Daily List Design

By Talha Ather and Jake Davis

Table of Contents

Project Summary

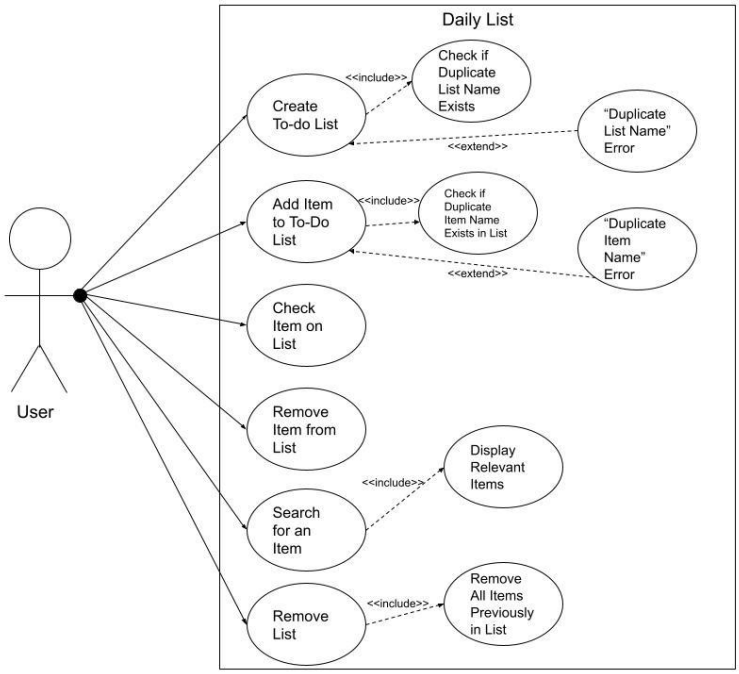
- Project Name:
    - Daily List
- Team Members:
    - Talha Ather and Jake Davis
- Overview:
    - This app will be a todo list app. This app allows us to add new todo lists that users can custom name. These lists can then be selected and items can be added to each individual list. These items can be checked off and deleted when they are completed. There will also be search functionality within the app that will populate all the items on the list that contain the word that was searched for.

Project Requirements

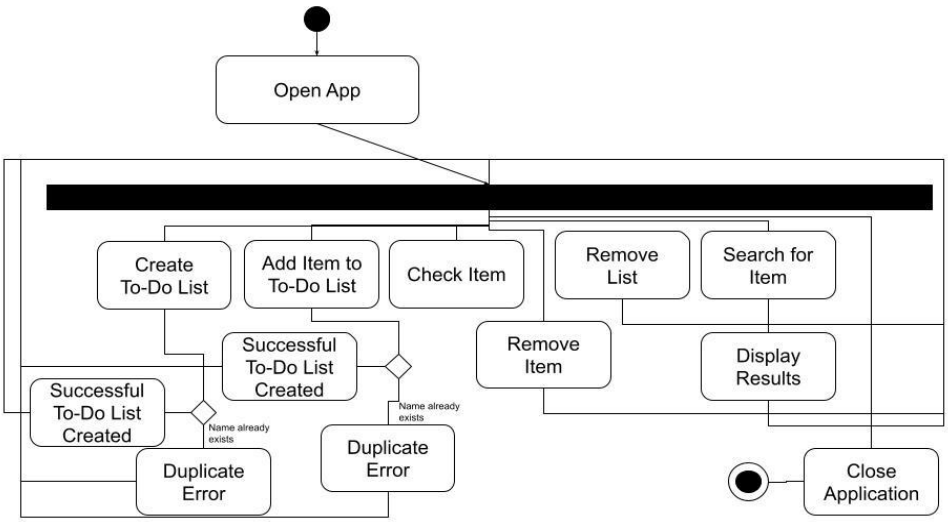
- Functional capabilities:
    - You will be able to create new to-do lists
    - You can add items to the to-do list
    - You can check the items on the list off
    - You can remove items from the to-do list
    - You can search for items based on search criteria
    - You can remove current to-do lists
- Constraints (such as platforms, number of users, etc.)
    - The application will be limited to iOS devices
    - This list will only be locally stored and cannot be synced between devices
- Non-functional characteristics (performance goals, security, usability, etc.).
    - The lists will be locally stored and will persist on the users device

Users and Tasks: Use Cases (Text or UML Diagrams)

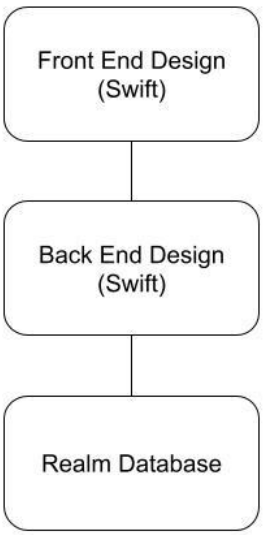
- There will only be one type of user, all should be able to perform every functional capability



UML Activity Diagram



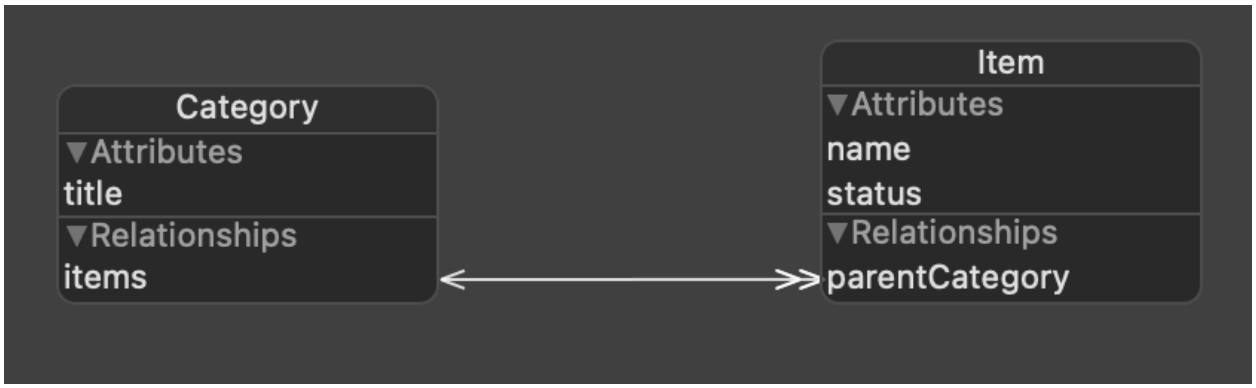
Architecture Diagram



- Our architecture is quite straightforward as everything will be stored locally.

Data Storage

- To persist data in our application we will be using Realm Database which is an open source alternative to SQLite and Core Data. The data will be stored locally on the device and has two tables. One will hold the category and the name and another will hold the items, with their name and status. One category can be associated with many items. A diagram of data details, tables, and relations can be seen below:

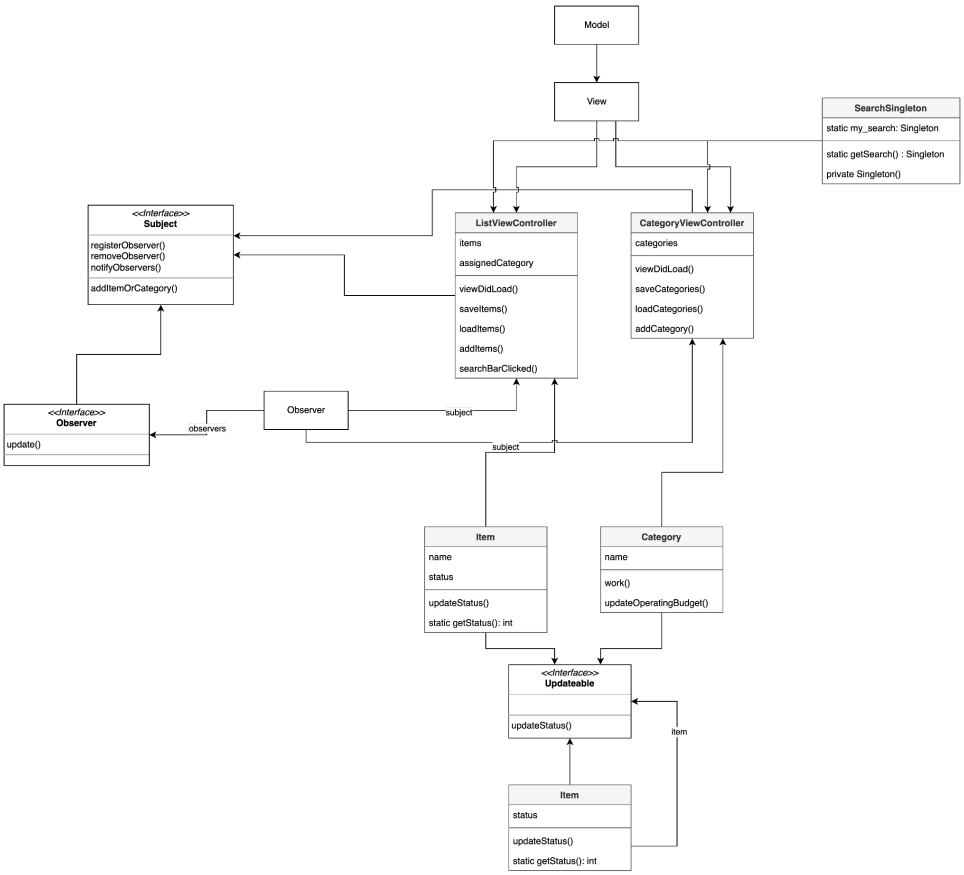


UI Mockups/Sketches

- Working Prototype: https://www.figma.com/proto/NzH3Z4tEOcLd7DzWdyHPdI/Daily-List?node-id=1-3&scaling=scale-down&page-id=0%3A1&starting-point-node-id=1%3A3
- Screen Sketches: https://www.figma.com/file/NzH3Z4tEOcLd7DzWdyHPdI/Daily-List?node-id=0-1&t=2F0dcsCnPw8Yx2RS-0

UML Class Diagram & Pattern Use

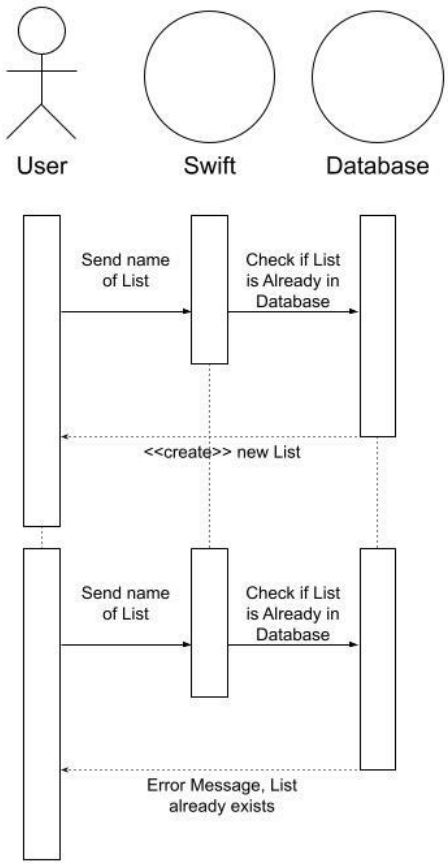
- MVC
- Singleton
- Observer
- Delegate



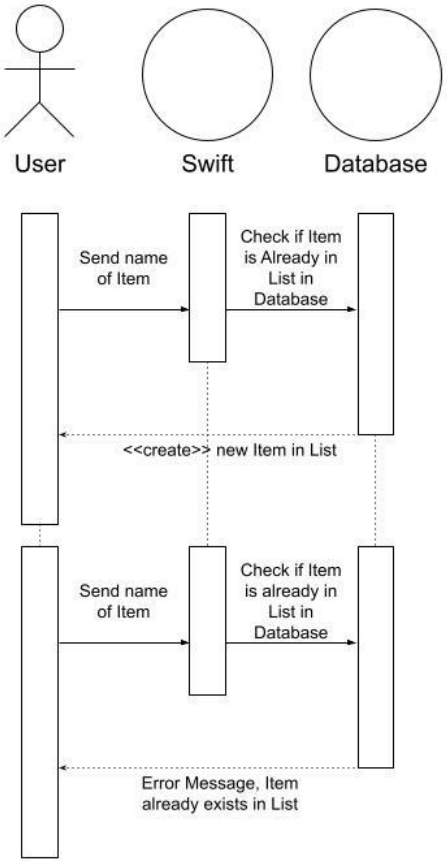
User Interactions/UML Sequence Diagrams

Our program will have the following interactions:

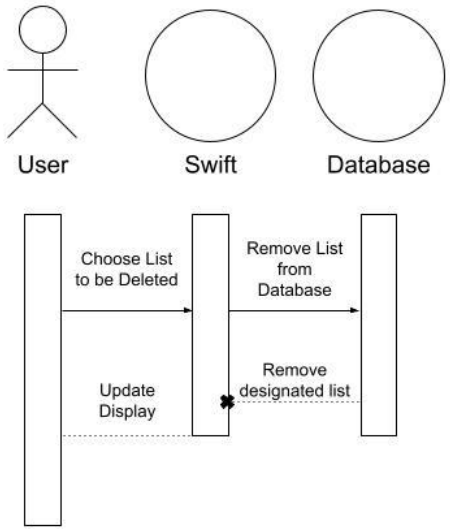
1. Add List
    a. The “Add List” interaction will include the MVC, as it will be added to the model which will in turn update the view. It will also delegate to send the information.



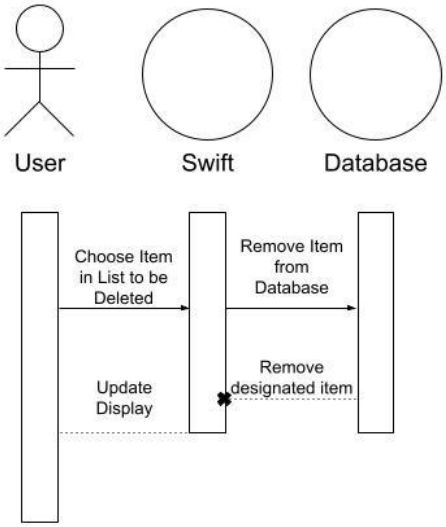
2. Add Item to List
    a. The “Add Item to List” interaction is very similar to that of adding lists, but instead of looking at all lists and their names, it will only look for item names in the designated list. It will also use an observer to update information



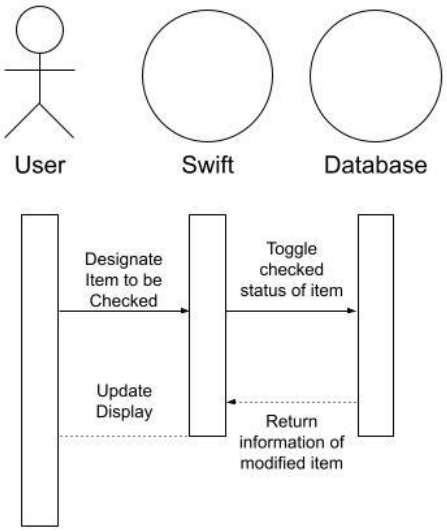
3. Remove List
    a. The “Remove List” interaction will use the MVC to update visually when a list is removed from the database



4. Remove Item From List
    a. This will work very similarly to “Remove List”, but instead of removing the entire list, only one item within a list is removed



5. Check Item on List
    a. This interaction will use the aforementioned MVC and will also use an observer to update the display with what has changed



6. Search for Items on List
    a. This will use the MVC, along with a Singleton to ensure that only a single instance a search occurs at any given time