Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»

Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Розробка мобільного додатку керування аудіофайлами з урахуванням музичного смаку користувача»
за освітньою програмою: «Інженерія програмного забезпечення»
зі спеціальності: «121 Інженерія програмного забезпечення»
Виконав: студент групи «ПЗ1812»

_____ /Максим РІЗНИЧЕНКО/
(підпис студента) (Ім'я ПРІЗВИЩЕ)

Керівник: _____ /доц. Олександр ІВАНОВ/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: _____ /доц. Олена КУРОП'ЯТНИК/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Дніпро – 2022 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»

Department «Computer information technology»

Explanatory Note
to Bachelor's Thesis

on the topic: «Development of mobile application of audio files management which includes consideration user`s musical taste»
according to educational curriculum «Software engineering»
in the Speciality: «121 Software engineering»

Done by the student of the group PZ1812: /Maksym RIZNYCHENKO/

Scientific Supervisor: /Oleksandr IVANOV/

Normative controller: /Olena KUROPIATNYK/

Dnipro – 2022

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Факультет «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: бакалавр
Освітня програма: «Інженерія програмного забезпечення»
Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____ /Вадим ГОРЯЧКІН/
(підпис)
Дата _____

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра
студенту Різниченку Максиму Олександровичу
1. Тема роботи: «Розробка мобільного додатку керування аудіофайлами з урахуванням музичного смаку користувача»
Керівник роботи: Іванов Олександр Петрович, доцент
затверджені наказом № 77 ст від 08.12.2021
2. Строк подання студентом роботи: 14.06.2022 р.
3. Вихідні дані до роботи: _____

4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):
4.1 Аналітична частина: опис аналогів, збір вимог до програмного забезпечення.
4.2 Основна частина: зовнішнє та внутрішнє проектування.
4.3 Розробка програми : вибір мови програмування, опис алгоритмічних структур.
4.4 Тестування та налагодження: тестування білою скринькою, тестування чорною скринькою, налагодження програми.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): діаграма бізнес-процесів, діаграма прецедентів, схема екранів додатку та їх взаємодії, діаграма класів, ескізи екранів, діаграма діяльності, діаграма послідовностей, блок-схеми.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вступ	20.04.2022-02.05.2022	
2	Збір та аналіз вимог	02.05.2022-18.05.2022	
3	Проектування	18.05.2022-25.05.2022	
4	Розробка програми	25.05.2022-05.06.2022	
5	Тестування та налагодження	05.06.2022-08.06.2022	
6	Аналіз результатів та оформлення пояснювальної записки	08.06.2022-13.06.2022	
7	Подання кваліфікаційної роботи до кафедри	14.06.2022	
8	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	22.06.2022	

Студент _____ Максим РІЗНИЧЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ доц. Олександр ІВАНОВ
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка складається з вступу, переліку умовних познак, символів, скорочень і термінів, 4 розділів, загальних висновків та рекомендацій:

- вступ – в даному розділі описується сутність розробки, її актуальність. Складається з 2 сторінок;
- розділ 1 збір та аналіз вимог – у цьому розділи описуються аналоги програми, попередній збір інформації, її аналіз. Складається з 12 сторінок;
- розділ 2 проектування – у цьому розділі описуються вхідні і вихідні дані, формування задачі, розробка проекту, проектування інтерфейсу користувача, ескізи екранів, аналіз проекту, проектування динаміки системи, моделювання примітивних типів, описується структура сховища. Складається з 32 сторінок;
- розділ 3 розробка програми – включає в себе вибір мови програмування, опис алгоритмічних структур. Складається з 5 сторінок;
- розділ 4 тестування та налагодження – включає в себе тестування проекту чорною та білою скринькою, аналіз та виправлення помилок. Складається з 21 сторінки;
- загальні висновки та рекомендації. Складається з 1 сторінки;
- список літератури – включає в себе бібліографічний список використаної літератури. Складає 2 сторінки;
- додатки – містить текст програми.

Кількість таблиць: 13 штук.

Кількість рисунків: 35 штук.

Ключові слова: аудіофайли, метадані, MP3, мобільний застосунок, рекомендації, аналіз смаку користувача, система, розробка, користувач.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАК, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Матеріальний — принципи дизайну сайтів, програмного забезпечення і застосунків, а також правила дизайну інтерфейсів для операційної системи Android від компанії Google.

Мобільний застосунок – програмне забезпечення, призначене для роботи на смартфонах та інших мобільних пристроях.

Метадані – інформація, що використовується для опису даних, що містяться у самому файлі.

Операційна система (ОС) – базовий комплекс програм, що виконує керування апаратною складовою пристрою.

Стрімінг – мультимедіа, яке користувач безперервно отримує від провайдера потокової трасляції.

ПЗ – програмне забезпечення.

Валідація – процес підтвердження відповідності.

ООП – об'єктно-орієнтоване програмування.

MP3 – формат файлу для зберігання аудіоінформації.

ВСТУП

Потреби користувача сьогодення диктують спрощення та підвищення зручності використання інформації для збереження так необхідного вільного часу. Саме тому попит на використання комп'ютерних версій додатків швидко падає. Обираючи платформу для написання додатку враховувались вимоги сучасної людини. Зараз майже кожний має у кишені смартфон з встановленими додатками різних категорій. Сьогодні існують різні платформи для смартфонів. Серед них є: BlackBerry, Symbian OS, Windows Mobile, Android, IOS. Було проведено дослідження, в результаті якого найбільш популярною платформою серед користувачів смартфонів є платформа Android, бо вона э однією з найбільш простих платформ.

Мільйони програмних продуктів зараз переписуються програмістами для адаптації комп'ютерних версій на платформи смартфонів. Більшість сучасних смартфонів завдяки оптимізації не поступаються функціоналом комп'ютеру, тож розробка додатків на мобільну платформу Android – це вимога часу.

Кожен смартфон має доступ до мережі Інтернет, за допомогою якого людина з легкістю отримує інформацію різного виду, в тому числі і аудіофайли. Через це виникла гостра потреба у створенні зручного мобільного додатку керування аудіофайлами прямо у кишені кожного користувача.

Об'єктом дипломної роботи є розвиток у сфері керування аудіофайлами на пристрої платформи Android.

Предметом дипломної роботи є мобільний додаток керування аудіофайлами на пристрої користувача.

Метою дипломної роботи є розробка мобільного додатку керування аудіофайлами з урахуванням музичного смаку користувача на базі платформи Android.

Задачі та мета дослідження:

1) проаналізувати аналогічні мобільні додатки керування аудіофайлами наявні у мережі Інтернет;

2) дослідити способи та алгоритми аналізу аудіофайлів для рекомендацій користувачу;
3) розробити мобільний додаток під керуванням операційної системи Android;
4) провести тестування та налагодження мобільного додатку керування аудіофайлами з урахуванням музичного смаку користувача.

РОЗДІЛ 1 ЗБІР ТА АНАЛІЗ ВИМОГ

1.1 Опис аналогів

1.1.1 Spotify

Spotify – стрімінговий інтернет-сервіс із музикою, що наразі доступний у 92 країнах світу. Цей музичний сервіс дозволяє прослуховувати музичні композиції відомих лейблів. Він надає можливість слухати музику без її завантаження на пристрій.

Spotify був запущений у Швеції і зараз має найбільшу у світі кількість користувачів. На цей сервіс припадає 36% світового ринку, що робить його найпопулярнішим музичним стрімінговим сервісом у світі.

За бажанням сервіс надає можливість оформити одну з наступних підписок:

- індивідуальна преміум підписка – 4,99 доларів;
- сімейна підписка на 6 осіб – 7,99 доларів;
- студентська підписка – 2,49 доларів;
- підписка «duo» з можливістю розділяти плейлисти на двох – 6,49 доларів;

Наявний також безкоштовний пробний період користування з підпискою преміум – один місяць.

Також сервісом можливо користуватися безкоштовно, але з деякими обмеженнями, погіршеною якістю аудіофайлів та достатньо великою кількістю голосової реклами.

Головною особливістю Spotify є те, що він працює у парі з алгоритмами персоналізованих музичних рекомендацій на основі смаків користувача, що працює на основі штучного інтелекту й підлаштовується під прослуховування, збереження та лайки [1]. Сервіс також має деякі категорії рекомендацій для слухача:

- spotify Radio – плейлисти на основі пісні, виконавця;
- release Radar – релізи композицій від артистів, які подобаються слухачу;
- daily Mix – поєднання улюбленої музики користувача з треками, які потенційно відповідають його смакам.

Програмні алгоритми Spotify надзвичайно швидко визначають музичні смаки окремих користувачів. Коли користувачу потрібно знайти нову музику, сервіс не зациклюється на відомих та популярних артистах постійно рекомендуючи одні і ті самі композиції та відомі імена. Компанія інвестує велику кількість грошей у нейромережі та штучний інтелект, саме тому алгоритми сервісу подобаються мільйонам користувачів по всьому світу [2].

15 липня 2020 року Spotify нарешті з’явився в Україні і зараз ним користуються як українські артисти для розміщування власних композицій, так і звичайні користувачі для прослуховування улюбленої музики.

Стрімінг має веб-версію, версію для мобільних додатків та застосунки для Windows та MacOS [3].

Зображення дизайну та керуючих елементів сторінки плеєру сервісу Spotify наведено нижче на рисунку 1.1:

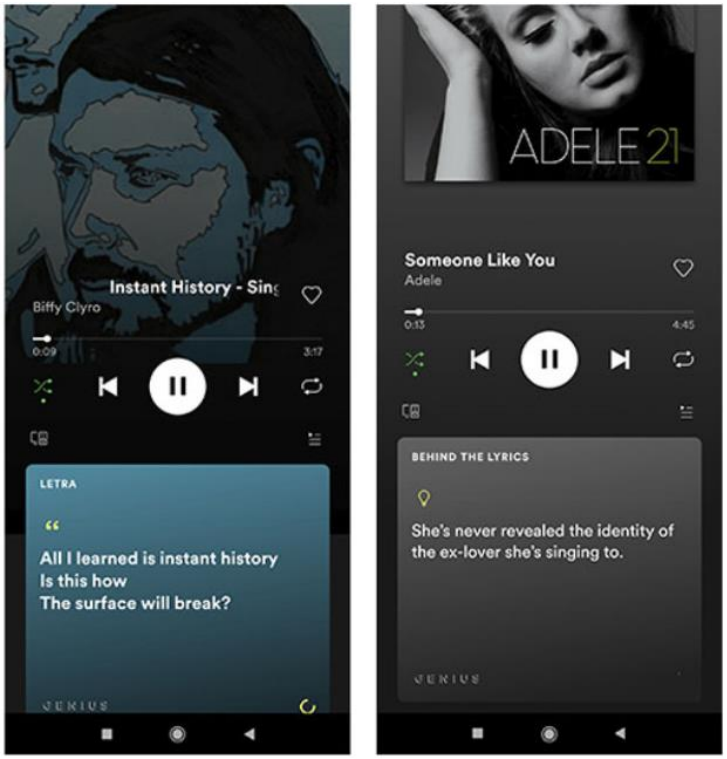

Рисунок 1.1 – Вигляд сторінки плеєру сервісу Spotify

1.1.2 Apple Music

Apple Music – сервіс для потокового прослуховування музики, розроблений компанією Apple, що забезпечує різноманітні способи прослуховування музики:

- доступ до мільйонів пісень в Apple Music у разі оформлення підписки;
- отримання рекомендацій на основі смаку конкретного користувача;
- пошук найсучаснішої музики та поради від музичних експертів;
- перегляд відеокліпів;
- підбір унікальних радіостанцій;
- прослуховування музики з власної фонотеки, зокрема придбану музику з iTunes Store, а також пісні синхронізовані з власного пристрою.

Сервіс Apple Music вважається головним конкурентом Spotify. Компанія Apple завжди робить ставку на вручну відібраний кураторами контент, тож сильна сторона цього сервісу – плейлисти, складені музичними експертами.

Програма Apple Music має інтерфейс із вкладками. Розділ «Для тебе» включає музичні рекомендації. Розділ «Моя музика» містить куплені користувачем треки, які можна прослуховувати через стрімінг. Розділ «Радіо», де звучать треки вибраного жанру або артистів.

Під час створення сервісу враховувались відгуки користувачів таким чином, щоб зробити інтерфейс більш інтуїтивним.

Apple Music працює на пристроях під управлінням IOS версії 8.4 і вище, в iTunes версії 12.2 і вище та в пристроях Apple Watch, а також Android версії 4.3 і вище.

Основні переваги Apple Music:

- вручну підібрані плейлисти музики за жанрами, настроєм та іншими характеристиками;
- наявний інтерфейс українською мовою;
- інтеграція з сервісом Shazam, який дозволяє у реальному часі ідентифікувати музичні композиції, а потім відкривати їх у Apple Music;
- власна радіостанція Beats 1;

Недоліком сервісу є слабкі алгоритми рекомендацій, у порівнянні з конкурентами.

Apple Music має тримісячний безкоштовний період, а після завершення вартість складає 4.99 доларів.

Зображення дизайну та керуючих елементів сторінки плеєру сервісу Apple Music наведено нижче на рисунку 1.2:

Рисунок 2.2 – Вигляд сторінки плеєру сервісу Apple Music

1.1.3 Deezer

Deezer – французький сервіс, який також досяг певної популярності, створений для прослуховування музики в потоковому форматі. Deezer має застосунки для мобільних пристроїв iOS, Android, Windows Phone.

Підписки сервісу Deezer поділяються на такі категорії:

– deezer Free – наявність реклами, обмеження в 6 пропусків на годину, промотовування відсутнє, не має офлайн режиму, звукова якість MP3 128kbit/s;

- deezer Premium – відсутня реклама, необмежене пропускання та промотування, наявність офлайн режиму, звукова якість MP3 320kbit/s, вартість 3.99 доларів;
- deezer Family – відсутня реклама, необмежене пропускання та промотування, наявність офлайн режиму, звукова якість MP3 320kbit/s, шість аккаунтів, вартість 5.99 доларів;
- deezer HiFi - відсутня реклама, необмежене пропускання та промотування, наявність офлайн режиму, звукова якість FLAC, вартість 7.99 доларів.

Сервіс має досить цікаву функцію Flow – єдиний безперервний потік музики на основі вподобань користувача, що запускається однією кнопкою. Також Deezer – єдиний стримінговий сервіс, що має українського редактора – ним став Сергій Кейн, головний редактор видання Comma.

Основні переваги сервісу Deezer:

- функція Flow;
- зручний інтерфейс українською мовою.

Основним недоліком сервісу є обмеження на вподобання – лайкнути можна не більше ніж 2 000 треків;

Зображення дизайну та керуючих елементів сторінки плеєру сервісу Deezer наведено нижче на рисунку 1.3:

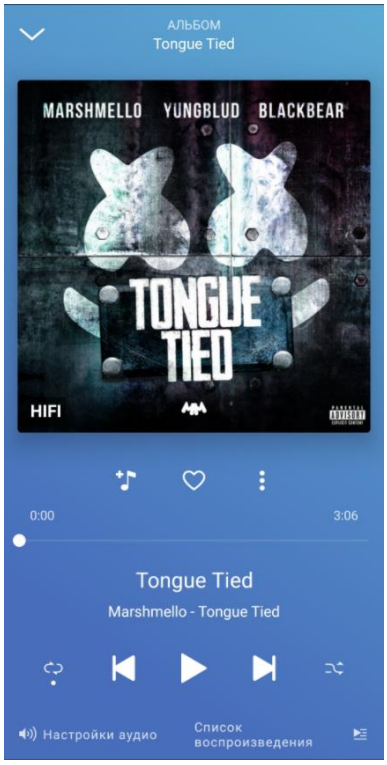

Рисунок 3.3 – Вигляд сторінки плеєру сервісу Deezer

1.1.4 Fizy

Fizy – музичний сервіс мобільного оператора Turkcell, розвитком якого в Україні займається мобільний оператор lifecel. Міжнародна музикальна платформа оголосила про відкриття повного доступу до функцій свого додатку для усіх користувачів смартфонів в Україні. Завантаження додатку та перший місяць користування сервісом безкоштовні, а для користувачів оператора lifecell трафік не тарифікується.

Користувачі платформи мають доступ до створення плейлистів аудіофайлів, відкривати для себе нову музику у плейлистах, які підготувала команда Fizy.

Після завершення користування безкоштовним періодом абоненти lifecell можуть обрати для себе зручний вид підписки:

– базовий Fizy Basic, вартість 49 грн.;

– преміум Fizy Premium, вартість 59 грн. на місяць;

Абоненти інших операторів мають сплатити наступні тарифи:

– базовий Fizy Basic, вартість 2.49 доларів;
– преміум Fizy Premium, вартість 2.99 доларів.

Основні переваги сервісу Fizy:

– вбудовані українські радіостанції;
– інтерфейс українською мовою;

Головним недоліком додатку є відносно маленький каталог у порівнянні з іншими додатками такого масштабу.

Зображення дизайну та керуючих елементів сторінки плеєру сервісу Fizy наведено нижче на рисунку 1.4:

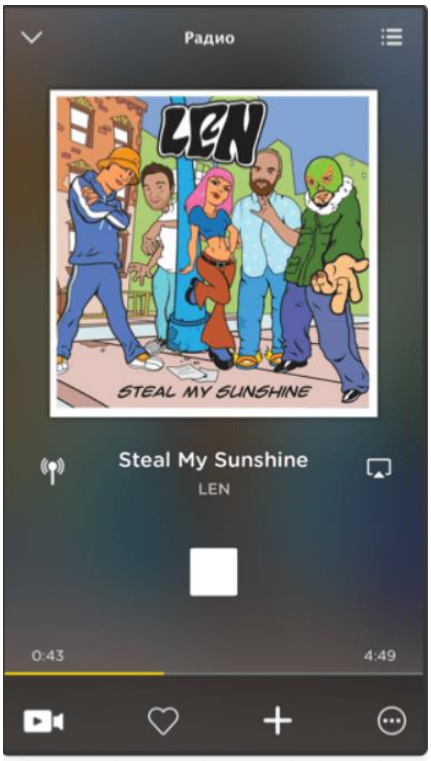

Рисунок 4.4 – Вигляд сторінки плеєру сервісу Fizy

1.2 Збір вимог до програмного забезпечення

Для більш детального встановлення та визначення функціональних вимог до розроблюваної системи було використано методи попереднього збору первинної та

вторинної інформації щодо вимог та функцій, які користувачі очікують побачити в розроблюваній системі [4].

1.2.1 Використані методи отримання первинної інформації

Під час використання методів отримання первинної інформації було обрано метод анкетування, що є найбільш формалізованою формою опитування майбутніх користувачів системи. Цей метод дозволяє отримати розгорнуті відповіді щодо системи на заздалегідь підготовлені і стандартизовані запитання. Основним інструментом анкетування є анкета – структурована послідовність питань, складених таким чином, щоб з'ясувати факти або відношення, які є інструментом для фіксування даних.

Основною метою проведення анкетування є отримання точної інформації від майбутніх користувачів системи. Одним з найважливіших умов є те, щоб користувачам одним і тим же способом були поставлені запитання для формування більш чіткої картини представлення системи.

Анкетування включає в себе наступні етапи:

- підготовчий етап;
- оперативний етап;
- результуючий етап.

При складанні анкети потрібно чітко визначити цілі анкети, щоб майбутній користувач мав чітке уявлення про мету дослідження. Перед кожним важливим дослідженням потрібно ретельно розробити і протестувати використані в ньому анкети. Непрофесійний підхід до складання анкет неодмінно призводить до перекручування реальної картини або отримання результатів, які не співпадають з реальністю.

Сьогодні опитування майбутніх користувачів потрібно розподіляти на основні взаємодії. Це означає, що користувачів потрібно запитувати, коли вони контактують з програмним продуктом. Така форма опитування називається сенсорним аналізом.

Опитування, під час користування має досить великий вплив на подальший розвиток програмної системи.

Поставлені запитання до користувачів, під час проведення анкети представлені нижче:

Таблиця 1.1 – Запитання анкети для користувачів

Номер запитання	Запитання	Варіанти відповідей
1	2	3
1.	Ваш вік?	− Менше 18; − 18-24; − 25-35; − 36-49; − 50-60 − Старше 60 років.
2.	Стать	− Чоловіча; − Жіноча.
3.	Що на вашу думку є основним елементом, який показує якість додатку?	− Вибір кольорів; − Вдале розташування елементів керування; − Швидкодія; − Зручність у використанні.
4.	Що на Вашу думку найголовніше у розробці мобільного додатку керування аудіофайлами?	− Креативність та творчий підхід розробників; − Структурованість та відповідність вимогам;

Продовження табл. 1.1

1	2	3
5.	Як Ви ставитесь до подальшого супроводження додатку?	– Потрібно зберігати додаток в первинному вигляді; – Потрібно супроводжувати додаток та випускати нові версії з виправленими помилками, навіть, якщо додаток дещо змінить свою структуру.
6.	Чи потрібно впровадження додатку на різних платформах?	– Додаток повинен бути ексклюзивним для користувачів однієї системи; – Додаток повинен мати змогу працювати на будь-якому мобільному пристрої;

Продовження табл. 1.1

1	2	3
7.	Які функції є обов'язковими для додатків такого виду?	○ Функція пошуку; ○ Функція рекомендацій; ○ Реєстрація та авторизація; ○ Можливість змінити кольорову палітру; ○ Функція створення власних плейлистів;
8.	Скільки часу Ви витрачаєте на мобільні додатки такого виду?	– Менше години; – Від 2 до 4 годин на день; – Більше 4 годин.

1.2.2 Використані методи отримання вторинної інформації

Основними джерелами отримання вторинної інформації, які були використані під час збору вимог до розроблюваної системи стали пошукові джерела, що наявні у мережі онлайн, публікації статистичної інформації міжнародних видань та звіти окремих фірм, книги та періодичні видання.

Традиційний аналіз звітів фірм-власників подібних систем керування аудіофайлами, який призводить до становлення ланцюжка логічних міркувань та інтерпритації змісту документів, також дозволив вирішити низку питань, які стояли під час формування функціональних вимог щодо розроблюваної системи.

Висновки на основі аналізу та збору вимог

Зібрані вимоги були ретельно проаналізовані для виявлення в них повторів та суперечностей, що призвело до перегляду вимог. Відразу ж вимогам було присвоєно

класифікацію та пріоритет. Аналіз вимог визначив послуги, очікувані від системи і обмеження, яким система повинна підкорятися.

Для більш наглядної демонстрації було обрано найбільш сучасний метод виявлення та демонстрації вимог – метод прототипування. Програмні прототипи конструюються для візуалізації системи або її частини для відображення реальної картини розроблюваної системи. Прототип являє собою графічний інтерфейс користувача і моделює поведінку системи на різні дії користувача. Прототипи дозволяють детально оцінити корисність системи на початку її реалізації.

Після зборі інформації, її було переведено у графічне представлення для кращого сприймання. Система має забезпечувати прегляд аудіофайлів, їх керування та надавати рекомендації на основі аналізу. Тоді бізнес-процес виконується таким чином:

1. Користувач входить до системи, відкриваючи мобільний додаток.
2. Система відразу ж відображає наявні на пристрої аудіофайли.
3. Користувач створює запит на прослуховування.
4. Система відображає екран керування аудіофайлами.
5. Користувач додає аудіофайл до списку «улюблених».
6. Система проводить аналіз наявних аудіофайлів та створює список рекомендацій для користувача.
7. Користувач створює запит на перегляд рекомендованих аудіофайлів.
8. Система відображає список рекомендацій.
9. Користувач виходить з системи.
10. Система робить збереження внесених змін.

В результаті проведеного аналізу та збору вимог методами отримання первинної та вторинної інформації щодо розроблюваної системи було створено бізнес-процес для роботи мобільного додатку керування аудіофайлами з урахуванням музичного смаку користувача. Бізнес-процес зображено на наступному рисунку 1.4:

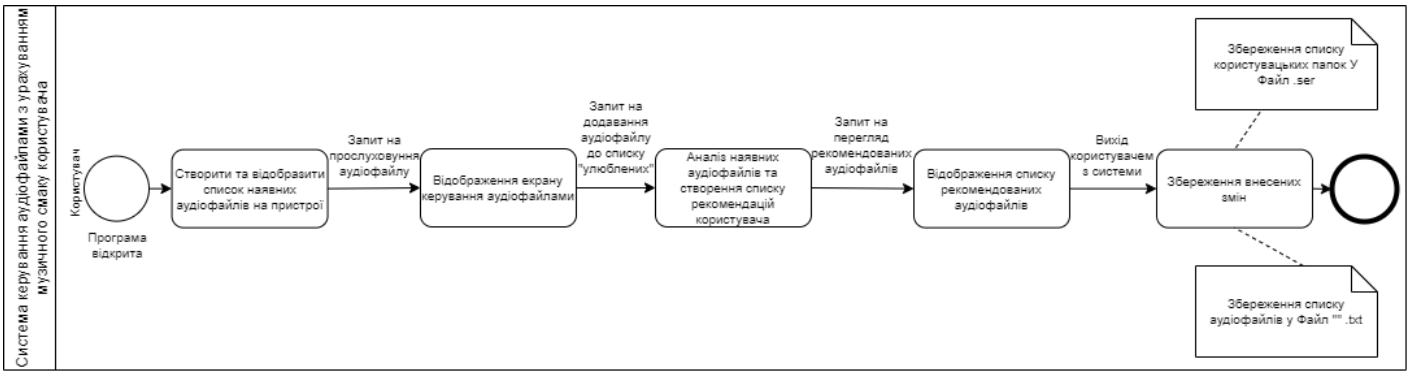

Рисунок 5.5 – Бізнес-процес для забезпечення відображення списку рекомендацій

РОЗДІЛ 2 ПРОЕКТУВАННЯ

2.1 Зовнішнє проектування

2.1.1 Функціональне призначення

Функціональне призначення – програмний продукт повинен надавати можливість перегляду та керування аудіофайлами на пристрої користувача та за потреби надавати власні рекомендації аудіофайлів на основі аналізу музичного смаку конкретного користувача.

2.1.2 Експлуатаційне призначення

Експлуатаційне призначення – оперативність пошуку та відтворення аудіофайлів, скорочення часу для отримання інформації про рекомендовані аудіофайли, зручність виконання операцій перегляду та відтворення аудіофайлів.

2.1.3 Функціональні вимоги

Програмний продукт має забезпечувати можливості:

- перегляду списку аудіофайлів наявних на поточному пристрої користувача у корінній папці «Music»;
- відображення назв аудіофайлів у списку;
- програвання аудіофайлу;
- керування програванням поточним аудіофайлом, а саме:
  - регулювання гучності;
  - перехід до наступного аудіофайлу;
  - перехід до попереднього аудіофайлу;
  - реалізація паузи аудіофайлу;
  - реалізація відтворення аудіофайлу;
  - додавання поточного аудіофайлу до власного списку;
  - швидке зберігання аудіофайлу до списку «улюблених» аудіофайлів;

        - перемотка аудіофайлу у будь-яке місце в рамках хронометражу аудіофайлу.
    - пошуку аудіофайлів за назвою;
    - створення власних списків аудіофайлів;
    - видалення аудіофайлів з власних списків;
    - видалення власних списків аудіофайлів;
    - перегляду списку «улюблених» аудіофайлів;
    - перегляду списку «нещодавних» аудіофайлів;
    - формування списку рекомендацій з використанням «улюблених» та «нещодавніх» аудіофайлів;
    - переходу до списку рекомендацій аудіофайлів;
    - збереження результатів роботи системи у файл;
    - керування аудіофайлом з панелі повідомлень, а саме:
        - перехід до наступного аудіофайлу;
        - перехід до попереднього аудіофайлу;
        - реалізація паузи аудіофайлу;
        - реалізація відтворення аудіофайлу;
        - відображення назви аудіофайлу.

Область застосування: програмний продукт призначений для застосування рядовими користувачами, які потребують інструмент для управління аудіофайлами на пристрої.

2.1.4 Вхідні дані

Вхідними даними мобільного додатку є:

- параметри пошуку аудіофайлу – назва аудіофайлу (літери або цифри довжиною від 0 до 255 символів).
- аудіофали наявні на пристрої (формат аудіофайлів - .mp3 або .wav).

– параметри для створення власного списку – назва списку (літери або цифри від 0 до 12 символів).

Кодування усіх символів – ASCII.

2.1.5 Вихідні дані

Вихідними даними мобільного додатку є:

– формування списків аудіофайлів;
– відображення списку усіх доступних аудіофайлів;
– відображення аудіодоріжки в реальному часі;
– відображення списку рекомендованих аудіофайлів;
– відображення списку «улюблених» аудіофайлів;
– відображення списку «нещодавніх» аудіофайлів;
– файл формату .txt з списками назв аудіофайлів наявних у кожній папці;
– файл формату .ser з списком типу ArrayList з назвами та шляхом де зберігаються малюнки до відображуваних папок.

2.1.6 Опис зовнішнього інформаційного середовища

Програма коректно працюватиме для смартфонів під керуванням платформи ОС Android версії 5.0 та вище.

Параметри для пошуку аудіофайлу та параметри для створення власного списку вводяться користувачем з клавіатури смартфона з відповідним контролем введення. Пошук наявних аудіофайлів на пристрої користувача відбувається шляхом аналізу внутрішньої директорії пристрою, зокрема у корінній папці “Music” файлів формату .mp3 та .wav. Обрання пунктів меню та взаємодія з інтерфейсом користувача відбувається шляхом натискання на певні елементи керування на екрані пристрою.

Специфікація функціональних вимог представлена у вигляді діаграми прецедентів. На ній графічно показана сукупність прецедентів та суб’єктів, а також відношення між ними. Графічно діаграма прецедентів представлена на рисунку 2.1:

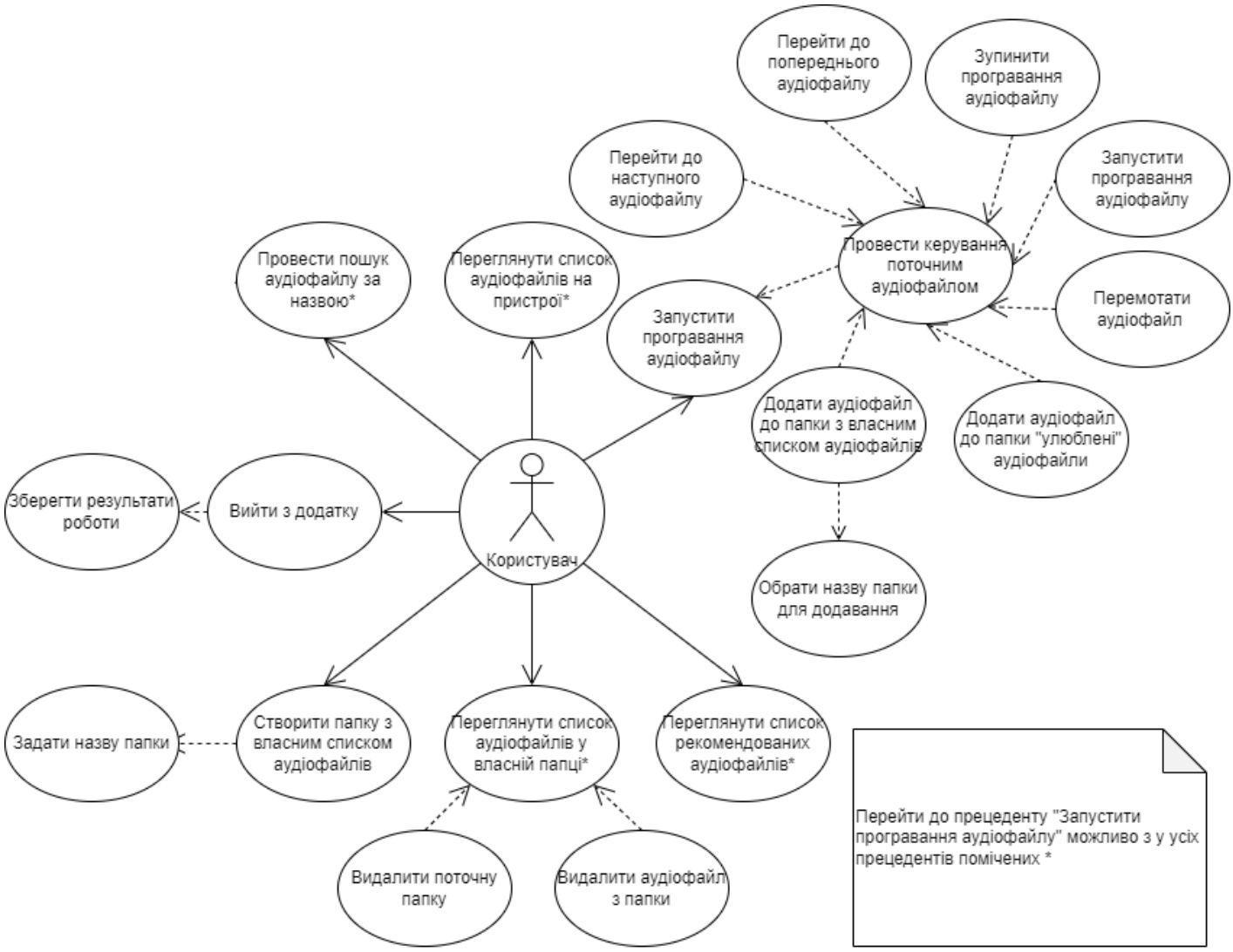

Рисунок 2.1 – Діаграма прецедентів

2.2 Внутрішнє проектування

2.2.1 Статична архітектура системи, моделювання залежностей

Мобільний додаток можна розбити на 4 логічні групи екранів взаємодії:

- головна сторінка, з якої відбувається перехід до 2 екранів (екран плеєру, екран користувацької папки);

– сторінка користувацької папки;
– сторінка плеєру;
– сторінка додавання аудіофайлу до користувацької папки.

Схема екранів системи та їх взаємодії представлена на рисунку 2.2:

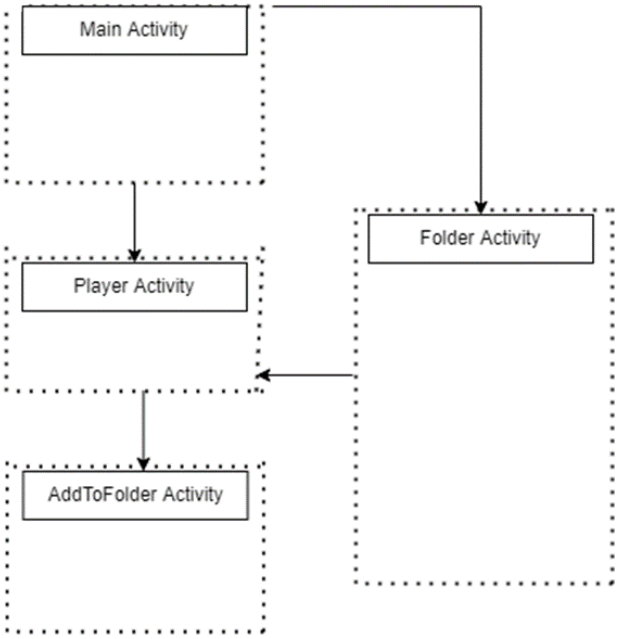

Рисунок 2.2 – Схема екранів додатку та їх взаємодії

Кожен екран мобільного додатку реалізується окремим класом, що розширює базовий клас AppCompatActivity бібліотеки appcompat e v7.

Модуль «Main Activity» головної сторінки є керуючим. Саме з нього можна перейти до наступних сторінок ієрархії додатку.

Модуль «Player Activity» є модулем керування аудіофайлом. Завдяки ньому, можна переходити до наступного або ж попереднього аудіофайлу, ініціювати відтворення або ж паузу, швидко додавати аудіофайл у папку «улюблені», перемотувати аудіофайл у будь-яке місце його хронометражу.

Модуль «Folder Activity» дозволяє відображати інформацію про користувацькі папки, аудіофайли, що розміщені в них. Цей модуль також взаємодіє з модулем плеєру шляхом натискання на будь-який аудіофайл у папці.

Модуль «AddToFolder Activity» дозволяє користувачу додати аудіофайл у користувацьку папку.

Діаграма класів пов'язана з головним екраном зображена на рисунку 2.3 :

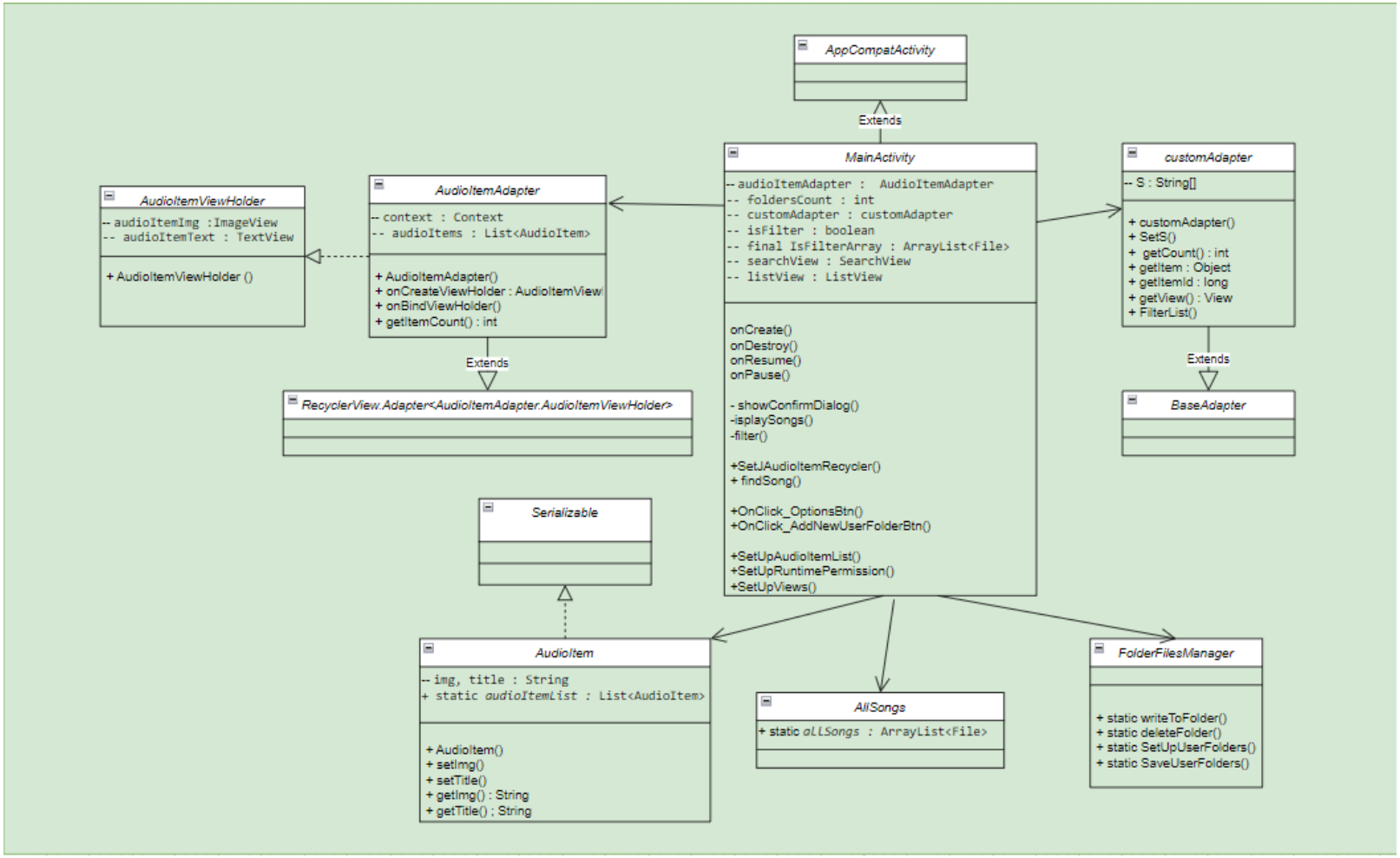

Рисунок 2.3 – Діаграма класів пов'язана з головним екраном

Діаграма класів пов'язана з екраном плеєру зображена на рисунку 2.4:

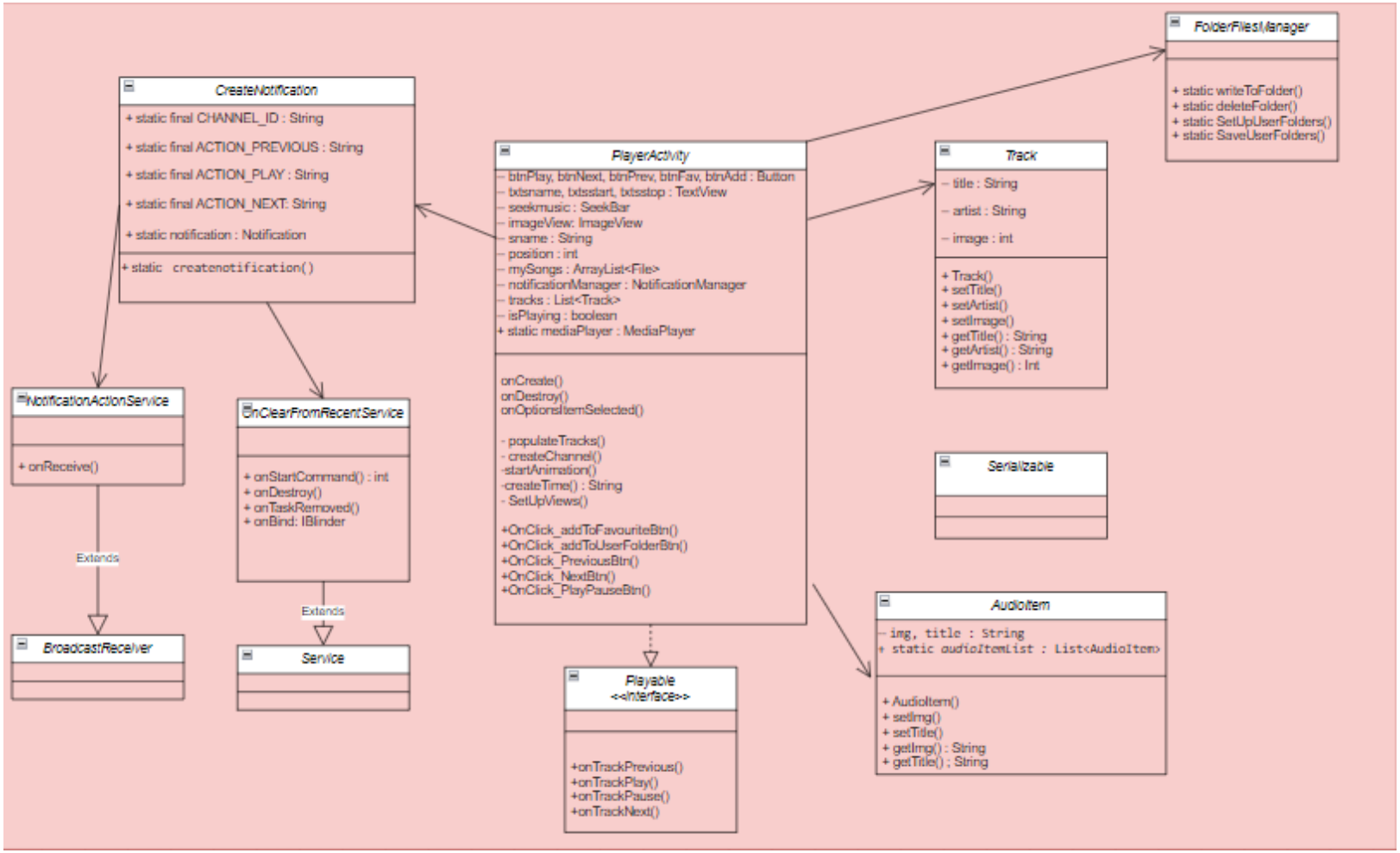

Рисунок 2.4 – Діаграма класів пов'язана з екраном плеєру

Діаграма класів пов'язана з екраном додавання аудіофайлу до користувацької папки зображена на рисунку 2.5:

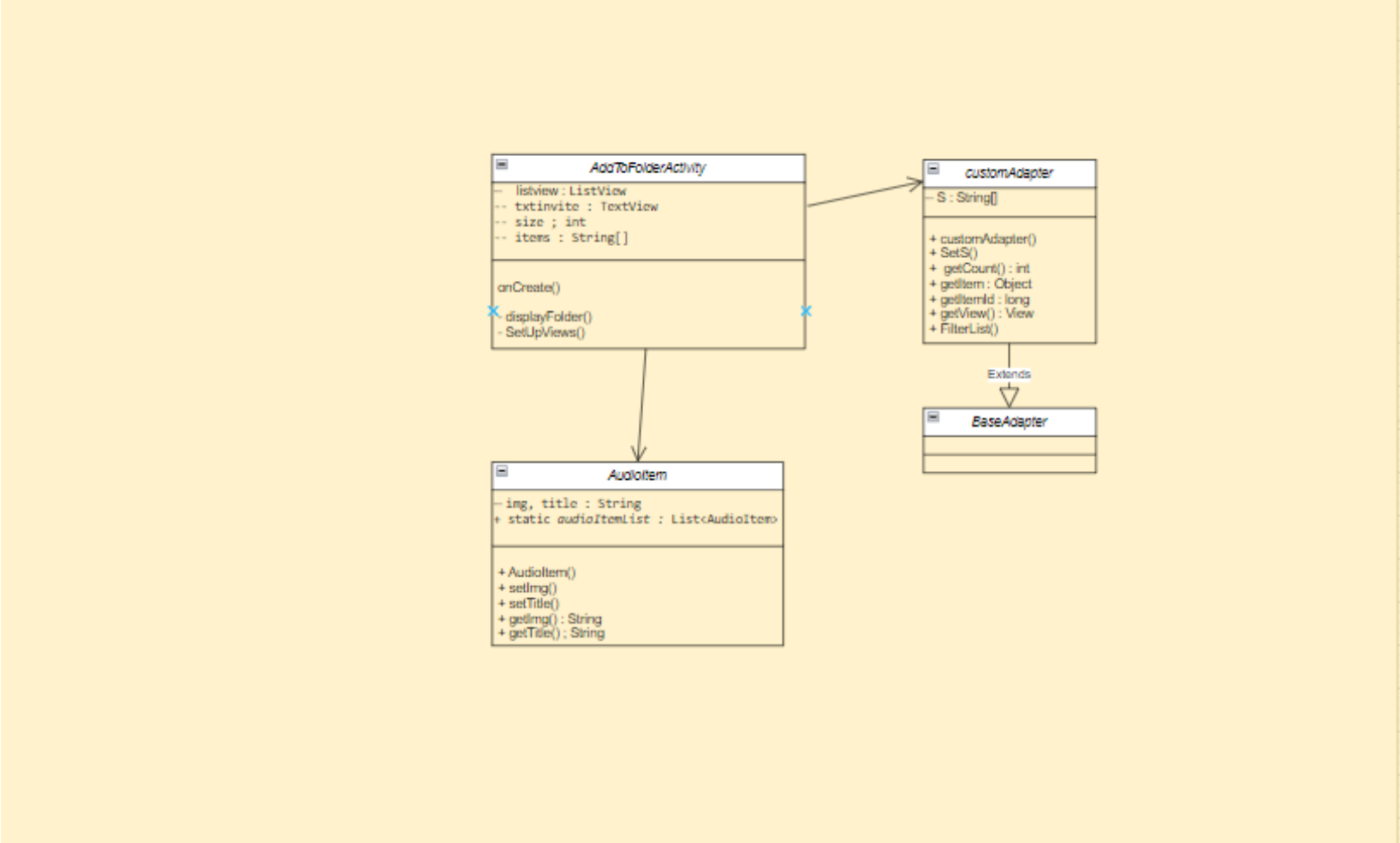

Рисунок 2.5 – Діаграма класів пов'язана з екраном додавання аудіофайлу до користувацької папки

Діаграма класів пов'язана з екраном користувацької папки зображена на рисунку 2.6:

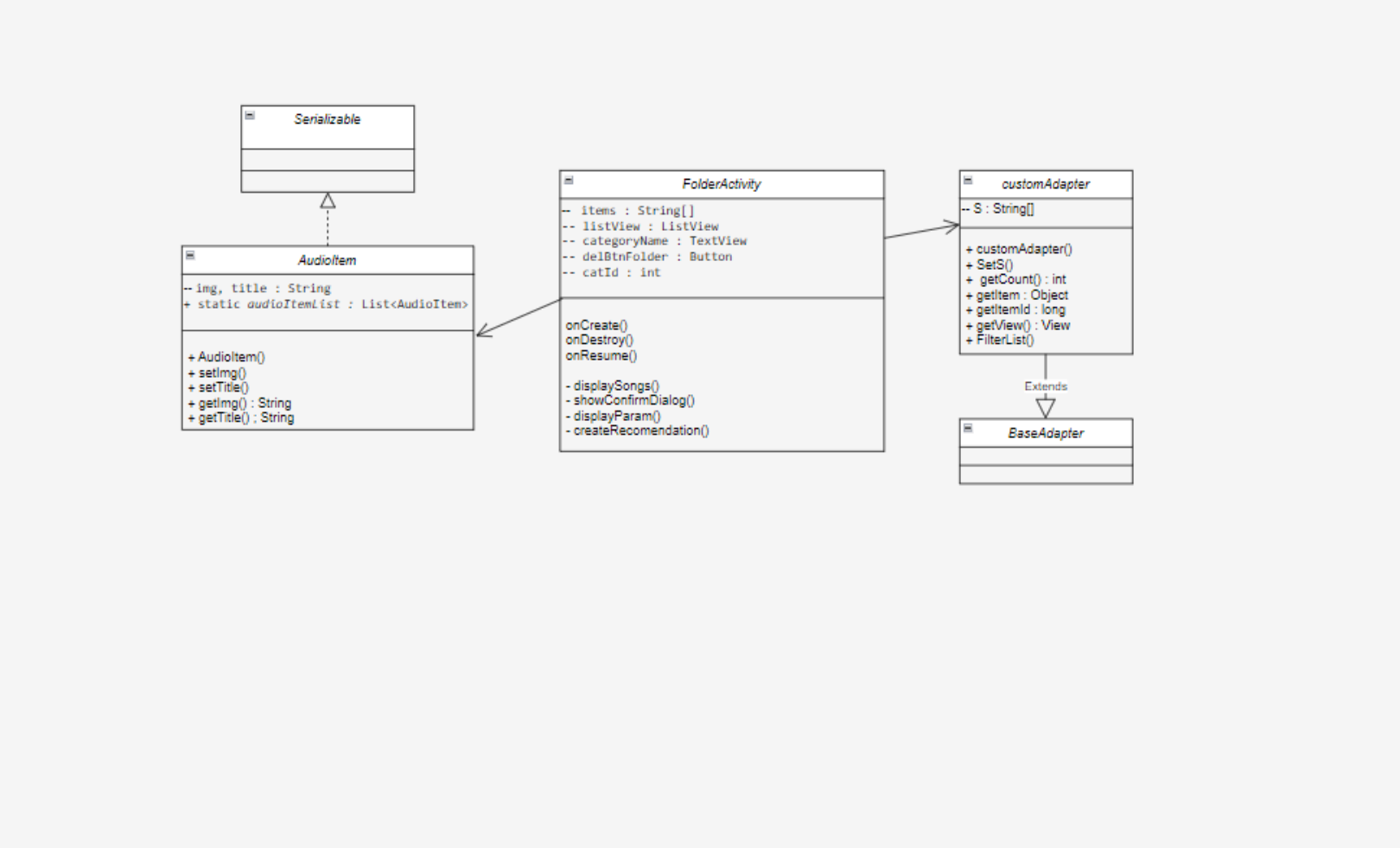

Рисунок 2.6 – Діаграма класів пов'язана екраном користувацької папки

2.2.2 Структура внутрішнього сховища даних

Система вміщує в себе декілька різних структур даних: AllSongs, AudioItem, Track. Всі вони є локальними структурами даних додатку.

Поле allSongs типу ArrayList<File> публічного класу AllSongs представляє собою сховище для елементів типу File формату .mp3 або ж .wav – аудіофайли, що були зчитані з внутрішнього сховища пристрою корінної папки “Music”.

Поле audioItemList типу List<AudioItem> публічного класу AudioItem представляє собою сховище для елементів типу AudioItem – елементи, що являють собою користувацькі папки, які створені під час роботи з додатком.

Поля title типу String, artist типу String, image типу int публічного класу Track представляють собою структуру даних для зберігання інформації про окремий аудіофайл, його назву, виконавця, та номер зображення.

У додатку передбачена функція зберігання результату роботи користувача, що заноситься до окремих файлів, які зберігаються у корінній папці додатку. Зокрема для кожної користувацької папки зберігається текстовий файл формату .txt з переліком аудіофайлів, які користувач додав до власної папки. Окрім цього, також передбачено збереження користувацьких папок, які були створені під час роботи. Вони також зберігаються у корінній папці у файлі формату .ser.

2.2.3 Особливості інтерфейсу користувача

Інтерфейс користувача – це засоби, за допомогою яких користувач взаємодіє з системою. З одного боку відбувається взаємодія користувача з інтерфейсом, а з іншого реакція системи на цю взаємодію. Розробка інтерфейсу користувача – нелінійний процес, тому що відсутній фіксований алгоритм від початку до кінця проектування. Будь-який аспект проектного рішення може впливати на інші аспекти, до того ж результат цього впливу не завжди позитивно відображається на весь проект. При проектуванні інтерфейсу користувача особливу увагу було приділено сценарію, структурі діалогу, взаємному розташуванню і розміру відображуваних об’єктів, кольоровій палітрі та анімації [5].

Під час розробки інтерфейсу користувача враховувались декілька головних принципів, які прослідковуються упродовж розробки усієї програми.

Корисність – використання повідомлень, які допомагатимуть користувачу у роботі. Використання зрозумілих термінів як для повідомлень, так і для усіх текстів на екрані: текстів кнопок, назв заголовків, функцій. При розробці інтерфейсу враховувалась більш проста лексика та правильний тон, адже невдала термінологія та тон можуть призвести до того, що користувач звинуватить у помилках сам себе.

Поблажливість – достатній зворотній зв’язок з користувачем та створення умов для негайних та зворотній дій. Недостатність цього принципу може викликати збільшення часу на виконання певної задачі.

Можливість орієнтування – забезпечення користувачу до будь-якої частини інтерфейсу. Користувач повинен мати можливість вільно орієнтуватися в інтерфейсі, просуватися вперед і назад по низхідній та висхідній структурі інтерфейсу.

Дружність – забезпечення користувача повідомленнями про різного роду помилки та повідомленнями про поточний стан системи. Уникання ситуацій при яких користувач може зіткнутися з помилками.

Простота інтерфейсу – забезпечення легкого та інтуїтивного використання інтерфейсу, швидкого звикання користувача до наявних можливостей інтерфейсу. При вдалому проектуванні користувач повинен не помічати інтерфейс.

Інтерактивність – всюди, де це можливо потрібно дозволити користувачеві маніпулювати елементами інтерфейсу.

Розпізнавання – підтримка інтерфейсом довгочасної пам’яті. Користувачеві легше обрати якийсь об’єкт зі списку, ніж згадувати його назву для введення у порожній рядок.

Інформування – наявність візуальних малюнків. Користувач повинен швидко орієнтуватися у програмному забезпеченні, повинен знати в якому місці програмного забезпечення він знаходиться, які дії виконуються наразі.

Зворотній зв’язок – програма повинна мати реакцію на будь-яку дію користувача.

Узгодженість – користувач під час роботи з системою може використовувати прийоми роботи з деякою частиною іншої системи.

Організація – збільшення візуальної ясності інтерфейсу, полегшити сприйняття інформації шляхом використання груп об’єктів в меню або списки.

Сумісність – користувач може перенести свої знання та навички з іншої подібної програми, якщо він нею користувався, у нову.

Відношення – інтерфейс повинен мати естетичну привабливість та цілісність.

Щодо вимог до основних візуальних атрибутів відображення інформації, враховувались наступні пункти.

Колір є одним з атрибутів при розробці інтерфейсу користувача. Він повинен застосовуватись обережно, адже люди мають різноманітні фізіологічні, психологічні та емоційні реакції на кольори. Один з основних принципів використання кольорів – це не нашкодити користувачу. Операційні системи пропонують стандартні кольорові схеми і палітри, які слід використовувати при розробці інтерфейсу користувача.

За основу для вибору кольорів додатку було обрано принципи material design – основи дизайну сайтів та програмного забезпечення для інтерфейсів операційної системи Android від компанії Google, основна ідея якого полягає в інтерфейсі, вигляд якого наслідує правила вигляду паперових карток в реальному житті. Кольорова палітра додатку зображена на рисунку 2.7:

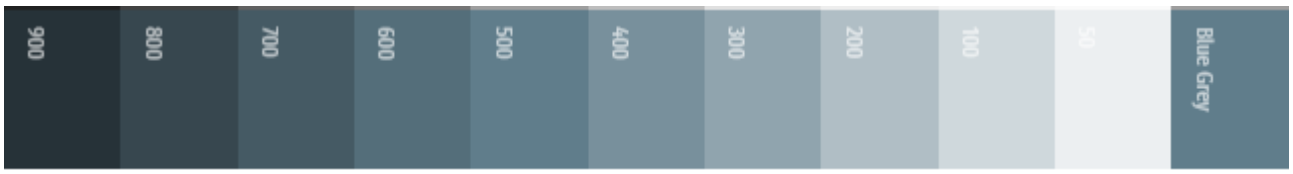

Рисунок 2.7 – Кольорова палітра основних акцентів додатку

Рисунок 2.8 – Кольорова палітра допоміжних акцентів додатку

Розташування та розмір графічних елементів також є важливим атрибутом при розробці інтерфейсу користувача. Групування елементів, їх розташування повинне бути структуроване та відповідати певному алгоритму розміщення. Вирівнювання елементів повинне бути однакове протягом усієї програми.

На основі сказаного вище біли розроблені ескізи головних екранів додатку.

Ескізи мобільного додатку мають схематичний характер та служать для зручності та розуміння розробником системи, що проектується.

Структура мобільного додатку складається з наступних екранів:

- екран завантаження додатку;
- екран головної сторінки;
- екран плеєру;
- екран перегляду списку аудіофайлів у папці;
- екран додавання аудіофайлу у папку.

Схематичне зображення екрану завантаження додатку продемонстроване на рисунку 2.9:

Рисунок 2.9 – Ескіз екрану завантаження додатку

Схематичне зображення екрану головної сторінки продемонстроване на рисунку 2.10:

Рисунок 2.10 – Ескіз екрану головної сторінки

Схематичне зображення екрану плеєру продемонстроване на рисунку 2.11:

Рисунок 2.11 – Ескіз екрану плеєру

Схематичне зображення екрану перегляду списку аудіофайлів у папці продемонстроване на рисунку 2.12:

Рисунок 2.12 – Ескіз екрану перегляду списку аудіофайлів у папці

Схематичне зображення екрану додавання аудіофайлу у папку продемонстроване на рисунку 2.13:

Рисунок 2.13 – Ескіз екрану додавання аудіофайлу у папку

2.2.4 Моделювання процесу виконання операцій

Діаграма діяльності мобільного додатку при формуванні списку рекомендацій зображена на рисунку 2.14:

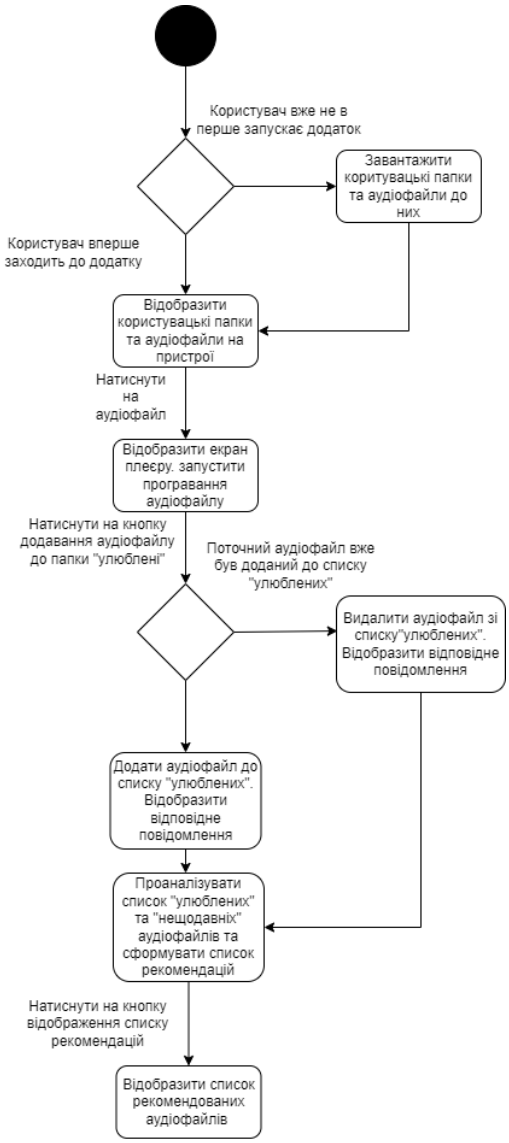

Рисунок 2.14 – Діаграма діяльності при формуванні списку рекомендацій

Початковим станом мобільної системи є запуск мобільного додатку. Під час завантаження користувачу відображається екран завантаження. Якщо вхід користувача до системи відбувається не в перше, то система завантажує користувацькі папки та аудіофайли в них до внутрішнього сховища програми. Якщо користувач входить вперше, завантаження не відбувається. Після натискання на аудіофайл відкривається екран плеєру. Користувач натискає на кнопку додавання аудіофайлу до «улюблених»

аудіофайлів. Якщо поточний аудіофайл вже є у списку «улюблених», то аудіофайл видаляється зі списку. Якщо його не було у списку, то він туди додаться. Після цього формується новий список рекомендацій. Після натискання на кнопку відображення рекомендацій відображається новий список рекомендацій.

2.2.5 Моделювання словника системи

Для моделювання словника системи виконаємо аналіз зовнішніх специфікацій системи.

Ідентифіковані сутності:

track, audioItem, allSongs, createNotification, folderFilesManager

Ідентифіковані обов’язки:

- Track – збереження даних про конкретний аудіофайл;
- audioItem – збереження інформації про конкретну папку користувача ;
- allSongs – збереження інформації про наявні аудіофайли на пристрої ;
- createNotification – створення рядку повідомлень з елементами керування програванням аудіфайлу;
- folderFilesManager – управління процесами збереження та видалення з файлів ;

Додатково зазначимо активності:

- головна сторінка;
- сторінка плеєру;
- сторінка користувацької папки;
- сторінка екрану завантаження;
- сторінка додавання аудіофайлу у користувацьку папку.

Атрибути та операції, необхідні для виконання обов’язків кожної сутності-класу наведемо у таблиці 2.1:

Таблиця 2.1 Сутності, атрибути та методи

Сутність	Атрибути	Методи
1	2	3
Track	Title – назва аудіофайлу Artist – ім’я виконавця Image – номер малюнку	Заповнити назву Заповнити виконавця Заповнити номер малюнку
audioItem	Img – розташування малюнку папки в директорії title – назва папки audioItemList - список користувацьких папок	Заповнити назву папки Заповнити розташування малюнку папки в директорії
createNotification	Номер каналу повідомлень Поле відображення стану програвання попереднього аудіофайлу Поле відображення стану програвання наступного аудіофайлу Поле відображення стану зупинки відтворення аудіофайлу	Створити повідомлення

Продовження табл 2.1

1	2	3
folderFilesManager		Записати до файлу Прочитати з файлу Видалити файл Завантажити користувацькі папки Зберегти користувацькі папки
Головна сторінка	Поле адаптеру відображення користувацьких папок Поле кількості користувацьких папок Поле підтвердження пошуку Поле для збереження відсортаваних пошуком аудіофайлів Поле елементу пошуку Поле елементу списку	Відобразити діалог Відобразити аудіофайли Знайти аудіофайли на пристрої Провести пошук серед аудіофайлів за символами Натискання на кнопку додавання нової папки Примінити налаштування списку користувацьких папок Створити діалог запрошення дозволу на читання файлів

Продовження табл. 2.1

1	2	3
Сторінка плеєру	Поля для кнопок Текстові поля інформації Поле прогресу програвання аудіофайлу Поле для малюнку Поле поточної позиції у списку аудіофайлів Поле назви аудіофайлу Список аудіофайлів для програвання Поле підтвердження процесу програвання аудіофайлу	Ініціювати створення повідомлення Завантажити поточний список аудіофайлів Почати програвання анімації Натискання на кнопку додавання аудіофайлу до користувацької папки Натискання на кнопку додавання до списку «улюблених» Натискання на кнопку переходу до попереднього аудіофайлу Натискання на кнопку переходу до наступного аудіофайлу Натискання на кнопку відтворення аудіофайлу

Продовження табл. 2.1

1	2	3
Сторінка користувацької папки	Поле списку аудіофайлів у поточній папці Поле елементу списку Поле для відображення інформації Поле елементу видалення папки Поле номеру категорії папки	Відобразити список аудіофайлів Відобразити діалог
Сторінка екрану завантаження	Поле для відображення логотипу	Відобразити логотип додатку
Сторінка додавання аудіофайлу у користувацьку папку.	Поле для відображення повідомлення Список для відображення користувацьких папок	Відобразити користувацькі папки

2.2.6 Моделювання примітивних типів.

Наведемо опис специфікацій класів програми, їх атрибути та виконувані операції.

Опис специфікацій класу Track.

Атрибути класу:

- змінна, що зберігає назву аудіофайлу;
- змінна, що зберігає виконавця аудіофайлу;
- змінна, що зберігає жанр аудіофайлу;
- змінна, що зберігає тривалість аудіофайлу;
- змінна, що зберігає назву альбому аудіофайлу;

- змінна, що зберігає бітрейт аудіофайлу;
- змінна, що зберігає дату створення аудіофайлу;
- змінна, що зберігає кількість натискань на аудіофайл користувачем;
- змінна, що зберігає сам файл;
- змінна, що відповідає за порядковий номер малюнку до аудіофайлу.

Клас використовується для зберігання інформації про аудіофайл.

Опис специфікацій класу FolderFilesManager.

Клас реалізовує методи:

- метод, що читає список з файлу;
- метод, що записує список до файлу;
- метод, що видаляє записаний файл;
- метод, що записує користувацькі папки до файлу;
- метод, що читає користувацькі папки з файлу.

Опис специфікацій класу AudioItem.

Атрибути класу:

- змінна, що зберігає шлях до малюнку папки;
- змінна, що зберігає назву папки;
- змінна, що зберігає список з папками.

Опис специфікацій класу AllSongs.

Атрибути класу:

- змінна, що зберігає усі наявні аудіофайли;
- змінна, що зберігає інформацію про всі наявні аудіофайли.

Опис специфікацій класу CreateNotification.

Атрибути класу:

- змінна, що зберігає статичну назву каналу для виводу повідоплення;
- змінна, що зберігає статичну назву для дії «перейти до минулого файлу»;
- змінна, що зберігає статичну назву для дії «почати програвання»;

- змінна, що зберігає статичну назву для дії «перейти до наступного файлу».

Клас реалізовує метод створення повідомлення для користувача.

Наведемо опис специфікацій екранів програми, їх атрибути та виконувані операції.

Опис специфікацій екрану AddToFolderActivity.

Атрибути класу:

- змінна, що зберігає кількість користувацьких папок;
- змінна, що зберігає назви користувацьких папок;
- змінна, що зберігає текст для відображення;
- змінна, що зберігає посилання на список.

Клас реалізовує методи:

- відображення користувацьких папок;
- налаштування елементів екрану.

Опис специфікацій екрану FolderActivity.

Атрибути класу:

- змінна, що зберігає назви аудіофайлів;
- змінна, що зберігає посилання на список;
- змінна, що зберігає текст для відображення;
- змінна, що зберігає посилання на кнопку видалення.
- змінна, що зберігає порядковий номер папки.

Клас реалізовує методи:

- відображення аудіофайлів;
- відображення діалогу;
- налаштування списку;
- створення рекомендацій для користувача.

Опис специфікацій екрану MainActivity.

Атрибути класу:

- змінна, що зберігає кількість папок;
- змінна, що вказує чи відбувається пошук;
- змінна, що містить посилання на список;
- змінна, що містить посилання на поле пошуку.

Клас реалізовує методи:

- відображення діалогу;
- відображення аудіофайлів;
- пошук серед наявних аудіофайлів;
- налаштувати елементи екрану;
- натискання на кнопку додававння папки;
- відображення запиту на дозвіл роботи зі сховищем.

Опис специфікацій екрану PlayerActivity.

Атрибути класу:

- змінна, що зберігає посилання на кнопку «перейти до наступного»;
- змінна, що зберігає посилання на кнопку «перейти до попереднього»;
- змінна, що зберігає посилання на кнопку «почати програвання»;
- змінна, що зберігає посилання на кнопку «додати до улюблених»
- змінна, що зберігає посилання на текст «назва аудіофайлу»;
- змінна, що зберігає посилання на текст «початковий час програвання»;
- змінна, що зберігає посилання на текст «кінцевий час програвання»;
- змінна, що зберігає посилання на елемент «відображення процесу програвання»;
- змінна, що зберігає посилання на малюнок аудіофайлу;
- змінна, що отримує назву аудіофайлу;
- змінна, що отримує позицію аудіофайлу;
- змінна, що зберігає посилання на елемент керування програванням;
- змінна, що зберігає список аудіофайлів;
- змінна потоку;

- змінна для відображення повідомлення;
- змінна, що зберігає список інформації про аудіофайли;
- змінна, що показує чи грає аудіофайл;

Клас реалізовує методи:

- відображення анімації;
- створення потоку;
- натискання на кнопку переходу до наступного;
- натискання на кнопку переходу до попереднього;
- натискання на кнопку початку програвання;
- натискання на кнопку додавання до списку «улюблених»;
- натискання на кнопку додавання до користувацької папки;
- відображення повідомлення;
- заповнення інформації про аудіофайли.

Перейдемо до опису інтерфейсної частини класів програми.

Опис інтерфейсної частини класу FolderFilesManager представлено у таблиці 2.2:

Таблиця 2.2 Опис інтерфейсної частини класу FolderFilesManager

Назва	Опис	Вхід	Вихід
1	2	3	4
readFromFolder(String folderName)	Читає список назв аудіофайлів з файлу	String folderName – назва файлу для читання	ArrayList<File> - список аудіофайлів, назви яких знаходяться у файлі

Продовження табл. 2.2

1	2	3	4
writeToFolder(ArrayList<File> arraySong, String folderName)	Записує до файлу список назв аудіофайлів	ArrayList<File> arraySong – список аудіофайлів, назви яких записуються до файлу String folderName – назва файлу до якого відбувається запис	
deleteFolder(String folderName)	Видаляє файл за назвою	String folderName – назва файлу для видалення	
SetUpUserFolders()	Читає з файлу користувацькі папки		
SaveUserFolders()	Зберігає до файлу користувацькі папки		

Опис інтерфейсної частини класу CreateNotification представлено у таблиці 2.3:

Таблиця 2.3 Опис інтерфейсної частини класу CreateNotification

Назва	Опис	Вхід	Вихід
createnotification(Context context, Track track, int playbutton, int pos, int size	Створює повідомлення у вигляді елементу керування програванням аудіофайлів	Context context – контекст створення Track track – інформація про аудіофайл int playbutton – кнопка програвання int pos – позиція аудіофайлу у списку int size – довжина списку для програвання	

Опис інтерфейсної частини класу AddToFolderActivity представлено у таблиці 2.4:

Таблиця 2.4 Опис інтерфейсної частини класу AddToFolderActivity

Назва	Опис	Вхід	Вихід
displayFolder(List<AudioItem> audioItemList)	Відображає список папок	List<AudioItem> audioItemList – список користувацьких папок	

Опис інтерфейсної частини класу FolderActivity представлено у таблиці 2.5:

Таблиця 2.5 Опис інтерфейсної частини класу FolderActivity

Назва	Опис	Вхід	Вихід
displaySongs(ArrayList<File> mySongs)	Відображає список аудіофайлів у папці	List<AudioItem> audioItemList – список аудіофайлів	
showConfirmDialog(int i)	Відображає діалог видалення з користувачем	int i – порядковий номер аудіофайлу	
displayParam(int catId)	Відображення для конкретної папки відповідно до категорії	int catId – порядковий номер папки	
ArrayList<File> createRecomendation()	Створює список рекомендацій для користувача		ArrayList<File> - список рекомендованих аудіофайлів

Опис інтерфейсної частини класу MainActivity представлено у таблиці 2.6:

Таблиця 2.6 Опис інтерфейсної частини класу MainActivity

Назва	Опис	Вхід	Вихід
showConfirmDialog()	Відображення діалогу створення нової папки		
displaySongs()	Відображає наявін аудіофайли		
filter(String newText)	Проводить пошук серед аудіофайлів	String newText – текст для пошуку	
OnClick_AddNewUserFolderBtn(View view)	Натискання на кнопку створення нової папки, відображення діалогу		
SetUpAudioItemList()	Налаштувує елементів списку		
SetUpRuntimePermission()	Виведення діалогу на запит роботи зі сховищем		

Опис інтерфейсної частини класу PlayerActivity представлено у таблиці 2.7:

Таблиця 2.7 Опис інтерфейсної частини класу PlayerActivity

Назва	Опис	Вхід	Вихід
1	2	3	4
populateTracks()	Заповнення інформації про аудіофайли		
StartAnimation(View view)	Відображення анімації для елементів		
String createTime(int duration)	Створення текстового поля для відображення часу програвання	int duration – тривалість аудіофайлу	String – текстове поле для відображення поточного часу програвання
SetUpViews()	Налаштовує елементи екрану		

Продовження табл. 2.7

1	2	3	4
OnClick_addToFavouriteBtn(View view)	Спрацьовує при натисканні на кнопку додавання до «улюблених», оновлює список «улюблених треків» шляхом додавання поточного аудіофайлу, змінює малюнок для кнопки		
OnClick_addToUserFolderBtn(View view)	Спрацьовує при натисканні на кнопку додавання до власної папки, відкриває новий екран для вибору папки		

Продовження табл. 2.7

1	2	3	4
OnClick_PreviousBtn(View view)	Спрацьовує при натисканні на кнопку переходу до попереднього аудіофайлу		
OnClick_NextBtn(View view)	Спрацьовує при натисканні на кнопку переходу до наступного аудіофайлу		
OnClick_PlayPauseBtn(View view)	Спрацьовує при натисканні на кнопку початку/зупинки програвання		

2.2.7 Проектування динаміки системи

Створимо діаграму послідовностей для процесу відображення рекомендацій користувачу. Зображення діаграми послідовностей наведено на рисунку 2.15:

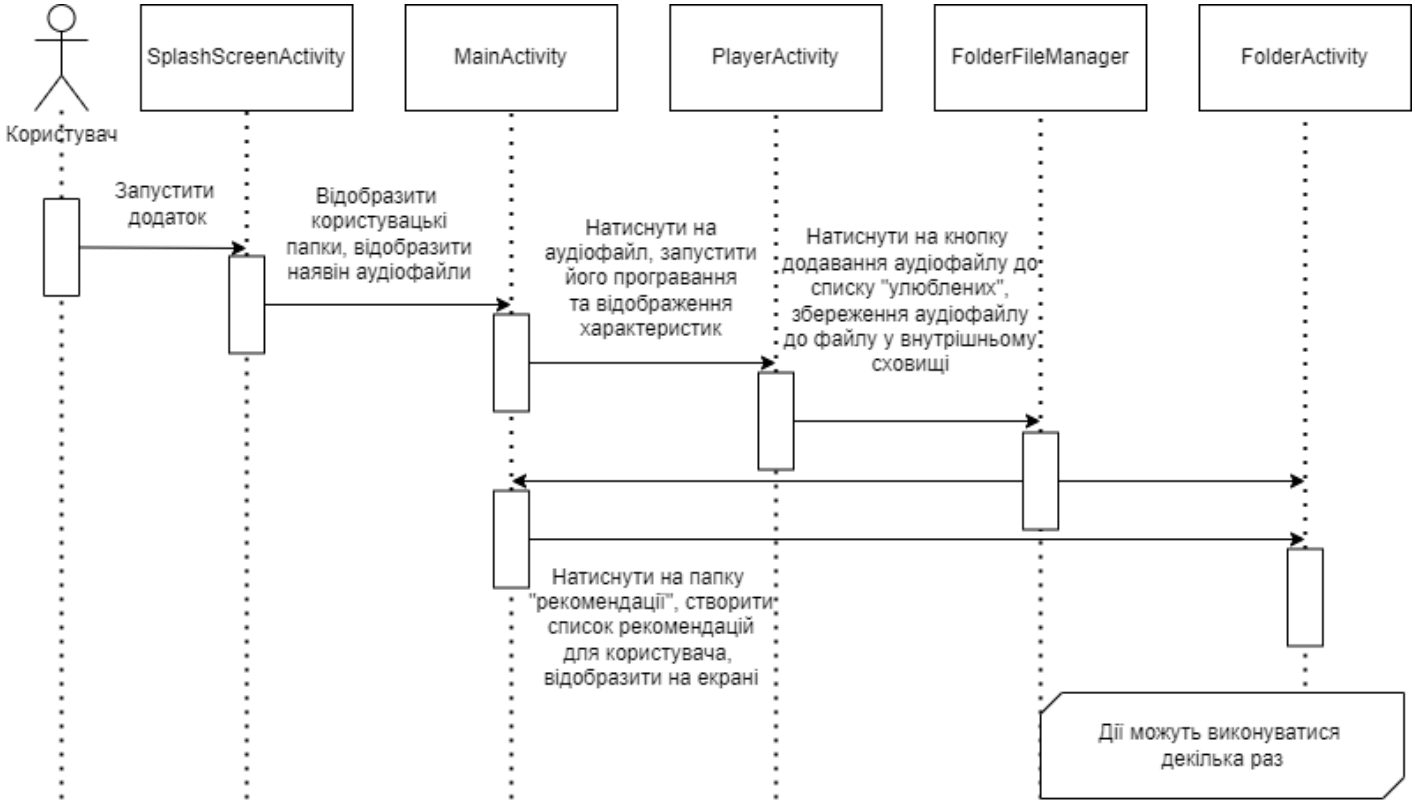

Рисунок 2.15 – Діаграма послідовностей для процесу створення рекомендацій користувачу

Дії, що показані на діаграмі можуть виконуватись нескінченну кількість раз з різними аудіофайлами, результат рекомендацій буде різний для кожного доданого аудіофайлу до списку «улюблених».

Висновки на основі проектування

Читаючи про структуру проекту, можна побачити, що усі структурні елементи проекту мають свої чіткі функції та області за які вони відповідають. За бажанням проект може бути доповнений новими функціями та вдосконалений за потребою.

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМИ

3.1 Вибір мови програмування

Мобільний додаток – це програмне забезпечення, що розроблено та призначено для роботи на мобільних пристроях.

Мобільні додатки можна розподілити за такими факторами:

1. Технології, на яких вони побудовані;
2. Платформи, для яких вони побудовані;
3. Користувачі, для яких призначено додаток;
4. Вимоги до продуктивності програми.

В залежності від цих факторів мобільні додатки класифікуються на три категорії, які наведені на рисунку 3.1:

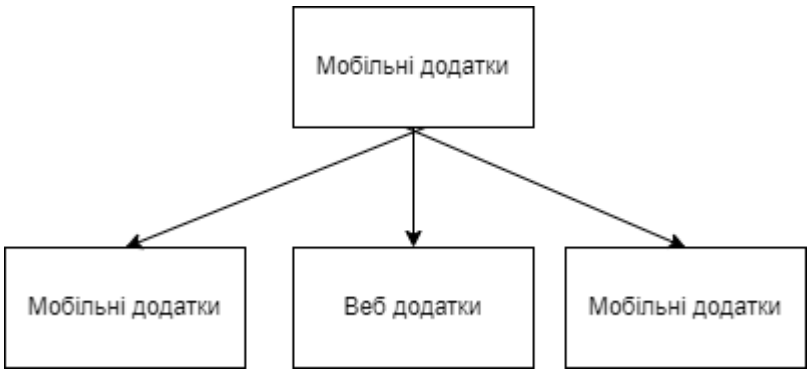

Рисунок 3.1 – Види мобільних додатків

Мобільний додаток керування аудіофайлами, що розробляється під час виконання дипломної роботи за класифікацією відноситься до нативних додатків. Додатки такого типу створені та націлені на певний пристрій, що означає певну операційну систему, у даному випадку систему Android. Перевага додатків такого роду э у тому, що вони легко спілкуються з пристроєм, відображення сторінок додатку триває без будь-яких проблем або збивань. Ці програми є більш зручними для користувачів, простіші в експлуатації та можуть легко переносити швидку діяльність користувача.

Недоліками розробки таких програм є те, що для їх розробки використовуються складні мови програмування такі як Java, Kotlin, Swift, Python, C #. Такі програми

працюють лише на окремих платформах, до цього ж їх обслуговування є також дороговитратним заняттям.

Також мобільний додаток керування аудіофайлами можна віднести до такої категорії мобільних додатків як контент-додатки. Вони націлені на надання таких послуг як прослуховування музики, перегляд фільмів, фотографій, цифрових книг. Сьогодні такі додатки різко набирають свою популярність.

Для написання мобільного додатку слід приділити особливу увагу щодо вибору мови програмування. Для такої задачі є багато варіантів, тож є серед чого вибирати. В одному випадку, для деяких додатків розробнику можуть зовсім не знадобитися більшість функцій певної мови, в іншому ж для однієї програми може знадобитися навіть більше ніж одна мова програмування. Головним питанням щодо мов програмування, що постало перед мною був вибір серед Java та Kotlin.

Мова Kotlin сьогодні стрімко набирає своєї популярності. За словами фанатів синтаксис Kotlin є простішим, чистішим, що призводить до меншого роздування коду. Це допомагає більше зосередитися на розв’язанні актуальної проблеми, а не на складному синтаксисі. Крім цього Kotlin та Java можна використовувати разом в одному проекті, і це робить її дійсно потужним інструментом.

З іншого боку, мова Java є однією з найпопулярніших мов сьогодні. На електронних ресурсах можна знайти безліч інформації про алгоритми та способи кодування саме на Java. У мови є велика та розвинута спільнота розробників, і це означає, що можна легко отримати шукану інформацію.

Отже, мій вибір зупинився на мові програмування Java, адже розробляючи програму для мобільних пристроїв за допомогою Java, я можу створити будь-який додаток, який тільки зможу придумати.

3.2 Опис алгоритмічних структур

Однією з цілей створення даного проекту було створення зручного алгоритму рекомендацій на основі вподобань користувача. Для реалізації цієї мети було вивчено способи, якими можна дістати інформацію з аудіофайлів. Аудіофайли формату mp3

вміщують в себе інформацію, яка розповідає про нього набагато більше, ніж просто назву. Метадані – це інформація про ім’я композитора, автора аранжировки, тривалість, альбом, кількість ударів в секунду і навіть текст аудіофайлу. До музичних файлів додаються спеціальні блоки інформації ID3tag, tag – ярлик або ж мітка.

Для того, щоб не викликати збою в сумісності мітка додається у кінець файлу, завдяки чому може з легкістю ігноруватися за потреби. Це дає можливість додавати різну потрібну інформацію до будь-якого музичного файлу[6].

Під час створення алгоритму рекомендацій враховуються метадані останнього доданого аудіофайлу до списку «улюблених», завдяки чому створюються рекомендації, які схожі за своїми характеристиками на аудіофайл, що сподобався користувачу.

Алгоритм створення рекомендацій наведено нижче:

```
/*Функція створення списку рекомендацій
* Повертається значення типу ArrayList<File> - список з ауділфайлами рекомендацій
* Вхідні дані: програвання користувачем аудіофайлів,
* додавання користувачем аудіофайлів до списку улюблених*/
private ArrayList<File> createRecomendation() {
    ArrayList<File> recomendation = new ArrayList<>();
    //Often played tracks
    for(Track track : AllSongs.allTracks){
      if(track.getCountOfPlays() > 3 && !recomendation.contains(track.getFile()))
      {
        recomendation.add(track.getFile());
      }
    }
    //Last favorite song
    ArrayList<File> favoriteArrayList = FolderFilesManager.readFromFolder("favoritesong");
    if(!favoriteArrayList.isEmpty() && !recomendation.contains(favoriteArrayList.get(favoriteArrayList.size() - 1))) {
      recomendation.add(favoriteArrayList.get(favoriteArrayList.size() - 1));
    }
    if(!favoriteArrayList.isEmpty()){

      MediaMetadataRetriever mediaMetadataRetriever = new MediaMetadataRetriever();
      mediaMetadataRetriever.setDataSource(favoriteArrayList.get(favoriteArrayList.size() - 1).getPath());
      String artist = mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_ARTIST);
      String genre = mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_GENRE);
      String duration = mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_DURATION);
      String album = mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_ALBUM);
      String date = mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_DATE);
```

```
        for (Track track : AllSongs.allTracks) {
            if (!track.getTitle().equals(favoriteArrayList.get(favoriteArrayList.size() - 1).getName())) {
                if (track.getArtist().equals(artist) && track.getArtist() != "Unknown artist") {
                    recomendation.add(track.getFile());
                    continue;
                }
                if (track.getGenre().equals(genre) && track.getGenre() != "Unknown genre") {
                    recomendation.add(track.getFile());
                    continue;
                }
                if (track.getDuration().equals(duration) && track.getDuration() != "Unknown duration") {
                    recomendation.add(track.getFile());
                    continue;
                }
                if (track.getAlbum().equals(album) && track.getAlbum() != "Unknown album") {
                    recomendation.add(track.getFile());
                    continue;
                }
                if (track.getDate().equals(date) && track.getDate() != "Unknown date") {
                    recomendation.add(track.getFile());
                    continue;
                }
            }
        }
        mediaMetadataRetriever.release();

    }
    return recomendation;

}
```

Аналізуючи автора, альбом, кількість ударів в секунду, дату створення, жанр, тривалість можна порекомендуавти користувачу аудіофайли зі схожими характеристиками. Окрім цього до уваги також береться кількість натискань на кожний аудіофайл. Якщо користувач декілька разів прослуховував один і той самий аудіофайл можна сказати, що від до вподоби користувачу і порекомендувати для прослуховування наступного разу.

На основі аналізу цих даних формується список рекомендацій, який відображається у спеціальному розділі у вигляді списку назв аудіофайлів, які можна увімкнути для програвання.

Наступна таблиця зображує каскадне представлення алгоритму, що представлено на таблиці 3.1:

Таблиця 3.1 – Каскадне представлення алгоритму рекомендацій

Задача	Підзадачі	
Створення списку рекомендацій	Рекомендації по кількості прослуховувань	Підрахувати кількість прослуховувань для окремого аудіофайлу
		Додати аудіофайли з великою кількістю прослуховувань до списку рекомендацій
	Рекомендації на основі аналізу метаданих усіх наявних аудіофайлів	Проаналізувати метаданні останнього улюбленого аудіофайлу
		Проаналізувати метаданні усіх наявних аудіофайлів
		Порівняти наявні метадані
	Виведення результату	Створити список з усіх наявних рекомендацій
		Повернути готовий список

РОЗДІЛ 4 ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ

Тестування – процес виявлення помилок у програмному забезпеченні з метою їх подальшого виправлення. Головною метою тестування є доведення програми до стану виведення коректної інформації при певних вхідних даних [7]. Розділимо тестування програмної системи на такі категорії:

- Тестування чорною скринькою(без доступа до тексту програми);
- Тестування білою скринькою(з доступом до тексту програми).

4.1 Тестування білою скринькою

Проведемо тестування функції createRecomendation(), яка створює список рекомендованих аудіофайлів для користувача.

Вхідні дані: ArrayList<Track> allTracks = new ArrayList<>() – список, що складається з об’єктів класу Track, які містять інформацію про окремий аудіофайл.

ArrayList<File> favoriteArrayList = FolderFilesManager.readFromFolder("favoritesong") – список улюблених аудіофайлів користувача.

Вихідні дані: список аудіофайлів типу ArrayList<File> - список з аудіофайлами рекомендацій.

Нижче наведено текст функції create recomendstion():

```
/*Функція створення списку рекомендацій
* Повертається значення типу ArrayList<File> - список з ауділфайлами рекомендацій
* Вхідні дані: програвання користувачем аудіофайлів,
* додавання користувачем аудіофайлів до списку улюблених*/
private ArrayList<File> createRecomendation() {
    ArrayList<File> recomendation = new ArrayList<>();
    //Often played tracks
    for(Track track : AllSongs.allTracks){
      if(track.getCountOfPlays() > 3 && !recomendation.contains(track.getFile()))
      {
        recomendation.add(track.getFile());
      }
    }
    //Last favorite song
    ArrayList<File> favoriteArrayList = FolderFilesManager.readFromFolder("favoritesong");
    if(!favoriteArrayList.isEmpty() && !recomendation.contains(favoriteArrayList.get(favoriteArrayList.size() - 1))) {
```

```
        recomendation.add(favoriteArrayList.get(favoriteArrayList.size() - 1));
    }
    if(!favoriteArrayList.isEmpty()){

        MediaMetadataRetriever mediaMetadataRetriever = new MediaMetadataRetriever();
        mediaMetadataRetriever.setDataSource(favoriteArrayList.get(favoriteArrayList.size() - 1).getPath());
        String artist = mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_ARTIST);
        String genre = mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_GENRE);
        String duration = mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_DURATION);
        String album = mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_ALBUM);
        String date = mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_DATE);

        for (Track track : AllSongs.allTracks) {
            if (!track.getTitle().equals(favoriteArrayList.get(favoriteArrayList.size() - 1).getName())) {
                if (track.getArtist().equals(artist) && track.getArtist() != "Unknown artist") {
                    recomendation.add(track.getFile());
                    continue;
                }
                if (track.getGenre().equals(genre) && track.getGenre() != "Unknown genre") {
                    recomendation.add(track.getFile());
                    continue;
                }
                if (track.getDuration().equals(duration) && track.getDuration() != "Unknown duration") {
                    recomendation.add(track.getFile());
                    continue;
                }
                if (track.getAlbum().equals(album) && track.getAlbum() != "Unknown album") {
                    recomendation.add(track.getFile());
                    continue;
                }
                if (track.getDate().equals(date) && track.getDate() != "Unknown date") {
                    recomendation.add(track.getFile());
                    continue;
                }
            }
        }
        mediaMetadataRetriever.release();

    }
    return recomendation;

}
```

Зображення блок-схеми функції createRecomendation наведено на рисунку 4.1:

Рисунок 4.1 – Блок-схема функції createRecomendation()

Тести до функції createRecomendation наведено у таблиці 4.1:

Таблиця 4.1 Тести до функції createRecomendation()

№ тесту	Вхідні значення		Вихідні значення
	Список наявних аудіофайлів	Улюблений аудіофайл	
1	2	3	4
1.	[0] = “01_nas_the_genesis” [1] = “01_pink_floyd_in _the_flesh” [2] = “02_made_in_ukraine_ rozpryagaite_hloptsi_konei” [3] = “02_pink_floyd_the _thin_ice” [4] = “03_kalush_kent” [5] = “03_made_in_ukraine_natalya_falion_sama_faina” [6] = “04_kalush_zori” [7] = “04_okean_elzi _nezalezhnist” [8] = “05_kalush_mosti” [9] = “10_okean_elzi_na_nebi” [10] = “12_okean_elzi_koli_navkolo _ni_dushi” [11] = “12_okean_elzi_koli _tebe_nema” Параметр «Кількість натискань» для кожного аудіофайлу < 3	Порожній список – немає жодного улюбленого трека	Порожній список

Продовження табл. 4.1

1	2	3	4
2.	[0] = “01_nas_the_genesis” [1] = “01_pink_floyd_in _the_flesh” [2] = “02_made_in_ukraine_ rozpryagaite_hloptsi_konei” [3] = “02_pink_floyd_the _thin_ice” [4] = “03_kalush_kent” [5] = “03_made_in_ukraine_natalya_falion_sama_ faina” [6] = “04_kalush_zori” [7] = “04_okean_elzi _nezalezhnist” [8] = “05_kalush_mosti” [9] = “10_okean_elzi_na_nebi” [10] = “12_okean_elzi_koli_navkolo _ni_dushi” [11] = “12_okean_elzi_koli _tebe_nema” Параметр «Кількість натискань» для кожного аудіофайлу < 3	[sizeOfArray - 1] = “04_kalush_zori.mp3”	[0] = “04_kalush_zori” [1] =“01_nas_the_genesis” [2] =“03_kalush_kent” [3] =“05_kalush_mosti”

Продовження табл. 4.1

1	2	3	4
3.	[0] = “01_nas_the_genesis” [1] = “01_pink_floyd_in _the_flesh” [2] = “02_made_in_ukraine_ rozpryagaite_hloptsi_konei” [3] = “02_pink_floyd_the _thin_ice” [4] = “03_kalush_kent” [5] = “03_made_in_ukraine_natalya_falion_sama_faina” [6] = “04_kalush_zori” [7] = “04_okean_elzi _nezalezhnist” [8] = “05_kalush_mosti” [9] = “10_okean_elzi_na_nebi” [10] = “12_okean_elzi_koli_navkolo _ni_dushi” [11] = “12_okean_elzi_koli _tebe_nema” Параметр «Кількість натискань» для треку “04_kalush_zori” > 3	Порожній список – немає жодного улюбленого трека	Список з одного елементу: [0] = “04_kalush_zori”

Тестування методом покриття операторів описано у табл. 4.2:

Таблиця 4.2 Тестування методом покриття операторів

№ тесту	Номер оператора																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	+	+	+	-	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+
2	+	+	+	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
3	+	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+

При виконанні тестів, кожен оператор виконався хоча б один раз.

Тестування методом покриття рішень описано у табл. 4.3:

Таблиця 4.3. Тестування методом покриття рішень

№ тесту	Номер рішення										
	3	5	7	8	12	13	14	15	16	17	18
1	+	+	+	+	-	-	-	-	-	-	-
2	+	+	+	+	+	+	+	+	+	+	+
3	+	+	+	+	-	-	-	-	-	-	-

При виконанні тестів, кожен напрямок переходу був реалізований хоча б раз.

4.2 Тестування чорною скринькою

Тестування чорною скринькою проведемо за результатами опису роботи функцій програми. Пройдемо декілька наступних пунктів.

Натиснемо на іконку додатку в емуляторі смартфону під керуванням системи Android, тим самим завантаживши його.

Зображення іконки додатку наведено на рисунку 4.2:

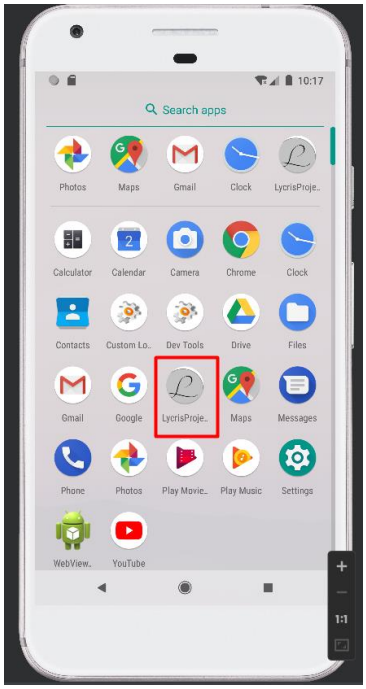

Рисунок 4.2 – Зображення іконки додатку

Спробуємо виконати пошук аудіофайлу з назвою «02_pink_floyd_the_thin_ice». Зображення поля для пошуку наведено на рисунку 4.3:

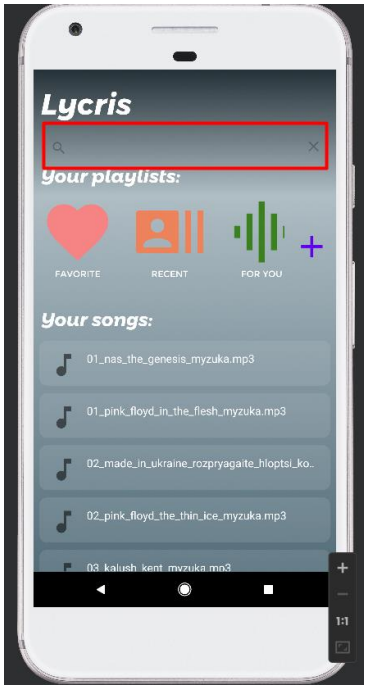

Рисунок 4.3 – Зображення поля для пошуку

Введемо ключове слово «pink», одразу ж виконується пошук. Зображення виконаного пошуку наведено на рисунку 4.4:

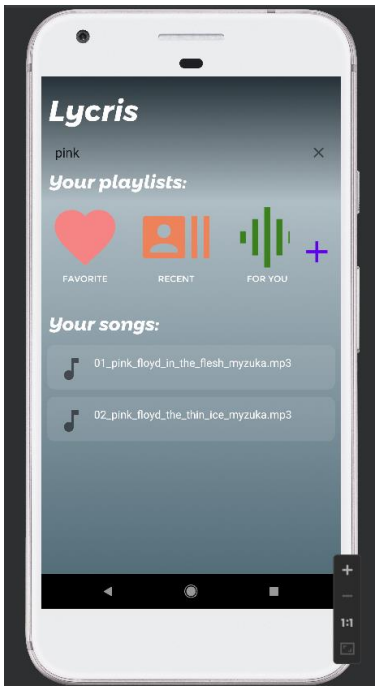

Рисунок 4.4 – Зображення поля для пошуку

Натиснемо на шуканий другий за порядком аудіофайл з назвою. Зображення відкритої сторінки плеєру на рисунку 4.5:

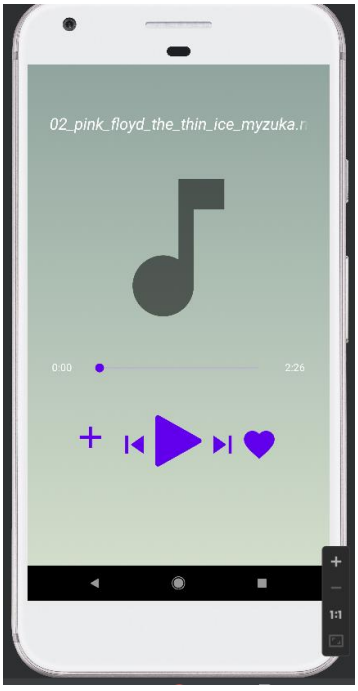

Рисунок 4.5 – Зображення сторінки плеєру

Спробуємо додати аудіофайл до списку «улюблених». Зображення сторінки плеєру після натискання кнопки «додати до улюблених» наведено на рисунку 4.6:

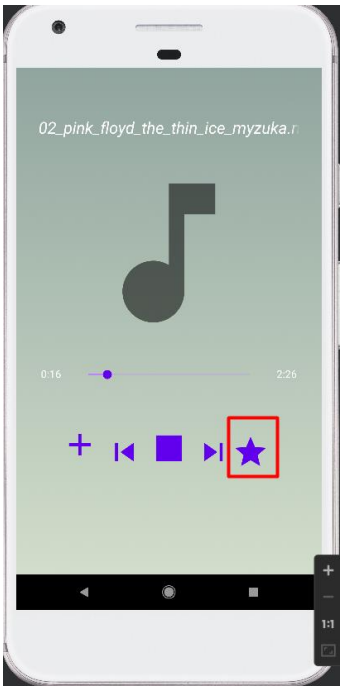

Рисунок 4.6 – Зображення сторінки після натискання кнопки «додати до улюблених»

Перейдемо до папки «улюблені». Зображення натискання на папку «улюблені» наведено на рисунку 4.7:

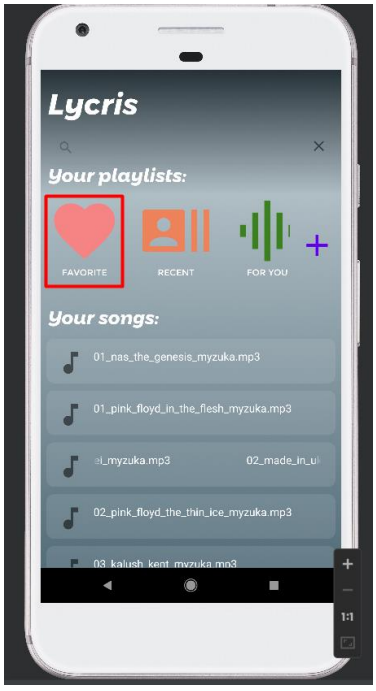

Рисунок 4.7 – Зображення головної сторінки, виділена папка «улюблені»

Перевіримо чи додався аудіофайл у папку «улюблені». Зображення списку аудіофайлів папки «улюблені» наведено на рисунку 4.8:

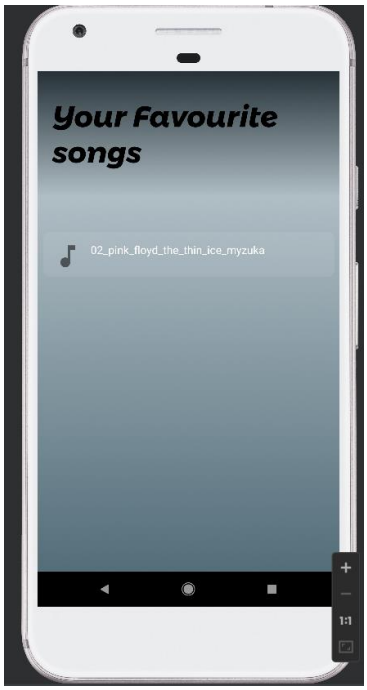

Рисунок 4.8 – Зображення списку аудіофайлів папки «улюблені»

Перейдемо до папки «рекомендацій» та перевіримо рекомендації від програми. Зображення списку аудіофайлів папки «рекомендації» наведено на рисунку 4.9:

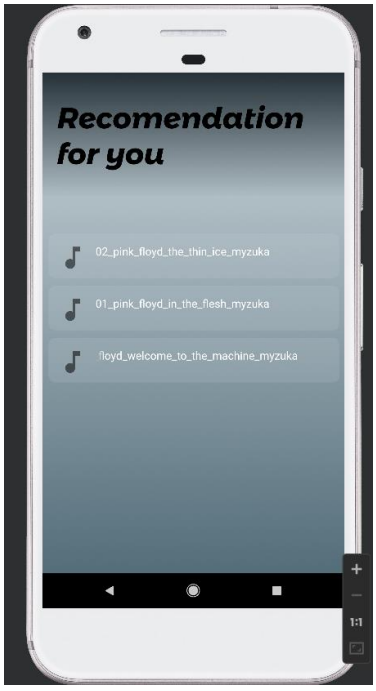

Рисунок 4.9 – Зображення списку аудіофайлів папки «рекомендації»

Як бачимо, рекомендації дійсно мають схожий характер з аудіофайлами зі списку «улюблених». Спробуємо створити нову власну папку. Зображення створення нової папки наведено на рисунку 4.10:

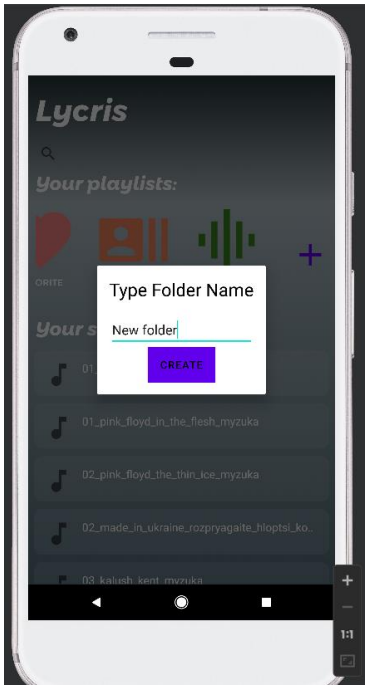

Рисунок 4.10 – Зображення створення нової власної папки

Зображення відображення нової папки наведено на рисунку 4.11:

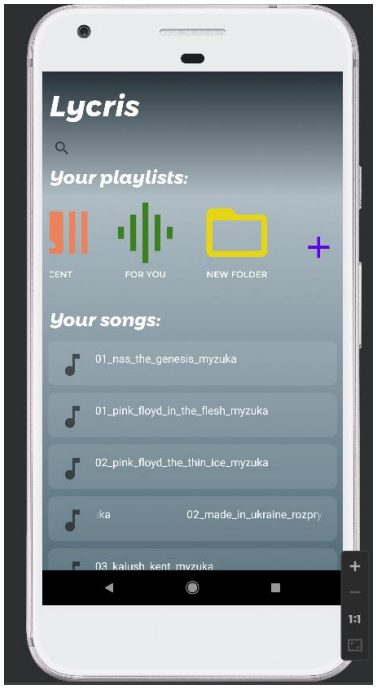

Рисунок 4.11 – Зображення відображення нової власної папки

Спробуємо додати аудіофайл до власної папки. Зображення додавання аудіофайлу до власної наведено на рисунку 4.12:

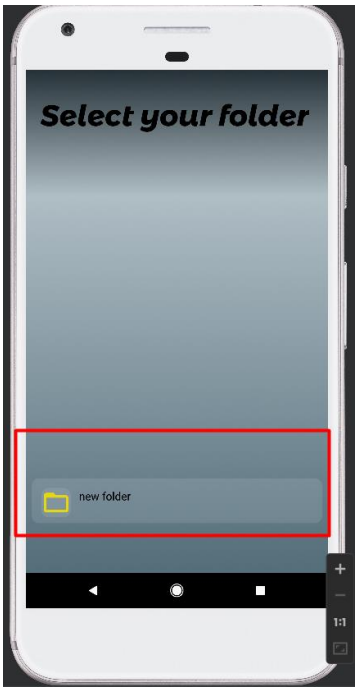

Рисунок 4.12 – Зображення додавання аудіофайлу до власної папки

Зображення відображення аудіофайлу у власній папці наведено на рисунку 4.13:

Рисунок 4.13 – Зображення відображення аудіофайлу у власній папці

Також за необхідності аудіофайл можна видалити з власної папки. Так само можна видалити усю папку. Спробуємо відкрити папку «нещодавні» і перевіримо відображення аудіофайлів, які прослуховувались.

Зображення відображення аудіофайлів у папці «нещодавні» наведено на рисунку 4.14:

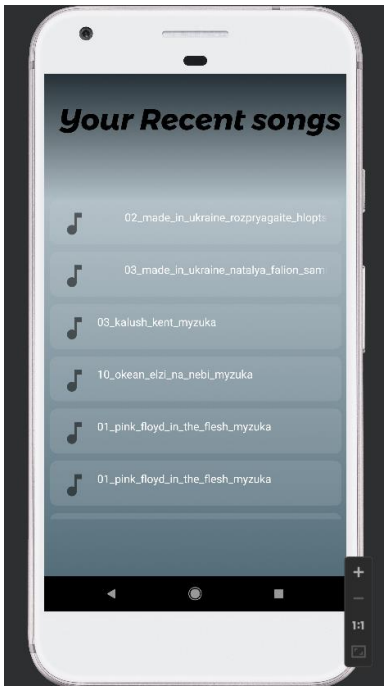

Рисунок 4.14 – Зображення відображення аудіофайлів у папці «нещодавні»

Як бачимо основні функції працюють без помилок, перейдемо до безпосереднього налагодження програми.

4.3 Налагодження програми

Налагодження програми було націлене на виявлення синтаксичних, семантичних та логічних помилок [8]. Під час налагодження було знайдено декілька помилок, які не були помічені під час інших етапів розробки програмного продукту. Зокрема, під час роботи програми було знайдено помилки звернення до пустого списку, відсутність деяких перевірок на існування конкретних елементів у списках. Також знайдено специфічну помилку про відсутність прав додатку на роботу з внутрішнім сховищем пристрою, для вирішення якої знадобилось створювати конструкцію для запиту користувачеві про надання такого дозволу.

Основні помилки були виправлені завдяки зручному налагоджувачу, що наявний у середовищі розробки AndroidStudio. Виставлялись точки преривання та переглядались значення змінних у конкретний час виконання програми. Це допомогло не тільки усунути наявні помилки, а й вдосконалити код, запобігаючи виникненню нових. Протокол налагодження програми наведений у таблиці 4.4:

Таблиця 4.4 Протокол налагодження програми

Опис помилки	Опис ситуації	Способи усунення	Дії, що були застосовані для усунення
1	2	3	4
Звернення до списку, який не має елементів	Якщо користувач не надав права на роботу зі сховищем пристрою, то програма не може отримати доступ до аудіофайлів на пристрої	Додати перевірку на наповненість списку аудіофайлів Припиняти роботу додатку без надання дозволу	Додати перевірку на наповненість списку if (files != null)
Користувач міг додавати аудіофайли до списку «нещодавніх»	При виборі користувачем папки у яку він хотів додати аудіофайл була можливість додати до автоматично створюваного списку «нещодавніх»	Приховати додавання до такого списку Видалити відображення цього списку	Список папки «нещодавніх» аудіофайлів був видалений для відображення користувачеві при обранні папки

Продовження табл. 4.4

1	2	3	4
Валідація даних при створенні нової папки	Користувач міг ввести занадто велику назву для папки	Обмежити кількість символів для назви Збільшити область для відображення назви	Було обмежено кількість символів до 12 для коректного відображення
Назва аудіофайлу була занадто великою для відображення у списку	Аудіофайл міг містити велику кількість символів, в результаті чого користувач бачив не усю назву	Змінити відображення назви на плаваючий текст, щоб назва з’являлась послідовно Обмежити кількість відображених символів	Було змінено відображення на плаваюче
Метадані деяких файлів не були заповнені	Рекомендації щодо аудіофайлів без метаданих не можливо було створити	Видалити рекомендації для таких аудіофайлів Додати обмеження по рекомендаціям	Було створено обмеження за яких аудіофайли такого виду ігнорувались під час створення рекомендацій

Висновки на основі тестування

Тестування – один з найважливіших етапів у розробці мобільних додатків, адже розробка мобільного додатку ведеться на комп'ютері, а виконання самої програми буде відбуватись на смартфоні. Саме тому додаток під час тестування було завантажено на декілька різних пристроїв під керуванням системи Android. Саме відображення на реальних пристроях дозволило визначити декілька додаткових помилок у програмній системі.

ЗАГАЛЬНІ ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ

Результатом виконаної роботи є мобільний додаток, який реалізовує функції керування аудіофайлами, з урахуванням музичного смаку користувача. Розроблено дружній інтерфейс користувача, який інтуітивно допомагає швидко пристосуватись до роботи з додатком. Програму можна з легкістю передати на пристрій під керуванням системи Android і вона буде виконувати свої функції. Проектування виконане таким чином, що систему можна вдосконалити для випуску наступних версій.

Область застосування – користувачі, які потребують зручний інструмент для керування аудіофайлами з урахуванням музичного смаку.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Аналіз музичного смаку від Spotify [Електронний ресурс] – https://towardsdatascience.com/analyzing-music-taste-64202f602bcd
2. Як Spotify використовує AI, ML та Big Data у системах рекомендацій [Електронний ресурс] - https://prjctrmag.com/ai-in-spotify
3. Аналоги сервісів керування аудіофайлами [Електронний ресурс] – https://nachasi.com/cards/2020/01/16/what-is-spotify/
4. І62 Інженерія програмного забезпечення [Текст]: навчальний посібник – В. М. Горячкін, О. В. Горбова, О. С. Куроп'ятник; Український державний університет науки і технологій. – Дніп-ро, 2022. – 140 с – https://lider.diit.edu.ua/pluginfile.php/118953/mod_resource/content/1/Посібник_ОСБАкалавр-2022%20студ.pdf
5. Проектування інтерфейсу користувача [Електронний ресурс] – http://elar.khnu.km.ua/jspui/bitstream/123456789/1415/2/Rozdil1.pdf
6. Класифікація метаданих великих даних [Електронний ресурс] – http://dspace.nbuv.gov.ua/bitstream/handle/123456789/162702/04-Zakharova.pdf?sequence=1
7. Тестування програмного забезпечення [Електронний ресурс] – https://lider.diit.edu.ua/pluginfile.php/113432/mod_resource/content/1/Основы%20инженерии%20качества%20программных%20систем.pdf
8. Програмування мобільних пристроїв: навчальний посібник для дистанційного навчання / К.Т. Кузьма. – Миколаїв: СПД Румянцева Г. В., 2021. – 128 с. – http://dspace.mdu.edu.ua/jspui/bitstream/123456789/907/1/Кузьма_Програмування%20мобільних%20пристроїв.pdf

ДОДАТКИ

ТЕКСТ ПРОГРАМИ

Екран AddToFolderActivity.

Текст файлу AddToFolderActivity.java:

```
package com.example.lycrisproject.activities;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import com.example.lycrisproject.FolderFilesManager;
import com.example.lycrisproject.R;
import com.example.lycrisproject.model.AudioItem;

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

public class AddToFolderActivity extends AppCompatActivity {

    private ListView listView;
    private TextView txtinvite;
    private int size;
    private String[] items;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_to_folder);

        SetUpViews();

        displayFolder(AudioItem.audioItemList);
        if(items.length == 2 ) Toast.makeText(this, "You have no folders", Toast.LENGTH_LONG).show();
    }

    private void displayFolder(List<AudioItem> audioItemList) {
        size = audioItemList.size();
        items = new String[size];
```

```
    for(int i = 0; i < size; i++)
    {
        items[i] = audioItemList.get(i).getTitle();
    }

    customAdapter customAdapter = new customAdapter();
    listView.setAdapter(customAdapter);
    txtinvite.setText("Select your folder");


    listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
            Intent intent = getIntent();
            Bundle bundle = intent.getExtras();
            File song = (File) bundle.get("song");

            ArrayList<File> NewUserFolderArray = FolderFilesManager.readFromFolder(AudioItem.audioItemList.get(i).getTitle());
            NewUserFolderArray.add(song);
            FolderFilesManager.writeToFolder(NewUserFolderArray,AudioItem.audioItemList.get(i).getTitle() );
            Toast.makeText(AddToFolderActivity.this, "Added", Toast.LENGTH_SHORT);
            finish();
        }
    });
}
class customAdapter extends BaseAdapter {

    @Override
    public int getCount() {
        return size;
    }

    @Override
    public Object getItem(int i) {
        return null;
    }

    @Override
    public long getItemId(int i) {
        return 0;
    }

    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        View myView  = getLayoutInflater().inflate(R.layout.folder_item, null);
        TextView foldername = myView.findViewById(R.id.txtFolderName);
        if(i == 0 || i == 1 || i == 2) {
            myView.setClickable(false);
            myView.setVisibility(View.INVISIBLE);
```

```
            foldername.setText(items[i]);
            foldername.setSelected(false);
        }else
        {
            foldername.setSelected(true);
            foldername.setText(items[i]);
        }

        return myView;
    }
  }
  private void SetUpViews(){
    listView = findViewById(R.id.folderViewList);
    txtinvite = findViewById(R.id.folderinvite);
  }

}
```

Екран FolderActivity.

Текст файлу FolderActivity.java:

```
package com.example.lycrisproject.activities;

import androidx.appcompat.app.AppCompatActivity;

import android.app.Dialog;
import android.content.Intent;
import android.media.MediaMetadataRetriever;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import com.example.lycrisproject.FolderFilesManager;
import com.example.lycrisproject.R;
import com.example.lycrisproject.model.AllSongs;
import com.example.lycrisproject.model.AudioItem;
import com.example.lycrisproject.model.Track;

import java.io.File;
import java.util.ArrayList;

public class FolderActivity extends AppCompatActivity {
```

```
private String[] items;
private ListView listView;
private TextView categoryName;
private Button delBtnFolder;

private int catId;

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_folder);

   listView = findViewById(R.id.folderList);
   categoryName = findViewById(R.id.categoryName);
   delBtnFolder = findViewById(R.id.delButtonFolder);

   Intent i = getIntent();
   Bundle bundle = i.getExtras();

   catId = bundle.getInt("categoryId", 0);

   displayParam(catId);

}
@Override
protected void onResume() {
   super.onResume();
   listView.invalidateViews();
   displayParam(catId);
}

@Override
protected void onDestroy() {
   super.onDestroy();

}

private void displaySongs(ArrayList<File> mySongs)
{
   items = new String[mySongs.size()];
   for(int i = 0; i < mySongs.size(); i++)
   {
      items[i] = mySongs.get(i).getName().replace(".mp3", "").replace(".wav", "") ;

   }

   customAdapter customAdapter = new customAdapter();
   listView.setAdapter(customAdapter);
```

```
    listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
            String songName =  (String) listView.getItemAtPosition(i);
            startActivity( new Intent(getApplicationContext(), PlayerActivity.class)
                    .putExtra("songs", mySongs)
                    .putExtra("songname", songName)
                    .putExtra("pos", i));
        }
    });

}
class customAdapter extends BaseAdapter
{
    @Override
    public int getCount() {
        return items.length;
    }

    @Override
    public Object getItem(int i) {
        return null;
    }

    @Override
    public long getItemId(int i) {
        return 0;
    }

    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        View myView;
        if (catId > 2)
        {
            myView  = getLayoutInflater().inflate(R.layout.list_item_with_del, null);
            TextView textsong = myView.findViewById(R.id.txtSongName);
            Button delbtn = myView.findViewById(R.id.delButtonList);
            //delbtn.setVisibility(View.INVISIBLE);
            delbtn.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    showConfirmDialog(i);
                }
            });
            textsong.setSelected(true);
            textsong.setText(items[i]);
        }
        else
```

```
        {
            myView  = getLayoutInflater().inflate(R.layout.list_item, null);
            TextView textsong = myView.findViewById(R.id.txtSongName);
            textsong.setSelected(true);
            textsong.setText(items[i]);
        }

        return myView;
    }
}

private void showConfirmDialog(int i) {
    final Dialog dialog = new Dialog(FolderActivity.this);
    dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
    dialog.setCancelable(true);
    dialog.setContentView(R.layout.dialog_item);



    final EditText folderName = dialog.findViewById(R.id.name_et);
    folderName.setVisibility(View.INVISIBLE);
    final Button dialogbtn = dialog.findViewById(R.id.submit_button);
    dialogbtn.setText("Delete");
    final TextView dialogtxt= dialog.findViewById(R.id.DialogText);
    dialogtxt.setText("Delete song?");

    dialogbtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            ArrayList<File> NewUserFolderArray = FolderFilesManager.readFromFolder(AudioItem.audioItemList.get(catId).getTitle());
            NewUserFolderArray.remove(i);
            FolderFilesManager.writeToFolder(NewUserFolderArray,AudioItem.audioItemList.get(catId).getTitle() );

            dialog.dismiss();
            displayParam(catId);
        }
    });

    dialog.show();
}

private void displayParam(int catId)
{
    switch (catId)
    {
        case 0:
        {
            categoryName.setText("Your Favourite songs");
            ArrayList<File> favoriteArrayList = FolderFilesManager.readFromFolder("favoritesong");
            displaySongs(favoriteArrayList);
```

```
    if(favoriteArrayList.size() == 0) Toast.makeText(this, "You have no favourite songs", Toast.LENGTH_SHORT).show();
    break;
}
case 1:
{
    categoryName.setText("Your Recent songs");
    ArrayList<File> recentArrayList = FolderFilesManager.readFromFolder("recentsong");
    displaySongs(recentArrayList);
    if(recentArrayList.size() == 0) Toast.makeText(this, "You have no recent songs", Toast.LENGTH_SHORT).show();
    break;
}
case 2:
{
    categoryName.setText("Recomendation\nfor you");
    ArrayList<File> recomendation = createRecomendation();
    displaySongs(recomendation);
    break;
}
default:{
    delBtnFolder.setVisibility(View.VISIBLE);
    delBtnFolder.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            final Dialog dialog = new Dialog(FolderActivity.this);
            dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
            dialog.setCancelable(true);
            dialog.setContentView(R.layout.dialog_item);


            final EditText folderName = dialog.findViewById(R.id.name_et);
            folderName.setVisibility(View.INVISIBLE);
            final Button dialogbtn = dialog.findViewById(R.id.submit_button);
            dialogbtn.setText("Delete");
            final TextView dialogtxt= dialog.findViewById(R.id.DialogText);
            dialogtxt.setText("Delete folder?");

            dialogbtn.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    dialog.dismiss();
                    FolderFilesManager.deleteFolder(AudioItem.audioItemList.get(catId).getTitle());
                    AudioItem.audioItemList.remove(catId);
                    finish();
                }
            });
            dialog.show();
        }
    });
    categoryName.setText("Songs in folder \"" + AudioItem.audioItemList.get(catId).getTitle() + "\"");
```

```
            ArrayList<File> NewUserFolderArray = FolderFilesManager.readFromFolder(AudioItem.audioItemList.get(catId).getTitle());
            displaySongs(NewUserFolderArray);
            if(NewUserFolderArray.size() ==0) Toast.makeText(this, "You have no songs in this folder", Toast.LENGTH_SHORT).show();
        }



    }
}


private ArrayList<File> createRecomendation() {
    ArrayList<File> recomendation = new ArrayList<>();
    //Often played tracks
    for(Track track : AllSongs.allTracks){
        if(track.getCountOfPlays() > 3 && !recomendation.contains(track.getFile()))
        {
            recomendation.add(track.getFile());
        }
    }
    //Last favorite song
    ArrayList<File> favoriteArrayList = FolderFilesManager.readFromFolder("favoritesong");
    if(!favoriteArrayList.isEmpty() && !recomendation.contains(favoriteArrayList.get(favoriteArrayList.size() - 1))) {
        recomendation.add(favoriteArrayList.get(favoriteArrayList.size() - 1));
    }
    if(!favoriteArrayList.isEmpty()){

        MediaMetadataRetriever mediaMetadataRetriever = new MediaMetadataRetriever();
        mediaMetadataRetriever.setDataSource(favoriteArrayList.get(favoriteArrayList.size() - 1).getPath());
        String artist = mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_ARTIST);
        String genre = mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_GENRE);
        String duration = mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_DURATION);
        String album = mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_ALBUM);
        String date = mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_DATE);

        for (Track track : AllSongs.allTracks) {
            if (!track.getTitle().equals(favoriteArrayList.get(favoriteArrayList.size() - 1).getName())) {
                if (track.getArtist().equals(artist) && track.getArtist() != "Unknown artist") {
                    recomendation.add(track.getFile());
                    continue;
                }
                if (track.getGenre().equals(genre) && track.getGenre() != "Unknown genre") {
                    recomendation.add(track.getFile());
                    continue;
                }
                if (track.getDuration().equals(duration) && track.getDuration() != "Unknown duration") {
                    recomendation.add(track.getFile());
                    continue;
```

```
                }
                if (track.getAlbum().equals(album) && track.getAlbum() != "Unknown album") {
                    recomendation.add(track.getFile());
                    continue;
                }
                if (track.getDate().equals(date) && track.getDate() != "Unknown date") {
                    recomendation.add(track.getFile());
                    continue;
                }
            }
        }
        mediaMetadataRetriever.release();

    }
    return recomendation;

  }

  /* private int compareSymb(String a, String b){
      a = a.replaceAll("[^A-Za-za-яA-Я0-9]+", "");
      b = b.replaceAll("[^A-Za-za-яA-Я0-9]+", "");


      MediaMetadataRetriever mediaMetadataRetriever = new MediaMetadataRetriever();
      mediaMetadataRetriever.setDataSource(AllSongs.allSongs.get(3).getPath());
      String album = mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_GENRE);
      return 0;
   }*/

}
```

Екран MainActivity.

Текст файлу MainActivity.java:

```
package com.example.lycrisproject.activities;

import static com.example.lycrisproject.model.AudioItem.audioItemList;

import android.Manifest;
import android.animation.Animator;
import android.app.Dialog;
import android.content.Intent;
import android.media.MediaMetadataRetriever;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
```

```
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.SearchView;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.lycrisproject.FolderFilesManager;
import com.example.lycrisproject.R;
import com.example.lycrisproject.adapter.AudioItemAdapter;
import com.example.lycrisproject.model.AllSongs;
import com.example.lycrisproject.model.AudioItem;
import com.example.lycrisproject.model.Track;
import com.karumi.dexter.Dexter;
import com.karumi.dexter.PermissionToken;
import com.karumi.dexter.listener.PermissionDeniedResponse;
import com.karumi.dexter.listener.PermissionGrantedResponse;
import com.karumi.dexter.listener.PermissionRequest;
import com.karumi.dexter.listener.single.PermissionListener;

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

public class MainActivity extends AppCompatActivity {

    private AudioItemAdapter audioItemAdapter;
    private int foldersCount;
    private customAdapter customAdapter;

    private boolean isFilter = false;
    private final ArrayList<File> IsFilterArray = new ArrayList<>();

    private SearchView searchView;

    private ListView listView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        SetUpViews();
```

```
        SetUpRuntimePermission();

        SetJAudioItemRecycler(audioItemList);

        AllSongs.allSongs = findSong(Environment.getExternalStorageDirectory());

        MediaMetadataRetriever mediaMetadataRetriever = new MediaMetadataRetriever();
        for(int i = 0; i < AllSongs.allSongs.size(); i++){
            mediaMetadataRetriever.setDataSource(AllSongs.allSongs.get(i).getPath());
            AllSongs.allTracks.add(new Track(AllSongs.allSongs.get(i).getName(),
mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_ARTIST),
mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_GENRE),0, 0, AllSongs.allSongs.get(i),
mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_DURATION),mediaMetadataRetriever.extractMetadata(MediaMetadataR
etriever.METADATA_KEY_ALBUM), mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_BITRATE),
mediaMetadataRetriever.extractMetadata(MediaMetadataRetriever.METADATA_KEY_DATE)));
        }
        mediaMetadataRetriever.release();

        SetUpAudioItemList();

        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {
                return false;
            }

            @Override
            public boolean onQueryTextChange(String newText) {
                filter(newText);
                return true;
            }
        });

    }

    private void showConfirmDialog() {
        final Dialog dialog = new Dialog(MainActivity.this);
        dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
        dialog.setCancelable(true);
        dialog.setContentView(R.layout.dialog_item);

        final EditText folderName = dialog.findViewById(R.id.name_et);
        final Button dialogbtn = dialog.findViewById(R.id.submit_button);

        dialogbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String name = folderName.getText().toString();
                if(name.length() != 0) {
```

```
                audioItemList.add(new AudioItem(foldersCount, "ic_baseline_folder_open_24", name));
                foldersCount++;
                dialog.dismiss();
            }
        }
    });

    dialog.show();
}
public void SetJAudioItemRecycler(List<AudioItem> audioItemList) {
    RecyclerView.LayoutManager layoutManager = new LinearLayoutManager(this, RecyclerView.HORIZONTAL, false);
    RecyclerView audioItemRecycler = findViewById(R.id.AudioItemRecycler);
    audioItemRecycler.setLayoutManager(layoutManager);

    audioItemAdapter = new AudioItemAdapter(this, audioItemList);
    audioItemRecycler.setAdapter(audioItemAdapter);

}

public ArrayList<File> findSong (File file) {
    ArrayList<File> arrayList = new ArrayList<>();

    File[] files = file.listFiles();
    if (files != null) {
        for (File singlefile : files) {
            if (singlefile.isDirectory() && !singlefile.isHidden()) {
                arrayList.addAll(findSong(singlefile));
            } else {
                if (singlefile.getName().endsWith(".mp3") || singlefile.getName().endsWith(".wav")) {
                    arrayList.add(singlefile);
                }
            }
        }
    }
    return arrayList;
}
private void displaySongs() {
    final ArrayList<File> mySongs = findSong(Environment.getExternalStorageDirectory());

    String[] items = new String[mySongs.size()];
    for(int i = 0; i < mySongs.size(); i++)
    {
        items[i] = mySongs.get(i).getName().replace(".mp3", "").replace(".wav", "") ;

    }

    customAdapter = new customAdapter(items);
    listView.setAdapter(customAdapter);
```

```
    listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
            ArrayList<File> recentArrayList = FolderFilesManager.readFromFolder("recentsong");
            recentArrayList.add(mySongs.get(i));
            FolderFilesManager.writeToFolder(recentArrayList, "recentsong");
            for(Track track : AllSongs.allTracks){
                if(mySongs.get(i).getName().equals(track.getTitle())){
                    track.setCountOfPlays(track.getCountOfPlays() + 1);
                    break;
                }
            }

            String songName =  (String) listView.getItemAtPosition(i);
            if(isFilter){
                mySongs.clear();
                mySongs.addAll(IsFilterArray);
                isFilter = false;
            }
            startActivity( new Intent(getApplicationContext(), PlayerActivity.class)
            .putExtra("songs", mySongs)
            .putExtra("songname", songName)
            .putExtra("pos", i));
        }
    });

}
public class customAdapter extends BaseAdapter {

    String[] S;

    public void setS(String[] s) {
        S = s;
    }

    public customAdapter(String[] s) {
        S = s;
    }

    @Override
    public int getCount() {
        return S.length;
    }

    @Override
    public Object getItem(int i) {
        return null;
    }
```

```
    @Override
    public long getItemId(int i) {
        return 0;
    }

    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        View myView  = getLayoutInflater().inflate(R.layout.list_item, null);
        TextView textsong = myView.findViewById(R.id.txtSongName);
        textsong.setSelected(true);
        textsong.setText(S[i]);

        return myView;
    }
    public void FilterList(ArrayList<File> filteredList){
        int size = filteredList.size();
        String[] string = new String[filteredList.size()];
        for(int i = 0; i < size; i++)
        {
            string[i] = filteredList.get(i).getName();
        }
        setS(string);
        IsFilterArray.clear();
        IsFilterArray.addAll(filteredList);
        isFilter = true;
        notifyDataSetChanged();
    }
}

private void filter(String newText) {
    ArrayList<File> filteredList = new ArrayList<>();
    for(File f :AllSongs.allSongs){
        if(f.getName().toLowerCase().contains(newText)){
            filteredList.add(f);
        }
    }
    customAdapter.FilterList(filteredList);
}

public void OnClick_AddNewUserFolderBtn(View view){
    showConfirmDialog();
}

public void SetUpAudioItemList(){
    FolderFilesManager.SetUpUserFolders();
    if(audioItemList.size() == 0) {
        audioItemList.add(new AudioItem(0, "ic_baseline_favorite_24", "Favorite"));
        audioItemList.add(new AudioItem(1, "ic_baseline_recent_actors_24", "Recent"));
        audioItemList.add(new AudioItem(2, "ic_baseline_recomendation_24", "For you"));
```

```
            FolderFilesManager.SaveUserFolders();
            foldersCount =  audioItemList.size();
        }
        audioItemAdapter.notifyDataSetChanged();
    }
    public void SetUpRuntimePermission() {
        Dexter.withContext(this).withPermission(Manifest.permission.READ_EXTERNAL_STORAGE).withListener(new PermissionListener() {
            @Override
            public void onPermissionGranted(PermissionGrantedResponse permissionGrantedResponse) {
                displaySongs();
            }

            @Override
            public void onPermissionDenied(PermissionDeniedResponse permissionDeniedResponse) {

            }

            @Override
            public void onPermissionRationaleShouldBeShown(PermissionRequest permissionRequest, PermissionToken permissionToken) {
                permissionToken.continuePermissionRequest();
            }
        }).check();

    }
    public void SetUpViews(){
        searchView = findViewById(R.id.SearchViewMain);
        listView = findViewById(R.id.listViewSong);
        TextView appNametxt = findViewById(R.id.textView2);
        TextView playlisttxt = findViewById(R.id.textView);
        TextView songstxt = findViewById(R.id.textView3);

        Animation animation = AnimationUtils.loadAnimation(this, androidx.preference.R.anim.abc_slide_in_top);
        appNametxt.startAnimation(animation);
        playlisttxt.startAnimation(animation);
        searchView.startAnimation(animation);
        Animation animation_2 = AnimationUtils.loadAnimation(this, androidx.preference.R.anim.abc_slide_in_bottom);
        listView.startAnimation(animation_2);
        songstxt.startAnimation(animation_2);

    }

    @Override
    protected void onResume() {
        super.onResume();
        audioItemAdapter.notifyDataSetChanged();
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
```

```
        FolderFilesManager.SaveUserFolders();
    }
    @Override
    protected void onPause() {
        super.onPause();
        FolderFilesManager.SaveUserFolders();
    }


}
```

Екран PlayerActivity.

Текст файлу PlayerActivity:

```
package com.example.lycrisproject.activities;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.animation.AnimatorSet;
import android.animation.ObjectAnimator;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.graphics.PorterDuff;
import android.media.MediaPlayer;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;

import com.example.lycrisproject.CreateNotification;
import com.example.lycrisproject.FolderFilesManager;
import com.example.lycrisproject.Playable;
import com.example.lycrisproject.R;
import com.example.lycrisproject.model.Track;
import com.example.lycrisproject.services.OnClearFromRecentService;

import java.io.File;
import java.util.ArrayList;
```

```
import java.util.List;

public class PlayerActivity extends AppCompatActivity implements Playable {

    private Button btnPlay, btnNext, btnPrev, btnFav, btnAdd;
    private TextView txtsname, txtsstart, txtsstop;
    private SeekBar seekmusic;
    private ImageView imageView;

    private String sname;
    public static MediaPlayer mediaPlayer;
    private int position;
    private ArrayList<File> mySongs;
    Thread updateSeekbar;

    private NotificationManager notificationManager;
    private List<Track> tracks;

    private boolean isPlaying = false;

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        if(item.getItemId()==android.R.id.home)
        {
            onBackPressed();
        }
        return super.onOptionsItemSelected(item);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        mediaPlayer.pause();
        mediaPlayer.stop();
        if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
            notificationManager.cancelAll();
        }
        unregisterReceiver(broadcastReceiver);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_player);

        SetUpViews();

        if(mediaPlayer != null) {
            mediaPlayer.stop();
```

```
    mediaPlayer.release();
}

Intent i = getIntent();
Bundle bundle = i.getExtras();

mySongs = (ArrayList) bundle.getParcelableArrayList("songs");
position = bundle.getInt("pos", 0);
txtsname.setSelected(true);

Uri uri = Uri.parse(mySongs.get(position).toString());
sname = mySongs.get(position).getName();
txtsname.setText(sname);

mediaPlayer = MediaPlayer.create(getApplicationContext(), uri);
mediaPlayer.start();

updateSeekbar = new Thread() {
    @Override
    public void run() {
        int totalDuration = mediaPlayer.getDuration();
        int currentPos = 0;
        while (currentPos <= totalDuration + 20)
        {
            try{
                sleep(500);
                if(mediaPlayer.isPlaying()) {
                    currentPos = mediaPlayer.getCurrentPosition();
                    seekmusic.setProgress(currentPos);
                }
            }
            catch (InterruptedException | IllegalStateException e)
            {
                e.printStackTrace();
            }
        }

    }
};
seekmusic.setMax(mediaPlayer.getDuration());
updateSeekbar.start();
seekmusic.getProgressDrawable().setColorFilter(getResources().getColor(R.color.purple_200), PorterDuff.Mode.MULTIPLY);
seekmusic.getThumb().setColorFilter(getResources().getColor(R.color.purple_500), PorterDuff.Mode.SRC_IN);

seekmusic.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int i, boolean b) {

    }
```

```
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
        if (mediaPlayer.isPlaying())
        {
            btnPlay.performClick();
        }

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
        mediaPlayer.seekTo(seekBar.getProgress());
        btnPlay.performClick();

    }
});

String endTime = createTime(mediaPlayer.getDuration());
txtsstop.setText(endTime);

final Handler handler = new Handler();
final int delay = 1000;

handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        String currentTime = createTime(mediaPlayer.getCurrentPosition());
        txtsstart.setText(currentTime);
        handler.postDelayed(this, delay);
    }
}, delay);


mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mP) {
        String endTime = createTime(mediaPlayer.getDuration());
        txtsstop.setText(endTime);

        mediaPlayer.stop();
        mediaPlayer.release();
        position = ((position+1)%mySongs.size());
        Uri u = Uri.parse(mySongs.get(position).toString());
        mediaPlayer = MediaPlayer.create(getApplicationContext(), u);
        sname = mySongs.get(position).getName() ;
        txtsname.setText(sname);
        mediaPlayer.setOnCompletionListener(this);
        mediaPlayer.start();
```

```
                btnPlay.setBackgroundResource(R.drawable.ic_baseline_stop_24);
                btnFav.setBackgroundResource(R.drawable.ic_baseline_favorite_24);
                StartAnimation(imageView);

                seekmusic.setMax(mediaPlayer.getDuration());
                endTime = createTime(mediaPlayer.getDuration());
                txtsstop.setText(endTime);
                if(FolderFilesManager.readFromFolder("favoritesong").contains(mySongs.get(position)))
btnFav.setBackgroundResource(R.drawable.ic_baseline_star_24);
                CreateNotification.createnotification(PlayerActivity.this, tracks.get(position), R.drawable.ic_baseline_stop_24, 1, tracks.size()-1);
            }
        });

        if(FolderFilesManager.readFromFolder("favoritesong").contains(mySongs.get(position)))
btnFav.setBackgroundResource(R.drawable.ic_baseline_star_24);

        populateTracks();
        if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
            createChannel();
            registerReceiver(broadcastReceiver, new IntentFilter("TRACKS_TRACKS"));
            startService(new Intent(getBaseContext(), OnClearFromRecentService.class));
        }
        CreateNotification.createnotification(PlayerActivity.this, tracks.get(position), R.drawable.ic_baseline_stop_24, 1, tracks.size()-1);
    }

    private void populateTracks(){
        tracks = new ArrayList<>();

        int size = mySongs.size();

        for(int i = 0; i < size; i ++){
            tracks.add(new Track(mySongs.get(i).getName(), "Unknown Artist", "", R.drawable.cloud, 0, mySongs.get(i), "", "", "", ""));
        }
    }

    private void createChannel() {
        if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
            NotificationChannel channel = new NotificationChannel(CreateNotification.CHANNEL_ID,
                    "KOD DEV", NotificationManager.IMPORTANCE_LOW);

            notificationManager = getSystemService(NotificationManager.class);
            if(notificationManager != null){
                notificationManager.createNotificationChannel(channel);
            }

        }
    }

    private void StartAnimation(View view) {
```

```
    ObjectAnimator animator = ObjectAnimator.ofFloat(imageView, "rotation", 0f, 360f);
    animator.setDuration(1000);
    AnimatorSet animatorSet = new AnimatorSet();
    animatorSet.playTogether(animator);
    animatorSet.start();
}
private String createTime(int duration) {
    String time = "";
    int min = duration/1000/60;
    int sec = duration/1000%60;

    time+= min + ":";

    if (sec < 10)
    {
        time+="0" ;
    }
    time+=sec;

    return time;
}

private void SetUpViews(){
    btnPlay = findViewById(R.id.playButton);
    btnNext = findViewById(R.id.nextButton);
    btnPrev = findViewById(R.id.prevButton);
    btnFav = findViewById(R.id.favButton);
    btnAdd = findViewById(R.id.addButton);
    txtsname = findViewById(R.id.txtsn);
    txtsstart = findViewById(R.id.txtsstart);
    txtsstop = findViewById(R.id.txtsstop);
    seekmusic = findViewById(R.id.seekBar);
    imageView = findViewById(R.id.imageView);
}

public void OnClick_addToFavouriteBtn(View view){
    ArrayList<File> favoriteArrayList = FolderFilesManager.readFromFolder("favoritesong");
    if(!favoriteArrayList.contains(mySongs.get(position)))
    {
        favoriteArrayList.add(mySongs.get(position));
        FolderFilesManager.writeToFolder(favoriteArrayList,"favoritesong");
        btnFav.setBackgroundResource(R.drawable.ic_baseline_star_24);
        Toast.makeText(this, "Added", Toast.LENGTH_LONG).show();
    }
    else
    {
        favoriteArrayList.remove(favoriteArrayList.indexOf(mySongs.get(position)));
        FolderFilesManager.writeToFolder(favoriteArrayList,"favoritesong");
        btnFav.setBackgroundResource(R.drawable.ic_baseline_favorite_24);
```

```
                Toast.makeText(this, "Deleted", Toast.LENGTH_LONG).show();
            }
        }
        public void OnClick_addToUserFolderBtn(View view){
            startActivity( new Intent(getApplicationContext(), AddToFolderActivity.class)
                    .putExtra("song",mySongs.get(position)));
        }
        public void OnClick_PreviousBtn(View view) {
            String endTime = createTime(mediaPlayer.getDuration());
            txtsstop.setText(endTime);

            mediaPlayer.stop();
            mediaPlayer.release();
            position = ((position - 1) < 0)?(mySongs.size() - 1 ):(position-1);
            Uri u = Uri.parse(mySongs.get(position).toString());
            mediaPlayer = MediaPlayer.create(getApplicationContext(), u);
            sname = mySongs.get(position).getName();
            txtsname.setText(sname);
            mediaPlayer.start();
            btnPlay.setBackgroundResource(R.drawable.ic_baseline_stop_24);
            btnFav.setBackgroundResource(R.drawable.ic_baseline_favorite_24);
            StartAnimation(imageView);

            seekmusic.setMax(mediaPlayer.getDuration());
            endTime = createTime(mediaPlayer.getDuration());
            txtsstop.setText(endTime);
            if(FolderFilesManager.readFromFolder("favoritesong").contains(mySongs.get(position)))
btnFav.setBackgroundResource(R.drawable.ic_baseline_star_24);
            CreateNotification.createnotification(PlayerActivity.this, tracks.get(position), R.drawable.ic_baseline_stop_24, 1, tracks.size()-1);
        }
        public void OnClick_NextBtn(View view){
            String endTime = createTime(mediaPlayer.getDuration());
            txtsstop.setText(endTime);

            mediaPlayer.stop();
            mediaPlayer.release();
            position = ((position+1)%mySongs.size());
            Uri u = Uri.parse(mySongs.get(position).toString());
            mediaPlayer = MediaPlayer.create(getApplicationContext(), u);
            sname = mySongs.get(position).getName();
            txtsname.setText(sname);

            mediaPlayer.start();
            btnPlay.setBackgroundResource(R.drawable.ic_baseline_stop_24);
            btnFav.setBackgroundResource(R.drawable.ic_baseline_favorite_24);
            StartAnimation(imageView);

            seekmusic.setMax(mediaPlayer.getDuration());
            endTime = createTime(mediaPlayer.getDuration());
```

```
        txtsstop.setText(endTime);
        if(FolderFilesManager.readFromFolder("favoritesong").contains(mySongs.get(position)))
btnFav.setBackgroundResource(R.drawable.ic_baseline_star_24);
        CreateNotification.createnotification(PlayerActivity.this, tracks.get(position), R.drawable.ic_baseline_stop_24, 1, tracks.size()-1);
    }
    public void OnClick_PlayPauseBtn(View view){
        if (mediaPlayer.isPlaying())
        {
            btnPlay.setBackgroundResource(R.drawable.ic_play);
            mediaPlayer.pause();
            CreateNotification.createnotification(PlayerActivity.this, tracks.get(position), R.drawable.ic_play, 1, tracks.size()-1);
        }
        else
        {
            btnPlay.setBackgroundResource(R.drawable.ic_baseline_stop_24);
            mediaPlayer.start();
            CreateNotification.createnotification(PlayerActivity.this, tracks.get(position), R.drawable.ic_baseline_stop_24, 1, tracks.size()-1);
        }
    }

    BroadcastReceiver broadcastReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            String action = intent.getExtras().getString("actionname");

            switch (action){
                case CreateNotification.ACTION_PREVIOUS:
                    onTrackPrevious();
                    break;
                case CreateNotification.ACTION_PLAY:
                    if(isPlaying){
                        onTrackPause();
                    }else
                    {
                        onTrackPlay();
                    }
                    break;
                case CreateNotification.ACTION_NEXT:
                    onTrackNext();
                    break;
            }
        }
    };

    @Override
    public void onTrackPrevious() {
        btnPrev.performClick();
    }
```

```
    @Override
    public void onTrackPlay() {
        btnPlay.performClick();
        isPlaying = true;
    }

    @Override
    public void onTrackPause() {
        isPlaying = false;
        btnPlay.performClick();
    }

    @Override
    public void onTrackNext() {
        btnNext.performClick();
    }
}
```

Екран SplashScreenActivity.

Текст файлу SplashScreenActivity:

```
package com.example.lycrisproject.activities;

import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;

@SuppressLint("CustomSplashScreen")
public class SplashScreenActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
        finish();
    }
}
```

Текст файлу AudioItemAdapter:

```
package com.example.lycrisproject.adapter;

import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
```

```
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.lycrisproject.activities.FolderActivity;
import com.example.lycrisproject.R;
import com.example.lycrisproject.model.AudioItem;

import java.util.List;

public class AudioItemAdapter extends RecyclerView.Adapter<AudioItemAdapter.AudioItemViewHolder> {

    Context context;
    List<AudioItem> audioItems;

    public AudioItemAdapter(Context context, List<AudioItem> audioItems) {
        this.context = context;
        this.audioItems = audioItems;
    }

    @NonNull
    @Override
    public AudioItemViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View audioItems = LayoutInflater.from(context).inflate(R.layout.audio_item, parent, false);
        return new AudioItemAdapter.AudioItemViewHolder(audioItems);
    }

    @Override
    public void onBindViewHolder(@NonNull AudioItemViewHolder holder, @SuppressLint("RecyclerView") int position) {

        int imageId = context.getResources().getIdentifier(audioItems.get(position).getImg(), "drawable", context.getPackageName());
        holder.audioItemImg.setImageResource(imageId);

        holder.audioItemText.setText(audioItems.get(position).getTitle());

        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                Intent intent = new Intent(context, FolderActivity.class);
                intent.putExtra("categoryId", position);
                context.startActivity(intent);
            }
        });
    }

    @Override
```

```
    public int getItemCount() {
        return audioItems.size();
    }

    public static final class AudioItemViewHolder extends RecyclerView.ViewHolder{

        ImageView audioItemImg;
        TextView audioItemText;

        public AudioItemViewHolder(@NonNull View itemView) {
            super(itemView);

            audioItemImg = itemView.findViewById(R.id.audioItemImgView);
            audioItemText = itemView.findViewById(R.id.audioItemTextView);
        }
    }
}
```

Текст файлу AllSongs:

```
package com.example.lycrisproject.model;

import java.io.File;
import java.util.ArrayList;

public class AllSongs {
    public static ArrayList<File> allSongs = new ArrayList<>();
    public static ArrayList<Track> allTracks = new ArrayList<>();
}
```

Текст файлу AudioItem:

```
package com.example.lycrisproject.model;



import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class AudioItem implements Serializable{

    private String img, title;
    public static  List<AudioItem> audioItemList = new ArrayList<>();


    public AudioItem(int id, String img, String title, String date, String janre) {
        this.img = img;
        this.title = title;

    }

    public AudioItem(int id, String img, String title) {
```

```
        this.img = img;
        this.title = title;
    }

    public void setImg(String img) {
        this.img = img;
    }

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


    public String getImg() {
        return img;
    }

    public String getTitle() {
        return title;
    }


}
```

Текст файлу Track:

```
package com.example.lycrisproject.model;

import android.media.MediaMetadataRetriever;

import com.example.lycrisproject.FolderFilesManager;

import java.io.File;
import java.util.ArrayList;

public class Track {


    private String title;
    private String artist;
    private String genre;
    private String duration;
    private String album;
    private String bitrate;
    private String date;

    public void setGenre(String genre) {
        this.genre = genre;
    }
```

```
public String getGenre() {
    return genre;
}

private int image;
private int countOfPlays;
private File file;

public void setFile(File file) {
    this.file = file;
}

public File getFile() {
    return file;
}

public void setCountOfPlays(int countOfPlays) {
    this.countOfPlays = countOfPlays;
}

public int getCountOfPlays() {
    return countOfPlays;
}

public void setDuration(String duration) {
    this.duration = duration;
}

public void setAlbum(String album) {
    this.album = album;
}

public void setBitrate(String bitrate) {
    this.bitrate = bitrate;
}

public void setDate(String date) {
    this.date = date;
}

public String getDuration() {
    return duration;
}

public String getAlbum() {
    return album;
}
```

```
public String getBitrate() {
    return bitrate;
}

public String getDate() {
    return date;
}

public Track(String title, String artist, String genre, int image, int countOfPlays, File file, String duration, String album, String bitrate, String date) {
    this.title = title;
    if(artist == null){
        this.artist = "Unknown artist";
    }else{
        this.artist = artist;
    }
    this.image = image;
    this.countOfPlays = countOfPlays;
    this.file = file;
    if(genre == null){
        this.genre = "Unknown genre";
    }else{
        this.genre = genre;
    }
    if(duration == null){
        this.duration = "Unknown duration";
    }else{
        this.duration = duration;
    }
    if(bitrate == null){
        this.bitrate = "Unknown bitrate";
    }else{
        this.bitrate = bitrate;
    }
    if(album == null){
        this.album = "Unknown album";
    }else{
        this.album = album;
    }
    if(date == null){
        this.date = "Unknown date";
    }else{
        this.date = date;
    }
}

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

```
    public void setArtist(String artist) {
        this.artist = artist;
    }

    public void setImage(int image) {
        this.image = image;
    }

    public String getTitle() {
        return title;
    }

    public String getArtist() {
        return artist;
    }

    public int getImage() {
        return image;
    }
}
```

Текст файлу CreateNotification:

```
package com.example.lycrisproject;

import android.app.Notification;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Build;
import android.support.v4.media.session.MediaSessionCompat;

import androidx.core.app.NotificationCompat;
import androidx.core.app.NotificationManagerCompat;

import com.example.lycrisproject.model.Track;
import com.example.lycrisproject.services.NotificationActionService;

public class CreateNotification {
    public static final String CHANNEL_ID = "channel1";
    public static final String ACTION_PREVIOUS = "actionprevious";
    public static final String ACTION_PLAY = "actionplay";
    public static final String ACTION_NEXT = "actionnext";



    public static Notification notification;

    public static void createnotification(Context context, Track track, int playbutton, int pos, int size){
```

```
if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
    NotificationManagerCompat notificationManagerCompat = NotificationManagerCompat.from(context);
    MediaSessionCompat mediaSessionCompat = new MediaSessionCompat(context, "tag");

    Bitmap icon = BitmapFactory.decodeResource(context.getResources(), track.getImage());

    PendingIntent pendingIntentPrevious;
    int drw_previous;
    if(pos == 0){
       pendingIntentPrevious = null;
       drw_previous = 0;
    }else {
       Intent intentPrevius = new Intent(context, NotificationActionService.class)
             .setAction(ACTION_PREVIOUS);
       pendingIntentPrevious = PendingIntent.getBroadcast(context, 0,intentPrevius, PendingIntent.FLAG_UPDATE_CURRENT);
       drw_previous = R.drawable.ic_baseline_skip_previous_24;
    }
    Intent intentPlay = new Intent(context, NotificationActionService.class)
          .setAction(ACTION_PLAY);
    PendingIntent pendingIntentPlay = PendingIntent.getBroadcast(context, 0,intentPlay, PendingIntent.FLAG_UPDATE_CURRENT);

    PendingIntent pendingIntentNext;
    int drw_next;
    if(pos == -1){
       pendingIntentNext = null;
       drw_next = 0;
    }else {
       Intent intentNext = new Intent(context, NotificationActionService.class)
             .setAction(ACTION_NEXT);
       pendingIntentNext = PendingIntent.getBroadcast(context, 0,intentNext, PendingIntent.FLAG_UPDATE_CURRENT);
       drw_next = R.drawable.ic_baseline_skip_next_24;
    }

    notification = new NotificationCompat.Builder(context, CHANNEL_ID)
          .setSmallIcon(R.drawable.ic_baseline_music_note_24)
          .setContentTitle(track.getTitle())
          .setContentText(track.getArtist())
          .setLargeIcon(icon)
          .setOnlyAlertOnce(true)
          .setShowWhen(false)
          .addAction(drw_previous, "Previous", pendingIntentPrevious)
          .addAction(playbutton, "Play", pendingIntentPlay)
          .addAction(drw_next, "Next", pendingIntentNext)
          .setStyle(new androidx.media.app.NotificationCompat.MediaStyle()
             .setShowActionsInCompactView(0,1,2)
             .setMediaSession(mediaSessionCompat.getSessionToken()))
          .setPriority(NotificationCompat.PRIORITY_LOW)
          .build();
    notificationManagerCompat.notify(1, notification);
```

```
        }
    }

}
```

Текст файлу FolderFilesManager:

```
package com.example.lycrisproject;

import static com.example.lycrisproject.model.AudioItem.audioItemList;

import android.os.Environment;

import com.example.lycrisproject.model.AllSongs;
import com.example.lycrisproject.model.AudioItem;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

public class FolderFilesManager {
    public static ArrayList<File> readFromFolder(String folderName) {
        ArrayList<File> arrayList = new ArrayList<>();
        try
        {
            FileInputStream fileInputStream =new FileInputStream("/data/data/com.example.lycrisproject/" + folderName);
            InputStreamReader inputStreamReader = new InputStreamReader(fileInputStream);
            BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
            String lines;
            while ((lines = bufferedReader.readLine()) != null){
                for(File song : AllSongs.allSongs){
                    String s = song.getName();
                    if(song.getName().equals(lines))
                    {
                        arrayList.add(song);
                        break;
                    }
                }
            }
            fileInputStream.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
```

```
            e.printStackTrace();
        }

        return arrayList;
    }
    public static void writeToFolder(ArrayList<File> arraySong, String folderName) {
        try
        {
            FileOutputStream fileOutputStream = new FileOutputStream("/data/data/com.example.lycrisproject/" + folderName);

            for(int i = 0; i < arraySong.size(); i ++)
            {
                fileOutputStream.write(arraySong.get(i).getName().toString().getBytes());
                fileOutputStream.write("\n".getBytes());
            }
            fileOutputStream.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public static void deleteFolder(String folderName){
        String filePath = "/data/data/com.example.lycrisproject/" + folderName;
        File file = new File(filePath);
        if (file.exists()) {
            file.delete();
        }
    }
    public static void SetUpUserFolders(){
        ArrayList<AudioItem> arrayList = new ArrayList<>();
        try {
            FileInputStream fis = new FileInputStream("/data/data/com.example.lycrisproject/"+"myFile.ser");
            ObjectInputStream ois = new ObjectInputStream(fis);
            Object obj = ois.readObject();
            ois.close();
            arrayList = (ArrayList<AudioItem>) obj;
            fis.close();
            ois.close();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        audioItemList.clear();
        audioItemList.addAll(arrayList);
    }
    public static void SaveUserFolders(){
        try {
```

```
            FileOutputStream fos = new FileOutputStream("/data/data/com.example.lycrisproject/"+"myFile.ser");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject((Object) AudioItem.audioItemList);
            fos.close();
            oos.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Текст файлу Playable:

```
package com.example.lycrisproject;

public interface Playable {
    void onTrackPrevious();
    void onTrackPlay();
    void onTrackPause();
    void onTrackNext();
}
```

Текст файлу activity_add_to_folder:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".activities.AddToFolderActivity"
    android:background="@drawable/gradient_1_side_panel">

    <TextView
        android:id="@+id/folderinvite"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:layout_marginTop="30dp"
        android:fontFamily="@font/montserrat_alternates_bold_italic"
        android:text="@string/app_short_name"
        android:textColor="@color/black"
        android:textSize="40sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:text="@string/add_to_folder_invite" />

    <ListView
        android:id="@+id/folderViewList"
        android:layout_width="wrap_content"
        android:layout_height="350dp"
        android:divider="@android:color/transparent"
```

```
        android:dividerHeight="10.0sp"
        android:padding="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.35"
        app:layout_constraintStart_toStartOf="parent" />

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

Текст файлу activity_folder:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/gradient_1_side_panel"
    tools:context=".activities.FolderActivity">

  <TextView
      android:id="@+id/categoryName"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginStart="20dp"
      android:layout_marginTop="30dp"
      android:fontFamily="@font/montserrat_alternates_bold_italic"
      android:text="@string/app_short_name"
      android:textColor="@color/black"
      android:textSize="40sp"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      tools:text="Category_name" />

  <ListView
      android:id="@+id/folderList"
      android:layout_width="413dp"
      android:layout_height="451dp"
      android:layout_marginTop="68dp"
      android:divider="@android:color/transparent"
      android:dividerHeight="10.0sp"
      android:padding="8dp"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="0.0"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toBottomOf="@+id/categoryName"
      tools:ignore="TouchTargetSizeCheck" />

  <Button
```

```
    android:id="@+id/delButtonFolder"
    android:layout_width="40dp"
    android:layout_height="40dp"
    android:layout_alignParentEnd="true"
    android:layout_marginStart="20dp"
    android:layout_marginTop="15dp"
    android:background="@drawable/ic_baseline_delete_24"
    android:backgroundTint="@color/black"
    android:textColor="@color/black"
    android:visibility="invisible"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/categoryName"
    tools:ignore="SpeakableTextPresentCheck,TouchTargetSizeCheck" />


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

Текст файлу activity_main:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/gradient_1_side_panel"
    tools:context=".activities.MainActivity">

    <ListView
        android:id="@+id/listViewSong"
        android:layout_width="wrap_content"
        android:layout_height="330dp"
        android:divider="@android:color/transparent"
        android:dividerHeight="10.0sp"
        android:fadeScrollbars="false"
        android:fastScrollAlwaysVisible="false"
        android:fastScrollEnabled="false"
        android:padding="8dp"
        android:scrollbars="none"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView3" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/AudioItemRecycler"
        android:layout_width="0dp"
        android:layout_height="115dp"
        android:layout_marginStart="10dp"
        android:layout_marginTop="5dp"
```

```
    android:layout_marginEnd="70dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView"
    tools:ignore="MissingConstraints" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="10dp"
    android:layout_marginTop="15dp"
    android:fontFamily="@font/montserrat_alternates_bold_italic"
    android:text="@string/app_short_name"
    android:textColor="@color/white"
    android:textSize="40sp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:text="@string/app_short_name" />

<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="10dp"
    android:layout_marginBottom="5dp"
    android:fontFamily="@font/montserrat_alternates_bold_italic"
    android:text="Your playlists:"
    android:textColor="@color/white"
    android:textSize="25dp"
    app:layout_constraintBottom_toTopOf="@+id/AudioItemRecycler"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/SearchViewMain"
    tools:text="Your playlists:" />

<TextView
    android:id="@+id/textView3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="10dp"
    android:layout_marginTop="41dp"
    android:fontFamily="@font/montserrat_alternates_bold_italic"
    android:text="Your songs:"
    android:textColor="@color/white"
    android:textSize="25dp"
    app:layout_constraintBottom_toTopOf="@+id/listViewSong"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/AudioItemRecycler"
    tools:text="@string/app_short_name" />
```

```
    <Button
        android:id="@+id/addButtonMain"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_marginTop="5dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="50dp"
        android:background="@drawable/ic_baseline_add_24"
        android:onClick="OnClick_AddNewUserFolderBtn"
        app:layout_constraintBottom_toTopOf="@+id/listViewSong"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        tools:ignore="SpeakableTextPresentCheck">

    </Button>

    <androidx.appcompat.widget.SearchView
        android:id="@+id/SearchViewMain"
        android:layout_width="406dp"
        android:layout_height="50dp"
        android:scrollbarFadeDuration="250"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />

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

Текст файлу activity_player:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".activities.PlayerActivity"
    android:background="@drawable/gradient_2_bg_color"
    android:orientation="vertical"
    android:weightSum="10">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="7"
        android:gravity="center"
        android:orientation="vertical">
        <TextView
            android:id="@+id/txtsn"
            android:layout_margin="20dp"
```

```
    android:ellipsize="marquee"
    android:marqueeRepeatLimit="marquee_forever"
    android:padding="10dp"
    android:singleLine="true"
    android:text="Song name"
    android:textColor="@color/white"
    android:textSize="22sp"
    android:textAlignment="center"
    android:textStyle="italic"

    android:layout_width="match_parent"
    android:layout_height="wrap_content">

</TextView>

<ImageView
    android:id="@+id/imageView"
    android:layout_marginBottom="8dp"
    android:src="@drawable/ic_baseline_music_note_24"
    android:layout_width="250dp"
    android:layout_height="250dp">

</ImageView>
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="60dp"
    >
    <SeekBar
        android:id="@+id/seekBar"
        android:layout_centerInParent="true"
        android:layout_alignParentBottom="true"
        android:layout_margin="20dp"
        android:layout_marginBottom="40dp"
        android:layout_width="250dp"
        android:layout_height="wrap_content">

    </SeekBar>
    <TextView
        android:id="@+id/txtsstart"
        android:layout_toLeftOf="@+id/seekBar"
        android:layout_centerInParent="true"
        android:layout_alignParentLeft="false"
        android:layout_marginLeft="20dp"
        android:text="0:10"
        android:textColor="@color/white"
        android:textSize="14sp"

        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
```

```
        </TextView>

        <TextView
            android:id="@+id/txtsstop"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="false"
            android:layout_centerInParent="true"
            android:layout_marginRight="20dp"
            android:layout_toRightOf="@+id/seekBar"

            android:text="4:10"
            android:textColor="@color/white"
            android:textSize="14sp">

        </TextView>
    </RelativeLayout>
</LinearLayout>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="3">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <Button
            android:id="@+id/playButton"
            android:layout_width="70dp"
            android:layout_height="70dp"
            android:layout_centerHorizontal="true"
            android:background="@drawable/ic_baseline_stop_24"
            android:onClick="OnClick_PlayPauseBtn"></Button>

        <Button
            android:id="@+id/nextButton"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_marginTop="15dp"
            android:layout_toRightOf="@+id/playButton"
            android:background="@drawable/ic_baseline_skip_next_24"
            android:onClick="OnClick_NextBtn">

        </Button>

        <Button
            android:id="@+id/prevButton"
            android:layout_width="50dp"
```

```
            android:layout_height="50dp"
            android:layout_marginTop="15dp"
            android:layout_toLeftOf="@+id/playButton"
            android:background="@drawable/ic_baseline_skip_previous_24"
            android:onClick="OnClick_PreviousBtn">

        </Button>

        <Button
            android:id="@+id/favButton"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_marginTop="15dp"
            android:layout_toRightOf="@+id/nextButton"
            android:background="@drawable/ic_baseline_favorite_24"
            android:onClick="OnClick_addToFavouriteBtn">

        </Button>

        <Button
            android:id="@+id/addButton"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_marginTop="5dp"
            android:layout_marginEnd="10dp"
            android:layout_toLeftOf="@+id/prevButton"
            android:background="@drawable/ic_baseline_add_24"
            android:onClick="OnClick_addToUserFolderBtn"
            app:layout_constraintEnd_toEndOf="parent"
            tools:visibility="visible">

        </Button>
    </RelativeLayout>
  </LinearLayout>

</LinearLayout>
```

Текст файлу audio_item:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_marginEnd="25dp"
    android:layout_height="wrap_content"
    android:background="#00000000">

    <androidx.cardview.widget.CardView
        android:id="@+id/cardView"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:cardCornerRadius="2dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:cardBackgroundColor="@android:color/transparent"
        app:cardElevation="0dp">


        <LinearLayout
            android:id="@+id/AudioItemBg"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical"
            >

            <ImageView
                android:id="@+id/audioItemImgView"
                android:layout_width="100dp"
                android:layout_height="100dp"
                android:background="@android:color/transparent"
                android:contentDescription="@string/app_short_name"
                tools:layout_editor_absoluteX="152dp"
                tools:layout_editor_absoluteY="126dp"
                tools:srcCompat="@drawable/ic_baseline_favorite_24" />

            <TextView
                android:id="@+id/audioItemTextView"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:fontFamily="@font/montserrat_subrayada"
                android:text="@string/app_short_name"
                android:textColor="@color/white"
                android:textAlignment="center"
                android:textSize="12sp" />




        </LinearLayout>
        ,
    </androidx.cardview.widget.CardView>>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Текст файлу dialog_item:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:layout_margin="16dp">

    <TextView
        android:id="@+id/DialogText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="24sp"
        android:layout_gravity="center"
        android:textColor="@color/black"
        android:text="@string/folder_dialog"
        android:layout_marginBottom="16dp"/>

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:imeOptions="actionNext"
        android:id="@+id/name_et"
        android:maxLength="12"/>

    <Button
        android:id="@+id/submit_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_gravity="center"
        android:background="@drawable/gradient_1_side_panel"
        android:textColor="@color/black"
        android:text="@string/create_string" />
</LinearLayout>
```

Текст файлу folder_item:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_marginEnd="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginStart="8dp"
    app:cardBackgroundColor="@android:color/transparent"
    app:cardElevation="0dp">

    <RelativeLayout
        android:padding="8dp"
        android:background="@drawable/list_bg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
```

```
    <ImageView
        android:id="@+id/imfFolder"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_marginStart="5dp"
        android:layout_marginTop="5dp"
        android:background="@drawable/list_bg"
        android:src="@drawable/ic_baseline_folder_open_24">

    </ImageView>
    <TextView
        android:id="@+id/txtFolderName"
        android:layout_alignParentEnd="true"
        android:layout_marginStart="5dp"
        android:layout_marginEnd="5dp"
        android:layout_toEndOf="@+id/imfFolder"
        android:padding="6dp"
        android:textColor="@color/black"
        android:text="Song Name"
        android:textSize="15sp"
        android:singleLine="true"
        android:marqueeRepeatLimit="marquee_forever"
        android:ellipsize="marquee"
        android:scrollHorizontally="true"
        android:layout_width="wrap_content"
        android:layout_height="40dp"></TextView>


  </RelativeLayout>

</androidx.cardview.widget.CardView>
```

Текст файлу janre_category_item:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_marginEnd="20dp"
    android:layout_height="wrap_content">


    <TextView
        android:id="@+id/categoryTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:fontFamily="@font/montserrat_alternates_bold_italic"
        android:text="Поп"
        android:textAlignment="center"
        android:textColor="@color/black"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Текст файлу list_item:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_marginEnd="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginStart="8dp"
    app:cardBackgroundColor="@android:color/transparent"
    app:cardElevation="0dp">

    <RelativeLayout
        android:padding="8dp"
        android:background="@drawable/list_bg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <ImageView
            android:id="@+id/imgsong"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:layout_alignParentStart="true"
            android:layout_alignParentTop="true"
            android:layout_marginStart="5dp"
            android:layout_marginTop="5dp"
            android:src="@drawable/ic_baseline_music_note_24">

        </ImageView>
        <TextView
            android:id="@+id/txtSongName"
            android:layout_alignParentEnd="true"
            android:layout_marginStart="5dp"
            android:layout_marginEnd="5dp"
            android:layout_toEndOf="@+id/imgsong"
            android:padding="6dp"
            android:textColor="@color/white"
            android:text="Song Name"
```

```
            android:textSize="15sp"
            android:singleLine="true"
            android:marqueeRepeatLimit="marquee_forever"
            android:ellipsize="marquee"
            android:scrollHorizontally="true"
            android:layout_width="wrap_content"
            android:layout_height="40dp"></TextView>


    </RelativeLayout>

</androidx.cardview.widget.CardView>
```

Текст файлу list_item_with_del:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginStart="8dp"
    app:cardBackgroundColor="@android:color/transparent"
    app:cardElevation="0dp">

    <RelativeLayout
        android:padding="8dp"
        android:background="@drawable/list_bg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <ImageView
            android:id="@+id/imgsong"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:layout_alignParentStart="true"
            android:layout_alignParentTop="true"
            android:layout_marginStart="5dp"
            android:layout_marginTop="5dp"
            android:background="@drawable/list_bg"
            android:src="@drawable/ic_baseline_music_note_24">

        </ImageView>

        <Button
            android:id="@+id/delButtonList"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:layout_alignParentEnd="true"
```

```
            android:background="@drawable/ic_baseline_delete_24"
            android:backgroundTint="@color/black"
            android:textColor="@color/black"
            android:focusable="false"/>

        <TextView
            android:id="@+id/txtSongName"
            android:layout_width="228dp"
            android:layout_height="45dp"
            android:layout_marginStart="9dp"
            android:layout_marginEnd="5dp"
            android:layout_toStartOf="@+id/delButtonList"
            android:layout_toEndOf="@+id/imgsong"
            android:ellipsize="marquee"
            android:marqueeRepeatLimit="marquee_forever"
            android:padding="6dp"
            android:scrollHorizontally="true"
            android:singleLine="true"
            android:text="Song Name"
            android:textColor="@color/black"
            android:textSize="15sp"></TextView>




    </RelativeLayout>

</androidx.cardview.widget.CardView>
```

Текст файлу strings.xml:

```
<resources>
    <string name="app_name">LycrisProject</string>
    <string name="action_settings">Settings</string>
    <!-- Strings used for fragments for navigation -->
    <string name="first_fragment_label">First Fragment</string>
    <string name="second_fragment_label">Second Fragment</string>
    <string name="next">Next</string>
    <string name="previous">Previous</string>

    <string name="hello_first_fragment">Hello first fragment</string>
    <string name="hello_second_fragment">Hello second fragment. Arg: %1$s</string>
    <string name="app_short_name">Lycris</string>
    <string name="ok_string">Ok</string>
    <string name="create_string">Create</string>
    <string name="folder_dialog">Type Folder Name</string>
    <string name="add_to_folder_invite">Select your folder</string>
</resources>
```

Текст файлу stroke.xml:

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

```
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <solid android:color="#FFFFFF"/>
   <solid android:color="#00000000"></solid>
   <corners android:radius="0dp" />
   <padding android:left="0dp" android:top="0dp"
      android:right="0dp" android:bottom="0dp" />
   <stroke android:color="@color/black" android:width="1dp" />
</shape>
```

Текст файлу splashscreen.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:drawable="@color/white"/>
   <item >
      <bitmap android:src="@drawable/splashscreen_img"
         android:gravity="center">
      </bitmap>
   </item>

</layer-list>
```

Текст файлу list_bg.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
   android:shape="rectangle">
   <solid android:color="#26CFD8DC"></solid>
   <corners android:radius="10dip" />


</shape>
```

Текст файлу gradient_2_bg_color.xml:

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

   <gradient

      android:type="linear"
      android:angle="90"
      android:startColor="#D2DECB"
      android:endColor="#91A59F"

      >
   </gradient>>

</shape>
```

Текст файлу gradient_1_bg_color.xml:

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

```
    <gradient

        android:type="linear"
        android:angle="-90"
        android:startColor="#263238"
        android:centerX="0.25"
        android:centerY="0.25"
        android:centerColor="#B0BEC5"
        android:endColor="#546E7A"

        >
    </gradient>>

</shape>
```

Текст файлу build.gradle:

```
plugins {
    id 'com.android.application'
}



android {
    compileSdk 32

    defaultConfig {
        applicationId "com.example.lycrisproject"
        minSdk 27
        targetSdk 32
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"

        vectorDrawables.useSupportLibrary = true
    }

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

    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    buildFeatures {
```

```
        viewBinding true
    }
}

dependencies {

    implementation 'androidx.appcompat:appcompat:1.3.0'
    implementation 'com.karumi:dexter:6.2.2'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'androidx.media:media:1.1.0'
    implementation 'androidx.navigation:navigation-fragment:2.3.5'
    implementation 'androidx.navigation:navigation-ui:2.3.5'
    implementation 'com.android.car.ui:car-ui-lib:+'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'

}
```

Текст файлу AndroidManifest.xml:

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

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/cloud1"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/cloud1_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.LycrisProject">
        <activity
            android:name=".activities.SplashScreenActivity"
            android:theme="@style/SplashTheme"
            android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
        </activity>
        <activity
            android:name=".activities.AddToFolderActivity"
            android:exported="false" />
        <activity
```

```
            android:name=".activities.FolderActivity"
            android:exported="false" />
        <activity
            android:name=".activities.PlayerActivity"
            android:exported="false"
            android:screenOrientation="portrait" />
        <activity
            android:name=".activities.MainActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:screenOrientation="portrait"
            android:theme="@style/Theme.LycrisProject.NoActionBar">

        </activity>

        <meta-data
            android:name="preloaded_fonts"
            android:resource="@array/preloaded_fonts" />
        <receiver android:name=".services.NotificationActionService"/>
        <service android:name=".services.OnClearFromRecentService"/>
    </application>

</manifest>
```