**Module 6 Portfolio Milestone**

Micah Mock

Colorado State University Global

CSC475: Platform-based Development

Chintan Thakkar

24 December 2023

**Module 6 Portfolio Milestone**

In previous weeks, the planning, analysis of requirements, and layout design for a recipes app was discussed. In this paper, the implementation has taken place and an analysis of the source code along with screenshots will be presented.

First, the app has a homepage. This is scrollable, and the user can click each recipe to view, just as specified in the design phase. The bookmark icon shows the Saved Lists. The Add Recipe activity allows the user to add their own recipes to the app, and also select their own images. Scrolling down reveals the submit button. The Search feature shows how many results there are. The settings screen is sparse compared to the mockup specified in the design phase. However, the functions specified were not necessary to creating a functional app and were left out of development to save time.

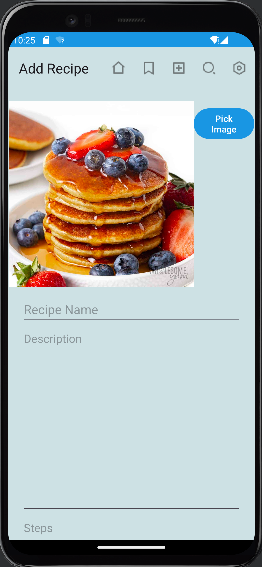
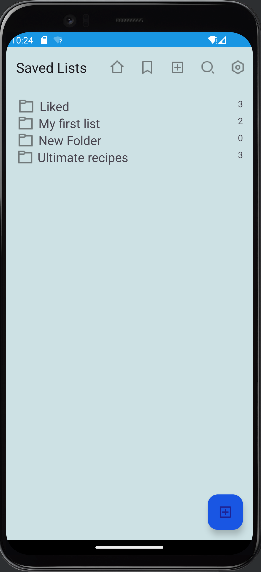
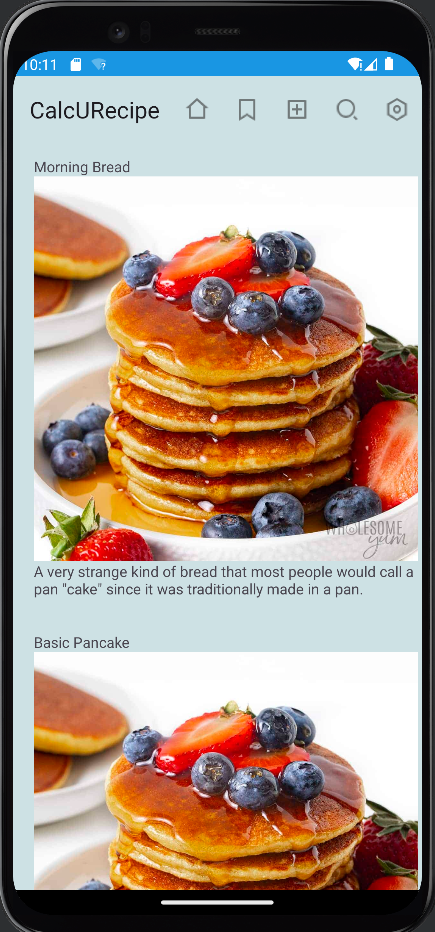


Image 1: Home Page Image 2: Saved Lists Image 3: Add Recipe

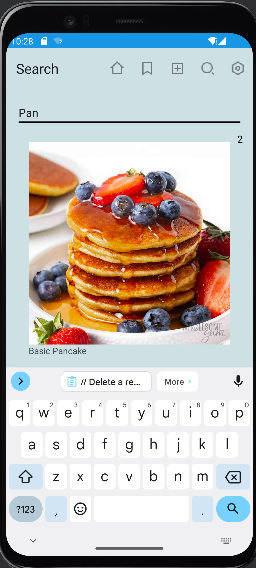
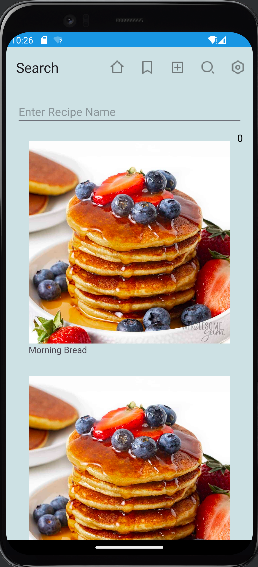
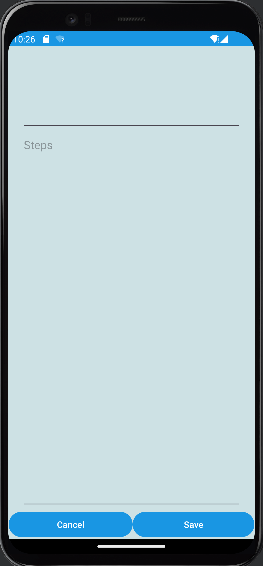


Image 4: Add Recipe Down Image 5: Search Image 6: Search “Pan” 2 Results

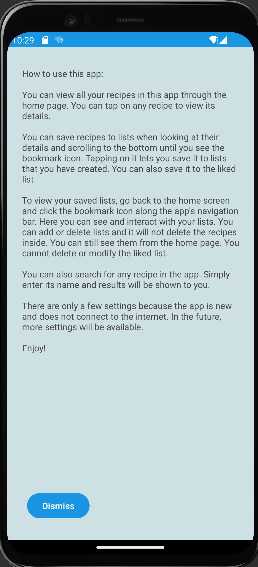
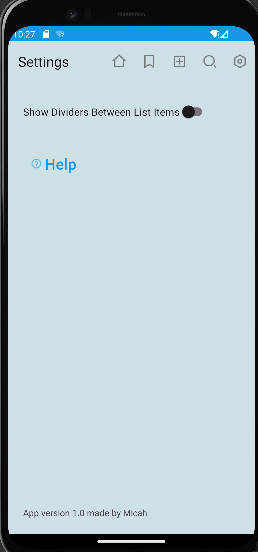


Image 7: Settings Image 8: Settings Help

Notice that the app bar is consistent across each of these activities. This was an implementation challenge due to the nature of android having multiple different ways to implement a toolbar widget. Implementation was not intuitive. Making the same icons on the same toolbar persist across multiple activities required altering the module level build Gradle. In addition, since the app bar is the same across multiple activities, there was a small issue of redundant code in each of the activities. Knowledge and understanding of inheritance was needed. By creating the OptionsMenuActivity class and extending it from the AppCompatActivity class, it allowed for the other activities such as the MainActivity and SettingsActivity to access its methods and reuse the code needed to operate the toolbar. Even after solving the problems of the toolbar, there are still visual design elements that can be improved with the toolbars in the app, specifically when viewing a recipe’s details. It implements a small animation when scrolling and is visually pleasing.

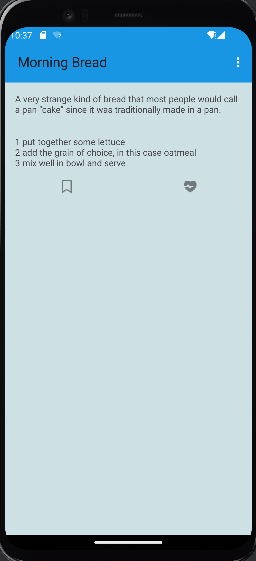
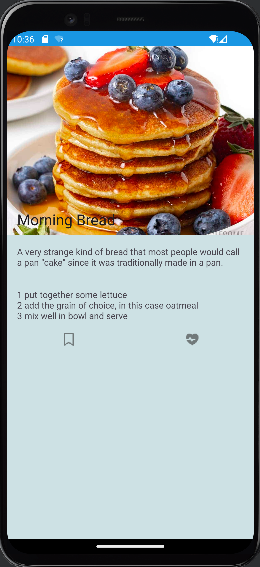


Image 9: Recipe Details Image 10: Recipe Details After Scrolling

The text for the title may blend into the background of certain images due to its color being black. More understanding of xml and the different android themes, light and dark, is needed to correct it, however this was specifically defined as out of scope for this project in the planning phase. It will have to be corrected in later development.

Implementing the ability to create and manage folders inside of the app was another challenge. It was ambitious, and a successful implementation must consider how to store the information for file storage. Since the user can store many recipes in many files, this is known as a many-to-many relationship.

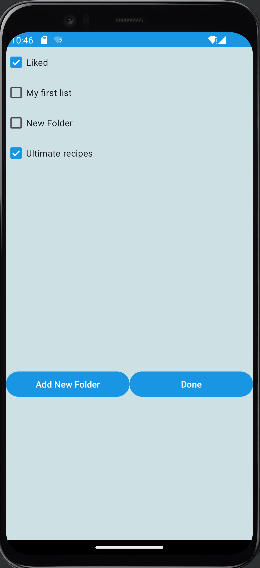
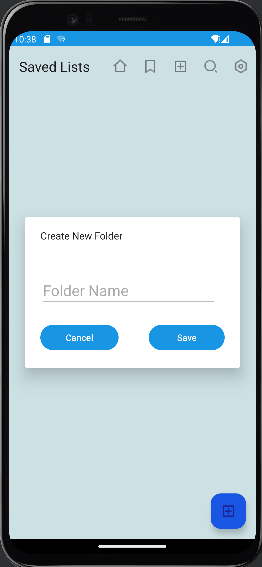


Image 11: Creating a new folder from the Saved Lists Image 12: Saving Recipe to Lists

There are two classes, which would be considered static objects in Java, those being the RecipeManager and the FolderManager. The FolderManager handles a mapping object in addition to the expected ListFolder object. This stores the data. Together, these two classes compose most of the logic and also allow the data for the recipes to be passed between the activities. It is very important to store the things such as folders and recipes inside of an ArrayList structure. However, this kind of structure cannot be passed between one activity to another inside of android very easily. Android has a way to pass data structures, but it is complex and prone to error and programmers must juggle the different versions of android. The libraries provided in android became updated at API 33, so to make this function work with both older versions and the newest version, you would need both new code and legacy code. Instead we can use knowledge of object-oriented programming to create a manager object known as a singleton. Different classes can interact with this manager object to share the necessary data instead. This is in line with the Model-View-Controller software design pattern. In total, there are twenty-three different Kotlin files to handle the logic behind the app, twelve imported drawable resources, and twenty layout files to handle the user interface. The classes are colored differently due to Android Studio’s version control capabilities with git.

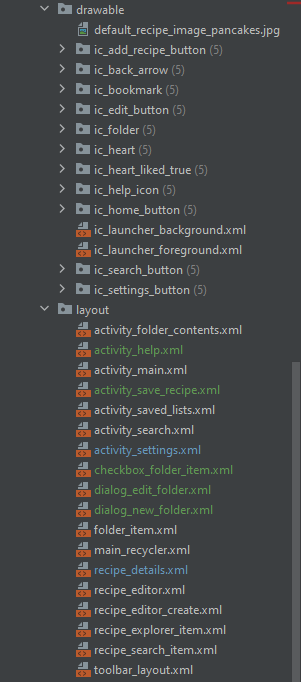
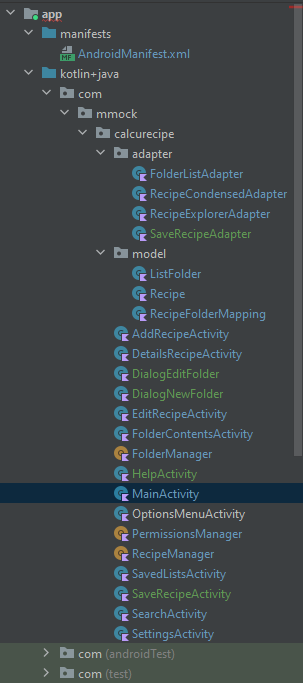


Image 13: Project Classes Image 14: Project Layout Files

In summary, the implementation for this project took twice as long as expected due to the challenges faced along the way. There are still numerous bugs that must be corrected using testing and maintenance. However, by following an agile development methodology, development could have stopped before the Saved Lists function was implemented. It could have stopped sooner, before the Search function was implemented. Each new feature resulted in a complete and functional application due to the iterative design and implementation process. It allowed for rapid design, and break points to stop and test the app. However, inexperience often resulted in poor coding decisions that would cause difficulty with future work. Naming conventions for the files and variables should have been planned better. With each iteration, every new feature being implemented, most previously existing code needed to be modified and would result in unforeseen bugs. As the app got bigger and thus more complex, it became increasingly difficult to add new features while maintaining previous functionality. This is a principle to take into account when considering software implementation.

**Conclusion**

Implementing this android app was a challenge. Certain features were difficult to implement correctly due to the variety of research sources and the fundamental behavior of android’s environment. Knowledge of data management and design were required to find solutions to the problems that were faced. Adding features while following the agile design methodology, that is building a fully functional app with each iteration, allowed for rapid design.

**References**

Horton, J. (2019, April 30). *Android Programming with Kotlin for Beginners*. Packt Publishing Ltd.