Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №4
на тему «Дослідження способів роботи з медіаданими»
З дисципліни «Розробка мобільних застосувань під Android»

Виконала:
студентка 3 курсу
групи ІС-21
Бєльчик Софія

Київ-2025

Мета роботи - дослідити яким чином платформа Андроїд надає можливість оброблювати аудіо-файли та відео-файли та отримати практичні навички щодо використання інструментів відтворення медіа-даних.

Варіант - 14

Завдання:
Написати програму під платформу Андроїд, яка має інтерфейс для запуску аудіо-файлів та відео-файлів. Мінімально інтерфейс має надавати можливість Програвати/Зупиняти/Призупиняти відтворення відео-файлу або аудіо-файлу, який зберігається у внутрішньому сховищі.
Посилання гітхаб:https://github.com/SofiaBielchik/task4.git

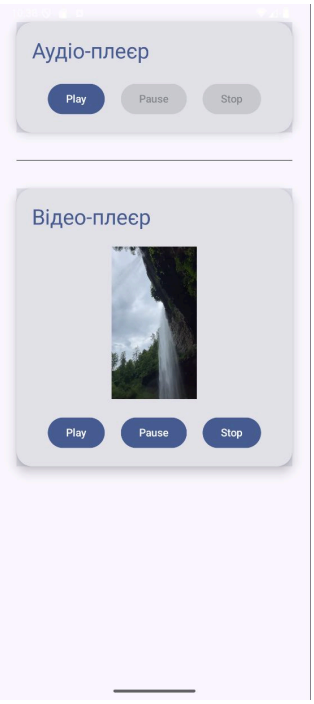

Рис. 1. Аудіо-плеєр: початковий екран

- Користувач ще не натиснув Play у секції “Аудіо-плеєр”.

Рис. 2. Аудіо-плеєр: грає плеєєр

- Користувач натиснув Play у секції “Аудіо-плеєр”.

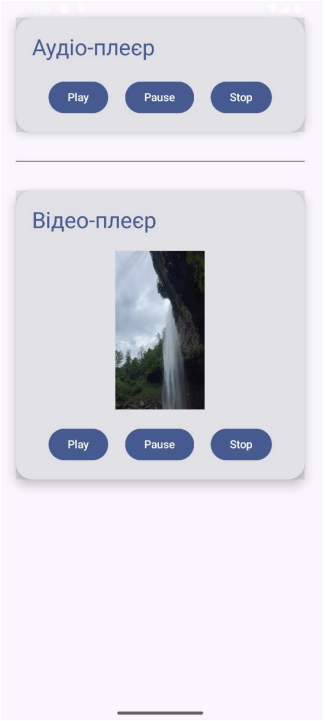

Рис. 3. Відео-плеєр: натискання Play (відео відтворюється)

- Користувач у секції “Відео-плеєр” натиснув Play - увімкнулося відео.

Висновок:

У лабораторній роботі №4 реалізовано Android-застосунок із двома незалежними плеєрами: аудіо- та відео- плеєрами. За допомогою класів MediaPlayer (для аудіо) та VideoView (для відео) реалізовано базові дії: Play, Pause, Stop, які працюють із ресурсами у папці res/raw. Було вивчено:

- Як налаштовувати MediaPlayer — створення, запуск, пауза, зупинка.
- Як налаштовувати VideoView — запускати, призупиняти, зупиняти.
- Як синхронізувати стани кнопок (активний/неактивний) залежно від стану відтворення.

Зроблено серію скріншотів, що демонструють роботу програми у різних станах. Отриманий досвід можна застосовувати для створення складніших медіа-застосунків (наприклад, із відтворенням потокового відео або з еквалайзером).

Контрольні запитання:

1. **Способи підключення Інтернет-ресурсів до мобільного застосунку**
    - HTTP/HTTPS-запити за допомогою HttpUrlConnection, або бібліотек: Volley, Retrofit, OkHttp.
    - WebView — вбудовує веб-сторінки в Activity.
    - Sync Adapter, WorkManager, Firebase Cloud Messaging (FCM) для обміну даними з сервером у фоновому режимі.
    - Socket-з’єднання (TCP/UDP) для реального часу (ігри, чати) за допомогою Socket або сторонніх бібліотек (наприклад, Socket.IO).

2. **Різниця між внутрішнім та зовнішнім сховищем**
    - Internal Storage: приватний простір програми (/data/data/<package>). Інші додатки не бачать файли, не вимагає дозволів. Після видалення програми всі файли зникають.
    - External Storage: спільна пам’ять (SD-карта чи внутрішня флеш-пам’ять, доступна всім додаткам). Потребує дозволів

READ_EXTERNAL_STORAGE / WRITE_EXTERNAL_STORAGE (для старих Android), а на нових — запит “Storage access” через SAF / scoped storage. Дані можуть залишатися після видалення програми (якщо не використовувати “app-specific” зовнішню папку).

3. **Категорії файлів при збереженні в зовнішньому сховищі**
    - Media categories:
        - Environment.DIRECTORY_MUSIC (аудіофайли),
        - Environment.DIRECTORY_PICTURES (зображення),
        - Environment.DIRECTORY_MOVIES (відео),
        - Environment.DIRECTORY_DCIM (фотокамера / камера),
        - Environment.DIRECTORY_DOWNLOADS (завантаження).
    - App-specific directories (для кожного додатку — getExternalFilesDir(...)), доступні лише даному додатку й очищуються після його видалення.
    - Cache (кеш-каталог): getExternalCacheDir() для тимчасових файлів, які система може видалити у разі нестачі місця.

4. **Властивості спеціалізованих інструментів для відтворення аудіо-файлів**
    - MediaPlayer (стандартний API Android):
        - Підтримка форматів MP3, WAV, AAC тощо;
        - Методи: .create(), .start(), .pause(), .stop(), .seekTo(), .release().
        - Події: setOnPreparedListener, setOnCompletionListener, setOnErrorListener.
        - Потрібно вручну керувати ресурсами (звільняти release()), слід запускати у спрощеному життєвому циклі (prepare → start → stop → release).
    - ExoPlayer (більш гнучкий від Google):
        - Підтримує потокове відтворення (HTTP / HLS / DASH), адаптивний бітрейт, субтитри;
        - Легко інтегрується із RecyclerView для переліку треків;
        - Має модульну архітектуру й розширювані компоненти (AudioRenderer, VideoRenderer);

        - Автоматично керує буферизацією, підтримує DRM;
        - Більший розмір APK, складніша конфігурація, але потужніший у потокових сценаріях.

5. **Властивості спеціалізованих інструментів для відтворення відео-файлів**
    - VideoView (спрощений API):
        - Швидко налаштовується: setVideoURI(), start(), pause(), stopPlayback().
        - Має вбудований контролер (MediaController), який можна під'єднати командою videoView.setMediaController(mediaController).
        - Підтримує локальні ресурси (res/raw/...) або URIs (http://...).
        - Обмежений контролем (не підходить для кастомних UI), менше можливостей налаштування буферизації.
    - ExoPlayer (як VideoView, але просунутіше):
        - Підтримує адаптивний стрімінг (HLS, DASH), DRM, кастомні рендерери.
        - Дає повний контроль над UI (можна робити свої кнопки, прогрес-бар).
        - Підтримує одночасне відтворення кількох канальців, можливість прискорювати/уповільнювати відтворення.
        - Більш складний у налаштуванні, потребує додаткових залежностей у Gradle.