**V- Closet Project Design**

# 1. PROJECT NAME

V-Closet

# 2. GROUP MEMBERS

Scott Lam, Leo Ras, Thomas Simmons

# 3. REVISED REQUIREMENTS

We will be using the same requirements from the Requirements document.

# 4. DESIGN DESCRIPTION

The V-Closet website will consist of 9-10 pages using a minimalistic design with a black-and-white color scheme. The website will mostly be done by components. This means features, user interface elements, or inputs will be done separately and integrated with the website. This allows for code reusability and easier to manage the code structure. There will be the use of some functions if needed or problems implementing a component.

One of the prevalent components that most pages will use is a top navigation bar that consists of the Home icon, Toggle Icon, and Settings Icons ([See Components Diagram](#_8l5uaqozv5om)). The Toggle icon is its own component that allows it to be reusable which is integrated into the top navigation bar file and will remain static. The Home icon when pressed will take the user back to the home page. There is no component needed for this as it reroutes the user back to the home page. The same concept is applied to the Settings icon as it directs the user to the Settings Page. The Toggle icon when pressed will change the background of the website to gray or an off-white color depending on the color theme present. The ToggleMode.js uses a state to keep track of what the theme is and allows it to set the theme via a ternary operator. The same thing is also applied to the toggle icon that will change from the moon to the sun depending on the theme. The Settings icon when pressed will take the user to the settings page where they can change the password or delete their account ([See User Interface Storyboard](#_n7yezsrsocyb)). Another option will be listed on the left called “About” which details the history, mission, and goals of V-Closet. The icons that are used are imported from a react icons library which lists hundreds of icons to use from.

The index.js file renders any page or component. The index file takes information from the app.js which outputs the main.jsx file which is the landing page. The index file is essentially used to render the other pages which means a significant amount of imports are made to the file. From the main.jsx file, the user can route through other pages via buttons or icons ([See Components Diagram](#_8l5uaqozv5om)).

When the user arrives at the landing page (or main.jsx), they will be prompted with two buttons: “Login” and “Sign Up” which when pressed, will take the user to the corresponding pages. The user may switch between the pages if they accidentally clicked the wrong one. The Login Page will incorporate a login system component that will use authorization given from the database Firebase. The Sign Up Page will incorporate a similar system but will add the newly created user to the database. The user must provide valid login or sign-up credentials to bypass the page to get to the home page. If not, a window will appear notifying the user that the information was incorrect or not sufficient. Sign-ups, for the time being, will be done via Gmail and email/password. Users will also be able to use email links to sign in. This has the added bonus of verifying the user’s email. Once the sign-up is successful, the user will be added to the database and then directed to the Home page. There will be separate components for the sign up system and login system.

The most challenging design aspect of the website will be the Home Page, Wardrobe Page, and Market Page. The Home Page will use a weather API to pull in weather information. To implement this, a separate component will be created called WeatherTracker pulls information from the weather API and tracks the user’s location. ([See Components Diagram](#_8l5uaqozv5om)). The Home Page will feature multiple images of different categories of clothing recommended for the user to wear. When one of the images is clicked, the user is directed to the Wardrobe Page in the category the user picked ([See User Interface Storyboard](#_n7yezsrsocyb)). The outfit recommendations are based on weather information. This would also mean the website will need to include a feature to keep track of the user’s location to provide accurate data and clothing recommendations. The website will also factor in rain and snow temperatures which will be discussed later on the Wardrobe Page.

The Recommendation.js component will consist of the key logic behind recommending the user what to wear. With the imported weather information from WeatherTracker.js, each clothing item will be run through a set of conditions according to that day’s weather stats. Each condition will add or subtract a total value for each clothing item. These checks will include the minimum, maximum, and current temperatures along with the forecast. The highest valued items will be recommended to the user. The goal of this project is to consistently recommend the same clothes based on the same set of weather conditions.

The Wardrobe Page is where the user’s inventory of clothing is. The layout of the page will have a left-side navigation bar that is categorized by the type of clothing. Some categories include hats, t-shirts, pants, footwear, etc. Also on the left side of the page, there will be a color sorting option for your clothes. Colors are chosen by what the user tagged in their clothing item. There may be a separate component implemented for this feature called ColorPicker that will sort the clothes by color in the wardrobe. Color will be represented as small circular icons to match the minimalistic design. Clothes are presented as images, so when the user clicks on the image, it will prompt the user to list the item or delete the item ([See User Interface Storyboard](#_n7yezsrsocyb)). A border will be provided for the selected item for ease of visibility. If the user selects the list item option, it will show a message showing it has been listed and will appear on the List Item Page to be viewed. From there, you will add the price for the item you listed. Another option in the left-side navigation panel is the “add item”. This will take the user to Add Item Page.

On the Add Item Page, they are able to upload photos of their clothes to be added to their virtual wardrobe. An Image Uploader component will be implemented to allow the user to upload pictures of their clothes. Before the user is able to upload their clothes, the website will ask for some information regarding their clothes. Some information includes whether the clothing is suitable for winter wear, rainy wear, hot wear, etc. Information about the clothing item itself such as category, size, color, etc ([See User Interface Storyboard](#_n7yezsrsocyb)). The user will also be asked to input information about the clothing itself and a description. This will help provide more accurate outfit recommendations for the user. Once the user is able to upload their clothes, the database logs the newly added clothes from the user ([See ERD](#_wweiq9untbrw)).

ImageUploader.js handles the database functions to upload the images to Firebase Storage. This component will verify that the user submits a valid file extension and acceptable size. While the image is being uploaded, this component calculates the upload progress and reports it to the user. There will be a lot of error handling to support the user in submitting the correct image. Submitting a file could pose challenges to some users, so the process should be very user friendly.

The Market Page will provide the most challenge compared to the Home Page and Wardrobe Page. The page will have three additional tabs compared to the wardrobe page. The recommend tab displays clothing recommendations for the user. The favorites tab displays all the items the user has favorited. The user can favorite items by clicking on an image which when clicked, will provide an icon on the top-left corner for the user to click on. This uses the Favourite.js component that will handle all the relevant actions. The listings tab displays all the listings the user has put up for sale. The user may remove the listing if they wish on this tab. When the user is on the market page, there will be sorting options provided such as the price and most recent listings. Categories of clothing will be on the left-side view navigation panel if the user wants a specific category ([See User Interface Storyboard](#_n7yezsrsocyb)). The market page will also introduce listings from other users. When a user clicks on an item, additional information appears such as the condition, category, contact details, description, etc. At the moment, V-Closet will not implement any sort of payment option. If the user wants to sell or buy a listed item, users will contact each other separately.

Firebase will handle all of our record-keeping in regard to users and their clothing items. The records of users and their authentication methods will be stored in the Authentication Module. Users will be able to authenticate with Gmail or use any other email provider but must use a password. The picture files of the clothing items will be stored in Storage because the files will be too big for JSON. The user's location will be taken and sent to the weather.gov API using JavaScript. The API will return JSON data containing the user’s relevant weather information. We will not have to store this data in Firebase. We will also not store the user’s location for security.

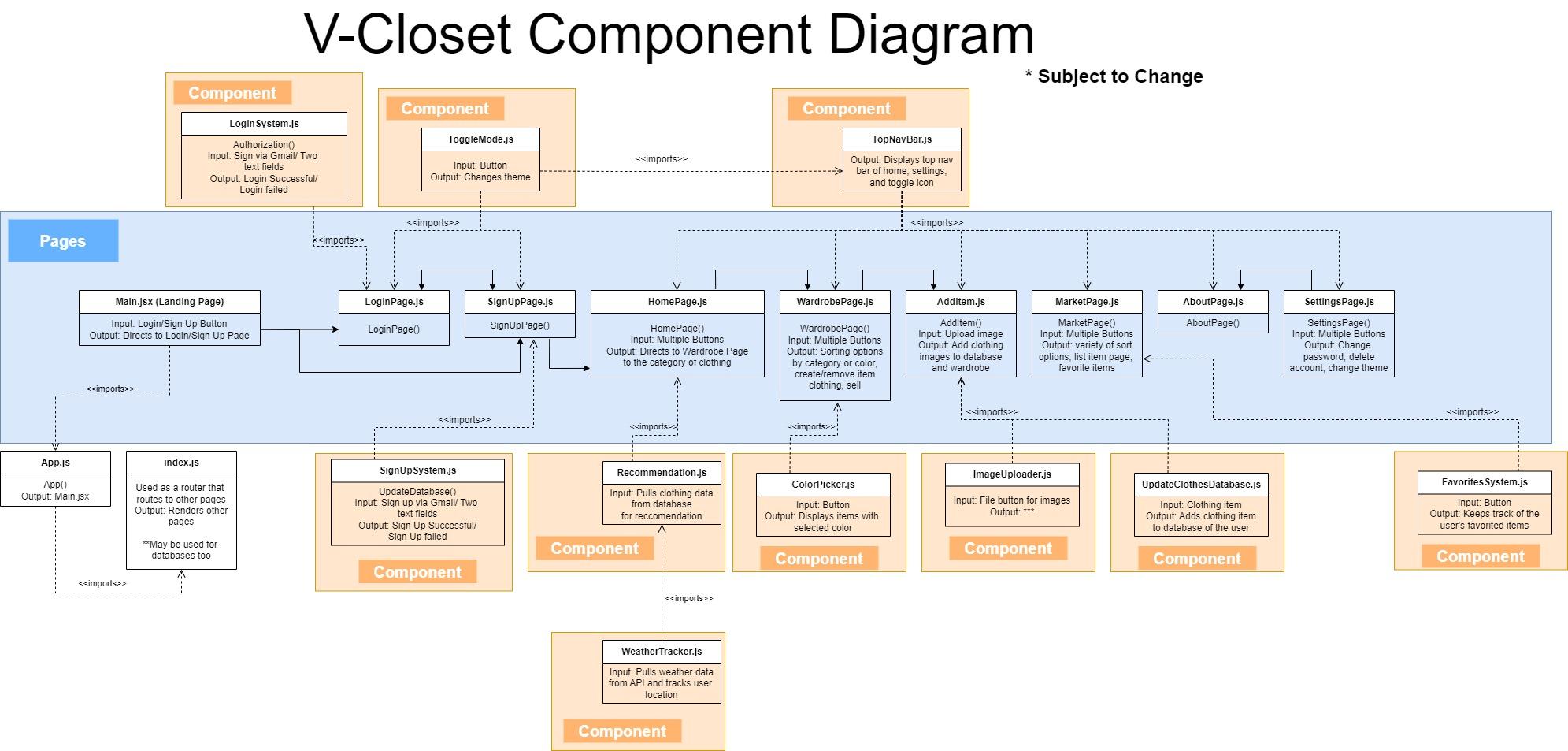
Clothing item details will be stored using JSON in the Firestore database. Clothing items will consist of: the User ID (uid), a link to the image of the item, the category of the item, the name of the item, the color of the item, a description of the item, and three booleans that identify if the item is suitable for warm or cold weather, or if the time is currently listed. Listed items on the marketplace will also be stored in Firestore using JSON. These listed items will be separate from the clothing items since they will have different information. When a listing is created, the listed item will have the same User ID, name, color, and image link. Listed items will additionally have a price, wear, contact, and new description.

Most of our pages and components will be using the data stored in Firebase, so it is crucial the information is easily readable and writable. The structure of all four data systems can be found in the ERD ([See ERD](#_wweiq9untbrw)).

## **Appendix - Block Diagram**

## 

## **Appendix - Component Diagram**



For full view: [ComponentDiagram.jpg](https://drive.google.com/file/d/1RCD0nI1TYUze0p2nvC3FuNRWNGufsG4f/view)

## **Appendix - User Interface Storyboard**

For full view: [UIStoryBoard.jpg](https://drive.google.com/file/d/1ZnBPAU_Id3w14ZGlVCuZN0wX0QKDSgLd/view)

## **Appendix - Entity Relationship Diagram(ERD)**

