FlickVibes – Design Proposal

GitHub Username : nirmaljeffrey

Index

Description

Intended User

Features

User Interface Mocks

Screen 1

Screen 2

Screen 3

Screen 4 and 5

Widget

Key Considerations

How will your app handle data persistence?

Describe any corner cases in the UX.

Describe any libraries you'll be using and share your reasoning for including them.

Describe how you will implement Google Play Services.

Required Tasks

Task 1: Project Setup

Task 2: Data model classes

Task 3: Network

Task 4: Android Architecture

Task 5: Database

Task 6: User Interface

Task 7: UI testing

Task 8: Implement Google Play Services and Microsoft Cognitive Emotion API

Task 8: Make the app production ready

Task 10: Prepare for app release

GitHub Username : nirmaljeffrey

Application Name: FlickVibes

Programming Language:Java

Description

FlickVibes is an android application that suggests movies to users based on their feelings. Most of the people tend to watch movies when they are feeling bored, depressed or a multitude of other feelings. This app uses emotion recognition APIs to detect the user's feelings. Users can also view most-popular, top-rated and upcoming movies. This app allows users to view trailers, ratings, reviews and cast of a selected movie.

The research papers and articles which inspired me to create FlickVibes is listed below,

1. https://www.sciencedirect.com/science/article/pii/S1877042814024331
2. https://www.webmd.com/mental-health/features/movie-therapy-using-movies-for-mental-health#2

Intended User

FlickVibes is aimed at people who love watching movies.

Features

- This app uses emotion recognition APIs to detect user’s feelings.
- This app suggests movies based on the user's feelings.
- Browse the most popular movie list.
- Browse the new releases movie list.
- Browse the top rated movie list.
- Bookmark movies to create your own collection of favourite movies.
- Search for movies, genre, etc.
- Widgets to display info about movies in home screen.

User Interface Mocks

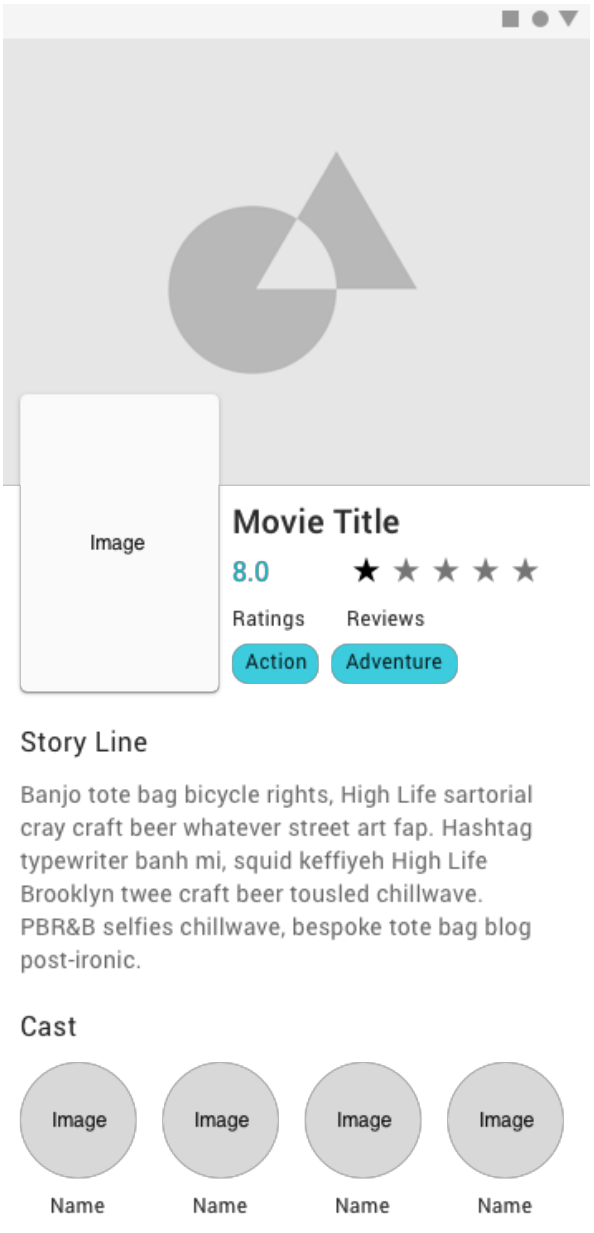
Screen 1

HOMEPAGE

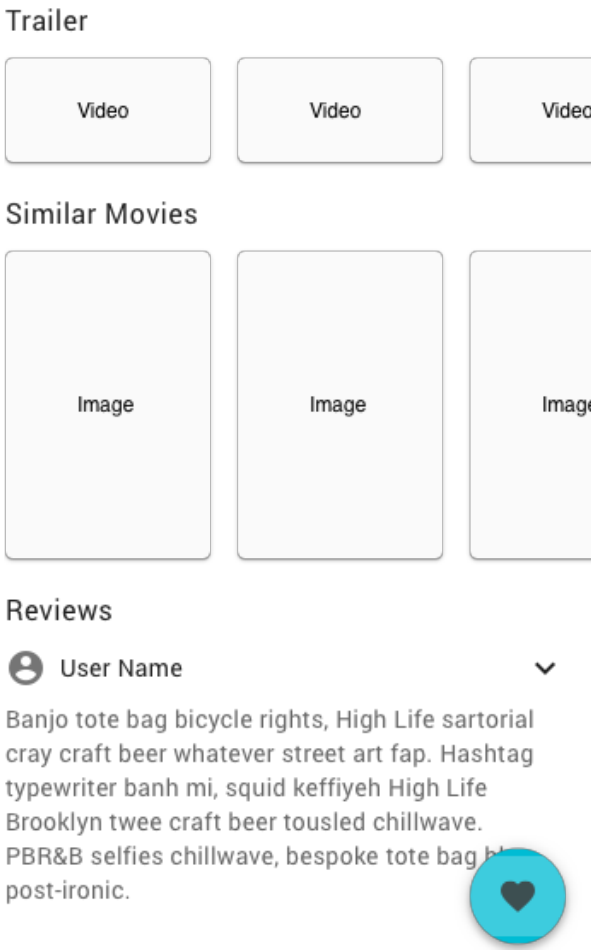
Homepage comes with a search option, a floating action button to trigger camera intent or image picker intent and displays movies in four categories, namely

- Most popular movies
- Top rated movies
- Newly released Movies
- Favourite movies.

Screen 2 (This screen is shown in two halves)



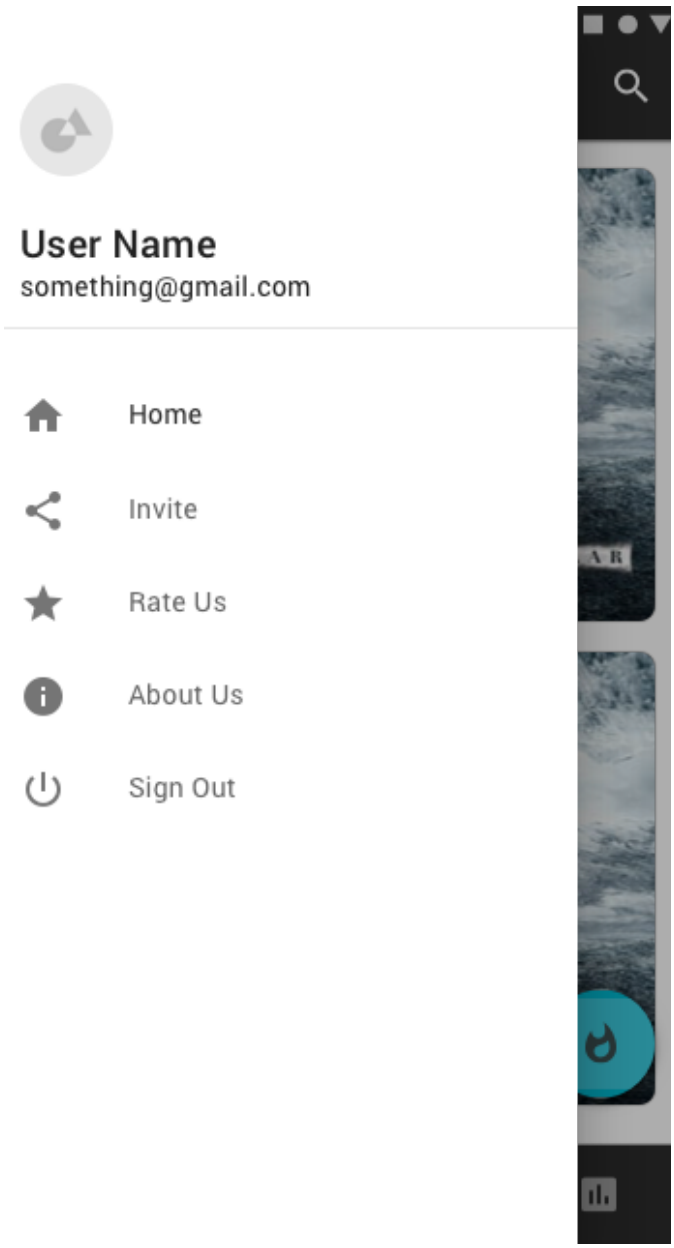
MOVIE DETAIL PAGE - PART 1



MOVIE DETAIL PAGE - PART 2

Movie Detail page displays title, ratings, genre, reviews, cast, storyline, trailer and a floating action button to bookmark the movie.

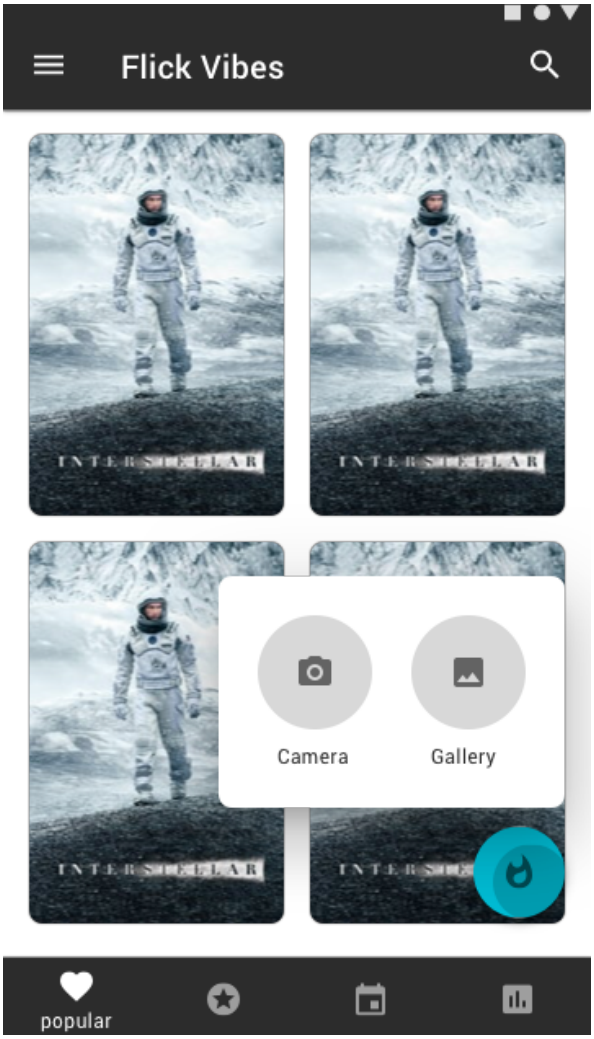
Screen 3



NAVIGATION DRAWER

Navigation Drawer displays available app features.

Screen 4 and 5

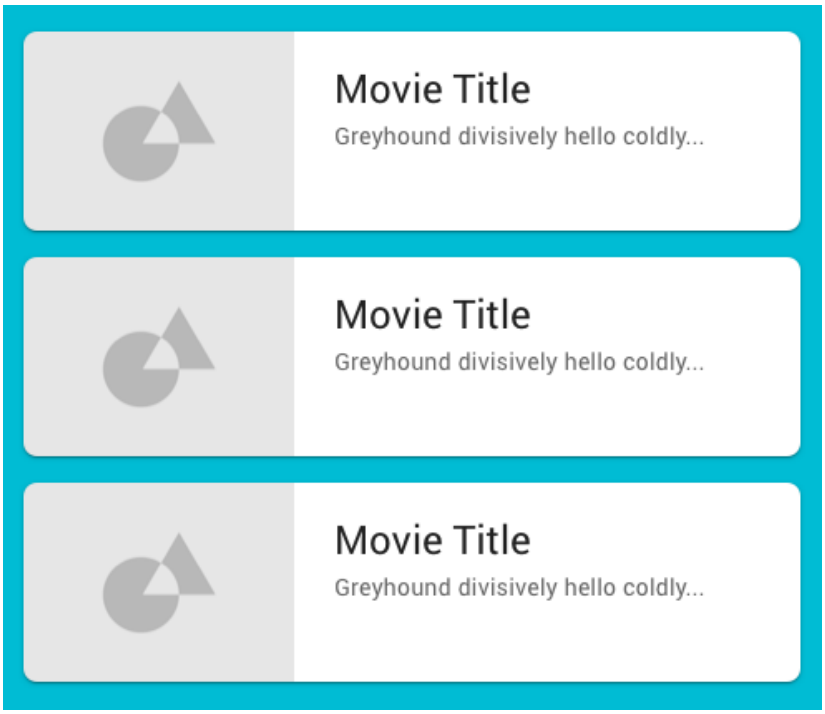


POPUP DIALOG

PROCESSED IMAGE

When floating action button in the homepage is clicked, a popup dialog asks the user to select camera intent or image picker intent. The image with user's face from the intent will be processed by the emotion recognition API to detect the feelings of the user. Based on the feelings of the user, the application retrieves movies list from movies API.

Widget



WIDGET

App widget displays the user's favourite movies list. When the user clicks on a list item, the application will open the movie detail page with the help of Intent service.

Key Considerations

How will your app handle data persistence?

FlickVibes will use room database to maintain local data.

Describe any edge or corner cases in the UX.

- **Unstable or missed network connection:** application never crashes in these situations.
- **Device Orientation Change:** application will handle long running operations correctly considering possible configuration changes.
- Application will never use UI thread (main thread) for long running operations.
- When the movie list in the favourite section is empty, the app will show an image and contains a text explaining how to populate the list.
- When more than one face or no face detected by emotion recognition API, the app will instruct the user to capture an image having a single face in it.

Describe any libraries you'll be using and share your reasoning for including them.

The project will utilize only the stable release versions of the libraries mentioned below,

- **Retrofit:** for network requests.
- **Glide:** for image loading.
- **Firebase:** for analytics, authentication, crash reports.
- **Dagger:** for dependency injection.
- **Stetho:** for debugging.
- **Leak Canary:** for memory leaks detection.

Describe how you will implement Google Play Services or other external services.

The application will use,

- Firebase for authentication, analytics and crash reporting.
- Microsoft Cognitive Emotion API for facial expression detection.
- The Movies Database API for request data related to movies.

Required Tasks

Task 1: Project Setup

- Create and setup new project in android studio.
- Configure libraries.
- Configure tools for debugging and memory leak detection.

Task 2: Data model classes

- Create data model classes to handle the responses received from movies database API.

Task 3: Network

- Implement a retrofit service which provides all necessary network API requests.
- App updates data in its cache at regular intervals using a SyncAdapter or JobDispacter.

Task 4: Android Architecture

- Implement MVVM architecture with a repository class.

Task 5: Database

- Using room persistent library and liveData, create a database to handle to all locally stored data.

Task 6: User Interface

- Using activities and fragments, create user interfaces as per the design guidelines.
- Implement the Widget.

Task 7: UI testing

- Write espresso testing for automated UI testing.

Task 8: Implement Google Play Services and Microsoft Cognitive Emotion API

- Implement Firebase for authentication, analytics and crash reporting.
- Implement Microsoft Emotion API to detect facial expressions of the user from the image.

Task 9:Make the app production ready

- Implement animations and improve the app visuals based on the material design guidelines.
- Provide RTL support.
- Provide Accessibility support.
- Provide content descriptions for UI elements.
- Move all the strings into strings.xml for localisation.

Task 10: Prepare for app release

- Remove the debug message.
- Create app icon.
- Create and sign the APK.
- Push the APK to the google play store.