. О. Методи двофакторної автентифікації користувачів в мобільних пристроях. 2021. URL: https://openarchive.nure.ua/bitstreams/7f481c42-5430-42bb-bfef-baa48c0cfaf8/download.
17. From game design elements to gamefulness / S. Deterding та ін. the 15th International Academic MindTrek Conference, м. Tampere, Finland, 28–30 верес. 2011 р. New York, New York, USA, 2011. URL: https://doi.org/10.1145/2181037.2181040
18. Джерело ігр для тестування: URL: https://play.google.com/store/apps/details?id=com.gokids.tablet&hl=uk

19. Гнідець, В. І. "Характеристики файрволів при контрольованому доступі до інформаційних ресурсів мобільної медіатеки." «Сучасна молодь в світі інформаційних технологій»: матеріали (2024): 148. URL: http://www.ksau.kherson.ua/files/konferencii/2024/05/mater_20_05_24.pdf#page=150
20. Вікторов Н. С. Рекомендаційна система медіа-контенту : дис. – КПІ ім. Ігоря Сікорського, 2024. URL: https://ela.kpi.ua/bitstream/123456789/63980/1/Viktorov_magistr.pdf

Додаток А

(обов’язковий)

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
“____”_____ 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську дипломну роботу

МОБІЛЬНИЙ ДОДАТОК ДЛЯ ЗАБЕЗПЕЧЕННЯ ОБМЕЖЕНОГО РЕЖИМУ ДОСТУПУ НА МОБІЛЬНОМУ ПРИСТРОЇ

08-34.БДР.006.00.000 ТЗ

Керівник: к.т.н., доц.
_____ Євгеній КРИЖАНОВСЬКИЙ
«__» _____ 2024 р.
Розробив студент гр. 2ІСТ-20б
_____Анастасія МОНАСТИРСЬКА
«__» _____ 2024 р.

Вінниця 2024

1. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2024р., та індивідуальне завдання на БДР, затверджене протоколом №__ засідання кафедри САІТ від «__» _____ 2024р.

2. Джерела розробки:

1) Android Studio, навчальний посібник. URL: https://developer.android.com/guide;

3. Мета і призначення роботи.

Метою дослідження є розробка мобільного додатку для забезпечення обмеженого режиму доступу на мобільному пристрої.

3. Вихідні дані для проведення робіт.

4. Методи дослідження:

1) Аналіз вимог;

2) Аналіз потреб користувачів

5. Етапи роботи і терміни їх виконання:

a) Аналіз предметної області _____ – _____

b) Вибір оптимальних інформаційних технологій _____ – _____

c) Розроблення додатку для забезпечення обмеженого режиму доступу на мобільному пристрої _____ – _____

d)Оформлення матеріалів до захисту БДР _____ – _____

7. Очікувані результати та порядок реалізації

Розроблений мобільний додаток для забезпечення обмеженого режиму доступу на мобільному пристрої.

8. Вимоги до розробленої документації

Текстова та ілюстративна частини роботи оформлені у відповідності до вимог «Методичних вказівок до виконання бакалаврських кваліфікаційних робіт для студентів спеціальностей: 124 «Системний аналіз», 126 «Інформаційні системи та технології» (освітня програма «Прикладні інформаційні технології»)».

9. Порядок приймання роботи

Публічний захист «___» _____ 2024 р.

Початок розробки «___» _____ 2024 р.

Граничні терміни виконання БДР «___» _____ 2024 р.

Розробив студент групи 2ІСТ-20б _____Анастасія МОНАСТИРСЬКА

Додаток Б

(обов'язковий)

Протокол перевірки бакалаврської дипломної роботи на наявність текстових запозичень

Назва роботи: «Мобільний додаток для забезпечення обмеженого режиму доступу на мобільному пристрої»
Тип роботи: бакалаврська дипломна робота
Підрозділ: кафедра САІТ

**Показники звіту подібності Unicheck**

Оригінальність 4.79% Схожість 95.01 %

Аналіз звіту подібності (відмітити потрібне)

● Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
○ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
○ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ (підпис) Сергій ЖУКОВ

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ (підпис) Анастасія МОНАСТИРСЬКА

Керівник роботи _____ (підпис) Євгеній КРИЖАНОВСЬКИЙ

# Додаток В

(довідниковий)

Фрагмент лістингу програми

## Код файлу додатку GameShortcutHelper.kt

```
class GameShortcutHelper {


   companion object {


      private val activityShortcuts = ActivityShortcuts.getInstance(context)

      private val permissionChecker = PermissionChecker(context)


      fun createGameShortcut(gamePackageName: String, gameIcon: Drawable) {

         if (permissionChecker.checkSelfPermission(Manifest.permission.READ_EXTERNAL_STORAGE) ==
PermissionChecker.PERMISSION_GRANTED) {

            // Перевірити, чи має гра доступ до особистих даних

            if (!hasGameAccessToPersonalData(gamePackageName)) {

               // Створити ярлик

               val shortcutInfo = ShortcutInfo.Builder(context, gamePackageName)

                  .setShortLabel(getGameName(gamePackageName))

                  .setIcon(gameIcon)

                  .setIntent(Intent(Intent.ACTION_VIEW, Uri.parse("package:$gamePackageName")))

                  .build()


               activityShortcuts.addShortcut(shortcutInfo)

            }

         }

      }


      private fun hasGameAccessToPersonalData(gamePackageName: String): Boolean {

         // Реалізуйте логіку для перевірки, чи має гра доступ до особистих даних

         return false // Замінити на вашу логіку

      }


      private fun getGameName(gamePackageName: String): String {
```

```
        // Реалізуйте логіку для отримання назви гри
        return gamePackageName // Замінити на вашу логіку
    }
  }
}
```

## Код файлу додатку AndroidManifest.kt

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">

  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round">

    <activity
      android:name=".MainActivity"
      android:exported="true">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>

    <receiver
      android:name=".ShortcutReceiver"
      android:exported="true">
      <intent-filter>
        <action android:name="com.android.launcher.action.INSTALL_SHORTCUT" />
      </intent-filter>
    </receiver>
```

```
        <service
            android:name=".ShortcutService"
            android:exported="true">
        </service>

    </application>

</manifest>
```

## Код файлу додатку LockManagerTest.kt

```
class LockManagerTest {

    @Test
    fun testUnlockWithSystemSettings() {
        val lockManager = LockManager()
        val systemSettings = mock(SystemSettings::class.java)
        `when`(systemSettings.isAvailable()).thenReturn(true)

        val result = lockManager.unlockWithSystemSettings()

        assertTrue(result)
    }
}
```

## Код файлу додатку LockManagerTest.kt

```
class LockManager {

    private var isLocked = true
    private val unlockKey = "123456" // Замінити на ваш ключ

    fun isLocked(): Boolean {
        return isLocked
    }

    fun lock() {
        isLocked = true
```

```
    }

    fun unlockWithMathProblem(): Boolean {
        val mathProblem = MathProblemGenerator.generateProblem()
        val userAnswer = getUserAnswer()

        if (mathProblem.isCorrectAnswer(userAnswer)) {
            unlock()
            return true
        } else {
            return false
        }
    }

    fun unlockWithSystemSettings(): Boolean {
        if (isSystemSettingsAvailable()) {
            launchSystemSettings()
            return true
        } else {
            return false
        }
    }

    fun unlockWithPinCode(): Boolean {
        val userInputPin = getUserPinCode()

        if (userInputPin == unlockKey) {
            unlock()
            return true
        } else {
            return false
        }
    }
```

Код файлу додатку MathProblemGenerator.kt

```
class MathProblemGenerator {
```

```
    fun generateProblem(): Pair<String, Int> {
        val num1 = Random.nextInt(1, 101)
        val num2 = Random.nextInt(1, 101)
        val operator = when (Random.nextInt(1, 5)) {
            1 -> "+"
            2 -> "-"
            3 -> "*"
            else -> "/"
        }
        val answer = when (operator) {
            "+" -> num1 + num2
            "-" -> num1 - num2
            "*" -> num1 * num2
            else -> num1 / num2
        }
        val problem = "$num1 $operator $num2"
        return Pair(problem, answer)
    }
}
```

Додаток Г

(обов’язковий)

**ІЛЮСТРАТИВНА ЧАСТИНА**

МОБІЛЬНИЙ ДОДАТОК ДЛЯ ЗАБЕЗПЕЧЕННЯ ОБМЕЖЕНОГО РЕЖИМУ ДОСТУПУ НА МОБІЛЬНОМУ ПРИСТРОЇ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2024 р.

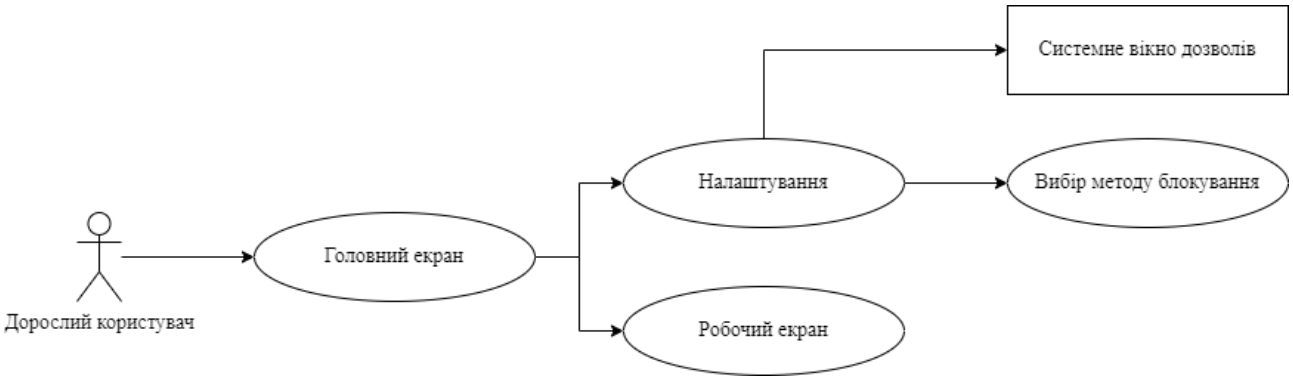


Рисунок Г1 – Діаграма use case

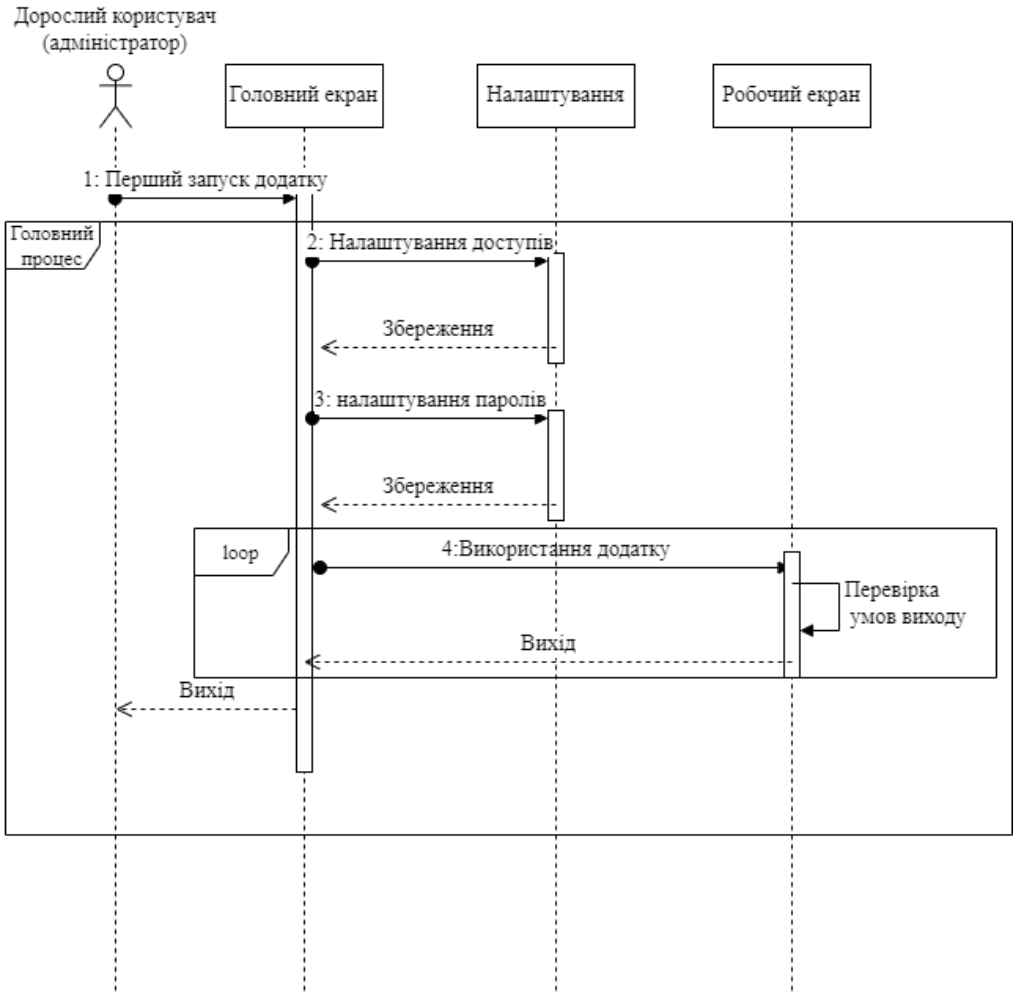


Рисунок Г2 – Діаграма взаємодії

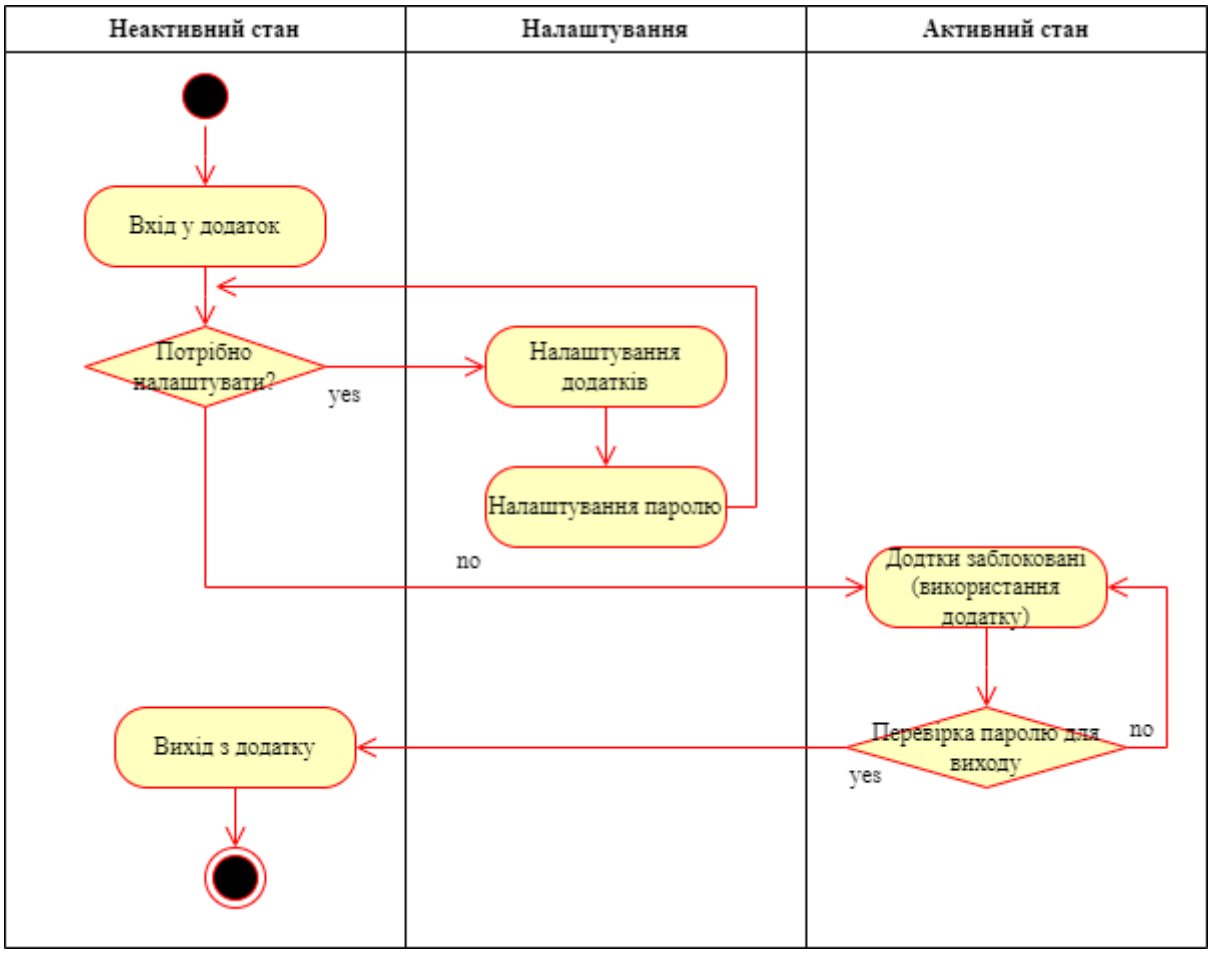


Рисунок Г3 – Діаграма варіантів використання

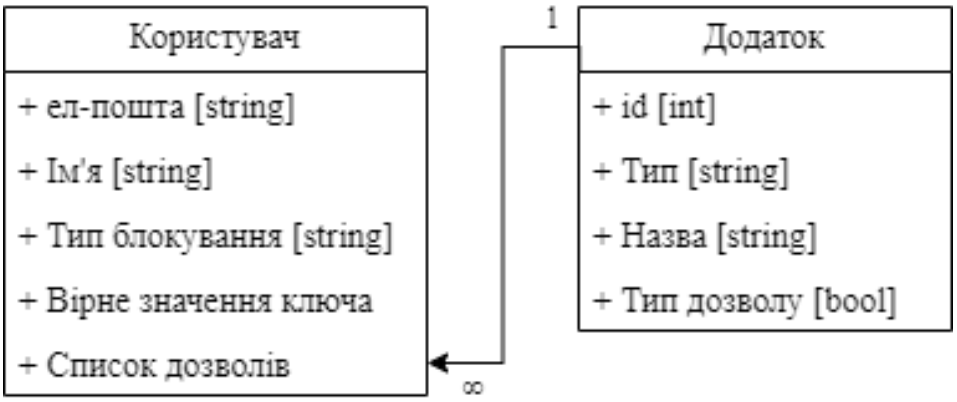


Рисунок Г4 – Діаграма класів

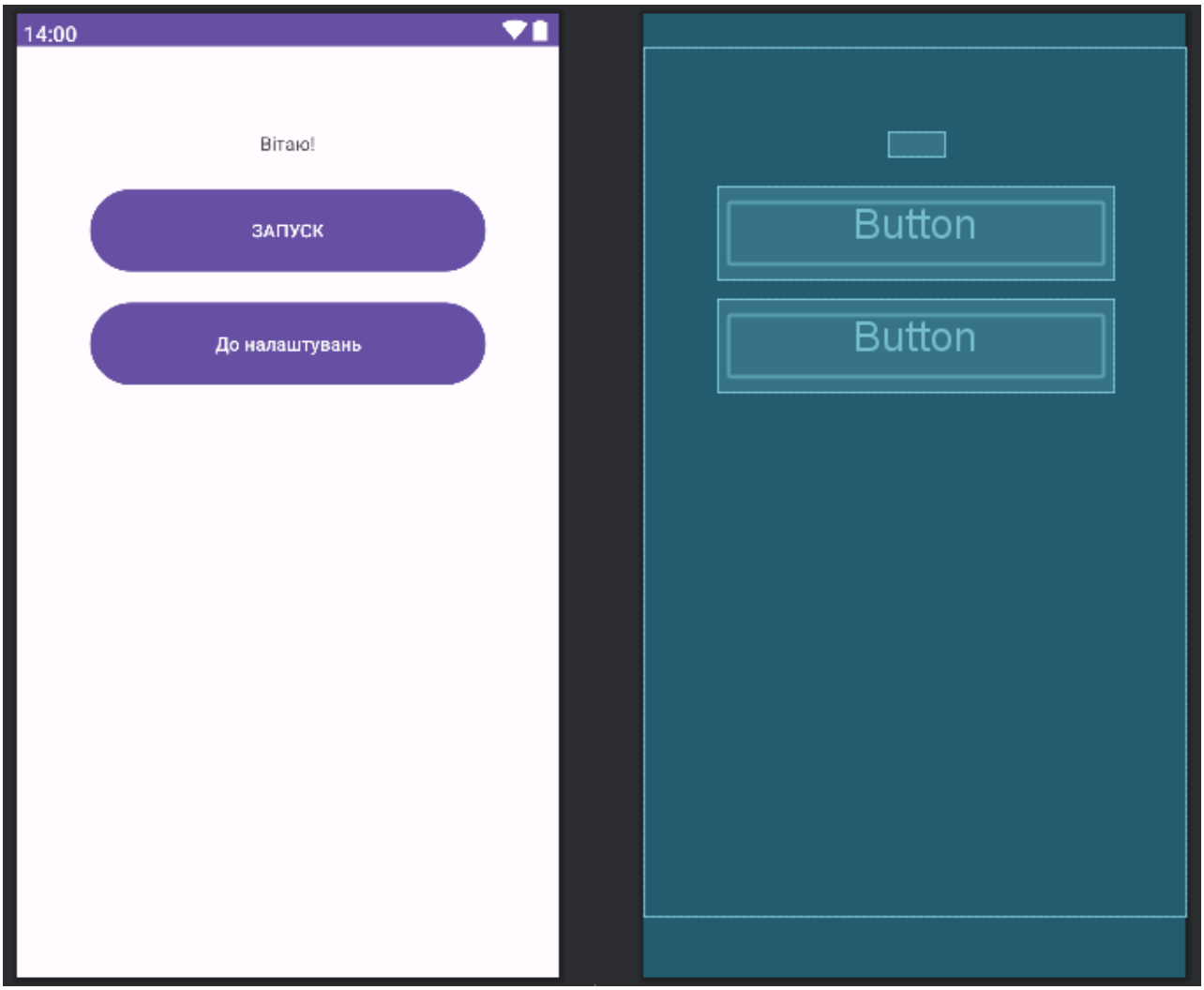


Рисунок Г5 – Домашня сторінка, найперша сторінка

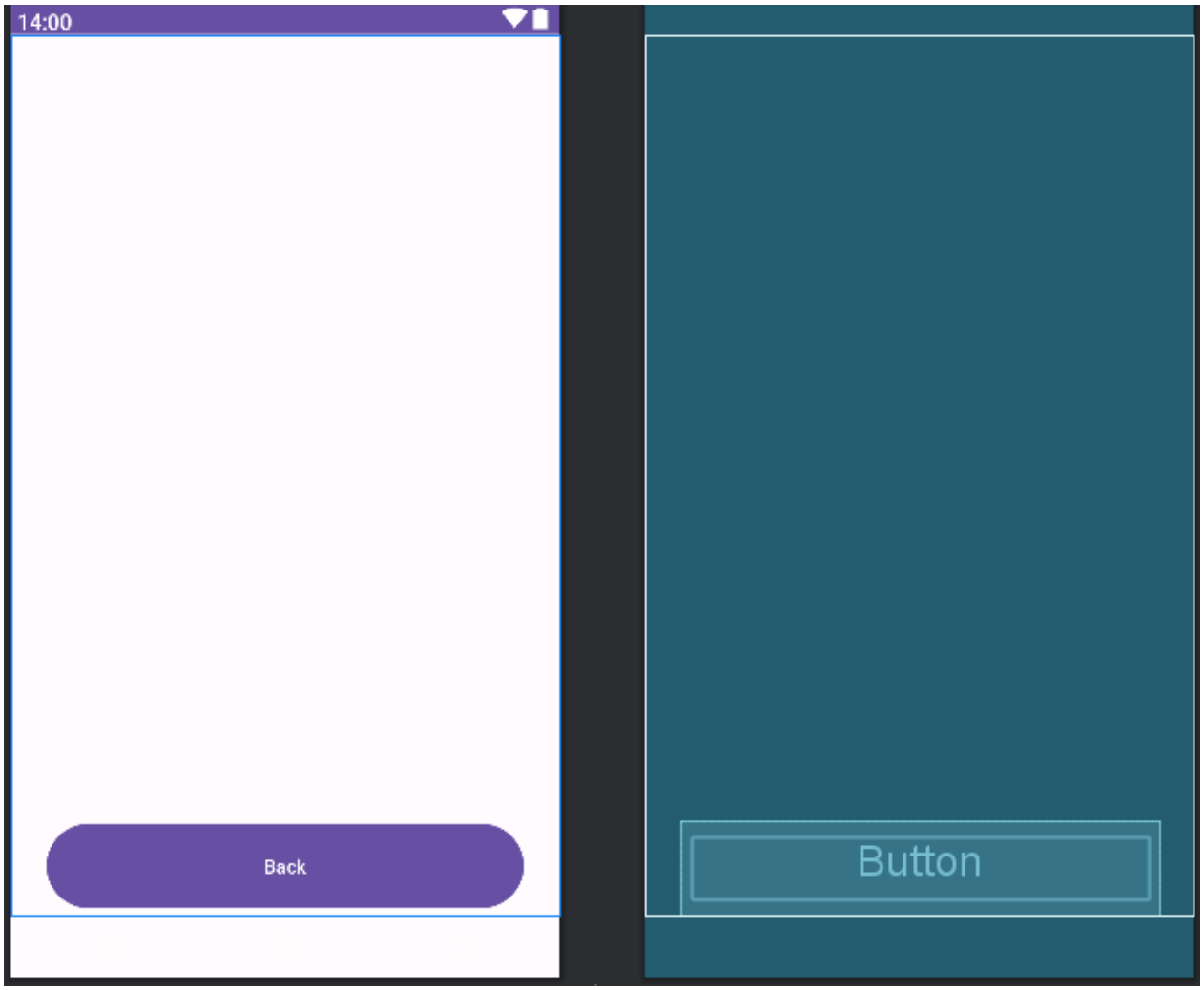


Рисунок Г 6 – Вікно, на якому відображатимуться додатки

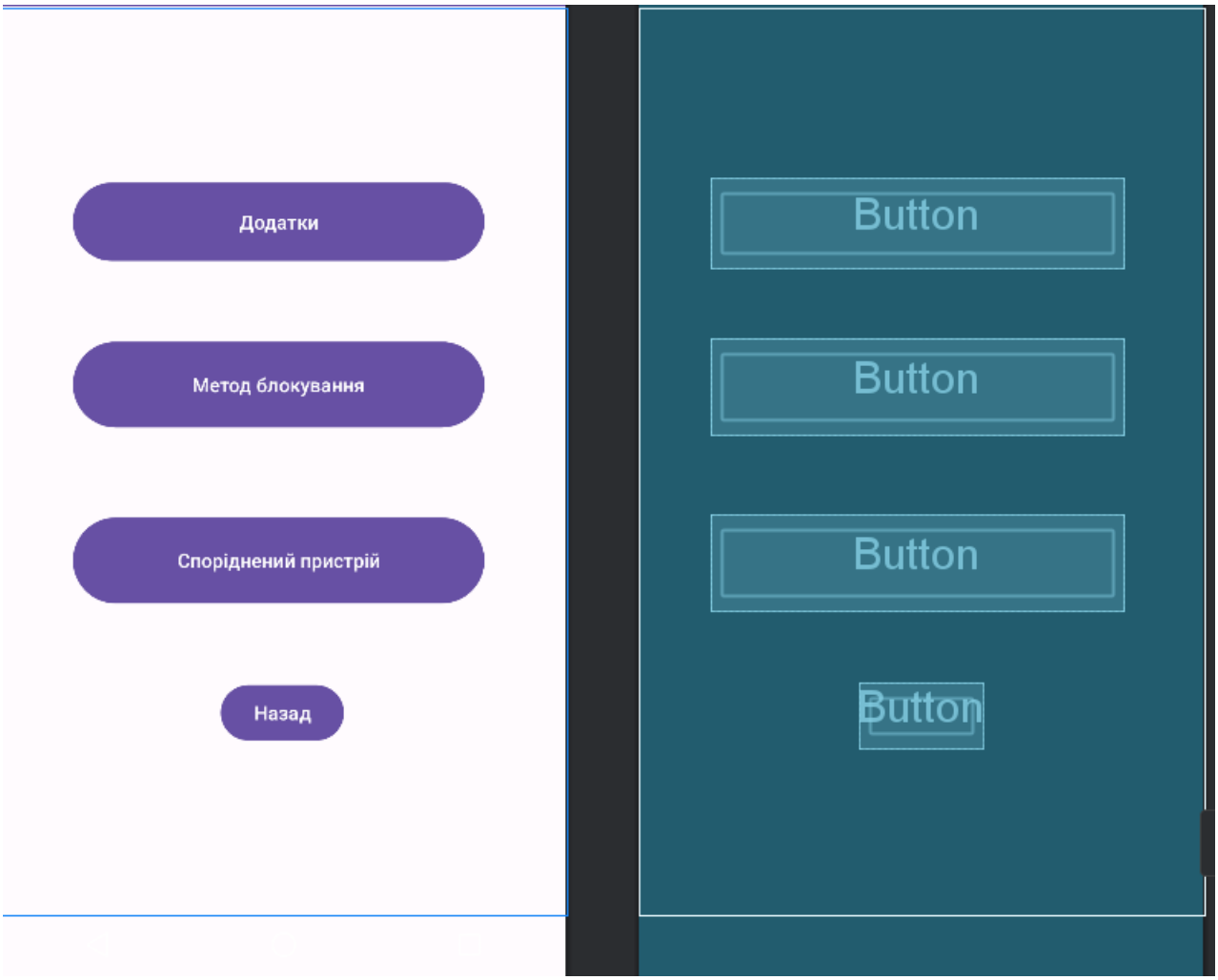


Рисунок Г7 – Сторінка налаштування

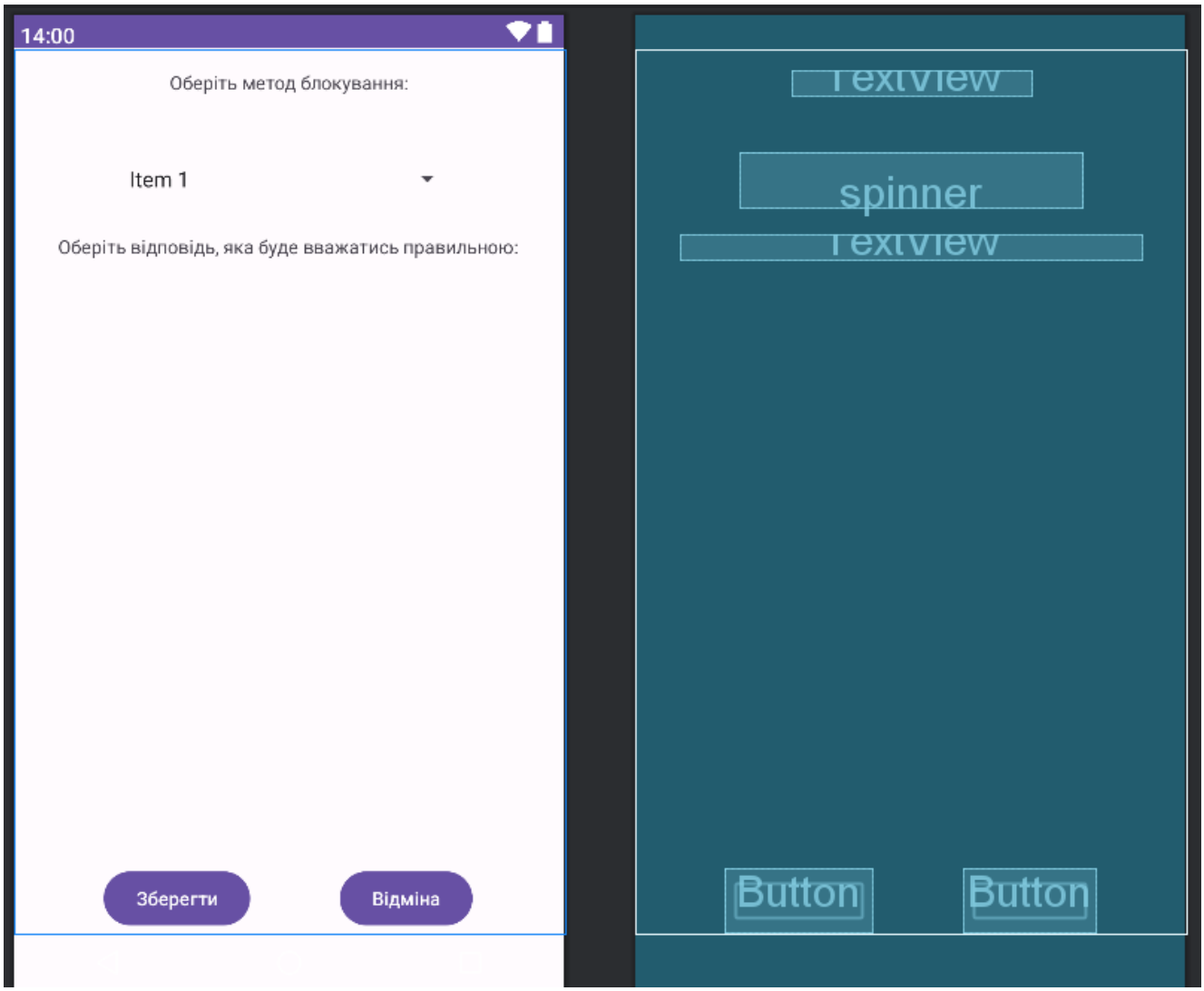


Рисунок Г8 – Сторінка налаштування блокування

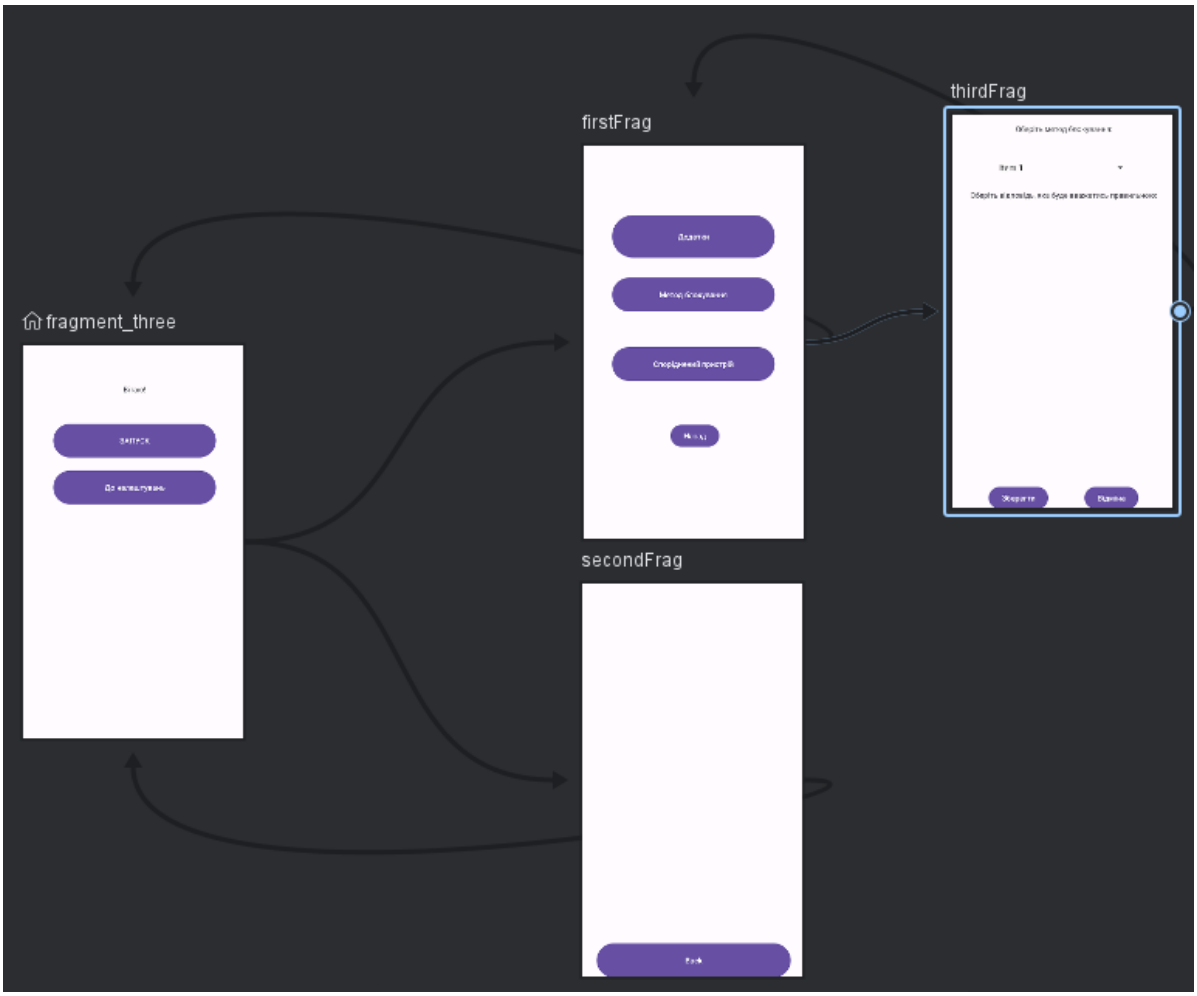


Рисунок Г9 – Загальний вигляд навігації