Design Document

Custom Application: MY FAVORITE COCKTAILS

Prashanta Halder

Android Capstone Project

ABOUT APPLICATION

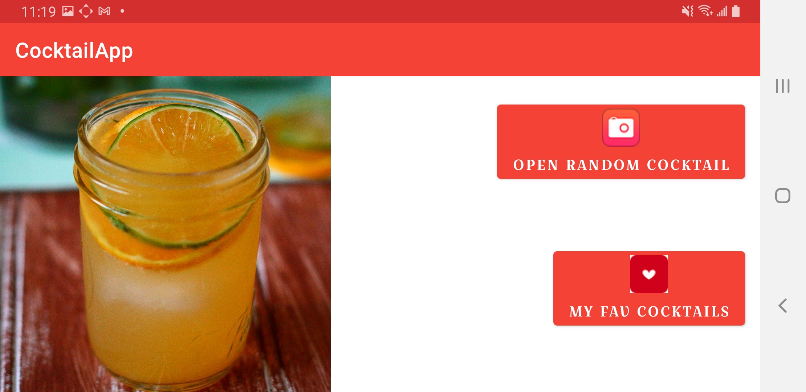
This application provides a user to view Cocktails and add any cocktail as favorite. The application can randomly get details about a particular cocktail from the internet and provides the user ability to save the cocktail as favorite. Once a cocktail is saved user can view the list of favorite cocktails and will have the ability to remove it from favorite list. Apart from this use will have the option to update the Picture of the favorite cocktails. The application uses APIs provided by Cocktail DB API (<https://www.thecocktaildb.com/api.php>)

Use Cases

* User can fetch a random cocktail (every time the application starts)
* User can view the details (name, category, ingredients & measures, instruction, image) of the random cocktail
* User can mark/unmark the cocktail from Favorite List of cocktails
* User can view the list of Favorite Cocktail (in offline mode)
* User can update the picture of the cocktail

Application Screens

1. Launch Screen/Home Screen



This would the first screen when user launches the application. The Home Screen will display the image of a random cocktail and provide user:

1. To view the details of the random cocktail
2. View all the saved favorite cocktails

\*\* More options can be added here when additional features are added (like search a cocktail by name or ingredient etc.)

This screen will adapt/change its layout when rotated.

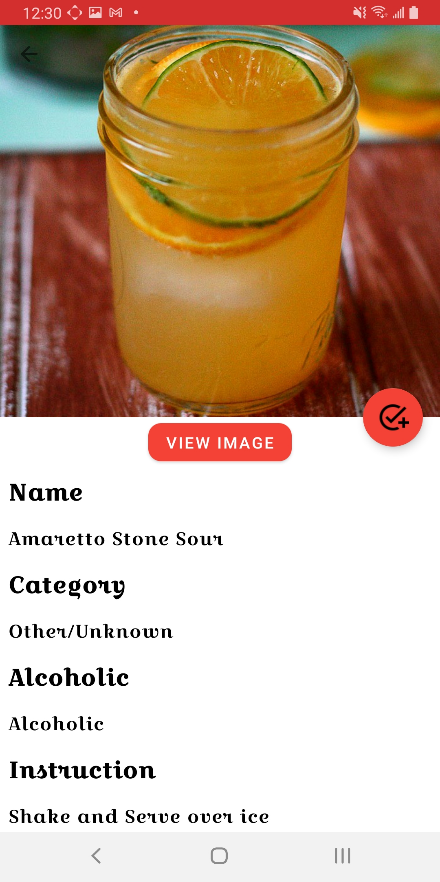
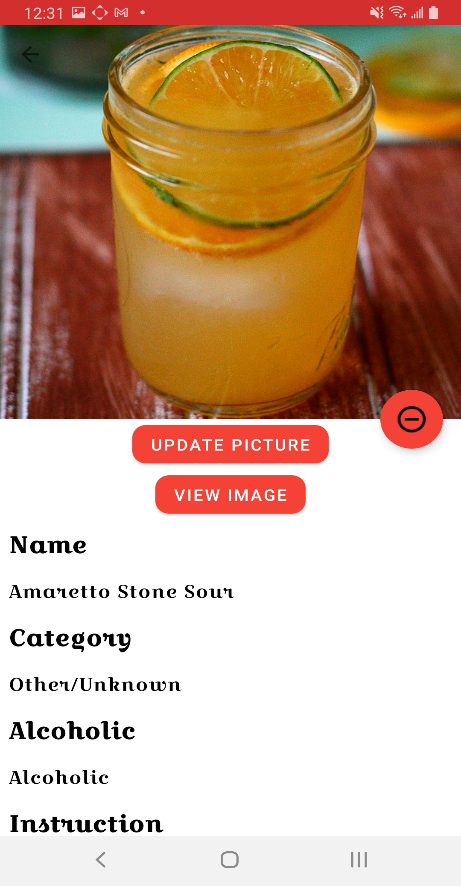
1. Cocktail Detail Screen

This screen displays the details of a particular cocktail. The detail will include the Image, Name, Category, Alcoholic Type, Instructions & Ingredients. The screen will be based on Coordinator Layout and will show a dynamics animation when scrolled. This screen works with both orientation modes.

In this screen user can mark the cocktail as favorite. When marked as favorite the application will save this cocktail in favorite local database.

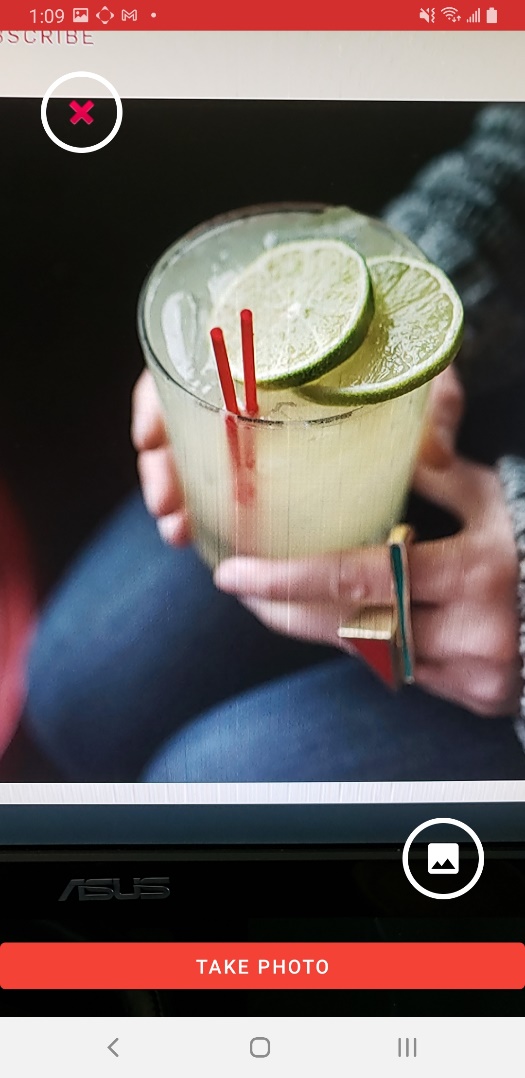
User would have the ability to view full screen image of the cocktail (by using “View Image” button).

User would have the ability to change the image of the cocktail (by using “Update Picture” button)

1. Image Capture Screen

In this screen user will be able to capture a new image of the cocktail using Device Camera and will also have the ability to preview the image. Once the picture is taken and preview user can save the new image for the cocktail. This feature will be available only for cocktails which are saved as favorites.



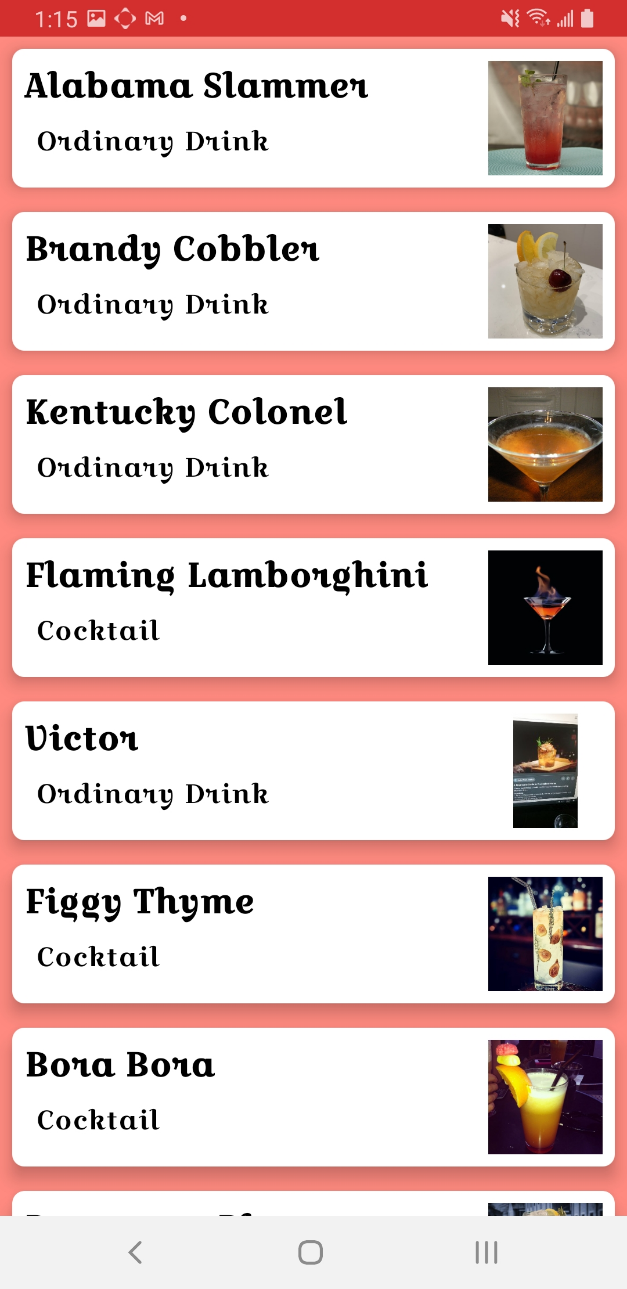
1. Image Preview Screen

In this screen user will be able view the full screen image of a cocktail. This screen can be launched from Cocktail Detail Screen or Camera Capture Screen



1. Favorite Cocktails Screen

In this screen user will be able view the list of all the cocktails which use has marked as favorite. Each item of the List will display the Coctail Name, Type and Thumbnail of the cocktail. Selecting or Clicking an item will display the Cocktail Detail Screen.



Application Architecture



The application follows MVVM Architecture pattern. The UI Layer is mainly composed of Fragment and Activities which Interacts with ViewModel Layer. The ViewModel objects provides data for each Specific UI Components and handles the business logic to communicate with underlying data sources (Models)

Repository modules handles at the data operations and abstracts the way data is retrieved and stored. The Repository layer provides an mediator role between different data source and their synchronization.

Project Rubric Requirements

**Android UI/UX**

|  |  |
| --- | --- |
| CRITERIA | MEETS SPECIFICATIONS |
| Build a navigable interface consisting of multiple screens of functionality and data. | Application includes at least three screens with distinct features using either the Android Navigation Controller or Explicit Intents.   * The application includes 4 Screen with distinct features * 1) HomeFragment – Which displays Random cocktail image and menu buttons * 2) SavedCocktailFragment– Which displays list of favorite cocktail items * 3) CocktailDetailFragment – Which shows the details of a specific cocktail item * 4) CameraCapture Activity – Which uses device camera to capture image that can be set to a cocktail item   The Navigation Controller is used for Fragment-based navigation and intents are utilized for Activity-based navigation.   * Navigation Component and Navigation Graph is used for Navigation between HomeFragment, SavedCocktailFragment and CocktailDetailFragment * Intents are used for Navigation to CameraCapture activity and PhotoViewActivity   An application bundle is built to store data passed between Fragments and Activities.   * Bundles using Safeargs are used to transfer data from SavedCocktailFragment to CocktailDetailFragment and from HomeFragment to CocktailDetailFragment * Bundles are used for Navigation to CameraCapture activity and PhotoViewActivity to transfer data. |
| Construct interfaces that adhere to Android standards and display appropriately on screens of different size and resolution. | Application UI effectively utilizes ConstraintLayout to arrange UI elements effectively and efficiently across application features, avoiding nesting layouts and maintaining a flat UI structure where possible.   * Layouts have been created using ConstraintLayout and CoordinatorLayout. They follow flat Layout heirarchy   Data collections are displayed effectively, taking advantage of visual hierarchy and arrangement to display data in an easily consumable format.   * Data Collections are displayed using RecyclerView and UI element are binded to View Model properties.   Resources are stored appropriately using the internal res directory to store data in appropriate locations including string\* values, drawables, colors, dimensions, and more.   * All Resources have been provided in color, string, drawables, dimensions , styles and themes in order to increase maintainability and resusability   Every element within ConstraintLayout should include the id field and at least 1 vertical constraint.   * All elements inside constraint layout has ids and the UI elements have been binded to ViewModel Properties. In cocktail\_list\_item.xml the “Description” TextView is vertically constraint to “Title” TextView   Data collections should be loaded into the application using ViewHolder pattern and appropriate View, such as RecyclerView.   * Data Collections are displayed using RecyclerView and UI element are binded to View Model properties. It uses the ViewHolder pattern. Used DataBindingViewHolder class to achieve this |
| Animate UI components to better utilize screen real estate and create engaging content. | Application contains at least 1 feature utilizing MotionLayout to adapt UI elements to a given function. This could include animating control elements onto and off screen, displaying and hiding a form, or animation of complex UI transitions.   * MotionLayout has been used in CameraCapture Activity. Here the back button is animated to become the Save button after an image is captured   MotionLayout behaviors are defined in a MotionScene using one or more Transition nodes and ConstraintSet blocks.   * MotionLayout has been defined in activity\_camera\_capture\_scene.xml and the trigger of the motion layout has been done in code   Constraints are defined within the scenes and house all layout params for the animation.   * The activity\_camera\_capture\_scene.xml contains the constraints for start and end |

**Local and Network data**

|  |  |
| --- | --- |
| CRITERIA | MEETS SPECIFICATIONS |
| Connect to and consume data from a remote data source such as a RESTful API. | The Application connects to at least 1 external data source using Retrofit or other appropriate library/component and retrieves data for use within the application.   * The application connects to Cocktail DB APIs using Retrofit and makes network data calls.   Data retrieved from the remote source is held in local models with appropriate data types that are readily handled and manipulated within the application source. Helper libraries such as Moshi may be used to assist with this requirement.   * The JSON responses coming back from Cocktail DB APIs are deserialized using Moshi to DTO and then transformation maps are used to convert the DTO to Application Model   The application performs work and handles network requests on the appropriate threads to avoid stalling the UI.   * The network request are made using Kotlin co-routines and uses IO thread to make network requests |
| Load network resources, such as Bitmap Images, dynamically and on-demand. | The Application loads remote resources asynchronously using an appropriate library such as Glide or other library/component when needed.   * The application uses Glide to load remote bitmap or local bitmap asynchronously and on-demand   Images display placeholder images while being loaded and handle failed network requests gracefully.   * Image display Placeholders has been provided and error images has been provided in BindingAdapter.kt   All requests are performed asynchronously and handled on the appropriate threads. |
| Store data locally on the device for use between application sessions and/or offline use. | The application utilizes storage mechanisms that best fit the data stored to store data locally on the device. Example: SharedPreferences for user settings or an internal database for data persistence for application data. Libraries such as Room may be utilized to achieve this functionality.   * The application uses ROOM database to store favorite cocktails   Data stored is accessible across user sessions.   * Since data is store in Local Database , it is available across all user sessions   Data storage operations are performed on the appropriate threads as to not stall the UI thread.   * Local Database operations are done in Kotlin Coroutines and performed on Non UI threads   Data is structured with appropriate data types and scope as required by application functionality.   * Data Types are chosen to support the corresponding UI functionality. Some data goes through transformation before being displayed to user |

**Android system and hardware integration**

|  |  |
| --- | --- |
| CRITERIA | MEETS SPECIFICATIONS |
| Architect application functionality using MVVM. | Application separates responsibilities amongst classes and structures using the MVVM Pattern:   * *Fragments*/*Activities* control the *Views* * Models houses the data structures, * *ViewModel* controls business logic.   Application adheres to architecture best practices, such as the observer pattern, to prevent leaking components, such as Activity Contexts, and efficiently utilize system resources.   * The application uses MVVM pattern. Details class diagram is provided earlier. |
| Implement logic to handle and respond to hardware and system events that impact the Android Lifecycle. | Beyond MVVM, the application handles system events, such as orientation changes, application switching, notifications, and similar events gracefully including, but not limited to:   * Storing and restoring state and information * Properly handling lifecycle events in regards to behavior and functionality * Implement bundles to restore and save data * Handling interaction to and from the application via *Intents* * Handling Android Permissions * The application uses bundles to save & restore data during orientation change. This can be seen in Camera Capture activity * Permissions are being requested for using device camera. |
| Utilize system hardware to provide the user with advanced functionality and features. | Application utilizes at least 1 hardware component to provide meaningful functionality to the application as a whole. Suggestion options include:   * Camera * Location * Accelerometer * Microphone * Gesture Capture * Notifications   Permissions to access hardware features are requested at the time of use for the feature.  Behaviors are accessed only after permissions are granted.   * The application uses device camera hardware to capture images of a cocktail. This is done in CameraCapture Activity. * Before the capture permission is being asked from the user and only the camera is used when user grants the permission |