МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота магістра

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПРОЄКТУВАННЯ СИСТЕМИ ПІДГОТОВКИ ДО НАДЗВИЧАЙНИХ СИТУАЦІЙ

Здобувач вищої освіти гр. ІН.м-13 Ярослав ЛІЧНИЙ

Науковий керівник,
кандидат фізико-математичних наук,
асистент кафедри комп'ютерних наук Ольга ШУТИЛЄВА

В.о. завідувача кафедри
кандидат технічних наук, доцент,
доцент кафедри комп'ютерних наук Ігор ШЕЛЕХОВ

Суми 2022

Сумський державний університет
(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*
Спеціальність *«122 – Комп'ютерні науки»*

Затверджую:
В.о. зав.кафедри _____
_____
"_____" _____ 2022р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Лічному Ярославу Ярославовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Інформаційна технологія проєктування системи підготовки до надзвичайних ситуацій*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі здобувачем вищої закінченого роботи_____

3. Вхідні дані до роботи _____
_____
_____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Огляд технологій, що застосовуються для підготовки до надзвичайних ситуацій; 2) Постановка завдання й формування завдань дослідження; 3) Огляд технологій, що використовуються під час розробки мобільних додатків 4) Огляд технологій для графічного інтерфейсу; 5) Практична реалізація додатку; 6) Аналіз та тестування результатів.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
_____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання

_____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	Огляд технологій, що застосовуються для підготовки до надзвичайних ситуацій		
2.	Постановка задачі та формування завдань дослідження.		
3.	Опис архітектури додатку		
4.	Підбір шаблону для дизайну користувацького інтерфейсу		
5.	Оформлення пояснювальної записки до кваліфікаційної магістерської роботи		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 67 стор., 23 рис., 1 табл., 1 додаток, 13 джерел.

Об’єкт дослідження – інформаційні процеси аналізу прийняття імпульсивних дій.

Мета роботи – розробка мобільного додатку для підготовки до надзвичайних ситуацій. Завдяки додатку, користувач матиме змогу мінімізувати потенційні негативні наслідки від надзвичайних ситуацій.

Методи дослідження – метод статистичних випробувань.

Результати – розроблено додаток для підготовки до надзвичайних ситуацій. Додаток створений для використання на платформі Андроїд. Під час створення використовувались актуальні підходи, технології, архітектура. Додаток написаний мовою програмування Java, дані зберігаються у базі даних Room.

МОБІЛЬНИЙ ДОДАТОК, ОПЕРАЦІЙНА СИСТЕМА, АНДРОЇД, ПРОГРЕС, БАЗА ДАНИХ, НАДЗВИЧАЙНІ СИТУАЦІЇ, ЮЗАБІЛІТІ, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС

ЗМІСТ

1. ЛІТЕРАТУРНИЙ ОГЛЯД ЗА ОБРАНОЮ ТЕМАТИКОЮ 6
1.1 Вибір платформи для використання 7
1.2 Огляд існуючих рішень 7
1.3 Проблеми та пошук рішення проблем внаслідок НС 11
1.4 Постановка задачі 12
2. МЕТОДИКА ВИРІШЕННЯ ПОСТАВЛЕНИХ ЗАДАЧ 13
2.1 Актівіті 13
2.2 Фрагменти 14
2.3 Патерн дизайну 16
2.4 Колір і контраст 19
3. ПРОГРАМНА РЕАЛІЗАЦІЯ 22
3.1. Середовище для створення 22
3.2 Створення бізнес-логіки 29
3.3 Зберігання даних 31
3.4 Взаємодія з візуальною частиною додатку 33
3.5 Тестування 35
ВИСНОВКИ 37
ДОДАТКИ 40
Додаток А. Код з візуальною частиною додатку 40
Додаток Б. Код з конфігурацією проекту 57
Додаток В. Код з бізнес-логікою додатку 60

ВСТУП

Актуальність роботи. Робота присвячена темі розробки мобільного додатку. У якості мобільної операційної системи обрано Андроїд. Дана операційна система є достатньо популярною та продовжує активно розвиватися, зокрема наразі дана ОС є найпопулярнішою для мобільних пристроїв як в Україні, так я в світі. Під час написання кваліфікаційної роботи, будуть описані проблеми, з котрими може зіткнутися людина та запропоновані поради, як саме підготуватись до тої чи іншої надзвичайної ситуації, яким чином мінімізувати негативні наслідки, котрі вона може принести. Буде чітко сформована проблема, яку вирішуватиме додаток, визначено, яким функціоналом додаток має володіти.

Оскільки, аудиторією додатку є люди будь якого віку, то він має бути зрозумілим для користувачів. Для інтуїтивного користування додатком буде обрано поширений шаблон UX та UI дизайну. Будуть описані його підходи та поняття, якими він описує, такі як колір, фігура, глибина. Це допоможе зробити дизайн зручним у користуванні із сучасними підходами. Окрім зручного користувацького інтерфейсу, додаток має працювати без мобільного зв’язку та інтернету, оскільки у надзвичайних ситуації такі ресурси можуть бути недоступними або доступ до них може бути надзвичайно обмеженим певний чай. Буде обрано надійну архітектуру розробки та яка підходить для майбутнього масштабування додатку та є рекомендованою від компанії, котра і займається розвитком операційної системи Андроїд. Створюватись додаток буде з розрахунком, щоб мати змогу працювати не тільки на останніх, а і на вже не досить актуальних версіях операційної системи Андроїд. Будуть описані технології, котрі задіяні для відображення інформації на екрані, а також технологіх бізнес-логіки. Будемо використовувати Room Database у якості бази даних.

1. ЛІТЕРАТУРНИЙ ОГЛЯД ЗА ОБРАНОЮ ТЕМАТИКОЮ

Надзвичайна ситуація – стан окремої території чи об’єкту господарювання, що характеризується порушенням нормальних умов життя і діяльності людей на території чи об’єкті, спричинене аварією, катастрофою, стихійним лихом, епідемією, епізоотією, епіфітотією, пожежею, застосуванням засобів ураження, що призвели або можуть призвести до загибелі людей, значних матеріальних втрат, істотного погіршення стану навколишнього природного середовища [1].

Надзвичайні ситуації бувають різних видів та можуть нести різні ефекти. Проаналізуємо, якого саме характеру бувають надзвичайні ситуації:

- техногенного характеру;
- природного характеру;
- воєнного характеру;
- соціально-політичного характеру.

Усі ці характери надзвичайних ситуацій є важливими. Можна відзначити, що у перших трьох видах надзвичайних ситуацій та у четвертому, дещо різни умови, у котрих може опинитися людина. Якщо для перших трьох правила поведінки будуть схожі та не будуть залежить від країни чи регіону, то для четвертого виду надзвичайних ситуацій не можна дати загальні поради, будь то участь чи ігнорування протестів, активне висловлювання політичних поглядів та так далі.

Тож, які спільні риси можуть мати проблеми першого, другого чи третього характерів:

- відсутність електроенергії;
- відсутність технічної та / або питної води;
- ховатися від об’єктів чи речовин, що можуть нашкодити здоров’ю;
- потреба до переміщення до іншого міста чи країни у надзвичайно стислі терміни.

1.1 Вибір платформи для використання

Оскільки надзвичайні ситуації можуть застати у будь який момент, то смартфон є вдалим вибором, як він є більш мобільним, ніж комп'ютер. Веб версії додатків не є занадто приємними для користування, тому мобільний додаток – це найкращій вибір у нашому випадку. Останні досліди показують, що 55 відсотків українців мають смартфони. Для молоді цей показник дорівнює 92 відсотків [2].

Також, треба відзначити, що відсоток людей старшого віку, які користуються смартфонами значно збільшується кожен рік. Серед категорії віком 45-60 років зараз спостерігається найбільш активне зростання цих пристроїв [3].

Не тільки більше людей починають користуватися смартфонами, та також є тенденція на те, що люди, які вже користуються смартфонами, проводять в них все більше та більше часу. Близько 40% свого життя люди зараз проводять у смартфонах [4].

Наразі, двома найбільш розповсюдженими мобільними операційними системами є Android та IOS. У сумі, вони займають абсолютну більшість у ринку смартфонів. Оскільки операційна система андроїд є найбільш розповсюдженою на території Україні, тому саме під неї буде створюватись додаток.

1.2 Огляд існуючих рішень

Огляд існуючих рішень будемо виконувати з використанням Google Play та Apple Store. Плей маркет – це магазин мобільних додатків, розроблений спеціально для операційної системи Андроїд. Він дозволяє знаходити і встановлювати гри, електронні книги, словники, музичні та відеоплеєри, завантажувати карти, фільми, музику, різноманітні програми на кшталт календарів, калькуляторів, новин, погоди тощо. Загалом, всі потрібні і не дуже

потрібні функції, якими так люблять користуватися сучасні користувачі. Play Market передустановлено на будь-якому гаджеті з ОС Android. Якщо ж, сталося так, що випадково або навмисно його було видалено, Play Market офіційно можна завантажити [5].

Тож, проаналізуємо додатки, які можуть бути конкурентами додатку, який буде розроблено у ході виконання магістерської роботи. Слід зазначити, що всі ці додатки створені для надзвичайних ситуацій, але мета їх використання є досить різною.

Disaster Alert – це безкоштовний мобільний додаток для громадського користування, який надає глобальній спільноті сповіщення про критичні небезпеки та інформацію, необхідну для забезпечення безпеки у будь-якій точці світу [6].

Цей додаток показує, де є надзвичайні ситуації, спричинені природою. Однак, у ньому немає порад про те, що робити та як підготуватись, якщо сталась надзвичайна ситуація.

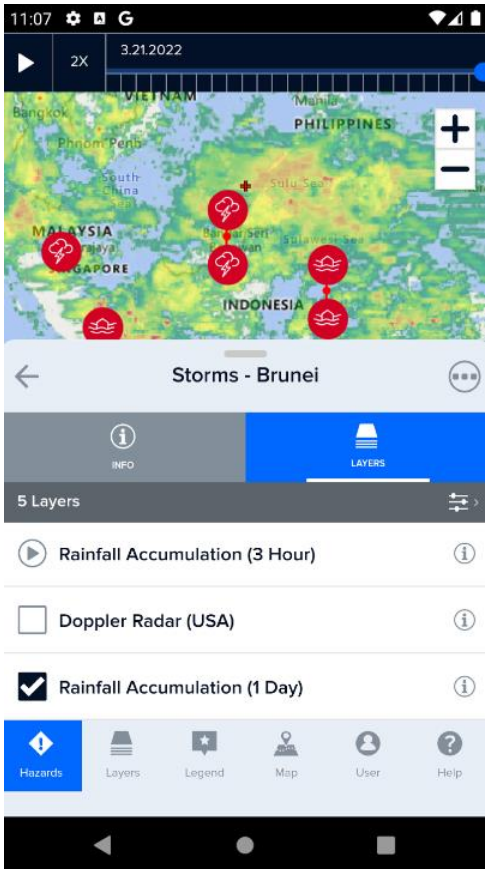

Рисунок 1.1 – Демонстрація повідомлення про шторм

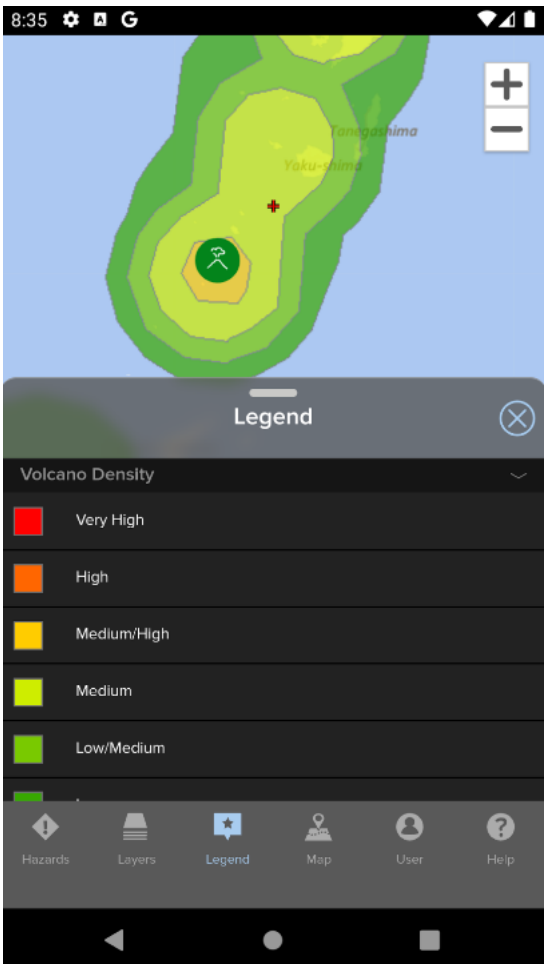

Рисунок 1.2 –Розділ Legend

Наступним для огляду є додаток «Особиста безпека». Цей додаток є більш зосередженим на тому, що може зробити мобільний пристрій під час надзвичайної ситуацій. Також, у даному випадку, надзвичайними ситуаціями є такі ситуації, як ДТП, небезпечна подорож по місту. У додатку є камера для фіксування надзвичайних ситуацій та кнопка виклику служб з надзвичайних ситуацій. Тож, цей додаток також не є прямим конкурентом нашого додатку.

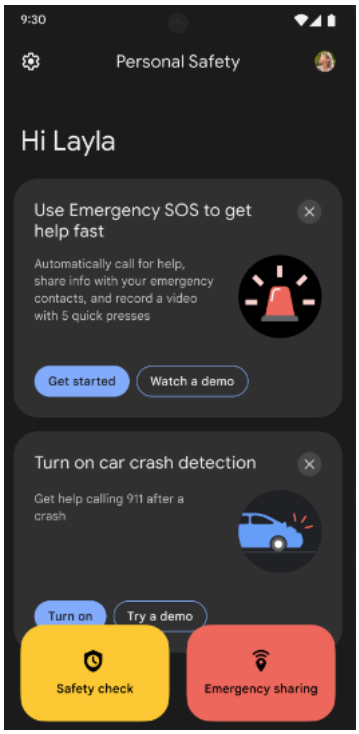

Рисунок 1.3 – Головна сторінка додатку

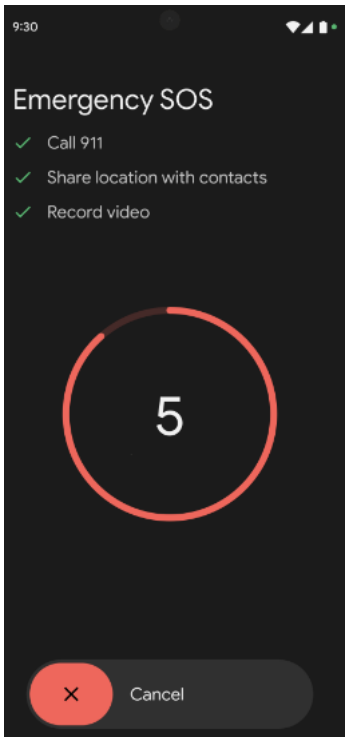

Рисунок 1.4 – Виклик служб з надзвичайних ситуацій

Рисунок 1.5 – Фіксування ДТП

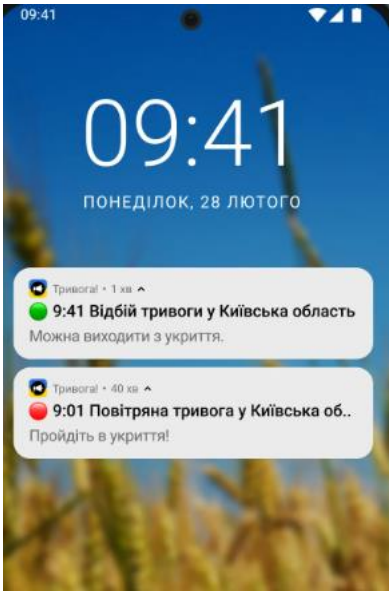

Рисунок 1.6 – Повідомлення від додатку

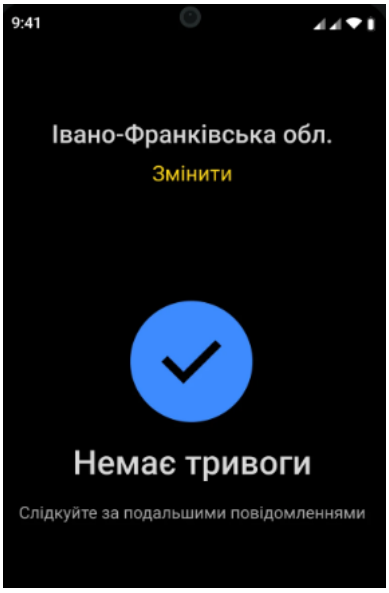

Рисунок 1.7 – Вибір локації

Як можемо бачити, цей додаток інформує про тревоги, але не дає інфрмацію, як в них поводитись.

1.3 Проблеми та пошук рішення проблем внаслідок надзвичайних ситуацій

Оскільки надзвичайні ситуації можуть бути навіть смертельними для людини, то треба мати хоча б базову підготовку до їх наслідків. Розберемо детально проблеми, з якими може зіштовхнутися людина через наслідки надзвичайних ситуацій.

1. **Відсутність електроенергії**. Оскільки більшість приладів, які використовуються у повсякденні є електричними, неможливість користуватись деякими з них може викликати сильне занепокоєння та обмежити можливість покривати базові потреби. Наразі у суспільстві маємо більші та більші тенденції на електрифікацію. Наприклад, газові плити все частіше замінюють електричними. Мікрохвильові пічки також є електричними приборами, так само, як і холодильник. Через це, покрити базову потребу у електроживленні, стає значно важче, ніж раніше. Також, опалення в приватних будинках, квартирах, комунальних установах, підприємствах може бути неможливим без електроенергії. У сильний мороз це може зробити неможливим функціонування підприємства.

2. **Відсутність води**. Питна вода має бути чистою, у іншому випадку людина може отримати серйозні проблеми зі здоров’ям, починаючи під погіршення стану шкіри чи отруєння, закінчуючи смертю, якщо внаслідок надзвичайного випадку немає доступу до води чи вона була значно наповнена отруйними речовинами. Технічна вода також важлива, щоб підтримувати базову гігієну [7-8].

3. **Укриття від об’єктів чи речовин, що можуть нашкодити здоров’ю**. Це може бути як викид отруйних речовин у повітря, так і засоби масового ураження (кулі, ракети, інші снаряди). Таким місцем може бути певна кімната у домі чи спеціальне укриття, які розміщені у містах. Слід відзначити, що такі місця мають бути токими, щоб там можна було провести довгий час, від декількох годин до декількох днів безперервно.

4. **Потреба до переміщення до іншого міста чи країни у надзвичайно стислі терміни**. Унаслідок деяких надзвичайних обставин, може бути потреба за декількох хвилин зібрати найнеобхідніші речі та покинути домівку на тривалий час. Треба мати необхідні речі з собою.

1.4 Постановка задачі

Виходячи із розглянутих можливих надзвичайних ситуацій функціонал додатку повинен вирішувати ряд проблем, описаних у розділі 1. Грунтуючись на отриманій інформації, можемо визначити, який функціонал потрібен користувачу та який додаток може технічно реалізувати. У додатку будемо мати таку сутність як Надзвичайна ситуація. У додатку будемо мати їх 3 види:

1. Загальні
2. Відключення води
3. Відключення електроенергії

У подальшому, перелік надзвичайних ситуацій може бути збільшено.

Користувач буде мати список з дій, котре може зробити для того, щоб підготуватись до тієї чи іншої надзвичайної ситуації. Список речей, які потрібно мати чи дій, котрі потрібно заблагочасно зробити, буде мати кожна надзвичайна ситуація. Коли людина виконує цю дію, тобто вона виконала пункт підготовки до надзвичайної ситуацій, вона натискає на чекбокс (CheckBox). Кожна з надзвичайних ситуацій буде мати свій прогрес. Це буде візуальний прогрес, щоб побачити, наскільки людина готові до тієї чи іншої надзвичайної ситуації. Із збільшенням кількості відмічених чекбоксів буде збільшуватись прогрес. Якщо щось з списку перестає бути таким, що задовольняє умовам виконання пункту підготовки, то чекбокс можно відтиснути. Тобто, якщо була заготовлена технічна вода, але її вилили, то можно в додатку відзначити, що цей пункт вже не є виконаним.

2. МЕТОДИКА ВИРІШЕННЯ ПОСТАВЛЕНИХ ЗАДАЧ

2.1 Активіті

Як і в операційній системі Віндовс використовуються вікна, так само і в операційній системі андроїд існують Activity, що представляє собою окремий екран. У Activity можна розташувати будь-які компоненти користувальницького інтерфейсу чи віджети на екрані телефону.

Важливо відзначити, що у Activity є життєвий цикл. Це означає, що вона може бути в одній з декількох стадій, в залежності від того, у якому стані знаходиться додатком. Подивимось, який життєвий цикл має Активіті.

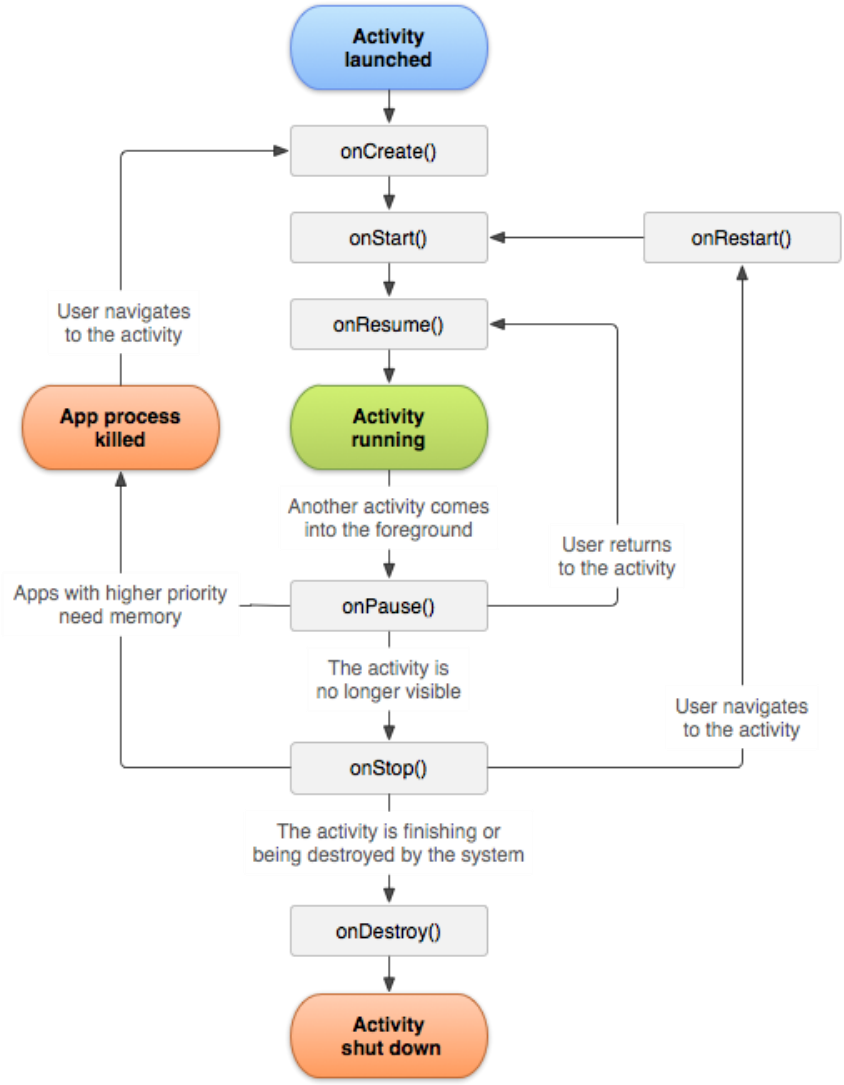

Рисунок 2.1 – Життєвий цикл активіті

Activity дає набір із шести зворотних викликів: onCreate(), onStart(), onResume(), onPause(), onStop() та onDestroy(). Кожен з цих зворотних викликів викликається, коли діяльність переходить у новий стан.

onCreate() – коли стається перша ініціалізації Activity. Потрібно виконати цей метод для виконання ініціалізації у Activity.

onStart() – викликається, коли Activity готова відображатися користувачеві в перший раз. Саме у цей час Activity готується стати інтерактивною та вийти на передній план.

onResume() – коли Activity переходить в цей стан, додаток починає взаємодіяти з користувачем. Activity буде продовжувати працювати у цьому стані, поки що-небудь не відведе фокус від додатка або Activity, наприклад, вхідний дзвінок. Якщо це станеться, то буде викликано метод onPause ().

onPause () – цей метод використовується для припинення дій, які не повинні відбуватися, поки Activity в стадії паузи. Виклик цього методу вказує на те, що користувач покинув додаток. Наприклад, вхідний дзвінок може перевести, програвач музики в стан паузи. Це повинно заглушити або зупинити відтворюється музику.

onStop () – цей метод викликається, якщо Activity більше не видно в додатку. Таке може статися, якщо довантажуючи інша Activity і вона займає весь екран пристрою. Коли викликає цей метод, Activity повідомляється перейти в стан припинення роботи. У цьому стані, система або викликає onRestart () для повернення взаємодії з Activity, або викликає метод onDestroy (), щоб прибрати Activity.

onDestroy () – викликається перед тим, коли Activity буде повністю завершена.

onRestart () – викликається, у тому випадку, коли Activity перезапускается після зупинки.

2.2 Фрагменти

Так як Активіті є дуже ресурсномісткий, то операційна система витрачає багато ресурсів на їх створення. Фрагменти є візуальним об’єктом, котрий можна встроїти в Активіті. Тому може на в одне і те ж Активіті вставляти декілька фрагментів та видаляти їх. Таким чином використовується значно менше ресурсів операційної системи Андроїд. Також, завдяки фрагментам можна краще інтегрувати користувацький інтерфейс для різних типів екранів.

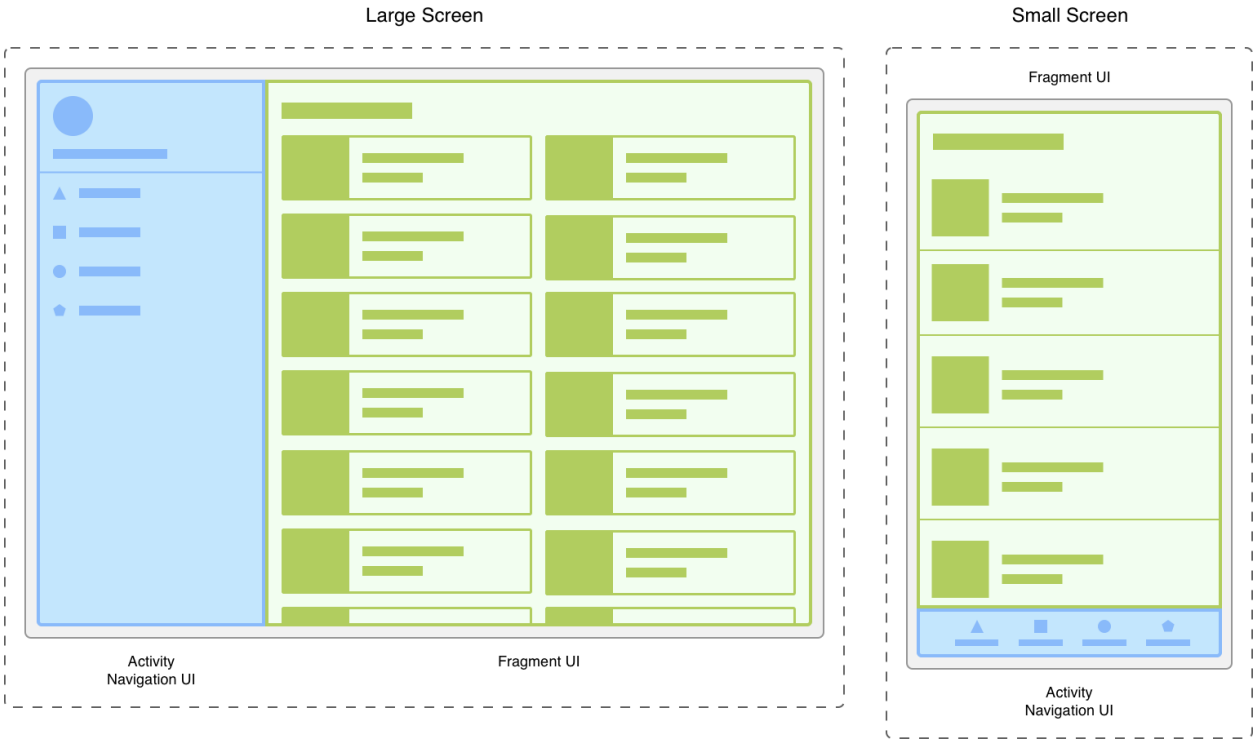

Рисунок 2.2 – Приклад фрагменту на планшеті та телефоні

Фрагменті також життєвий цикл. Він дещо схожий з життєвим циклом Активіті. Можемо побачити його на рисунку 2.3.

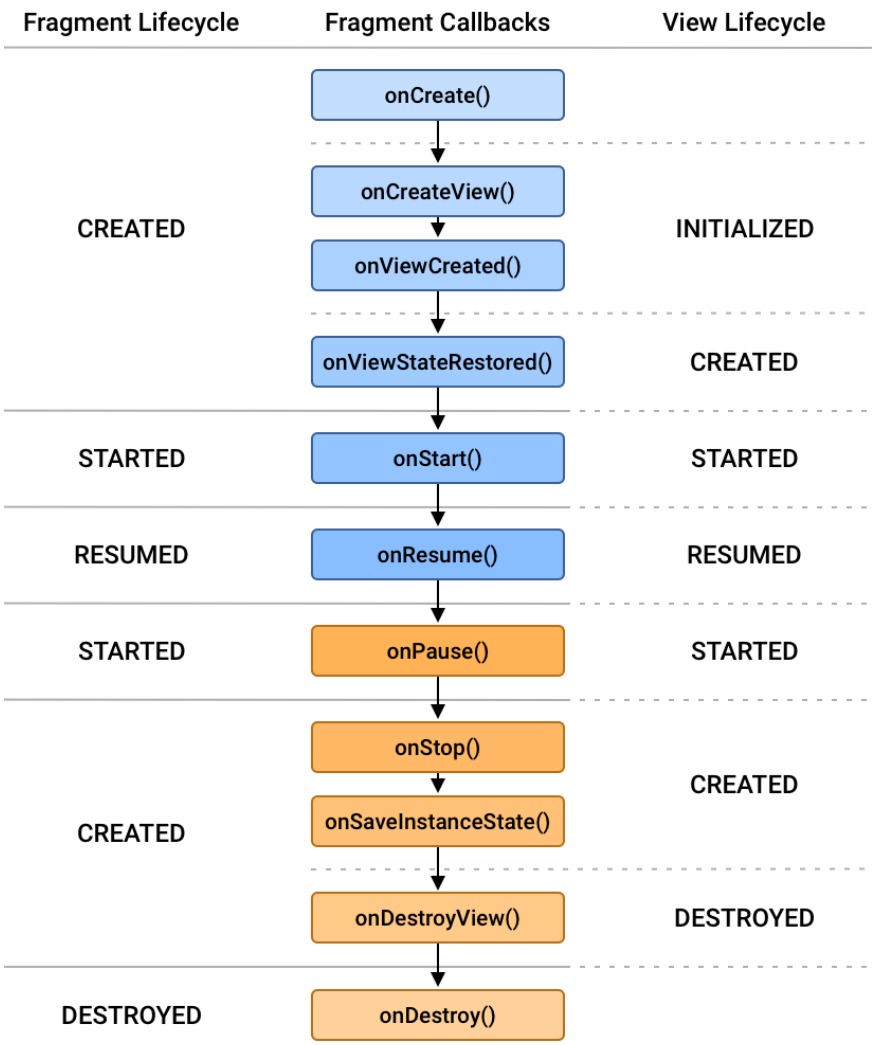

Рисунок 2.3 – Життєвий цикл фрагмента

2.3 Патерн дизайну

Компанія Google, котра займається розвитком мови програмування Android, також створила набір для створення UI-дизайну. Цей набір або паттерн отримав назву Material Design. Під кодовою назвою «Quantum Paper» було випущено Material Design як мову дизайну, яку дизайнери Android могли використовувати для створення кращих програм завдяки гарному дизайну. Google також використовував Material Design, щоб переглянути дизайн своїх програм.

«Material Design» став досить популярним, так як багато дизайнерів інтегрують його не лише в свої Android-проекти, але також і у інші проекти,

які пов’язані з інтернетом. Цей патерн був представлен Google в програмі Google I/O в 2014 році. У цьому році випуск Material Design викликав величезний резонанс серед дизайнерської спільноти – не лише для розробки Android, але й для програм і веб-сайтів iOS.

Матеріальний дизайн або ж с є більш комплексний, ніж абсолютна більшість інших систем проектування. Це саме тому, що він створювався не тільки для одного бренду чи навіть проекту. Загальна мета Material Design була в тому, щоб дозволити дизайнерам як найшвидше створювати програми, які могли б бути досить адаптивними, зручними для користування та легко масштабувалися.

Material Design був натхненний фізичним світом. Справді, те, що зробив Material Design, це відхід від звичного проектування абсолютно плоских інтерфейсів користувацького інтерфейсу до проектування поверхонь. Це було натхнення від папера та чорнил. Якщо дивитися уперед, то аркуш паперу буде виглядати плоским та двовимірним, але у реальному світі все це не так.

Google прагнув перенести фізику елементів з реального навколишнього середовища до цифрового світу Material Design. Саме тому слід використовувати усі паттерни від Google не окремо, а разом.

Як було зазначено раніше, Material Design використовує плоскі елементи. Ці елементи знаходяться у різних площинах і можуть поводитися, як папір або будь які інші об’єкти з реального світу. Розглянемо приклад з офіційного веб-ресурсу Material Design.

Можна бачити, як дія, котра виконується, впливає на те, у якій площині знаходиться сам елемент користувацького інтерфейсу.

Також, можемо побачити один з прикладів патернів та антипатернів. Антипатерн – це антонім слова патерн, тобто це підхід, котрий не рекомендується до використання. У нашому додатку треба використовувати список з елементів, котрі необхідні для надзвичайних ситуацій.

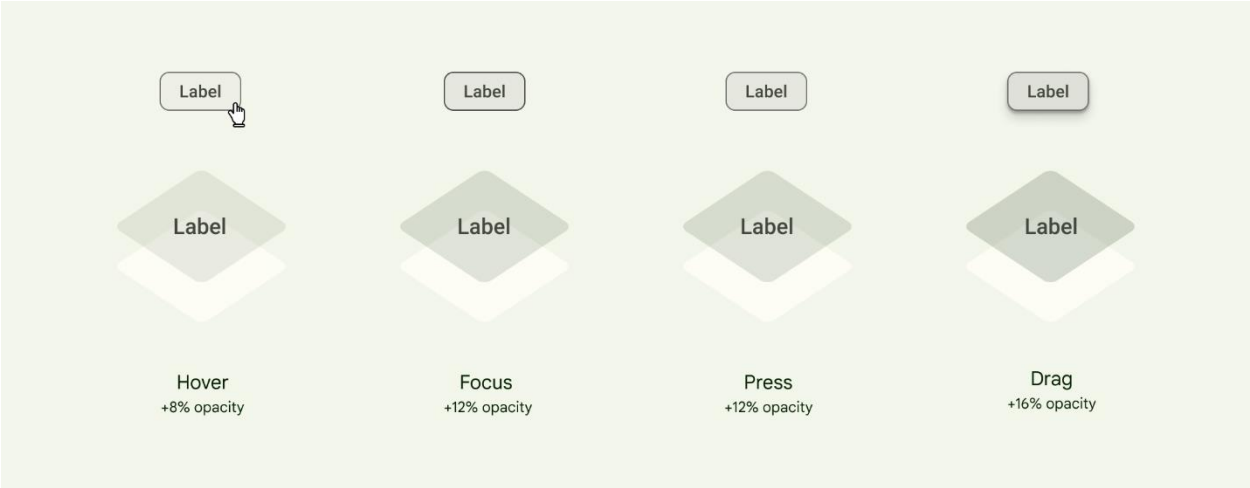

Рисунок 2.4 – Об'єм кнопки в залежності від її стану

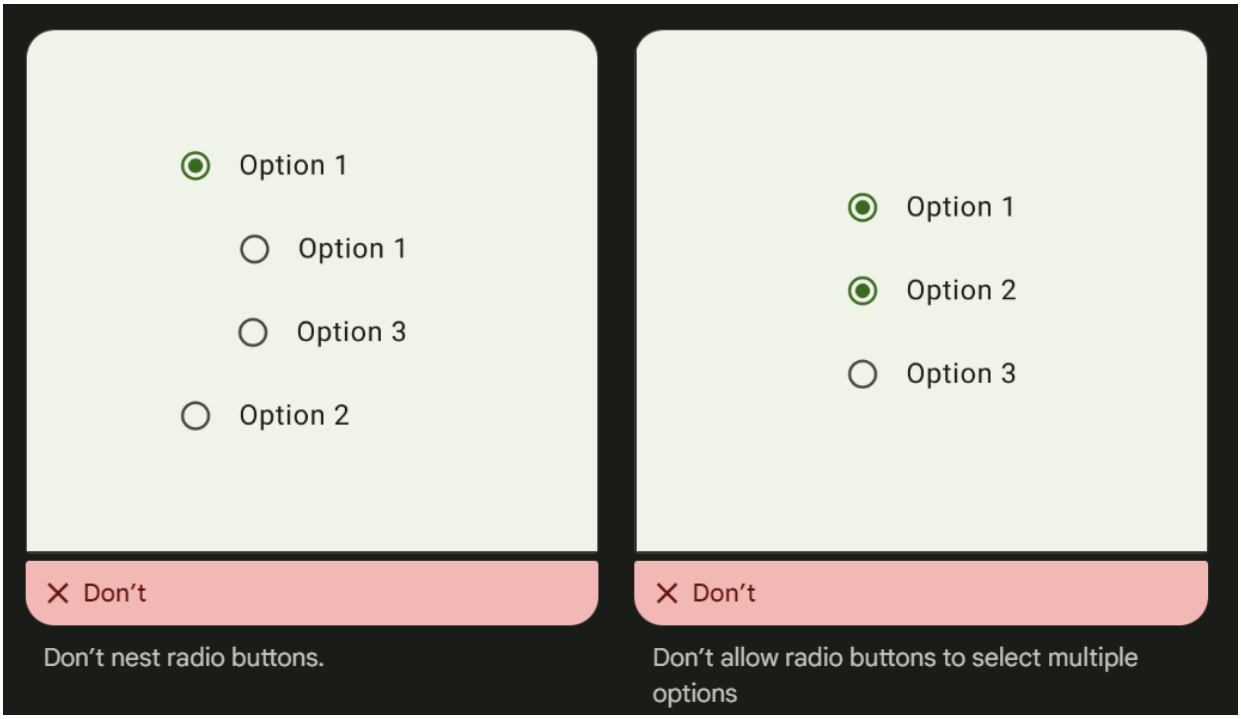

Рисунок 2.5 – Антипатерни дизайну

Таким чином можна було переконатися, що такий підхід не є рекомендованим до використання. Не треба використовувати option для складних ієрархій. Натомість, краще використовувати checkbox.

Ці патерни дозволяють користувачам додатку більш природно реагувати на UI, так як вони розуміють, як торкатися об'єктів, рухатися навколо них так само, як вони це б робили у фізичному середовищі.

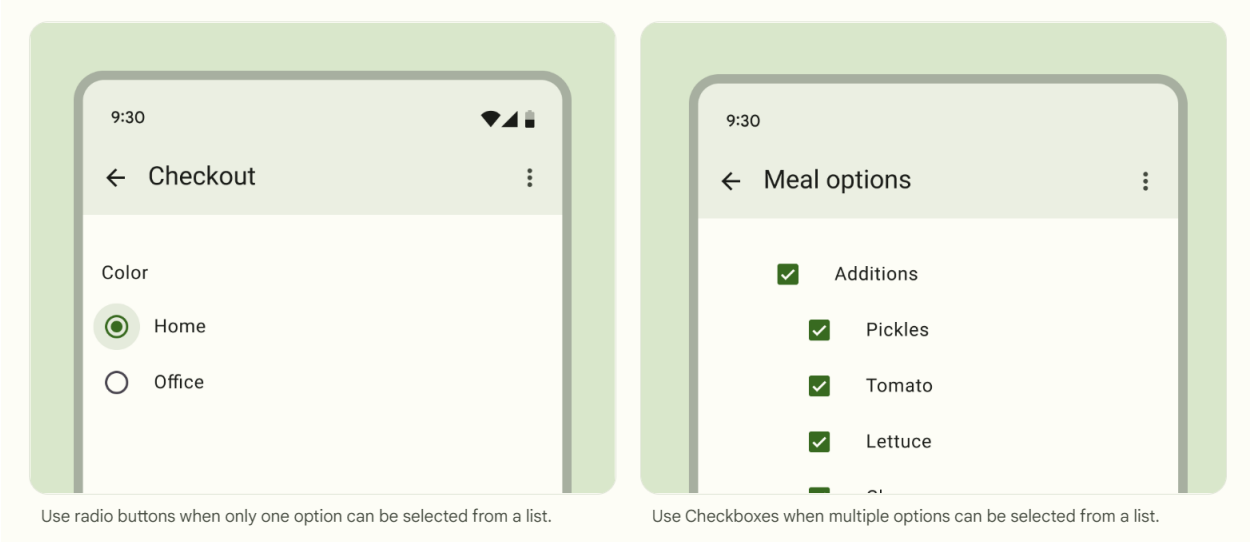

Рисунок 2.6 – Приклади вдалого дизайну чекбоксів та опшнів

Тобто, замість того, щоб просити користувачів додатку увійти в цифровий світ, який здається їм неприродним, набір паттернів Material Design застосовує основні принципи фізичного середовища до програм, додатків, сайтів.

Існує декілька версій Material Design. Остання версія, Material 3, забезпечує індивідуальний, адаптивний та виразний досвід – від динамічних кольорів й покращених можливостей до основ для макетів великих екранів чи маркерів дизайну.

2.4 Колір і контраст

Як кольори, так і контраст можна використовувати, щоб допомогти користувачам відчути вміст програми, взаємодіяти з потрібними елементами та інтуїтивно розуміти дії.

Колір здатен допомогти передати не тільки настрій чи тон, а і важливу інформацію. Для зручного використання можна вибрати основні, додаткові, акцентні кольори. Достатній колірний контраст між елементами може допомогти користувачам із слабким зором бачити та використовувати вашу програму.

Динамічний колір описує можливість системи кольорів застосовувати колір до різних джерел чи умов, наприклад, налаштування кольору та уподобань користувача. Система кольорів Material 3 і спеціальні схеми є основою включення динамічних кольорів у програмах. Матеріал інтерфейсу надає значення за замовчуванням для кольору та типу як відправну точку для налаштування.

Продукти можуть визначати додаткові спеціальні кольори, які мають значення (для семантичної цінності, доступності тощо). Визначення власних кольорів дозволяє командам включати та зберігати необхідні кольорові повністю. Динамічний колір, який працює як з компонентами Material, він також працює і з настроюваними компонентами.

Для інтерфейса користувача програма може отримати колір із базової схеми, з динамічних схем, які отримують схему з шпалер користувача, чи з динамічних схем. Приділено багати ували кольору. Кольорові теми розроблено такими, щоб бути достатньо гармонійними, забезпечувати доступність тексту та мати змогу відрізняти елементи інтерфейсу користувача та поверхні один від одного.

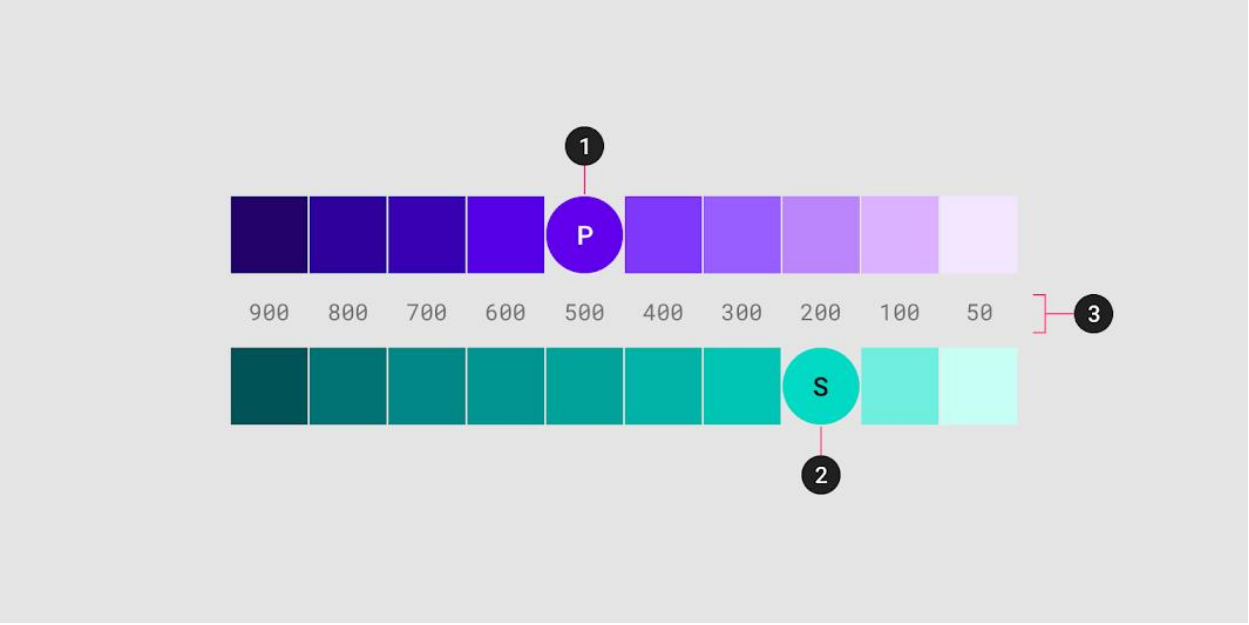

Рисунок *2.7* – Приклад вибору кольорів на о сайті Material Design

Розглянемо інструмент для створення палітри кольорів для власного додатку. Інструмент палітри Material Design або палітри Material Design 2014 доступні, щоб допомогти вибрати кольори.

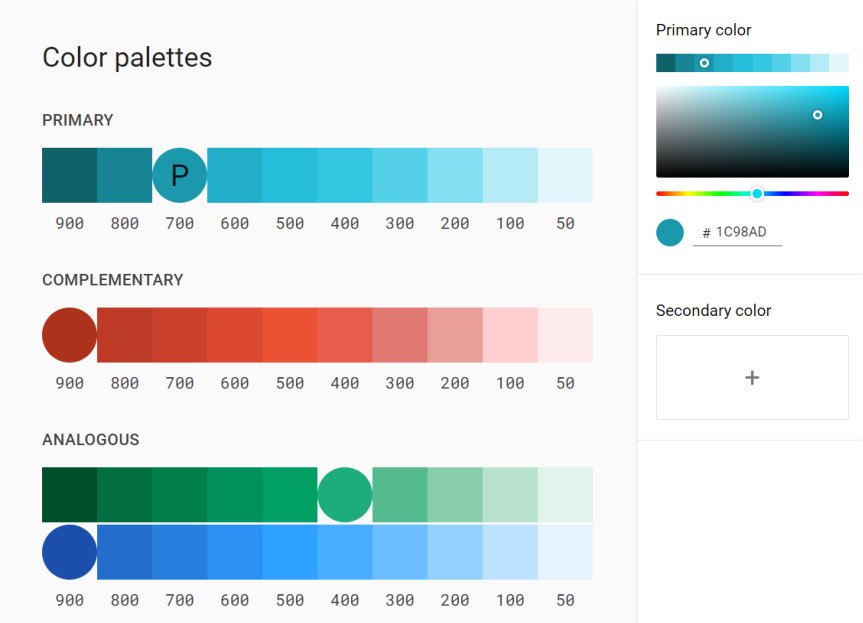

Рисунок 2.8 – Вибір кольорів для додатку у реальному часі

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1. Середовище для створення

Для створення візуальної частити додатку, інакше кажучи UI- та UX-дизайну, використовуємо Android Studio, що була розроблена компанією JetBrains. Цю середу розробки можна відзначити зручністю та надійністю. Для створення. Для створення візуальної частини додатку на андроїд є різні типи layout . Для нашого додатку обираємо Constraint Layout.

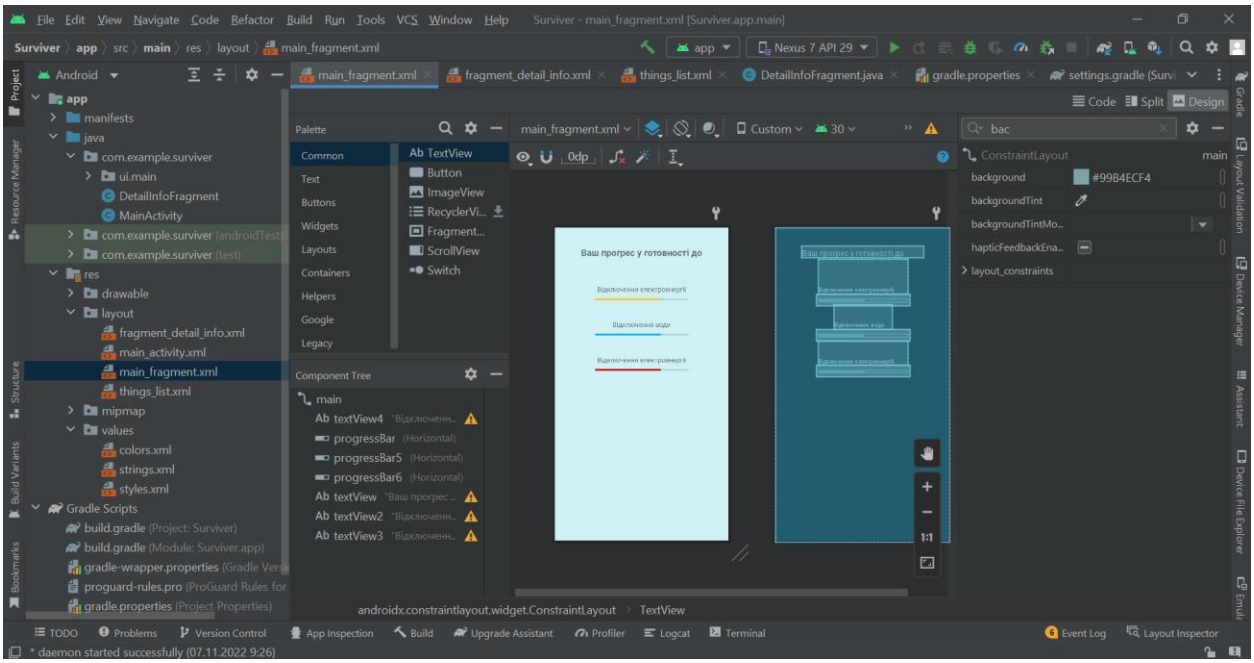

Рисунок 3.1 – Середовище розробки Android Studio

Середовище розробки Android Studio перетворює макет, який було створено за допомогою UI елементів у xml формат та навпаки. Для досягнення найкращого результату будемо використовувати xml-код та інструменти для UI дизайну від Android Studio.

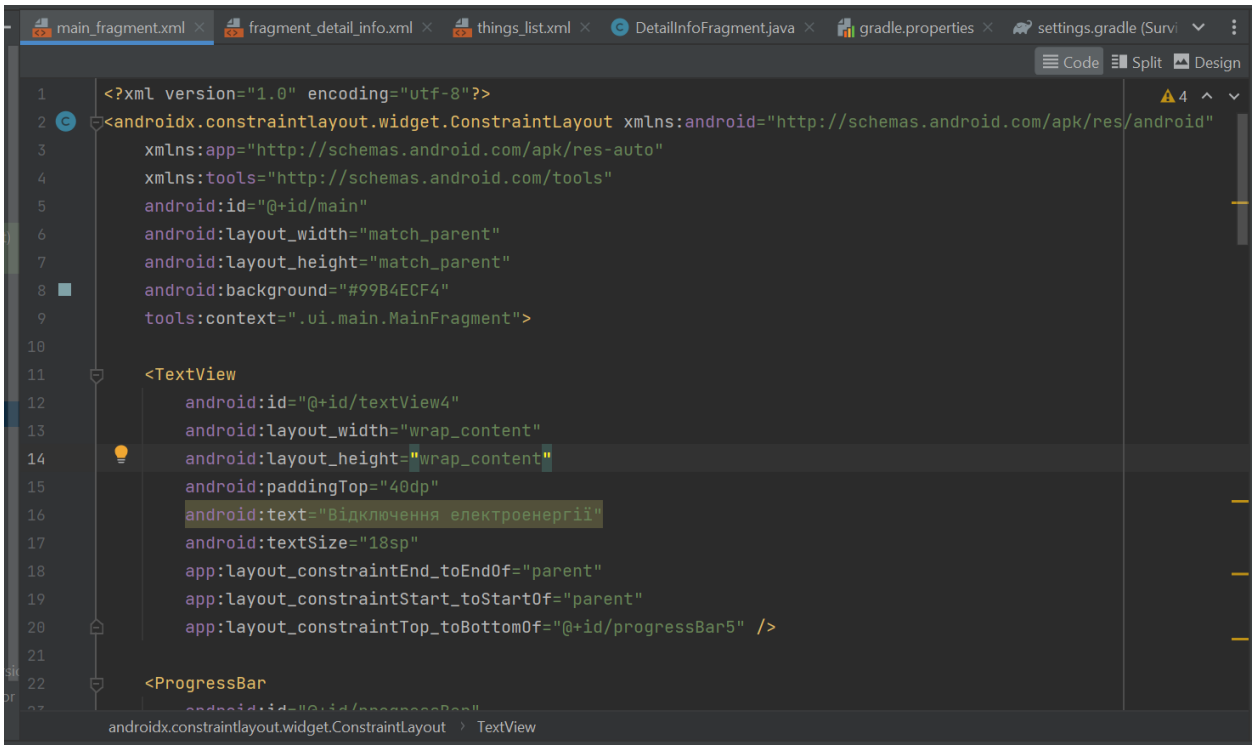

Рисунок 3.2 – Створення UI за допомогою коду

Розробка коду починається наступним чином. Необхідно проголосити, який констраінт лейауту будемо використовувати та описуємо його. Далі вже йдуть самі елементі. Наприклад, в нашому коді далі йде елемент TextView.

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/main"

android:layout_width="match_parent"

android:layout_height="match_parent"

android:background="#99B4ECF4"

tools:context=".ui.main.MainFragment">

< TextView

android:id="@+id/textView4"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

```
    android:paddingTop="40dp"
    android:text="Відключення електроенергії"
    android:textSize="18sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/progressBar5" />
```

Розглянемо типовий прогрес бар, котрий будемо використовувати на головному екрані нашого додатку. Тут ми визначаємо атрибути, розмір, кольори. Також, можемо визначити і інші властивості.

```
<ProgressBar
    android:id="@+id/progressBar"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="250dp"
    android:layout_height="30dp"
    android:max="100"
    android:progress="70"
    android:progressTint="#FFC107"
    app:layout_constraintEnd_toEndOf="@+id/textView2"
    app:layout_constraintHorizontal_bias="0.488"
    app:layout_constraintStart_toStartOf="@+id/textView2"
    app:layout_constraintTop_toBottomOf="@+id/textView2" />
```

Оскільки операційна система Android використовується для пристроїв з різними розмірами екранів, то не використовуємо абсолютні розмітки та розміри. Натомість, використовуємо відносні одиниці виміру при створенні візуальної частини нашого додатку.

У результаті, отримуємо головний екран, який відображає можливі надзвичайні ситуації та степінь готовності до них

Рисунок 3.3 – UX-дизвайн головної сторінки додатку

Для зонування показників різних надзвичайних ситуацій у нашій інформаційній системі, використаємо різні кольори для них. Це зробить візуальну частину додатку більш удобною для користування .

Можемо побачити, що користувацький інтерфейс не є перевантаженим різними елементами. Таким додатком буде легко користуватись людям усім власникам смартфонів, незалежно від віку

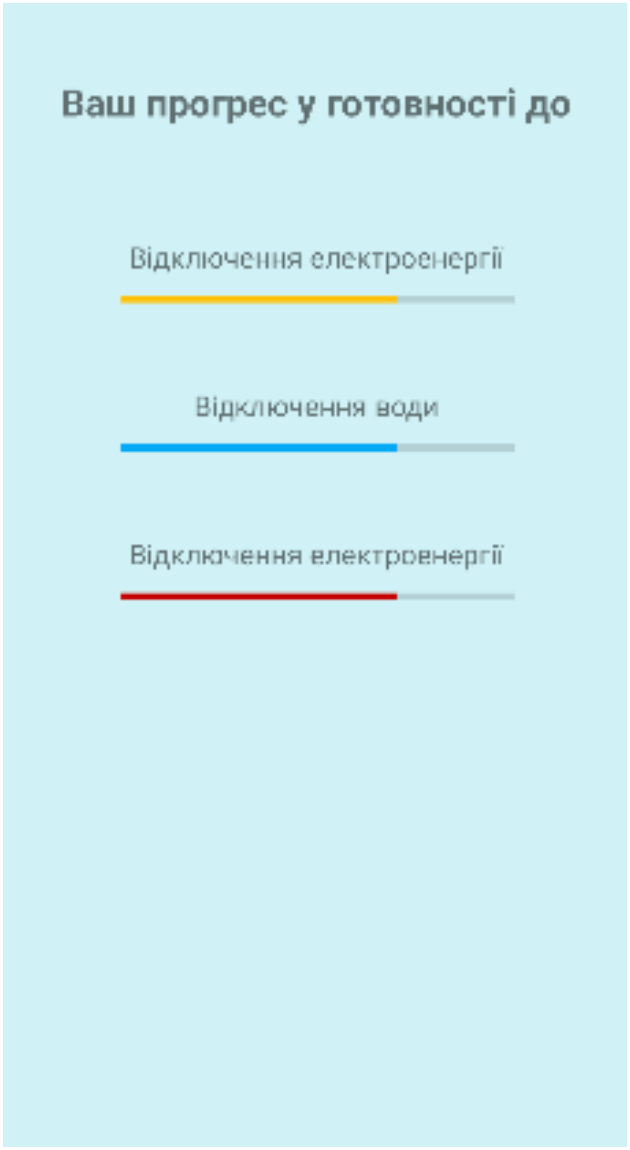

Рисунок 3.4 – UX-дизайн сторінки з надзвичайними ситуаціями

Коли здійснюється натискання на назву надзвичайної ситуації, то відкривається наступний екран, де можна побачити, яким чином можна до неї подготуватися. Під назвою надзвичайної витуації відображається список пунктів / речей, котрі потрібні для підготовки до надзвичайної ситуації. Дизайн є досить мінімалістичним. Ніяких зайвих елементів не було додано на сторінці.

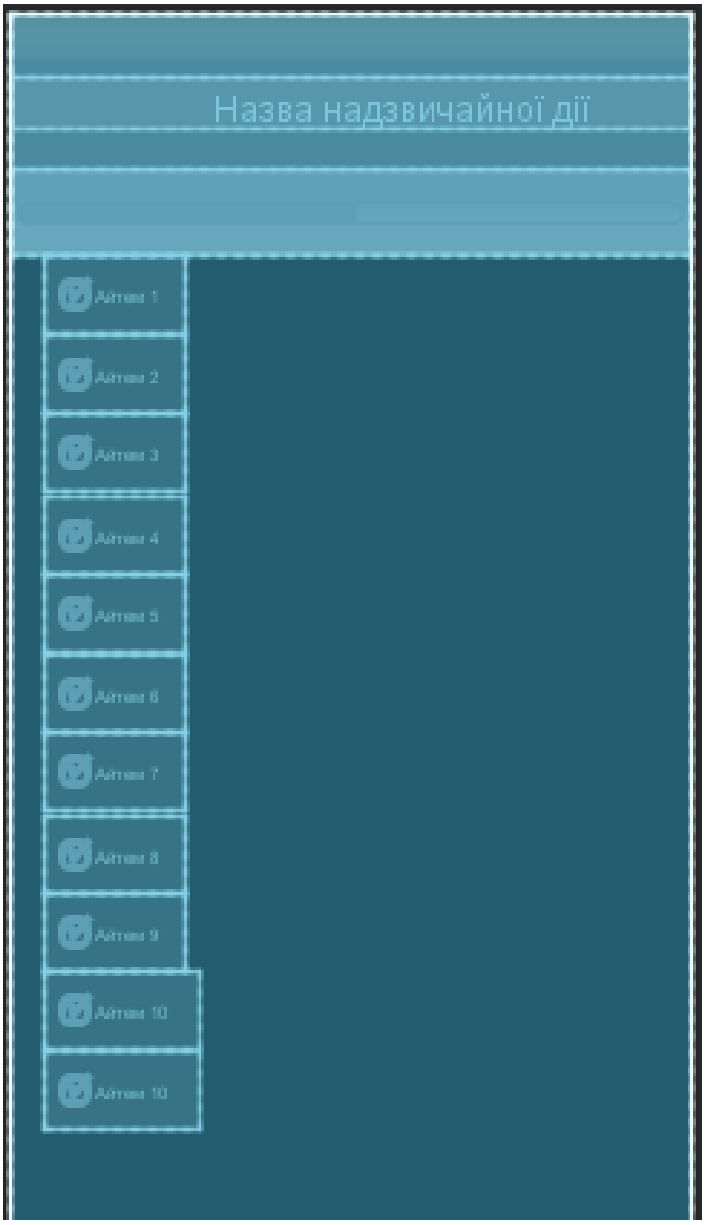

Рисунок 1.5 – UX-дизайн сторінки з надзвичайними ситуаціями

Верхня частина екрану винесена в окремий LinearLayout.

```
<LinearLayout
android:id="@+id/linearLayout"
android:layout_width="409dp"
android:layout_height="146dp"
android:background="#97C4E8"
android:orientation="vertical"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent">
```

```
    <TextView
        android:id="@+id/textView6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:text=" Назва надзвичайної дії"
        android:textAlignment="center"
        android:textSize="24sp"
        android:textStyle="bold" />

    <ProgressBar
        android:id="@+id/progressBar2"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="20dp"
        android:max="100"
        android:minHeight="30dip"
        android:progress="70"
        android:progressTint="#FFC107" />
</LinearLayout>
```

Рисунок 3.62 – UI-дизайн сторінки з надзвичайними ситуаціями

У результаті отримали досить мінімалістичний дизайн зі спокійними кольорами. Робота із список з дії підготовки до надзвичайної ситуації також є інтуїтивно простою.

3.2 Створення бізнес-логіки

Створення бізнес логіки відбувається за допомогою мови програмування Java у поєднанні з фреймворком Android. Цей фреймворк має безліч функцій та підходів для написання додатків.

Під час створення бізнес логіки будемо опиратися на розповсюджену практику програмування – використовувати принципи об'єктно-орієнтоване програмування (далі – ООП) [13].

Головною сутністю у нашому додатку є надзвичайна ситуація. Створюємо клас для представлення її у нашому коді.

```
public class EmergentSituation {
  @NonNull
  @PrimaryKey
  private String title;
  private List<Point> points;

}
```

Як бачимо, наша сутність має всього два поля. Поле «title» представляє назву надзвичайної ситуації, а поле «points» представляє пункти, які можуть допомогти у підготовці до надзвичайної ситуації. За допомогою анотації @NonNull позначаємо, що це поле не може бути пустим або мати значення null. А за допомогою анотації @PrimaryKey позначаємо, що це поле є ключем та обов’язково має бути унікальним.

Перейдемо до класу, котрий представляє points. Створюємо у ньому 2 поля. Перше представляє собою саму пораду, що робити у надзвичайній ситуації, а друге – чи є ця дія поміченою, як виконана.

```
public class Point {
  private String name;
  private boolean isChecked;

  public String getName() {
     return name;
  }

  public void setName(String name) {
     this.name = name;
  }

  public boolean isChecked() {
```

```
        return isChecked;
    }

    public void setChecked(boolean checked) {
        isChecked = checked;
    }
}
```

Для всіх створених нами класів додтримуємося принципів ООП. У даному випадку ми використовуємо принцип інкапсуляції. Доступ до полів даємо за допомогою спеціальних методів, які також називають гетери та сетери.

3.3 Зберігання даних

Анотації потрібні, бо дані щодо наших надзвичайних ситуацій будуть зберігатися у базі даних. Для цього додатку використовуємо RoomDatabase. Ця база даних є оптимальною для мобільних додатків. Вона є легкою та швидкою.

```
@Database(entities = {EmergentSituation.class}, version = 1)
@TypeConverters(Converter.class)
public abstract class EmergentSituationDatabase extends RoomDatabase {
    private static EmergentSituationDatabase INSTANCE;
    private static final int NUMBER_OF_THREADS = 4;

    public abstract ActionDao actionDao() throws SQLiteConstraintException;

    public static final ExecutorService databaseExecutor =
            Executors.newFixedThreadPool(4);

    public static EmergentSituationDatabase getDatabase(final Context context){
        if (INSTANCE == null) {
            synchronized (EmergentSituationDatabase.class) {
```

```
            if (INSTANCE == null) {
                INSTANCE = Room.databaseBuilder(context.getApplicationContext(),
                        EmergentSituationDatabase.class, "emerdgsit-database")
                    .fallbackToDestructiveMigration()
                    .build();
            }
        }
    }
    return INSTANCE;
  }
}
```

Розберемо детальніше анотації, котрі ми використовуємо у нашому коді.

Таблиця 3.1 – Використані анотації при розробці додатку

Entity (tableName = "emergentsituations")	Назва таблиці у базі даних.
@NonNull	Означає, що поле не може бути не ініціалізованим.
@PrimaryKey	Означає, що з цього поля буде складатися ключ для запису у базі даних.

Перейдемо до інтерфейсу, за допомогою котрого будемо робити запити до бази даних.

```
@Dao
public interface SituationDao {
    @Query("SELECT * FROM emergsit")
    Flowable<List<Situation>> getAllActions();

    @Query("SELECT * FROM emergsit WHERE title = :title")
    Action getSituationByTitle(String title);

    @Query("SELECT * FROM emergsit")
```

```
    List<Action> getSituationList();

    @Query("DELETE FROM emergsit")
    void deleteAllSituation();

    @Delete
    void deleteSituation(EmergentSituation situation);

    @Insert
    void insertSituation(EmergentSituation situation) throws SQLiteConstraintException;

}
```

3.4 Взаємодія з візуальною частиною додатку

При натисканні на будь який елемент нашого додатку, можна назначити певну дію. Це можуть бути як кнопки, так и поля з текстом або будь які інші елементи. Розглянемо приклад реакції нашого додатку на певні дії зі сторони користувача:

```
@Nullable
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container,
                         @Nullable Bundle savedInstanceState) {

    TextView noElectricityTextView = (TextView) findViewById(R.id.textView2);
    TextView noWaterTextView = (TextView) findViewById(R.id.textView3);
    TextView basicSituationsTextView = (TextView) findViewById(R.id.textView4);

    noElectricityTextView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            savedInstanceState.putString("situationName", "Відключення електроенергії");
```

```
            getActivity().getSupportFragmentManager().beginTransaction()
                    .replace(R.id.detailInfoFragment, new DetailInfoFragment())
                    .addToBackStack(null)
                    .commit();
        }
    });
    noWaterTextView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            savedInstanceState.putString("situationName", "Відключення води");
            getActivity().getSupportFragmentManager().beginTransaction()
                    .replace(R.id.detailInfoFragment, new DetailInfoFragment())
                    .addToBackStack(null)
                    .commit();
        }
    });
    basicSituationsTextView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            savedInstanceState.putString("situationName", "Базова ситуація");
            getActivity().getSupportFragmentManager().beginTransaction()
                    .replace(R.id.detailInfoFragment, new DetailInfoFragment())
                    .addToBackStack(null)
                    .commit();
        }
    });

    return inflater.inflate(R.layout.main_fragment, container, false);
}
```

Можемо бачити у цьому коді, що було знайдено елементи в xml документі, котрий представляє собою головну сторінку нашого додатку. Далі, за допомогою метода OnClickListener було створено дію, котра виконується при натисканні на назву надзвичайної ситуації.

3.5 Тестування

Подивимось, як виглядає наш додаток при вже внесених даних. Відкриваємо додаток на бачимо наступне.

Можемо побачити, що є прогрес тільки для базової ситуації. Натиснемо на неї, щоб побачити детальну інформацію та список дій, що допоможуть підготуватися до цією базової надзвичайної ситуації. Відкривається новий фрагмент з детальної інформацією.

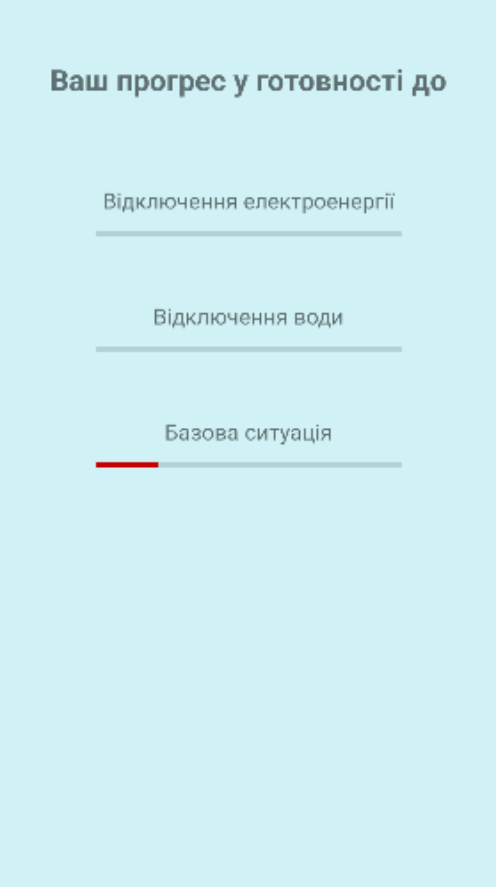

Рисунок 3.7 – Головне меню додатку

На рисунку 3.8 можемо бачити, що відкрився список з речами, котрі можуть допомогти у надзвичайній ситуації.

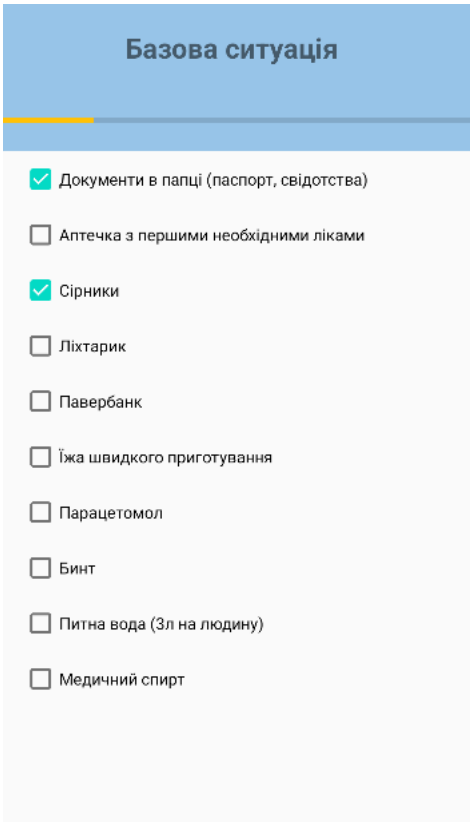

Рисунок 3.8 – Детальна інформація о базовій надзвичайній ситуації

Бачимо, що у є прогрес для цієї надзвичайної ситуації та також користувач може проставляти галочки у чекбокси для інформування про те, що необхідна дія підготовки є виконаною. Також, можена прибирати позначки з чекбоксів, що дія виконана.

ВИСНОВКИ

У результаті написання магістерської роботи, було досліджено існування проблеми підготовки до надзвичайних ситуацій. Було розглянуто, які є саме є надзвичайні ситуації, та як до них підготуватись. На базі цієї інформації було вирішено, який саме функціонал потрібен додатку для вирішення поставлених проблем. Для інтуїтивного користування додатком, було вирішено дотримуватись принципів мінімалізму для UX-дизайну. UI-дизайн було створено дотримуючись принципів Material Design.

Оскільки додаток працює на операційній системі Андроїд, то і методологія дизайну використовується від компанії Google, котра володіє операційною системою. У роботі проведено огляд технологій, які є актуальними для створення додатків на андроїд. Серед мов програмування під Андроїд була обрана одна з найбільш розповсюджених мов програмування – Java. Вона є перспективною та вже розвиненою. Патерн проектування, який було обрано для архітектури система було створено також компанією Google. Бізнес логіка додатка відповідає такій, що була описана у роботі. У якості бази даних обрано Room Database.

Тож, у сукупності всі вище описані фактори, роблять додаток актуальним, клієнтоорієнтованим та легким для підтримки чи подальшому розвитку.

СПИСОК ЛІТЕРАТУРИ

1. Надзвичайна ситуація - https://vue.gov.ua/Надзвичайна_ситуація

2. Дослідження: 63% населення планети мають доступ до інтернету, а в 67% є мобільний телефон - https://dev.ua/news/63-naselennia-planety-maiut-internet

3. Радіо свобода. За рік на 10% більше літніх українців почали користуватись смартфонами – статистика - https://www.radiosvoboda.org/a/28786830.html

4. УНІАН. Як зібрати аптечку під час війни: список основних ліків - https://www.unian.ua/health/shcho-poklasti-v-domashnyu-aptechku-pid-chas-viyni-spisok-likiv-11747491.html

5. УНІАН. Смартфони є у 55% українців, серед молоді - 92% (інфографіка) - https://www.unian.net/economics/telecom/10500207-smartfony-est-u-55-ukraincev-sredi-molodezhi-92-infografika.html

6. Суспільне. Немає води та світла: як одеситам підготуватись до можливих відключень - https://suspilne.media/292048-nemae-vodi-ta-svitla-ak-odesitam-pidgotuvatis-do-mozlivih-vidklucen/

7. Гагадгед. Play Market: що це таке і навіщо воно вам - https://gagadget.com/uk/44431-play-market-shcho-tse-take-i-navishcho-vono-vam/

8. Гугл плей. Disaster Alert- https://play.google.com/store/apps/details?id=disasterAlert.PDChttps://play.google.com/store/apps/details?id=disasterAlert.PDC

9. Гугл плей. Особиста безпека - https://play.google.com/store/apps/details?id=com.google.android.apps.safetyhub

10. Material Design. Introduction - https://material.io/design/introduction

11. Wikipedia. Google play - https://en.wikipedia.org/wiki/Google_Play

12. Офіційний сайт Material Design - https://m2.material.io/design/color/the-color-system.html#tools-for-picking-colors

13. Мєдіум - https://medium.com/@cancerian0684/what-are-four-basic-principles-of-object-oriented-programming-645af8b43727

ДОДАТКИ

Додаток А.
Код з візуальною частиною додатку

main_fragment.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/mainFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#99B4ECF4"
    tools:context=".ui.main.MainFragment">

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="40dp"
        android:text="Базова ситуація"
        android:textSize="18sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/progressBar5" />

    <ProgressBar
        android:id="@+id/progressBar"
```

```
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="250dp"
    android:layout_height="30dp"
    android:max="100"
    android:progress="70"
    android:progressTint="#FFC107"
    app:layout_constraintEnd_toEndOf="@+id/textView2"
    app:layout_constraintHorizontal_bias="0.488"
    app:layout_constraintStart_toStartOf="@+id/textView2"
    app:layout_constraintTop_toBottomOf="@+id/textView2" />

<ProgressBar
    android:id="@+id/progressBar5"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="250dp"
    android:layout_height="30dp"
    android:max="100"
    android:progress="70"
    android:progressTint="#03A9F4"
    app:layout_constraintEnd_toEndOf="@+id/textView2"
    app:layout_constraintHorizontal_bias="0.488"
    app:layout_constraintStart_toStartOf="@+id/textView2"
    app:layout_constraintTop_toBottomOf="@+id/textView3" />

<ProgressBar
    android:id="@+id/progressBar6"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="250dp"
    android:layout_height="30dp"
    android:max="100"
```

```
    android:progress="70"
    android:progressTint="#C60000"
    app:layout_constraintEnd_toEndOf="@+id/textView2"
    app:layout_constraintHorizontal_bias="0.488"
    app:layout_constraintStart_toStartOf="@+id/textView2"
    app:layout_constraintTop_toBottomOf="@+id/textView4" />

<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:text="Ваш прогрес у готовності до"
    android:textSize="24sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="70dp"
    android:text="Відключення електроенергії"
    android:textSize="18sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView" />
```

```
    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="40dp"
        android:text="Відключення води"
        android:textSize="18sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/progressBar" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

main_activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <fragment
        android:id="@+id/nav_host_fragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:defaultNavHost="true"
```

```
    app:navGraph="@navigation/main_graph" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

fragment_detail_info.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/mainFragment"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".ui.main.MainFragment">

   <LinearLayout
      android:id="@+id/linearLayout"
      android:layout_width="409dp"
      android:layout_height="146dp"
      android:background="#97C4E8"
      android:orientation="vertical"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent">

      <TextView
         android:id="@+id/textView6"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:layout_marginTop="40dp"
```

```
        android:text="          Назва надзвичайної дії"
        android:textAlignment="center"
        android:textSize="24sp"
        android:textStyle="bold" />

    <ProgressBar
        android:id="@+id/progressBar2"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="20dp"
        android:max="100"
        android:minHeight="30dip"
        android:progress="70"
        android:progressTint="#FFC107" />
</LinearLayout>

<CheckBox
    android:id="@+id/checkBox6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:minHeight="48dp"
    android:text="Айтем 6"
    app:layout_constraintStart_toStartOf="@+id/checkBox1"
    app:layout_constraintTop_toBottomOf="@+id/checkBox5" />

<CheckBox
    android:id="@+id/checkBox7"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:minHeight="48dp"
    android:text="Айтем 7"
    app:layout_constraintStart_toStartOf="@+id/checkBox1"
    app:layout_constraintTop_toBottomOf="@+id/checkBox6" />

<CheckBox
    android:id="@+id/checkBox8"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:minHeight="48dp"
    android:text="Айтем 8"
    app:layout_constraintStart_toStartOf="@+id/checkBox1"
    app:layout_constraintTop_toBottomOf="@+id/checkBox7" />

<CheckBox
    android:id="@+id/checkBox9"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:minHeight="48dp"
    android:text="Айтем 9"
    app:layout_constraintStart_toStartOf="@+id/checkBox1"
    app:layout_constraintTop_toBottomOf="@+id/checkBox8" />

<CheckBox
    android:id="@+id/checkBox10"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:minHeight="48dp"
    android:text="Айтем 10"
    app:layout_constraintStart_toStartOf="@+id/checkBox1"
```

```
    app:layout_constraintTop_toBottomOf="@+id/checkBox9" />

<CheckBox
    android:id="@+id/checkBox11"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:minHeight="48dp"
    android:text="Айтем 10"
    app:layout_constraintStart_toStartOf="@+id/checkBox10"
    app:layout_constraintTop_toBottomOf="@+id/checkBox10" />

<CheckBox
    android:id="@+id/checkBox3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:minHeight="48dp"
    android:text="Айтем 3"
    app:layout_constraintStart_toStartOf="@+id/checkBox2"
    app:layout_constraintTop_toBottomOf="@+id/checkBox2" />

<CheckBox
    android:id="@+id/checkBox4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:minHeight="48dp"
    android:text="Айтем 4"
    app:layout_constraintStart_toStartOf="@+id/checkBox1"
    app:layout_constraintTop_toBottomOf="@+id/checkBox3" />

<CheckBox
```

```
        android:id="@+id/checkBox5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:minHeight="48dp"
        android:text="Айтем 5"
        app:layout_constraintStart_toStartOf="@+id/checkBox1"
        app:layout_constraintTop_toBottomOf="@+id/checkBox4" />

    <CheckBox
        android:id="@+id/checkBox2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:minHeight="48dp"
        android:text="Айтем 2"
        app:layout_constraintStart_toStartOf="@+id/checkBox1"
        app:layout_constraintTop_toBottomOf="@+id/checkBox1" />

    <CheckBox
        android:id="@+id/checkBox1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:minHeight="48dp"
        android:text="Айтем 1"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linearLayout" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

ic_launcher_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="108dp"
    android:height="108dp"
    android:viewportWidth="108"
    android:viewportHeight="108">
    <path
        android:fillColor="#3DDC84"
        android:pathData="M0,0h108v108h-108z" />
    <path
        android:fillColor="#00000000"
        android:pathData="M9,0L9,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M19,0L19,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M29,0L29,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M39,0L39,108"
        android:strokeWidth="0.8"
```

```
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M49,0L49,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M59,0L59,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M69,0L69,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M79,0L79,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M89,0L89,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M99,0L99,108"
    android:strokeWidth="0.8"
```

```
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,9L108,9"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,19L108,19"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,29L108,29"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,39L108,39"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,49L108,49"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,59L108,59"
    android:strokeWidth="0.8"
```

```
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,69L108,69"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,79L108,79"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,89L108,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,99L108,99"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,29L89,29"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,39L89,39"
    android:strokeWidth="0.8"
```

```
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,49L89,49"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,59L89,59"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,69L89,69"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,79L89,79"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M29,19L29,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M39,19L39,89"
    android:strokeWidth="0.8"
```

```
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M49,19L49,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M59,19L59,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M69,19L69,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M79,19L79,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
</vector>
```

ic_launcher_foreground.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:aapt="http://schemas.android.com/aapt"
    android:width="108dp"
    android:height="108dp"
    android:viewportWidth="108"
```

```
    android:viewportHeight="108">
    <path android:pathData="M31,63.928c0,0 6.4,-11 12.1,-13.1c7.2,-2.6 26,-1.4
26,-1.4l38.1,38.1L107,108.928l-32,-1L31,63.928z">
        <aapt:attr name="android:fillColor">
            <gradient
                android:endX="85.84757"
                android:endY="92.4963"
                android:startX="42.9492"
                android:startY="49.59793"
                android:type="linear">
                <item
                    android:color="#44000000"
                    android:offset="0.0" />
                <item
                    android:color="#00000000"
                    android:offset="1.0" />
            </gradient>
        </aapt:attr>
    </path>
    <path
        android:fillColor="#FFFFFF"
        android:fillType="nonZero"
        android:pathData="M65.3,45.828l3.8,-6.6c0.2,-0.4 0.1,-0.9 -0.3,-1.1c-0.4,-0.2
-0.9,-0.1 -1.1,0.3l-3.9,6.7c-6.3,-2.8 -13.4,-2.8 -19.7,0l-3.9,-6.7c-0.2,-0.4 -0.7,-0.5 -
1.1,-0.3C38.8,38.328 38.7,38.828 38.9,39.228l3.8,6.6C36.2,49.428 31.7,56.028
31,63.928h46C76.3,56.028 71.8,49.428 65.3,45.828zM43.4,57.328c-0.8,0 -1.5,-
0.5 -1.8,-1.2c-0.3,-0.7 -0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1
1.2,1.8C45.3,56.528 44.5,57.328 43.4,57.328L43.4,57.328zM64.6,57.328c-0.8,0 -
1.5,-0.5 -1.8,-1.2s-0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1
1.2,1.8C66.5,56.528 65.6,57.328 64.6,57.328L64.6,57.328z"
```

```
        android:strokeWidth="1"
        android:strokeColor="#00000000" />
</vector>
```

colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#6200EE</color>
    <color name="colorPrimaryDark">#3700B3</color>
    <color name="colorAccent">#03DAC5</color>
</resources>
```

ДОДАТОК Б

Код з конфігурацією проекту

gradle.xml

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 32
    buildToolsVersion "33.0.0"

    defaultConfig {
        applicationId "com.example.surviver"
        minSdkVersion 29
        targetSdkVersion 32
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
        }
    }

    viewBinding{
```

```
        enabled = true
    }
}

dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'androidx.lifecycle:lifecycle-extensions:2.2.0'
    implementation 'androidx.cardview:cardview:1.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

    def room_version = "2.2.5"

    implementation "androidx.room:room-runtime:$room_version"
    annotationProcessor "androidx.room:room-compiler:$room_version"

    // optional - RxJava support for Room
    implementation "androidx.room:room-rxjava2:$room_version"

    // Java RX include
    implementation 'com.squareup.retrofit2:adapter-rxjava2:2.3.0'
    implementation "io.reactivex.rxjava2:rxandroid:2.0.1"
    implementation "io.reactivex.rxjava2:rxjava:2.1.7"

    // optional - Guava support for Room, including Optional and ListenableFuture
    implementation "androidx.room:room-guava:$room_version"
```

```
    // optional - Test helpers
    testImplementation "androidx.room:room-testing:$room_version"
}
```

Додаток В. Код з бізнес-логікою додатку

SituationDao.java

```
@Database(entities = {EmergentSituation.class}, version = 1)
public abstract class EmergentSituationDatabase extends RoomDatabase {
    private static EmergentSituationDatabase INSTANCE;
    private static final int NUMBER_OF_THREADS = 4;

    public abstract SituationDao situationDao() throws SQLiteConstraintException;

    public static final ExecutorService databaseExecutor =
            Executors.newFixedThreadPool(4);

    public static EmergentSituationDatabase getDatabase(final Context context){
        if (INSTANCE == null) {
            synchronized (EmergentSituationDatabase.class) {
                if (INSTANCE == null) {
                    INSTANCE =
Room.databaseBuilder(context.getApplicationContext(),
                            EmergentSituationDatabase.class, "emerdgsit-database")
                        .fallbackToDestructiveMigration()
                        .build();
                }
            }
        }
        return INSTANCE;
    }
}
```

SituationDao.java

```
@Dao
public interface SituationDao {
    @Query("SELECT * FROM emergsit")
    Flowable<List<EmergentSituation>> getAllActions();

    @Query("SELECT * FROM emergsit WHERE title = :title")
    EmergentSituation getSituationByTitle(String title);

    @Query("SELECT * FROM emergsit")
    List<EmergentSituation> getSituationList();

    @Query("DELETE FROM emergsit")
    void deleteAllSituation();

    @Delete
    void deleteSituation(EmergentSituation situation);

    @Insert
    void insertSituation(EmergentSituation situation) throws
SQLiteConstraintException;

}
```

EmergentSituation.java

```
public class EmergentSituation {
    @NonNull
    @PrimaryKey
    private String title;
    private List<Point> points;

    @NonNull
    public String getTitle() {
        return title;
    }

    public void setTitle(@NonNull String title) {
        this.title = title;
    }

    public List<Point> getPoints() {
        return points;
    }

    public void setPoints(List<Point> points) {
        this.points = points;
    }

    public EmergentSituation() {
    }
}
```

Point.java

```
public class Point {
    private String name;
    private boolean isChecked;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public boolean isChecked() {
        return isChecked;
    }

    public void setChecked(boolean checked) {
        isChecked = checked;
    }

    public Point(String name, boolean isChecked) {
        this.name = name;
        this.isChecked = isChecked;
    }
}
```

MainFragment.java

```
public class MainFragment extends Fragment {

    private MainViewModel mViewModel;
    private MainFragmentBinding binding;



    public static MainFragment newInstance() {
        return new MainFragment();
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container,
                             @Nullable Bundle savedInstanceState) {

        binding = MainFragmentBinding.inflate(inflater, container, false);

        TextView noElectricityTextView = binding.textView2;
        TextView noWaterTextView = binding.textView3;
        TextView basicSituationsTextView = binding.textView4;

        noElectricityTextView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                savedInstanceState.putString("situationName", "Відключення
електроенергії");
                getActivity().getSupportFragmentManager().beginTransaction()
```

```
            .replace(R.id.mainFragment, new DetailInfoFragment())
            .addToBackStack(null)
            .commit();
    }
});

noWaterTextView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        savedInstanceState.putString("situationName", "Відключення води");
        getActivity().getSupportFragmentManager().beginTransaction()
            .replace(R.id.mainFragment, new DetailInfoFragment())
            .addToBackStack(null)
            .commit();
    }
});

basicSituationsTextView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        savedInstanceState.putString("situationName", "Базова ситуація");
        getActivity().getSupportFragmentManager().beginTransaction()
            .replace(R.id.mainFragment, new DetailInfoFragment())
            .addToBackStack(null)
            .commit();
    }
});

ProgressBar noElectricityPrBar = binding.progressBar;
ProgressBar noWaterPrBar = binding.progressBar5;
```

```
        ProgressBar basicSituationsPrBar = binding.progressBar6;

        EmergentSituationDatabase db = Room.databaseBuilder(getContext(),
                EmergentSituationDatabase.class, "emerdgsit-
database").allowMainThreadQueries().build();

        return inflater.inflate(R.layout.main_fragment, container, false);
    }

    @Override
    public void onActivityCreated(@Nullable Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        mViewModel = ViewModelProviders.of(this).get(MainViewModel.class);
        // TODO: Use the ViewModel
    }

}
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_activity);
        if (savedInstanceState == null) {
            getSupportFragmentManager().beginTransaction()
                    .replace(R.id.mainFragment, MainFragment.newInstance())
```

```
                .commitNow();
        }
    }
}
```