```kotlin
package com.example.pdfviewer

import android.graphics.pdf.PdfRenderer
import android.os.Bundle
import android.os.ParcelFileDescriptor
import android.widget.Button
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import java.io.File
import java.io.IOException

class PdfActivity : AppCompatActivity() {

    private lateinit var pdfView: VectorPdfView
    private lateinit var btnPrev: Button
    private lateinit var btnNext: Button
    private lateinit var txtInfo: TextView

    private var renderer: PdfRenderer? = null
    private var currentPage: PdfRenderer.Page? = null
    private var descriptor: ParcelFileDescriptor? = null

    private var pageIndex = 0

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

        // Setup UI
        pdfView = findViewById(R.id.vectorPdfView)
        btnPrev = findViewById(R.id.btnPrevious)
        btnNext = findViewById(R.id.btnNext)
        txtInfo = findViewById(R.id.txtPageInfo)

        // 1. Simulate byte array
        val receivedBytes = byteArrayOf() // YOUR DATA HERE

        // 2. Save to temp file
        val file = PdfUtils.saveByteArrayToCache(this, receivedBytes)

        if (file != null) {
            setupRenderer(file)
        } else {
            Toast.makeText(this, "Failed to load PDF data",
Toast.LENGTH_SHORT).show()
        }
        btnPrev.setOnClickListener { showPage(pageIndex - 1) }
        btnNext.setOnClickListener { showPage(pageIndex + 1) }
    }

    private fun setupRenderer(file: File) {
        try {
            descriptor = ParcelFileDescriptor.open(file,
ParcelFileDescriptor.MODE_READ_ONLY)
            renderer = PdfRenderer(descriptor!!)
            showPage(0)
        } catch (e: IOException) {
            e.printStackTrace()
            Toast.makeText(this, "Invalid PDF",
Toast.LENGTH_SHORT).show()
        }
    }

    private fun showPage(index: Int) {
        val r = renderer ?: return
        if (index < 0 || index >= r.pageCount) return

        // CLOSE previous page before opening new one (Required by
API)
        currentPage?.close()

        currentPage = r.openPage(index)
        pageIndex = index

        // Pass the native page to our custom view
        pdfView.showPage(currentPage!!)

        updateUi()
    }

    private fun updateUi() {
        val count = renderer?.pageCount ?: 0
        txtInfo.text = "Page ${pageIndex + 1} of $count"
        btnPrev.isEnabled = pageIndex > 0
        btnNext.isEnabled = pageIndex < count - 1
    }

    override fun onDestroy() {
        super.onDestroy()
        // Clean up strictly to avoid memory leaks or file locks
        currentPage?.close()
        renderer?.close()
        descriptor?.close()
    }
```

```
}
```