EntertainMeme

# Table of contents

[**1 EntertainMeme**](#_f8zs9sc3armm) **2**

[1.1 Introduction](#_fw8plfasqaus) 2

[1.2 Description](#_vvrwcdoxnbk1) 2

[1.3 Features](#_b9kwdxfd0jw0) 2

[**2 Implementation**](#_oxntzfjgog4j) **3**

[2.1 Libraries used](#_ayngdqspds0f) 3

[2.2 Design Patterns](#_cts8nwax9i0) 4

[2.4.1 Singleton](#_lbptc1gwgfvh) 4

[2.4.2 Observer Pattern](#_bx5e6ck8ae5q) 4

[2.3 Folder Structure](#_2eznndyeknjk) 5

[2.4 Naming Conventions](#_z9tnn3dow2oa) 6

[2.5 Adding Comments](#_rn89de8dro4l) 7

[2.6 Code](#_mdnkhzu6cjoc) 7

[2.6.1 Optimization](#_jp1jo5khmh2k) 8

[2.6.2 Robustness](#_jh7os9uc32oj) 8

[**3 How to use the app**](#_f2owi2z01a5q) **9**

[**4 Project Requirements and Contributions**](#_4y0rbbhbw9kt) **14**

# 1 EntertainMeme

## 1.1 Introduction

Over the past decade, the use of social media has changed repeatedly from adding friends, chatting with friends, sharing activities, sharing funny videos to starting businesses. As a result, the use of social media has been over saturated with too many use cases which have made certain user experiences of using social media not as optimum. In particular the memes entertainment area, and that’s why we feel that there is a need to change the way people enjoy memes.

## 1.2 Description

EntertainMeme is a meme swiping app that lets users browse through memes like the way Tinder works. The memes are loaded randomly to ensure that the user is always entertained. The user can swipe right to favorite the meme or swipe left to skip the meme. The favorited memes will be saved in an inventory where the user can revisit them. On top of that, the user can also download the memes and share them with their friends.

## 1.3 Features

O Swipe fresh memes

O Collect as many memes as you can

O Download memes locally

O Share memes with your friends

# 2 Implementation

## 2.1 Libraries used

|  |  |
| --- | --- |
| **API** | **Usage** |
| Random Meme Generator (herokuapp) | Used to get a random meme that was scraped from the web |
| This API is well-documented and simple to use. We have taken a look at the memes that it provides and have found them to be viable. So, we decided to use this API as our main memes source provider. |
| **Library** | **Usage** |
| Volley | Used to retrieve the memes’ metadata from the API |
| Volley promised itself as a lightweight and fast networking tool which was what we needed to retrieve the memes’ metadata. After looking through the JSON response provided by the API, the data we are retrieving is very simple hence we decided not to use anything else more complex. |
| Glide | Used to load the meme image based on a url |
| An alternative to Glide is Picasso, but we decided to settle on Glide because of its better caching capabilities. That being said, we needed to cache the images because our users might want to rewind back to the memes they have swiped. We do not want them to experience any lag so having a caching technique is a good strategy. |
| Photoview | Used to display memes in full-size picture |
| Photoview is used to display the memes in full-size in-case the user cannot see the meme clearly. This library aims to further enhance the user experience of the app. |
| Recyclerview | Used to display a list of memes |
| An alternative to Recyclerview is Listview, but we decided to settle on Recyclerview because of its better optimization capabilities. Besides, our team planned to add in more customizations for the meme card to improve User Experience thus Recyclerview was used. |
| Card-stack-view | Used to let the user swipe left or right of the stack |
| This is the main library that we used as our app is all about swiping memes. Card-stack-view allows us to create a swipeable card stack view which is exactly what we needed. |
| SQLite | Used to store the memes’ metadata locally. |
| When the user has favorited a meme, the app has to save that meme into an inventory so that the user can revisit them anytime they want. We feel that having a local database is suffice for the features we implementing |

## 2.2 Design Patterns

### 2.4.1 Singleton

We used the singleton design pattern to implement the MemeLoader.java class file as we want only one instance of the MemeLoader to be created. We decided that only one instance should be created because we do not want the MemeLoader class to be created multiple times. By ensuring that there can only be one instance of the MemeLoader to be instantiated, the entire application will be communicating with only one MemeLoader. This effectively optimizes the app as it reduces the amount of RAM used and prevents memory leak.

### 2.4.2 Observer Pattern

We used the observer design pattern to implement the MemeLoader.java class file so that it can notify its observers when a new meme has been loaded. We decided to implement the observer design pattern because we do not want our users to experience any lag by preloading the memes. Since observers are notified when new memes are loaded, they can retrieve and add the memes to the cardstack first before the users hit the end. This effectively improve the users’ experience when using our app.

## 2.3 Folder Structure

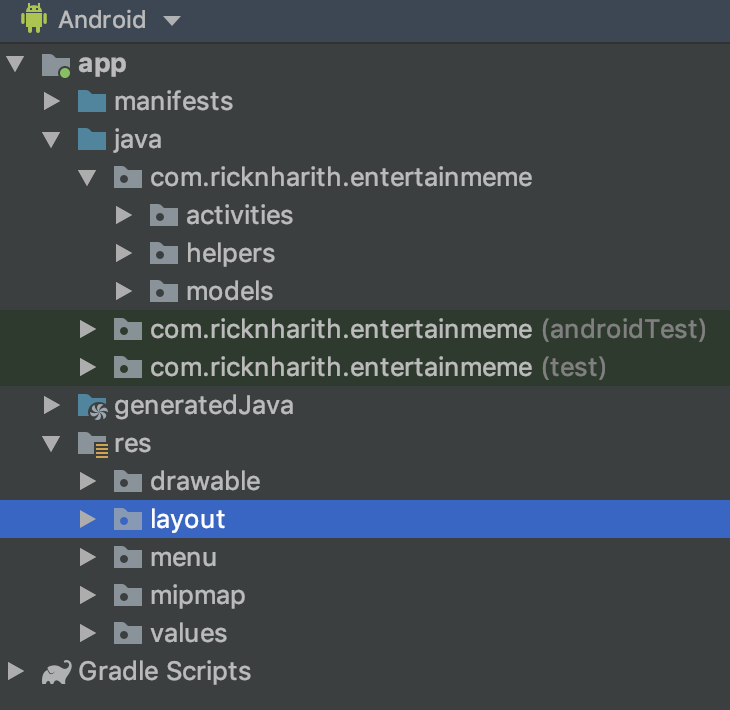


Figure shows the folder structure of our app

Before we started implementing the codes, we ensured that we named the files thoughtfully and also placed them in well-named folders to keep things organized. The three main folders that we created are ‘activities’, ‘helpers’ and ‘models’. These folders are inspired by the MVC design pattern whereas ‘activities’ and ‘helpers’ represent the controller and the data access layer of the app, and ‘models’ and ‘layout’ represent the models and the views of the app.

## 2.4 Naming Conventions

To ensure consistency, we followed a standard set of naming conventions to keep the codes neat. Below show the guidelines we have stuck to throughout the implementation process.

|  |  |
| --- | --- |
| **Object** | **Case** |
| Classes | PascalCase |
| layout | snake\_case |
| Ids of components | camelCase |
| Constructors | PascalCase |
| Functions | camelCase |
| Interfaces | camelCase |
| Variables | camelCase |
| DATABASE | UPPERCASE |

## 

## 2.5 Adding Comments

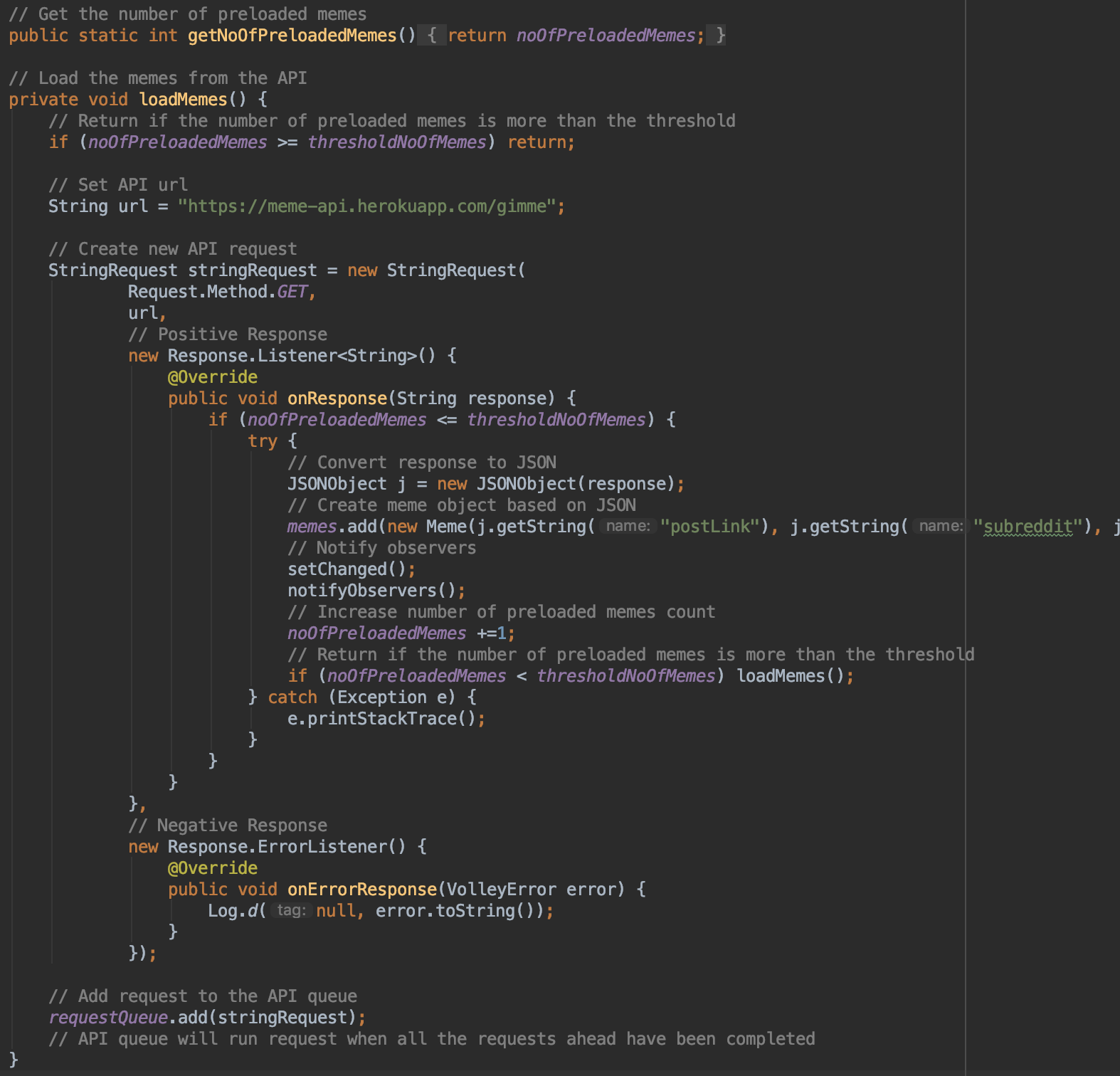


Figure shows an example of comments added to explain block of codes

Besides following a standard set of naming conventions, we added comments to each block of codes to describe what it does. By writing comments, we have helped each other to understand each other’s codes better, thus speeding up the development process.

## 2.6 Code

The source code of the app can be found in <https://github.com/rickkoh/entertainmeme>. Our team used github to facilitate the implementation of the app. Our team members were briefed to described each of the commits as descriptive as possible to keep the implementation process well-documented. All this information can be found in the ‘commits’ section of the repository.

### 

### 2.6.1 Optimization

To optimize the app, we implemented design patterns to increase the overall efficiency of using hardware resources. For instance, by using the singleton design pattern to implement the MemeLoader, only one instance of the MemeLoader can be created at a time which prevents multiple APIs requests from being sent. By doing so, the entire app communicates with only one instance of MemeLoader which reduces the amount of RAM used and prevents memory leak.

As for the observer design pattern, we used it to enable the app to load memes simultaneously, thus reducing the time the user has to wait for the memes to load, optimizing the use of internet resources.

Besides implementing design patterns, we also ensured that the redundant (inactive) activities are destroyed when the user is navigated to another page. For instance, we installed the inventory activity’s back button to run the ‘finished()’ function instead of creating a new intent so that the activity will be destroyed upon navigating back to the home page. This will effectively reduce the amount of resources used when the app is still running.

### 2.6.2 Robustness

Inside our SQLite db helper class, we implemented ‘try’ and ‘catch’ functions to prevent the app from crashing when the database runs into a problem. In addition, we have tried to break the app, but we could not. For example, we tried spamming the skip and like button to see if the memes will stop loading after awhile, but apparently it didn’t.

# 

# 3 How to use the app

|  |  |
| --- | --- |
| Home | When the user opens up the app, they can start browsing the memes immediately. |
| Home | Swipe left to skip the meme.  When the user swipes left, they can see the ‘x’ icon which indicates that they are about to skip the meme if they let go their touch. |
| Home | Swipe right to favorite the meme.  When the user swipes right, they can see the ‘x’ icon which indicates that they are about to favorite the meme if they let go their touch. |
| Home | After skipping the meme, the user can continue to browse for more memes. |
| Home | By pressing the button on the bottom left-hand side of the UI  (back button)  the previous skipped meme will reappear on the deck. |
| Inventory | When the user clicks on the inventory button located on the top right-hand corner of the Main Page, they will be directed to their inventory page.  In the user’s inventory, they will be able to revisit the memes they have favorited. |
| Drop-down menu | By clicking on the ellipsis button located at the top right-hand corner of the meme card, a drop-down menu will pop up.  Selecting the ‘Delete’ option will delete the meme.  Selecting the ‘Share’ option will open up a share sheet. |
| Photoview | When the user clicks on the meme card, they will be able to view the meme in full-screen size.  User is also able to zoom in and out in this view. |
| Sharesheet | When the user invoked the share sheet, they can choose where they want to share the image to. |

# 

# 4 Project Requirements and Contributions

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | Roughly What’s done | Status | By |
| Manifest Customization | * Title * No Action Bar * Portrait Mode only * Add Activities * Allow Internet Connection * Access Network State * Allow Write External Storage | O | Group |
| Intents | * Self-explanatory | O | Group |
| UI  (Front-end) | * General Layout (Harith) * AndroidX (Rick) * Images + Icons (Harith) * Image Buttons (Harith) * Colors (Harith) * MemeCard (Rick) * MemeSwipeCard (Rick) * AVLoadingIndicatorView (Rick) * MemeCard Overlay Effect (Rick) * Inventory (Harith) * Ellipsis Menu (Harith) * Share Sheet (Harith) | O | Group |
| RecyclerView | * Adapter * Viewholder * Listview * OnClickListener (Group) | O | Rick |
| SQLite | * Create table * Update table * Select meme * Insert meme * Update meme * Delete meme | O | Harith |
| JSON | * Read JSON Response * Convert JSON Response to Meme Object | O | Rick |
| Volley | * Create API Request * Send API Request * Get JSON Response | O | Rick |
| Glide | * Retrieve image from url * Load image to components * Load image to objects | O | Harith |
| Photoview | * Display meme in full size * Share Button * Share Sheet * Share/Save functions | O | Harith |
| Card-stack-view | * Cardstackview * Swipe Cards * Logic for loading the cards - Algorithms and Flow | O | Rick |
| Design Patterns | * Singleton design pattern * Observer design pattern | O | Rick |