Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №4

з дисципліни «Розробка мобільних застосувань під Android»

на тему

«ДОСЛІДЖЕННЯ СПОСОБІВ РОБОТИ З МЕДІАДАНИМИ»

Виконав:
студент групи ІС-21
Качалов А. М.

Викладач:
Орленко С. П.

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.

ЗАВДАННЯ

БАЗОВЕ (12/20 балів). 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.

ПОВНЕ (20/20). Функціональність базового додатку додатково розширюється наступними можливостями: - надати вибір типу файлу для відтворення (аудіо або відео) з будь-якого сховища на мобільному пристрої; - надати вибір завантаження файлу з Інтернету; - використовувати для реалізації обробки медіа-даних спеціалізовані інструменти (особливу увагу приділити програванню відео). Примітка: конкретних вимог до дизайну та вибору інструментів для виконання лабораторної роботи не передбачено, студент сам формує вигляд програми.

HARD TASK (не обов’язково). Конкретного завдання модифікацій додатку з попереднього завдання немає, студент сам обирає додаткову функціональність програми (можливі і інші варіанти), наприклад: - формування декількох списків відтворення (плейлистів), додавання можливості навігації по ним (вибір конкретного файлу, відтворення наступного/ попереднього треку, можливість «перемішати» їх); - передбачити виведення додаткових відомостей про аудіофайл, таких як назва пісні, виконавець, альбом та його обкладинка, рік випуску і т.д.; - відтворення музики у фоновому режимі з плейлиста; - можливість завантажувати нові файли з Інтернету та додавати їх до плейлиста (можна використовувати і онлайн сервіси для конвертації відео в аудіо); - реалізувати радіоплеєр (AM/FM або цифрове).

Хід роботи

1. Програмна реалізація

У програмній частині лабораторної роботи реалізовано мультимедійний мобільний застосунок для операційної системи Android, який дозволяє користувачеві відтворювати аудіо- та відеофайли з локальної пам’яті пристрою або за URL-посиланням. Основна логіка розміщена у класі MainActivity, що є центральною активністю додатку. У межах цього класу відбувається ініціалізація інтерфейсних елементів, налаштування плеєрів, обробка подій взаємодії з користувачем і керування процесами відтворення медіаконтенту. Спочатку оголошуються змінні для керування інтерфейсом: відеоплеєр, зображення обкладинки, назва обраного файлу, текстове поле для відображення швидкості відтворення та повзунок для її зміни. Також ініціалізуються два плеєри: стандартний MediaPlayer для аудіо та потужніший ExoPlayer для відтворення відео.

У нашому додатку реалізовано дві окремі процедури вибору файлу: одна — для аудіо, інша — для відео. При виборі аудіофайлу відбувається його підготовка до програвання через MediaPlayer, встановлюється відповідне зображення для обкладинки та автоматично запускається відтворення. Для відеофайлів використовується ExoPlayer, який виводить відео на спеціальний екран плеєра. У обох випадках додаток коректно зупиняє інший тип медіа, якщо він був активний, що дозволяє уникнути накладання звуків або візуального контенту.

```
//Вибір аудіофайлу та попереднє налаштування
private val audioPickerLauncher = registerForActivityResult(ActivityResultContracts.GetContent()) { uri ->
    uri?.let {
        //Передача у URI змінних
        isAudio = true
        selectedAudioUri = it
        txtFileName.text = getFileName(it) //Отримання імені файлу

        //Демонстрація обкладинки для аудіо
        imageCover.visibility = View.VISIBLE
        playerView.visibility = View.GONE
        imageCover.setImageResource(R.drawable.audio_placeholder)

        //Зупинка відео, для того щоб вони не накладались
        exoPlayer?.stop()
        exoPlayer?.release()
        exoPlayer = null

        //Підготовка плеєра для аудіо
        mediaPlayer?.release()
        mediaPlayer = android.media.MediaPlayer().apply {
            setDataSource( context: this@MainActivity, it) //Встановлення URI
            prepare() //Підготовка плеєра

        }
        //Встановлення шкали швидкості на значенні 1x
        seekBarSpeed.progress = 67
        mediaPlayer?.seekTo( msec: 67)
        mediaPlayer?.start()

    }
}
```

Рис. 1 – Фрагмент коду з реалізацією аудіоплеєра

```
//Вибір відеофайлу та попереднє налаштування
private val videoPickerLauncher = registerForActivityResult(ActivityResultContracts.GetContent()) { uri ->
    uri?.let {
        //Передача у URI змінних
        isAudio = false
        selectedVideoUri = it
        txtFileName.text = getFileName(it) //Отримання імені файлу

        //Демонстрація обкладинки для відео
        imageCover.visibility = View.GONE
        playerView.visibility = View.VISIBLE

        //Зупинка аудіо, для того щоб вони не накладались
        mediaPlayer?.stop()
        mediaPlayer?.release()
        mediaPlayer = null

        // Ініціалізуємо та налаштовуємо плеєр ExoPlayer
        exoPlayer?.release()
        exoPlayer = ExoPlayer.Builder( context: this).build()
        playerView.player = exoPlayer
        val mediaItem = MediaItem.fromUri(it)
        exoPlayer!!.setMediaItem(mediaItem)
        exoPlayer!!.prepare()

        //Встановлення шкали швидкості на значенні 1x
        exoPlayer?.seekTo( positionMs: 67)
        seekBarSpeed.progress = 67
        exoPlayer?.play()
    }
}
```

Рис. 2 – Фрагмент коду з реалізацією відеоплеєра

Особливістю застосунку є можливість керування швидкістю відтворення. Через повзунок користувач може встановити бажане значення у межах від 0.5x до 2.0x, що дає змогу адаптувати прослуховування або перегляд до власних потреб. Швидкість застосовується як до аудіо, так і до відео. Крім того, реалізовано функціональність завантаження медіафайлу з мережі. Користувач вводить URL у спеціальному діалоговому вікні, після чого додаток у фоновому потоці завантажує файл, зберігає його у тимчасовій пам’яті, визначає його тип за розширенням і готує до відтворення.

```
//Встановлення слухача для відслідковування зміни швидкості
seekBarSpeed.setOnSeekBarChangeListener(object : SeekBar.OnSeekBarChangeListener {
    override fun onProgressChanged(seekBar: SeekBar?, progress: Int, fromUser: Boolean) {
        val speed = 0.5f + progress / 200f * 1.5f //Діапазон від 0.5 до 2x
        txtSpeedValue.text = String.format("%.2fx", speed) //Зміна тексту відображення

        //В залежності від медіа встановлення швидкості
        if (isAudio && mediaPlayer != null) {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                val params = mediaPlayer!!.playbackParams ?: android.media.PlaybackParams()
                params.speed = speed
                mediaPlayer!!.playbackParams = params
            }
        } else if (!isAudio && exoPlayer != null) {
            exoPlayer!!.setPlaybackParameters(PlaybackParameters(speed))
        }
    }

    override fun onStartTrackingTouch(seekBar: SeekBar?) {}
    override fun onStopTrackingTouch(seekBar: SeekBar?) {}
})

//Встановлення шкали швидкості на значенні 1x
seekBarSpeed.progress = 67
mediaPlayer?.seekTo( msec: 67)
```

Рис. 3 – Фрагмент коду з реалізацією швидкості програвання медіафайлів

Також доступні елементи керування — кнопки "Play", "Pause" та "Stop", які відповідно запускають, призупиняють або зупиняють відтворення поточного медіа. Усі дії перевіряються залежно від того, який тип файлу активний, що

забезпечує правильну поведінку додатку у різних сценаріях. Наприкінці активності передбачено коректне звільнення ресурсів - зупинка плеєрів, що є важливим для уникнення витоків пам’яті.

```
//Обробник кнопки Play
findViewById<Button>(R.id.btnPlay).setOnClickListener {
    //Перевірка для відтворення
    if (isAudio) {
        if (mediaPlayer == null && selectedAudioUri != null) {
            mediaPlayer = android.media.MediaPlayer().apply {
                setDataSource( context: this@MainActivity, selectedAudioUri!!)
                prepare()
            }
        }
        mediaPlayer?.start()
    } else {
        exoPlayer?.play()
    }
}

//Обробник кнопки Pause
findViewById<Button>(R.id.btnPause).setOnClickListener {
    //Перевірка для паузи
    if (isAudio && mediaPlayer?.isPlaying == true) {
        mediaPlayer?.pause()
    } else if (!isAudio && exoPlayer?.isPlaying == true) {
        exoPlayer?.pause()
    }
}

//Обробник кнопки Stop
findViewById<Button>(R.id.btnStop).setOnClickListener {
    //Перевірка для скидання
    if (isAudio && mediaPlayer != null) {
        mediaPlayer?.stop()
        mediaPlayer?.release()
        mediaPlayer = null
    } else if (!isAudio && exoPlayer != null) {
        exoPlayer?.pause()
        exoPlayer?.seekTo( positionMs: 0)
    }
```

Рис. 4 – Фрагмент коду з реалізацією кнопок "Play", "Pause" та "Stop"

У файлі build.gradle.kts додали бібліотеку для медіафайлів **exoplayer**.

```
dependencies {
    implementation("com.google.android.exoplayer:exoplayer:2.19.1")
    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.appcompat)
    implementation(libs.material)
    implementation(libs.androidx.activity)
    implementation(libs.androidx.constraintlayout)
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
}
```

Рис. 5 – фрагмент коду з додаванням exoplayer

У файлі AnfroidManifest реалізовано вимогу лабораторної роботи до нашого застосунку, а саме дозвіл на «прочитування» внутрішнього сховища на телефоні та підключення до мережі Інтернет.

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

Рис. 6 – фрагмент коду з додаванням дозволів

У файлі activity_main реалізовано зручний і логічно структурований графічний інтерфейс користувача для роботи з медіафайлами. Основою є вертикальний LinearLayout, що містить усі елементи в порядку зверху вниз.

На початку розташовано горизонтальну панель з трьома кнопками: вибір аудіофайлу, завантаження за URL-посиланням та вибір відеофайлу. Далі йде текстовий блок, що показує назву обраного файлу. Для візуального представлення медіа передбачено два компоненти: ImageView для обкладинки аудіо та PlayerView для відтворення відео. Обидва елементи перемикаються в залежності від типу обраного файлу.Нижче розміщено ще одну горизонтальну

панель із кнопками керування відтворенням — Play, Pause та Stop. Завершує інтерфейс панель керування швидкістю відтворення, яка включає підпис, повзунок (SeekBar) та динамічне текстове поле зі значенням швидкості.

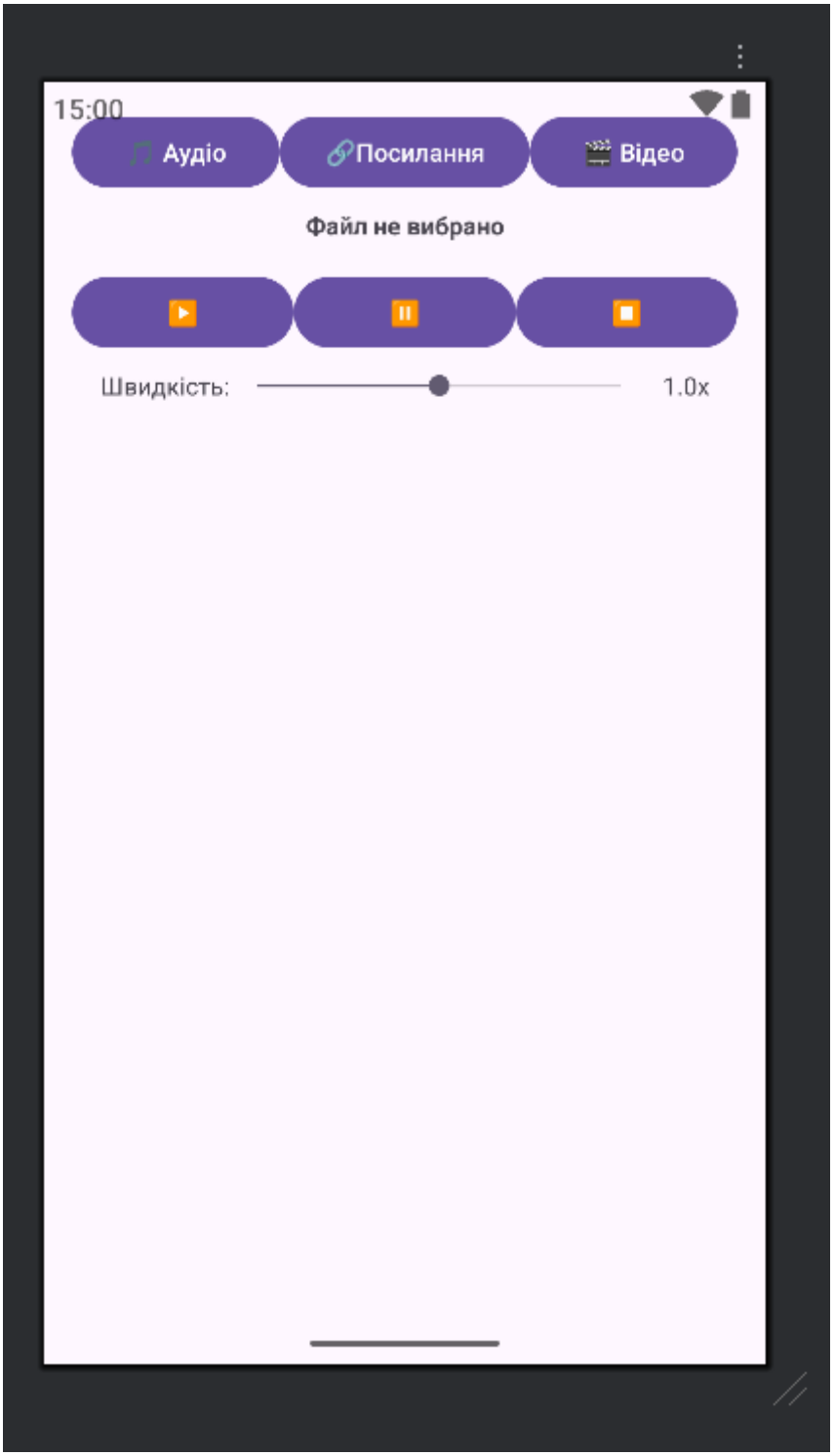

Рис. 7 – Зовнішній вигляд застосунку

Висновок

У ході виконання лабораторної роботи було досліджено, як платформа Android забезпечує обробку аудіо- та відеофайлів шляхом використання вбудованих інструментів для відтворення медіа. Здобуто практичні навички роботи з компонентами MediaPlayer та ExoPlayer, реалізовано вибір локальних медіафайлів, завантаження з інтернету за URL-посиланням, а також керування процесом відтворення (відтворення, пауза, зупинка) і зміна швидкості програвання. Отримані результати засвідчили гнучкість та ефективність Android-середовища у створенні інтерактивних медіазастосунків, а також поглибили розуміння принципів побудови сучасних мобільних інтерфейсів.

Контрольні питання

1. Підключення інтернет-ресурсів до мобільного застосунку в Android здійснюється кількома способами, залежно від потреб розробника. Найбільш поширеним підходом є використання HTTP або HTTPS-запитів для взаємодії з віддаленими серверами через REST API. Для цього можна використовувати вбудований клас HttpURLConnection або сторонні бібліотеки, такі як Retrofit чи Volley, які значно спрощують роботу з мережею. Також застосунок може використовувати WebView для безпосереднього відображення веб-сторінок усередині інтерфейсу. Щоб забезпечити безпечну та стабільну роботу, необхідно перевіряти наявність інтернет-з'єднання та запитувати відповідні дозволи в AndroidManifest.xml.
2. Внутрішнє сховище — це частина пам'яті пристрою, до якої має доступ лише сам додаток, тому вона підходить для зберігання конфіденційних або службових даних. Дані, що зберігаються у внутрішньому сховищі, автоматично видаляються під час деінсталяції застосунку. Зовнішнє сховище, навпаки, є загальнодоступним для інших додатків і користувача, і часто реалізується через SD-карту або спеціальний розділ внутрішньої пам'яті. Воно використовується для зберігання великих файлів, таких як медіаконтент або документи, однак потребує надання дозволів на читання або запис.
3. При збереженні файлів у зовнішньому сховищі Android класифікує їх за категоріями залежно від типу контенту. Серед основних категорій файлів можна виділити зображення, відео, аудіо, документи та завантаження. Для кожної категорії існують окремі публічні директорії, наприклад, папки «Pictures», «Music», «Movies» або «Downloads». Це дозволяє іншим застосункам, зокрема галереї або музичним плеєрам, автоматично розпізнавати відповідний контент. Правильна класифікація спрощує навігацію для користувача і забезпечує дотримання стандартів платформи.
4. Для відтворення аудіофайлів у Android використовуються спеціалізовані інструменти, такі як клас MediaPlayer та бібліотека ExoPlayer. MediaPlayer є

простим у використанні та підходить для відтворення локальних і потокових аудіо. Він підтримує основні формати, такі як MP3, AAC і WAV. ExoPlayer — це більш гнучке рішення, розроблене Google, яке підтримує адаптивне потокове передавання, кешування, зміну швидкості відтворення та інші розширені можливості. Обидва інструменти дозволяють керувати програванням, зупинкою, паузою, перемотуванням та прослуховуванням подій завершення чи помилок.

5. Для відтворення відеофайлів у Android також використовуються MediaPlayer та ExoPlayer, але, окрім цього, можна застосовувати компонент VideoView. MediaPlayer дозволяє відтворювати відео як із локальних джерел, так і з мережі, однак вимагає ручного управління поверхнею виводу, наприклад через SurfaceView. VideoView є обгорткою над MediaPlayer і забезпечує просте відображення відео у вигляді компонента інтерфейсу. ExoPlayer підтримує сучасні кодеки, DASH, HLS та DRM-захист, тому рекомендується для складніших випадків, зокрема відеострімінгу. Ці інструменти надають засоби контролю відтворення, керування буферизацією, масштабуванням і взаємодією з користувачем.