

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему РОЗРОБЛЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ
ПІДТВЕРДЖЕННЯ ЯКОСТІ ПРОЦЕСІВ І ПРОДУКТІВ МІЖНАРОДНИХ
НАВЧАЛЬНИХ ПРОЕКТІВ.
MOBILE APPLICATION DEVELOPMENT FOR
QUALITY ASSURANCE OF THE PROCESSES AND PRODUCTS IN
INTERNATIONAL EDUCATIONAL PROJECTS.

Виконав: студент(ка) 4 курсу, групи КНТ-127
Спеціальності 121 Інженерія програмного
забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Куковинець О.Є
(прізвище та ініціали)

Керівник Табунщик Г.В.
(прізвище та ініціали)

Рецензент Шитікова О.В.
(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 20 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА(КИ)

Куковинця Олега Євгеновича
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розроблення мобільного застосунку для підтвердження якості процесів і продуктів міжнародних навчальних проектів. Mobile application development for quality assurance of the processes and products in international educational projects.

керівник проекту (роботи) Табунщик Галина Володимирівна, к.т.н, професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проекту (роботи) 21 травня 2021 року

3. Вихідні дані до проекту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір та обґрунтування структури системи, яка проектується. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження програми. 5. Текст програми. 6. Технічне завдання.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Табунщик Г. В., професор		
Нормоконтроль	Камінська Ж. К., асистент		

7. Дата видачі завдання 5 березня 2021р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка архітектури програми.	4 тиждень	Розділ 2
4	Розробка програми.	5–6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми.	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи.	9 тиждень	

Студент (ка) _____ **Куковинець О.Є.**
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ **Табунщик Г.В.**
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломних кваліфікаційних робіт бакалавра:
90с., 41 рис., 3 таблиці, 3 додатки, 25 джерел.

МОБІЛЬНИЙ ЗАСТОСУНОК, ДОДАТОК, JSON, SQL, ANDROID,
СЕРВЕР, ОПИТУВАННЯ, JAVA, ANDROID STUDIO, ПРИСТРІЙ.

Об'єкт дослідження – програмні засоби, які призначені для проведення опитувань.

Предмет дослідження – застосунок призначений для проведення опитувань.

Мета роботи - розробка Android - застосунку із можливостями перегляду, пошуку та проходження опитувань, зберігання результатів та статистики кожного з них, з простим та приємним для користувача інтерфейсом.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мобільний пристрій під управлінням операційної системи Android версії 6 або вище, мови програмування Java та Python.

Результати. Розроблено програмне забезпечення, яке дозволяє користувачам працювати з опитуваннями, зберігати результати проходження за наявності підключення до мережі Інтернет.

Висновки. Створено Android - застосунок, який дозволяє отримати думку користувачів, давати оцінку будь-яким процесам методом проведення опитувань та збереженням відповідей на питання користувачами.

Галузь використання – проведення опитувань з метою дослідження думки людей, оцінки будь-яких процесів.

ABSTRACT

Explonatory note to the final qualifying work of the bachelor: 90 pages, 41 figures, 3 tables, 3 appendixes, 25 sources.

MOBILE APPLICATION, APP, JSON, SQL, ANDROID, SERVER, POLL, JAVA, ANDROID STUDIO, DEVICE.

The object of research is software that is designed to conduct surveys.

The subject of research - the application is designed to conduct surveys.

The purpose of the work is to develop an Android application with the ability to view, search and pass surveys, store results and statistics of each of them, with a simple and user-friendly interface.

Materials, methods and technical means: object-oriented programming, mobile device running Android operating system version 6 or higher, Java and Python programming languages.

Results. Software has been developed that allows users to work with surveys, save the results of the passage when connected to the Internet.

Conclusions. Android was created - an application that allows you to get the opinion of users, to evaluate any process by conducting surveys and saving answers to questions from users.

Field of use - conducting surveys to study people's opinions, evaluate any processes.

ЗМІСТ

Перелік умовних позначень та скорочень	7
Вступ.....	8
1 Аналіз предметної області	10
1.1 Знайомство із предметною областю.....	10
1.2 Огляд існуючих програмних засобів, що вирішують аналогічні завдання	10
1.3 Висновок за розділом 1	13
2 Вибір та обґрунтування структури системи, яка проектується	14
2.1 Вибір інструментарію розробки програмного забезпечення	14
2.2 Структурна схема програми	20
2.3 Опис файлів розроблюваної системи	21
2.4 Висновки за розділом 2.....	23
3 Основні рішення щодо реалізації компонентів системи.....	24
3.1 Головний екран(ActivityMain)	27
3.2 Вікна інформації та статистики опитувань	33
3.3 Форма проходження опитувань	36
3.4 Збереження даних та файлів	40
3.5 Висновки за розділом 3	44
4 Експлуатація, тестування та експериментальне дослідження застосунку.....	45
4.1 Експлуатація та процес роботи застосунку	45
4.2 Тестування застосунку	46
4.3 Висновки за розділом 4.....	47
Висновки.....	48
Перелік посилань	50
Додаток А Технічне завдання.....	52
Додаток Б Текст програми.....	56
Додаток В Слайди презентації	82

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД – База даних

ПЗ – Програмне забезпечення

ARM – Advanced RISC Machine

HTML – HyperText Markup Language

HTTP - HyperText Transfer Protocol

JSON – JavaScript Object Notation

MIPS – Million Instructions Per Second

PHP - Hypertext Preprocessor

SQL – Structured Query Language

SDK – Software Delevelopment Kid

ВСТУП

На сьогоднішній день проведення опитувань стало досить поширеним явищем. Опитування проводять всюди: на вулиці, по квартирах, при прийомі на роботу, в інтернеті і в безлічі інших ситуацій.

Опитування - це метод дослідження думки певних груп людей, метою якого є отримання інформації про що-небудь. Формати та методики опитування можуть бути різними і в залежності від ситуації можуть відрізнятися. Опитування характеризується чіткою кількістю питань з можливими варіантами відповідей.

Спочатку, проведення опитувань носило тільки науковий характер. Опитування проводилися в рамках наукових праць і досліджень. Такі опитування мають місце і зараз. Але сьогодні опитування також проводяться у сферах політики, медицини, спорту, освіти.

Проведення опитувань у сфері освіти є дуже важливим, так як дають зрозуміти чи достатньо ефективним є навчальний процес та дати йому оцінку. Система освіти повинна забезпечувати таку підготовку, яка дозволить успішно існувати в суспільстві, бути конкурентоспроможними на ринку праці, а також бути готовими до того, щоб приводити власні навички у відповідність до потреб, які диктує час.

Аналізуючи вище написане можна зробити висновок, що проведення опитувань у різних сферах життя є достатньо важливим, так як вони дають можливість дати оцінку будь-яким процесам, проаналізувати інтереси суспільства та зробити висновки.

Повного аналогу Android-застосунку, що розроблюється, не існує, так як застосунок матиме унікальний користувацький інтерфейс та дизайн, саме це доводить актуальність розробки.

Метою дипломного проекту є розробка Android - застосунку із можливостями перегляду, пошуку та проходження опитувань, зберігання результатів та статистики кожного з них, з простим та приємним для

користувача інтерфейсом. Програмне забезпечення призначено для підтвердження якості навчального процесу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Знайомство із предметною областю

Усе програмне забезпечення, метою якого є проведення опитувань, схоже між собою. Таке ПЗ дозволяє створювати, переглядати, шукати та проходити опитування користувачами, зберігати результати проходження та статистику кожного з них. Відмінність полягає у тому, що такі рішення можуть бути у вигляді мобільного чи веб-застосунку. Наявність сервера надає можливість користувачам з будь-якого місця та пристрою, за умови підключення до мережі Інтернет, відкрити застосунок та працювати з опитуваннями і саме тому програмне забезпечення, яке розроблюється, матиме клієнт – серверну архітектуру та вище наведений функціонал.

1.2 Огляд існуючих програмних засобів, що вирішують аналогічні завдання

Функціонал у існуючих аналогів однаковий але вони мають різний інтерфейс, дизайн та різне розташування керуючих елементів.

Застосунок, що розроблюється, має назву «Polls - WorldWide».

1.2.1 Застосунок «Pollscape»

Першим аналогом ПЗ, що розроблюється, є застосунок «Pollscape», інтерфейс наведено на рисунку 1.1. Застосунок працює добре, має посередній дизайн [1].

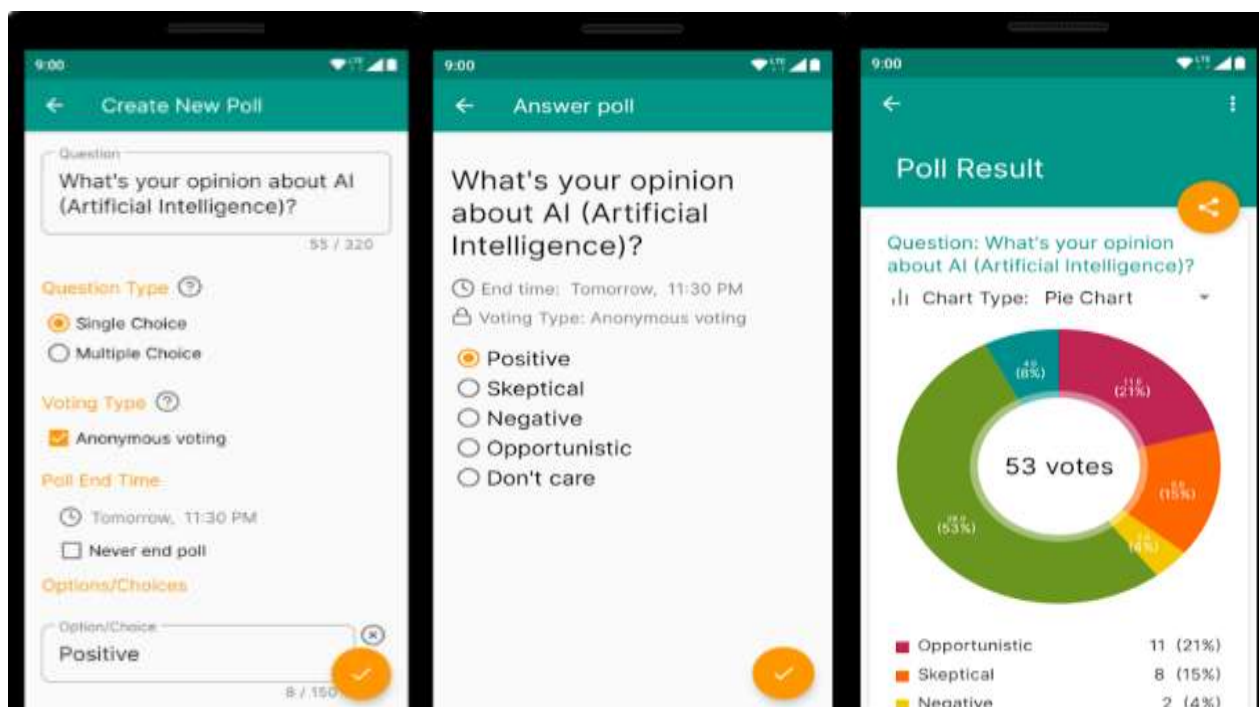


Рисунок 1.1 – Інтерфейс застосунку «Pollscape»

1.2.2 Застосунок «Poll For All»

На рисунку 1.2 наведено інтерфейс застосунку «Poll For All». Слабким місцем застосунку є стабільність роботи та інтерфейс [2].

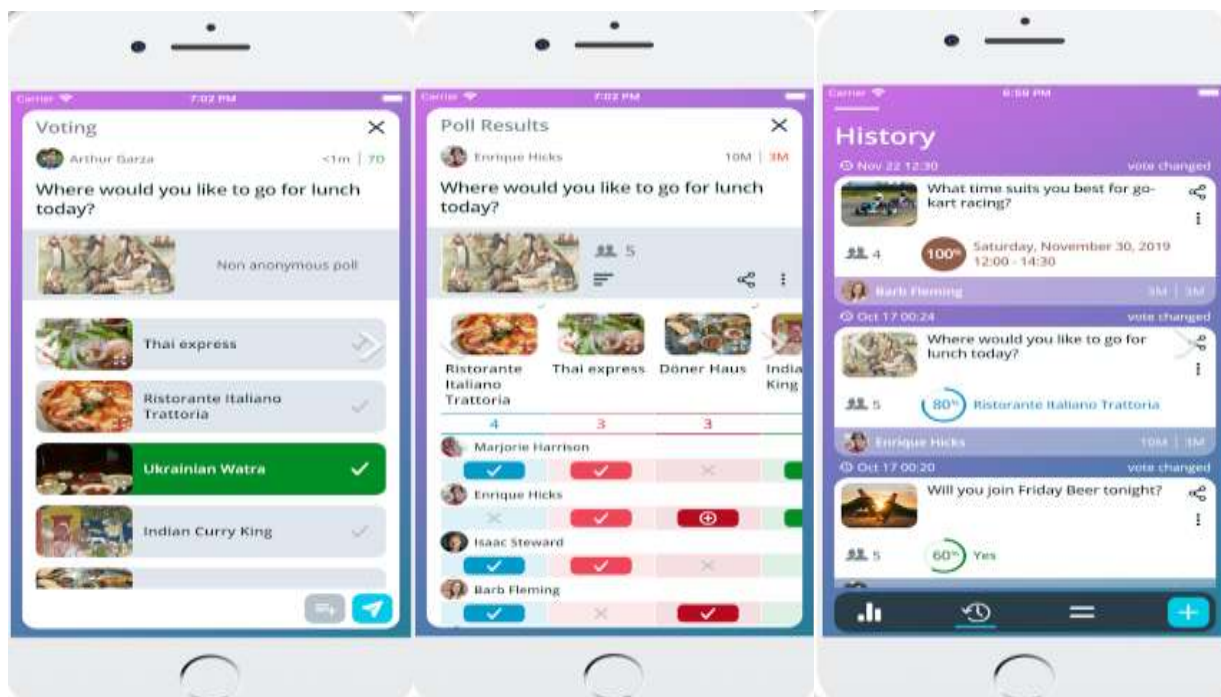


Рисунок 1.2 – Інтерфейс застосунку «Poll For All»

1.2.3 Застосунок «Pollie»

На рисунку 1.3 наведено інтерфейс застосунку «Pollie». Застосунок має відмінний дизайн але працює нестабільно [3].

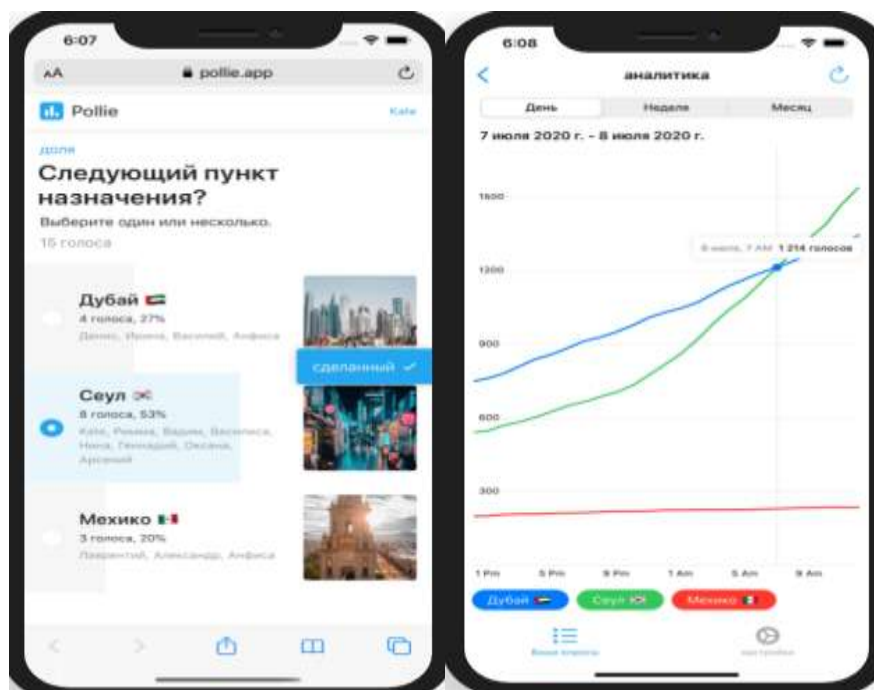


Рисунок 1.3 – Інтерфейс застосунку «Pollie»

1.2.4 Порівняльна таблиця існуючих та розроблюваного застосунку

Нижче в таблиці 1.1 наведено порівняльну характеристику застосунку, що розроблюється, з аналогами.

Таблиця 1.1 – Порівняльна характеристика

Назва застосунку	Інтерфейс	Зручність використання	Дизайн	Інтуїтивна зрозумілість	Стабільність роботи
Pollscape	+	+	-	+	+
Poll For All	-	+	+	-	-
Pollie	+	+	+	+	-
Polls - WorldWide	+	+	+	+	+

Оцінки встановлені згідно коментарям та оцінкам користувачів застосунків.

Застосунок «Pollscape» має непогану стабільність та інтерфейс, завдяки чому досягається зручність та інтуїтивна зрозумілість, але дизайн не є кращим.

Застосунок «Poll For All» має непоганий дизайн та зручність використання, але через інтерфейс не є інтуїтивно зрозумілим, а стабільність не є кращою.

Застосунок «Pollie» відрізняється тільки нестабільністю роботи, але інтерфейс та дизайн привабливі.

1.3 Висновок за розділом 1

У розділі 1 було ознайомлено з предметною областю розробки та цільовою аудиторією користувачів.

Було проведено аналіз трьох програмних засобів, що виконують аналогічні функції. На основі коментарів та оцінкам користувачів, було встановлено умовні оцінки. Результати було наведено у таблиці для порівняння. Жодне ПЗ, що виконує аналогічні функції, не задовольняє в повній мірі потреби користувачів, отже розробка власного ПЗ є доцільною.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЕКТУЄТЬСЯ

2.1 Вибір інструментарію розробки програмного забезпечення

Android – операційна система і платформа для мобільних телефонів та планшетних комп'ютерів, створена компанією Google на базі ядра Linux. Підтримується альянсом Open Handset Alliance (ОНА). Хоча Android базується на ядрі Linux, він стоїть дещо осторонь Linux-спільноти та Linux-інфраструктури. Базовим елементом цієї операційної системи є реалізація Dalvik віртуальної машини Java, і все програмне забезпечення і застосування спираються на цю реалізацію Java [4].

Перша версія Android була випущена 23 вересня 2008 року і носила назву 1.0 Astroboy, а наступна — 1.1 Bender. Від назв у честь відомих роботів згодом довелося відмовитися через розбіжності з правовласниками [5].

З 2008 року Android пережив численні оновлення, які поступово покращували операційну систему, додаючи нові функції, та виправляли помилки у попередніх випусках. І тепер кодове ім'я кожного великого релізу Android, починаючи з версії 1.5, являє собою назву якого-небудь десерту [6].

Перші букви найменувань в порядку версій відповідають літерами латинського алфавіту:

- 1.5 Cupcake («кекс»);
- 1.6 Donut («пампушка»);
- 2.0/2.1 Eclair («еклер» або «глазур»);
- 2.2 Froyo (скорочення від «заморожений йогурт»);
- 2.3 Gingerbread («імбирний пряник»);
- 3.0 Honeycomb («медові стільники»);
- 4.0 Ice Cream Sandwich («брикет морозива»);
- 4.1/4.2/4.3 Jelly Bean («желейні боби»);
- 4.4 KitKat (на честь однойменного бренду шоколадних батончиків, раніше планувалася назва Key Lime Pie);

- 5.0/5.1 Lollipop («льодяник»);
- 6.0/6.1 Marshmallow («зефір»);
- 7.0 N - Nougat («нуга»);
- 8.0 - Oreo («Орео»);
- 9.0 – Pie («пиріг»);
- 10 Q – Q (назва не розшифровується);
- 11 – R [7].

Деякі користувачі відзначають, що Android проявляє себе краще одного зі своїх конкурентів, Apple iOS, в ряді особливостей, таких як веб-серфінг, інтеграція з сервісами Google і інших. Також Android, на відміну від iOS, є відкритою платформою, що дозволяє реалізувати на ній більше функцій [8].

Незважаючи на початкову заборону на установку програм з «неперевіраних джерел» (наприклад, з карти пам'яті), це обмеження відключається штатними засобами в налаштуваннях пристрою, що дозволяє встановлювати програми на телефони та планшети без інтернет-підключення (наприклад, користувачам, які не мають Wi-Fi-точки доступу і не бажають витратити гроші на мобільний інтернет, який зазвичай коштує дорого), а також дозволяє будь-кому безкоштовно писати програми для Android і тестувати на своєму пристрої [9].

Android доступний для різних апаратних платформ, таких як ARM, MIPS, x86. Існують альтернативні Google Play магазини додатків: Amazon Appstore, Opera Mobile Store, GetUpps!, F-Droid. У версії 4.3 з'явилась підтримка багатокористувацького режиму [10].

У 84 % смартфонів, проданих у 3-ому кварталі 2014 року, була встановлена операційна система Android.

У березні 2017 року ОС Android стала найпопулярнішою ОС, з якої виходили в інтернет. Так з 37,93% користувачів заходили в інтернет із Android'a, а з Windows лише 37,91% користувачів. В Азії показники ще вищі – 52,2% і 29,2% відповідно [11].

Враховуючи вище наведені числа та переваги операційної системи

Android над іншими системами саме її обрано для розробки застосунку.

2.1.1 Огляд особливостей середовищ розробки

2.1.1.1 Огляд особливостей середовища розробки Android Studio

Android Studio — інтегроване середовище розробки (IDE) для платформи Android, представлене 16 травня 2013 року на конференції Google I/O менеджером по продукції корпорації Google — Еллі Паверс. 8 грудня 2014 року компанія Google випустила перший стабільний реліз Android Studio 1.0 [12].

Деякі особливості будуть пізніше розгорнуті для користувачів так як програмне забезпечення розвивається; наразі, передбачені такі функції:

- живі макети (layout): редагувальник WYSIWYG — живе кодування — подання (rendering) програми в реальному часі.
- консоль розробника: підказки по оптимізації, допомога по перекладу, стеження за напрямком, агітації та акції — метрики Google аналітики.
- резерви бета релізів та покрокові релізи.
- базування на Gradle.
- android-орієнтований рефакторинг та швидкі виправлення.
- lint утиліти для охоплення продуктивності, юзабіліті, сумісності версій та інших проблем.
- використання можливостей ProGuard та підписів до програм.
- шаблони для створення поширених Android дизайнів та компонентів.
- багатий редактор макетів (layouts) що дозволяє користувачам перетягнути і покласти (drag-and-drop) компоненти користувацького інтерфейсу, як варіант, переглянути одночасно макети (layouts) на різних конфігураціях екранів [13].

2.1.1.2 Огляд особливостей середовища розробки IntelliJ IDEA

IntelliJ IDEA — комерційне інтегроване середовище розробки для різних мов програмування (Java, Python, Scala, PHP та ін.) від компанії JetBrains. Система поставляється у вигляді урізаної по функціональності безкоштовної версії «Community Edition» і повнофункціональної комерційної версії «Ultimate Edition», для якої активні розробники відкритих проектів мають можливість отримати безкоштовну ліцензію. Сирцеві тексти Community-версії поширюються в рамках ліцензії Apache 2.0. Бінарні збірки підготовлені для Linux, Mac OS X і Windows [14].

Версія «Community Edition» середовища IntelliJ підтримує:

- інструменти (у вигляді плагінів) для проведення тестування TestNG і JUnit;
- системи контролю версій CVS, Subversion, Mercurial і Git;
- засоби складання Maven, Ant, Gradle;
- мови програмування Java, Scala, Clojure, Groovy і Dart;
- розробку застосунків для мобільної платформи Android.

Комерційна версія «Ultimate Edition» відрізняється наявністю підтримки додаткових мов програмування, підтримкою технологій Java EE, UML-діаграм, підрахунок покриття коду, можливістю роботи з фреймворками, засобами інтеграції з Perforce, Microsoft Team Foundation Server і Rational ClearCase [15].

2.1.1.3 Огляд особливостей середовища розробки Eclipse IDE

Eclipse - це безкоштовна IDE, розроблена некомерційною компанією Eclipse Foundation. Ця програма є базою, яка регулює процеси створення додатків. Переваги Eclipse:

- простий, зрозумілий та інтуїтивний інтерфейс;
- відмінно працює на комп'ютерах з низькою продуктивністю;
- має додаткові функції (для серверної роботи і аналізу бази даних);

- може підключатися до модулів;
- може працювати в груповому режимі (коли проект створюють кілька людей одночасно) [16].

Eclipse стала популярною кілька років тому і раніше займала лідируючі позиції. Хоча після виходу Android Studio Google вирішила перевести співпрацю з Eclipse на «другий план» [17].

2.1.1.4 Огляд особливостей середовища розробки Intel XDK

Intel XDK - програмне забезпечення корпорації Intel, яке забезпечує роботу над усім життєвим циклом розробки кросбраузерних мобільних додатків з використанням вебтехнологій (зокрема HTML5 і Cordova). Основною перевагою інструментарію Intel XDK є можливість розробки на CSS і JavaScript, після чого компілювати проект в файли для IOS, Android і Windows Phone - .ipa, .apk і .appx відповідно. Після чого дані файли можна завантажувати в магазини додатків [18].

2.1.1.5 Порівняльна таблиця середовищ розробки

Таблиця 2.1 – Порівняння IDE для мобільної розробки

Середовище розробки	Мова програмування	Мобільні платформи для розробки	Платня
Android Studio	Java, C/C++, Delphi, Kotlin	Android	ні
IntelliJ IDEA	Java, C/C++, PHP, Ruby, Python, SQL, Rust, Go, XML	Android, Windows phone	так
Eclipse IDE	Java, C/C++, PHP	Android, IOS	ні
Intel XDK	HTML	всі	так

Для розробки застосунку обрано середу розробки Android Studio, так як у неї найзручніший користувацький інтерфейс, багато доступної документації, підтримка відомих мов програмування, достатньо велика оцінка користувачів, безкоштовність користування і постійні оновлення, які додають середі розробці ще більше корисного функціоналу.

2.1.2 Вибір мови програмування

В даний час Java є найпопулярнішою мовою програмування для мобільної розробки на Android. Java є офіційною мовою для розробки Android - застосунків і підтримується Android Studio. Для будь-якого розробника мобільних додатків на ОС Android, першою і найкращою мовою програмування поки залишається Java, так як вона підтримується компанією Google і більшість додатків в Google Play побудовані саме на ній. Саме тому для розробки застосунку була обрана мова програмування Java [19].

Нижче розглянуті інші мови програмування на яких пишуться Android – застосунки.

Kotlin є офіційною і останньою введеною мовою Android, вона схожа на Java, але в багатьох відносинах трохи легше.

C # кілька зручніше для початківців на відміну від C або C ++. Він підтримується деякими дуже зручними інструментами, наприклад Unity і Xamarin, які відмінно підходять для розробки ігор і крос-платформних додатків.

Basic неймовірно приємний у використанні і є ідеальним початком для навчання кодування [20].

Corona - ще один багатоплатформовий інструмент, який використовує мову Lua (скриптова мова програмування, за ідеологією і реалізацією найближче до JavaScript) для реалізації логіки додатків, він значно спрощує процес збірки додатків і дозволяє викликати власні бібліотеки.

PhoneGap дозволяє створювати додатки, що використовують код, який зазвичай використовується для створення вебсайту: HTML, CSS [21].

2.1.3 PythonAnywhere

PythonAnywhere - назва хмарної платформи, призначеної переважно для запуску додатків Python. Включає запуск і редагування web-додатків на основі поширених вебфреймворків Python, базу даних MySQL і PostgreSQL, онлайн консоль Bash, веб-редактор коду. Сервер веб-додатків побудований на основі Nginx + uWSGI. У свою чергу PythonAnywhere розміщується на потужностях Amazon EC2 [22].

2.2 Структурна схема програми

Кожний застосунок має структуру виконуючих файлів та модулів, яка дозволяє йому працювати коректно.

Файл маніфесту містить в собі загальну інформацію про застосунок. Java – файли містяться в директорії java та містять в собі логіку застосунку. Директорія res містить в собі форми користувацького інтерфейсу, графічні налаштування застосунку, зображення та інше. Файли Gradle Scripts відповідають за автоматичну збірку проекту. На рисунку 2.1 зображено структуру файлів та модулів проекту.

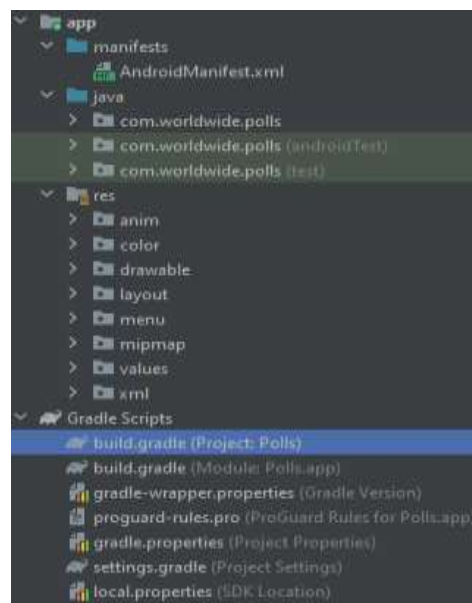


Рисунок 2.1 – Структура файлів та модулів проекту

2.3 Опис файлів розроблюваної системи

До набору java - файлів входять:

- Poll – клас, що містить інформацію про опитування, список питань, методи роботи із питаннями, додає дані до JSON файлу;
- BasePool – клас для отримання даних про опитування з JSON файлу та зберігає їх;
- Question – клас, що містить інформацію про питання, список відповідей, методи роботи із відповідями, додає дані до JSON файлу;
- BaseQuestion – клас для отримання даних про питання з JSON файлу та зберігає їх;
- Answer – клас, що містить інформацію про відповідь та методи, які відповідають за вибір відповіді і додає дані до JSON файлу;
- BaseAnswer – клас для отримання даних про відповідь з JSON файлу та зберігає їх;
- PollFilter – клас для сортування опитувань;
- FilterConfigs – клас конфігурації фільтра;
- SortBy – клас для вибіру параметру для сортування опитувань;
- SearchIn – клас для пошуку опитувань за назвою;
- Provider – клас для отримання посилання з одного класу на інший (наприклад з об'єкта Answer перейти до об'єкта Question);
- ContentManager – клас для зберігання обраних, завершених та звичайних опитувань;
- Timer – клас для відслідковування фактичного часу проходження опитування;
- DBManager – клас для роботи з базою даних, містить методи для створення, запису та зчитування даних;
- RequestPacker – клас для керування методами роботи з сервером;
- PollAdapter та PollHolder – класи, які відображають опитування та їх

дані;

- QuestionAdapter та QuestionHolder – класи, які відображають статистику питань.

До набору xml – файлів входять:

- activity_main – головне вікно застосунку;
- activity_poll – форма проходження опитування;
- activity_poll_info – форма інформації про опитування;
- activity_statistics – форма статистики за відповідями кожного питання опитування;

- answer_radio_button – кнопка відповіді на питання;
- fragment_meeting – форма відсутності обраних опитувань;
- fragment_poll_info – вікно інформації про опитування;
- poll_item – вікно для початку проходження опитування;
- poll_item_bubbles_horizontal – відображення інформації про опитування горизонтально;

- poll_item_bubbles_vertical – відображення інформації про опитування вертикально;

- search_filtres – форма сортування опитувань;
- statistics_answer_item – вікно для відображення відсотків обраної відповіді;

- statistics_question_item – вікно статистики за відповідями кожного питання опитування.

У директорії res (структуру наведено на рисунку 2.2) розміщені ресурси проекту:

- anim містить налаштування анімацій;
- color зберігає кольори застосунку;
- drawable містить зображення;
- layout зберігає файли користувацьких форм та вікон;
- menu містить вікна для меню застосунку;

- міртар містить стандартні іконки застосунку;
- values містить файли з кольорами, рядками та стилями;
- xml містить файл конфігурації мережевої безпеки.



Рисунок 2.2 – Структура файлів директорії res

2.4 Висновки за розділом 2

У розділі 2 було проведено аналіз можливостей операційної системи Android. Також, було проведено порівняння особливостей середовищ розробки та мов програмування, обрано Android Studio та мову програмування Java. Було спроектовано структуру застосунку, наведено детальний опис компонентів системи.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

Для коректної роботи програми потрібно підключення до інтернету. При першому запуску застосунку список опитувань порожній. Після перевірки на наявність підключення до мережі Інтернет, застосунок почне відправляти запити до серверу на отримання доступних опитувань та його статистики. Для обробки відповідей серверу було реалізовано інтерфейс `RequestInterface` (інтерфейс наведено на рисунку 3.1) та класи `RequestManager`, `RequestPacker`, `RequestParams`.

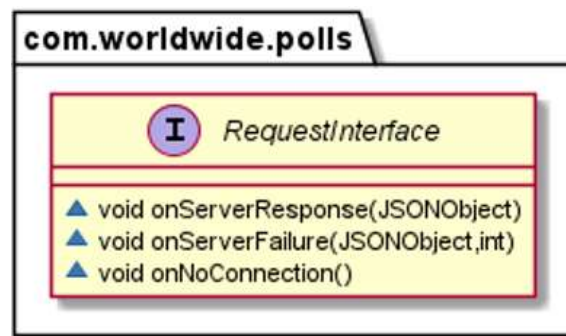


Рисунок 3.1 – Діаграма інтерфейсу `RequestInterface`

Цей інтерфейс має три методи, кожний з яких викликається у певному випадку.

- `onServerResponse` викликається, коли сервер обробив запит користувача без помилок, повертаючи дані у json вигляді;
- `onServerFailure` викликається тоді, коли відправлені на сервер дані не коректні або обробка завершилась із помилкою. Параметр `jsonResponse` має відповідний текст помилки або її код;
- метод `onNoConnection` викликається тоді, коли користувач немає доступу до інтернету, або серверна частина не відповідає на запити взагалі.

На рисунку 3.2 зображено повідомлення про помилку підключення до сервера.

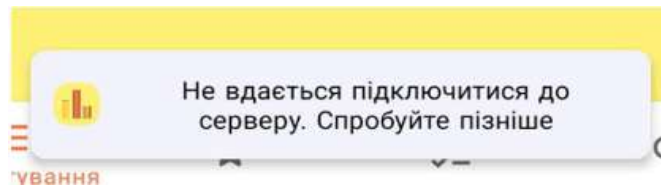


Рисунок 3.2 - Повідомлення про помилку підключення до сервера

На рисунку 3.3 зображено повідомлення про успішне оновлення опитувань.

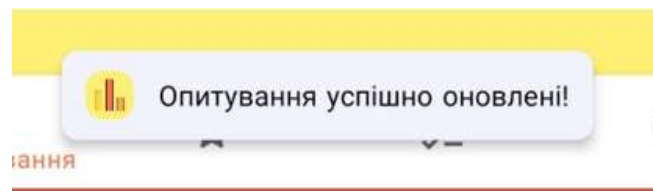


Рисунок 3.3 – Повідомлення про успішне оновлення опитувань

Клас `RequestParams` є класом-тримачем даних, що потрібні для запиту до серверу. Він містить в собі посилання, http метод (Get/Post) та дані, збережені у вигляді json. Об'єкт цього класу генерується у класі `RequestPacker` при визові статичних методів. Він компонує усі дані, генерує посилання та повертає `RequestParams` об'єкт. На рисунку 3.4 зображено класи `RequestParams` та `RequestPacker`.

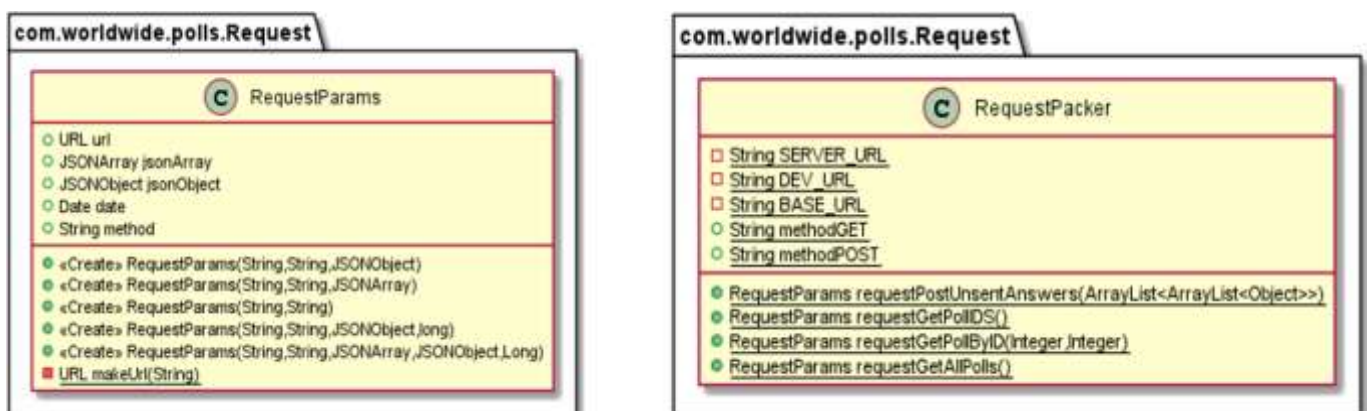


Рисунок 3.4 – Класи `RequestParams` та `RequestPacker`

Головний клас, що робить запити та обробляє відповіді –

RequestManager, що наслідує клас AsyncTask<URL, Void, String>. Він отримує посилання типу URL та повертає відповідь з серверу у вигляді String. Так, як запити до серверу займають деякий час, рекомендовано виконувати їх у фоновому процесі.

Виконання запиту виконується тільки після визову методу makeRequest(). Також до класу додано обробники помилок та інтерфейси типу Callback startAction та endAction. Це дозволяє виконувати деякі процеси до початку запиту та після завершення. Діаграму класів RequestManager та AsyncTask наведено на рисунку 3.5.

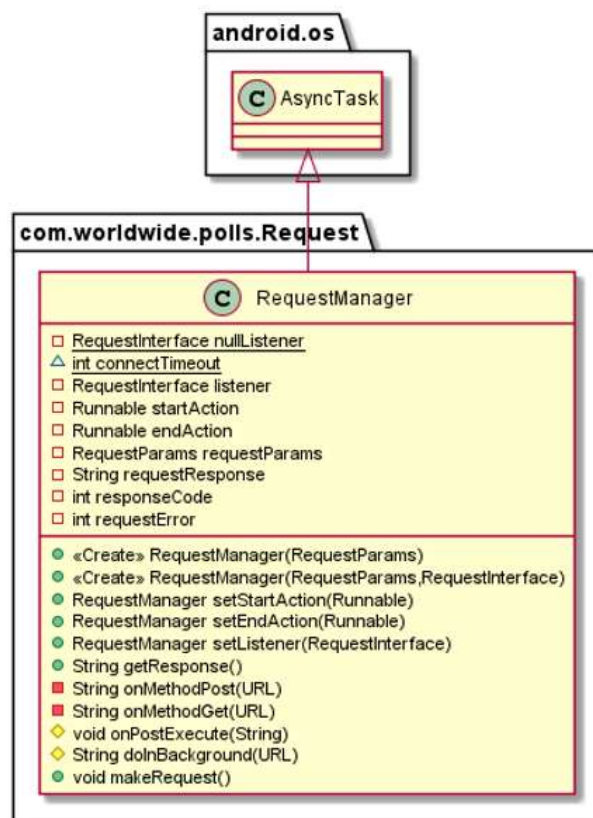


Рисунок 3.5 - Діаграма класів RequestManager та AsyncTask

Оновлення опитувань виконується тільки при наявності інтернет підключення та має свої особистості. Для зв'язку перш за все отримується список ідентифікаторів усіх опитувань, а далі у циклі виконуються запити на отримання інформації. При цьому, якщо опитування отримується вперше (немає збереженого файлу json із таким ідентифікатором у додатку), то статистика

переглядів опитування збільшується. Приклад реалізації:

```
public static RequestParams requestGetPollByID(Integer pollID, Integer
isNewViewer) {
    Uri.Builder b = Uri.parse(BASE_URL+"getPollByID").buildUpon();
    b.appendQueryParameter("pollID", String.valueOf(pollID));
    b.appendQueryParameter("isNewViewer", String.valueOf(isNewViewer));
    b.build();
    return new RequestParams(b.toString(), methodGET);
}

public void updateEveryPoll(ArrayList<Integer> jsonPollIDS){
    for (int i=0; i < jsonPollIDS.size(); i++) {
        int pollID = jsonPollIDS.get(i);
        boolean pathToPollExists = getPathToPoll(pollID).exists();
        RequestManager request = new
RequestManager(RequestPacker.requestGetPollByID(pollID,
boolToInt(!pathToPollExists)));
        request.setListener(updateEveryPollInterface);
        if (i == jsonPollIDS.size()-1)
            request.setEndAction(() -> {
                pollsAreUpdating = false;
                showToast(getString(R.string.polls_successfully_updated));
            });
        request.makeRequest();
    }
}
```

3.1 Головний екран(ActivityMain)

Головний екран застосунку складається із верхньої навігаційної панелі, контейнеру RecyclerView з опитуваннями, що займає більшу частину екрану, та нижнє меню для швидкого пересування між опитуваннями. На рисунку 3.6 зображено загальний вигляд головного екрану.

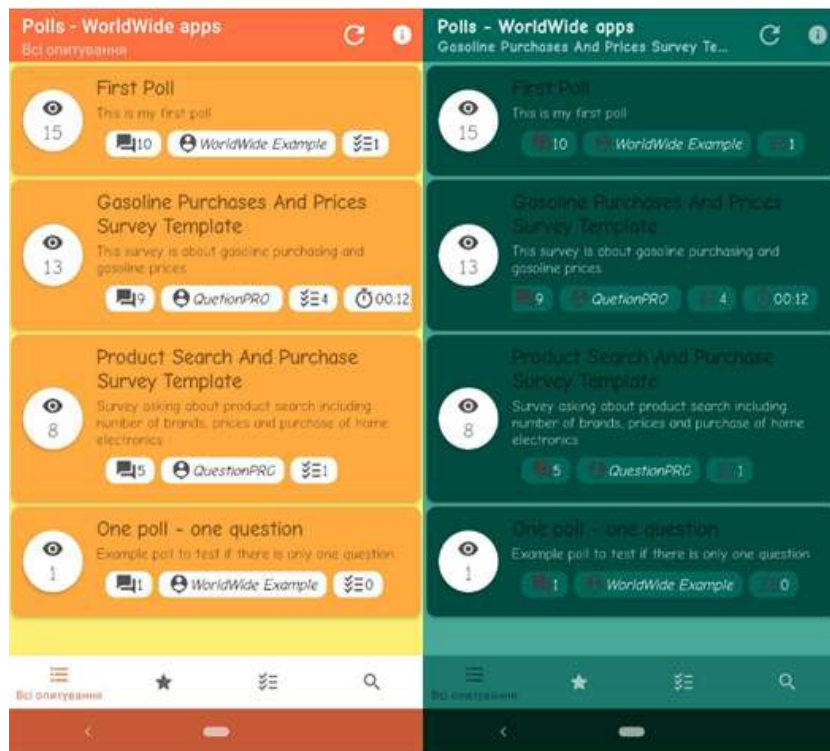


Рисунок 3.6 – Загальний вигляд головного екрану

3.1.1 Верхня панель керування

Верхня панель має усі необхідні елементи керування, та динамічно змінюються за потребами користувача. За замовченням панель має 2 іконки: оновлення опитувань та інформація щодо застосунку.

Додаткові іконки, що з'являються під час роботи із застосунком: іконка «Додати до/видалити із списку обраного» у формі зірки. При натисканні змінюється значення `Poll:: isFavourite` та викликається інтерфейс-`listenerOnActionListener`, що інформує о змінах в об'єкті та оновлює значення у базі даних:

```
class Poll {
    public void setFavourite(boolean newFavourite) {
        if (isFavourite != newFavourite) {
            isFavourite = newFavourite;
            listener.onAction(OnActionListener.Actions.POLL_CHANGE_FAVOURITE,
this, useListener);
        }
    }
}

Poll.setListener(new OnActionListener<Poll>() {
    @Override
    public void onAction(Actions action, Poll poll) {
```

```

        switch (action) {
            case POLL_CHANGE_FAVOURITE:
                dbManager.updatePollFavourite(poll.intID, poll.isFavourite());

                ActivityMain.this.showToast(ActivityMain.this.getString(poll.isFavourite() ?
                    R.string.action_add_favourite :
                    R.string.action_removed_favourite,
                    poll.name.substring(0, 10)));
                recyclerAdapter.notifyItemChangedByID(poll.intID);
                break;
        }
    });
}

```

Іконки «Фільтрація» та «Пошук» з'являються тільки під час роботи в пошуковому вікні. Іконки «Фільтрація» та «Пошук» зображені на рисунку 3.7.

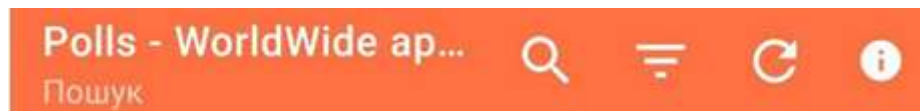


Рисунок 3.7 - Іконки «Фільтрація» та «Пошук»

При натисканні «Оновлення» буде відбуватись повне оновлення усіх опитувань: отримання та запис структурованих даних у файл з серверу та оновлення елементів контейнеру за допомогою методів `RecyclerView.Adapter.notifyDataSetChanged()` та `notifyItemChanged()`.

При натисканні іконки «Інфо» буде відображено діалогове вікно `AlertDialog` із поточною інформацією про застосунок та кнопки управління: зміна мови, зміна теми та зв'язок з розробником методом відправлення електронного листа на пошту.

Зміна теми відбувається за допомогою методу `setDefaultNightMode` класу `AppCompatActivity`:

```

int newMode;
if (AppCompatActivity.getDefaultNightMode() == AppCompatActivity.MODE_NIGHT_YES)
{
    newMode = AppCompatActivity.MODE_NIGHT_NO;
}else {
    newMode = AppCompatActivity.MODE_NIGHT_YES;
}
AppCompatActivity.setDefaultNightMode(newMode);

```

Зміна мови відбувається перед запуском додатку у методі `attachBaseContext` путем редагування конфігурацій контексту:

```
newBase.getResources().getConfiguration().setLocale(new Locale(language))
super.attachBaseContext(newBase);
```

Запуск вікна зв'язку (рисунок 3.8) виконується методом запуску нового `Intent`:

```
Intent sendEmail = new Intent(Intent.ACTION_SENDTO);
sendEmail.setData(Uri.parse("mailto:"));
sendEmail.putExtra(Intent.EXTRA_EMAIL, new String[]{CONSTS.APP_EMAIL});
sendEmail.putExtra(Intent.EXTRA_SUBJECT, CONSTS.APP_EMAIL_SUBJECT);
sendEmail.putExtra(Intent.EXTRA_TEXT, getString(R.string.email_extra_text));
startActivity(Intent.createChooser(sendEmail, getString(R.string.send_email)));
```

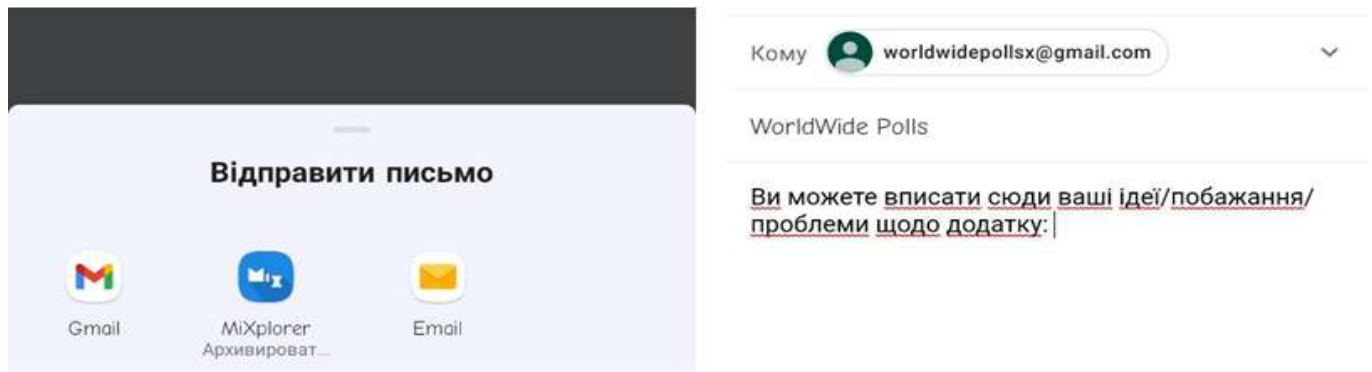


Рисунок 3.8 – Вікно зв'язку

При натисканні «Фільтрація» користувачу буде відображено впливаюче вікно із вибором таких параметрів:

- місця для пошуку (назва опитування, опис, автор, текст питань, текст відповідей);
- сортування(ідентифікатор, назва опитування, автор, кількість питань переглядів, проходжень);
- чи потрібно перегорнути список результатів;
- чи чутливий пошуковий текст до регістру;

На рисунку 3.9 зображено вікно фільтрації опитувань.

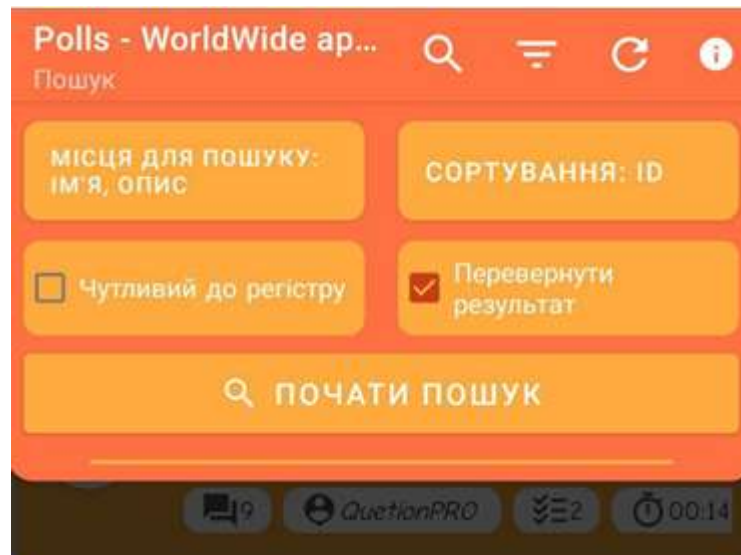


Рисунок 3.9 – Вікно фільтрації опитувань

3.1.2 Меню застосунку

Меню застосунку складається із чотирьох кнопок та реалізує його вид класс `com.google.android.material.bottomnavigation.BottomNavigationView`:

- всі опитування;
- обране;
- пройдені опитування;
- пошук.

На рисунку 3.10 зображено меню додатку.

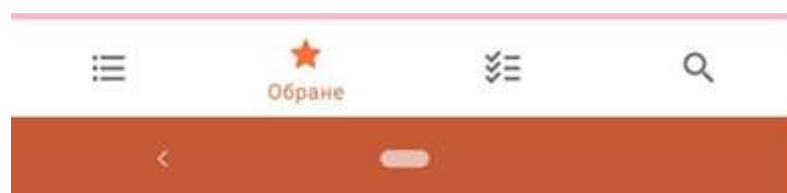


Рисунок 3.10 – Меню додатку

При натисканні на елемент меню у головний контейнер опитувань завантажуються відповідні дані та змінюється кольор бекграунду.

Якщо масив даних порожній, то на головний екран повертається відповідний попереджувальний текст (рисунок 3.11).

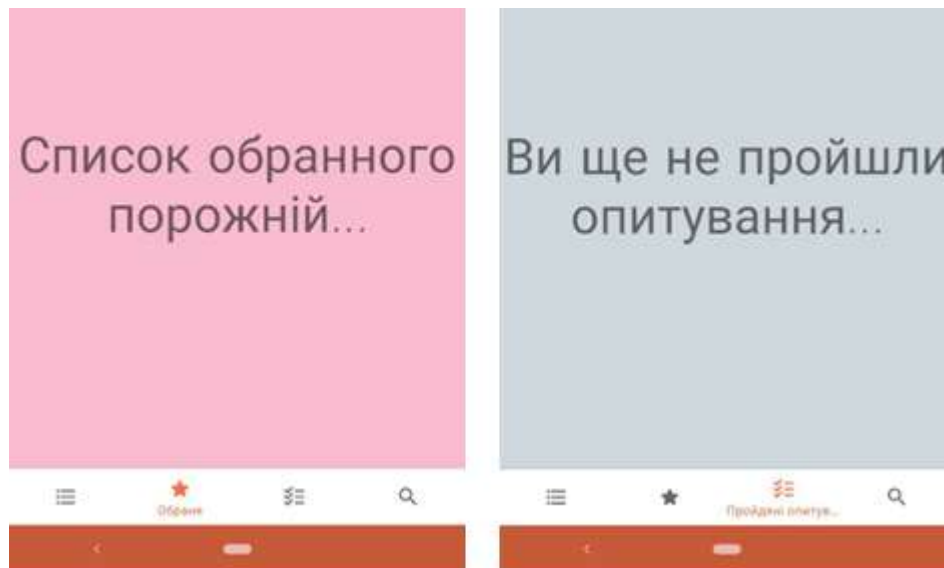


Рисунок 3.11 – Попереджувальний текст

3.1.3 Головний контейнер опитувань

Головний список реалізується за допомогою класів `androidx.recyclerview.widget.RecyclerView`, «тримачем» даних – адаптером `PollAdapter`, що наслідує `RecyclerView.Adapter<PollAdapter.PollHolder>`, та класом `PollAdapter.PollHolder` (наслідує `RecyclerView.ViewHolder`), що описує всю поведінку та вид елемента списка. На рисунку 3.12 зображено елемент списку, що описує об'єкт `Poll` (опитування).



Рисунок 3.12 - Елемент списку, що описує об'єкт Poll

Елемент складається з назви опитування, опису, іконки обраного у

правому верхньому кутку та горизонтального списку, що описують деяку статистику: автор, кількість питань, кількість переглядів, час проходження.

При натисканні на опитування відкривається форма інформації із подальшим вибором: перегляд статистики чи проходженні опитування.

3.2 Вікна інформації та статистики опитувань

Відображення форм «Інформація» та «Статистика» було реалізовано за допомогою фрагментів, як елементів багаторазового використання.

3.2.1 Фрагмент «Інформація» опитування (FragmentPollInfo)

Відкриття фрагменту здійснюється після кліку на один із поданих опитувань на головному екрані. Процес відкриття фрагменту зображено на рисунку 3.13.

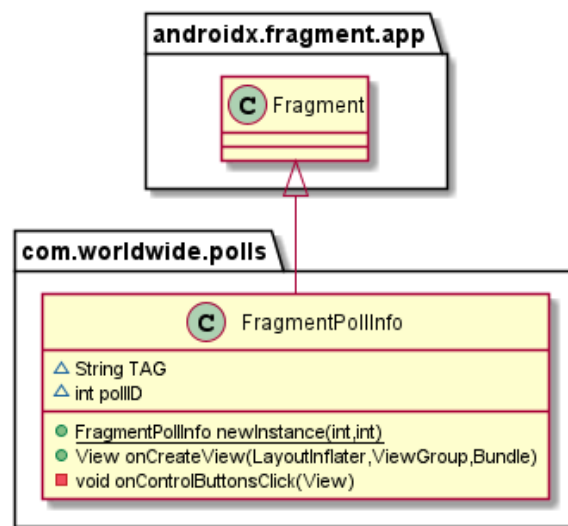


Рисунок 3.13 – Процес відкриття фрагменту

Обробка кліку на опитування:

```
CallbackUniqueID startPollInfoFragment = pollID -> {
    if (pollInfoIsShowing) return;
    togglePollInfoShowHide(true);
    Poll poll = ContentManager.getPollByUniqueID(pollID);
    if (poll == null) return;
```

```

updateFavouriteIcon(poll.isFavourite());
getSupportActionBar().setDisplayHomeAsUpEnabled(true);

menuFavourite.setVisible(true);

menuFavourite.setOnMenuItemClickListener(generateFavouriteItemClickListener(poll
ID));
    getSupportFragmentManager()
        .beginTransaction()
        .setCustomAnimations(R.anim.fragment_open_enter,
R.anim.fragment_open_exit, R.anim.fragment_fade_enter,
R.anim.fragment_fade_exit)
        .add(R.id.fragment, FragmentPollInfo.newInstance(pollID,
getColor(currentPairColorString.first()), "PollInfo")
        .addToBackStack("PollInfo")
        .commit();
};

```

У метод `FragmentPollInfo.newInstance` передаються параметри ідентифікатору опитування та кольор бекграунду, що буде дублюватися із тим, що встановлений на головному екрані. На рисунку 3.14 зображено відкритий фрагмент «Інформація» опитування.

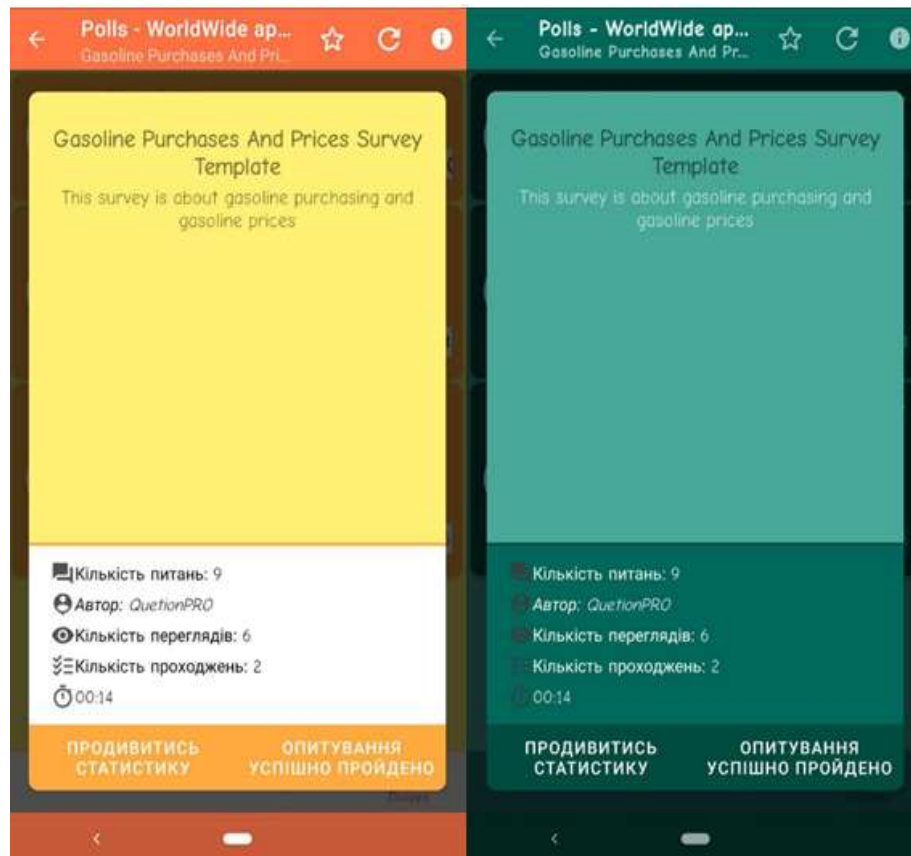


Рисунок 3.14 – Фрагмент «Інформація» опитування

Структура фрагменту виглядає наступним чином:

- назва опитування;
- опис опитування;
- відокремлена область із поточною статистикою опитування та час проходження, якщо опитування вже завершено користувачем;
- дві кнопки на вибір для подальшого переходу. Продивитись статистику та пройти опитування по натискання яких будуть відкриті відповідні вікна – `FragmentPollStatistics` та `ActivityPoll`.

Відкриття нових вікон здійснюється за допомогою наступного методу:

```
private void onControlButtonsClick(View view) {
    if (view.getId() == R.id.pollTakePoll) {
        Intent newIntent = new Intent(getActivity(), ActivityPoll.class);
        newIntent.putExtra(CONSTS.pollID, pollID);
        startActivity(newIntent);
    } else if (view.getId() == R.id.pollTakeStatistics) {
        getActivity().getSupportFragmentManager()
            .beginTransaction()
            .setCustomAnimations(R.anim.fragment_open_enter,
                R.anim.fragment_open_exit, R.anim.fragment_fade_enter,
                R.anim.fragment_fade_exit)
    }
}
```

```

        .add(R.id.fragment, FragmentPollStatistics.newInstance(pollID),
"PollStatistics")
        .addToBackStack("PollStatistics")
        .commit();
    }else{ return; }
}

```

3.2.2 Фрагмент «Статистика» опитування (FragmentPollStatistics).

Фрагмент статистики (рисунок 3.15) складається із списку питань використовуючи `androidx.recyclerview.widget.RecyclerView` та списком відповідей до кожного питання. Іконка галки показує збережений варіант відповіді користувача додатку. Справа показана статистика відповіді: кількість обрання відповіді та відсоток.

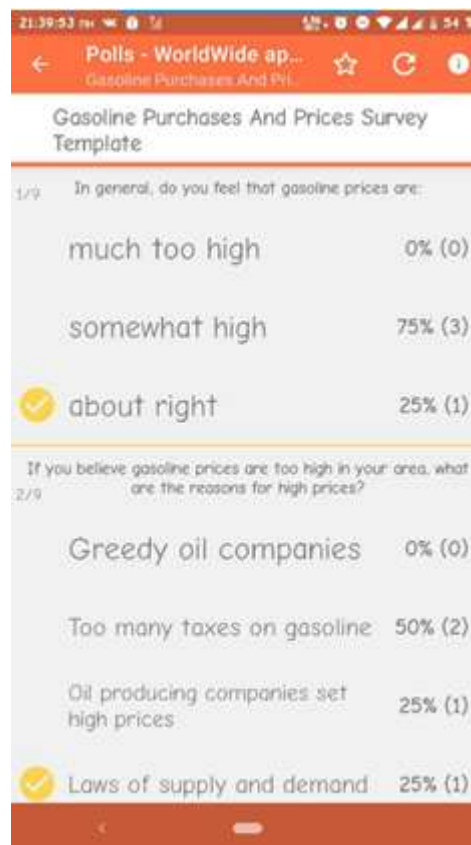


Рисунок 3.15 – Фрагмент статистики

3.3 Форма проходження опитувань

Процес проходження опитувань був винесений до окремого активіті.

Форма проста та приємна на вид та анімацію, що не менш важливо при проходженні опитувань користувачам. Розмір шрифту тексту запитання автоматично змінюється за допомогою вбудованих методів у класі TextView.

3.3.1 Структура форми

Форма складається із тексту запитання, елементів управління (показати попереднє питання або продовжити проходження), із повзунку, що показує поточний прогрес проходження, автоматично змінючи свій розмір залежно від кількості питань. Для гарного відображення максимальна рекомендована кількість питань – 50. Форма проходження опитувань зображена на рисунку 3.16.

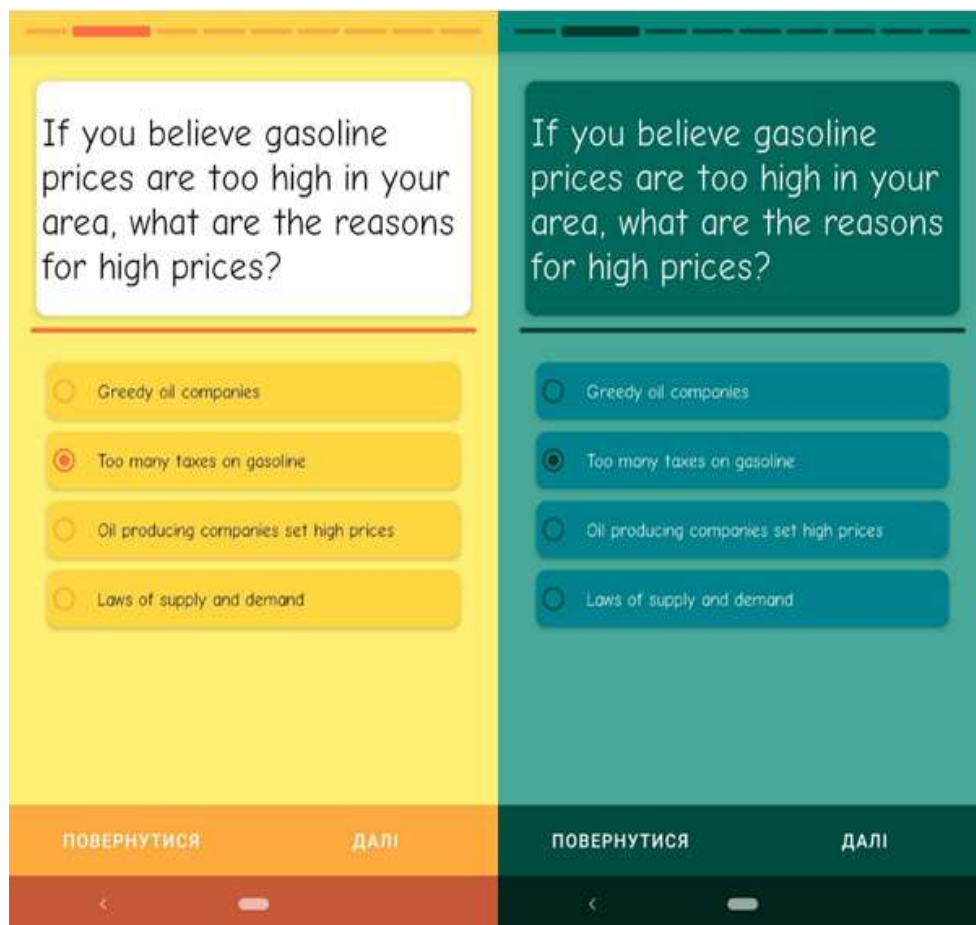


Рисунок 3.16 – Форма проходження опитування

Фіксованої максимальної кількості відповідей не встановлено, всі радіо-кнопки генеруються автоматично:

```
public void updateRadioButtonsInflate(Question question) {
    radios.removeAllViews();
    // ignoring changeListener to avoid Question.listener to be called
    ignoreOnChangeListener = true;
    for (int i = 0; i < question.getAnswersCount(); i++) {
        Answer answer = question.getAnswer(i);
        RadioButton radioButton = (RadioButton)
LayoutInflater.from(this).inflate(R.layout.answer_radio_button, radios, false);
        radioButton.setText(answer.answerText);
        radioButton.setEnabled(!poll.isCompleted());
        // Tag is used to get Answer object after check state changed
        // DO NOT CHANGE THE ORDER BELOW
        radios.addView(radioButton, i);
        radioButton.setTag(answer);
        radioButton.setChecked(answer.isChosen());
    }
    ignoreOnChangeListener = false;
}
```

Та метод, що обробляє натискання на радіо-кнопку:

```
public void onRadioButtonPress(RadioGroup group, int checkedId) {
    if (ignoreOnChangeListener) return;
    View view = findViewById(group.getCheckedRadioButtonId());
    if (view != null && view.getTag() != null) {
        getCurrentQuestion().setChosenAnswer((Answer) view.getTag());
        updateButtonState(buttonToNext, getCurrentQuestion().isAnswered());
        scaleUpButton(buttonToNext);
    }
}
```

Анімація верхнього повзунку:

```
public void configureDivider(int dividerIndex, boolean isActive, boolean
updateDividers) {
    if (dividerIndex < 0 || dividerIndex >= topDividers.getChildCount()) return;
    View divider = topDividers.getChildAt(dividerIndex);
    ((LinearLayout.LayoutParams) divider.getLayoutParams()).weight =
getWeightForDivider(isActive);
    divider.setEnabled(isActive);
    // Simplified; Scaled x2 if isActive else scaled to normal size x1
    divider.animate().scaleY(isActive ? 2f :
1f).setDuration(CONSTS.ANIMATION_DURATION).start();
    if (updateDividers)
        updateTopDividers();
}
```

Для уникнення помилок у додатку наявні константи типу булева, що відповідають за поведінку проходження питань. Якщо вони ввімкнені, то користувач зможе проходити питання у різному порядку, але перед завершенням його буде перекинуто на питання без відповіді, щоб уникнути помилок.

```

public void onNextButtonPress(View view) {
    try {
        if (!poll.isCompleted()) {
            if (PASS_UNCOMPLETED_QUESTIONS)
                getCurrentQuestion().completeQuestion();
            else
                getCurrentQuestion().completeQuestionWithError();
        }
    } catch (Exception e) {
        e.printStackTrace();
        // throwing Exception when "Next" button clicked and Question hasn't
        been answered before
        showErrorAndGoToQuestion(DIRECTION_NO_DIRECTION);
        return;
    }
    ...
}

```

Після завершення проходження користувачу буде показано вікно із його витраченим часом на опитування та автоматичне оновлення даних із сервером. У нових версіях текст більш насичений, додана подяка за виділений час. На рисунку 3.17 зображено вікно з витраченим часом на опитування.

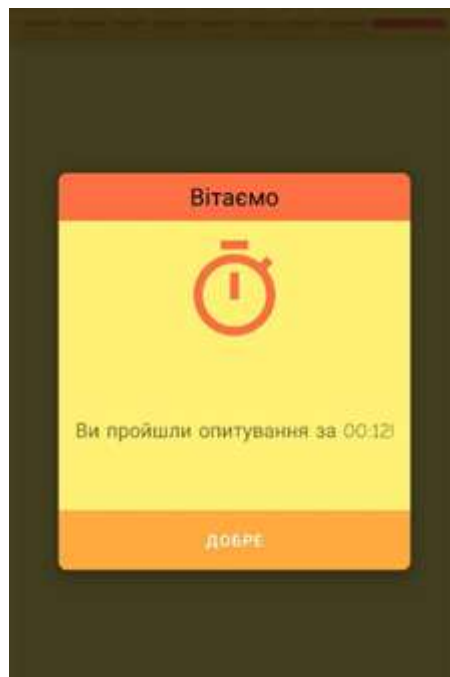


Рисунок 3.17 – Вікно з витраченим часом на опитування

Для підрахунку витраченого часу було написано клас `Timer`, що обчислює фактичний час проходження опитування. Для коректного підрахунку було використано методи аживіті, що інформують про зміну його життєвого

циклу.

У методі `AppCompatActivity.onCreate` генерується об'єкт `Timer`.

У методі `AppCompatActivity.onPause` визивається метод `Timer.pause()`, що оновлює фактичний час додаванням підрахунку «поточний час мінус час початку(`startDate`)»

У методі `AppCompatActivity.onResume` визивається метод `Timer.resume()`, що генерує нове значення змінної «`startDate`». Отримання відформатованого часу відбувається за допомогою методу `Timer.getFactTimeFormatted3parts(long factTime)` код якого нижче:

```
public static String getFactTimeFormatted3parts(long factTime){
    // Returns String like "00:00:15" seconds; Max is 23:59:59
    // It suits alot in our case
    // it cuts hours if they are null
    String formatted = LocalTime.ofSecondOfDay(factTime).toString();
    if (formatted.startsWith("00:"))
        return formatted.substring(3);
    return formatted;
}
```

Клас `Timer` зображено на рисунку 3.18.

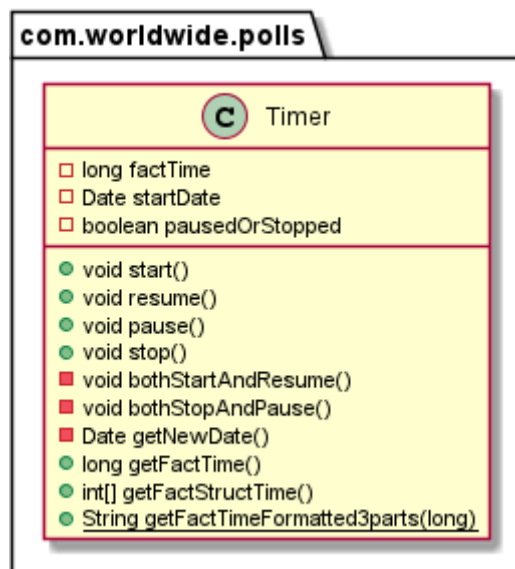


Рисунок 3.18 – Клас `Timer`

3.4 Збереження даних та файлів

Для збереження даних було використано статичний файл-інтерфейс під

назвою `com.worldwide.polls.CONSTS` із переліком констант для зручного використання. Він зберігає пошту, назви файлів, json-ключи, тривалість анімацій, та інші текстові дані, що повторно використовуються у додатку.

Та 3 метода зберігання користувацьких даних: бази даних, збереження у форматі json та записів `sharedPreferences`. `SharedPreferences` зберігає дані конфігурації користувача у форматі ключ-значення, а саме: обрана мова (англійська або українська) та тема додатку (день/ніч).

Реалізація зміни теми та мови додатку перед завантаженням із використанням `SharedPreferences`:

```
@Override
protected void attachBaseContext(Context newBase) {
    SharedPreferences prefs = getPreferences(newBase);
    if (!prefs.contains(CONSTS.PKEY_LANGUAGE))
        prefs.edit().putString(CONSTS.PKEY_LANGUAGE,
CONSTS.language_en).apply();
    String language = prefs.getString(CONSTS.PKEY_LANGUAGE, CONSTS.language_en);
    newBase.getResources().getConfiguration().setLocale(new Locale(language));

    if (!prefs.contains(CONSTS.PKEY_MODE))
        prefs.edit().putInt(CONSTS.PKEY_MODE,
AppCompatActivity.getDefaultNightMode()).apply();
    int mode = prefs.getInt(CONSTS.PKEY_MODE, AppCompatActivity.MODE_NIGHT_NO);
    updateNightMode(mode);
    super.attachBaseContext(newBase);
}
```

Формат json було використано для збереження структурованої інформації опитувань, яку повертає сервер.

3.4.1 SQLite

База даних має дві таблиці та зберігає дані, що стосуються опитувань та обраних відповідей.

Таблиця `Polls` має 4 поля:

- `PollID`: ідентифікатор опитування типу цілого числа;
- `isFavourite` та `isCompleted` поля мають тип істина. Але через те, що для SQLite не існує власного булевого типу даних відповідно до документації, дані зберігаються у вигляді цілого числа. Для `true` це 1, а для `false` це 0;
- `CompletingTime` це поле типу цілого числа, що зберігає інформацію

про час проходження опитування. Ці дані показують користувачу скільки часу він витратив на проходження.

У майбутньому ці дані будуть збиратися та показувати користувачеві середній час проходження, як статистика.

Таблиця PollQuestions має 4 поля: PollID, QuestionID, AnswerID та ResultSend

Перші три зберігають дані-ідентифікатори: опитування, питання та обраної відповіді. Коли користувач обирає та підтверджує одну з декількох відповідей - дані автоматично оновлюються у базі через інтерфейс-listener, а саме "OnActionListener", із параметром QUESTION_COMPLETE_STATE із перерахування "Actions".

Поле ResultSend може зберігати лише одне з трьох значень: SEND_RESULT_CAN_NOT_BE_UPLOADED, SEND_RESULT_HAVE_TO_BE_UPLOADED, SEND_RESULT_DONE. Доти, поки опитування ще не пройдено - ResultSend має значення SEND_RESULT_CAN_NOT_BE_UPLOADED для всіх обраних відповідей, що стосуються опитування.

Після того, як користувач завершує проходження, поле ResultSend змінює значення на SEND_RESULT_HAVE_TO_BE_UPLOADED, що означає, що дані можуть бути відправлені до серверу для обробки інформації та оновлення статистики. Якщо сервер повернув відповідь "успішно", дані поля ResultSend змінюються на значення SEND_RESULT_DONE.

3.4.2 Помічник DBManager для роботи із SQLite

Для зручної роботи із базою та спрощенням коду було створено клас, що виконує всі необхідні операції створення, оновлення та додавання записів.

Основні методи, що замінюються із класу SQLiteOpenHelper це onCreate та onUpgrade. onCreate відповідає за створення нової бази даних, а onUpgrade виконується тоді, коли попередня версія бази нижче за нову. Якщо поточна версія менша за попередню, то викликається метод onDowngrade.

Вигляд назв полів-констант та реалізація методів onCreate та onUpgrade, створення таблиць:

```
public static final String DATABASE_NAME = "PollData";
public static final String TABLE_POLLS = "Polls";
public static final String TABLE_POLL_QUESTIONS = "PollQuestions";
public static final String POLL_ID = "PollID";
public static final String IS_FAVOURITE = "IsFavourite";
public static final String IS_COMPLETED = "IsCompleted";
public static final String COMPLETING_TIME = "PollCompletingTime";
public static final String QUESTION_ID = "QuestionID";
public static final String ANSWER_ID = "AnswerID";
public static final String RESULT_SEND = "ResultSend";
public static final int SEND_RESULT_CAN_NOT_BE_UPLOADED = -1;
public static final int SEND_RESULT_HAVE_TO_BE_UPLOADED = 0;
public static final int SEND_RESULT_DONE = 1;

public DBManager(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
}
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(fmt("CREATE TABLE IF NOT EXISTS %s(%s INTEGER, %s boolean, %s
boolean, %s INTEGER)",
        TABLE_POLLS, POLL_ID, IS_FAVOURITE, IS_COMPLETED, COMPLETING_TIME));
    db.execSQL(fmt("CREATE TABLE IF NOT EXISTS %s(%s INTEGER, %s INTEGER, %s
INTEGER, %s INTEGER)",
        TABLE_POLL_QUESTIONS, POLL_ID, QUESTION_ID, ANSWER_ID,
RESULT_SEND));
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.delete(TABLE_POLLS, null, null);
    db.delete(TABLE_POLL_QUESTIONS, null, null);
    onCreate(db);
}
```

Додавання рядків до таблиці відбувається за допомогою методу класу ContentValues та вбудованого методу insert(String tableName, String ..., ContentValues cv). Перед тим, як оновити рядки у базі, викликається метод createQuestionRowIfNotExists/createPollRowIfNotExists, що перевіряють наявність рядків за відповідними ідентифікаторами та додають новий рядок, якщо це потрібно.

Приклад оновлення таблиці Poll:

```
public static int boolToInt(boolean value){ return value? 1 : 0; }
public String eq(String key, int value) { return fmt("%s=%d", key, value); }
public void updatePollFields(int pollID, Boolean isFavourite, Boolean
isCompleted, Long completingTime) {
    createPollRowIfNotExists(pollID);
    ContentValues cv = new ContentValues();
    if (isFavourite != null)
        cv.put(IS_FAVOURITE, boolToInt(isFavourite));
    if (isCompleted != null)
        cv.put(IS_COMPLETED, boolToInt(isCompleted));
}
```

```

        if (completingTime != null)
            cv.put(COMPLETING_TIME, completingTime);
        if (cv.size() == 0)
            return;
        getConnection().update(TABLE_POLLS, cv, eq(POLL_ID, pollID), null);
    }
    public void updatePollCompleted(int pollID, boolean isCompleted, long
completingTime) {
        updatePollFields(pollID, null, isCompleted, completingTime);
    }
    public void updatePollFavourite(int pollID, boolean isFavourite) {
        updatePollFields(pollID, isFavourite, null, null);
    }
}

```

Зчитування даних с бази виконується за допомогою реалізованого методу `read(String sqlQuery, String... args)` результатом є `ArrayList<ArrayList<Object>>`, що повертає масив даних із таблиці. Реалізація отримання значення, що залежить від типу cursor:

```

private Object getValue(Cursor cursor, int index) {
    int type = cursor.getType(index);
    if (type == Cursor.FIELD_TYPE_INTEGER) {
        return cursor.getInt(index);
    } else if (type == Cursor.FIELD_TYPE_STRING) {
        return cursor.getString(index);
    } else if (type == Cursor.FIELD_TYPE_BLOB) {
        return cursor.getBlob(index);
    } else if (type == Cursor.FIELD_TYPE_FLOAT) {
        return cursor.getFloat(index);
    } else if (type == Cursor.FIELD_TYPE_NULL) {
        return null;
    }
    return cursor.getString(index);
}

```

3.5 Висновки за розділом 3

У розділі 3 було розглянуто основні рішення щодо реалізації компонентів системи. Описана робота застосунку, основні процеси, взаємодія із сервером та методи збереження даних та файлів. Наведені діаграми класів, рисунки форм, повідомлень та вікон застосунку.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ЗАСТОСУНКУ

4.1 Експлуатація та процес роботи застосунку

Нижче наведено процес роботи застосунку, опис дій користувача, системи, серверу та бази даних SQLite.

Користувач запускає застосунок, система відправляє запит на дані опитувань до БД на сервері, сервер повертає дані, система генерує JSON файли та відображає їх користувачу у відкритій головній формі застосунку.

Користувач натискає на опитування, система відкриває форму проходження опитування та завантажує питання та відповіді з JSON файлу.

Користувач проходить опитування, система зберігає та відправляє результати проходження до бази даних на сервері, оновлює статистику та зберігає дані (обрані відповіді, час проходження) про опитування в SQLite.

Користувач в меню застосунку обирає пункт «Обрані», система відкриває відповідну форму та завантажує обрані опитування з бази даних SQLite.

Користувач в меню застосунку обирає пункт «Пройдені опитування», система відкриває відповідну форму та завантажує пройдені опитування з бази даних SQLite.

Користувач в меню застосунку обирає пункт «Пошук», система відкриває відповідну форму, користувач вводить назву опитування, система завантажує опитування з JSON файлів.

Користувач натискає на кнопку перегляду статистики опитування, система відкриває відповідну форму та завантажує статистику відповідей з серверу.

Користувач обирає фільтрацію опитувань, система відкриває вікно фільтрації, користувач обирає параметри фільтрації, система завантажує дані з JSON файлу.

Користувач натискає на кнопку оновлення опитувань, система відправляє запит на опитування до БД на сервері, сервер повертає дані, система генерує нові JSON файли з даних та відображає їх користувачу.

4.2 Тестування застосунку

Тестування застосунку проводилось на мобільних та планшетних пристроях з різними версіями ОС Android, мобільним підключенням до інтернету та wi-fi.

В ході тестування були проведені наступні перевірки:

- перевірка на завантаження застосунку;
- перевірка на адаптивність інтерфейсу;
- перевірка на завантаження опитувань;
- перевірка на збереження результатів проходження;
- перевірка на збереження пройдених та обраних опитувань;
- перевірка на пошук та сортування опитувань;
- перевірка на переходи між формами.

На рисунку 4.1 зображено інтерфейс застосунку на планшетному пристрої.

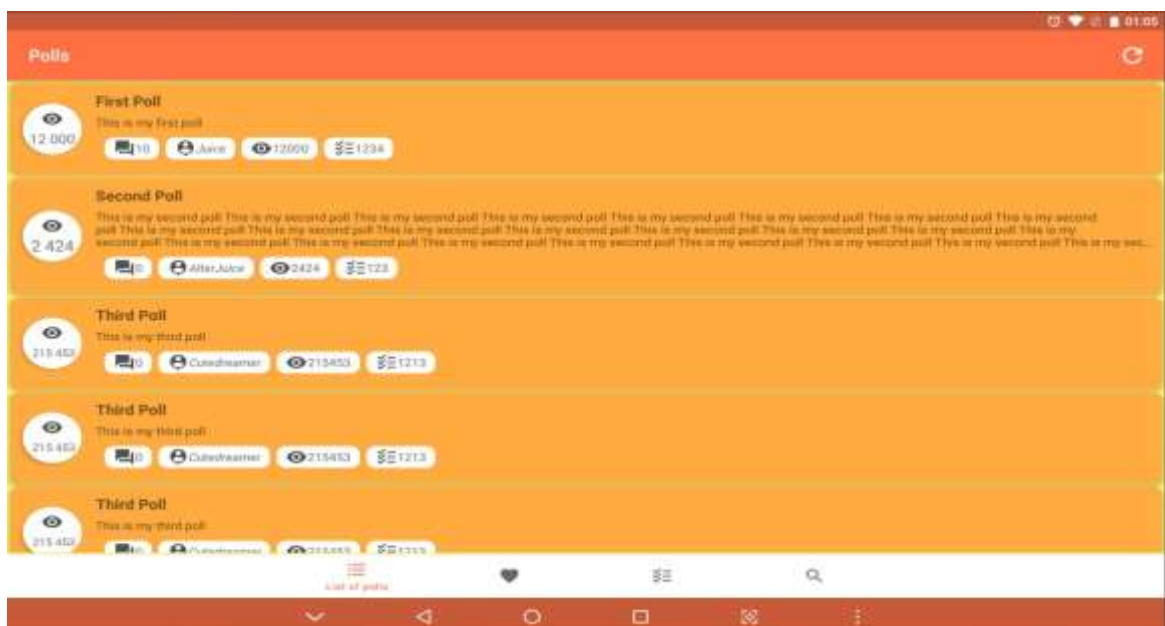


Рисунок 4.1 – Інтерфейс застосунку на планшетному пристрої.

Нижче в таблиці 4.1 наведено чек-лист тестування застосунку «Polls - WorldWide».

Таблиця 4.1 – Чек-лист тестування застосунку «Polls - WorldWide»

Перевірка	Телефон – Realme 6 Pro Версія Android - 10	Телефон - Xiaomi Note 10 Версія Android 11	Планшет – Lenovo Tab M10 Plus Версія Android– 9
Завантаження застосунку	пройдено	пройдено	пройдено
Адаптивність інтерфейсу	пройдено	пройдено	пройдено
Завантаження опитувань	пройдено	пройдено	пройдено
Збереження результатів проходження	пройдено	пройдено	пройдено
Збереження пройдених та обраних опитувань	пройдено	пройдено	пройдено
Пошук та сортування опитувань	пройдено	пройдено	пройдено
Переходи між формами	пройдено	пройдено	пройдено

Тестування розробленого програмного забезпечення було успішним на всіх пристроях з різними характеристиками. Усі дії, які виконує користувач у програмі виконуються швидко, приблизно за 0.5 секунди. До цих дій відносяться завантаження застосунку, робота з опитуваннями, перехід між формами, звертання до БД за даними та їх відображення, Усі елементи керування на користувацьких формах на різних пристроях не «з’їджують», тобто інтерфейс добре адаптується.

4.3 Висновки за розділом 4

У шостому розділі було обрано перевірки, які гарантуватимуть коректність роботи ПЗ, описано сценарій експлуатації та тестування.

За результатами тестування на трьох різних пристроях багів не виявлено, тестування за всіма перевітками пройшло успішно. Застосунок є стабільним на різних пристроях з ОС Android.

ВИСНОВКИ

Під час розробки Android – застосунку для дипломного проекту, були проаналізовані сучасні технології та інструменти для створення застосунків для операційної системи Android.

Для розробки застосунку обрано середу розробки Android Studio, так як у неї найзручніший користувацький інтерфейс, багато доступної документації, підтримка відомих мов програмування, достатньо велика оцінка користувачів, безкоштовність користування і постійні оновлення, які додають середі розробці ще більше корисного функціоналу.

Для проектування була обрана мова програмування Java, так як відповідає усім вимогам розробки застосунку а також є найпопулярнішою мовою програмування для мобільної розробки на Android, є офіційною мовою для розробки Android - застосунків і підтримується Android Studio.

Для зберігання даних були обрані локальна база даних SQLite та хмарний сервер PythonAnywhere, який підтримує мову Python.

Під час розробки застосунку були вивчені та застосовані:

- робота із формами та вікнами;
- обробка натискань на кнопки;
- робота із JSON файлами та базою даних;
- запити до серверу;
- розробка інтуїтивного інтерфейсу користувача.

Технічне завдання на розробку наведено у додатку А.

Створений проект відповідає основній меті розробки, а саме: застосунок «Polls – WorldWide» призначений для проведення опитувань та зберігання результатів.

Основні функції, які виконує застосунок:

- проходження опитувань;
- пошук та сортування опитувань;

- збереження результатів проходження;
- додавання обраних та зберігання пройдених опитувань.

Повний код проекту наведено у додатку В.

Для тестування застосунку були проведені наступні перевірки:

- перевірка на завантаження застосунку;
- перевірка на адаптивність інтерфейсу;
- перевірка на завантаження опитувань;
- перевірка на збереження результатів проходження;
- перевірка на збереження пройдених та обраних опитувань;
- перевірка на пошук та сортування опитувань;
- перевірка на переходи між формами.

За результатами тестування на трьох різних пристроях багів не виявлено, тестування за всіма перевітками пройшло успішно. Застосунок є стабільним на різних пристроях з ОС Android.

Шляхи удосконалення застосунку:

- адміністративна частина застосунку, яка дозволить створювати опитування та відправляти їх до БД на сервері;
- множинний вибір відповідей на запитання;
- підтримка малюнків у запитаннях;
- облікові записи користувачів;
- доопрацювання інтерфейсу для потреб користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Pollscape [Електрон. ресурс] – Режим доступу: <https://play.google.com/store/apps/details?id=com.app.pollscape.release&hl=ru&gl=US>
2. Poll For All [Електрон. ресурс] – Режим доступу: <https://www.pollforall.com/ru/index.html>
3. Pollie [Електрон. ресурс] – Режим доступу: <https://play.google.com/store/apps/details?id=nl.pibit.pollie&hl=ru&gl=US>
4. Android [Електрон. ресурс] – Режим доступу: <https://uk.wikipedia.org/wiki/Android>
5. Barry, Burd Android Application Development All-in-One For Dummies® / Barry Burd. - Москва: Машинобудування, 2011. - 816 с.
6. Голощاپов, Олексій Google Android. Програмування для мобільних пристроїв (+ CD-ROM) /Олексій Голощاپов. - М.: БХВ-Петербург, 2011. - 438 с.
7. Дерсі, Лорен Android за 24 години. Програмування додатків під операційну систему Google / Лорен Дерсі, Шейн Кондер. - М.: Рід Груп, 2011. - 464 с.
8. Документація Android Source Project [Електрон. ресурс] - Режим доступу: <https://source.android.com/setup/start/build-numbers>
9. Підтримка Android [Електрон. ресурс] - Режим доступу: <https://support.google.com/android/?hl=ru#topic=7313011>
10. Можливості операційної системи Android [Електрон. ресурс] - Режим доступу: http://android.com.ua/android_os.html
11. Програмування на Android [Електрон. ресурс] - Режим доступу: <http://android.mobile-review.com/>
12. Android Studio [Електрон. ресурс] - Режим доступу: https://uk.wikipedia.org/wiki/Android_Studio
13. Особенности Android Studio и других сред разработки [Електрон. ресурс] - Режим доступу: <https://habr.com/ru/post/433604/>

14. Давидов, Станіслав IntelliJ IDEA. Професійне програмування на Java. Найбільш повне керівництво (+ CD-ROM) / Станіслав Давидов, Олексій Єфімов. - М.: БХВ-Петербург, 2011. - 800 с.
15. IntelliJ IDEA [Електрон. ресурс] - Режим доступу: https://uk.wikipedia.org/wiki/IntelliJ_IDEA
16. Eclipse [Електрон. ресурс] - Режим доступу: <https://ru.wikipedia.org/wiki/Eclipse>
17. Документация Eclipse [Електрон. ресурс] - Режим доступу: <https://www.eclipse.org/>
18. Машнін, Т. С. Eclipse. Розробка RCP-, Web-, Ajax- і Android-додатків на Java / Т.С. Машнін. - М.: БХВ-Петербург, 2013. - 384 с.
19. IntelXDK [Електрон. ресурс] - Режим доступу: https://ru.wikipedia.org/wiki/Intel_XDK#
20. Нотон Java. Довідкове керівництво. Все, що необхідно для програмування на Java / Нотон, Патрік. - М.: Біном, 2015. - 448 с.
21. Сравнение языков программирования для Android разработки [Електрон. ресурс] - Режим доступу: <https://habr.com/ru/post/477578/#>:
22. PythonAnyWhere [Електрон. ресурс] - Режим доступу: <https://ru.wikipedia.org/wiki/PythonAnywhere>
23. Урок 34. Хранение данных. SQLite [Електрон. ресурс] – Режим доступу: <https://startandroid.ru/ru/uroki/vse-uroki-spiskom/74-urok-34-hranenie-dannyh-sqlite.html>.
24. SQL [Електрон. ресурс] - Режим доступу: <https://ru.wikipedia.org/wiki/SQL>
25. Команды SQL [Електрон. ресурс] - Режим доступу: <https://htmlacademy.ru/tutorial/php/sql>

ДОДАТОК А
Технічне завдання

A.1 Вступ

Найменування продукту – застосунок «Polls - WorldWide». Застосунок «Polls - WorldWide» призначен для проходження опитувань та збереження результатів проходження.

A.2 Підстави для розробки

Підставою для розробки служить наказ 30 березня 2021 р. № 103 Національного Університету «Запорізька Політехніка» на дипломних кваліфікаційних робіт.

Проект на тему «Розроблення мобільного застосунку для підтвердження якості процесів і продуктів міжнародних навчальних проектів. Mobile application development for quality assurance of the processes and products in international educational projects».

A.3 Призначення розробки

Метою розробки програмного забезпечення є перегляд, пошук, проходження опитувань, зберігання результатів та статистики кожного з них а також створення простого та приємного для користувача інтерфейсу. Головне призначення застосунку – дати оцінку якості навчальному процесу.

A.4 Вимоги до програми чи програмного виробу

A.4.1 Вимоги до функціональних характеристик

Функції, які має виконувати застосунок:

- проходження опитувань;
- пошук та сортування опитувань;
- збереження результатів проходження;

- додавання обраних та зберігання пройдених опитувань.

Вхідними даними є JSON файл з опитуваннями, який система створила з даних на сервері після завантаження застосунку.

Вихідними даними є результати проходження опитування.

A.4.2 Вимоги до надійності

Головна вимога з надійності до ПЗ – стабільна робота в будь-яких умовах. Необхідно запобігти закриття програми з причини помилки, запобігти невірні дії користувача.

A.4.3 Умови експлуатації

Загальні умови експлуатації співпадають з умовами експлуатації девайсу, на якому відбувається функціонування застосунку. Також для коректної роботи застосунку необхідне стабільне під'єднання до мережі Internet.

A.4.4 Вимоги до складу й параметрів технічних засобів

Технічні вимоги до обладнання:

- операційна система – Android 6 чи вище;
- вільний простір не менше 10 мб;
- не менше 1 Гб ОЗУ;
- процесор Qualcomm Snapdragon 450 1300MHz чи краще.

A.4.5 Вимоги до інформаційної й програмної сумісності

Програмні вимоги:

- версія Android 6 і вище;
- доступ до мережі інтернет;

- сервіс віртуальної клавіатури.

A.4.6 Вимоги до маркування й упакування

Програма повинна мати такі маркування: «Polls - WorldWide». У повному обсязі проект може бути записаний на компакт диск не менше 50мб.

A.4.7 Вимоги до транспортування й збереження

Вимоги до транспортування і зберігання аналогічні вимогам, що пред'являються до гнучких і лазерних дисків.

A.5 Вимоги до програмної документації

Програмна документація повинна включати: пояснювальну записку на розроблений продукт, вихідний код програми.

A.6 Техніко-економічні показники

Монетизація застосунку передбачає публікацію у Internet магазинах, можлива монетизація в якості платного скачування або монетизації реклами, що потім буде вмонтована в застосунок.

A.7 Порядок контролю й приймання

Передбачено захист дипломного проєкту у встановлені керівником проєкту строки.

ДОДАТОК Б
Текст програми

```
package com.worldwide.polls;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.app.AppCompatActivityDelegate;
import androidx.appcompat.widget.SearchView;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.recyclerview.widget.RecyclerView;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.util.Pair;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.CompoundButton;
import android.widget.FrameLayout;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.android.material.button.MaterialButton;
import com.google.android.material.checkbox.MaterialCheckBox;
import com.worldwide.polls.Request.RequestManager;
import com.worldwide.polls.Request.RequestPacker;
import com.worldwide.polls.Request.RequestParams;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

import static com.worldwide.polls.Utils.boolToInt;

public class ActivityMain extends AppCompatActivity implements
BottomNavigationView.OnNavigationItemSelectedListener{
    static String TAG = "ActivityMain";
    ConstraintLayout mainParent; // Main background
    // Top App Bar Views
    SearchView searchView;
    MenuItem searchItem;
    MenuItem searchFiltersItem;
    MenuItem menuFavourite;
    MenuItem menuUpdating;

    TextView itemsAreEmptyMsg; // TextView that shows centered message about
empty items
    RecyclerView recyclerView;
    PollAdapter recyclerAdapter;
```

```

BottomNavigationView navigationView; // Main bottom menu

FrameLayout searchFieldsParent;
LinearLayout searchFields;
MaterialButton searchSearchIn;
MaterialButton searchSortBy;
AlertDialog dialogSearchIn;
AlertDialog dialogSortBy;
MaterialCheckBox searchIsCaseSensitive;
MaterialCheckBox searchReverse;

Handler mHandler = new Handler();

FilterConfigs configs;

boolean pollsAreUpdating = false;
boolean filtersAreVisible = false;
boolean pollInfoIsShowing = false;
File pathToPolls;
DBManager dbManager;

Pair<Integer, Integer> currentPairColorString;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    dbManager = new DBManager(this);
    pathToPolls = new File(this.getFilesDir(), CONSTS.PATH_TO_POLLS);
    if (!pathToPolls.exists())
        pathToPolls.mkdir();

    try {
        ContentManager.loadAllPolls(dbManager, pathToPolls);
    } catch (JSONException e) {
        e.printStackTrace();
    }

    mainParent = findViewById(R.id.mainActivityParent);
    navigationView = findViewById(R.id.bottomMenu);
    recyclerView = findViewById(R.id.pollListRecycler);
    itemsAreEmptyMsg = findViewById(R.id.itemsEmptyMsg);
    searchSearchIn = findViewById(R.id.searchSearchIn);
    searchSortBy = findViewById(R.id.searchSortBy);
    searchIsCaseSensitive = findViewById(R.id.searchCaseSensitive);
    searchReverse = findViewById(R.id.searchReverse);
    searchFieldsParent = findViewById(R.id.searchFieldsParent);
    searchFields = findViewById(R.id.searchFields);
    navigationView.setOnItemClickListener(this);
    navigationView.addOnLayoutChangeListener((v, left, top, right, bottom,
oldLeft, oldTop, oldRight, oldBottom) -> {
    /*
        View.OnLayoutChangeListener()
        How to detect a layout resize?:
https://stackoverflow.com/a/34446944/8120181
        Log.d("onLayoutChange: New (LTRB): ", left+" "+top+" "+right+"
        "+bottom+"; Old: "+oldLeft+" "+oldTop+" "+oldRight+" "+oldBottom);

        Function changes fragmentContainer padding to avoid overlaying
navigationMenu on fragment
        ((ViewGroup.MarginLayoutParams)
fragmentContainer.getLayoutParams()).bottomMargin = bottomMarginPX;
        Margin is not updating in realTime so padding is more effective
        Why Zero: https://stackoverflow.com/a/15556218/8120181

```

```

        */
        recyclerView.setPadding(0,0,0, bottom-top);
    });

    // Setting background color centered info-text
    currentPairColorString = new Pair<>(R.color.backgroundFullPollList,
R.string.no_polls);
    configs = new FilterConfigs();
    recyclerViewAdapter = new PollAdapter(ContentManager.getPolls(),
startPollInfoFragment);
    recyclerView.setAdapter(recyclerViewAdapter);

    searchFieldsParent.setOnClickListener(v ->
toggleFilterViewShowHide(false));
    searchSearchIn.setOnClickListener(v->dialogSearchIn.show());
    searchSortBy.setOnClickListener(v->dialogSortBy.show());
    searchIsCaseSensitive.setChecked(configs.isCaseSensitive);

searchIsCaseSensitive.setOnCheckedChangeListener(this::onCaseChangeListener);
searchReverse.setChecked(configs.reverse);
searchReverse.setOnCheckedChangeListener(this::onCaseChangeListener);
dialogSearchIn = createSearchInDialog();
dialogSortBy = createSortByDialog();

Poll.setListener((action, poll) -> {
    if (pollsAreUpdating) return;
    int index = recyclerViewAdapter.getItemIndexByPollID(poll.intID);
    switch (action){
        case POLL_CHANGE_FAVOURITE:
            dbManager.updatePollFavourite(poll.intID,
poll.isFavourite());
            showToast(getString(poll.isFavourite() ?
R.string.action_add_favourite :
R.string.action_removed_favourite,
poll.name.substring(0, 10)));
            recyclerViewAdapter.notifyItemChangedByID(poll.intID);
            // recyclerViewAdapter.updateOnPosition(index, poll);
            break;
        case POLL_ADD_NEW_QUESTION:
            break;
        case POLL_CLEAR_QUESTIONS:
            break;
        case POLL_COMPLETE_STATE: // COMPLETE_STATE and COMPLETING TIME
ARE RUNNING AT THE SAME MOMENT BUT COMPL_TIME IS THE LAST
            break;
        case POLL_SET_COMPLETING_TIME:
            dbManager.updatePollCompleted(poll.intID,
poll.isCompleted(), poll.getCompletingTime());
            dbManager.updatePollQuestionsSendResult(poll.intID,
DBManager.SEND_RESULT_HAVE_TO_BE_UPLOADED);
            updateUnsentAnswersOfPoll(poll);
            break;
    }
    // recyclerViewAdapter.notifyDataSetChanged();
    // recyclerViewAdapter.notifyItemChangedByID(poll.intID);
});
Question.setListener((action, question) -> {
    switch (action){
        case QUESTION_ADD_NEW_ANSWER:
            // no any deeds
            break;
        case QUESTION_COMPLETE_STATE:

dbManager.updateQuestionFields(question.provider.getParent().intID,

```

```

        question.intID, question.getChosenAnswerID(),
DBManager.SEND_RESULT_CAN_NOT_BE_UPLOADED);
        break;
        case QUESTION_CLEAR_ANSWERS:
            // no any deeds
            break;
    }
});
// itemsAreEmptyMsg.setVisibility(View.VISIBLE);
// itemsAreEmptyMsg.setText(currentPairColorString.second);

searchFields.setTranslationY(-CONSTS.ANIMATION_TRANSLATION_Y);
setAppBarSubtitle(getString(R.string.list_of_polls));
updateTextSearchSortBy();
updateTextSearchSearchIn();
mHandler.postDelayed(this::onUpdateIconClick, 1000);
}

@Override
protected void attachBaseContext(Context newBase) {
    // https://stackoverflow.com/a/54489310/8120181
    // How to change language of app when user selects language?
    https://stackoverflow.com/a/46024733/8120181
    SharedPreferences x = getPreferences(newBase);
    if (!x.contains(CONSTS.PKEY_LANGUAGE))
        x.edit().putString(CONSTS.PKEY_LANGUAGE,
CONSTS.language_en).apply();
    String language = x.getString(CONSTS.PKEY_LANGUAGE,
CONSTS.language_en);
    newBase.getResources().getConfiguration().setLocale(new
Locale(language));

    if (!x.contains(CONSTS.PKEY_MODE))
        x.edit().putInt(CONSTS.PKEY_MODE,
AppCompatActivity.getDefaultNightMode()).apply();
    int mode = x.getInt(CONSTS.PKEY_MODE, AppCompatActivity.MODE_NIGHT_NO);
    updateNightMode(mode);
    super.attachBaseContext(newBase);
}

final OnFilterEnd onFilterEndUsual = new OnFilterEnd() {
    @Override
    public void onFilterProcessEnd(ArrayList<Poll> filteredPolls) {
        mainParent.setBackgroundResource(currentPairColorString.first);
        if (!filteredPolls.isEmpty()) {
            itemsAreEmptyMsg.setVisibility(View.INVISIBLE);
            recyclerAdapter.setItems(filteredPolls);
        } else {
            recyclerAdapter.clearItems();
            itemsAreEmptyMsg.setText(currentPairColorString.second);
            itemsAreEmptyMsg.setVisibility(View.VISIBLE);
            // Show to user with large text size that list is empty
        }
    }
};

public RequestInterface updateEveryPollInterface = new RequestInterface() {
    @Override
    public void onServerResponse(JSONObject jsonResponse) {
        JSONObject jsonPoll;
        int pollID;
        try {

```

```

        jsonPoll = jsonResponse.getJSONObject(CONSTS.JKEY_DATA);
        pollID = jsonPoll.getInt(CONSTS.JKEY_ID);
    } catch (JSONException e) {
        e.printStackTrace();
        Log.d(TAG, "Missed DATA key");
        return;
    }
    updatePollWithResponse(jsonPoll, pollID);
}
@Override
public void onServerFailure(JSONObject jsonResponse, int errorCode)
{/*showToast("Server error: code " + errorCode);*/}
@Override
public void onNoConnection() {/*Log.d(TAG, "onNoConnection");*/}
};

public RequestInterface updateAllPollsInterface = new RequestInterface() {
    @Override
    public void onServerResponse(JSONObject jsonResponse) {
        pollsAreUpdating = true;
        try {
            JSONArray x = jsonResponse.getJSONArray(CONSTS.JKEY_POLL_IDS);
            ArrayList<Integer> integers = new ArrayList<>();
            for (int i = 0; i < x.length(); i++) { integers.add((Integer)
x.get(i)); }

            updateEveryPoll(integers);
        } catch (JSONException e) {
            e.printStackTrace();
            pollsAreUpdating = false;
        }
    }

    @Override
    public void onServerFailure(JSONObject jsonResponse, int errorCode) {
        showToast("Server error: code " + errorCode+ ". Try again later");
    }
    @Override
    public void onNoConnection() {

showToast(getString(R.string.no_connection_unable_connect_the_server));
    }
};

private SharedPreferences getPreferences(Context context){
    return context.getSharedPreferences(CONSTS.PREFERENCE_FILENAME,
MODE_PRIVATE);
}
private String getPollFileName(int pollID){
    return String.format(CONSTS.POLL_FILENAME, pollID);
}

private File getPathToPoll(int pollID){
    return new File(pathToPolls, getPollFileName(pollID));
}

/**
 * Changes the menuFavourite's icon from filled to unfilled
 * @param isFavourite poll's isFavourite var
 */
public void updateFavouriteIcon(boolean isFavourite){
    menuFavourite.setIcon(isFavourite ? R.drawable.star_round_filled :
R.drawable.star_round_outlined);
}

```

```

        public MenuItem.OnMenuItemClickListener
generateFavouriteItemClickListener(int pollID){
            return new MenuItem.OnMenuItemClickListener() {
                @Override
                public boolean onMenuItemClick(MenuItem item) {
                    Poll poll = ContentManager.getPollByUniqueID(pollID);
                    if (poll == null) return false;
                    poll.setFavourite(!poll.isFavourite());
                    updateFavouriteIcon(poll.isFavourite());
                    return true;
                }
            };
        }

        CallbackUniqueID startPollInfoFragment = pollID -> {
            if (pollInfoIsShowing) return;
            togglePollInfoShowHide(true);
            Poll poll = ContentManager.getPollByUniqueID(pollID);
            if (poll == null) return;
            updateFavouriteIcon(poll.isFavourite());
            menuFavourite.setVisible(true);
        };

menuFavourite.setOnMenuItemClickListener(generateFavouriteItemClickListener(pollID));
getSupportFragmentManager()
    .beginTransaction()
    .setCustomAnimations(R.anim.fragment_open_enter,
R.anim.fragment_open_exit, R.anim.fragment_fade_enter, R.anim.fragment_fade_exit)
    .add(R.id.fragment, FragmentPollInfo.newInstance(pollID,
getColor(currentPairColorString.first()), "PollInfo")
    .addToBackStack("PollInfo")
    .commit();
};

        private void showToast(String message) {
            // runOnUiThread() -> Snackbar.make(mainParent, message,
com.google.android.material.snackbar.BaseTransientBottomBar.LENGTH_SHORT)
            //
            .setAnimationMode(com.google.android.material.snackbar.BaseTransientBottomBar.ANIMATION_MODE_FADE).show();
            runOnUiThread() -> Toast.makeText(ActivityMain.this, message,
Toast.LENGTH_SHORT).show();
        }

        public void onUpdateIconClick(){
            updateAllPolls();
            ArrayList<ArrayList<Object>> unsentAnswers =
dbManager.getAnswersByResult(DBManager.SEND_RESULT_HAVE_TO_BE_UPLOADED);
            updateUnsentAnswers(unsentAnswers);
        }

        public void updateAllPolls(){
            new RequestManager(RequestPacker.requestGetPollIDs(),
updateAllPollsInterface).makeRequest();
        }

        public void updateUnsentAnswersOfPoll(Poll poll) {
            ArrayList<ArrayList<Object>> unsentAnswers =
dbManager.getAnswersByResult(poll.intID, DBManager.SEND_RESULT_HAVE_TO_BE_UPLOADED);
            updateUnsentAnswers(unsentAnswers);
        }

```

```

    }

    public void updateUnsentAnswers(ArrayList<ArrayList<Object>> unsentAnswers)
    {
        if (unsentAnswers.isEmpty()) {
            Log.d(TAG, "Unsent answers are empty. Exiting updating!");
            return;
        }
        Log.d(TAG, "Unsent answers are not empty: "+unsentAnswers.toString());
        RequestParams unsetAnswersParams;
        unsetAnswersParams =
RequestPacker.requestPostUnsentAnswers(unsentAnswers);
        new RequestManager(unsetAnswersParams, new RequestInterface() {
            @Override
            public void onServerResponse(JSONObject jsonResponse) {
                dbManager.setAnswersAsSent(unsentAnswers,
DBManager.SEND_RESULT_DONE);
                updateAllPolls();
            }

            @Override
            public void onServerFailure(JSONObject jsonResponse, int errorCode)
        {
                showToast(getString(R.string.server_failure_unsent));
            }

            @Override
            public void onNoConnection() {
                showToast(getString(R.string.no_connection_unsent));
            }
        }).makeRequest();
    }

    public void updatePollWithResponse(JSONObject jsonPoll, Integer pollID){
        File pathToPoll = getPathToPoll(pollID);
        try {
            // The function returns boolean data type representing whether the
new file is created or not.
            if (pathToPoll.createNewFile()) {
                dbManager.createPollRowIfNotExists(pollID);
                Log.d(TAG, "File created");
            }
            Utils.saveToFile(jsonPoll, pathToPoll); // saving to file
        } catch (IOException e) {
            Log.d(TAG, "createNewFile or saveToFile error");
            e.printStackTrace();
            return;
        }
        Poll p = ContentManager.updatePoll(dbManager, pathToPoll); // updating
Poll object from resource
        if (p == null) {
            Log.d(TAG, "Poll is null after ContentManager.updatePoll()");
            return;
        }
        int itemIndex = recyclerAdapter.getItemIndexByPollID(p.intID);
        if (itemIndex != -1) {
            recyclerAdapter.updateOnPosition(itemIndex, p);
        } else {
            recyclerAdapter.addItem(p);
        }
    }

    public void updateEveryPoll(ArrayList<Integer> jsonPollIDS){

```

```

        for (int i=0; i < jsonPollIDS.size(); i++) {
            int pollID = jsonPollIDS.get(i);
            boolean pathToPollExists = getPathToPoll(pollID).exists();
            RequestManager request = new
RequestManager(RequestPacker.requestGetPollByID(pollID,
boolToInt(!pathToPollExists)));
            request.setListener(updateEveryPollInterface);
            if (i == jsonPollIDS.size()-1)
                request.setEndAction(() -> {
                    pollsAreUpdating = false;
                    showToast(getString(R.string.polls_successfully_updated));
                });
            request.makeRequest();
        }
    }

    @Override
    public void onBackPressed() {
        if (pollInfoIsShowing) {
            getSupportFragmentManager().popBackStack();
            if (getSupportFragmentManager().getBackStackEntryCount() == 1){
                togglePollInfoShowHide(false);
                setAppBarSubtitle(null);
                setTopMenuOnDependence();
                menuFavourite.setVisible(false);
                menuFavourite.setOnMenuItemClickListener(null);
            }
        }else{
            if (filtersAreVisible)
                toggleFilterViewShowHide(false);
            else {
                super.onBackPressed();
            }
        }
    }

    @Override
    protected void onStop() {
        /*
        Save scroll-position while switching between Fragments:
https://stackoverflow.com/a/48612294/8120181

        For future use, If you are switching between Fragments
        within the same activity and all you want to do is save scroll-position
for recyclerview
        and then restore recyclerview to the same scroll-position, you can do
as follows:

        In your onStop()/onDestroyView()/onPause() whatever callback is more
appropriate, do this:
        Parcelable recyclerViewState =
recyclerView.getLayoutManager().onSaveInstanceState();

        And In your onStart()/onCreateView()/onResume() whatever callback is
more appropriate, do this:
recyclerView.getLayoutManager().onRestoreInstanceState(recyclerViewState);

        This way you can successfully keep your recyclerView's state in a
parcelable
        and restore it whenever you want.
        */
        if (recyclerView != null)
            ContentManager.recyclerViewState =

```

```

recyclerView.setLayoutManager().onSaveInstanceState();

        super.onStop();
    }

    @Override
    protected void onStart() {
        if (recyclerView != null && ContentManager.recyclerViewState != null) {
recyclerView.setLayoutManager().onRestoreInstanceState(ContentManager.recyclerViewSta
te);
        }
        super.onStart();
    }

    void updateNightMode(int mode){
        AppCompatDelegate.setDefaultNightMode(mode);
    }

    public void infoSendEmail(){
        Intent sendEmail = new Intent(Intent.ACTION_SENDTO);
        // sendEmail.setDataAndType(Uri.parse("mailto:"), "message/rfc822");
        sendEmail.setData(Uri.parse("mailto:"));
        sendEmail.putExtra(Intent.EXTRA_EMAIL, new String[]{CONSTS.APP_EMAIL});
        sendEmail.putExtra(Intent.EXTRA_SUBJECT, CONSTS.APP_EMAIL_SUBJECT);
        sendEmail.putExtra(Intent.EXTRA_TEXT,
getString(R.string.email_extra_text));
        startActivity(Intent.createChooser(sendEmail,
getString(R.string.send_email)));
    }
    public void infoChangeLanguage(){
        SharedPreferences x = getPreferences(getApplicationContext());
        String currentLanguage =
getResources().getConfiguration().getLocales().get(0).getLanguage();
        if (currentLanguage.equals(CONSTS.language_en)){
            x.edit().putString(CONSTS.PKEY_LANGUAGE,
CONSTS.language_ua).apply();
        }else{
            x.edit().putString(CONSTS.PKEY_LANGUAGE,
CONSTS.language_en).apply();
        }
        Intent refresh = new Intent(this, ActivityMain.class);
        finish();
        startActivity(refresh);
    }
    public void infoChangeDayNight(){
        SharedPreferences x = getPreferences(getApplicationContext());
        int newMode;
        if (AppCompatDelegate.getDefaultNightMode() ==
AppCompatDelegate.MODE_NIGHT_YES) {
            newMode = AppCompatDelegate.MODE_NIGHT_NO;
        }else {
            newMode = AppCompatDelegate.MODE_NIGHT_YES;
        }
        x.edit().putInt(CONSTS.PKEY_MODE, newMode).apply();
        updateNightMode(newMode);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Creates top menu options
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.top_menu, menu);
        searchItem = menu.findItem(R.id.action_top_search);
    }

```

```

        menuUpdating = menu.findItem(R.id.action_top_update);
        menuFavourite = menu.findItem(R.id.action_top_add_favourite);
        searchFiltersItem = menu.findItem(R.id.action_top_search_filters);

        MenuItem appInfo = menu.findItem(R.id.action_show_info);
        appInfo.setOnMenuItemClickListener(new
MenuItem.OnMenuItemClickListener() {
            @Override
            public boolean onOptionsItemSelected(MenuItem item) {
                // How to send emails and intent:
https://stackoverflow.com/a/15022153/8120181
                // How to change app day/night theme:
https://www.journaldev.com/19352/android-daynight-theme-night-mode
                new AlertDialog.Builder(ActivityMain.this)
                    .setTitle(R.string.app_info)
                    .setMessage(getString(R.string.app_info_text,
getString(R.string.app_name), getString(R.string.app_creator),
CONSTS.APP_EMAIL))
                    .setNeutralButton(R.string.send_feedback_email,
(dialog, which) -> infoSendEmail())
                    .setNegativeButton(R.string.change_language, (dialog,
which) -> infoChangeLanguage())
                    .setPositiveButton(R.string.change_day_night_theme,
(dialog, which) -> infoChangeDayNight())
                    .create()
                    .show();
                return true;
            }
        });

        menuUpdating.setOnMenuItemClickListener(new
MenuItem.OnMenuItemClickListener() {
            @Override
            public boolean onOptionsItemSelected(MenuItem item) {
                onUpdateIconClick();
                return true;
            }
        });

        searchFiltersItem.setOnMenuItemClickListener(new
MenuItem.OnMenuItemClickListener() {
            @Override
            public boolean onOptionsItemSelected(MenuItem item) {
                toggleFilterViewShowHide(!filtersAreVisible);
                searchFields.requestLayout();
                return true;
            }
        });

        menuFavourite.setOnMenuItemClickListener(new
MenuItem.OnMenuItemClickListener() {
            @Override
            public boolean onOptionsItemSelected(MenuItem item) {
                return false;
            }
        });

        searchView = (SearchView) searchItem.getActionView();
        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener()
{
            @Override
            public boolean onQueryTextSubmit(String query) {
                configs.setSearchingText(query);
                if (query.isEmpty()) {
                    configs.setFilterFunc(poll -> true);
                } else{
                    // false is set to prevent the filterFunc return all the

```

polls

```

        // and pass the search query
        configs.setFilterFunc(poll -> false);
    }
    onSearchClick(null);
    return true;
}
@Override
public boolean onQueryTextChange(String newText) {
    return false;
}
});
return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    toggleSearchItemsShowHide(false);
    menuFavourite.setVisible(false);
    return super.onPrepareOptionsMenu(menu);
}

private AlertDialog createSearchInDialog() {
    return new AlertDialog.Builder(this)
        .setTitle(R.string.search_search_in_title)
        .setCancelable(true)
        .setMultiChoiceItems(R.array.array_search_in,
SearchIn.getBooleans(configs.getSearchIn()), (dialog, which, isChecked) -> {
    // Lambda Listener: new
DialogInterface.OnMultiChoiceClickListener()
        configs.setSearchIn(Flags.toggleFlag(configs.searchIn,
SearchIn.getSearchIn(which), isChecked));
    })
        .setPositiveButton(R.string.confirm_button_ok, (dialog, which)
-> {
        // Lambda Listener: new DialogInterface.OnClickListener()
        updateTextSearchSearchIn();
        dialog.dismiss();
    }).create();
}

private AlertDialog createSortByDialog() {
    return new AlertDialog.Builder(this)
        .setTitle(R.string.search_sort_by_title)
        .setSingleChoiceItems(R.array.array_sort_by,
SortBy.get(configs.sortByItem), (dialog, which) -> {
        // Lambda Listener: new DialogInterface.OnClickListener()
        configs.setSortBy(SortBy.get(which));
        updateTextSearchSortBy();
        dialog.dismiss();
    }).create();
}

public void onSearchClick(View view){
    searchView.clearFocus();
    new PollFilter(configs).filterWithListener(ContentManager.getPolls(),
onFilterEndUsual);
    mHandler.postDelayed(() -> toggleFilterViewShowHide(false), 150);
}

private void onCaseChangeListener(CompoundButton buttonView, boolean
isChecked){

```

```

        if (buttonView.getId() == R.id.searchCaseSensitive) {
            configs.setIsCaseSensitive(isChecked);
        } else if (buttonView.getId() == R.id.searchReverse) {
            configs.setReverse(isChecked);
        }
    }

    private void updateTextSearchSortBy() {
        searchSortBy.setText(getString(R.string.search_conclusion_sort_by,
            getString(configs.getSortByItem().toStringID())));
    }

    private void updateTextSearchSearchIn() {
        List<Integer> stringIDs = SearchIn.getStringIDs(configs.getSearchIn());
        List<String> strings = new ArrayList<>();
        for (int i = 0; i < stringIDs.size(); i++)
            strings.add(getString(stringIDs.get(i)).toLowerCase());
        searchSearchIn.setText(getString(R.string.search_conclusion_search_in,
            String.join(", ", strings)));
    }

    private boolean searchItemsCanBeVisible() {
        // Returns if the current item in navigation view == searchTab
        return navigationView.getSelectedItemId() == R.id.navigation_search;
    }

    private void setTopMenuOnDependence() {
        // Sets default options to AppBar depending on current opened tab
        toggleSearchItemsShowHide(searchItemsCanBeVisible());
    }

    setAppBarSubtitle(navigationView.getMenu().findItem(navigationView.getSelectedItemId()
    ).getTitle().toString());
    }

    private void toggleTopMenuShowHide(boolean makeVisible, boolean
    forceMakeVisible) {
        if (forceMakeVisible || searchItemsCanBeVisible()) {
            toggleSearchItemsShowHide(makeVisible);
        }
    }

    private void toggleSearchItemsShowHide(boolean makeVisible) {
        searchItem.setVisible(makeVisible);
        searchFiltersItem.setVisible(makeVisible);
    }

    private void togglePollInfoShowHide(boolean makeVisible) {
        pollInfoIsShowing = makeVisible;
        // toggleNavigationMenuShowHide(!makeVisible);
        // Decided to return false in NavView OnNewItemSelected listener
        toggleTopMenuShowHide(!makeVisible, makeVisible);
    }

    public void toggleNavigationMenuShowHide(boolean makeVisible) {
        if (makeVisible) {
            navigationView.animate().alpha(1).withStartAction(() ->
            navigationView.setVisibility(View.VISIBLE)).setDuration(300).start();
        } else {
            // This Height Changes show the realTime changes via
            OnLayoutChanges listener
            navigationView.animate().alpha(0).withEndAction(() ->
            navigationView.setVisibility(View.GONE)).setDuration(300).start();
        }
    }
}

```

```

        private void toggleFilterViewShowHide(boolean makeVisible){
            searchFieldsParent.setVisibility(makeVisible? View.VISIBLE :
View.INVISIBLE);
            if (makeVisible){

searchFields.animate().translationY(0).alpha(1).setDuration(CONSTS.ANIMATION_DURATION
).withStartAction(() -> {
                // Start Action
                searchFields.setVisibility(View.VISIBLE);
                filtersAreVisible = true;
            }).start();
            }else{
                searchFields.animate().translationY(-
CONSTS.ANIMATION_TRANSLATION_Y).alpha(0).setDuration(CONSTS.ANIMATION_DURATION).withE
ndAction(() -> {
                    // End Action
                    searchFields.setVisibility(View.INVISIBLE);
                    filtersAreVisible = false;
                }).start();
            }
        }

        public void setAppBarTitle(String title){
            ActionBar bar = getSupportActionBar();
            if (bar != null) bar.setTitle(title);
        }

        public void setAppBarSubtitle(String subtitle){
            ActionBar bar = getSupportActionBar();
            if (bar != null) bar.setSubtitle(subtitle);
        }

        @Override
        public boolean onNavigationItemSelected(@NonNull MenuItem item) {
            if (pollInfoIsShowing) return false;
            if (pollsAreUpdating) {
                showToast(getString(R.string.wait_polls_are_updating));
                return false;
            }
            setAppBarSubtitle(item.getTitle().toString());
            boolean searchTabIsSelected = item.getItemId() ==
R.id.navigation_search;
            try {
                toggleSearchItemsShowHide(searchTabIsSelected);
            }catch (Exception e){
                e.printStackTrace();
                return false;
            }
            if (!searchTabIsSelected) {
                // Hide the top search line
                searchItem.collapseActionView();
            }

            if (navigationView.getSelectedItemId() != item.getItemId()) {
                // Make text invisible for a few moments If lastItem != new item
else save old
                itemsAreEmptyMsg.setVisibility(View.INVISIBLE);
            }
            if (item.getItemId() == R.id.navigation_poll_list){
                currentPairColorString = new Pair<>(R.color.backgroundFullPollList,
R.string.no_polls);
                ContentManager.setAllPolls(onFilterEndUsual);
            }
        }

```

```

        //
mainParent.setBackgroundResource(Utils.getAttrResId(ActivityMain.this,
R.attr.backgroundColor));
        }else if (item.getItemId() == R.id.navigation_favourite){
            currentPairColorString = new
Pair<>(R.color.backgroundFavouriteList, R.string.no_favourite_polls);
            ContentManager.setFavouritePolls(onFilterEndUsual);
        }else if (item.getItemId() == R.id.navigation_completed_poll_list){
            currentPairColorString = new
Pair<>(R.color.backgroundCompletedList, R.string.no_completed_polls);
            ContentManager.setCompletedPolls(onFilterEndUsual);
        }else if (item.getItemId() == R.id.navigation_search){
            currentPairColorString = new Pair<>(R.color.backgroundSearchList,
R.string.no_search_results_check_search_in_places);
            ContentManager.setAllPolls(onFilterEndUsual);
        }else{ return false; }
        return true;
    }
}

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.Collection;

public class Poll {
    public static boolean useListener = true;

    public static void setListener(OnActionListener<Poll> newListener) {
        listener = newListener;
    }

    public static OnActionListener<Poll> listener = (changeType, poll) -> {
        /* Implementation is empty to avoid NullPointerException in case null
listener */
    };

    // int Or Integer???
    // int null = 0 // int cannot be null and it auto casts to 0
    // Integer null = null; // Integer can be null
    // Integer is more effective for these project to avoid comparing errors
    // https://stackoverflow.com/questions/423704/which-one-to-use-int-or-integer/423718

    public final int intID;
    public final String name;
    public final String description;
    public final String author;
    public final int statCountViews;
    public final int statCountCompletes;

    private long completingTime = 0;
    private boolean isFavourite;
    private boolean isCompleted;
    private final ArrayList<Question> questions;

    public Poll(int intID, String name, String description) {
        this(intID, name, description, "Unknown");
    }
    public Poll(int intID, String name, String description, String author) {
        this(intID, name, description, author, 0, 0);
    }

```

```

    }
    public Poll(int intID, String name, String description, String author, int
statCountViews, int statCountCompletes) {
        this(intID, name, description, author, statCountViews,
statCountCompletes, null);
    }

    public Poll(int intID, String name, String description, String author,
        int statCountViews, int statCountCompletes, ArrayList<Question>
questions) {
        this.intID = intID;
        this.name = name;
        this.description = description;
        this.author = author;
        this.statCountViews = statCountViews;
        this.statCountCompletes = statCountCompletes;
        this.questions = new ArrayList<>();
        setQuestions(questions);
    }

    public long getCompletingTime() { return completingTime; }
    public void setCompletingTime(long completingTime) {
        if (this.completingTime == 0) {
            this.completingTime = completingTime;
        }
    }

    listener.onAction(OnActionListener.Actions.POLL_SET_COMPLETING_TIME, this,
useListener);
    }
    }
    public boolean isFavourite() {
        return isFavourite;
    }
    public boolean isCompleted() {
        return isCompleted;
    }
    }

    private void setCompleted(boolean newCompleted) {
        if (isCompleted != newCompleted) {
            isCompleted = newCompleted;
            listener.onAction(OnActionListener.Actions.POLL_COMPLETE_STATE,
this, useListener);
        }
    }
    public void setFavourite(boolean newFavourite) {
        if (isFavourite != newFavourite) {
            isFavourite = newFavourite;
            listener.onAction(OnActionListener.Actions.POLL_CHANGE_FAVOURITE,
this, useListener);
        }
    }
    }

    public int getUnAnsweredQuestionIndex(){
        for (int i = 0; i < questions.size(); i++) {
            if (!questions.get(i).isAnswered())
                return i;
        }
        return -1;
    }
    public boolean hasUnAnsweredQuestions(){
        return getUnAnsweredQuestionIndex() != -1;
    }
    }

    public void completePoll(long completingTime) throws Exception {

```

```

        if (hasUnAnsweredQuestions())
            throw new Exception("Poll has unanswered questions");
        setCompleted(true);
        setCompletingTime(completingTime);
    }

    public int getQuestionsCount() {
        return questions.size();
    }

    public ArrayList<Question> getQuestions() {
        return questions;
    }

    public Question getQuestion(int index){ return questions.get(index); }

    public Question getQuestionByID(int questionID) {
        for (Question question : questions)
            if (question.intID == questionID)
                return question;
        return null;
    }

    public Question getLastQuestion() {
        return questions.isEmpty() ? null : questions.get(questions.size() -
1);
    }

    public void clearQuestions() {
        listener.onAction(OnActionListener.Actions.POLL_CLEAR_QUESTIONS, this,
useListener);
        questions.clear();
    }

    public void setQuestions(Collection<Question> questions) {
        clearQuestions();
        addQuestions(questions);
    }

    public void addQuestion(Question question) {
        question.provider.setParent(() -> Poll.this);
        questions.add(question);
        listener.onAction(OnActionListener.Actions.POLL_ADD_NEW_QUESTION, this,
useListener);
    }

    public void addQuestions(Collection<Question> questions) {
        if (questions == null) return;
        for (Question question : questions)
            addQuestion(question);
    }

    public Object getObjectForComparing(SortBy sortBy) {
        switch (sortBy) {
            case NAME:
                return this.name;
            case AUTHOR:
                return this.author;
            case QUESTIONS_COUNT:
                return this.questions.size();
            case VIEWS_COUNT:
                return this.statCountViews;
            case COMPLETES_COUNT:

```

```

        return this.statCountCompletes;
    default:
    case ID:
    case NO_SORT:
        return this.intID;
    }
}

    public boolean containsSearchingText(String searchingText, boolean
isCaseSensitive,
                                     boolean searchInName, boolean
searchInDescription,
                                     boolean searchInAuthor, boolean
searchInQuestionText,
                                     boolean searchInQuestionAnswerText) {

        // Each comparison is placed on every line to avoid unnecessary
searches
        if (searchInName && Utils.strContains(name, searchingText,
isCaseSensitive))
            return true;
        if (searchInDescription && Utils.strContains(description,
searchingText, isCaseSensitive))
            return true;
        if (searchInAuthor && Utils.strContains(author, searchingText,
isCaseSensitive))
            return true;
        if (searchInQuestionText)
            for (Question question : questions)
                if (question.containsSearchingText(searchingText,
isCaseSensitive, searchInQuestionAnswerText))
                    return true;
        return false;
    }

    @Override
    public String toString() {
        return String.format("Poll{intID=%d, name='%s', description='%s',
author='%s', isFavourite=%b, isCompleted=%b, statCountViews=%d,
statCountCompletes=%d, questions=%s",
                             intID, name, description,author, isFavourite, isCompleted,
statCountViews, statCountCompletes, questions);
    }

    JSONObject toJSON() throws JSONException {
        JSONObject jsonObject = new JSONObject()
            .put(CONSTS.JKEY_POLL_ID, intID)
            .put(CONSTS.JKEY_POLL_NAME, name)
            .put(CONSTS.JKEY_POLL_DESCRIPTION, description)
            .put(CONSTS.JKEY_POLL_AUTHOR, author)
            .put(CONSTS.JKEY_POLL_STAT_COUNT_VIEWS, statCountViews)
            .put(CONSTS.JKEY_POLL_STAT_COUNT_COMPLETES,
statCountCompletes);

        JSONArray jsonArray = new JSONArray();
        for (Question q : questions) { jsonArray.put(q.toJSON()); }
        jsonObject.put(CONSTS.JKEY_POLL_QUESTIONS_LIST, jsonArray);
        return jsonObject;
    }
}

package com.worldwide.polls;

import android.os.AsyncTask;

```

```

import java.util.ArrayList;
import java.util.Arrays;

public class PollFilter extends AsyncTask<Poll, Void, ArrayList<Poll>> {

    public final FilterConfigs filterConfigs;
    private OnFilterEnd onFilterEndListener;

    @Override
    protected void onPostExecute(ArrayList<Poll> polls) {
        this.onFilterEndListener.onFilterProcessEnd(polls);
    }

    @Override
    protected ArrayList<Poll> doInBackground(Poll... polls) {
        return filter(new ArrayList<>(Arrays.asList(polls)));
    }

    public static boolean compare(Poll obj1, Poll obj2, SortBy sortByItem) {
        Object comparingObject1 = obj1.getObjectForComparing(sortByItem);
        Object comparingObject2 = obj2.getObjectForComparing(sortByItem);
        if (Utils.isInteger(comparingObject1) &&
Utils.isInteger(comparingObject2)) {
            return compareInts((Integer) comparingObject1, (Integer)
comparingObject2);
        } else if (Utils.isString(comparingObject1) &&
Utils.isString(comparingObject2)) {
            return compareStrings((String) comparingObject1, (String)
comparingObject2);
        }
        return false;
    }

    public static boolean compareInts(Integer obj1Value, Integer obj2Value) {
        return obj1Value > obj2Value;
    }

    public static boolean compareStrings(String obj1Value, String obj2Value) {
        obj1Value = obj1Value.toLowerCase();
        obj2Value = obj2Value.toLowerCase();
        for (int i = 0; i < Math.min(obj1Value.length(), obj2Value.length());
i++)
            if (obj1Value.charAt(i) != obj2Value.charAt(i))
                return obj1Value.charAt(i) < obj2Value.charAt(i);
        return obj1Value.length() < obj2Value.length();
    }

    public PollFilter() { filterConfigs = new FilterConfigs(); }
    public PollFilter(FilterConfigs configs){ filterConfigs = configs; }
    public PollFilter(String searchingText){ filterConfigs = new
FilterConfigs(searchingText); }
    public PollFilter(Filter<Poll> filterFunc) { filterConfigs = new
FilterConfigs(filterFunc); }
    public PollFilter(Filter<Poll> filterFunc, String searchingText) {
filterConfigs = new FilterConfigs(filterFunc, searchingText); }
    public PollFilter(Filter<Poll> filterFunc, String searchingText, boolean
isCaseSensitive, boolean reverse, byte searchInFlags, SortBy sortByItem) {
        filterConfigs = new FilterConfigs(filterFunc, searchingText,
isCaseSensitive, reverse, searchInFlags, sortByItem);
    }

    public void filterWithListener(ArrayList<Poll> items, OnFilterEnd

```

```

onFilterEndListener){
    // Maybe it will allow to run process in another thread and get the
    result into listener...
    if (onFilterEndListener != null) {
        this.onFilterEndListener = onFilterEndListener;
        this.execute(items.toArray(new Poll[0]));
    }
}

public ArrayList<Poll> filter(ArrayList<Poll> polls){
    if (filterConfigs.filterFunc == null)
        return polls;
    return reverse(sortItems(filterItems(polls, filterConfigs),
filterConfigs.sortByItem), filterConfigs.reverse);
}

public static ArrayList<Poll> filterItems(ArrayList<Poll> polls,
FilterConfigs configs) {
    if (polls == null) return polls; // || polls.size() < 2
    boolean searchInName = Flags.hasFlag(configs.searchIn, SearchIn.NAME);
    boolean searchInDescription = Flags.hasFlag(configs.searchIn,
SearchIn.DESCRPTION);
    boolean searchInAuthor = Flags.hasFlag(configs.searchIn,
SearchIn.AUTHOR);
    boolean searchInQuestionText = Flags.hasFlag(configs.searchIn,
SearchIn.QUESTIONS);
    boolean searchInQuestionAnswerText = Flags.hasFlag(configs.searchIn,
SearchIn.ANSWERS);
    boolean hasMatching;
    for (int i = 0; i < polls.size(); i++) {
        // Each element removed only when it has not passed any continue
statements
        // Each 'continue' elements means the matching of searchingText and
values
        // and prevents deleting from arrayList
        Poll poll = polls.get(i);
        if (configs.filterFunc.filter(poll)){
            continue;
        }
        if (configs.searchingText != null &&
!configs.searchingText.isEmpty()) {
            hasMatching = poll.containsSearchingText(configs.searchingText,
configs.isCaseSensitive,
                searchInName, searchInDescription, searchInAuthor,
                searchInQuestionText, searchInQuestionAnswerText);
            if (hasMatching)
                continue;
        }
        polls.remove(i--);
    }
    return polls;
}

public static ArrayList<Poll> sortItems(ArrayList<Poll> polls, SortBy
sortByItem){
    if (polls == null || sortByItem == SortBy.NO_SORT) return polls;
    return sortGnome(polls, sortByItem);
}

public static ArrayList<Poll> sortGnome(ArrayList<Poll> polls, SortBy
sortByItem){
    int i = 0;
    while (i < polls.size()){
        // If App will freeze or crashes - check loop

```

```

        // Log.d("sortGnome:", polls.toString());
        if (i == 0 || !compare(polls.get(i-1), polls.get(i), sortByItem)){
            i++;
            continue;
        }
        Poll tmp = polls.get(i);
        polls.set(i, polls.get(i-1));
        polls.set(i-1, tmp);
        i--;
    }
    return polls;
}

public static ArrayList<Poll> reverse(ArrayList<Poll> items, boolean
reverse){
    if (items == null || items.isEmpty()) return items;
    if (reverse) {
        int itemsSizeMaxIndex = items.size() - 1;
        for (int i = 0; i < items.size() / 2; i++) {
            Poll tmp = items.get(i);
            items.set(i, items.get(itemsSizeMaxIndex - i));
            items.set(itemsSizeMaxIndex - i, tmp);
        }
    }
    return items;
}

}

package com.worldwide.polls;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.Collection;

public class Question {
    public static boolean useListener = true;
    public static OnActionListener<Question> listener = (changeType, answer) -> {
        /* Implementation is empty to avoid NullPointerException in case null
listener */
    };

    public static void setListener(OnActionListener<Question> newListener) {
        listener = newListener;
    }

    // public final int countMultiSelections;
    public final String questionText;
    public final int intID;
    public final int countCompletes;
    public final Provider<Poll> provider = new Provider<>();
    private final ArrayList<Answer> answers;

    public Question(int intID, String questionText) {
        this(intID, questionText, 0);
    }

    public Question(int intID, String questionText, int countCompletes) {
        this(intID, questionText, countCompletes, new ArrayList<>());
    }
}

```

```

    public Question(int intID, String questionText, int countCompletes,
Collection<Answer> answers) {
        this(intID, questionText, countCompletes, answers, -1);
    }

    public Question(int intID, String questionText, int countCompletes,
Collection<Answer> answers, int chosenAnswerID) {
        this.intID = intID;
        this.questionText = questionText;
        this.countCompletes = countCompletes;
        this.answers = new ArrayList<>();

        setAnswers(answers);
        setChosenAnswer(chosenAnswerID);
    }

    public ArrayList<Answer> getAnswers() {
        return answers;
    }

    public int getAnswersCount() {
        return answers.size();
    }

    public void setAnswers(Collection<Answer> answers) {
        clearAnswers();
        addAnswers(answers);
    }

    public boolean canMakeQuestionCompleted() {
        return isAnswered();
    }

    public boolean isAnswered() {
        return getChosenAnswerID() != -1;
    }

    public void completeQuestion() {
        if (canMakeQuestionCompleted())
            listener.onAction(OnActionListener.Actions.QUESTION_COMPLETE_STATE, this,
useListener);
    }

    public void completeQuestionWithError() throws Exception {
        if (!canMakeQuestionCompleted())
            throw new Exception("Question is not answered yet!");
        listener.onAction(OnActionListener.Actions.QUESTION_COMPLETE_STATE, this,
useListener);
    }

    public void clearAnswers() {
        listener.onAction(OnActionListener.Actions.QUESTION_CLEAR_ANSWERS, this,
useListener);
        answers.clear();
    }

    public void addAnswers(Collection<Answer> answers) {
        for (Answer answer : answers) addAnswer(answer);
    }

    public void addAnswer(Answer answer) {
        answer.provider.setParent(() -> Question.this);
        answers.add(answer);
    }

```

```

        if (answer.isChosen())
            this.setChosenAnswer(answer);
        listener.onAction(OnActionListener.Actions.QUESTION_ADD_NEW_ANSWER, this,
useListener);
    }

    public Answer getAnswer(int index) {
        return answers.get(index);
    }

    public Answer getLastAnswer() {
        return answers.isEmpty() ? null : answers.get(answers.size() - 1);
    }

    public Answer getAnswerByID(int answerID) {
        for (Answer answer : answers)
            if (answer.intID == answerID)
                return answer;
        return null;
    }

    public Answer getChosenAnswer() {
        for (Answer answer : answers)
            if (answer.isChosen())
                return answer;
        return null;
    }

    public int getChosenAnswerID() {
        Answer answer = getChosenAnswer();
        if (answer == null)
            return -1;
        return answer.intID;
    }

    public void setChosenAnswer(int answerID) {
        if (answerID == -1) return;
        for (int i = 0; i < answers.size(); i++) {
            if (answers.get(i).intID == answerID) {
                setChosenAnswer(answers.get(i));
            }
        }
    }

    public void setChosenAnswer(Answer chosenAnswer) {
        if (answers.contains(chosenAnswer)) {
            for (Answer answer : answers)
                if (answer.equals(chosenAnswer))
                    answer.setIsChosen(true);
        }
    }

    public void clearChosenAnswer() {
        for (Answer answer : answers) answer.setIsChosen(false);
    }

    public boolean containsSearchingText(String searchingText, boolean
isCaseSensitive, boolean searchInQuestionAnswerText) {
        if (Utils.strContains(questionText, searchingText, isCaseSensitive))
            return true;
        if (searchInQuestionAnswerText)
            for (Answer answer : answers)
                if (answer.containsSearchingText(searchingText, isCaseSensitive))
                    return true;
    }

```

```

        return false;
    }

    @Override
    public String toString() {
        return String.format("Question{intID=%d, questionText='%s',
countCompletes=%d, answers=%s",
            intID, questionText, countCompletes, answers);
    }

    JSONObject toJSON() throws JSONException {
        JSONObject jsonObject = new JSONObject();
        jsonObject.put(CONSTS.JKEY_QUESTION_ID, intID);
        jsonObject.put(CONSTS.JKEY_QUESTION_TEXT, questionText);
        jsonObject.put(CONSTS.JKEY_QUESTION_STAT_COUNT_COMPLETES, countCompletes);

        JSONArray jsonArray = new JSONArray();
        for (Answer a : answers) {
            jsonArray.put(a.toJSON());
        }
        jsonObject.put(CONSTS.JKEY_QUESTION_ANSWERS_LIST, jsonArray);
        return jsonObject;
    }
}

package com.worldwide.polls;

import java.util.ArrayList;
import java.util.List;

public class SearchIn {

    static final byte NO_WHERE = 0;
    static final byte NAME = 1 << 1;
    static final byte DESCRIPTION = 1 << 2;
    static final byte AUTHOR = 1 << 3;
    static final byte QUESTIONS = 1 << 4;
    static final byte ANSWERS = 1 << 5;
    static final byte ALL = NAME | DESCRIPTION | AUTHOR | QUESTIONS | ANSWERS;
    static final int COUNT = 5; // NAME, DESCRIPTION, AUTHOR, QUESTIONS, ANSWERS

    private SearchIn() {
    }

    public static byte getSearchIn(int index) {
        switch (index) {
            case 0:
                return NAME;
            case 1:
                return DESCRIPTION;
            case 2:
                return AUTHOR;
            case 3:
                return QUESTIONS;
            case 4:
                return ANSWERS;
            default:
                return NO_WHERE;
        }
    }

    public static List<Integer> getStringIDs(byte flags) {
        List<Integer> integers = new ArrayList<>();
    }
}

```

```

        if (Flags.hasFlag(flags, NAME)) integers.add(R.string.search_in_place_name);
        if (Flags.hasFlag(flags, DESCRIPTION))
integers.add(R.string.search_in_place_description);
        if (Flags.hasFlag(flags, AUTHOR))
integers.add(R.string.search_in_place_author);
        if (Flags.hasFlag(flags, QUESTIONS))
integers.add(R.string.search_in_place_questions);
        if (Flags.hasFlag(flags, ANSWERS))
integers.add(R.string.search_in_place_answers);
        if (integers.isEmpty())
            integers.add(R.string.search_in_not_selected);
        return integers;
    }

    public static boolean[] getBooleans(byte flags) {
        boolean[] states = new boolean[COUNT];
        for (int i = 0; i < COUNT; i++) {
            states[i] = Flags.hasFlag(flags, getSearchIn(i));
        }
        return states;
    }
}

```

ДОДАТОК В
Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Кафедра програмних засобів

«Розроблення мобільного застосунку для
підтвердження якості процесів і продуктів
міжнародних навчальних проектів.

Mobile application development for quality assurance
of the processes and products in international
educational projects»

Виконав
ст.гр. КНТ-127

Куковинець О.Є

Керівник
Професор, к.т.н.

Табунщик Г.В.

Рисунок В.1 – Слайд 1

Мета роботи, об'єкт та предмет дослідження

- мета роботи - розробка Android - застосунку із можливостями перегляду, пошуку та проходження опитувань, зберігання результатів та статистики кожного з них, з простим та приємним для користувача інтерфейсом.
- об'єкт дослідження – програмні засоби, які призначені для проведення опитувань;
- предмет дослідження – застосунок призначений для проведення опитувань;
- галузь використання – проведення опитувань з метою дослідження думки людей, оцінки будь-яких процесів.

Рисунок В.2 – Слайд 2

Задачі роботи

Для досягнення мети в дипломній кваліфікаційній роботі потрібно вирішити наступні задачі:

- проаналізувати предметну область;
- описати функціональні вимоги розроблюваного застосунку;
- аргументувати вибір технологій, середовища розробки та мови програмування;
- розробити мобільний застосунок для проведення опитувань;
- виконати тестування застосунку;

3

Рисунок В.3 – Слайд 3

Порівняння аналогів

Назва застосунку	Інтерфейс	Зручність використання	Дизайн	Інтуїтивна зрозумілість	Стабільність роботи
<u>Pollscape</u>	+	+	-	+	+
Poll For All	-	+	+	-	-
<u>Pollie</u>	+	+	+	+	-
<u>Polls - WorldWide</u>	+	+	+	+	+

4

Рисунок В.4 – Слайд 4

Функції системи

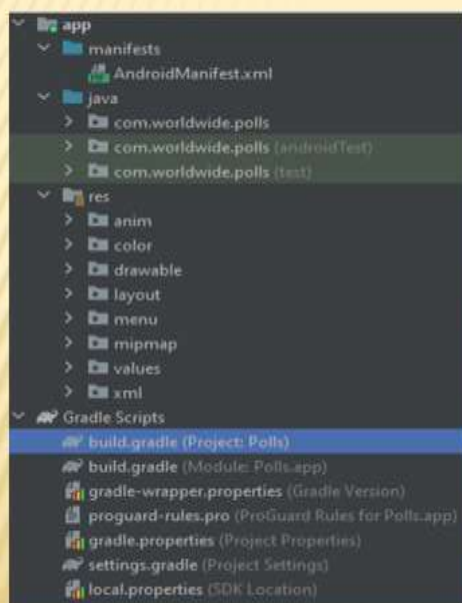
Основні функції, які виконує застосунок:

- перегляд та проходження опитувань;
- пошук та сортування опитувань;
- зберігання результатів проходження;
- перегляд статистики проходжень опитувань;
- додавання обраних та зберігання пройдених опитувань.

5

Рисунок В.5 – Слайд 5

Схема структури застосунку



До структури застосунку входять:

- файл маніфесту;
- директорія Java;
- директорія res;
- файли Gradle Scripts:

6

Рисунок В.6 – Слайд 6

Збереження даних та файлів

Имя	Тип
Таблицы (4)	
Answers	
answerID	integer
questionID	integer
pollID	integer
answerText	text
statSelectionsCount	integer
Polls	
pollID	integer
name	text
description	text
author	text
statCountViews	integer
statCountCompletes	integer
Questions	
questionID	integer
pollID	integer
questionText	text
countCompletes	integer
sqlite_sequence	
Индексы (0)	
Представления (0)	
Триггеры (0)	

Рис. 1 – БД Polls.db на сервері

Таблицы (3)	
PollQuestions	
pollID	integer
PollQuestionID	integer
PollAnswerID	integer
ResultSend	integer
Polls	
pollID	INTEGER
IsFavourite	boolean
IsCompleted	boolean
PollCompletingTime	integer
android_metadata	
Индексы (0)	
Представления (0)	
Триггеры (0)	

Рис. 2 – SQLite БД PollData

7

Рисунок В.7 – Слайд 7

Збереження даних та файлів

```
{
  "id": 1,
  "name": "First Poll",
  "description": "This is my first poll",
  "author": "Olga",
  "pollCountViews": 12000,
  "pollCountCompletes": 1234,
  "questions": [
    {
      "id": 11,
      "text": "First Question",
      "questionCountCompletes": 100,
      "answers": [
        {
          "id": 111,
          "text": "First Answer First Question END",
          "answerSelectionsCount": 0,
          "answerSelectionsPercent": 0
        },
        {
          "id": 112,
          "text": "Second Answer First Question",
          "answerSelectionsCount": 0,
          "answerSelectionsPercent": 0
        },
        {
          "id": 113,
          "text": "Third Answer First Question",
          "answerSelectionsCount": 0,
          "answerSelectionsPercent": 0
        }
      ]
    }
  ]
},
{
  "id": 12,
  "text": "Second Question Second Question Second",
  "questionCountCompletes": 0,
  "answers": [
    {
      "id": 121,
      "text": "First Answer Second Question",
      "answerSelectionsCount": 0,
      "answerSelectionsPercent": 0
    },
    {
      "id": 122,
      "text": "Second Answer Second Question",
      "answerSelectionsCount": 0,
      "answerSelectionsPercent": 0
    }
  ]
}
```

Рис. 3 – JSON файл з опитуванням

com.worldwide.polls	
cache	
code_cache	
databases	
PollData	
files	
JSONPolls	
Poll_1.json	
Poll_2.json	
Poll_3.json	
Poll_4.json	

Рис. 4 – Структура даних на девайсі

8

Рисунок В.8 – Слайд 8

Порівняння середовищ розробки

Середовище розробки	Мова програмування	Мобільні платформи для розробки	Платня
Android Studio	Java , C/C++ , Delphi , Kotlin	Android	ні
IntelliJ IDEA	Java , C/C++ , PHP , Ruby , Python , SQL , Rust , Go , XML	Android , Windows phone	так
Eclipse IDE	Java , C/C++ , PHP	Android , IOS	ні
Intel XDK	HTML	всі	так
XCode	C++ , Java , Python	IOS , OS X	ні

9

Рисунок В.9 – Слайд 9

Вибір мови програмування

Для розробки [застосунку](#) обрана мова програмування [Java](#).

Переваги [Java](#):

- найпопулярніша мова програмування для мобільної розробки під ОС [Android](#):
- офіційна мова для розробки [Android](#) - [застосунків](#) і підтримується [Android Studio](#):
- підтримується компанією [Google](#) і більшість додатків в [Google Play](#) побудовані саме на ній.

10

Рисунок В.10 – Слайд 10

Інтерфейс мобільного застосунку

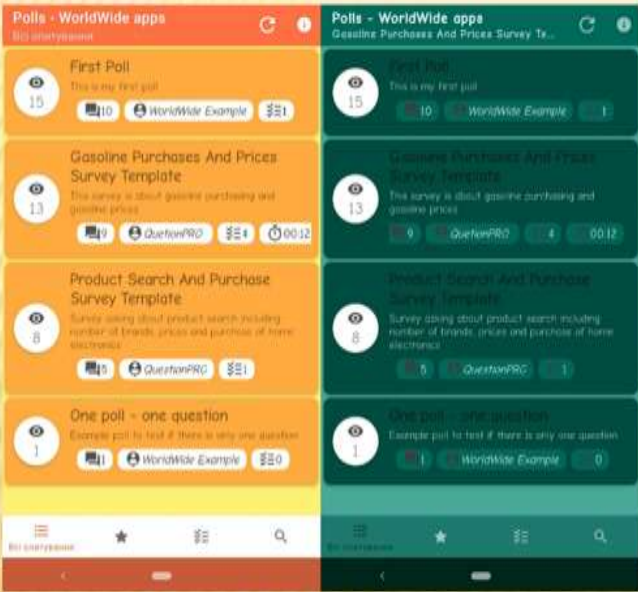


Рис. 5 – Головний екран застосунку



Рис. 6 – Вікно інформації опитування

11

Рисунок В.11 – Слайд 11

Інтерфейс мобільного застосунку

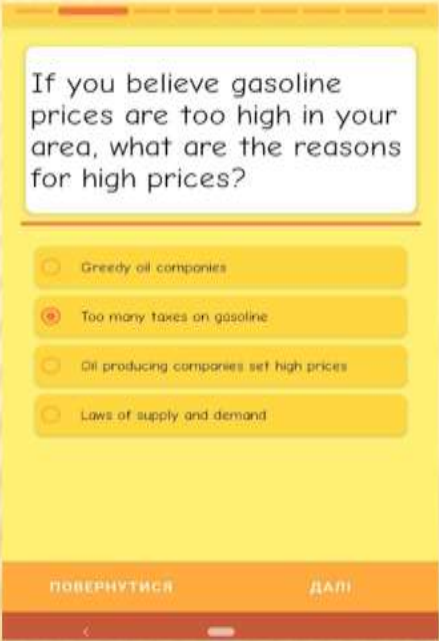


Рис. 7 – Форма проходження опитування



Рис. 8 – Форма перегляду статистики

12

Рисунок В.12 – Слайд 12

Інтерфейс мобільного застосунку

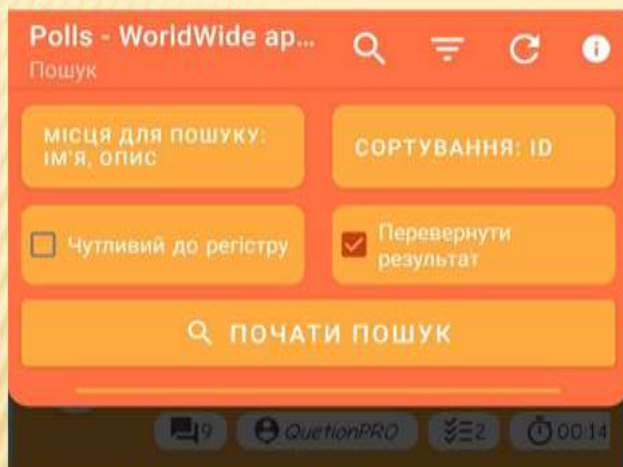


Рис. 9 – Вікно фільтрації опитувань



Рис. 10 – Вікно часу проходження

13

Рисунок В.13 – Слайд 13

Тестування застосунку

Перевірка	Телефон – Realme 6 Pro Версія Android - 10	Телефон - Xiaomi Note 10 Версія Android 11	Планшет – LenovoI Tab M10 Plus Версія Android – 9
Завантаження застосунку	пройдено	пройдено	пройдено
Адаптивність інтерфейсу	пройдено	пройдено	пройдено
Завантаження опитувань	пройдено	пройдено	пройдено
Збереження результатів проходження	пройдено	пройдено	пройдено
Збереження пройдених та обраних опитувань	пройдено	пройдено	пройдено
Пошук та сортування опитувань	пройдено	пройдено	пройдено
Переходи між формами	пройдено	пройдено	пройдено

14

Рисунок В.14 – Слайд 14

Висновки

Під час розробки застосунку були вивчені та застосовані:

- робота із формами та вікнами;
- обробка натискань на кнопки;
- робота із JSON файлами та базою даних;
- запити до серверу;
- розробка інтуїтивного інтерфейсу користувача.

15

Рисунок В.15 – Слайд 15

Висновки

Шляхи удосконалення застосунку:

- адміністративна частина застосунку, яка дозволить створювати опитування та відправляти їх до БД на сервері;
- множинний вибір відповідей на запитання;
- підтримка малюнків у запитаннях;
- облікові записи користувачів;
- доопрацювання інтерфейсу для потреб користувачів.

16

Рисунок В.16 – Слайд 16

Дякую за увагу!

17

Рисунок В.17 – Слайд 17