**Virtual Closet App**

**Team 20 Design Document**

**Team Members:** Divay Gupta, Kartik Uppalapati, Maeve Tra, Meghna Rajagopalan, Oleg Aprelikov

## Index

1. **Purpose** 
   1. **Functional Requirements**
   2. **Non-Functional Requirements**
2. **Design Outline** 
   1. **High-level Overview**
   2. **Sequence of Events**
3. **Design Issues**
   1. **Functional Issues**
   2. **Non-functional Issues**
4. **Design Details**
   1. **Classes Design**
   2. **Sequence Diagram**
   3. **Navigation Flow Map**
   4. **UI Mockup**

## Purpose

People have a lot on their minds these days. Sometimes they lose track of what clothes they have and which ones need to be washed. Sometimes they can’t decide what to wear tomorrow, or they forget to check the weather and get their shoes wet in the rain. Such small details can become a big inconvenience in everyday’s life. Clothing is an essential part of one’s identity and confidence level, but keeping track of what to wear, when to wear it, and what is available to wear requires a lot of effort. Our app is here to help take away that burden. It is designed to serve as the user’s personal assistant, specifically when it comes to closet management. It allows the user to take inventory of their clothes, keep track of their laundry, and decide on clothes to wear from our recommendation algorithm.

There are other virtual closet apps on the market, but none of them have the level of customization ours will have. The main features include keeping track of laundry, recommending clothes based on the weather, syncing with the user’s Google calendar to recommend clothes for specific events, and reminding the user to wash or iron a clothing item if they need that item for a future event. The user will be able to set their preferences on when, how often and what they want to be reminded of. This high level of customization, along with the recommendation algorithm, allows the user to tailor the experience to their specific needs. We aim to make the app a true personal assistant that the user can access anytime and anywhere.

### Functional Requirements

**User account**

1. As a user, I would like to be able to register an account using my Google account (gmail).
2. As a user, I would like to be able to login and manage my account.
3. As a user, I would like to be able to reset my password.
4. As a user, I would like to be able to sync my Google calendar with the app.

**Managing closet**

1. As a user, I would like to be able to add clothing items to my closet
2. As a user, I would like to be able to remove clothing items
3. As a user, I would like to be able to select a specific clothing item and see the details for that item on a details page
4. As a user, I would like to be able to edit details of all my clothing items.
5. As a user, I would like to be able to open my phone’s camera from the app and take pictures of my clothes to add them to the closet.
6. As a user, I would also like the option to upload photos from my phone’s gallery.
7. As a user, I would like to have the details of the items (color, sleeve length, etc) to be recognized by an Artificial intelligence and filled out automatically.
8. As a user, I would like to be able to edit any details that were recognized incorrectly by the Artificial Intelligence, and also report that the mistake was made.
9. As a user, I would like to be able to view all my items in the closet, under a specific category in the closet (tops, bottoms, hats, shoes, etc).
10. As a user, I would like to be able to search for an item using the search bar.
11. As a user, I would like to be able to filter the items based on their colors, materials, etc.
12. As a user, I would like to extract my item from its original background and put it on a custom background (if time allows).

**Clothing Recommendations**

1. As a user, I would like to receive recommendations on what to wear based on the weather and/or my schedule in my Google calendar.
2. As a user, I would like to receive recommendations for one type of clothing item (tops, bottoms, shoes, hats, etc) at a time.
3. As a user, I would also like to receive full outfit combination recommendations if I choose the option to allow it (if time allows)
4. As a user, I would like to be able to approve or decline the recommendations.
5. As a user, I would like to be able to receive new recommendations if I decline the old one.
6. As a user, I would like the clothing item to be automatically put in the “Laundry basket” after I have accepted it a certain (user-specified) number of times
7. As a user, I would like to be able to say that an item is my favorite so that it is recommended more often.

**Outfit Combinations**

1. As a user, I would like to be able to mix and match items in my closet to create outfit combinations.
2. As a user, I would like to be able to save an outfit combination that I like.
3. As a user, I would like to be able to browse my saved outfit combinations.
4. As a user, I would like to be able to edit a saved outfit combination.
5. As a user, I would like to be able to delete a saved outfit combination.
6. As a user, I would like to be able to name a saved outfit combination.
7. As a user, I would like to be able to search for a saved combination using its name.
8. As a user, I would like to be able to upload a picture of myself and then see the clothing items modelled on the picture. (if time allows)
9. As a user, I would like to be able to assign an outfit combination for a specific date.
10. As a user, I would like to be able to set weather preferences for a specific outfit combination so that the app recommends that outfit on a day with that weather (if time allows)

**Laundry**

1. As a user, I would like to be able to select a clothing item in the closet and see if it is in the “Laundry basket” or not on the item’s details page
2. As a user, I would like to be able to see what items are in the “Laundry basket” (looks similar to seeing what clothes are in the main closet)
3. As a user, I would like to be able to select a clothing item in the “Laundry basket” and see its details on a details page.
4. As a user, I would like to be able to add items to the “Laundry basket” using a toggle on the details page of that specific clothing item
5. As a user, I would like to be able to remove items from the “Laundry basket” using a toggle on the details page of that specific clothing item
6. As a user, I would like to be able to clear all items from the “Laundry basket” at once.
7. As a user, I would like to be able to tell if an item is in the “Laundry basket” from the main closet (“To Laundry” and “In Laundry” button).
8. As a user, I would like to be able to set how often I want to do my laundry.
9. As a user, I would like to be reminded to wash an item if it is still in the “Laundry basket” (the reminder is sent periodically, with the time period being the user-selected time for how often they want to do laundry; default is 1 week)
10. As a user, I would like to be reminded to wash an item if I need the item for an upcoming event, or if the outfit has been assigned to an upcoming date.
11. As a user, I would like to be able to set how far ahead I want to be reminded to do my laundry.
12. As a user, I would like to be reminded to do laundry if my last laundry load exceeds the duration set above (i.e. if I want to do laundry every week, remind me if I haven’t done it on the 8th day).
13. As a user, I would like to receive notifications from the app to do laundry.
14. As a user, I would like to see the last time a clothing item was washed on its details page (if time allows)
15. As a user, I would like to set reminders for washing a specific item for a specific time period that might be different from the normal laundry time period (example, for jackets that are not washed as often) (if time allows)

### Non-functional Requirements

**Performance**

As a developer,

1. I would like the application to be launched in 5 seconds.
2. I would like the Artificial intelligence to be able to recognize an item within 5 seconds with 80% accuracy.
3. I would like the application to be able to process and update a new item to the user’s closet within 1 second with no need to exit/refresh the application.
4. I would like the application to be able to support consecutive “Add to Closet” requests from the user.
5. I would like the application to support offline access, in the sense that the user can still view their closet and laundry, as well as add and remove items from closet/laundry (if time allows).

**Compatibility**

As a developer,

1. I would like the application to be compatible with the latest version of Android.
2. I would like the application to be compatible with the latest version of iOS. (if time allows)
3. I would like the Artificial Intelligence feature to be integrated for Android devices.
4. I would like the Artificial Intelligence feature to be integrated for iOS devices. (if time allows)
5. I would like the application to be able to sync with Google Calendars.
6. I would like the application to be able to sync with different calendar platforms (if time allows).

**Usability**

As a developer,

1. I would like the application to have an aesthetically pleasing UI design.
2. I would like the application to have a user-friendly and easy-to-navigate UI design.
3. I would like the application to support dark mode (if time allows).
4. I would like the application to be able to run on web browsers so the user can access their closet whenever and wherever (if time allows).

**Security**

As a developer,

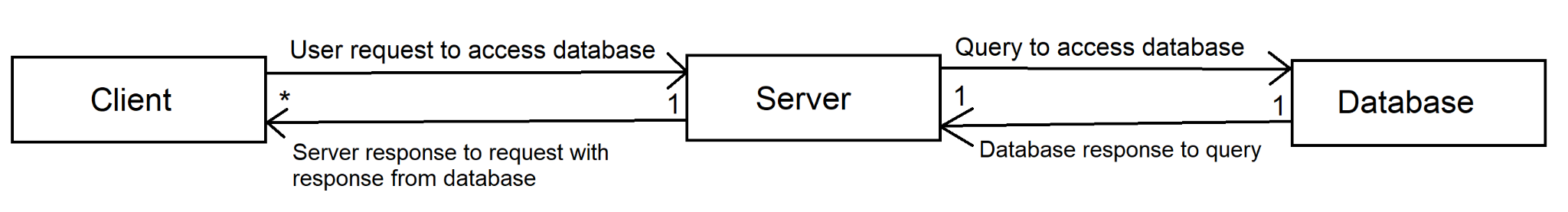
1. I would like to protect the user's password stored in the database.
2. I would like to protect and anonymize any user’s information collected within the app, such as camera photos and location data (if time allows).

## Design Outline

Design Model: Client-server model

We chose this model because this app is an organization app, and each user’s clothing items and other data will be stored in a database on the server. The same database contains data from all users. Whenever a user wants to access their clothing inventory, requests will be made to the server for the user’s data, and the server makes queries to the database. Once the database responds to the query with the requested data, the server returns that data to the client.

Client-server model diagram:



Client:

* This is the user’s end of the system
* All of the user’s actions are enabled by the client
* Whenever the user wants to access their clothing inventory, either to view it or edit it, the client makes a request to the server, and interprets the server’s response
* There are multiple clients - 1 per user

Server:

* There is only 1 server, which handles all the clients
* The server receives requests from the client
* When the server receives a request from the client, it makes the necessary query to the database.
* When the database responds to the query, the server interprets the response, and then uses the response to respond to the client

Database:

* The database stores the data of all the clients - email address, account password, images of clothes, details about clothes, etc.
* The database receives queries from the server, and responds appropriately, sending the requested data back to the server and/or making the requested changes to the data.

## Design Issues

### Functional Issues

1. Should we create a separate screen for the laundry basket, or have a filter on the closet screen?  
   Option 1: 1 screen, which is the closet screen, with a filter option that allows the user to only view clothes that are in the laundry basket.  
   Option 2: 2 separate screens; one for closet and one for laundry basket  
   Option 3: To the user, it looks like 2 separate screens, but we only create one screen, with the same structure, for both the closet and the laundry basket. The difference is that whichever option the user selects triggers a different query (to the database) that loads either the virtual closet or the laundry basket.  
     
   Decision: We chose option 3 because it seems like the most efficient choice for us as developers, so that we don’t have to create 2 different screens with the same structure, while also being more user-friendly than having the user filter their closet every time they want to see the laundry basket.
2. What should the Home page (landing page after login) look like?  
   Option 1: Simple list with just text, showing options for viewing the closet, viewing the laundry basket, viewing the clothing recommendations, view outfit combinations, etc.  
   Option 2: Tile grid (2 columns) with logos and text, showing options for viewing the closet, viewing the laundry basket, viewing the clothing recommendations, view outfit combinations, etc.  
   Option 3: Tile list with logos, below the weather for that day and a clothing recommendation for the day, along with reminders  
   Option 4: Weather for that day, a clothing recommendation for the day, along with reminders; Options for opening closet, laundry basket, clothing recommendations, outfit combinations, etc. in tabs in a navigation bar at the bottom of the screen (with logos)  
     
   Decision: We chose Option 4. At first we were leaning towards Option 3, because that would allow more room for all the options (closet, laundry basket, etc). However, now we have chosen Option 4, as the tab list can be scrollable and it uses the space more effectively.
3. Should we recommend an entire outfit or a single piece of item?

Option 1: An entire outfit

Option 2: A piece of item

Decision: We chose option 2, to recommend a piece of item instead of an entire outfit.. We provide a platform for them to do this so the user can mix and match items up to their liking because their choice of outfit can be unpredictable and subjective.

1. What happens after the user rejects the initial recommendation?

Option 1: No more recommendation, the user chooses from the closet

Option 2: Continue recommending one at a time until the user accept

Option 3: Present a gallery of recommendations for the users to choose from

Decision: We chose Option 3. By presenting the user with a gallery of recommendations, they will be able to go back and forth and pick out the most desired item that will still be suitable for the weather and events. Also it will decrease the stress on the system and its response time because it does not have to calculate and update the UI for each new recommendation.

### Non-Functional Issues

1. What type of application should our app be?

Option 1: Web app

Option 2: Native app

Decision: We chose Option 2 because we need to use the camera to take pictures of the clothing item. Also, we are hoping to support offline access to the closet so we think a native app would make it easier.

1. What operating system should the application support?

Option 1: Android

Option 2: iOS

Option 3: Both

Decision: We chose Option 3 as our goal after finding out that we can develop a native application for both Android and iOS using Flutter. Even though using Flutter means that we will be using a different programming language, Dart, instead of other languages we are used to, we think Dart is worth learning if it allows our application to support both operating systems. However, even though this is our goal, there may be additional obstacles in future sprints that prevent us from finishing the iOS version of the application.

1. What type of database is the most appropriate for our application?

Option 1: Relational database such as SQL

Option 2: NoSQL database

Decision: We chose Option 2 using Firebase because it supports cross-platform applications (Android and iOS), real-time data synchronization, and offline access. Moreover, a noSQL database is horizontally scalable, making it easier to grow the user’s database compared to the fixed SQL databases. It also allows more flexibility for us in storing data of different types of clothing items.

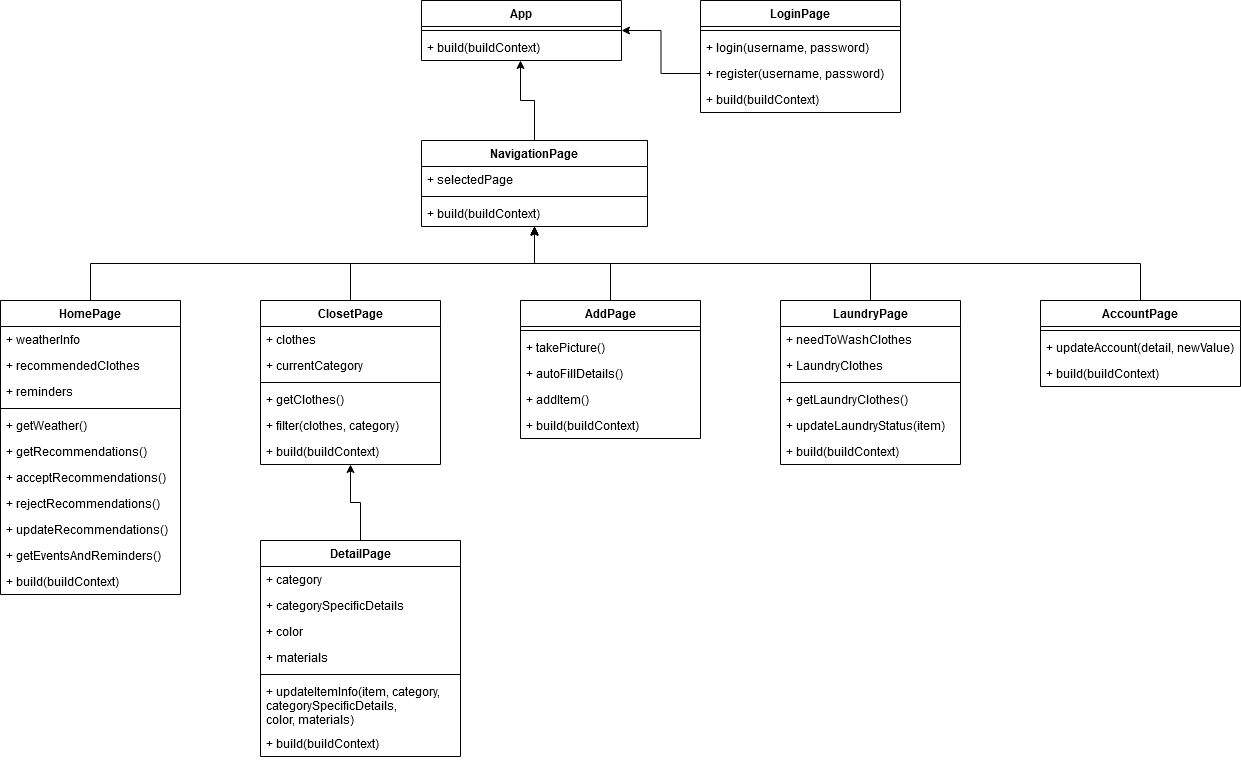
1. Which API should we use to check the weather?  
   Option 1: OpenWeatherMAP API  
   Option 2: Weatherbit API

Decision: We chose option 1 because we found a package to use this API within the Flutter environment (weather 2.0.1 from pub.dev).

## Design Details

### Class Design

Flutter is centered around pages and the following class diagram reflects this:



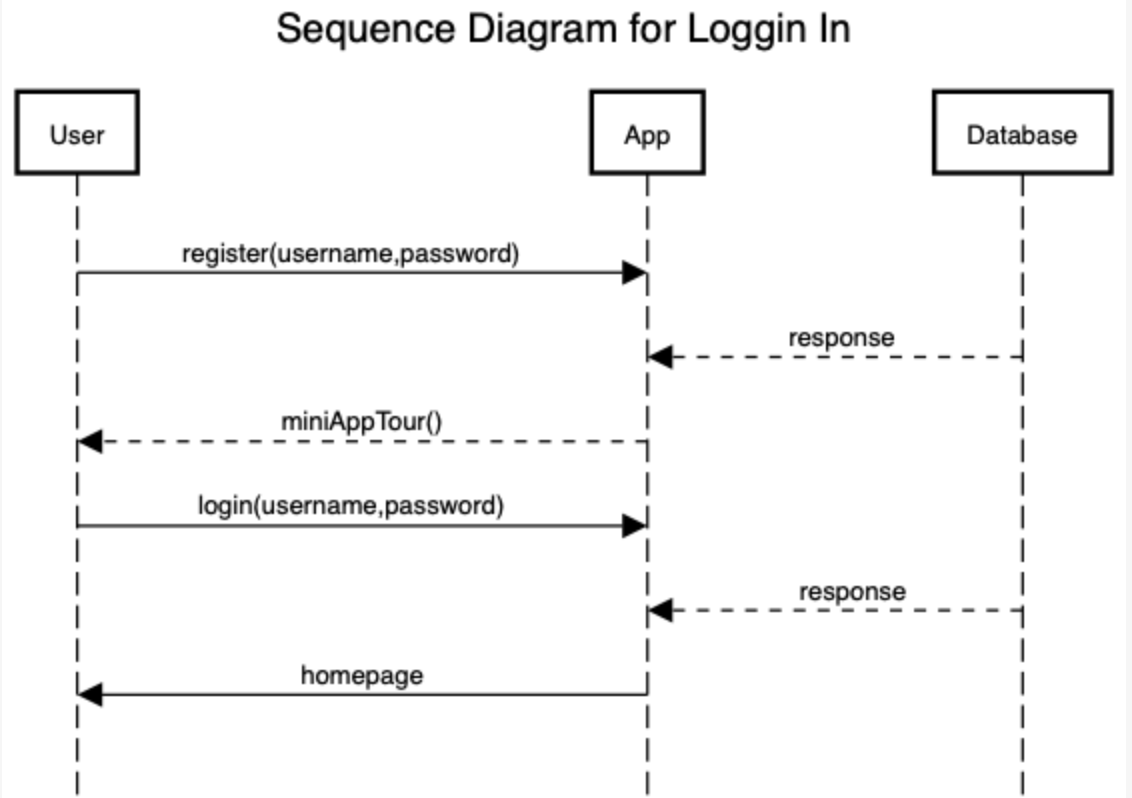
Since we don’t know what problems we’ll run into, the diagram may miss some necessary fields/methods.

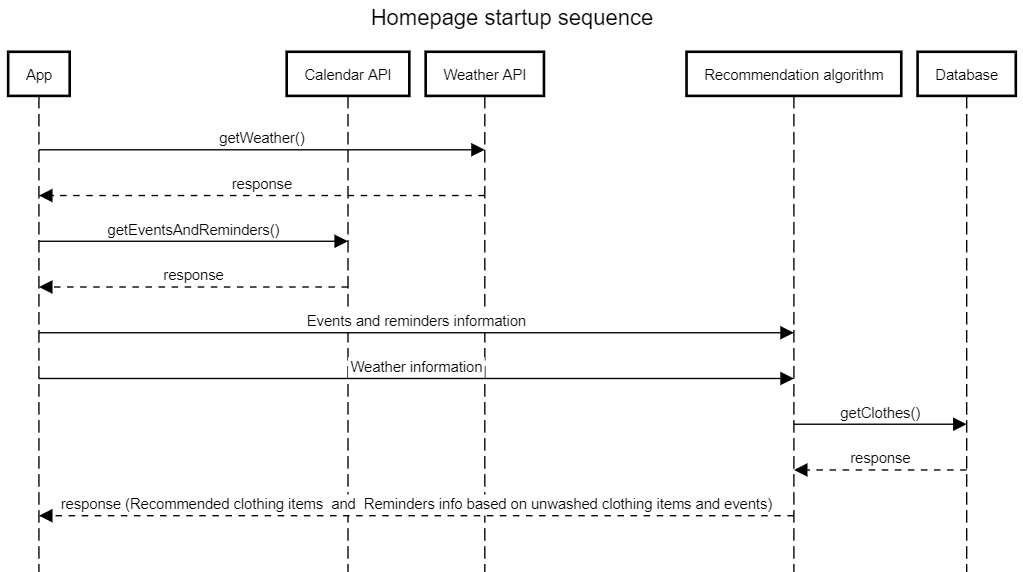
Each page is represented by one or two classes depending on whether it has any state. The arrows represent one page building another inside the build() method.

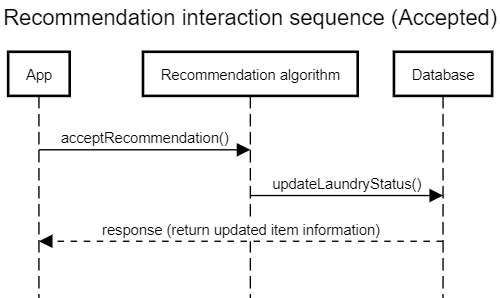
| Class | Description | Relationships |
| --- | --- | --- |
| App | Flutter’s application entry point. | If the user is not logged in, open the LoginPage. Otherwise, open the NavigationPage. |
| LoginPage | Allows the user to log into the app. | Once logged in, transfers the user to NavigationPage. |
| NavigationPage | A page that creates a navigation bar and chooses between a number of pages to display based on the current tab. |  |
| HomePage | Default page that displays the weather, clothing recommendations and reminders. |  |
| ClosetPage | The page that displays all of the available clothing in a grid. Allows to filter clothes by category. | If one of the items is pressed, opens a DetailPage with that item. |
| DetailPage | Displays detailed information about a particular item of clothing. |  |
| AddPage | Class for the page for adding new clothes to the virtual closet. This page opens the phone’s camera, and then asks the user to enter details about the clothing item into text boxes. |  |
| LaundryPage | Class for the page for the “Laundry basket”. This page shows the user all the clothes that are in their laundry basket. |  |
| AccountPage | Class for the page displaying the user’s account details. It allows them to make edits to all editable account details. |  |

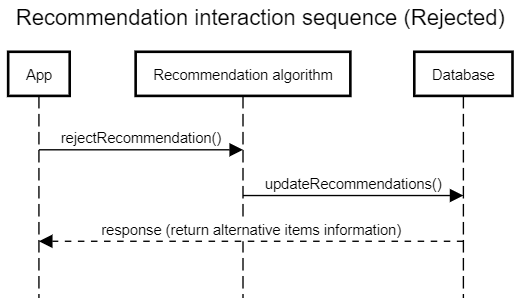
### Sequence Designs

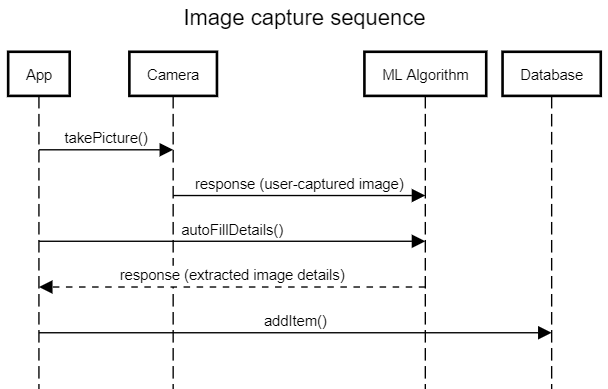
The following sequence diagrams detail the specifics of the functioning of the different sections of the app. Details may be added or removed as needed as the app is built and tested.

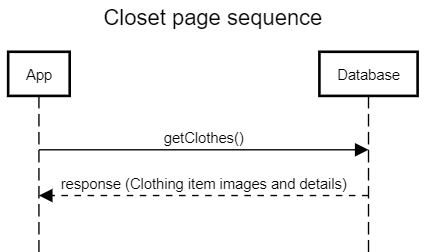


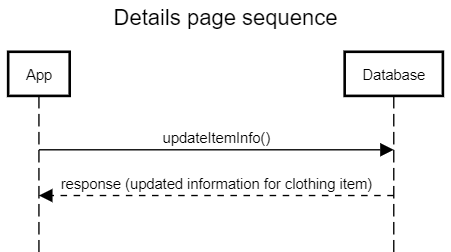


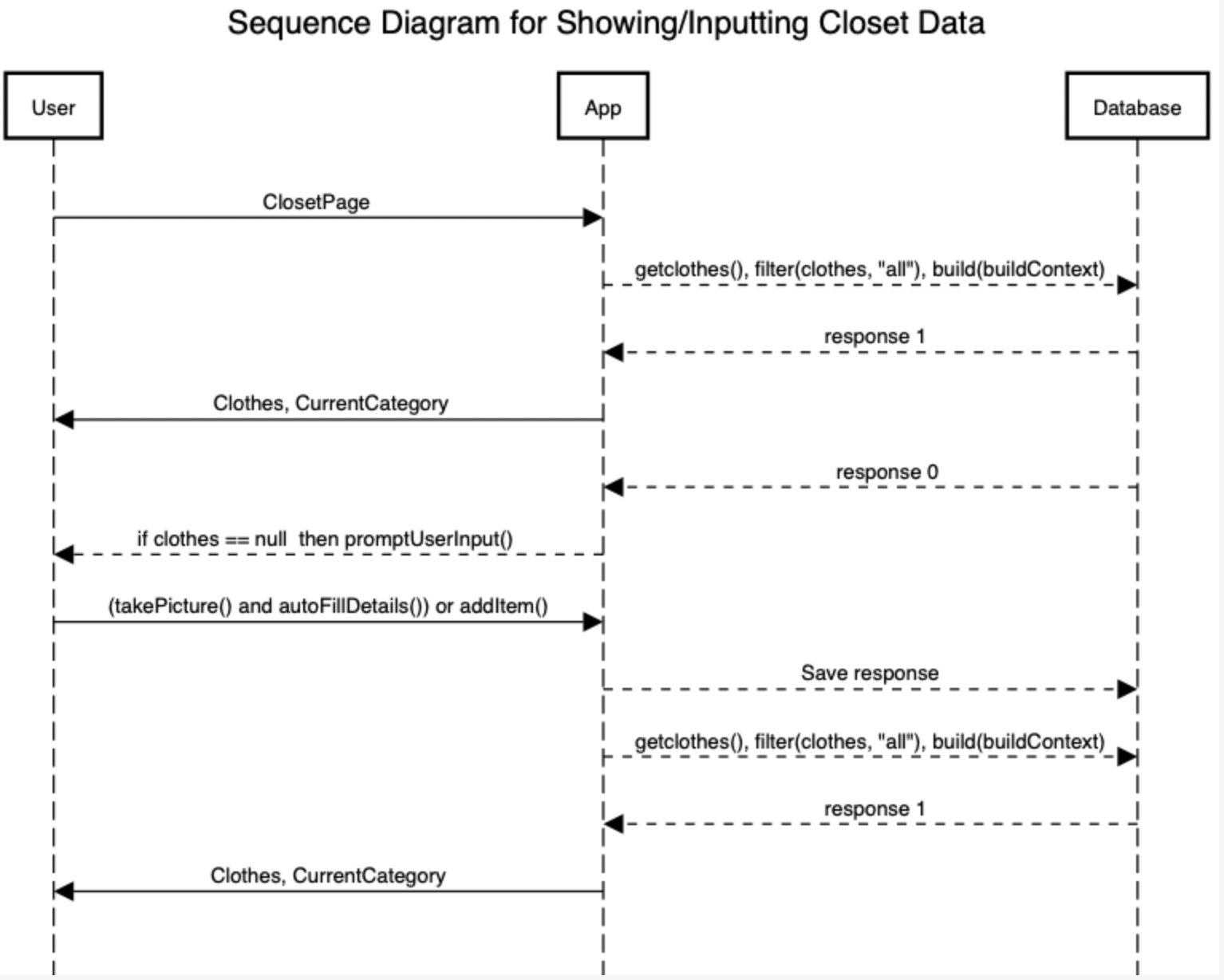


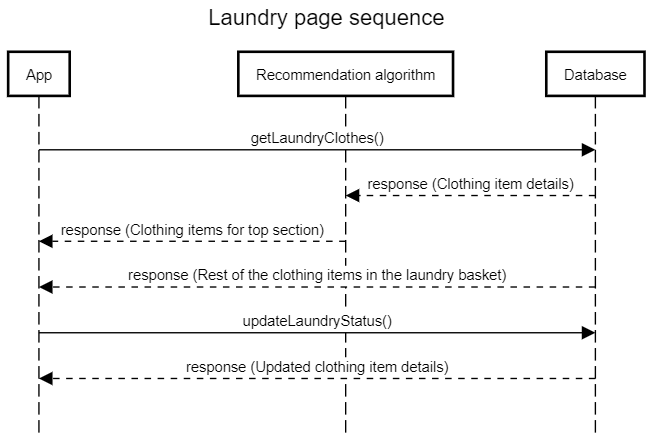






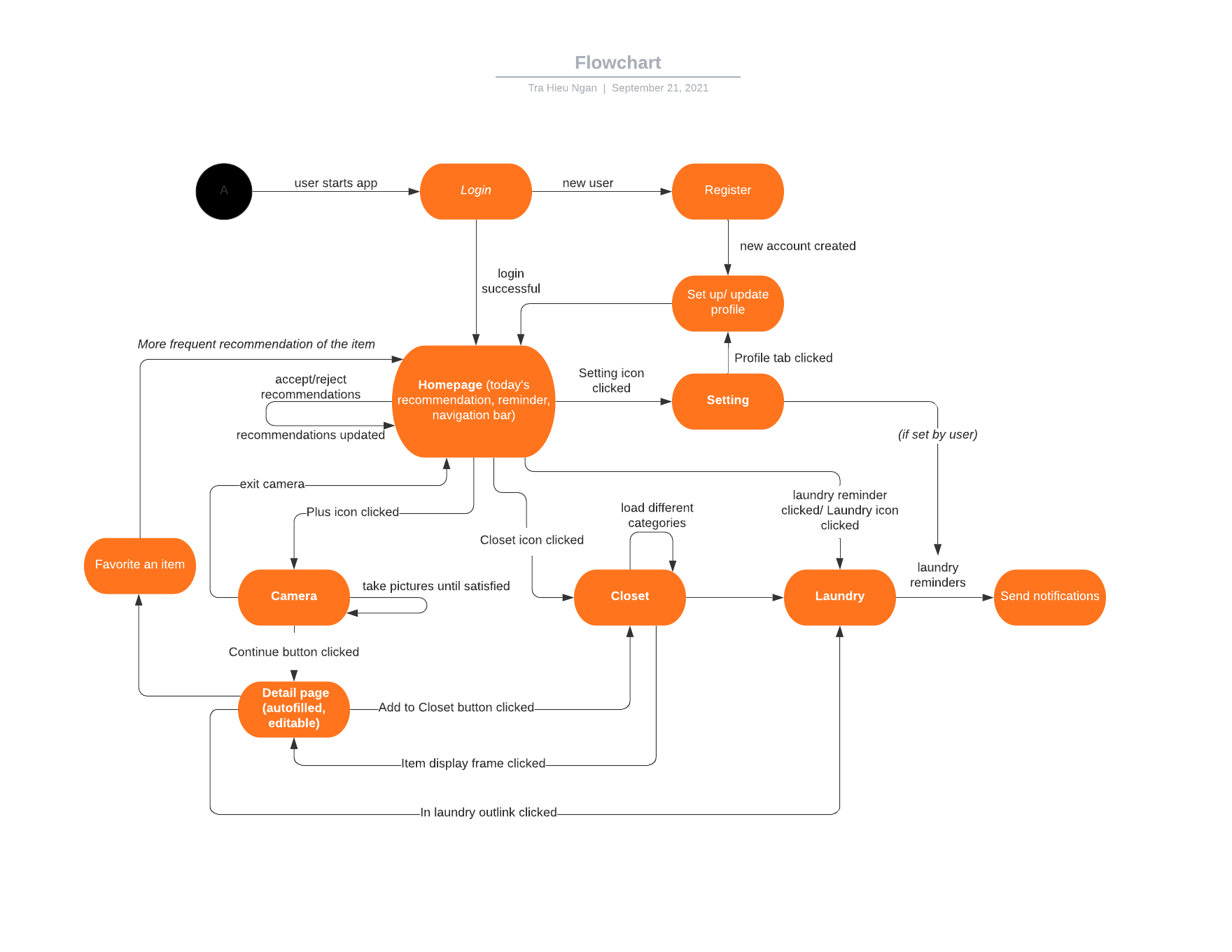






### Navigation Flow Map

The user is required to login/register an account to access the homepage. After login, all modules are accessible through the navigation bar at the bottom of the page. The user can use this bar to go back and forth between any tab at all times. Every module in the app contains functional buttons/links that will take the user to other modules or submodules. We aim to make these as intuitive as possible so the user can easily predict where a button/link will take them.



### 

### UI Mockup

* **Navigation bar**



The navigation bar will be at the bottom of the app and contain five different sections of the app. The first icon is for the homepage. The second is to view the closet. The third is to access the camera and add more items. The fourth is for the laundry basket. And the last icon is to manage the user’s account. More details of each section will be discussed below.

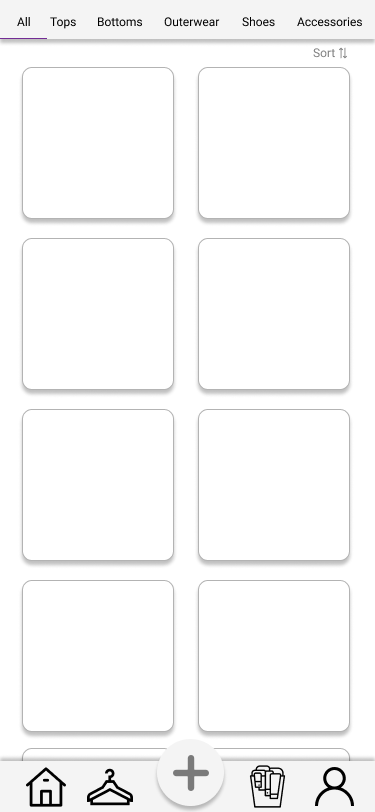
* **Homepage**



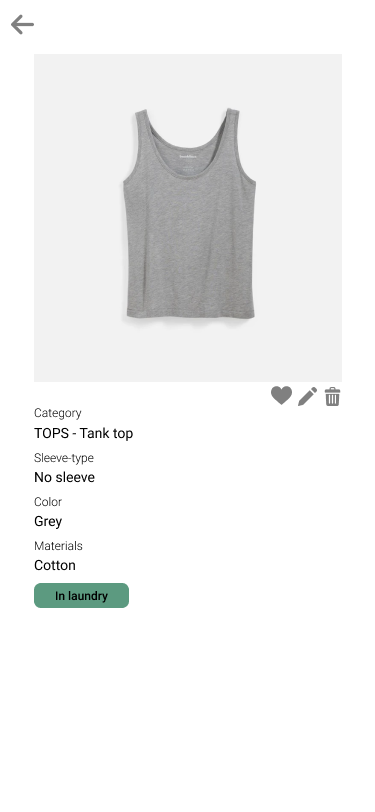
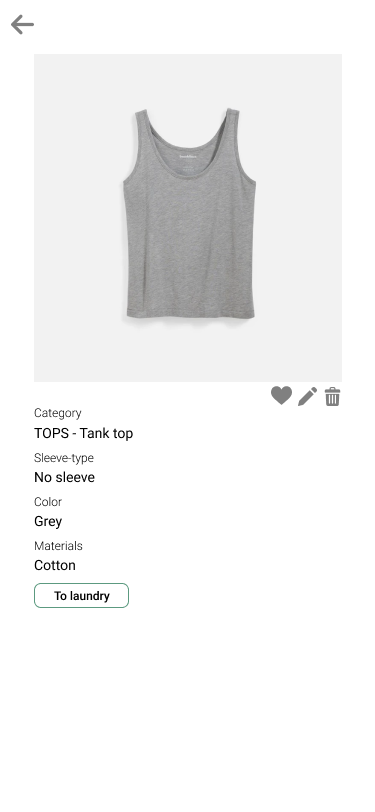
The homepage will feature today’s weather and personal events, as well as the recommendations based on all the factors above. The user can tap on the picture of an item to bring up the details of that item (see below). The user can double tap the picture to accept the recommendation or swipe left/right to see other recommendations. Once the user accepts a recommendation, the item is automatically added to laundry.

There is a section for reminders. There will be reminders if the user needs to wash an item/do laundry in general because they might need an item for a close-future event (the user can set preferences for how far ahead they want the reminders to be). By tapping on one of these reminders, the app will direct the user to the laundry page.

* **The closet**

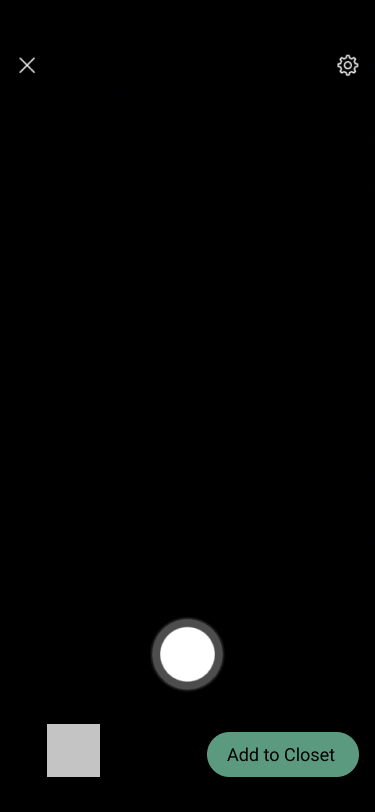


The closet features different categories within the user’s database. The user can choose “All” to view the entire closet or choose one specific category. The items will be displayed in grid view, with the ability to sort based on the items’ subcategory (for example, Tops would have Blouse, TShirt, Shirt, etc.). Once the user tap on an item, the detail page will pop up:



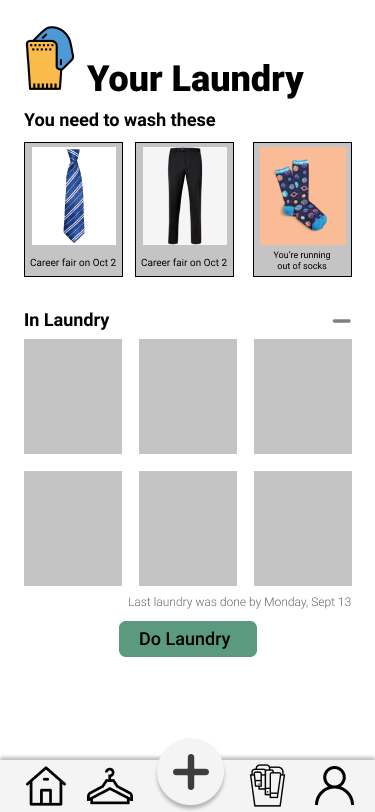
The detail page will have the item’s picture (taken by the user) and the details of that item. The user can edit these details by choosing the pencil button or deleting the item from the database. The user can also favorite an item, which will make that item appear more often in the recommendations. The user can also add an item to laundry (if it’s not already in laundry) by choosing the “To laundry” button. After clicking on that, the item will be added to the laundry basket and the button will become “In laundry”. Vice versa, if the item is already in the laundry, the button will be “In laundry” and the user can choose to remove it using the same button

* **Camera access**



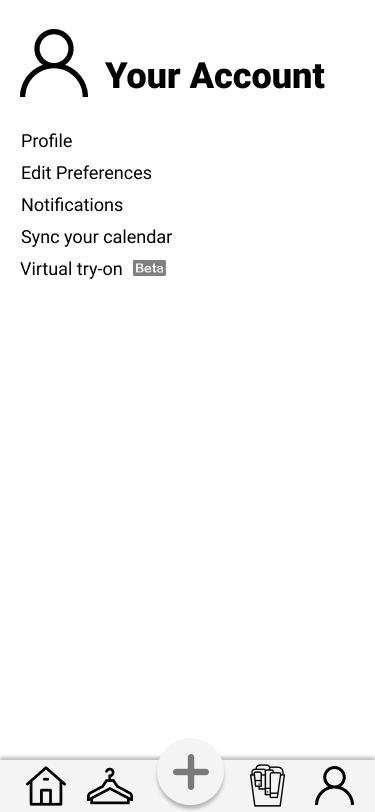
By choosing the plus (+) button from the navigation bar, the app will open the camera page. The user can take pictures of new items using this camera page and once they are satisfied with the picture, they can choose “Add to Closet” and the item will be added to the user’s database.

* **The laundry basket**



The laundry will have two sections. The top section features all the items that need to be washed, the deadline (if available), and the reason. These are what trigger the reminders on the homepage. The bottom section contains all the items that are in “laundry” (added from the recommendation and the user’s manual input from the closet tab). The user can remove the items from the laundry if they wish to by choosing the minus (-) section before they tap on the “Do Laundry” button. After that, all the items that are “In Laundry” will be updated in the closet.

* **Account management**



The account tab will contain the user’s profile (name, DOB, etc.). The user can edit their preferences from here (how far ahead should the reminders be, how often they want to do laundry, etc.). The notifications can also be set here if the user wants to receive notifications for the recommendation, reminders, and laundry. The user can also sync their calendar from here.