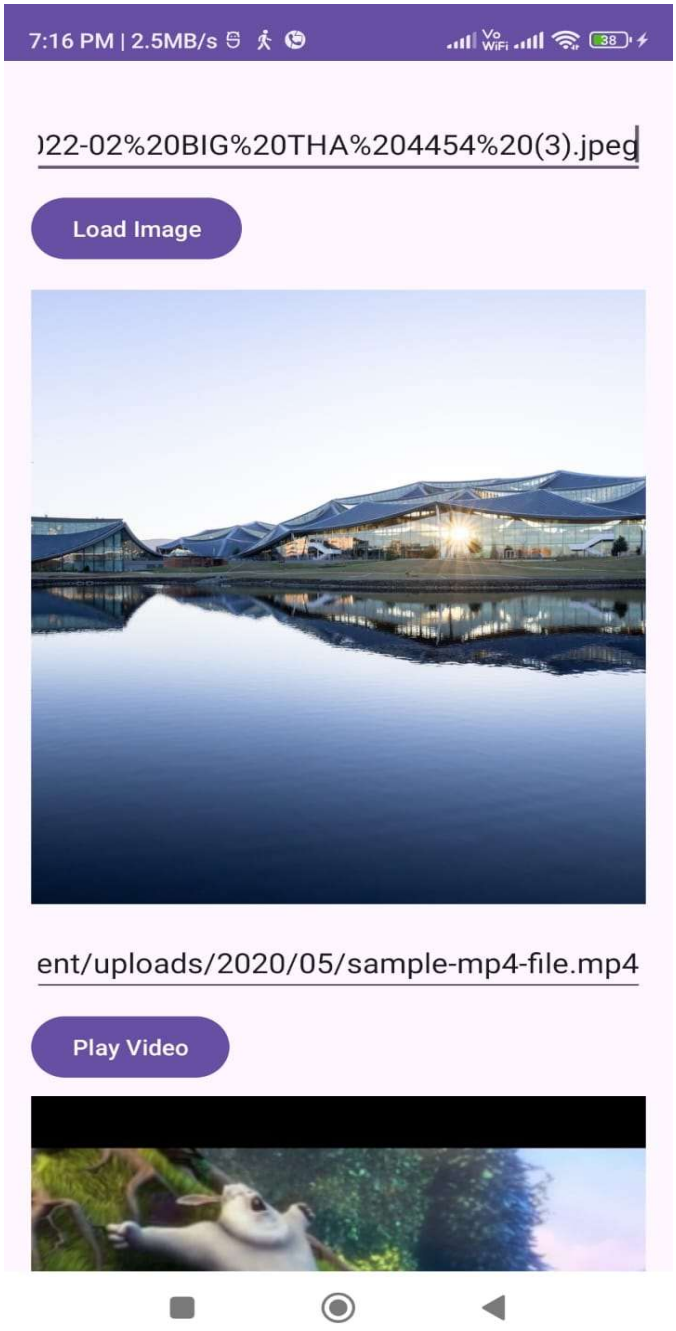
Saurabh Singh

12110796

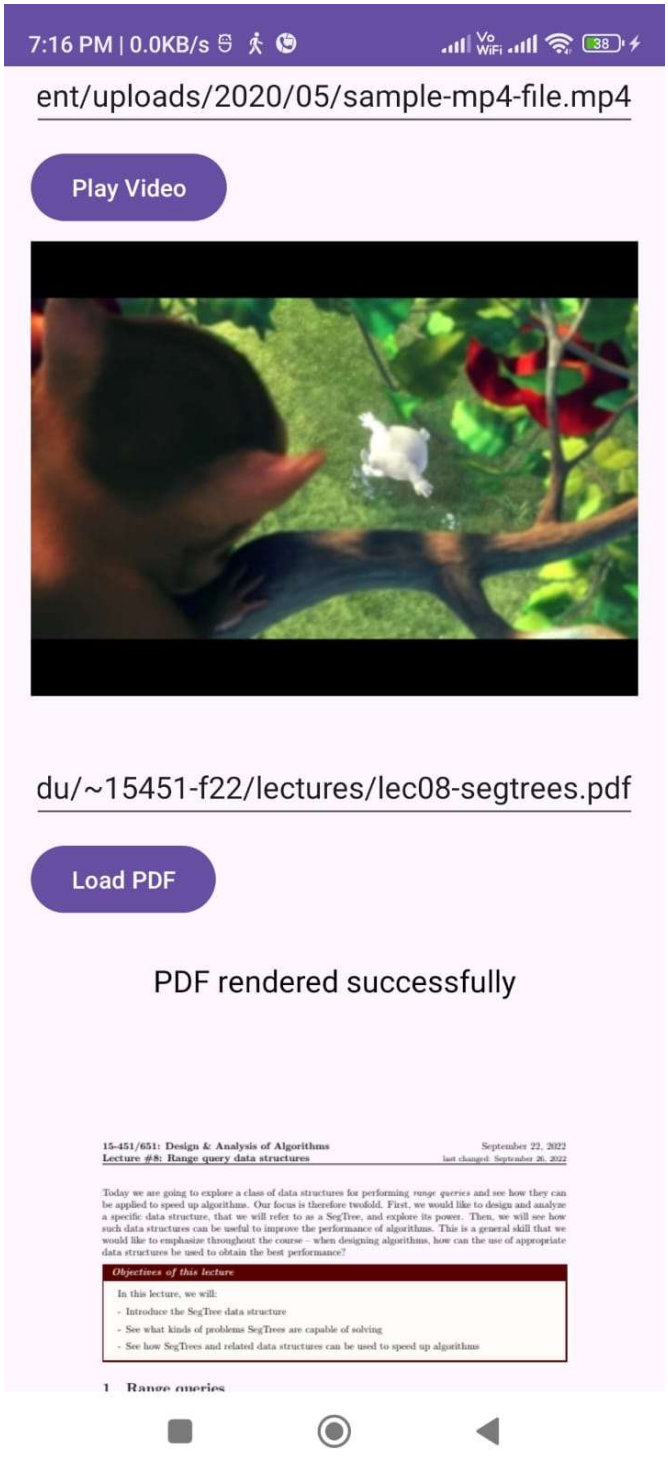
44

Q3. Create an Android app using Kotlin Couroutines to implement a background service that processes image, PDF file, videos tasks efficiently.

Screenshots



Loading Image



Loading video and rendering pdf

XML Code with explanations

The XML layout utilizes a ScrollView to ensure that all components fit within the screen, allowing users to scroll vertically if the content exceeds the screen height. Inside the ScrollView, a LinearLayout arranges the components vertically with a 16dp padding for better spacing.

The layout includes fields for user input and corresponding action buttons. Users can enter URLs for an image, a video, and a PDF, and then use the associated buttons to load and display each type of content.

The ImageView component is used to display the image fetched from the provided URL, while the VideoView plays the video. For PDF files, the layout provides a TextView to indicate loading progress and another ImageView to display the first page of the PDF.

This arrangement ensures that all user inputs and outputs are accessible and neatly organized, improving the overall usability of the app.

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

    <LinearLayout
        android:id="@+id/main"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp">

        <!-- Image URL Input -->
        <EditText
            android:id="@+id/imageUrlInput"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Enter Image URL"
            android:inputType="textUri"
            android:layout_marginTop="16dp"/>

        <!-- Load Image Button -->
        <Button
            android:id="@+id/loadImageButton"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Load Image"
    android:layout_marginTop="8dp"/>

<!-- Coroutine ImageView -->
<ImageView
    android:id="@+id/coroutineImage"
    android:layout_width="400dp"
    android:layout_height="400dp"
    android:contentDescription="Image Desc"
    android:layout_marginTop="16dp"/>

<!-- Video URL Input -->
<EditText
    android:id="@+id/videoUrlInput"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Enter Video URL"
    android:inputType="textUri"
    android:layout_marginTop="16dp"/>

<!-- Play Video Button -->
<Button
    android:id="@+id/buttonPlayVideo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Play Video"
    android:layout_marginTop="8dp"/>

<!-- Video View -->
<VideoView
    android:id="@+id/videoView"
    android:layout_width="match_parent"
    android:layout_height="300dp"
    android:layout_marginTop="8dp"
    android:layout_marginBottom="16dp"/>

<!-- PDF URL Input -->
<EditText
    android:id="@+id/pdfUrlInput"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Enter PDF URL"
```

```
        android:inputType="textUri"
        android:layout_marginTop="16dp"/>

    <!-- Load PDF Button -->
    <Button
        android:id="@+id/loadPdfButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Load PDF"
        android:layout_marginTop="8dp"/>

    <!-- PDF Loading Text -->
    <TextView
        android:id="@+id/pdfTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Loading PDF from the internet"
        android:textSize="18sp"
        android:textColor="@color/black"
        android:gravity="center"
        android:padding="8dp"
        android:layout_marginBottom="16dp"
        android:layout_marginTop="16dp"/>

    <!-- PDF Image View -->
    <ImageView
        android:id="@+id/pdfImageView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:adjustViewBounds="true"
        android:scaleType="fitCenter"
        android:layout_marginTop="16dp"/>

  </LinearLayout>
</ScrollView>
```

Kotlin Code with explanations

The Couroutineimp class extends AppCompatActivity and serves as the main activity for the application. It manages the loading and displaying of images, videos, and PDFs from URLs provided by the user. The layout includes various UI components such as ImageView, VideoView, EditText, Button, and TextView, all of which are initialized in the onCreate method.

Image Loading: When the "Load Image" button is clicked, the loadImage function is called with the URL provided by the user. This function uses Kotlin coroutines to perform a network request on the IO dispatcher to fetch the image data. Once the data is retrieved, it is converted into a Bitmap and displayed in the ImageView on the main thread.

Video Playback: Similarly, clicking the "Play Video" button triggers the playVideo function. This function fetches the video URL and sets it to the VideoView, which then starts playback. The video URL is parsed into a Uri and handled asynchronously.

PDF Handling: The "Load PDF" button initiates the loadPdf function. This function starts by displaying a loading message, then downloads the PDF file using downloadPdf. Once the file is successfully downloaded, it is processed to render the first page as an image using the renderPdf function. The resulting image is then displayed in an ImageView, and the status text is updated to reflect the completion of the rendering process.

Throughout the code, coroutines are used to handle network operations and file processing asynchronously, ensuring that the main thread remains responsive and smooth. Each of these operations is performed in the background and updates the UI on the main thread using withContext(Dispatchers.Main).

This approach effectively separates UI updates from background tasks, providing a smooth user experience while handling different types of media content.

```
package com.example.ultiamatecouroutine

import android.graphics.BitmapFactory
import android.graphics.Bitmap
import android.graphics.pdf.PdfRenderer
import android.net.Uri
import android.os.Bundle
import android.os.ParcelFileDescriptor
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import android.widget.TextView
import android.widget.VideoView
import androidx.appcompat.app.AppCompatActivity
import androidx.lifecycle.lifecycleScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
```

```
import kotlinx.coroutines.withContext
import okhttp3.OkHttpClient
import okhttp3.Request
import java.io.File
import java.io.FileOutputStream

class Couroutineimp : AppCompatActivity() {
    private lateinit var imageView: ImageView
    private lateinit var videoView: VideoView
    private lateinit var playVideoButton: Button
    private lateinit var pdfTextView: TextView
    private lateinit var pdfImageView: ImageView
    private lateinit var imageUrlInput: EditText
    private lateinit var videoUrlInput: EditText
    private lateinit var pdfUrlInput: EditText
    private lateinit var loadImageButton: Button
    private lateinit var loadPdfButton: Button

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.allcouroutinedesign)


        imageView = findViewById(R.id.coroutineImage)
        videoView = findViewById(R.id.videoView)
        playVideoButton = findViewById(R.id.buttonPlayVideo)
        pdfTextView = findViewById(R.id.pdfTextView)
        pdfImageView = findViewById(R.id.pdfImageView)
        imageUrlInput = findViewById(R.id.imageUrlInput)
        videoUrlInput = findViewById(R.id.videoUrlInput)
        pdfUrlInput = findViewById(R.id.pdfUrlInput)
        loadImageButton = findViewById(R.id.loadImageButton)
        loadPdfButton = findViewById(R.id.loadPdfButton)


        loadImageButton.setOnClickListener {
            val imageUrl = imageUrlInput.text.toString()
            loadImage(imageUrl)
        }


        playVideoButton.setOnClickListener {
            val videoUrl = videoUrlInput.text.toString()
            playVideo(videoUrl)
        }
```

```
    loadPdfButton.setOnClickListener {
        val pdfUrl = pdfUrlInput.text.toString()
        loadPdf(pdfUrl)
    }
}

private fun loadImage(url: String) {
    lifecycleScope.launch(Dispatchers.IO) {
        val imageData = fetchImage(url)
        if (imageData != null) {
            val bitmap = BitmapFactory.decodeByteArray(imageData, 0, imageData.size)
            withContext(Dispatchers.Main) {
                imageView.setImageBitmap(bitmap)
            }
        }
    }
}

private suspend fun fetchImage(url: String): ByteArray? {
    return withContext(Dispatchers.IO) {
        val client = OkHttpClient()
        val request = Request.Builder().url(url).build()
        val response = client.newCall(request).execute()
        if (response.isSuccessful) {
            response.body?.bytes()
        } else {
            null
        }
    }
}

private fun playVideo(url: String) {
    lifecycleScope.launch {
        val videoUri = fetchVideoUrl(url)
        withContext(Dispatchers.Main) {
            videoUri?.let {
                videoView.setVideoURI(it)
                videoView.start()
            }
        }
    }
}

private suspend fun fetchVideoUrl(url: String): Uri? {
    return withContext(Dispatchers.IO) {
        val client = OkHttpClient()
```

```
        val request = Request.Builder().url(url).build()
        val response = client.newCall(request).execute()
        if (response.isSuccessful) {
            Uri.parse(url)
        } else {
            null
        }
    }
}

private fun loadPdf(url: String) {
    lifecycleScope.launch(Dispatchers.IO) {
        withContext(Dispatchers.Main) {
            pdfTextView.text = "Downloading PDF..."
        }

        val pdfFile = downloadPdf(url)
        if (pdfFile != null) {
            withContext(Dispatchers.Main) {
                pdfTextView.text = "PDF downloaded, rendering the first page..."
            }

            val pdfBitmap = renderPdf(pdfFile)
            pdfBitmap?.let {
                withContext(Dispatchers.Main) {
                    pdfImageView.setImageBitmap(it)
                    pdfTextView.text = "PDF rendered successfully"
                }
            }
        } else {
            withContext(Dispatchers.Main) {
                pdfTextView.text = "Failed to download PDF"
            }
        }
    }
}

private suspend fun downloadPdf(url: String): File? {
    return withContext(Dispatchers.IO) {
        val client = OkHttpClient()
        val request = Request.Builder().url(url).build()
        val response = client.newCall(request).execute()

        if (response.isSuccessful) {
            val pdfFile = File(cacheDir, "Downloaded_pdf.pdf")
            val fos = FileOutputStream(pdfFile)
```

```
                response.body?.byteStream()?.use { inputStream ->
                    fos.use { outputStream ->
                        inputStream.copyTo(outputStream)
                    }
                }
                pdfFile
            } else {
                null
            }
        }
    }

    private suspend fun renderPdf(file: File): Bitmap? {
        return withContext(Dispatchers.IO) {
            val fileDescriptor = ParcelFileDescriptor.open(file,
ParcelFileDescriptor.MODE_READ_ONLY)
            val pdfRenderer = PdfRenderer(fileDescriptor)
            val page = pdfRenderer.openPage(0)

            val width = page.width
            val height = page.height
            val bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888)

            page.render(bitmap, null, null, PdfRenderer.Page.RENDER_MODE_FOR_DISPLAY)
            page.close()
            pdfRenderer.close()
            bitmap
        }
    }
}
```