**Міністерство освіти і науки України**

**Луцький національний технічний університет**
(повне найменування закладу вищої освіти)

**Факультет комп'ютерних та інформаційних технологій**
(повне найменування факультету)

**Кафедра комп'ютерної інженерії та кібербезпеки**
(повне найменування кафедри)

# КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

## ІНТЕРАКТИВНИЙ ДОДАТОК «EMERGENCY APP» ЗАСОБАМИ ANDROID STUDIO

## INTERACTIVE APPLICATION «EMERGENCY APP» BY MEANS OF ANDROID STUDIO

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІс-21
Якобчук Богдан Анатолійович

(підпис)

Керівник:
к.т.н., доцент
Христинець Наталія Анатоліївна

(підпис)

Кваліфікаційну роботу
допущено до захисту
« 07 » червня 2024 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та кібербезпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ проф. Н.Чернящук
« 10 » 01 2024 р.

# ЗАВДАННЯ
# НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

*Якобчуку Богдану Анатолійовичу*
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Інтерактивний додаток «Emergency app» засобами Android Studio*

Керівник роботи *к.т.н., доцент Христинець Наталія Анатоліївна*

затверджені наказом закладу вищої освіти від «30» грудня 2023 року № 459/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 11.06.2024р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області та різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
*Вступ*
*Аналіз стану побудови програмних додатків*
*Дослідження методів побудови тренувальних додатків*
*Розробка програми: проектування меню, елементів, функцій зовнішньої взаємодії*
*Висновки*

5. Перелік графічного (ілюстративного) матеріалу:
*Огляд технологій для реалізації проекту*
*Алгоритми функціонування додатку*
*Розробка інтерфейсу*
*Тестування додатку*

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| *Аналіз проблеми за темою роботи та постановка дослідження* | *Христинець Н.А., доцент* | | |
| *Теоретичне дослідження та практична реалізація* | *Христинець Н.А., доцент* | | |
| *Практична реалізація об'єкта проектування* | *Христинець Н.А., доцент* | | |
| *Нормоконтроль* | *Багнюк Н.В., доцент* | | |
| *Гарант ОП* | *Лавренчук С.В., доцент* | | |
| *Показник запозичень тексту* | ***___%*** | | |
| *Академічна доброчесність* | *Міскевич О.І.,асистент* | | |

7. Дата видачі завдання 10.01.2024 р.

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | *Розділ 1. Аналіз предметної області* | до 20.02.2024 р. | Виконано |
| 2. | *Розділ 2. Методи побудови тренувальних додатків та інструменти розробки* | до 20.03.2024 р. | Виконано |
| 3. | *Розділ 3. Розробка програми «Emergency App»* | до 04.05.2024 р. | Виконано |
| 4. | *Висновки та пропозиції* | до 07.05.2025 р. | Виконано |
| 5. | *Формування списку використаних джерел* | до 10.05.2024 р. | Виконано |
| 6. | *Формування додатків* | до 20.05.2024 р. | Виконано |
| 7. | *Оформлення ілюстративного матеріалу* | до 28.05.2024 р. | Виконано |
| 8. | *Нормоконтроль* | до 01.06.2024 р. | Виконано |
| 9. | *Інструментальна перевірка на академічний плагіат* | до 04.06.2024 р | Виконано |
| 10. | *Представлення кваліфікаційної роботи бакалавра до захисту* | до 11.06.2024 р. | Виконано |

**Здобувач вищої освіти** _____ (підпис) Якобчук Б.А. (прізвище, ініціали)

**Керівник кваліфікаційної роботи** _____ (підпис) Христинець Н.А. (прізвище, ініціали)

# АНОТАЦІЯ

Якобчук Б.А. Інтерактивний додаток «Emergency App» засобами Android Studio. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2024. 69 с.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел, додатків.

Перший розділ присвячено огляду предметної олбасті. Тут розглядаються існуючі рішення додатків, для навчання діям у надзвичайних ситуаціях, вибір платформи, інструментів та засобів розробки такого додатку. Обрано: платформу Android, мову програмування Java та інструментарій Android Studio. Також в цьому розділі здійснено аналіз того, як обрана IDE, буде взаємодіяти з IoT.

В другому розділі описано особливості засобів розробки, алгоритми візуального функціонування додатку та відлагодження роботи додатку.

Третій розділ присвячено опису розроблених активностей, до яких належить: головне меню, меню рівнів та різні етапи проходження рівня 1. Також, у даному розділі описана реалізація голосового управління у додатку.

Ключові слова: навчання, екстрена ситуація, Android-додаток, голосове управління, Java, Android Studio.

# ANNOTATION

Yakobchuk B.A. Interactive application «Emergency App» by means of Android Studio. Manuscript.

Qualification work for bachelor's degree in Computer Engineering, specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2024. 73 p.

The qualification work consists of an introduction, three chapters, conclusions, a list of references, and appendices.

The first chapter is devoted to an overview of the subject area. It discusses the existing solutions of applications for training in emergency situations, the choice of platform, tools and means of developing such an application. The Android platform, the Java programming language, and the Android Studio toolkit were selected. This section also analyzes how the selected IDE will interact with IoT.

The second section describes the features of the development tools, algorithms for the visual functioning of the application, and debugging the application.

The third section is devoted to the description of the developed activities, which include: the main menu, the level menu, and the various stages of passing level 1. This section also describes the implementation of voice control in the application.

Keywords: training, emergency, Android application, voice control, Java, Android Studio.

# ЗМІСТ

# ВСТУП

Актуальність теми. У сучасному світі технологій важливою складовою є розробка інноваційних рішень для підвищення безпеки та підготовки громадян до надзвичайних ситуацій. Враховуючи стрімкий розвиток мобільних пристроїв та їх потенціал, особливого значення набуває створення інтерактивних додатків для навчання діям у надзвичайних ситуаціях. У цьому контексті розробка інтерактивного додатку «Emergency App» засобами Android Studio стає актуальним завданням, що відповідає потребам сучасного суспільства.

Стан вивченості проблеми. Проблема підготовки населення до дій у надзвичайних ситуаціях є актуальною і постійно досліджується науковцями та практиками. Це пояснюється тим, що належні дії можуть зменшити ризики для майна, здоров’я та навіть врятувати життя. На сьогоднішній день існують різноманітні підходи та технології для цієї мети, однак для сучасного користувача ключовими є інтерактивність та доступність навчального матеріалу.

Метою кваліфікаційної роботи є розробка та впровадження інтерактивного додатку «Emergency App», за допомогою інструментів Android Studio спрямованого на навчання користувачів ефективним діям у надзвичайних ситуаціях. Основним завданням є аналіз сучасних підходів до навчання, вибір оптимальних методів та технологій для реалізації додатку, а також його функціональне вдосконалення для забезпечення максимальної ефективності навчання.

Об’єктом дослідження є інтерактивний додаток «Emergency App». Одним з центральних аспектів проекту є використання Інтернету речей, зокрема мікрофону, для реалізації голосового управління у додатку.

Предметом дослідження є методи та технології розробки мобільного додатку «Emergency App».

Для досягнення мети було поставлено наступні завдання:

- провести аналітичний огляд інтернет-ресурсів, наукових публікацій, літературних джерел з питань розробки інтерактивних додатків та визначити платформу, інструментарій та мову програмування для реалізації проекту;
- дослідити об’єкти IoT, що включені в роботу додатка, та визначити спосіб їх інтеграції з IDE;
- визначити класи, які будуть використані в середовищі мови програмування для створення функціоналу додатку та аргументувати вибір;
- розробити головне меню, меню рівнів додаку та реалізувати запуск активностей для візуального функціонування та програмно реалізувати перехід між ними;
- створити елементи взаємодії у додатку та визначити, які з цих елементів будуть інтерактивні;
- впровадити функціонал розпізнавання голосових команд, налаштувати його взаємодію з програмним кодом та дослідити особливості такої взаємодії;
- провести збірку і тестування програми, перевірити коректність роботи активностей.

Апробація. Якобчук Б.А. Створення простого голосового помічника у додатку для платформи Android. *Комп’ютерно-інтегровані технології: освіта, наука, виробництво*. 2024. №55. С. 124-129 (Додаток В).

# РОЗДІЛ 1
# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Аналіз стану побудови програмних додатків для поведінки в надзвичайних ситуаціях

У теперішньому світі, як і раніше, відбуваються неочікувані ситуації, які несуть небезпеку для життя чи здоров’я людей. Саме через це важливо знати, як потрібно діяти у таких ситуаціях. Одним із можливих варіантів навчання є програмні додатки. На ринку існує багато додатків, спрямованих на навчання користувачів правильній поведінці в надзвичайних ситуаціях. Вони розроблені для різних платформ, можуть мати різну структуру та наповнення. Перед тим, як розробити власний додаток, було ретельно проаналізовано існуючі програмні рішення, щоб зрозуміти їхні переваги, недоліки та потенційні можливості для покращення.

З метою аналізу в даному науковому дослідженні розглянемо деякі існуючі програмні додатки, призначені для навчання поведінці в надзвичайних ситуаціях. В аналізі будуть враховані їх функціональні можливості, платформа, доступність та особливості використання.

«Disaster Master» [1] – це інтерактивний онлайн веб-додаток, призначений для дітей, який має на меті навчити їх, як діяти під час різних небезпечних ситуацій, таких як природні катастрофи, пожежі, землетруси тощо. Ця гра спрямована на підвищення обізнаності про небезпеки та природні катастрофи, шляхом проходження сценаріїв. Гра є безкоштовною та має простий і зрозумілий інтерфейс. Вона містить реалістичні сценарії, що допомагають користувачам набути практичних навичок, поглибившись у реальність ситуацій. Також, цей додаток має інтерактивні елементи, такі як вибір дій, відповіді на питання та вирішення завдань. Недоліками гри «Disaster Master» є відсутність локалізації та адаптивності під різні типи пристроїв.

Гра «Disaster Mind Game» [2] – навчальна гра, розроблена Агентством з управління надзвичайними ситуаціями США (FEMA), яка навчає старшокласників, як зберігати спокій і приймати тверді рішення під час

катастрофи. У цій грі гравцям потрібно швидко приймати рішення перед трьома стихійними лихами: хуртовиною, лісовою пожежею та повінню. Ускладнюючи їхні пошуки, таємничий провідник плете перипетії на цьому шляху. Особливістю цього додатка є те, що його можна грати як поодинці, так і у групі. Недоліками додатка є відсутність локалізації та те, що веб-додаток доступний не у всіх браузерах і не зверстаний для всіх пристроїв.

«First Aid & Emergency Techniques» [3] є додатком, призначеним для надання першої допомоги та надання технічних порад у невідкладних ситуаціях. Він містить інформацію про різні медичні стани, поради щодо надання першої допомоги та техніки виживання. Додаток може бути корисним для навчання та підготовки до надзвичайних ситуацій. Програма розроблена для платформи Android, є безкоштовною та доступна на Google Play Store. Недоліками додатку є те, що він не містить інтерактивних елементів, а натомість містить багато тексту.

«B-Ready App» [4] – це додаток, який надає користувачам інформацію та інструменти для підготовки до різних видів надзвичайних ситуацій, таких як природні катастрофи, терористичні акти, епідемії та інші види катастроф. Додаток включає в себе розділи, які допомагають зрозуміти, як діяти під час надзвичайних ситуацій, як знайти допомогу та як підготувати себе та свою родину. «B-Ready App» доступний для iPhone, Android та iPad. Додаток також може бути корисним для бізнесів, які хочуть забезпечити безпеку своїх співробітників. Однак, суттєвим недоліком додатку є те, що він є платним. Також, у додатку відсутня реалізація локалізації.

«Stop Disasters!» [5] є інтерактивною освітньою грою, яка навчає гравців профілактичним заходам для зменшення ризику природних катастроф. Гра дозволяє вивчити різні сценарії, такі як землетруси, повені, торнадо, пожежі тощо. Гравці можуть приймати рішення щодо будівництва, розміщення інфраструктури та евакуації населення. Гра доступна онлайн на веб-сайті Stop Disasters. Вона не вимагає встановлення додаткових програм або платформ. Гра доступна для всіх користувачів з доступом до Інтернету. Інтерфейс простий та інтуїтивно зрозумілий. Гра може бути корисною для навчання учнів, студентів

та простих громадян. Недоліками гри є обмеження в реалістичності, оскільки вона спрощує складність природних катастроф. Вона не замінює практичного навчання та реальних ситуацій.

«Emergency Ready» [6] – це додаток, який надається Південнокорейським агентством з управління надзвичайними ситуаціями. У ньому присутні наступні функції: пошук укриттів, пошук медичних центрів невідкладної допомоги, пошук пожежних департаментів та поліцейських відділень, посібник з безпеки, екстрені контакти. Додаток доступний на Google Play для Android. Також, цей додаток надає корисні функції для надзвичайних ситуацій. Інтерфейс додатку інтуїтивно зрозумілий для користувачів, однак додаток не має реалізації на інші мови, крім корейської та англійської. Також, «Emergency Ready» не містить інтерактивних дій, які б допомогли користувачам зрозуміти більш детально, як діяти у екстрених ситуаціях.

Для проведення підсумків аналізу програмних додатків, які навчають поведінки в надзвичайних ситуаціях, було створено таблицю 1.1 з результатами досліджень. Зазначені основні характеристики кожного додатку, такі як функціональні можливості, платформа, доступність, особливості використання та недоліки.

Таблиця 1.1 – Результати аналізу стану побудови програмних додатків для поведінки в надзвичайних ситуаціях

| Назва додатку | Функціональні можливості | Платформа | Доступність | Особливості використання | Недоліки |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |
| Disaster Master | Інтерактивні сценарії, вибір дій, простий інтерфейс | Веб-додаток | Безкоштовний | Реалістичні сценарії, зрозумілий інтерфейс | Відсутність адаптивності для різних пристроїв |
| Disaster Mind Game | Інтерактивні сценарії, гра у групі, простий інтерфейс | Веб-додаток | Безкоштовний | Гра у групі, швидке прийняття рішень | Відсутність підтримки у деяких браузерах |

Продовження таблиці 1.1

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| First Aid & Emergency Techniques | Інформація про першу допомогу, поради | Android | Безкоштовний | Інформативний додаток, доступний на Google Play Store | Велика кількість тексту, без інтерактивних елементів |
| B-Ready App | Інформація та інструменти для підготовки | iPhone, Android, iPad | Платний | Доступність для різних платформ, корисний для бізнесів | Відсутність локалізації |
| Stop Disasters! | Інтерактивні сценарії, доступність онлайн | Веб-додаток | Безкоштовний | Простий інтерфейс, корисний для всіх | Спрощення складності природних катастроф |
| Emergency Ready | Пошук укриттів, медичні центри, безпечні місця | Android | Безкоштовний | Інтуїтивний інтерфейс, корисний для надзвичайних ситуацій | Відсутність інтерактивних дій, відсутність локалізації |

Провівши підсумки аналізу, було виявлено, що кожен із додатків має свої переваги та недоліки. Однак, через проведений аналіз існуючих рішень стає очевидним, що існує певна прогалина в локалізації цих додатків для нашої країни, а також ці додатки не мають достатнього рівня інтерактивності у діях.

У зв’язку з цим, розробка нового додатку має велике значення та затребуваність. Перевагами цього додатку можуть бути: локалізація для нашої країни, інтерактивна інформація та інтерфейс, безкоштовний доступ та інші інноваційні технології.

Отже, розробка нового додатку у цій області може вирішити існуючі проблеми та недоліки інших програмних рішень, забезпечуючи користувачам

доступ до якісної та ефективної інформації про поведінку в надзвичайних ситуаціях.

## 1.2 Вибір інструментарію та вибір платформи

При розробці нового додатку першим питанням постає вибір платформи. На сьогоднішній день різні платформи користуються великим попитом, але найбільш популярними є платформи, на яких працюють операційні системи Windows, macOS, Android і iOS. Кожна з цих платформ має свої переваги, особливості та обмеження. Наприклад, Windows та macOS в основному використовуються на персональних комп'ютерах та ноутбуках і забезпечують широкий функціонал для розробки додатків різних логік та інтерфейсів. Натомість, Android та iOS призначені для мобільних пристроїв, таких як смартфони та планшети, і мають свої унікальні особливості, такі як мобільність, спеціалізовані API та інтерфейси користувача. Тому, вибір платформи залежить від цільової аудиторії, функціональних вимог та стратегії розробки.

Було розглянуто основні характеристики та особливості цих платформ, щодо розробки для них нових програмних застосунків, які пердставлені в таблиці 1.2 та обрано платформу Android. Вона є найпопулярнішою операційною системою для мобільних пристроїв у світі.

Таблиця 1.2 – Порівняння платформ на можливості розробки нових додатків

| Платформа | Примітка | Присутність на ринку | Гнучкість розробки | Спільнота розробників |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| Windows | Підтримується на ПК, ноутбуках, нетбуках | Велика | Висока | Велика |
| macOS | Ексклюзивно для продукції Apple | Обмежена | Висока | Середня |

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| Android | Використовується на смартфонах та планшетах | Дуже велика | Висока | Дуже велика |
| iOS | Ексклюзивно для продукції Apple | Обмежена | Висока | Велика |

Продовження таблиці 1.2

Велика частина аудиторії користується смартфонами та планшетами на базі Android, що робить цю платформу привабливою для розробки додатків. Також, Android надає розробникам широкий набір інструментів і технологій для створення різноманітних додатків. Гнучкість розробки додатків для цієї платформи дозволяє адаптувати програми до різних розмірів екранів, версій операційної системи та потреб користувачів. Android працює на широкому спектрі пристроїв різних виробників та різної ціни.

Отже, на основі аналізу популярних платформ для розробки додатків, можна зробити висновок, що платформа Android задовольняє потребам та вирішує завдання найкраще. Її велика популярність, гнучкість та доступність стали вирішальними факторами у виборі. Таким чином, при розробці предмету кваліфікаційної роботи, було обрано платформу Android для найбільш ефективного досягнення мети проекту.

Наступним кроком розробки додатку є вибір інструментарію. В основі сучасної технології розробки лежить IDE. IDE – це інтегроване середовище розробки, що об’єднує в собі різноманітні інструменти, які допомагають програмістам створювати програмне забезпечення більш ефективно та продуктивно. IDE включає в себе текстовий редактор для написання коду, компілятор для перетворення програмного коду на виконуваний файл, відладчик для виявлення та виправлення помилок, інструменти для автоматизації рутинних задач, а також різноманітні плагіни та розширення, що доповнюють функціональність середовища розробки.

Найбільш поширеними IDE для розробки додатків Android є Android Studio, IntelliJ IDEA з плагінами для Android розробки, Eclipse та NetBeans (рис. 1.1). Вибір конкретно IDE залежить від поставленої задачі та складності її реалізації.


Рисунок 1.1 – Найпоширеніші Android IDE

Було розглянуто кожну з цих IDE, їхні сильні та слабкі сторони (табл. 1.3) та сформовано висновок, що найкращим варіантом для реалізації додатку буде Android Studio. Android Studio – це офіційне середовище розробки для платформи Android, розроблене на базі IntelliJ IDEA. Основна мета Android Studio – надати розробникам інструмент, що оптимізований саме для створення Android додатків з нуля або їх підтримки.

Одна з ключових переваг Android Studio полягає в його інтеграції з Android SDK (Software Development Kit), який містить усі необхідні компоненти для розробки, такі як Android API, емулятори, інструменти для відлагодження та інше. Крім того, Android Studio має широкий набір плагінів, що доповнюють його функціональність і дозволяють працювати більш ефективно. Також, вони дозволяють бути більш гнучкими у розробці та дозволяють створювати адаптивні додатки. Завдяки вбудованим інструментам

для тестування та аналізу продуктивності, розробники можуть оптимізувати свої додатки ще на етапі розробки. Інтеграція з системами контролю версій, такими як Git, забезпечує зручність у командній роботі та управлінні проектами.

Таблиця 1.3 – Найпопулярніші IDE для розробки Android-додатків

| Інструмент | Сильні сторони | Слабкі сторони |
|---|---|---|
| Android Studio | Офіційне середовище розробки для Android, велика кількість функціональностей | Вимагає великих ресурсів системи для великих проектів |
| IntelliJ IDEA | Платформа, на якій базується Android Studio, підтримка різних мов програмування | Може бути менш оптимізований для розробки Android додатків |
| Eclipse | Гнучкість та розширюваність через широкий вибір плагінів | Менша підтримка для Android порівняно з Android Studio |
| NetBeans | Добра підтримка для роботи з Java та іншими мовами програмування, легкість використання | Може не мати такої розгорнутої підтримки для розробки Android додатків |

Отже, проаналізувавши всі сильні та слабкі сторони найпопулярніших IDE для розробки Android додатків, було сформовано висновок, що Android Studio найбільш підходить для даного проекту. Аргументувати цей вибір можна тим, що Android Studio має переваги у великому спектрі інструментів, які спеціально адаптовані для розробки Android додатків. IntelliJ IDEA, Eclipse та NetBeans також мають свої переваги, але для реалізації даного додатку, важливою була спеціалізованість і оптимізованість саме під платформу Android.

Наступним кроком розробки додатку є вибір мови програмування. Android Studio підтримує дві мови програмування: Java і Kotlin (рис. 1.2). Як Java, так і Kotlin мають рівні можливості для створення високоякісних Android додатків. Обидві мови підтримують роботу з Android SDK, включаючи доступ до Android API та інших необхідних компонентів для розробки. Вони також обидві мають можливості для роботи з макетами інтерфейсу, відлагодження

коду та тестування додатків. Через це, вибір мови програмування у проекті є досить складним.


Рисунок 1.2 – Мови програмування, підтримувані Android Studio

Java є традиційною і широко використовуваною мовою програмування для розробки Android додатків. Але її особливістю є кросплатформеність, тобто те, що розробляти мовою Java можна не тільки на Android, але і на багато інших пристроїв. Java має широкий досвід користувачів та велику базу знань, що робить її зручним вибором для різних потреб та задач. Вона також відома своєю великою кількістю бібліотек і фреймворків, що полегшують розробку та розширюють функціонал додатків.

Kotlin, натомість, є новішою мовою програмування, яка набирає популярність серед розробників Android додатків. Вона пропонує безпеку типів, розширений функціонал та зручний синтаксис, що робить код більш безпечним і зрозумілим. Kotlin також повністю сумісний з Java, що дозволяє поступово переходити до нової мови, не втрачаючи сумісності з існуючим кодом.

У даному проекті обираючи Java для розробки додатку на Android, було враховано її популярність та широкий досвід використання. Також, одним із важливих аспектів вибору, було те, що вже були отримані деякі базові знання цієї мови. Крім того було зрозуміло, що наявність великої кількості бібліотек і фреймворків для Java значно полегшить розробку функціоналу додатку.

## 1.3 Інтеграція IoT з Android Studio

На сьогоднішній день IoT набуває дуже великої популярності та затребуваності. Інтернет речей (IoT) – це концепція, яка описує мережу фізичних об’єктів (пристроїв, автомобілів, датчиків тощо), які обладнані технологіями збирання та обміну даними через Інтернет. IoT дозволяє цим об’єктам збирати та обмінювати даними, що робить їх «розумними» та дозволяє взаємодіяти з їхнім середовищем або іншими пристроями. Іншими словами IoT, це взаємодія фізичних пристроїв з програмами, кодами або іншими пристроями.

Зв’язок додатку на Android з IoT, або, якщо сказати інакше інтеграція зовнішніх датчиків у додатки значно розширює функціональність та можливості мобільних додатків. Датчики можуть бути різноманітними, включаючи датчики руху, температури, вологості, світла тощо.

Налаштування інтеграції додатку із зовнішнім пристроєм відбувається на фізичному і програмному рівні. Для початку необхідно з’єднати мобільний пристрій з зовнішнім датчиком за допомогою Bluetooth, USB або Wi-Fi залежно від типу датчика та його способу підключення. Для підключення датчику до мобільного додатку потрібно використовувати API. Android надає ряд API для роботи з різними типами датчиків. Для зчитування даних з датчиків використовуються методи, які надаються відповідними класами API. Отримані дані з датчика обробляються в додатку згідно з вимогами функціональності. Це може включати аналіз даних, відображення їх на інтерфейсі користувача або виконання певних дій залежно від значень датчика. Також, ці дані можуть використовуватися для взаємозв’язку з іншими компонентами додатку, такими як візуалізація даних, сповіщення користувача, взаємодія з веб-службами тощо.

Конкретно в Android Studio інтеграції з IoT відбувається завдяки платформі IoT-Ignite. Для того, щоб створити проект IoT-Ignite слід дотримуватися стандартного алгоритму, який буде описано нижче. Для початку потрібно відкрити Android Studio, створити та налаштувати проект (рис. 1.3). В

галереї вибору активностей вибираємо пристрій, для якого розробляється додаток та пусту активність.

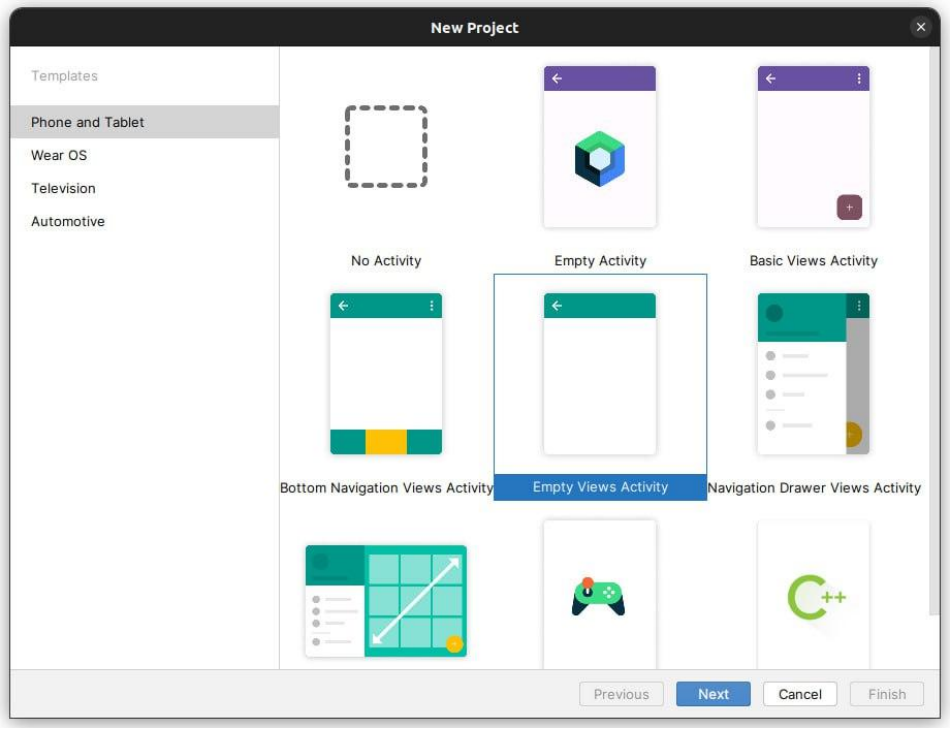


Рисунок 1.3 – Створення та налаштування проекту

У наступному вікні необхідно вибрати рівень API, на який орієнтується проект та мову програмування (рис. 1.4). У поле «Application name» вводимо ім’я проекту та вибираємо розміщення проекту. Решту налаштуваннь залишаються незмінними.

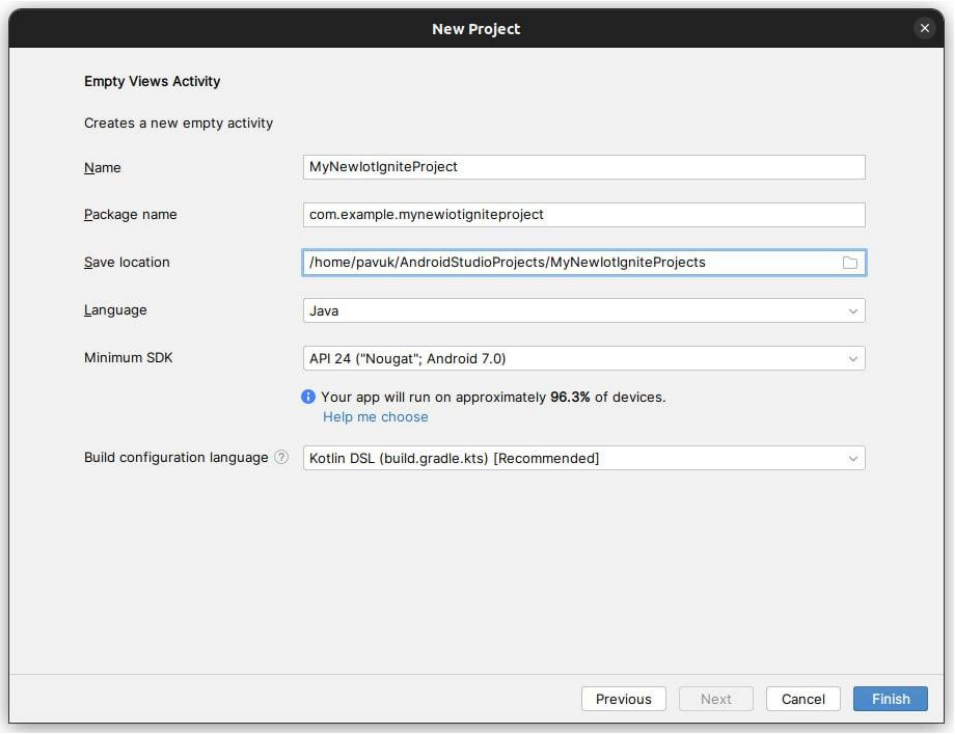


Рисунок 1.4 – Вибір рівню API та мови програмування

Далі розпочнеться процес завантаження залежностей Android Studio у проект. Це займе деякий час. Коли збірку Gradle буде завершено, знизу у вікні буде відповідне повідомлення. У нижній частині вікна з'явиться відповідне повідомлення. Потім слід налаштувати репозиторії (рис. 1.5) та залежності (рис. 1.6) для Інтернету речей (IoT) до модуля app:build.gradle.

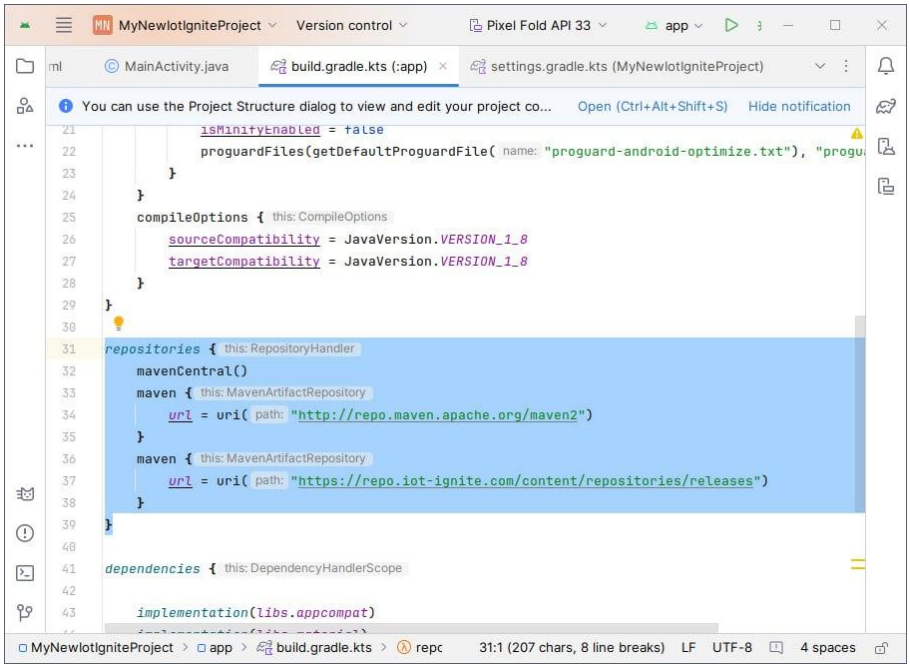


Рисунок 1.5 – Налаштування репозиторіїв

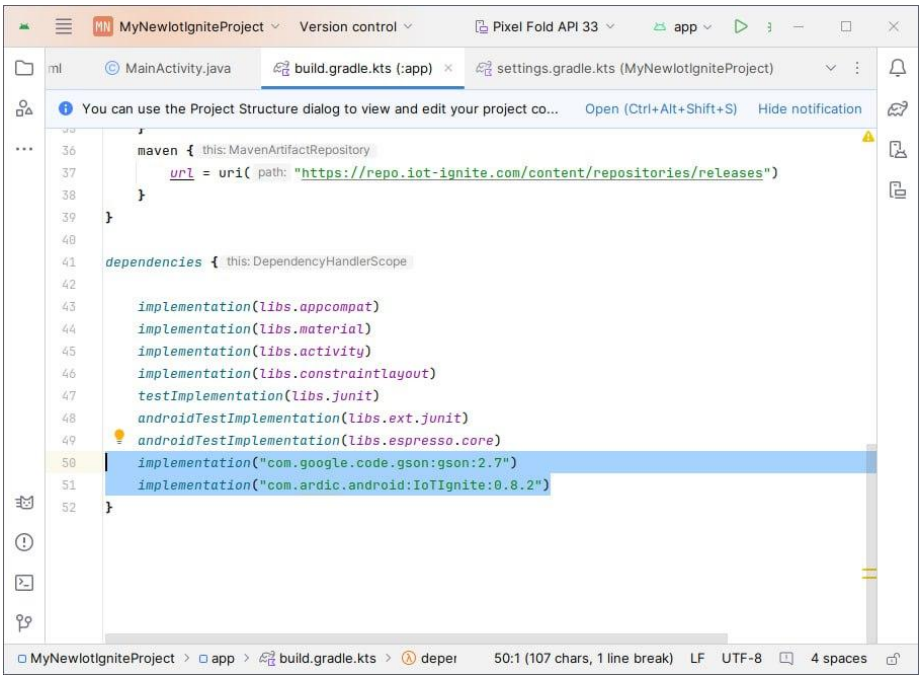


Рисунок 1.6 – Налаштування залежностей

Після завантаження нових залежностей проект повністю готовий для підключення зовнішніх датчиків та початку роботи з ними. IoT-Ignite дає обширний функціонал для роботи зі всіма видами зовнішніх пристроїв.

Отже, Android Studio вдало інтегрується з IoT за допомогою IoT-Ignite. Ця платформа забезпечить безпеку даних та взаємодію, обробку і реагування на дані пристроїв, навіть коли вони офлайн. Тобто Android Studio та IoT-Ignite повністю гарантують мені розробку додатку для платформи Android з використанням Інтернет речей.

# РОЗДІЛ 2
# МЕТОДИ ПОБУДОВИ ТРЕНУВАЛЬНИХ ДОДАТКІВ ТА ІНСТРУМЕНТИ РОЗРОБКИ

## 2.1 Особливості розробки і вибору інструментів

Можливості розробки додатків мовою Java у Android Studio є досить широкими. Програмування додатку таким способом складається з двох частин. Перша частина передбачає створення макету XML, який складається з різних контейнерів, текстових, графічних та інших елементів. Друга частина – це кодування. У Android Studio об’єднання макету і коду називається активністю, яку можна створити через графічний інтерфейс. З цього починається створення додатку.

Додаток Emergency App, який є предметом дослідження даної кваліфікаційної роботи, використовує 10 активностей, 2 діалогових вікна, 114 графічних зображень, 17 файлів, 24 файли анімацій, 1 файл маніфесту, 10 звукових файлів, 2 файли шрифтів та інші проектні файли.

Макети XML – це файли, які використовуються для оформлення зовнішнього вигляду користувацького інтерфейсу додатка. У них описується розміщення та взаємодія різних елементів і компонентів інтерфейсу, таких як кнопки, тексти, зображення, списки тощо. Макети XML дозволяють визначати, як буде відображатися контент на екрані пристрою для різних розмірів екрану і орієнтацій пристрою. Крім того, вони підтримують можливість встановлення параметрів стилю і вигляду елементів, щоб створювати зручний і привабливий інтерфейс користувача.

У даному додатку використовуються елементи макету XML: текстові TextView, графічні ImageView, контейнери RelativeLayout та LinearLayout, кнопки Button. Кожен з цих елементів має свої атрибути, такі як позиція, розміри, вирівнювання тощо. Цих атрибутів є безліч і для кожного елемента вони можуть відрізнятися.

Частина, яка відповідає за кодування, представлена файлами Java. У таких файлах є стандартний перелік елементів, з яких вони складаються. Це

можуть бути імпортовані класи, змінні, методи, параметри та атрибути (рис. 2.1). Загалом ці елементи схожі на інші елементи інших мов, однак у мові Java є свої особливості структури, синтаксису та використання.

При використанні будь-якого елемента макету у Java-файлах, Android Studio пропонує імпортувати необхідні для роботи класи. Імпорт може відбуватися автоматично або вручну. Для ручного імпорту класу потрібно спочатку ввести слово «import», після чого вводиться пакет, з якого імпортується клас, та назва самого класу.


Рисунок 2.1 – Структура файлів Java

В «Emergency App» використовуються 35 імпортованих класів, починаючи з базових і закінчуючи унікальними. Кожен з цих класів має своє призначення та важливість для проекту. Було розглянуто і описано кожен з цих класів та їх призначення:

- android.Manifest – клас, який надає доступ до функцій, пов’язаних з дозволами;

- android.app.Activity – базовий клас для усіх активностей в додатку;
- android.app.Dialog – клас для створення діалогових вікон;
- android.content.ClipData – клас, який представляє набір даних, які можуть бути скопійовані або переміщені в обмінний буфер;
- android.content.ClipDescription – клас, який містить опис даних, які знаходяться в обмінному буфері;
- android.content.DialogInterface – інтерфейс, який використовується для взаємодії з діалоговими вікнами;
- android.content.Intent – клас, який використовується для здійснення зв’язку між різними компонентами додатку;
- android.content.pm.PackageManager – клас, який надає доступ до пакетних операцій, таких як отримання інформації про встановлені додатки;
- android.graphics.Bitmap – клас для роботи з зображеннями у форматі бітмап;
- android.graphics.Canvas – клас для малювання графіки на певній поверхні;
- android.graphics.Color – клас для роботи з кольорами;
- android.graphics.drawable.ColorDrawable – клас, який представляє одноцільовий фон;
- android.net.Uri – клас, який представляє ідентифікатори ресурсів, таких як файли або контакти;
- android.os.Bundle – клас, який використовується для передачі даних між активностями;
- android.os.Environment – клас, який надає доступ до середовища виконання;
- android.text.Editable – клас, який представляє редагований текст;
- android.text.TextWatcher – інтерфейс, який використовується для відстеження змін у текстових полях;
- android.view.DragEvent – клас, який представляє події перетягування;

- android.view.View – базовий клас для всіх елементів інтерфейсу користувача;
- android.view.Window – об’єкт, який представляє вікно додатка;
- android.view.WindowManager – клас, який надає доступ до параметрів відображення вікна;
- android.widget.Button – елемент інтерфейсу користувача для відображення кнопки;
- android.widget.EditText – елемент інтерфейсу користувача для введення тексту;
- android.widget.ImageView – елемент інтерфейсу користувача для відображення зображення;
- android.widget.RadioButton – елемент інтерфейсу користувача для відображення кнопки вибору;
- android.widget.RadioGroup – група елементів інтерфейсу користувача для відображення кнопок вибору;
- android.widget.RelativeLayout – контейнер для розміщення дочірніх елементів відносно інших елементів або батьківського контейнера;
- android.widget.TextView – елемент інтерфейсу користувача для відображення тексту;
- android.widget.Toast – клас для відображення коротких повідомлень користувачеві;
- androidx.activity.EdgeToEdge – клас для роботи з відображенням елементів на всю площину екрану;
- androidx.appcompat.app.AlertDialog – діалогове вікно підтримки діалогів з версії підтримки бібліотеки AppCompat;
- androidx.appcompat.app.AppCompatActivity – базовий клас для активностей в додатках з підтримкою діалогових вікон;
- androidx.core.app.ActivityCompat – клас для допомоги в роботі з дозволами у додатках;
- androidx.core.content.ContextCompat – клас для доступу до ресурсів додатка;

- androidx.core.content.FileProvider – провайдер файлів для доступу до файлів додатка через URI.

Після імпорту класів, в файлах Java, відбувається оголошення змінних. Оголошення відбувається наступним чином: спочатку вводиться модифікатор доступу змінної, потім тип, потім назва змінної. Модифікатори доступу є різними, але в моєму проекті використовується тільки private. Цей модифікатор оголошує змінну тільки в межах файлу чи методу, в залежності від того, де вона оголошується.

Тип змінних Java також є досить обширним поняттям. Він визначає тип даних, який може бути збережений у цій змінній. Також, він визначає обмеження на значення, які можуть бути призначені цій змінній, а також операції, які можна виконати над цими значеннями. Існує дві категорії типів змінних: примітивні типи (int, double, char, boolean та інші) і посилальні типи. Посилальні типи – це типи даних, які вказують на об’єкти у пам’яті, а не на самі значення. Посилальні типи включають класи, інтерфейси, масиви та інші посилальні типи даних.

«Emergency App» використовує наступні типи даних: int, ImageView, TextView, MediaPlayer, String, EditText, Dialog, Button, Bitmap, Canvas, File, Uri, Intent, ClipData, ClipDescription, View, Window, ViewCompat, WindowInsetsCompat, Insets, FileOutputStream, PackageManager, AlertDialog, AlertDialog та FileOutputStream. Ці типи містяться в імпортованих класах і відповідають елементам, які їх використовують.

Наступним елементом у файлах Java є використовувані методи. Методи – це фрагменти коду, які виконують певні дії. Вони можуть приймати аргументи, виконувати обчислення, змінювати стан об’єктів та повертати значення. Методи допомагають організувати програми, роблять код більш читабельним та підтримують принципи модульності та розширюваності.

У проекті «Emergency App» використовуються наступні методи: onCreate(Bundle) (ініціалізація компонентів і встановлення початкових значень для активності), onClick(View) (встановлення обробника подій для кнопок або інших віджетів, виклик при натисканні), startDragAndDrop(null, shadowBuilder,

view, 0) (початок операції перетягування для віджета), setSystemUiVisibility(int) (встановлення видимості системного інтерфейсу), setContentView(int) (встановлення вмісту віджету для відображення на екрані активності), setFlags(int, int) (встановлення прапорців для вікна активності), findViewById(int) (пошук віджета за його ідентифікатором в макеті XML), setOnLongClickListener(View.OnLongClickListener) (встановлення обробника подій для тривалого натискання на віджет), setOnClickListener(View.OnClickListener) (встановлення обробника подій для натискання на віджет), setText(CharSequence) (встановлення текстового змісту віджета), show() (Відображення діалогового вікна на екрані) та setVisibility(int) (встановлення видимості віджета відповідно до зазначеного значення видимості).

Кожен з методів має свої параметри та атрибути. Параметри та атрибути – це деталі, що визначають поведінку об’єктів у програмі або їхні властивості. Вони уточняють роботу методів, дають більш детальні вказівки. В проекті було використано наступні параметри та атрибути: layout_width, layout_height, id, src, visibility, text, onClick, FLAG_FULLSCREEN, FLAG_HIDE_NAVIGATION, SYSTEM_UI_FLAG_FULLSCREEN,SYSTEM_UI_FLAG_HIDE_NAVIGATION, BACKGROUND_COLOR, activity, theme, round, gravity, centerInParent, centerHorizontal, centerVertical, wrap_content, match_parent, invisible, visible, ЯUTYPE, drop, ACTION_DROP, ACTION_SEND, EXTRA_STREAM, TRANSPARENT, provider, OVER_SCROLL_NEVER, title, button, ARGB_8888, CompressFormat, flush, close та багато інших.

### 2.2 Алгоритм візуального функціонування додатку

Додаток «Emergency App» є досить обширним додатком. Він містить головне меню, меню рівнів та різні етапи проходження рівня 1 «Аварія на хімічній станції». Програмування візуального функціонування додатка здійснювалося за попередньо розробленим алгоритмом. Цей алгоритм містить у

собі покрокові функціональні можливості додатка, які користувач зможе використовувати у процесі проходження.

Візуальне функціонування додатка можна розділити на такі етапи: головне меню, меню рівнів, початкова активність із ознайомленням і завданням, вибір джерел інформації, герметизація дверей, герметизація вікон та вентиляції, створення засобу захисту систем дихання, тестування правильного вибору одягу для виходу на вулицю, збір тривожного ранця та отримання сертифіката про проходження рівня 1. Кожен з цих етапів має свої функціональні можливості. У ході виконання роботи було розроблено алгоритм візуального функціонування для кожного з етапів.

Головне меню гри розроблене за алгоритмом, який зображено на рисунку 2.2. При запуску головного меню додатка, користувачеві повинні бути доступні наступні функції: перехід у меню вибору рівнів, вихід з додатка, виклик/закриття інформаційних панелей. Також у панелі інформації про автора повинна бути функція переходу на соціальну мережу автора додатка.


Рисунок 2.2 – Алгоритм візуального функціонування меню додатку

На рисунку 2.3 представлено візуальний вигляд елементів, які відповідають за функції головного меню, представлені в алгоритмі. Він містить елемент початку гри, елемент виходу з додатку, три елементи інформаційних

панелей, підложка для елементів меню та елемент для переходу на соціальну мережу. Всі ці елементи є графічними елементими для відображення візуального функціонування додатку.

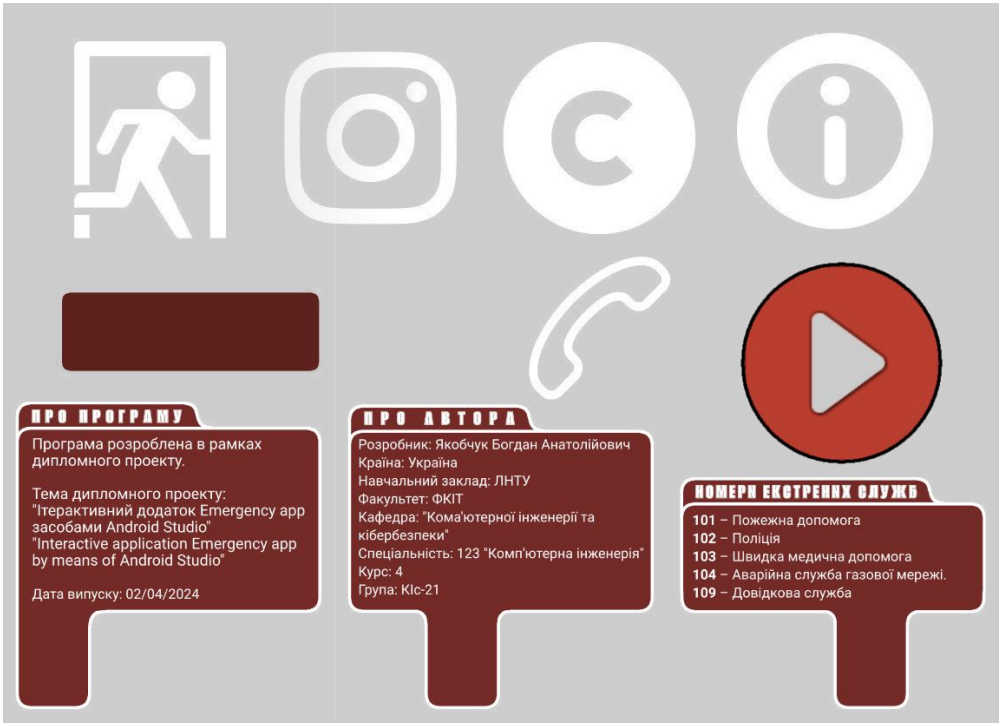


Рисунок 2.3 – Графічні елементи для головноного меню додатку

Наступний алгорим розроблений для реалізації меню вибору рівнів (рис. 2.4). Він містить функцію для переходу назад до головного меню та функцію запуску наступної активності.


Рисунок 2.4 – Алгоритм візуального функціонування меню вибору рівнів

Для реалізації функцій описаних вище, в меню вибору рівнів, було створено графічні елементи засобами Photoshop, а саме елемент для переходу до попередньої активності та елемент для запуску рівня 1 (рис. 2.5).


Рисунок 2.5 – Графічні елементи для меню вибору рівнів

Алгоритм запуску першого рівня, або ж алгоритм початкової активності із ознайомленням і завданням (рис. 2.6), є дещо складнішим ніж попередні. Після запуску активності викликається діалогове вікно з описом рівня та можливістю продовження чи закриття рівня. При закритті діалогового вікна відбувається перехід до попередньої активності, а при підтверджені – остаточний запуск рівня. Після закриття діалогового вікна відкривається активність 1 рівня з початком сценарію.


Рисунок 2.6 – Алгоритм візуального функціонування початкової активності із ознайомленням і завданням

Для реалізації активності за даним алгоритмом були створені наступні грфічні елементи (рис. 2.7): елемент закриття діалогового вікна, елемент переходу від діалогового вікна до рівня, елемент переходу від активності з ознайомленням та завданням до наступної активності. Саме діалогове вікно не є графічним зображенням, а являє собою адаптивний, запрограмований і зверстаний графічний контейнер. Графічний елемент «стрілки» є стандартним для усіх наступних активностей. Він слугує для переходу між підказками та активностями.


Рисунок 2.7 – Графічні елементи для активності ознайомлення із завданням

Наступна активність мала на меті навчити користувача правильно обирати джерела інформації при екстреній ситуації. Алгоритм, розроблений для опису візуальних функцій даної активності (рис. 2.8) включає в себе: функції ввімкнення/вимкнення телевізору, функції відкриття/закриття дверей, функція

користування голосовим помічником, функція переходу між підказками. Візуальні функції голосового помічника будуть описані нижче.


Рисунок 2.8 – Алгоритм візуального функціонування активності вибору джерел інформації

Елементи для реалізації візуальних функціональностей активності вибору джерел інформації (рис. 2.9) включають в себе: елемент телевізору, елемент виклику голосового помічника, елемент для відкриття/закриття дверей.

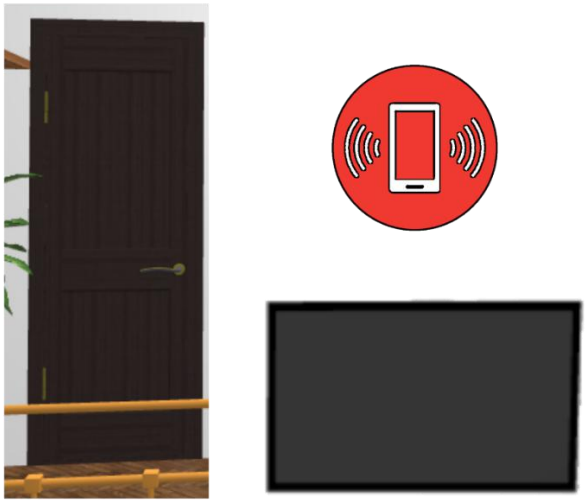

Рисунок 2.9 – Графічні елементи для активності вибору джерел інформації

Для актиності герметизації дверей був створений алгоритм (рис. 2.10), який містить функціональності імітації змочення ганчірок, закриття дверей мокрим покривалом та закриття щілин мокрими ганчірками. Для реалізації даного алгоритму було створено спеціальні графічні елементи.


Рисунок 2.10 – Алгоритм візуального функціонування активності геризації дверей

Герметизація вікон та вентиляції відбувається за таким же послідовним алгоритмом, як і герметизація дверей і не потребує додаткового опису. Для імітації герметизації було використано графічний елемент скотчу.

Для наступної активності, функціональність якої є створення засобу захисту систем дихання, було створено алгоритм, який зображено на рисунку 2.11. Цей алгоритм містить функції імітації створення йодованого розчину, вимочення вати у розчині, вимочення маски у розчині, додавання вати у маску.

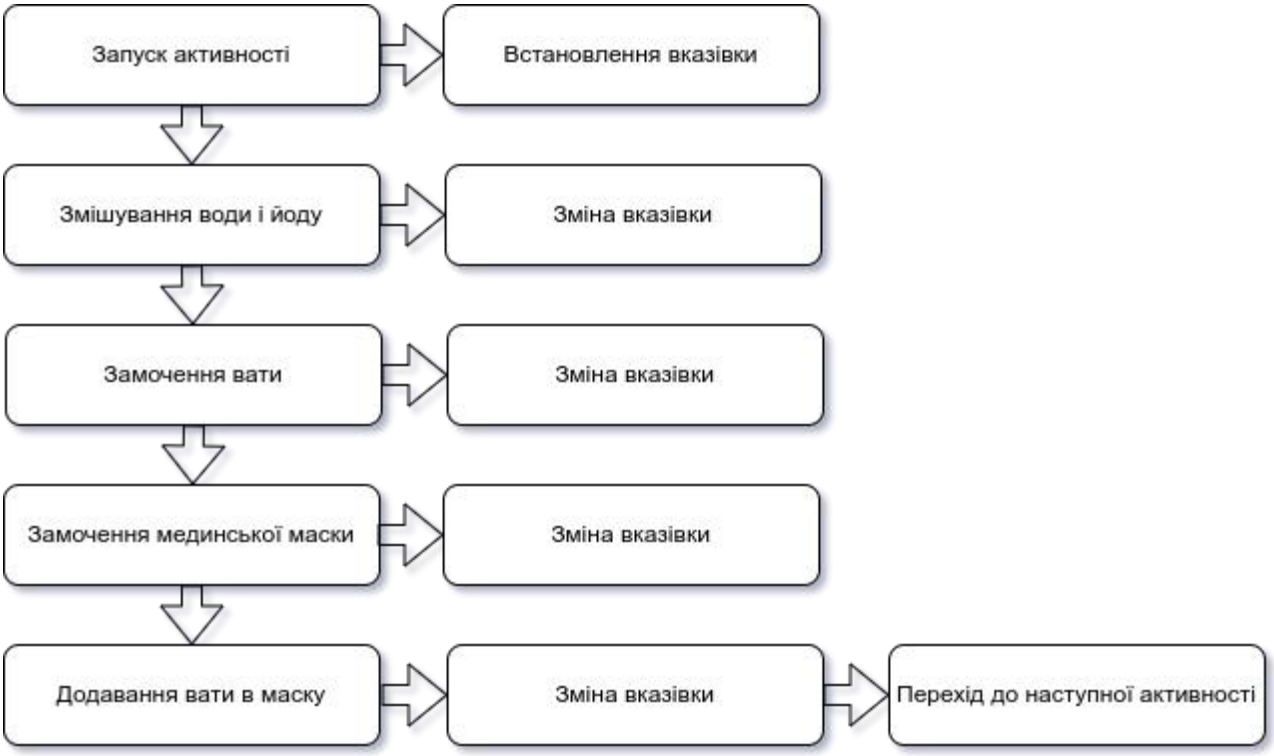


Рисунок 2.11 – Алгоритм візуального функціонування активності створення засобу захисту систем дихання

Для створення візуального функціонування активності були створені графічні елементи (рис. 2.12), а саме: чиста вати, йодована вата, банка чистої води, банка з йодованим розчином, йод, чиста маска, маска вимочена в йодованому розчині.

Рисунок 2.12 – Графічні елементи для активності створення засобу захисту систем дихання

Для активності тестування правильного вибору одягу було створено алгоритм для одного з питань(рис. 2.13), який описує варіанти вибору відповідей та результати цих відповідей. За таким самим принципом реалізовано функціонування для решти питань. Завдання цієї активності не просто перевірити знання, але й проводити тестування до тих пір, доки правильна відповідь не буде вибрана. При виборі кожного варіанту відповіді відбувається реакція, яка включає в себе підсвічення елементу червоним або зеленим кольором, залежить від того чи правильна відповідь чи ні, а також вивід текстової підказки щодо вибраної відповіді.

Для реалізації активності за алгоритмом описаним вище було створено елементи тестів (рис. 2.14). Ці елементи містять не тільки зображення предметів тестування, але й зображення цих елементів із підсвіткою. При виборі правильного варіанту відповідей правильний предмет тестування підсвічується зелени, а всі інші, неправильні варіанти відповідей підсвічуються червоним. У варіанті коли користувач спершу вибирає неправильний варіант відповіді, предмет відповіді стає червоним.

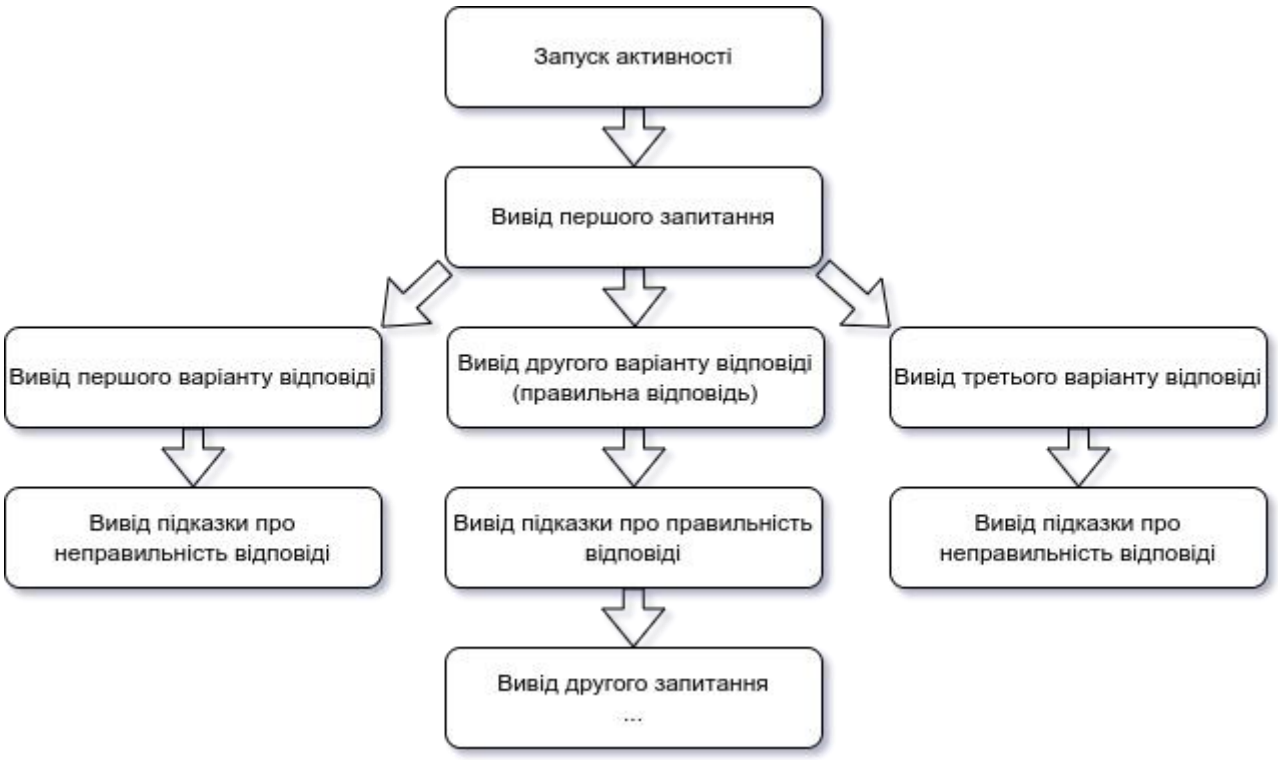

Рисунок 2.13 – Алгоритм візуального функціонування активності тестування вибору правильного одягу

Рисунок 2.14 – Графічні елементи для активності тестування правильного вибору одягу

Наступна активність збору ранцю немає особливих візуальних елементів. Дії у ній відбуваються паралельно, тобто користувач може у будь-якій послідовності перемістити речі до рюкзаку. Після того як остання річ переміщена відкривається перехід до останньої активності – отримання сертифікату. Алгоритм візуального функціонування активності отримання сертифіката містить дії вводу прізвища та ім’я та функції поширення сертифіката.

В основному, для візуального функціонування додатку «Emergency App», було використано графічні елементи розроблені засобами Photoshop. Однак, якщо говорити про повне функціонування візуальних елементів взаємодії користувача з додатком, то важливо згадати про стан активних елементів. Для кожного елемента при натисненні змінюється стиль зображення. На рисунку 2.15 зображено візуальний приклад такої взаємодії. Зміна стилю картинки відбувається шляхом заміни одного зображення на інше через файли

стилів. Код, який відповідає за цю зміну містить посилання на дві картинки і умову, щоб при натисненні на кнопку, стан state_pressed, картинка мінялася. Ознайомитись з цим кодом можна на лістингу 2.1.

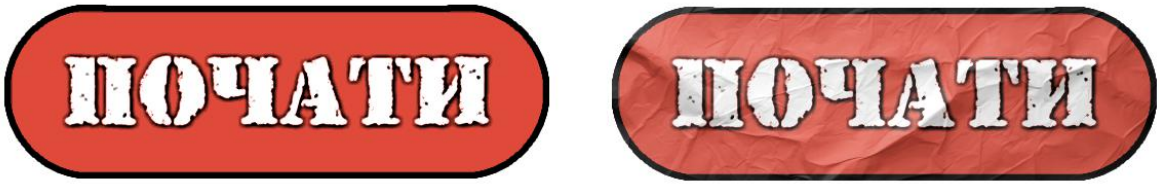


Рисунок 2.15 – Приклад відображення зміни графічних елементів взаємодії при натисненні

Лістинг 2.1 – Код, який відповідає за зміну картинки при натисненні

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/continue_btn_pressed" android:state_pressed="true" />
    <item android:drawable="@drawable/continue_btn" />
</selector>
```

Кінець лістингу 2.1

Алгоритм візуального функціонування додатку є об’ємним, однак, підсумовуючи, додаток містить багато візуальних елементів, графічних елементів, адаптивних блоків, анімацій, що суттєво робить вигляд додатка більш привабливим. Елементи взаємодії зроблені таким чином, що вони не зливаються із заднім планом та мають чітке положення і зрозуміле призначення, що залишає приємний досвід від користування додатком.

### 2.3 Налагодження роботи додатка

Перед тим, як випустити додаток для загального користування, потрібно повністю налагодити його роботу. Це означає, що програма повинна відкриватися та коректно працювати на будь-якому смартфоні операційної системи Android. Сюди можна віднести не тільки код який, який відвідає за

функціонування додатку, але й файли розміток, які вказують на положення та властивості об’єктів керування.

Як згадувалося вище, в Android Studio програма складається з активностей, тобто вікон програми. Однак просто створити активність недостатньо для того, щоб вона відображалася у додатку. Для цього потрібно додати її до файлу маніфесту.

Файл AndoidManifest.xml – це файл у якому міститься інформація для системи Android. Обсяги цієї інформації залежать від функціональності додатку. Він може містити дозволи на користування іншими компонентами смартфону, описує вікна програми, сторонні сервіси, вказує на імена класів, які реалізовують активності програми, визначає процеси, які компоненти програми використовують, вказує мінімальний рівень інтерфейсу API та містить список бібліотек, які додаток використовує під час роботи. Кожна програма повинна містити файл маніфесту, саме під таким ім’ям, в кореневій директорії проекту. Основне призначення цього файлу – дати для системи Android інформації про програму ще до її запуску. Тобто, при збірці та запуску програми файл маніфесту читається першим.

В додатку «Emergency App» файл маніфесту містить наступні елементи структури, або ж теги: <uses-permission>, <application>, <provider>, <activity>. Разом ці теги забезпечують коректну роботу додатку. Кожен з них має свої правила написання, структуру та вміст.

Тег <application> – це тег, який описує саму програму, її назву, значок, дозвіл на резервне копіювання, тощо. Саме в цьому елементі було вказано шлях до файлу іконки, який відображує програму в смартфоні.

Тег <uses-permission> надає користувачам дозволи для інших компонентів, програм та сервісів. У файлі маніфесту додатку «Emergeny App» цей тег використовується для отримання доступу до зовнішнього сховища. Відбувається це через функцію android permission, яка називається WRITE_EXTERNAL_STORAGE. Цей дозвіл дозволяє зберігати сертифікат або поширювати його, після проходження рівню.

Тег <provider> слугує для оголошення компоненту постачальника контенту. Тобто, у додатку цей тег використано для того, щоб оголосити всі файли ресурсів та шлях до їхнього перебування. Без цього сам додаток не використовував би їх при збірці.

Тег <activity> є тегом, який призначений для оголошення активностей програми. Тільки ті активності, які записані в файлі маніфесту будуть доступні при запуску програми. У файлі програми «Emergency App» описано 7 таких активностей. При цьому у кожній з них є додаткові атрибути, які вказують чи можна експортувати активність окремо та атрибути орієнтації екрану. Також, серед цих активностей визначено головну MainActivity. На те, що активність головна, вказує один з фільтрів тегу <intent-filter>, а саме дія android.intent.action.MAIN. Головна активність у будь-якому додатку може бути тільки одною, і запускається вона першої після файлу маніфесту, при запуску додатку.

Враховуючи всю важливість файлу маніфесту, було оглянуто його функції і призначення саме в додатку «Emergency App» та обґрунтовано те, що цей файл сприяє узгодженню взаємодії додатку з операційною системою Android. Він знайомить систему з інформацією про додаток та ресурсами, які будуть використовуватися у додатку, на програмному рівні.

Однак, для повного налагодження роботи додатку на всіх смартфонах операційної системи Android, важливим є розміщення елементів керування таким чином, щоб вони не були прив’язані до конкретної точки екрану, адже смартфони мають різну ширину, висоту та роздільну здатність. В ході розробки додатку було створено багато таких елементів. Основною особливістю цих елементів є відносні розміри та розташування, яке прив’язане до рамок екрану, або до горизонтальної чи вертикальної ліній. Таке розміщення забезпечує адаптивне відображення для будь-яких розмірів екрану.

Розміщення елементів на екрані активності відбувається через файли розмітки XML. Для того, щоб якось повпливати на елемент потрібно написати тип елементу та написати атрибути. Правила написання елементів та аритбутів відображено на рисунку 2.17 У кожного елемента обов’язково повинна бути

висота і ширина, на що призначені атрибути layut_height та layout_width відповідно. Після написання тегу пишеться двокрапка та значення в лапках. Для висоти та ширини значення може бути або фіксованим або відносним. Фіксоване значення вказується в одній з одиниць вимірювання, у нашому випадку використано dp. Одиниця вимірювання dp – це абстрактна одиниця, яка дозволяє однаково виглядати на різних розмірах екрану. Тобто, якщо вказати висоту та ширину в dp, при визаченні розмірів елементів, це забезпечить коректне відображення на всіх розмірах екранів.

```
<Елемент
    ім'я_атрибуту: “значення”
/>
```

Рисунок 2.17 – Правила написання елементів розмітки та атрибутів

Функціональні елементи, тобто ті, які мають певний код, було використано атрибут id. Він надає для елемента унікальне ім’я в межах активності. Це ім’я буде використано при оголошенні елементу в файлі з кодом Java.

Інші атрибути, в залежності від типу елементів можуть відрізнятися. Оскільки, у додатку «Emergency App», основним, найбільш використовуваним типом, є зображення (ImageView), то доречно описати атрибути, які було використано саме для цього типу.

Для фонових зображень, які не мають ніякого функціонального призначення було використано атрибути scaleType та src. Атрибут scaleType призначений для створення розмірного стилю зображень. У додатку для фонових зображень було використано значення centerCrop, що вирівнює картинку по центру і обрізає зображення так, щоб ні одна частина екрану не була пустою. Атрибут src вказується для будь-якого зображення. Він містить відносний шлях до файлу, яким є це зображення.

Оскільки, деякі елементи мають однакове призначення, і повинні виводитися групами, то у додатку використано спеціальні контейнери. Ці

контейнери є невидимими. Їх призначення обмежувати та структурувати контент. Головною їх особливістю є те, що при створені можна вказати атрибути не тільки до елементу контейнера, але і до елементів які знаходяться всередині, наприклад, вивести елементи в одни ряд по три стовпці. Це значно прискорює та удосконалює налагодження коректної робти у додатку. При потребі ми можемо вказати фонове зображення або колір для контейнера, що зробить його видимим.

Інші зображення, які є кнопками, мають додаткові атрибути, які дозволяють елементам правильно розміщуватися один між одним. До використовуваних у додатку можна віднести: padding, margin, paddingTop, paddingBottom, paddingLeft, paddingRight, marginTop, marginBottom, marginLeft, marginRight. Всі ці атрибути забезпечують зовнішні та внутрішні відступи для зображень, та вказуються у dp.

Отже, було проведено налагодження роботи додатка. В процесі було узгоджено взаємодію додатку з операційною системою Android та взаємодію користувача з додатком. Розташування елементів керування реалізовано таким чином, щоб вони завжди були доступними на екрані, не накладалися та відображалися коректно на будь-яких розмірах екрану смартфонів.

# РОЗДІЛ 3
# РОЗРОБКА ПРОГРАМИ «EMERGENCY APP»

## 3.1 Програмування меню і рівнів

При запуску додатку, меню відіграє ключову роль, адже це перше, що користувач бачить. Тобто, меню виступає в ролі титульної сторінки додатку. Однак, аналізуючи готові дослідження та відгуки користувачів, варто зазначити, що є декілька аспектів, яких слід дотримуватися при конструюванні меню: простота та зручність, послідовність, легкий доступ, анімації та візуальні ефекти, групування та правильна організація елементів.

Враховуючи це, було створене меню, яке відповідає усім вимогам (рис. 3.1). Це меню містить в центрі велику кнопку, а знизу групу елементів різної функціональності. Велика кнопка в центрі відповідає за перехід до наступної активності.

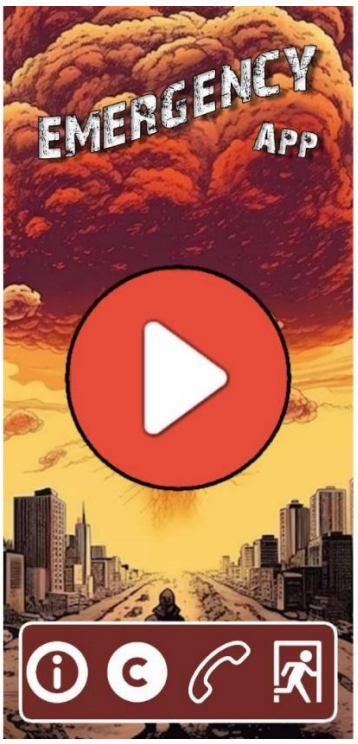


Рисунок 3.1 – Зовнішній вигляд головного меню додатку

Для того, щоб реалізувати перехід з головного меню додатку до наступної активності було використано Java-код, який зображено на

лістингу 3.1. У цьому коді спочатку йде звернення до об’єкту по id та присвоєння йому методу setOnClickListener. Це означає, що код спрацює тоді, коли на кнопку буде натиснуто. В фігурних дужках спочатку створюється намір перейти до наступної активності, а потрім відбувається сам перехід.

Лістинг 3.1 – Кодування кнопки почати головного меню

```
39          imgPlay.setOnClickListener(new View.OnClickListener() {
40              @Override
41              public void onClick(View view) {
42                  // Перехід до активності з вибором рівнів
43                  Intent intent = new Intent( packageContext: MainActivity.this, LevelsActivity.class);
44                  startActivity(intent);
45              }
46          });
```

Кінець лістингу 3.1

Для елементу виходу було написано код, який повністю закриває додаток. Цей код спрацьовує при натисненні та використовує функцію finishAffinity для закриття всіх активностей. Повний код для кнопки виходу можна побачити на лістингу 3.2.

Лістинг 3.2 – Кодування кнопки виходу з додатку

```
48          imgExit.setOnClickListener(new View.OnClickListener() {
49              @Override
50              public void onClick(View view) {
51                  // Закриття всіх активностей програми
52                  finishAffinity();
53              }
54          });
```

Кінець лістингу 3.2

Інші три елементи нижньої панелі головного меню відповідають за виведення довідкових вікон (рис. 3.2). Перший елемент виводить інформацію про програму. Другий елемент виводить інформацію про автора та додатково містить елемент переходу на профіль соціальної мережі «Instagram». Третій елемент містить у собі довідкову інформацію про номери екстрених служб.


Рисунок 3.2 – Зовнішній вигляд викликаних довідкових вікон головного меню

Ці три елементи запрограмовані по одному принципу. На лістингу 3.3 описано код, який відповідає за функціонування першого діалогового вікна. В цьому коді описано, що при натисненні на перший елемент відбувається перевірка, чи перше довідкове вікно ще не викликане. Якщо вікно вже викликане, тоді кнопка закриває перше вікно. В іншому випадку, якщо довідкове вікно не викликане, тоді при натисненні на перший елемент воно викликається. Виклик вікна чи приховування його відбувається через надання стану VISIBLE або INVISIBLE.

Лістинг 3.3 – Приклад кодування виклику довідкових вікон

```
56          imgAboutProgram.setOnClickListener(new View.OnClickListener() {
57              @Override
58              public void onClick(View v) {
59                  if (menuAboutProgram.getVisibility() == View.VISIBLE) {
60                      // Якщо картинка меню вже видима, то ховаємо всі картинки меню
61                      menuAboutProgram.setVisibility(View.INVISIBLE);
62                      menuAboutAuthor.setVisibility(View.INVISIBLE);
63                      menuCall.setVisibility(View.INVISIBLE);
64                      insta.setVisibility(View.INVISIBLE);
65                  } else {
66                      // Інакше показуємо картинку меню про програму та ховаємо інші
67                      menuAboutProgram.setVisibility(View.VISIBLE);
68                      menuAboutAuthor.setVisibility(View.INVISIBLE);
69                      menuCall.setVisibility(View.INVISIBLE);
70                      insta.setVisibility(View.INVISIBLE);
71                  }
72              }
73          });
```

Кінець лістингу 3.3

Однак для зручності і коректного виводу всіх вікон, при натисненні на один з елементів виводу довідкових вікон, було додано код, щоб решту вікон ховалися. Це забезпечує швидку та правильну навігацію. Користувачеві не приходиться спочатку закривати одне вікно для виводу іншого, це відбувається автоматично.

Як згадувалося раніше, довідкове вікно про автора містить кнопку посилання на соціальну мереже автора. Для реалізації такого функціоналу було розроблено код для кнопки соціальної мережі. Його представлено на лістингу 3.4. Цей код спрацьовує при натисненні на іконку «Instagram». Він містить посилання на соціальну мережу, пакет типу посилання, намір перейти за посиланням та сам перехід. Також, запрограмована додаткова умова, щоб якщо в смартфоні встановлено додаток «Instagram», то відбувається перехід туди, а якщо ні, то запускається перенаправлення через браузер.

Лістинг 3.4 – Код, який відповідає за перехід в профіль автора соціальної мережі «Instagram»

```
        insta.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String instagramProfileUrl = "https://www.instagram.com/b.yakobchuk_?igsh=cHA3cG9oYW5hZ3d0";
                Uri uri = Uri.parse(instagramProfileUrl);
                Intent intent = new Intent(Intent.ACTION_VIEW, uri);
                intent.setPackage("com.instagram.android");
                try {
                    startActivity(intent);
                } catch (ActivityNotFoundException e) {
                    // Якщо додаток Instagram не встановлено, відкриваємо Instagram у веб-браузері
                    intent.setPackage(null);
                    startActivity(intent);
                }
            }
        });
    }
}
```

Кінець лістингу 3.4

Наступною після активності головного меню є меню вибору рівнів. На момент написання даної кваліфікаційної робити, в додатку «Emergency App» розроблено тільки один рівень, але в майбутньому у додатку можливі зміни та удосконалення. Меню рівнів містить елементи повернення до головного меню та кнопку переходу до першого рівня.

Код для кнопки повернення, який відображено в лістингу 3.5 спрацьовує при натисненні на елемент та містить у собі намір переходу та сам перехід. Для кнопки старту першого рівня було розроблено аналогічний код. Його можна побачити на лістингу 3.6.

Лістинг 3.5 – Код для кнопки повернення до головного меню, меню рівнів

```
39          btnBack.setOnClickListener(new View.OnClickListener() {
40              @Override
41              public void onClick(View view) {
42                  // Повернення до головного меню
43                  Intent intent = new Intent( packageContext: LevelsActivity.this, MainActivity.class);
44                  startActivity(intent);
45              }
46          });
```

Кінець лістингу 3.5

Лістинг 3.6 – Код для кнопки початку першого рівня

```
29          btnLevel1.setOnClickListener(new View.OnClickListener() {
30              @Override
31              public void onClick(View view) {
32                  Intent intent = new Intent( packageContext: LevelsActivity.this, Level1.class);
33                  startActivity(intent);
34              }
35          });
```

Кінець лістингу 3.6

Підсумовуючи, у даному розділі було описано реалізацію меню та рівнів додатку «Emergency App» Ці активності відповідають сучасним стандартам користувацького інтерфейсу, забезпечуючи зручність, простоту та інтуїтивну зрозумілість для користувачів. Елементи керування розміщенні так, щоб не накладатися та не прив’язуватися до конкретної точки екрану. Розміщення елементів виконано груповим методом, так щоб при знайомлені користувач одразу розумів, що йому потрібно робити. Надалі можливе розширення функціональності додатку та додавання нових рівнів чи можливостей.

## 3.2 Створення інтерактивних елементів додатку

Для того, щоб зацікавити сучасного користувача та конкурувати з існуючими додатками на ринку, в додатку «Emergency App», було реалізовано багато різних інтерактивних елементів. Такі елементи дають змогу користувачеві більш реалістично зануритись у процес гри. Також, вони роблять додаток цікавішим.

Створення інтерактивних елементів вимагає знання спеціальних функцій та методів. В Android Studio функціонал таких елементів реалізовується мовою Java. Функціональність інтерактивних елементів може бути різною, залежить від стилю та напрямку додатку. Додаток «Emergency App» спрямований на те, щоб навчити користувача, як правильно діяти в екстрених ситуаціях, тому він включає різні типи інтерактивних елементів. До найпростіших з них, можна віднести функціонали увімкнення/вимкнення телевізору, відкриття/закриття дверей, покроковий вибір елементів тощо. Більш складний функціонал включає перетягування елементів та зміну стану елементу в залежності від позиції.

В першій активності, додатку «Emergency App» реалізовано два інтерактивних елемента. Ця активність розказує користувачеві, як правильно дізнаватися інформацію при екстреній ситуації. Функціональність інтерактивних елементів даної активності дозволяє вмикати та вимикати телевізор, а також відкривати та закривати двері. Дані дії відбуваються тоді, коли користувач натискає пальцем на один із предметів.

Розглянемо більш детально те, як було реалізовано такі функціональності. Візуально функціонал відкриття та закриття дверей складається з двох картинок, розміщених одна на одній. Перша картинка являє собою зображення закритих дверей, а друга відкритих. При натиснені на одну з картинок, вона стає невидимою, а натомість інша стає видимою. Також, паралельно відбувається озвучення відкриття та закриття дверей. Код, який призначений для відкриття дверей використовує setVisibility для встановлення стану видимості картинки відкритих дверей та стану невидимості для картинки закритих дверй. Токож, паралельно запускається раніше оголошений голосовий

запис відкриття дверей. З повним кодом відкриття дверей можна ознайомитися на лістингу 3.7.

Лістинг 3.7 – Код, який призначений для відкриття дверей

```
184         closeDoor.setOnClickListener(new View.OnClickListener() {
185             @Override
186             public void onClick(View v) {
187                 openDoor.setVisibility(View.VISIBLE);
188                 closeDoor.setVisibility(View.INVISIBLE);
189                 mediaPlayerOpenDoor.start();
190             }
191         });
```

Кінець лістингу 3.7

Для закриття дверей використано подібний код, однак він навпаки робить картинку відкритих дверей невидимою, а картинку закритих дверей видимою. Також, у коді відбувається запуск звуку закриття дверей. Поєднання цих двох кодів створює інтерактивний, рухливий елемент дверей. Перевагою такого елементу є те, що користувач за власним бажанням може змінювати сцену кімнати.

Другий інтерактивний елемент даної активності – телевізор. Він також складається з двох картинок та має схожий функціонал з дверима. Код для виключення телевізору аналогічний до кодів відкриття та закриття дверей, однак, код увімкнення телевізору відрізняється тим, що для голосового запису встановлено додаткову конструкцію setLooping. Код, який відповідає за функціонал увімкнення телевізору представлено у лістингу 3.8.

Лістинг 3.8 – Код, який призначений для увімкнення телевізору

```
202         TVImageOff.setOnClickListener(new View.OnClickListener() {
203             @Override
204             public void onClick(View v) {
205                 TVImageOn.setVisibility(View.VISIBLE);
206                 TVImageOff.setVisibility(View.INVISIBLE);
207                 mediaPlayerNewsTV.setLooping(true);
208                 mediaPlayerNewsTV.start();
209             }
210         });
211     }
```

Кінець лістингу 3.8

Конструкція setLooping використовується для створення або зупинку зациклення голосового запису. Значення «true» відповідає за створення зациклення. Тобто запис, у якому говориться про причини екстреної ситуації, після ввімкнення телевізору, буде прокручуватися до тих пір, доки телевізор не вимкнуть. При вимкнені телевізору спрацьовує функція медіа-плеєру stop().

Складніший функціонал представлено в активностях виготовлення засобів захисту шляхів дихання та збору рюкзака. В активності захисту шляхів дихання, при проходженні, важлива послідовність дій, тому, через код було реалізовано перевірки, які не дають виконати певні дії до початку їх черги.

Завданням цієї активності є: імітація створення засобу захисту, який допоможе при вибуху на хімічній станції. На лістингу 3.9 представлено код, який забезпечує функціонал інтерактивних елементів у даному вікні. Основа інтерактиву представляється через перетягування елементів. Активність представляє собою панель предметів, до яких належить йод, вода і вата, та медицинську маску. Користувачеві потрібно згідно підказок реалізувати перетягування елементів в правильному порядку. Перевірка правильності черги покрокових елементів, перевіряється у тандемі із стрілкою, яка відповідає за переключення між підказками та активностями.

Лістинг 3.9 – Код, який дозволяє перетягувати елементи

```
78      private View.OnLongClickListener longClickListener = new View.OnLongClickListener() {
79          @Override
80          public boolean onLongClick(View view) {
81              // Перевірка, чи видима стрілка
82              if (nextButton.getVisibility() != View.VISIBLE) {
83                  // Отримання даних, які перетягуються
84                  View.DragShadowBuilder shadowBuilder = new View.DragShadowBuilder(view);
85                  // Початок перетягування з даними
86                  view.startDragAndDrop( data: null, shadowBuilder, view,  flags: 0);
87                  return true;
88              } else {
89                  return false; // Користувач не може взаємодіяти з предметами, якщо стрілка видима
90              }
91          }
92      };
```

Кінець лістингу 3.9

У даному коді, дії опрацьовуються після довгого натиску на предмет. Для реалізації такого функціоналу було використано метод OnLongClickListener. Цей метод запускає сценарій коду тільки тоді, коли на предмет було натиснуто довгим натиском.

Для забезпечення перетягування було використано метод DragAndDrop. Даний метод дозволяє перетягувати і відпускати певний елемент. В даній активності реалізовано, що коли елемент при перетягувані відпускається і його позиція співпадає з іншим конкретним елементом, то запускається інший код реакції.

Спочатку у активності створення засобу дихальних шляхів відкривається підказка, у якій говориться користувачеві про те, що потрібно зробити. В момент виводу цієї першої підказки переміщення елементів при довгому натисненні не відбувається. Після прочитання підказки користувач повинен натиснути стрілку продовження. Після цього стрілка продовження зникає, а натомість стає доступною можливість перетягування елементів. Користувачеві виводиться нова підказка про те, що йому першим потрібно зробити, а саме створити йодовий розчин, змішавши йод з водою. Після змішання предмет йоду зникає, а предмет банки з водою перетворюється на предмет банки з йодованим розчином. Слід замітити, що на даному етапі, при переміщені будь-якого іншого предмету ніякої реакції не відбувається. Тобто, користувачеві доступні тільки ті дії, які описані у підказці.

Наступна підказка говорить про те, що користувачеві потрібно імітувати вимочення вати у йодованому розчині, тобто перетягнути предмет вати на предмет банки з йодованим розчином. Після перетягування предмет чистої вати перетворюється на предмет вати, вимоченої у йодованому розчині.

Далі відображається підказка про те, що слід зробити теж саме з маскою, що і з ватою. При перетягувані предмету маски на предмет банки з йодованим розчином, банка із розчином зникає, а медицинська маска стає вимоченою в йодному розчині. Остання підказка говорить про те, що потрібно помістити вимочену вату в маску і як одягнути мастку при потребі.

Такий інтерактивний функціонал дозволяє зануритися у процес створення засобу захисту, що робить гру більш реальною та цікавішою. Повний код для функціональності цієї активності можна переглянути у додатку А.

Подібний функціонал представлено також у активності збору тривожного ранцю. В даній активності потрібно навчитися, які речі потрібно збирати при виникненні такої екстреної ситуації. Тут реалізовано довільне перетягування елементів, без черги. Користувач повинен перетягнути один з предметів на рюкзак. Цей предмет тоді зникає та виводиться підказка, щодо нього. Після переміщення останнього елементу, стає видимою стрілка, яка переводить у наступну активність.

Усі ці функціонали мають інтерактивні елементи, що прикрашає додаток та робить його сучаснішим. Також, такі елементи в інтерфейсі дають змогу конкурувати з існуючими подібними додатками. Інтерактивні елементи «Emergency App» роблять його не лише корисним інструментом для навчання, але й захопливою грою, яка може зацікавити користувачів різного віку.

### 3.3 Впровадження функції розпізнавання голосових команд

Головною особливістю додатку «Emergency App», є впровадження у нього функції розпізнавання голосових команд. Реалізована ця функція у вигляді голосового помічника, який персонаж може викликати у власному телефоні. Цей помічник називається «Expert Avon». Скористатися ним можна в активності, яка відповідає за те, щоб навчити користувача вибирати правильні джерела інформації, при скоєні екстреної ситуації.

Як вже говорилося вище, дана активність містить іконку телефону, при натисненні на яку вилазить вікно з телефоном та відкритим голосовим помічником. Цей голосовий помічник містить кнопку виклику довідкового меню голосових команд та кнопку мікрофону, яка відповідає за увімкнення помічника. При натисненні на цю кнопку вилазить вікно розпізнавання мовлення Google. Ознайомитися із візуальним виглядом та роботою «Expert Avon» можна на рисунку 3.3.

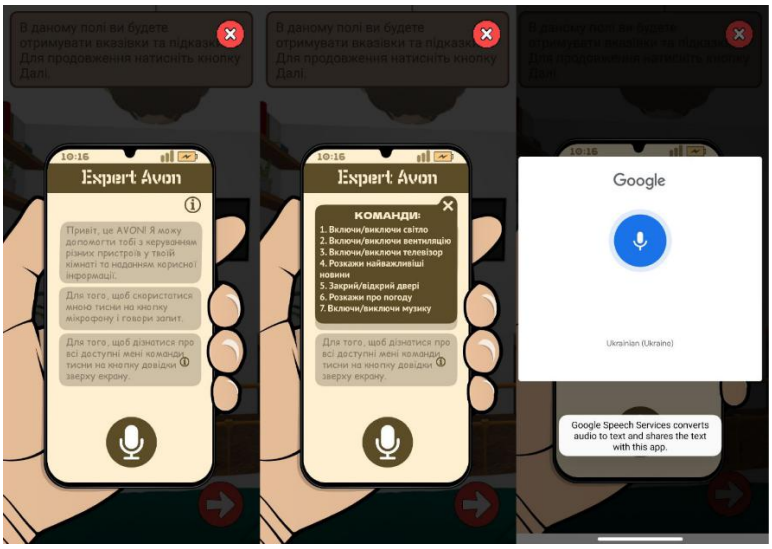


Рисунок 3.3 – Зовнішній вигляд та робота голосового помічника «Expert Avon»

Розглянемо функціональності та можливості всіх елементів керування у даному голосовому помічнику. У вікні голосового помічника виведені блоки ознайомлення. У них написано про помічник та як ним скористатися. В правому верхньому куті є іконка довідкового меню, яка, при натисненні відкриває меню, у якому описано всі голосові команди, на які запрограмовано реакцію. Код, який відповідає за відкриття та закриття довідкового меню команд міститься у лістингу 3.10.

Лістинг 3.10 – Код, який відповідає за відкриття та закриття довідкового меню

```
infoButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        info.setVisibility(View.VISIBLE);
        infoButton.setVisibility(View.INVISIBLE);
        infoCloseButton.setVisibility(View.VISIBLE);
    }
});
infoCloseButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        info.setVisibility(View.INVISIBLE);
        infoButton.setVisibility(View.VISIBLE);
        infoCloseButton.setVisibility(View.INVISIBLE);
    }
});
```

Кінець лістингу 3.10

При натисненні на довідкове меню, сама кнопка стає невидимою, а натомість з’являється хрестик для закриття вікна. Цей вікно зроблене зображенням, тому, для його виклики або схову достатньо вказати видимий стан або VISIBLE, або INVISIBLE.

Ще один елемент, який відповідає за закриття телефону, розміщений в правому верхньому куті екрану. Його функціонал є простим. Він просто робить всі елементи телефону та помічника невидимими.

Найбільшим функціоналом, у голосовому помічнику, володіє кнопка мікрофону. Для розпізнавання мовлення у коді використано Google Speach API. Google Speach API – це сервіс від Google, який дозволяє перетворювати аудіо у текст. Саме через нього додаток розпізнає мовлення та реагує на певні команди.

Фактична реалізація перетворення аудіо в текст міститься на лістингу 3.11. У цьому коді показано створення спеціального методу, який відповідає за розпізнавання мовлення. Цей метод містить намір почати розпізнавання мовлення, мову, яку потрібно розпізнавати, фактичне розпізнання та деякі інші додаткові елементи.

Лістинг 3.11 – Код, який перетворює аудіо в текст

```
213     ivate void startVoiceRecognitionActivity() {
214       Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
215       intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE,  value: "uk-UA");
216       intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
217       intent.putExtra(RecognizerIntent.EXTRA_PROMPT, (CharSequence) null);
218       startActivityForResult(intent, VOICE_RECOGNITION_REQUEST_CODE);
```

Кінець лістингу 3.11

Після зчитування голосової команди, у додатку запрограмовано декілька реакцій, а саме: увімкнення/вимкнення світла, увімкнення/вимкнення вентиляції, увімкнення/вимкнення телевізору, озвучення останніх новин, відкриття/закриття дверей, озвучення інформації про погоду, увімкнення/вимкнення музики. Всі ці реакції зроблені за схожим принципом. Для одних потрібно увімкнути якийсь голосовий запис, іншим необхідно зробити невидимим чи видимим якийсь елемент або елементи.

Розглянемо на прикладі однієї з команд те, як було реалізовано у коді реакцію на команду. На лістингу 3.12 відображено код, який відповідає за увімкнення та вимкнення телевізору, після зчитання певних голосових команд. Описується новий метод onActivityResult. У ньому результат перетворення аудіо в мовлення заноситься у масив. Після цього відбувається перевірка, чи відповідає отриманий текст, команді «включи телевізор». Якщо так, то відбувається певні дії, у цьому випадку увімкається голосовий запис репортажу. Також, виводиться спливаєче повідомлення про те, що телевізо увімкнено. А якщо зчитаний текст не відповідає першій команді, то перевіряється наступна.

Лістинг 3.12 – Приклад коду, який відповідає за реакцію на голосову команду

```
221     @Override
222     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
223         super.onActivityResult(requestCode, resultCode, data);
224
225         if (requestCode == VOICE_RECOGNITION_REQUEST_CODE && resultCode == RESULT_OK) {
226             ArrayList<String> matches = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
227
228             if (matches.contains("включи телевізор")) {
229                 TVImageOn.setVisibility(View.VISIBLE);
230                 Toast.makeText( context: this,  text: "Телевізор увімкнено", Toast.LENGTH_SHORT).show();
231                 mediaPlayerNewsTV.setLooping(true);
232                 mediaPlayerNewsTV.start();
233             } else if (matches.contains("виключи телевізор")) {
234                 TVImageOn.setVisibility(View.INVISIBLE);
235                 TVImageOff.setVisibility(View.VISIBLE);
236                 Toast.makeText( context: this,  text: "Телевізор вимкнено", Toast.LENGTH_SHORT).show();
237                 mediaPlayerNewsTV.pause();
238             } else {
239                 Toast.makeText( context: this,  text: "Невідома команда", Toast.LENGTH_SHORT).show();
240             }
241         }
242     }
```

Кінець лістингу 3.12

Якщо цей метод отримує невідому команду, то користувачеві виводится спливаюче повідомлення «Невідома команда». Інші команди зроблені аналогічно цій, і не потребують додаткового розгляду. Повний код, для даної активності міститься у додатках до кваліфікаційної роботи.

Отже, дана активність пропонує широкий функціонал розпізнавання голосових команд. Таких команд, у коді, може бути безліч, і реакції на них

можуть бути різними. Така активність робить програму унікальною та цікавою у роботі з нею.

### 3.4 Збірка і тестування програми

Після розробки додатку «Emergency App», було проведено збірку та тестування програми. Для тестування було обрано смартфон Redmi Note 8 Pro. Збірка була виконана в розширення APK, що являє собою стандартне розширення для встановлюваних файлів Android.

Перша активність рівня 1 містить діалогове вікно з поясненням рівню та активність початку сценарію. В даній активності присутня текстова анімації. Кожна кнопка працює коректно та виконує задні завдання. Ознайомитися з візуальним виглядом даної активності можна на рисунку 3.4.

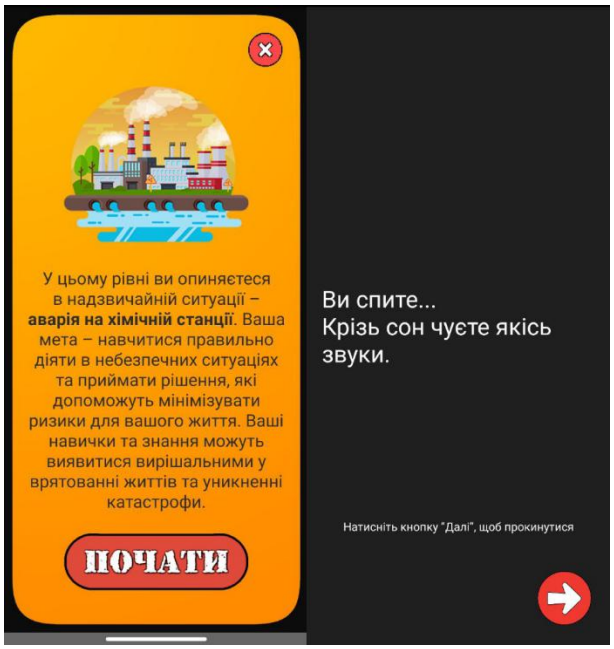


Рисунок 3.4 – Результат тестування активності ознайомлення з рівнем 1

Наступною була протестована активність вибору джерел інформації (рис. 3.5). У ній були певірені функції телевізору, дверей та голосового помічника. Всі функції працюють коректно. Для звіту результату тестування було виведено не всі знімки екрану. Тестування активності герметизаціїї

деверей (рис. 3.6) відбулося успішно. Було перевірено валідацію порядку вибору предметів, та успішно виявлено коректну роботу.

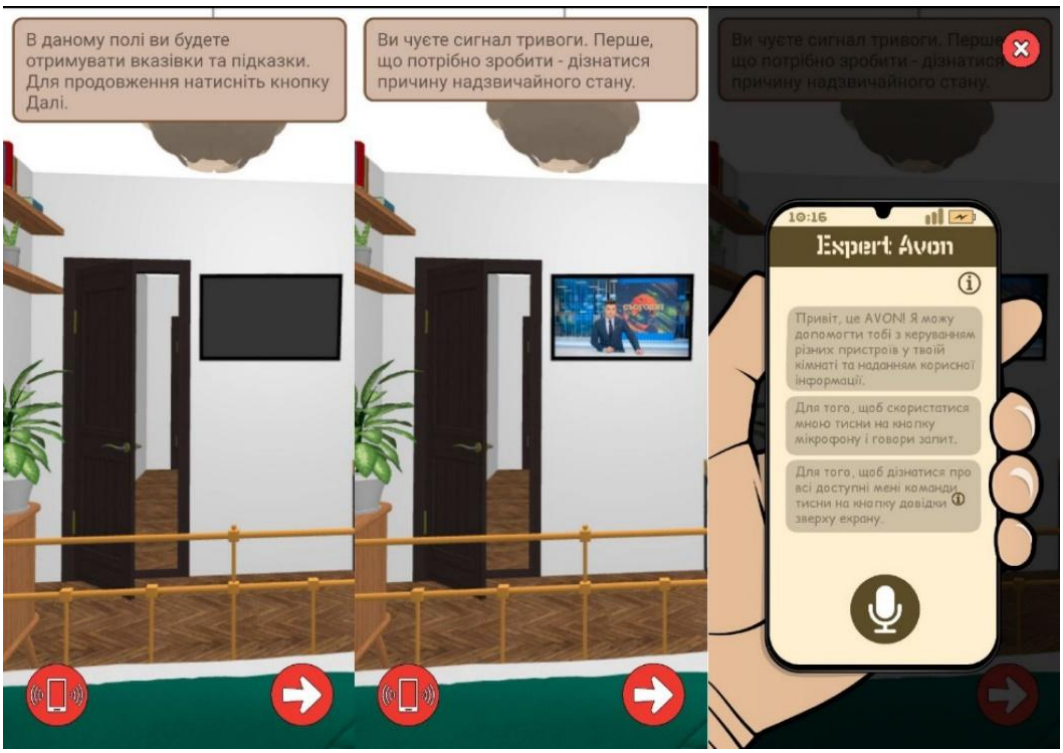


Рисунок 3.5 – Результат тестування активності вибору джерел інформації


Рисунок 3.6 – Результат тестування активності герметизації дверей

Наступною було протестовано активність герметизації вікон та вентиляції. При тестувані не було виявлено багів чи невідповідностей із запрограмованим функціоналом. Ознайомитися з результатами тестування можна на рисунку 3.7.

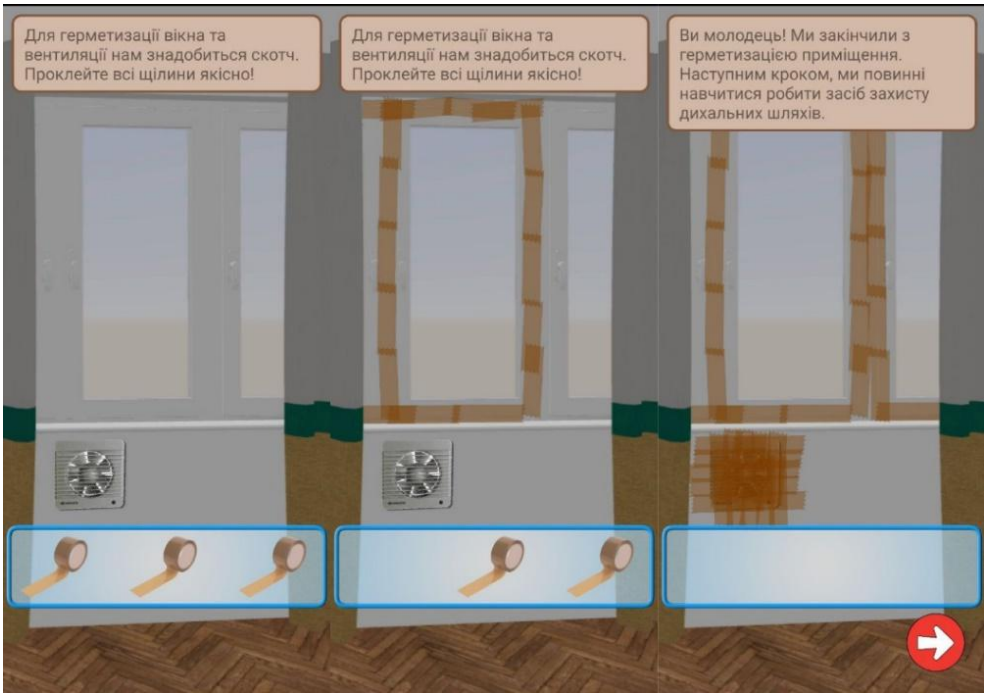


Рисунок 3.7 – Результат тестування активності герметизації вікон та вентиляції

В активності створення засобу захисту шляхів дихання (рис. 3.8), було протестовано функціонал перетягування елементів та реакцію на змішування предметів. Також, було перевірено чи відбувається змішування по черзі. Під час тестування ніяких багів не було виявлено.

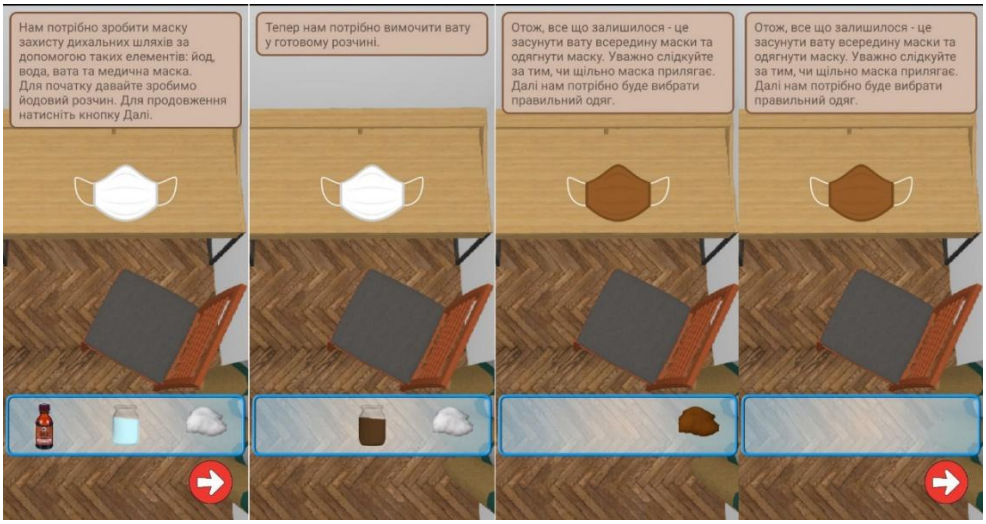


Рисунок 3.8 – Результат тестування активності захисту шляхів дихання

В наступній активності вибору одягу (рис. 3.9), було протестовано правильність відповідей та реакцій на будь-яку відповідь. В результаті тестування було затверджено коректну роботу активності та правильніть вибору елементів.


Рисунок 3.9 – Результат тестування активності вибору зовнішнього одягу

В активності збору ранцю було протестовано можливість перетягувати елементи та правильність виводу підказок після перетягування. З візуальними результатами тестування можна ознайомитися на рисунку 3.10. При тестувані затверджено коректну роботу даної активності.


Рисунок 3.10 – Результат тестування активності збору тривожного ранцю

В останній активності отримання сертифікату (рис. 3.11), було протестовано коректність заповнення ім’я та прізвища у сертифікат та можливість поділитися сертифікатом. В результаті тестування було виявлено коректну роботу активності. Також, було перевірено функцію завантаження сертифікату в галерею, через кнопку поділитися.

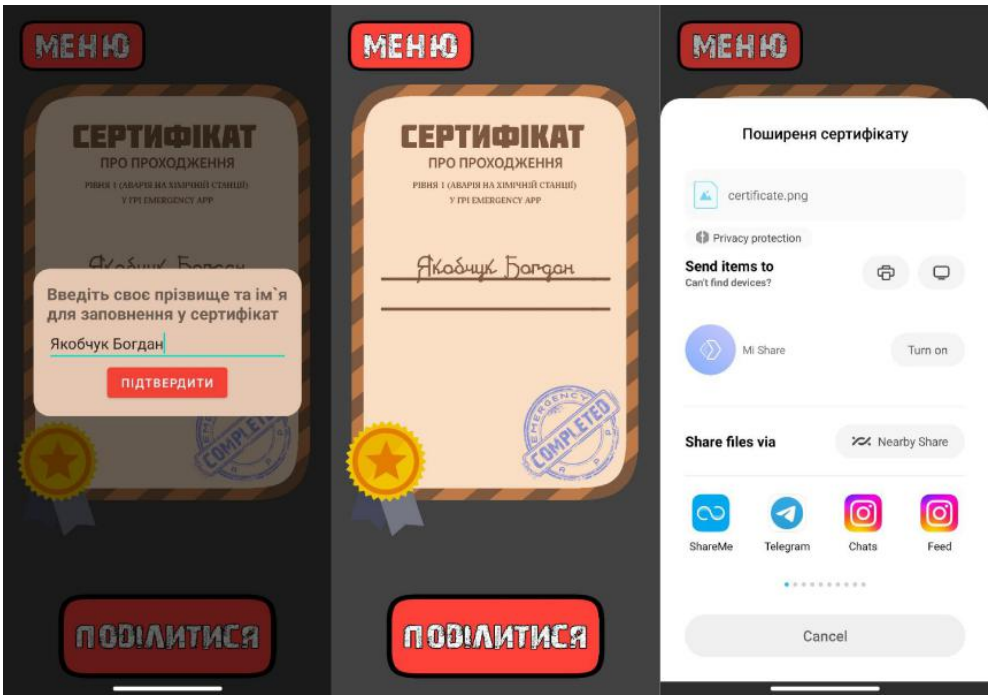


Рисунок 3.11 – Результат тестування активності отримання сертифікату

Отже, було про ведено тестування всіх описаних активностей та функціїй. При тестувані ніяких багів виявлено не було. Це пітверджує коректну роботу додатку. Кожна функція працює так, як запрограмована. Візуальних багів на пристрої тестування, також не було виявлено.

# ВИСНОВКИ

Інтерактивний додаток «Emergency App» є механізмом навчання теоретичних та практичних навиків з питань запобігання надзвичайних ситуацій.

В результаті виконання кваліфікаційної роботи було:

- проведено аналітичний огляд інтернет-ресурсів, наукових публікацій, літературних джерел з питань розробки інтерактивних додатків та визначено, що у якості платформи доцільно обрати платформу Android, а інструментарій та мову програмування, для реалізації проекту – Android Studio і Java;
- досліджено об’єкти IoT, що включені в роботу додатка, та визначено те, що у IDE Android Studio, інтеграція з IoT відбувається через платформу IoT-Ignite;
- визначено класи, які використані в середовищі мови пограмування Java для створення функціоналу додатку, та аргументовано їх вибір через простоту та розробницькі можливості, які вони пропонують;
- розроблено головне меню, меню рівнів додаку та реалізовано запуск активностей для візуального функціонування, програмно реалізовано перехід між ними;
- створено елементи у додатку та визначено особливі, інтерактивні елементи, з якими користувач взаємодіє;
- впроваджено функціонал розпізнавання голосових команд, налаштовано його взаємодію з програмним кодом та досліджено, що особливості такої взаємодії полягають у дозволах, правильно налаштованому перетворені аудіо в текст та реакціями на голосові команди;
- проведено збірку і тестування програми та перевірено, що всі активності додатку працюють коректно, мають чіткий і зрозумілий, інтерактивний інтерфейс.

Результати кваліфікаційної роботи підтверджують актуальність обраної теми. Розробка інтерактивного додатку «Emergency App» не тільки дозволила детально дослідити процес створення мобільних додатків, але й сприяла

формуванню ефективного навчального інструменту для підготовки громадян до надзвичайних ситуацій. Це підтверджує важливість подальших досліджень та вдосконалення даного додатку для ширшого застосування у суспільстві.

Таким чином, проведена робота досягла своєї мети – створено ефективний, інтерактивний інструмент для навчання діям у надзвичайних ситуаціях, що сприятиме підвищенню рівня безпеки і готовності громадян до можливих загроз.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Disaster master. Plan Ahead for Disasters | Ready.gov. URL: https://www.ready.gov/kids/games/data/dm-english/ (дата звернення: 09.02.2024).

2. Play the disaster game. Home | FEMA.gov. URL: https://www.fema.gov/about/organization/region-8/disaster-mind-game (дата звернення: 09.02.2024).

3. Apps Code valley. First Aid & Emergency Techniques – apps on google play. Android Apps on Google Play. URL: https://play.google.com/store/apps/details?id=appscodevalley.firstaid.emergency.techniques.medicines88&pli=1 (дата звернення: 09.02.2024).

4. B-Ready app | disaster central. Home | Disaster Central. URL: https://disastercentral.org/b-ready-app (дата звернення: 09.02.2024).

5. Stop Disasters!. URL: https://www.stopdisastersgame.org/stop_disasters/ (дата звернення: 09.02.2024).

6. Emergency ready app – apps on google play. Android Apps on Google Play. URL: https://play.google.com/store/apps/details?id=kr.go.nema.disasteralert_eng (дата звернення: 09.02.2024).

7. Creating IoT-Ignite Project in Android Studio – IoT-Ignite Devzone. IoT-Ignite Devzone. URL: https://devzone.iot-ignite.com/knowledge-base/creating-iot-ignite-project-android-studio/ (дата звернення: 11.02.2024).

8. Build an internet of things app | Android Developers. Android Developers. URL: https://developer.android.com/training/cars/apps/iot#java (дата звернення: 11.02.2024).

9. Android Project folder Structure – GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/android-project-folder-structure/ (дата звернення: 14.02.2024).

# ДОДАТКИ

# Додаток А

## Функціональний код для активності створення засобу захисту дихання

```
package space.pavuk.emergencyapp;

import android.content.Intent;
import android.os.Bundle;
import android.view.DragEvent;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class ProtectionBreathingActivity extends AppCompatActivity {
    private ImageView itemOne, itemTwo, itemThree, itemFour, itemFive, mask,
iodizedMask;
    private ImageView nextButton;
    private TextView textHints;
    private int hintCount = 0;

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_protection_breathing);

        Window w = getWindow();

w.setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.Layout
Params.FLAG_FULLSCREEN);
        View decorView = getWindow().getDecorView();
        int uiOptions = View.SYSTEM_UI_FLAG_FULLSCREEN |
View.SYSTEM_UI_FLAG_HIDE_NAVIGATION;
        decorView.setSystemUiVisibility(uiOptions);

        // Отримання посилань на всі елементи макету
        itemOne = findViewById(R.id.itemOne);
        itemTwo = findViewById(R.id.itemTwo);
        itemThree = findViewById(R.id.itemThree);
        itemFour = findViewById(R.id.itemFour);
        itemFive = findViewById(R.id.itemFive);
        mask = findViewById(R.id.mask);
        iodizedMask = findViewById(R.id.iodizedMask);
        nextButton = findViewById(R.id.nextButton);
        textHints = findViewById(R.id.textHints);

        // Налаштування подій для перетягування
        itemOne.setOnLongClickListener(longClickListener);
        itemTwo.setOnLongClickListener(longClickListener);
        itemThree.setOnLongClickListener(longClickListener);
        itemFour.setOnLongClickListener(longClickListener);
        itemFive.setOnLongClickListener(longClickListener);
```

```
        mask.setOnLongClickListener(longClickListener);
        iodizedMask.setOnLongClickListener(longClickListener);

        // Призначення місця для приземлення (drop) для всіх елементів
        itemTwo.setOnDragListener(dragListener);
        itemFour.setOnDragListener(dragListener);
        iodizedMask.setOnDragListener(dragListener);

        nextButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (hintCount == 0){
                    textHints.setText(R.string.instruction_13);
                    nextButton.setVisibility(View.INVISIBLE);
                    hintCount++;
                } else {
                    Toast.makeText(ProtectionBreathingActivity.this, 'Переходимо до захисту
тіла', Toast.LENGTH_SHORT).show();
                    Intent intent = new Intent(ProtectionBreathingActivity.this,
ClothesActivity.class);
                    startActivity(intent);
                }
            }
        });
    }

    private View.OnLongClickListener longClickListener = new
View.OnLongClickListener() {
        @Override
        public boolean onLongClick(View view) {
            // Перевірка, чи видима стрілка
```

```
            if (nextButton.getVisibility() != View.VISIBLE) {
                // Отримання даних, які перетягуються
                View.DragShadowBuilder shadowBuilder = new
View.DragShadowBuilder(view);
                // Початок перетягування з даними
                view.startDragAndDrop(null, shadowBuilder, view, 0);
                return true;
            } else {
                return false; // Користувач не може взаємодіяти з предметами, якщо стрілка
видима
            }
        }
    };

    private View.OnDragListener dragListener = new View.OnDragListener() {
        private boolean isSecondStepCompleted = false;



        @Override
        public boolean onDrag(View v, DragEvent event) {
            int action = event.getAction();
            switch (action) {
                case DragEvent.ACTION_DROP:
                    View view = (View) event.getLocalState();
                    // Перевірка, чи це правильний приземлення (drop)
                    if (view.getId() == R.id.itemOne && v.getId() == R.id.itemTwo) {
                        // Логіка для першого кроку
                        itemOne.setVisibility(View.INVISIBLE);
                        itemTwo.setVisibility(View.INVISIBLE);
                        itemFour.setVisibility(View.VISIBLE);
```

```
                itemThree.setVisibility(View.VISIBLE);
                nextButton.setVisibility(View.INVISIBLE);
                textHints.setText(R.string.instruction_14);
            } else if (view.getId() == R.id.itemThree && v.getId() == R.id.itemFour) {
                // Логіка для другого кроку, якщо другий крок вже виконаний
                itemThree.setVisibility(View.INVISIBLE);
                itemFive.setVisibility(View.VISIBLE);
                mask.setVisibility(View.VISIBLE);
                textHints.setText(R.string.instruction_15);
                isSecondStepCompleted = true;
            } else if (view.getId() == R.id.mask && v.getId() == R.id.itemFour &&
isSecondStepCompleted) {
                // Логіка для третього кроку, якщо другий крок вже виконаний
                mask.setVisibility(View.INVISIBLE);
                itemFour.setVisibility(View.INVISIBLE);
                iodizedMask.setVisibility(View.VISIBLE);
                itemFive.setVisibility(View.VISIBLE);
                textHints.setText(R.string.instruction_16);
            } else if (view.getId() == R.id.itemFive && v.getId() == R.id.iodizedMask) {
                // Логіка для останнього кроку
                itemFive.setVisibility(View.INVISIBLE);
                iodizedMask.setVisibility(View.VISIBLE);
                nextButton.setVisibility(View.VISIBLE);
            }
            break;
        }
        return true;
    }
  };
}
```

# Додаток Б
## Функціональний код для роботи голосового помічника

```
private void startVoiceRecognitionActivity() {
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, 'uk-UA');
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT, (CharSequence) null);
    startActivityForResult(intent, VOICE_RECOGNITION_REQUEST_CODE);
  }

  @Override
  protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == VOICE_RECOGNITION_REQUEST_CODE && resultCode ==
RESULT_OK) {
      ArrayList<String> matches =
data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);

      if (matches.contains('включи телевізор')) {
        TVImageOn.setVisibility(View.VISIBLE);
        Toast.makeText(this, 'Телевізор увімкнено', Toast.LENGTH_SHORT).show();
        mediaPlayerNewsTV.setLooping(true);
        mediaPlayerNewsTV.start();
      } else if (matches.contains('виключи телевізор')) {
        TVImageOn.setVisibility(View.INVISIBLE);
        TVImageOff.setVisibility(View.VISIBLE);
        Toast.makeText(this, 'Телевізор вимкнено', Toast.LENGTH_SHORT).show();
        mediaPlayerNewsTV.pause();
      } else if (matches.contains('розкажи найважливіші новини')) {
        Toast.makeText(this, 'Розказую про новини...',
Toast.LENGTH_SHORT).show();
        mediaPlayerNewsAvon.start();
      } else if (matches.contains('Закрий двері') && closeDoor.getVisibility() ==
View.INVISIBLE) {
        closeDoor.setVisibility(View.VISIBLE);
        openDoor.setVisibility(View.INVISIBLE);
        mediaPlayerCloseDoor.start();
        Toast.makeText(this, 'Двері замкнено', Toast.LENGTH_SHORT).show();
      } else if (matches.contains('Відкрий двері') && closeDoor.getVisibility() ==
View.VISIBLE) {
```

```
            closeDoor.setVisibility(View.INVISIBLE);
            openDoor.setVisibility(View.VISIBLE);
            mediaPlayerOpenDoor.start();
            Toast.makeText(this, 'Двері відчинено', Toast.LENGTH_SHORT).show();
        } else if (matches.contains('Включи музику')) {
            mediaPlayerMusic.start();
            Toast.makeText(this, 'Музику ввімкнено. Приємного прослуховування!',
Toast.LENGTH_SHORT).show();
        } else if (matches.contains('Виключи музику')) {
            mediaPlayerMusic.pause();
            Toast.makeText(this, 'Музику вимкнено', Toast.LENGTH_SHORT).show();
        } else if (matches.contains('розкажи про погоду')) {
            mediaPlayerWether.start();
            Toast.makeText(this, 'Розказую про погоду...', Toast.LENGTH_SHORT).show();
        } else if (matches.contains('Включи світло')) {
            lightImage.setVisibility(View.VISIBLE);
            Toast.makeText(this, 'Включаю світло', Toast.LENGTH_SHORT).show();
        } else if (matches.contains('виключи світло')) {
            lightImage.setVisibility(View.INVISIBLE);
            Toast.makeText(this, 'Виключив світло', Toast.LENGTH_SHORT).show();
        } else if (matches.contains('включи вентиляцію')) {
            mediaPlayerVentilation.start();
            Toast.makeText(this, 'Включаю вентиляцію', Toast.LENGTH_SHORT).show();
        } else if (matches.contains('виключи вентиляцію')) {
            mediaPlayerVentilation.pause();
            Toast.makeText(this, 'Виключив вентиляцію', Toast.LENGTH_SHORT).show();
        } else if (matches.contains('Закрий двері') && closeDoor.getVisibility() !=
View.INVISIBLE) {
            closeDoor.setVisibility(View.VISIBLE);
            openDoor.setVisibility(View.INVISIBLE);
            Toast.makeText(this, 'Двері вже закриті', Toast.LENGTH_SHORT).show();
        } else if (matches.contains('Відкрий двері') && closeDoor.getVisibility() !=
View.VISIBLE) {
            closeDoor.setVisibility(View.INVISIBLE);
            openDoor.setVisibility(View.VISIBLE);
            Toast.makeText(this, 'Двері вже відчинені', Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(this, 'Невідома команда', Toast.LENGTH_SHORT).show();
        }
    }
}
```