ReadMe File PROG7314_Part 2

-By

Sheldon Chettiar (ST10302323)

Teolan Govender (ST10259514)

Shalin Reddy (ST10063069)

Yadav Maganbeharie (ST10350794)

GitHub Link: https://github.com/IIEWFL/prog7314-part-2-4pixels/tree/main

Video Demo Link https://youtu.be/g-cx020so74

Pixel Review Mobile Application

Overview

The **Pixel Review Mobile App** is an Android application developed using **Android Studio** with **Kotlin** (and optionally Java) as the primary programming language. The app is designed to provide users with a modern, interactive platform to explore, review, and engage with digital content or products.

This document serves as a **comprehensive step-by-step guide** for setting up, running, and maintaining the project in a local Android development environment.

Prerequisites

Before getting started, ensure you have the following tools installed and configured:

1. Android Development Setup

1. Android Studio (latest stable version recommended)
   Download: https://developer.android.com/studio
2. Java Development Kit (JDK 17 or later)
   - Android Studio includes a JDK by default.
   - To confirm:
     File > Project Structure > SDK Location > JDK Location
3. Gradle
   - Pre-configured in Android Studio (no manual install required).
4. Android SDK Platform Tools
   - Installed automatically with Android Studio.
5. Device for Testing
   - Either an Android Emulator (via Android Studio)

- Or a physical Android device connected via USB with USB Debugging enabled.

2. Node.js Backend (PixelReviewAPI) Setup

1. Node.js (v18 or later)
   Download: https://nodejs.org/
   - Verify installation:
   - node -v
   - npm -v
2. Visual Studio Code
   Download: https://code.visualstudio.com/
3. Required Global Packages
   - npm install -g nodemon
4. Local Dependencies
   Navigate to your API folder and install required packages:
   - cd pixel-review-api
   - npm install express dotenv cors axios

   Optional (for caching, debugging, etc.):
   - npm install node-cache
5. Project Structure Example

```
pixel-review-api/
├— controllers/
├— routes/
├— utils/
├— index.js
├— package.json
├— .env
```

6. Environment Variables (.env file)
   Create a .env file in your API root:

   PORT=5000

   IGDB / Twitch Credentials

   TWITCH_CLIENT_ID=tt2lte5tbli8lvci797neheriqeobh

   TWITCH_CLIENT_SECRET=w66cw7kps5qw6x3ie7jq1j2yfzeoyb

7. Setting up the API in android studio
   Replace the current code in the retrofitClient class and the security configuration layout with the following and replace the IP address with your own which can be retrieved by running the command ipconfig in the command prompt

   IP Address

```
C:\Users\Teola>ipconfig

Windows IP Configuration
Wireless LAN adapter Local Area Connection* 2:

  Media State . . . . . . . . . . . : Media disconnected
  Connection-specific DNS Suffix  . :

Wireless LAN adapter WiFi:

  Connection-specific DNS Suffix  . :
  Link-local IPv6 Address . . . . . : fe80**************
  IPv4 Address. . . . . . . . . . . : 10.114.224.***  --This IP Address
  Subnet Mask . . . . . . . . . . . : 255************
  Default Gateway . . . . . . . . . : 10.************

Ethernet adapter Bluetooth Network Connection:
```

   Retrofit client class for local API

```
package com.example.bmo.services

import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

object RetrofitClient {

    private const val BASE_URL = "http://YourAPIAddress:5000/api/"

    val api: IgdbApiService by lazy {
        Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
            .create(IgdbApiService::class.java)
    }
}
```

network_security_config.xml

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
  <domain-config cleartextTrafficPermitted="true">
    <!-- For emulator -->
    <domain includeSubdomains="true">10.0.2.2</domain>
    <domain includeSubdomains="true">localhost</domain>

    <!-- For phone -->
    <domain includeSubdomains="true">YOUR IP ADDRESS</domain>
  </domain-config>
</network-security-config>
```

8. Run the API Server
    - In the command prompt run: npm run dev
    - The server will start on:
    - http://localhost:5000

9. Test the API
   Open your browser or Postman and visit:

   http://localhost:5000/api/igdb/trending

3. Running the Render API

The API was hosted using a render webservice. The server enters a state of sleep when not being used, to wake it up launch the app and wait 15 seconds and relaunch the app.

Project Setup

Follow the steps below to properly set up the **Pixel Review** project on your machine.

Step 1: Extract the Project Files

1. Locate the downloaded PixelReview.zip file.
2. Right-click and select **Extract All...** (on Windows) or **Extract Here** (on macOS/Linux).
3. Ensure the folder structure looks as follows:

```
PixelReview/
├── app/
│   ├── src/
│   │   ├── main/
│   │   │   ├── java/com/example/pixelreview/
│   │   │   ├── res/
│   │   │   └── AndroidManifest.xml
│   └── build.gradle
        ├── gradle/
        ├── build.gradle
        ├── settings.gradle
└── README.md
```

Step 2: Open the Project in Android Studio

1. Launch **Android Studio**.
2. Click **File > Open**.
3. Browse to the extracted PixelReview folder.
4. Click **OK** to open the project.
5. Wait for Gradle to sync and complete the initial build (this may take a few minutes).
6. Confirm that no build errors appear in the **Build Output** window.

Step 3: Set Up an Android Emulator (Optional)

1. Go to **Tools > Device Manager**.
2. Click **Create Device**.
3. Select a device model (e.g., *Pixel 7* or *Pixel 6 Pro*) and click **Next**.
4. Choose a system image (Android 13 or newer recommended).
5. Download the image if required, then click **Finish**.
6. Launch the emulator once setup is complete.

Step 4: Connect a Physical Android Device (Optional Alternative)

If you prefer testing on a real Android phone:

1. Enable **Developer Options**:

- Go to **Settings > About phone > Build number**
- Tap the build number **seven times** to activate Developer Mode.

2. Enable **USB Debugging**:
   - Go to **Settings > Developer options > USB debugging** and toggle it on.
3. Connect the device via USB cable.
4. Accept the "Allow USB Debugging" prompt on your device.

Step 5: Build and Run the Application

1. In Android Studio, click the **Run** ▶️ icon on the toolbar.
2. Select your target device (emulator or connected phone).
3. Wait for the build to complete.
4. The **Pixel Review** app should automatically launch on the selected device.

Project Structure

Folder/File	Description
app/src/main/java/	Contains Kotlin/Java source code, including Activities, Fragments, and ViewModels
app/src/main/res/	Stores all app resources (layouts, drawables, strings, and icons)
app/src/main/AndroidManifest.xml	Defines app permissions, entry points, and component declarations
build.gradle	Handles module-level build configuration and dependencies
gradle/	Contains Gradle wrapper files for build automation
settings.gradle	Defines modules included in the build process

Common Development Tasks

Clean and Rebuild the Project

If you experience unexpected build behaviour:

1. Navigate to **Build > Clean Project**.
2. Then select **Build > Rebuild Project**.

Sync Gradle Files

If Gradle dependencies are missing or not resolving:

- Click **File > Sync Project with Gradle Files**.

Update Dependencies

To keep the project compatible with the latest Android SDK:

1. Open build.gradle (Module: app)
2. Check for version update prompts and apply them as needed.

Generating an APK File

To create an installable APK:

1. Go to **Build > Build Bundle(s) / APK(s) > Build APK(s)**.
2. Once complete, click **Locate** in the notification.
3. The file path will be similar to:
4. app/build/outputs/apk/debug/app-debug.apk
5. You can install this APK manually on any Android device by transferring it via USB or uploading it to a shared folder.

Troubleshooting Guide

Issue	**Possible Cause**	**Suggested Solution**
Gradle sync failed	Missing SDK or network connection	Check SDK installation and retry sync
App won't launch	Emulator misconfiguration	Wipe data or recreate the AVD
Build errors	Corrupted cache or dependency mismatch	Run **File > Invalidate Caches / Restart**
Device not detected	USB debugging disabled or cable issue	Re-enable debugging and use a data-capable cable

Developer Notes

- The **Pixel Review** app follows standard Android architecture (MVVM or MVC).
- Use **Logcat** to debug and trace runtime activity.
- Maintain consistent code formatting and naming conventions for better collaboration.
- Update dependencies periodically to stay aligned with Android SDK releases.

Custom Features

1. The app will recommend games based on the genres of games from the user's library.
2. Each Game will have a rating score so users will know if the game is worth purchasing.
3. The recent games recycler view displays the latest games in real time to keep the users updated.

Screenshots

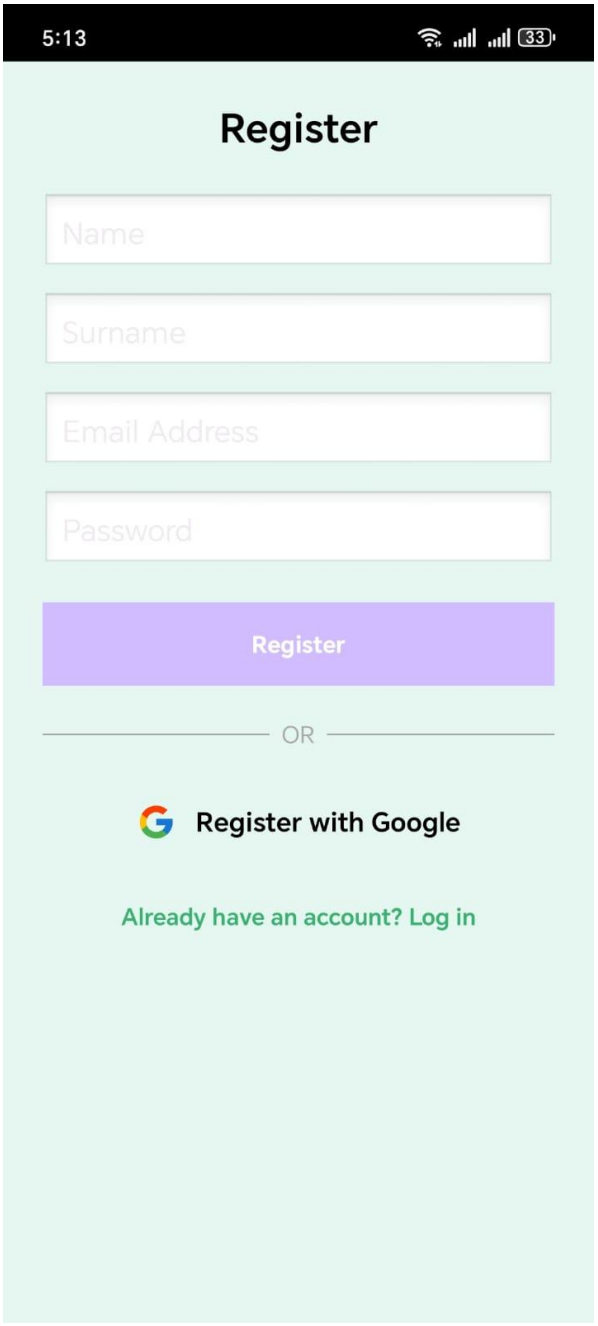


Figure 1 Register page and Login Page

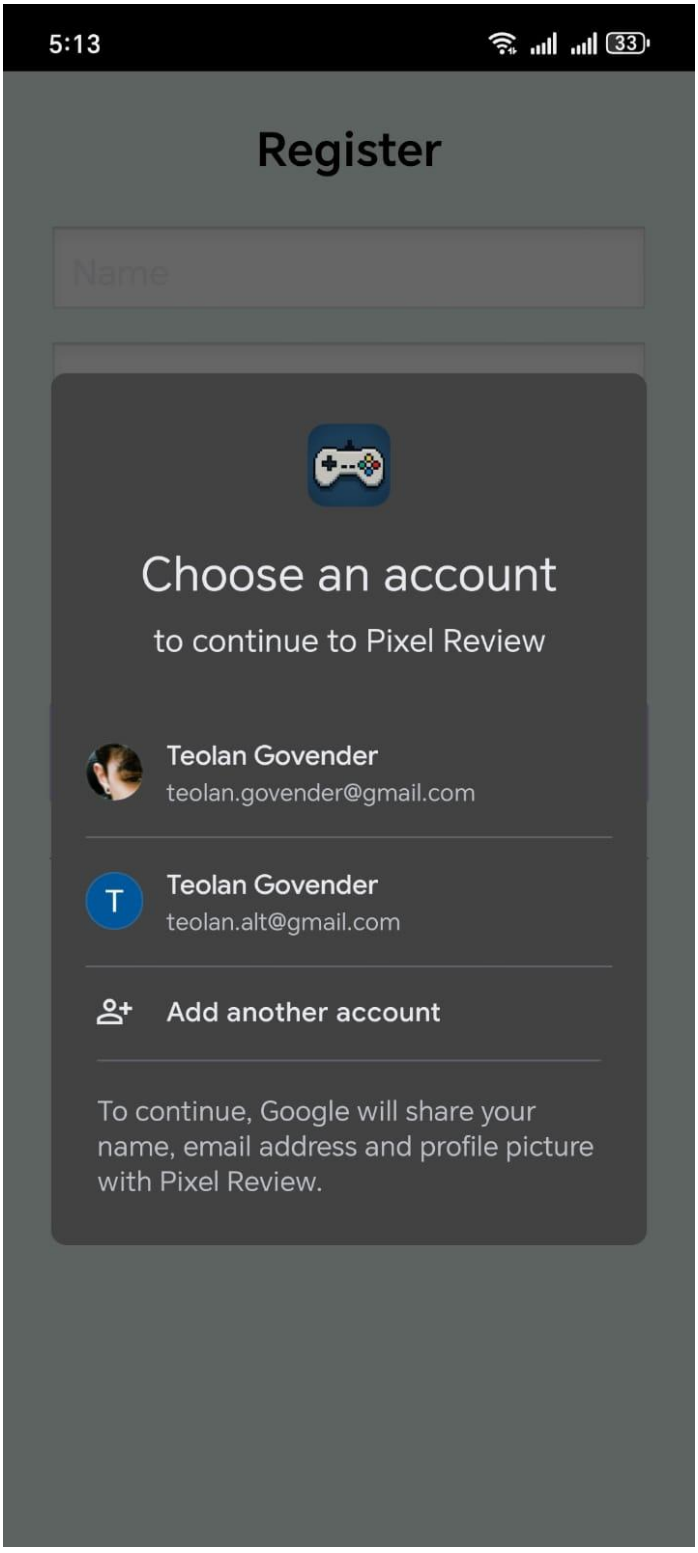


Figure 2 Login Page with SSO

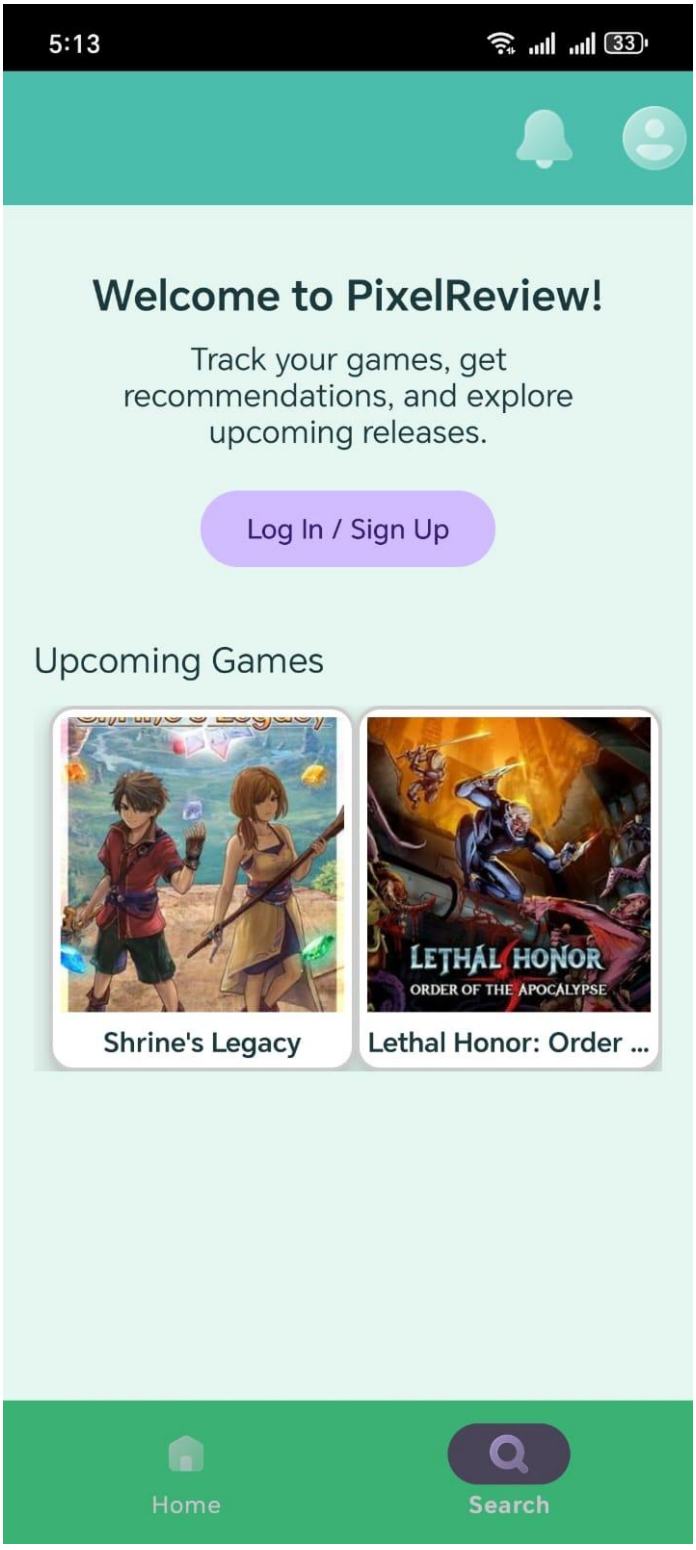


Figure 3 Homepage for Guest users

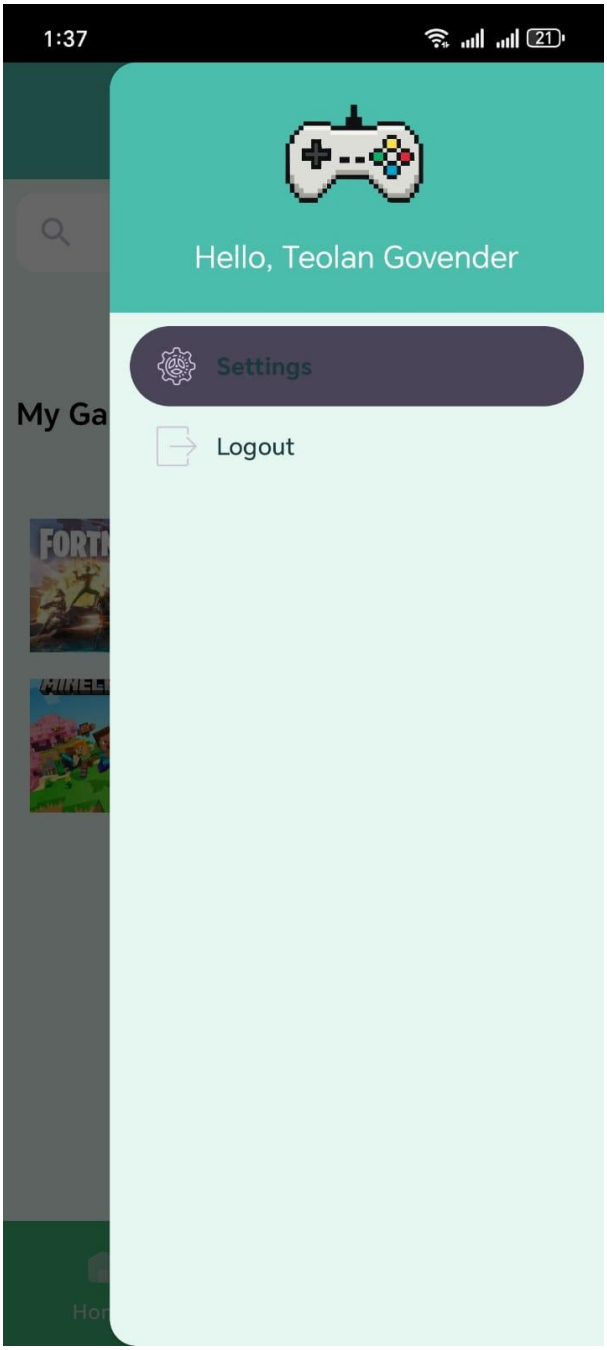


Figure 4 Settings and Logout Page

1:37

Gran T

Filter Sort By

Gran Turismo

3.5/5

Racing, Simulator • PlayStation Po...

Gran Turismo Sport: Spec II

0.0/5

• PlayStation 4 • 2019

Gran Turismo 5 Prologue

3.2/5

Racing, Simulator • PlayStation 3 •...

Gran Turismo

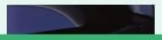
4.1/5

Racing, Simulator • PlayStation • 1...

Math Gran Prix

0.0/5

Puzzle, Racing • Atari 2600 • 1982



Home Search Library

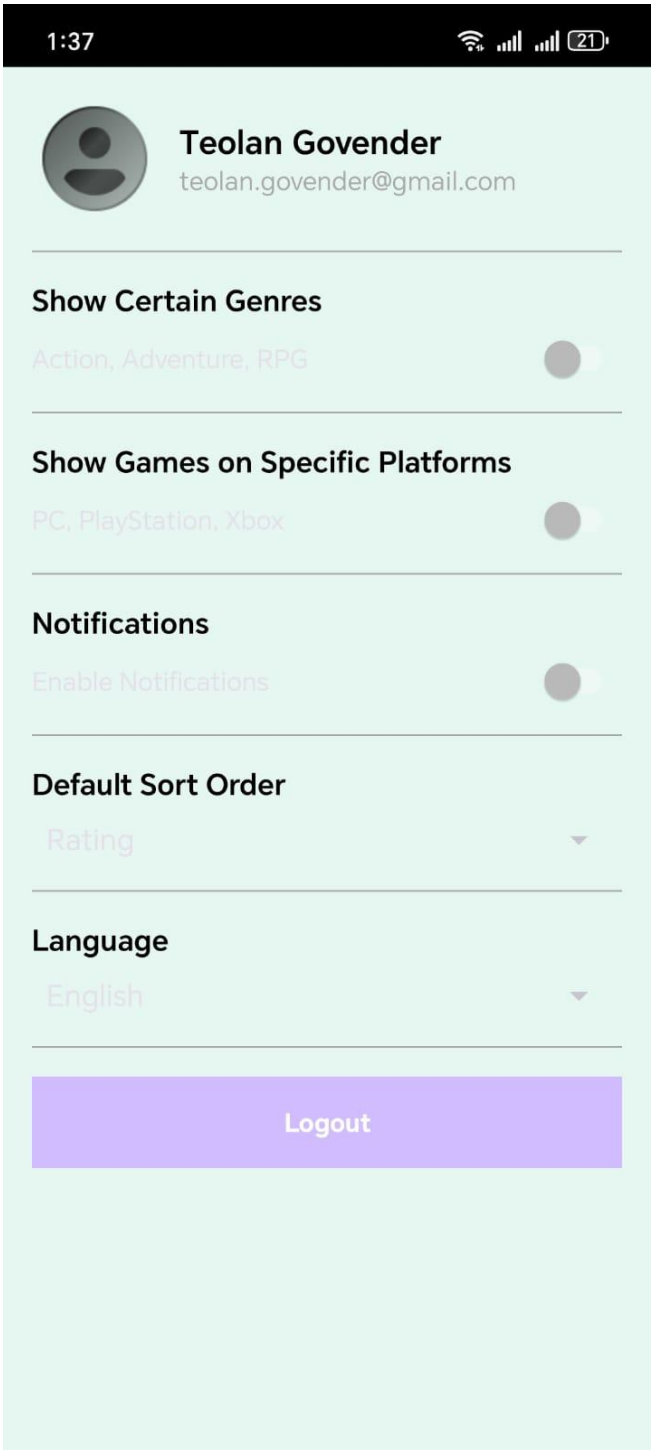


Figure 5 Settings Page

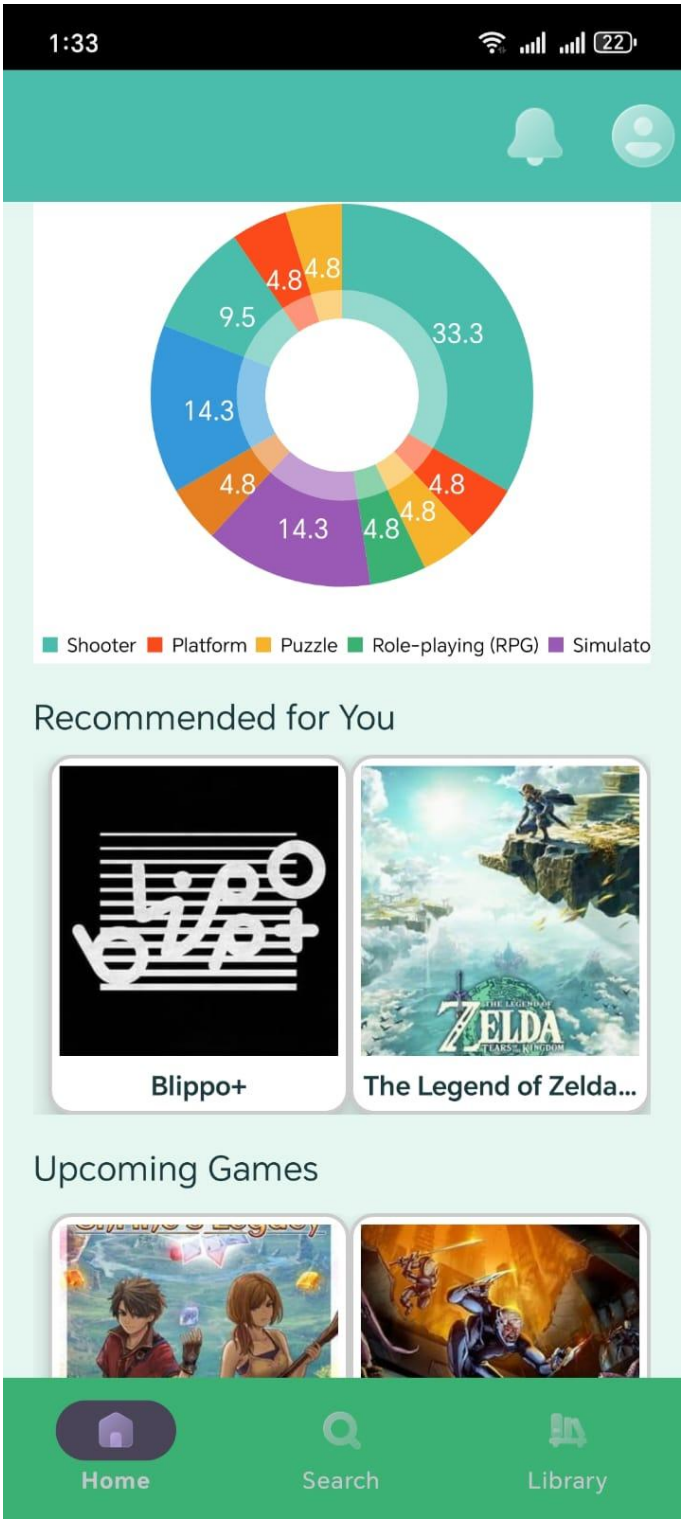


Figure 6 Home Page

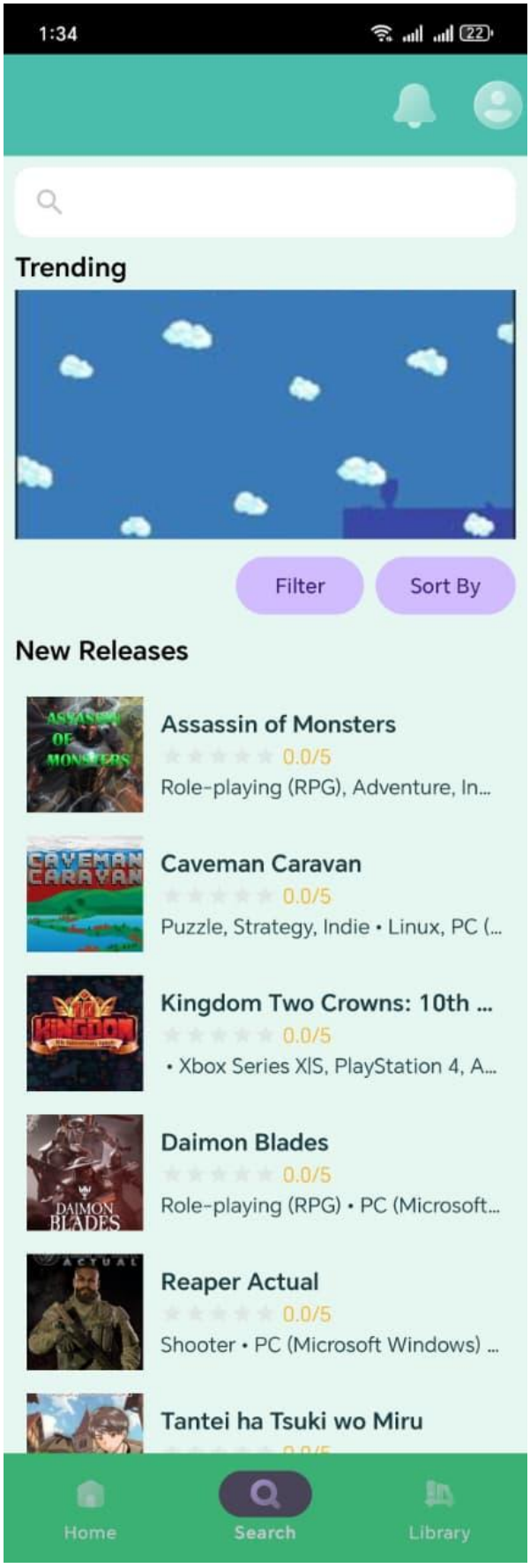


Figure 7 Search Page

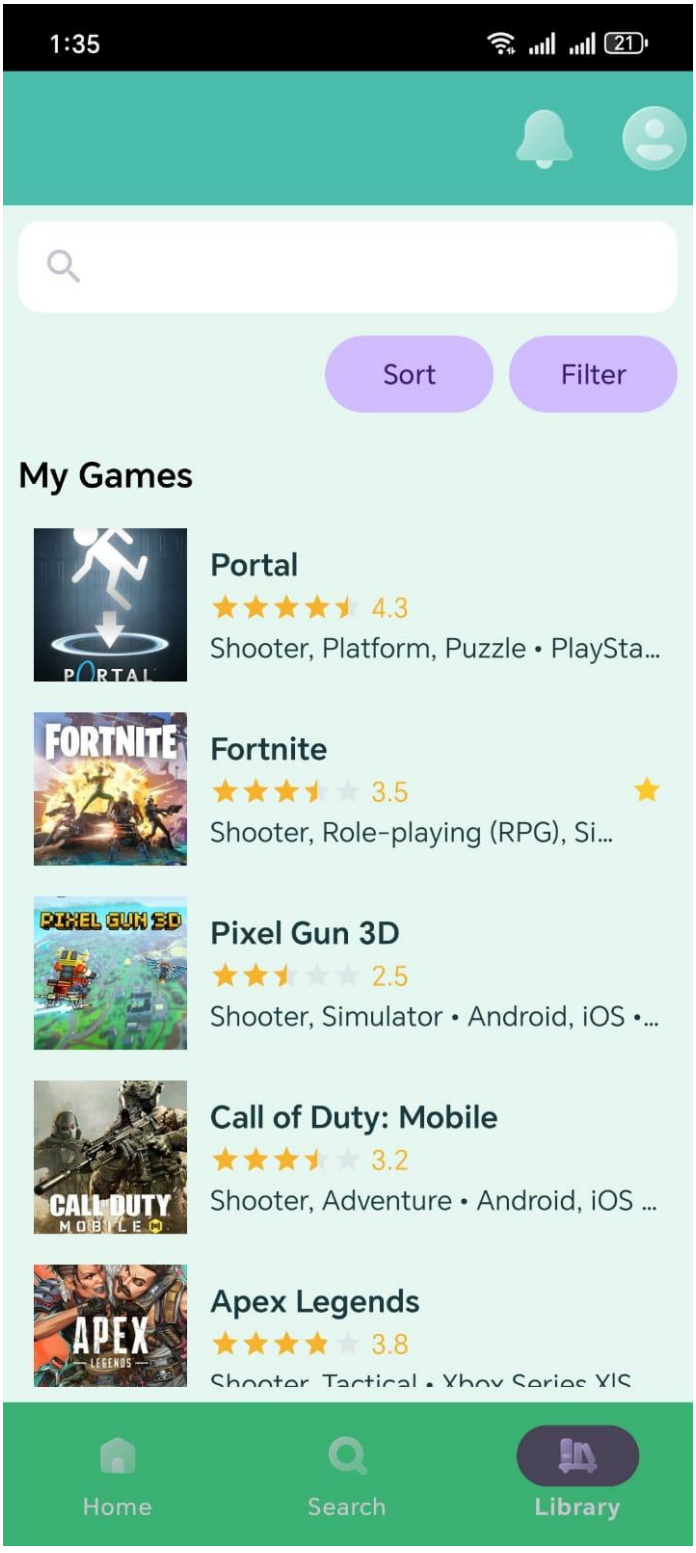


Figure 8 Library Page

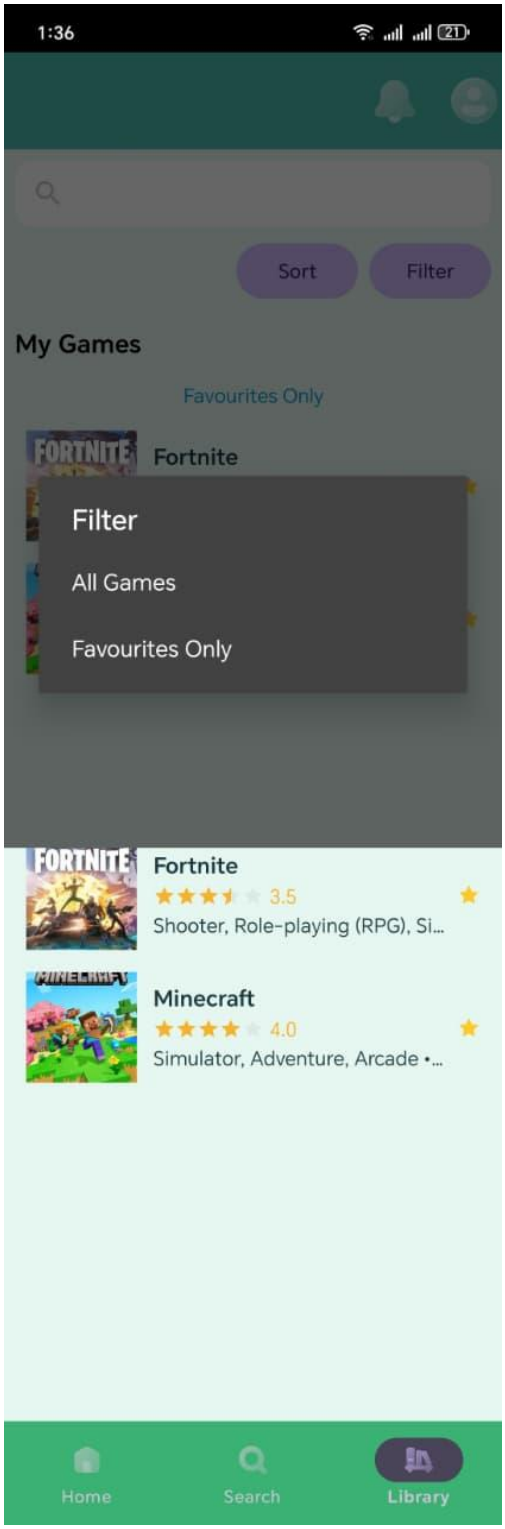


Figure 9 Filter option for Library

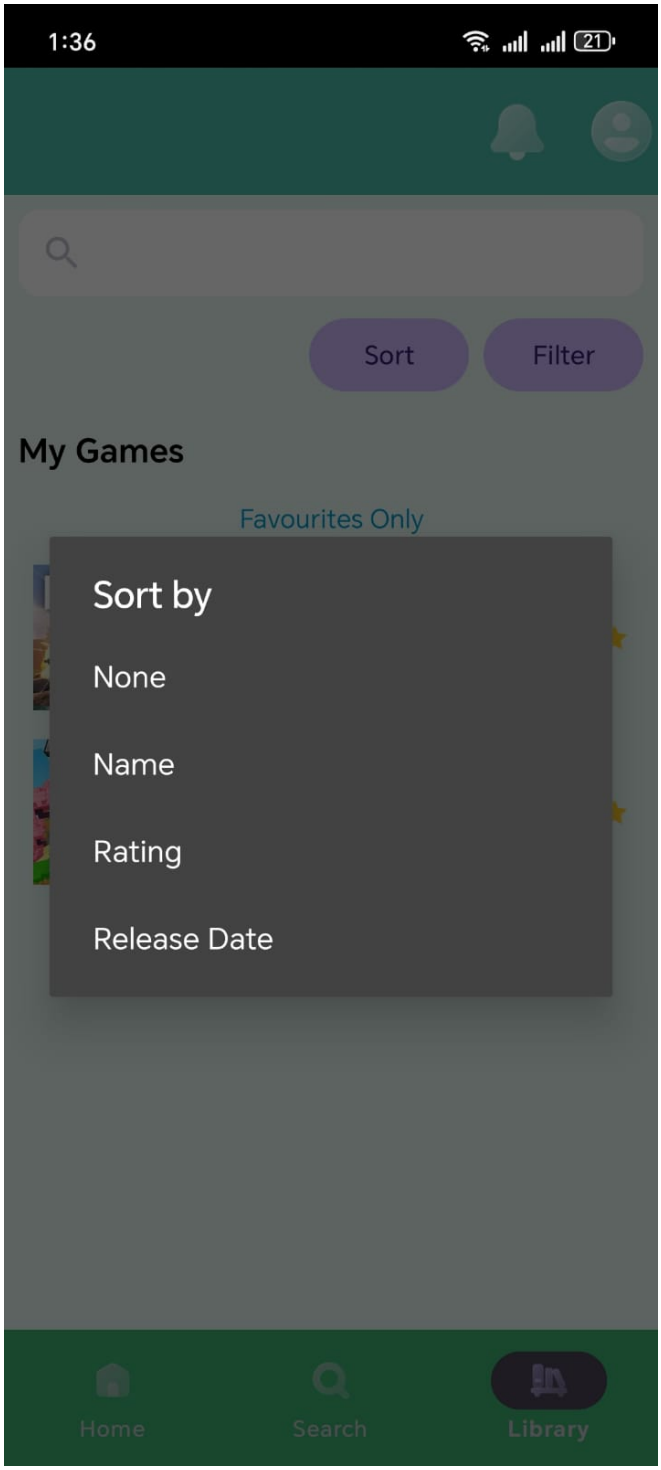


Figure 10 Sort By Function

Figure 11 Description of game chosen and screenshots of the game

Licence

This project is intended for **educational and demonstration purposes** only. Redistribution, modification, or commercial use requires written permission from the developer.

Declaration of AI Assistance

I, **Sheldon Chettiar**, **Teolan Govender, Shalin Reddy and Yadav Maganbeharie** hereby declare that we are the original author of this submission. While preparing and completing this work, we made use of **artificial intelligence (AI) tools**, specifically **OpenAI's ChatGPT (GPT-5)**, for the following **permitted purposes**:

- To assist with **clarifying programming concepts** related to Android development using Kotlin.
- To help with **structuring and formatting written documentation**, including this README file.
- To obtain **grammatical and stylistic improvements** for better clarity and professional presentation.
- To **generate general guidance** and **troubleshooting steps** related to Android Studio setup and Gradle configuration.

All AI-generated content was **reviewed, verified, and edited by our group** to ensure it aligns with the module's learning outcomes and Varsity College's academic integrity policy.
No part of this work was copied or submitted without personal understanding or proper contextual adaptation.