

Project Report

Waste Management App

Course Code : CSE226

Course Title: Developing Android Apps

Submitted By:

Name:- Nagakalyan Ravuri

Reg No:- 12220944

Name:- Mandadapu Chandravadhan

Reg No:- 12222491

Name:- A. Akash Goud

Reg No:- 12221995

Submitted To:

DR. Vikas Sharma



L O V E L Y
P R O F E S S I O N A L
U N I V E R S I T Y

LOVELY PROFESSIONAL UNIVERSITY

DECLARATION

I A.Akash Goud a student of Bachelor of Technology under CSE discipline at Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own work and is genuine

Your name: A.Akash Goud

Registrartion Number: 12221995

Table of Contents

1. Project Description

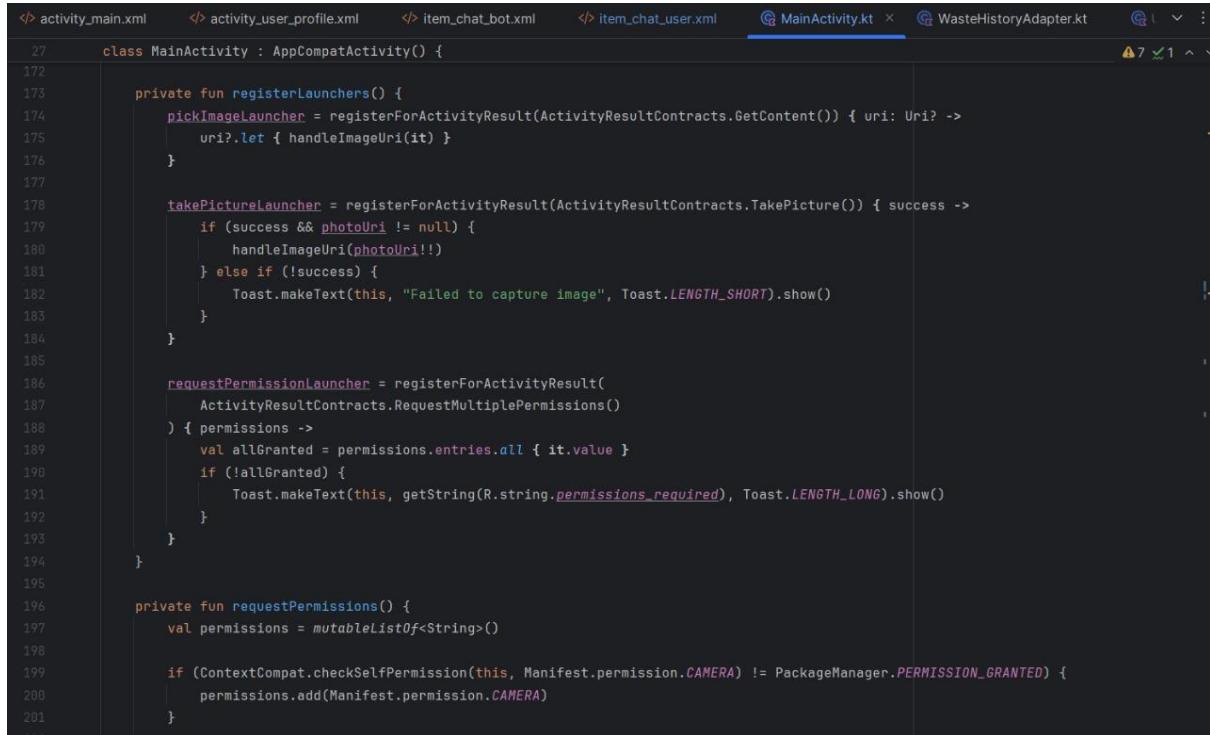
Waste Management is an Android app that blends computer vision and conversational AI to promote responsible disposal habits. Users can capture or pick a waste image, run the TensorFlow Lite classifier on-device, and immediately see the waste category plus confidence. The app then combines the classification with curated disposal instructions and stores both, so each scan contributes to a searchable waste history. A built-in AI chatbot—currently rule-based with Gemini-ready hooks—delivers recycling tips, eco facts, and guidance on tricky materials. The entire experience follows MVVM best practices, uses coroutines for smooth background work, and supports multiple languages via dedicated resource files.

2. Technology Used (Front End/Back End)

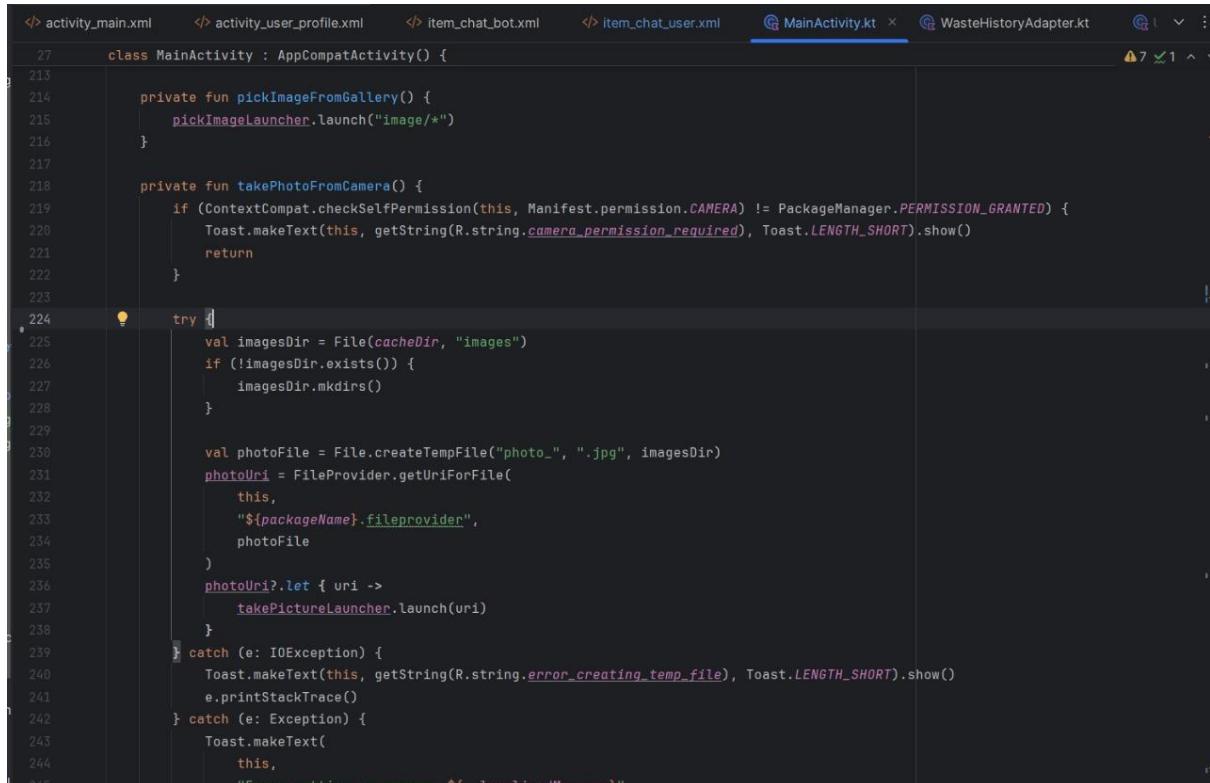
Android Studio, TensorFlow lite, Coroutines, API's.

3. Screenshots of the project

1.Main Activity Code:-

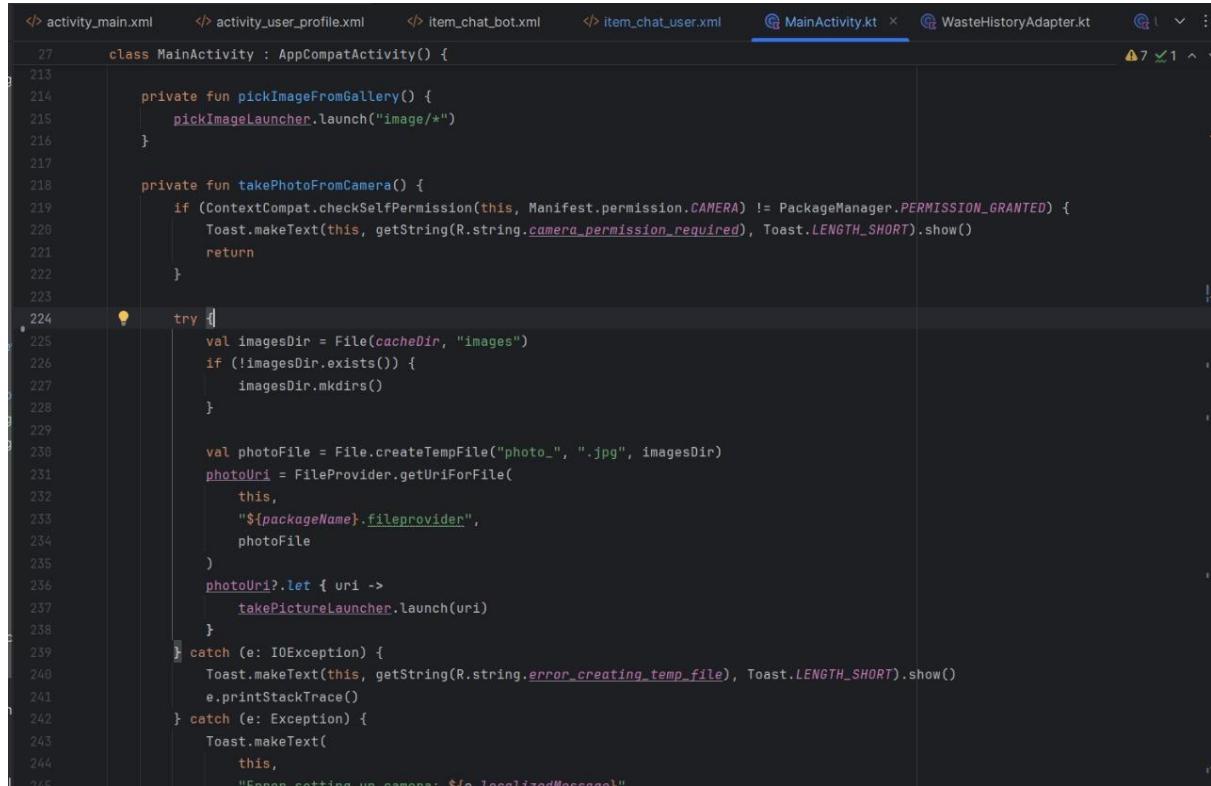


```
27     class MainActivity : AppCompatActivity() {
172
173     private fun registerLaunchers() {
174         pickImageLauncher = registerForActivityResult(ActivityResultContracts.GetContent()) { uri: Uri? ->
175             uri?.let { handleImageUri(it) }
176         }
177
178         takePictureLauncher = registerForActivityResult(ActivityResultContracts.TakePicture()) { success ->
179             if (success && photoUri != null) {
180                 handleImageUri(photoUri!!)
181             } else if (!success) {
182                 Toast.makeText(this, "Failed to capture image", Toast.LENGTH_SHORT).show()
183             }
184         }
185
186         requestPermissionLauncher = registerForActivityResult(
187             ActivityResultContracts.RequestMultiplePermissions()
188         ) { permissions ->
189             val allGranted = permissions.entries.all { it.value }
190             if (!allGranted) {
191                 Toast.makeText(this, getString(R.string.permissions_required), Toast.LENGTH_LONG).show()
192             }
193         }
194     }
195
196     private fun requestPermissions() {
197         val permissions = mutableListOf<String>()
198
199         if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
200             permissions.add(Manifest.permission.CAMERA)
201         }
202     }
203 }
```



```
27     class MainActivity : AppCompatActivity() {
213
214     private fun pickImageFromGallery() {
215         pickImageLauncher.launch("image/*")
216     }
217
218     private fun takePhotoFromCamera() {
219         if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
220             Toast.makeText(this, getString(R.string.camera_permission_required), Toast.LENGTH_SHORT).show()
221             return
222         }
223
224         try {
225             val imagesDir = File(cacheDir, "images")
226             if (!imagesDir.exists()) {
227                 imagesDir.mkdirs()
228             }
229
230             val photoFile = File.createTempFile("photo_", ".jpg", imagesDir)
231             photoUri = FileProvider.getUriForFile(
232                 this,
233                 "${packageName}.fileprovider",
234                 photoFile
235             )
236             photoUri?.let { uri ->
237                 takePictureLauncher.launch(uri)
238             }
239         } catch (e: IOException) {
240             Toast.makeText(this, getString(R.string.error_creating_temp_file), Toast.LENGTH_SHORT).show()
241             e.printStackTrace()
242         } catch (e: Exception) {
243             Toast.makeText(
244                 this,
245                 "Error setting up camera: ${e.localizedMessage}"
246             ).show()
247     }
248 }
```

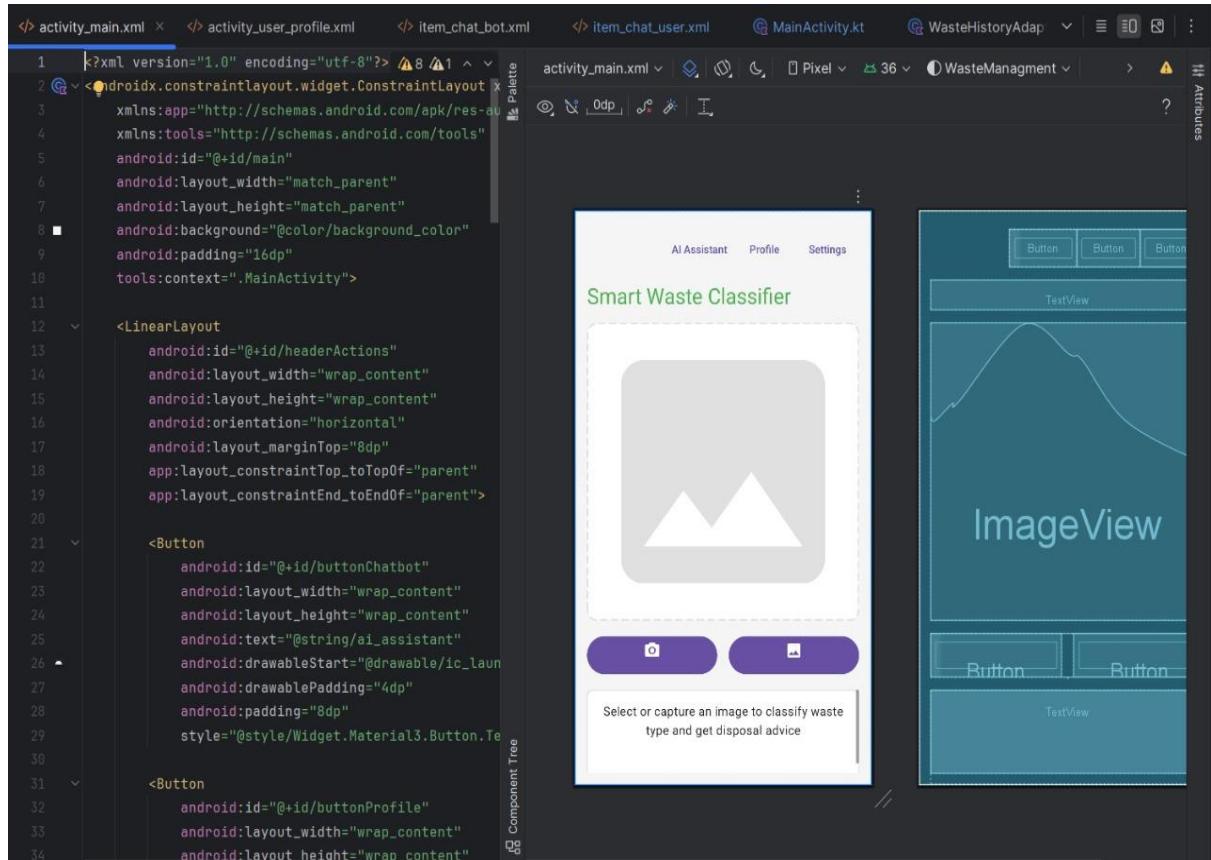
2.Tensor Flow Code:-



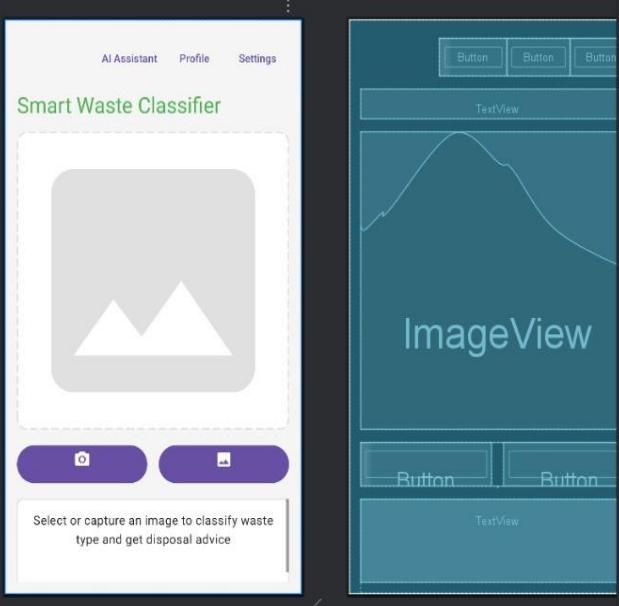
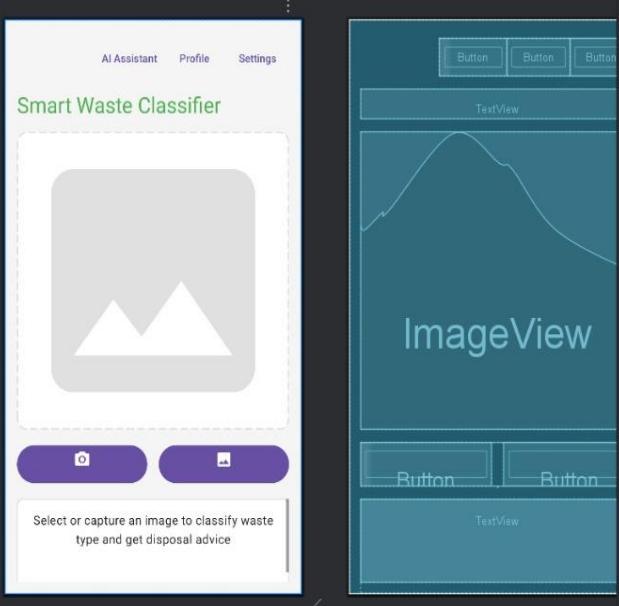
```
activity_main.xml activity_user_profile.xml item_chat_bot.xml item_chat_user.xml MainActivity.kt WasteHistoryAdapter.kt
```

```
27 class MainActivity : AppCompatActivity() {  
28  
29     private fun pickImageFromGallery() {  
30         pickImageLauncher.launch("image/*")  
31     }  
32  
33     private fun takePhotoFromCamera() {  
34         if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {  
35             Toast.makeText(this, getString(R.string.camera_permission_required), Toast.LENGTH_SHORT).show()  
36             return  
37         }  
38  
39         try {  
40             val imagesDir = File(cacheDir, "images")  
41             if (!imagesDir.exists()) {  
42                 imagesDir.mkdirs()  
43             }  
44  
45             val photoFile = File.createTempFile("photo_", ".jpg", imagesDir)  
46             photoUri = FileProvider.getUriForFile(  
47                 this,  
48                 "$packageName".fileprovider,  
49                 photoFile  
50             )  
51             photoUri?.let { uri ->  
52                 takePictureLauncher.launch(uri)  
53             }  
54         } catch (e: IOException) {  
55             Toast.makeText(this, getString(R.string.error_creating_temp_file), Toast.LENGTH_SHORT).show()  
56             e.printStackTrace()  
57         } catch (e: Exception) {  
58             Toast.makeText(  
59                 this,  
60                 "Error setting up camera: ${e.localizedMessage}",  
61                 Toast.LENGTH_SHORT).show()  
62         }  
63     }  
64 }
```

3.Activity Main XML Code:-



```
activity_main.xml activity_user_profile.xml item_chat_bot.xml item_chat_user.xml MainActivity.kt WasteManagement.kt
```

```
<?xml version="1.0" encoding="utf-8"?> A screenshot of the Android Studio interface showing the XML code for activity_main.xml and the corresponding UI preview. The UI features a header with 'AI Assistant', 'Profile', and 'Settings' buttons. Below the header is a large central area containing a white rounded rectangle with a mountain icon, a purple button labeled 'Select or capture an image to classify waste type and get disposal advice', and two smaller purple buttons at the bottom. To the right of this main area is a blue sidebar titled 'Smart Waste Classifier' with sections for 'Button', 'TextView', and 'Image'. The XML code lists various components like ConstraintLayout, LinearLayout, Button, and TextView with their respective attributes.  
1 <?xml version="1.0" encoding="utf-8"?>   
2 <ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools="http://schemas.android.com/tools" android:id="@+id/main" android:layout_width="match_parent" android:layout_height="match_parent" android:background="@color/background_color" android:padding="16dp" tools:context=".MainActivity">  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12 <LinearLayout android:id="@+id/headerActions" android:layout_width="wrap_content" android:layout_height="wrap_content" android:orientation="horizontal" android:layout_marginTop="8dp" app:layout_constraintTop_toTopOf="parent" app:layout_constraintEnd_toEndOf="parent">  
13  
14  
15  
16  
17  
18  
19  
20  
21 <Button android:id="@+id/buttonChatbot" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="@string/ai_assistant" android:drawableStart="@drawable/ic_laun android:drawablePadding="4dp" android:padding="8dp" style="@style/Widget.Material3.Button.Text"/>  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31 <Button android:id="@+id/buttonProfile" android:layout_width="wrap_content" android:layout_height="wrap_content"/>
```


15:03

0.00 0 75% 66%

AI Assistant

Profile

Settings

Smart Waste Classifier



Select or capture an image to classify
waste type and get disposal advice

15:03

0.18 KB/S 5G 66%

Waste management Classifier

)

Conclusion

The project already delivers a cohesive experience: capture/choose waste images, classify them reliably on-device, surface contextual advice, keep a running history, and chat with an assistant for broader sustainability questions. The codebase is cleanly structured (MVVM), dependency-ready for cloud AI, and localized, making it easy to maintain or extend. Even with the mock fallback, the classifier handles common waste categories, ensuring the app stays functional without network or model file issues.

Future Scope

- **Real LLM integration:** swap the rule-based chatbot with OpenAI/Gemini/Claude calls, including streaming replies and better context handling.
- **Realtime camera classification:** continuous preview inference for faster feedback, plus batching multiple photos.
- **Model improvements:** auto-download updated TFLite models, apply quantization, add edge detection or preprocessing filters.
- **Analytics & gamification:** show personal impact stats, leaderboards, streaks, or community challenges.
- **Voice + accessibility:** voice commands, TTS responses, and better screen-reader support.
- **Cloud sync & sharing:** sync history across devices, export PDFs, or share classification details with municipalities.

4. GitHub URL Link

<https://github.com/akashgoudaithagoni/Waste-Management-Android-App1>