**ACKNOWLEDGEMENTS**

We would like to express our special thanks to our supervisor, mentor and Android instructor, Mr Polite Nwosu, who worked actively to provide us with the protected academic time to pursue the goals outlined in this project and his able guidance and support with completing this eProject.

We are also grateful to all of those with whom we have had the pleasure to work during this and other related projects. Each of the members of our project team provided extensive personal and professional guidance and taught each other a great deal about Android Development.

We would also like to extend our gratitude to Aptech Ajah for their help in acquiring all the resources needed for the project.

**SYNOPSIS**

This project, Soft Drink Recipes, is an Android application that displays all the recipes under certain categories. It allows you to:

* view the categories of drink recipes that can be made.
* view the drinks under a category.
* view the steps involved in creating the drinks specified in the app.
* share the drink recipes across all your social networks, email etc.

**USER GUIDE**

The application starts off with a navigation menu on the left with the following menu items:

* Home : This is where you can view the popular drinks and popular categories of drinks on the application. Upon selecting a drink or category, you will be taken to the page of the drink to view its recipes or the page of the category to view its drinks.
* Categories: This is where you can view all the different categories of the drinks present on the application. Selecting a category in the list will take you to the page where you can view the drinks under that category.
* My Favorites: This is where you can view all your selected favorite drinks. Each drink page contains a “Add to Favorites” button. Upon clicking this button, the drink will be added to the favorites of the user. All favorites will appear on this My Favorites page.

Each page also contains the search button at the bottom of the application that launches the search dialog and enables search across categories or drink recipes.

**DATABASE STRUCTURE**

**DRINKS TABLE**

*DRINKS\_COL\_ID* + " INTEGER PRIMARY KEY AUTOINCREMENT," +  
*DRINKS\_COL\_NAME* + " VARCHAR(200) UNIQUE," +  
*DRINKS\_COL\_CATEGORY* + " INTEGER," +  
*DRINKS\_COL\_FAV\_COUNT* + " INTEGER," +  
*DRINKS\_COL\_DETAILS* + " TEXT," +  
*DRINKS\_COL\_IMAGE\_URI* + " VARCHAR(200)," +  
*DRINKS\_COL\_RECIPE* + " TEXT," +  
"FOREIGN KEY(" + *DRINKS\_COL\_CATEGORY* + ") REFERENCES " + *CATEGORIES\_TABLE\_NAME* + "(" + *CATEGORIES\_COL\_ID* + ")"

**CATEGORIES TABLE**

*CATEGORIES\_COL\_ID* + " INTEGER PRIMARY KEY AUTOINCREMENT," +  
*CATEGORIES\_COL\_NAME* + " VARCHAR(200) UNIQUE," +  
*CATEGORIES\_COL\_FAV\_COUNT* + " INTEGER DEFAULT 0," +  
*CATEGORIES\_COL\_IMAGE\_URI* + " VARCHAR(200)," +  
*CATEGORIES\_COL\_DETAILS* + " TEXT" +

**FAVORITES TABLE**

*FAVORITES\_COL\_ID* + " INTEGER PRIMARY KEY AUTOINCREMENT," +  
*FAVORITES\_COL\_DRINK\_ID* + " INTEGER UNIQUE," +  
"FOREIGN KEY(" + *FAVORITES\_COL\_DRINK\_ID* + ") REFERENCES " + *DRINKS\_TABLE\_NAME* + "(" + *DRINKS\_COL\_ID* + ")" +

**DEVELOPER’S GUIDE**

The “main” module of the app holds the source code for the application. It is broken down into the assets, java and res folders.  
  
**ASSETS FOLDER**

The assets folder contains the JSON files that contain the content that are used in the app. There are the “categories.json” and the “drinks.json” files.

* The categories.json file contains the details of all the categories in the app. This file is parsed with a JSON parser class and inserted into the “CATEGORIES” table in the SQLiteDatabase of the application.
* The “drinks.json” file contains the details of the drinks in the app. |This file is parsed with the JSON parser class and inserted into the “DRINKS” table in the database of the app.

**JAVA FOLDER**

The java folder contains the main logic of the application. An MVC model is used with this application and hence the folders are split into the adapters, controllers, models, ui and util folders.

* Adapter folder contains the adapters that are used with all the recycler views in the app. They include:
  + the CategoryAdapter - that is used to display the Categories list used in the Home Page of the app to display the popular categories list. It is also used in the Categories Page that contains a list of all the categories of drinks in the application.
  + The DrinkAdapter – that is used to display the drinks list used in the HomePage of the application for the Popular Drinks portion of the page. It is also used for the Similar Drinks portion of the Drinks page and the Drinks in this Category portion of the Categories page.
  + The RecipeAdapter – that is used to display the list of steps for each drink recipe.
  + The SearchAdapter – that is used to display the list of search results on the search results page when a user makes a search
* Controller folder contains the logic to interact the database to retrieve data for each page, either drink, category or search. It includes just one controller class:
  + DrinkController – Used to query the database for all the data pertaining to drinks and categories. It is also responsible for search queries. It includes the following methods:
    - seedDB – this method is responsible for seeding the database with all the data from the JSON files in the assets folder. This method is only called once when the application is installed to prevent duplication.
    - DrinkController constructor – this method instantiates a new DrinkController object in the calling method using the context of the calling fragment or activity. The returned object can be used to query the database with its methods.
    - getDrink – this method takes the target ID (the primary key) of the drink in the database and returns a DrinkModel object with the data from the column in the database where the ID is present.
    - getCategory - this method takes the target ID (the primary key) of the category in the database and returns a CategoryModel object with the data from the column in the database where the ID is present.
    - insertDrink – this method is used to insert drink recipe details for a single drink into the database.
    - insertCategory – this method is used to insert category details for a single category into the database.
    - insertFav – this method is used to insert a drink into the FAVORITES table in the database. This table represents the favorites of the user.
    - toJSONString – this method is responsible for converting the provided JSON file to a parse-able JSON string.
    - insertSeedCategories – this method is used within the seedDB method to insert seed categories into the database.
    - insertSeedDrinks – this method is used within the seedDB method to insert seed drinks into the database.
    - getPopularDrinks – this method is used to get the most popular drinks from the database. This leverages the favCount column in the database that is incremented based on the amount of clicks a drink gets. This method returns the drinks with the highest favCount in descending order.
    - getPopularCategories – this method is the equivalent of getPopularDrinks for categories.
    - getAllDrinks – this method is used to pull all the drinks from the database and return the data in an arraylist of type DrinkModel.
    - getAllCategories – this method is the equivalent of getAllDrinks for categories.
    - searchProducts – this method takes a search query string and searches the database for products that have a category or drink that matches the string. Any matching rows in the database are compiled into an list of DrinkModels.
    - getDrinksbyCategory – this method takes an id of a category and gets all the drinks that match that category in the database in the form of a list of DrinkModels.
    - removeFav – this method removes a drink from the FAVORITES table in the database.
    - insertFav – this method adds a drink into the FAVORITES table in the database.
    - isFav – this method checks if a drink is in the FAVORITES table in the database. Returns a Boolean true or false.
* The models folder contains the models for the different types of data that are pulled from the database. These models contain the different fields in formats that are usable across different views in the application. They include:
  + The CategoryModel – this method contains the fields:
    - name - to represent the name of the category.
    - dbID - the ID of the category in the database.
    - categoryDetails – to contain the details of the category.
    - imageURI – to contain the name of the image file for the category in the drawable folder of the application.
  + The DrinkModel – this method contains the fields:
    - Name – to represent the name of the drink.
    - ID – the ID of the drink in the database.
    - drinkDetails – the text details of the drink in the database.
    - categoryID – the ID of the category the drink belongs to in the database.
    - Category – the name of the category that the drink belongs to in the database.
    - drinkImageURI – the name of the image file for the drink in the drawable folder of the database.
    - drinkRecipe – a string that contains the recipes of the drink delimited with ‘;’.
    - The DrinkModel also contains different getters and setters for the fields that it contains as well as a constructor for creating a DrinkModel object with all these fields.
  + The SearchModel – this contains the fields that are important for the search results view. The fields include:
    - The name of the search result – a drink or a category.
    - The type of search result – either a category or a drink.
    - The ID of the drink or category in the database.
    - The imageURI for the search result, either drink or category.
    - The fields also have their getters and setters in the drinkModel.
* The ui folder contains the different views that the application uses. These include:
  + The CategoriesFragment – This fragment replaces the current fragment on the MainActivity. The view generated will contain the Categories textview and the recyclerview that lists all the categories. The recyclerview for all the categories makes use of a CategoryAdapter to fill the category\_sample view of each CategoryModel generated from the database by the DrinkController object’s getAllCategories method.
  + The FavoritesFragment – This fragment replaces the current fragment on the MainActivity. The view generated will contain the Favorites textview and the recyclerView that lists all the current favorites of the user pulled from the database using the DrinkController’s getFavorites method. The Fragment also checks if the user has no favorites and replaces the recyclerview with a textview that indicates “You have no favorites”. The onViewStateRestored and onResume methods are also used to update the Fragment upon resumption from the backstack, based on favorites activities on other pages.
  + The HomeFragment – This fragment is the default fragment on the Main Activity. It contains the popularCategories recyclerview (using the CategoryAdapter) and the popularDrinks recyclerview (using the DrinkAdapter). These views have their content pulled from the database using the DrinkController’s getPopularCategories and getPopularDrinks methods respectively.
  + The SearchActivity – This view is responsible for processing searchQueries from other activities and fragments across the application and displaying the results in a view. The doMySearch method in this activity takes the query from the search dialog or searchView from other fragments or activities and calls the DrinkController’s searchProducts method passing the query as a parameter. The search results are then used with the SearchAdapter to display it in the recyclerview for the SearchActivity. It also checks if there are no searchResults and updates the view accordingly with a “No search results” textview.
  + The SingleCategoryFragment – this fragment is responsible for displaying the view for a single category from the database. It uses the CategoryModel object obtained via the DrinkController’s getCategory method, to populate the different views across the page. This includes the ImageView for the image of the category, the recyclerview that displays the drinks that belong to the category, and the categoryDetails textview that displays the details of the category.
  + The SingleDrinkFragment – this fragment contains the logic that creates the view for a single drink. It uses a DrinkModel object to populate the different views across the page. This includes the following:
    - The TextView drinkTitle with the name of the drink.
    - The ImageView drinkImage obtained using the ImageURI from the DrinkModel object.
    - The drinkDetails textview with the details of the drink’s details.
    - The drinkRecipeView recyclerview that contains the recipes of the drink obtained from the DrinkModel. The string obtained from the database is split using the regex “;” into a String array used by the RecipeAdapter.
    - The similarDrinksView recyclerview contains the drinks in the category of the current drink excluding the current drink. These drinks are obtained using the DrinkController’s getDrinksByCategory method and are populated into the similarDrinksView using the DrinkAdapter.

The addFav button is responsible for adding or removing the drink to the user’s favorites. An onClickListener on the button checks the current text of the button and inserts or removes from the database accordingly. Finally, the shareRecipe button is responsible for initiating the Android share dialog to be able to share a drink recipe across social networks, emails etc.

* The MainActivity – this activity houses all the fragments in the app. It also contains the logic for the NavigationView, the Drawer and the different Nav Menu items. It contains the following:
  + The onNavigationItemSelected method – this method checks the nav item selected and launches the corresponding fragment.
  + The switchContent method – this is responsible for changing the current fragment within the MainActivity by appropriately replacing the current fragment with the new incoming fragment using the FragmentTransaction class.
  + The onBackPressed method – this method overrides the classic onBackPressed method to correctly displace and replace the right fragments on the Main Activity as the activity is fragment-based.
* The util folder contains the AppStart class that extends from an Application class. This class is responsible for all the functions that should be called on the installation of the application. It contains the onCreate() method that runs the database seeding logic within the DrinkController class.
* The res folder contains the different resources that are used with the application. These include :
  + images within the drawable-?? folders,
  + the layouts for the different fragments and activities,
  + the menu folder that outlines how the nav menu should be implemented,
  + the mipmap that also contain lower quality images and icons used in the application,
  + the navigation used to draw out the navigation controls and path of the navigation MainActivity’s fragments.
  + the values folder that contains all the hardcoded strings, styles and colors and their corresponding IDs that are designed for the application.
  + and an xml folder to contain any extra non-standard layouts that are used by the application.