**COMSATS University Islamabad,   
Park Road, Chak Shahzad, Islamabad Pakistan**

SOFTWARE DESIGN DESCRIPTION   
(SDD DOCUMENT)

for

**VARDROBE**  
Version 1.0

***By***

**Ali Tahir CIIT/FA17-BSe-017/ISB**

**Ashhad Ahsan Rehman CIIT/FA17-BSe-027/ISB**

***Supervisor*Dr.Syed Sohaib Ali**

***Bachelor of Science in Software Engineering (2017-2021)***

**Contents**

[**1.**](#_1fob9te) **Introduction 1**

[**2.**](#_2et92p0) **Design methodology and software process model 1**

[**3.**](#_tyjcwt) **System overview 1**

[3.1 Architectural design 2](#_1t3h5sf)

[3.2 Process flow/Representation 3](#_4d34og8)

[3.2.1 Login: 3](#_2s8eyo1)

[3.2.2 Search Item: 4](#_17dp8vu)

[3.2.3 See item details: 5](#_3rdcrjn)

[3.2.4 View Cart and Checkout: 5](#_26in1rg)

[3.2.5 AR and VR view: 6](#_lnxbz9)

[3.2.6 Customer support: 9](#_35nkun2)

[**4.**](#_1ksv4uv) **Design models 10**

[4.1 Class Diagram: 10](#_44sinio)

[4.2 Sequence Diagram: 11](#_2jxsxqh)

[4.2.1 Login: 11](#_z337ya)

[4.2.2 Signup: 11](#_3j2qqm3)

[4.2.3 Search Product: 12](#_1y810tw)

[4.2.4 View Product: 13](#_4i7ojhp)

[4.2.5 Checkout: 13](#_2xcytpi)

[4.2.6 Chat: 14](#_1ci93xb)

[**5.**](#_3whwml4) **Data design 15**

[Data dictionary 15](#_2bn6wsx)

[**6.**](#_qsh70q) **Algorithm & Implementation 17**

[**7.**](#_3as4poj) **Human interface design 19**

[7.1. Screen images 21](#_1pxezwc)

[7.2. Screen objects and actions 23](#_49x2ik5)

[23](#_147n2zr)

[**8.**](#_3o7alnk) **Appendix I 24**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

**Application Evaluation History**

|  |  |
| --- | --- |
| **Comments (by committee)**  **\*include the ones given at scope time both in doc and presentation** | **Action Taken** |
|  |  |
|  |  |

**Supervised by**

**<Supervisor’s Name>**

Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Introduction

The idea of VARDROBE is inspired by virtual reality and virtually try on’s. This system will aim to provide an e-commerce platform to the users where they will be able to try out the products which they wish to buy before those products even reach their doorstep through the concept of virtually trying on their product through their devices, furthermore this system also aims to implement augmented reality to provide a Realtime view of the product through the android device of the customer. Modules of this system are:

* Searching of products by keyword and sorting techniques
* Similar product recommendation system and Web Scrapping
* Virtual try-on and AR view for supported products
* Backend Database management and user accounts
* Transaction management, security, and customer support
* Front end data display and category wise management of products

# Design methodology and software process model

The software methodology that we will be using in our project will be XP (Extreme Programming). This methodology focuses on releasing parts of a project whose development is achieved through short development cycles. These features allow us to adapt to the changing needs and functionalities that may be embedded in our system. Our team will be able to focus more on customer satisfaction as this app is intended for the common people to use so its success is based on their positive feedback, plus the development team will be able to focus more on the coding and design phase rather than the extensive documentation. Each activity will be broken down into simpler ones so the goals are easier to understand and through frequent release, there will be ample enough time to review the product and to make amends in it during the testing phase. Furthermore, this methodology allows me and my teammate to work in a pair and it ensures smooth communication between us through which we can collaborate to a greater extent, we can also get extensive customer feedback which is crucial to our project, and if we are not sure about a specific domain related to our project we can later change the functionalities according to our liking. We will take an object-oriented approach towards the development of our project because we cannot just list different functionalities that our product may have in code rather we will divide the functionalities into objects and use them as so in our project to achieve the intended core functions.

# System overview

This system is related to already existing e-commerce platforms, although this specific platform of ours aims at a higher customer satisfaction rate by improving the idea of e-commerce through the introduction of Virtual Try on’s now achievable by the use of machine learning. VARDROBE aims to improve e-commerce by the implementation of Virtual Try on and Augmented Reality. This system will provide a user interface to the users so they can select products based on their likings and then provide a Virtual try on or AR view if the product supports it. All the products and their details plus customer details are stored in the backend database using firebase. Furthermore, Scrapper API is used and machine learning models are deployed to implement AR or VR view. Transactions are made through Skrill or RazorPay.

## Architectural design

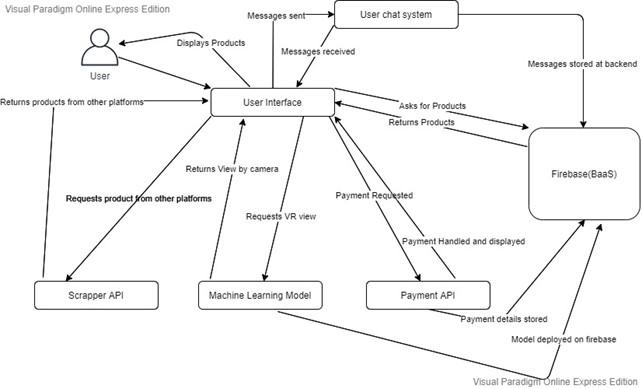
Major components of this system include the backend firebase database service, the front end user interface developed using flutter. Incorporated in the app are the scrapper API and machine learning models. Furthermore, the payment API is also integrated for user transactions. The relationship between these modules/components or subsystems is explained below:

The user interface provides the login. signup and user account functionalities to the user by interacting with firebase at the backend. User requests login, signup, and other user account-related features such as favorite products, address details, and payment info which is kept in firebase, user interface requests these features from firebase and displays it to the user. Also, the front-end user interface displays the products stored in the database and provides chat options with the sellers and a customer support section where users can ask for help from the customer support team. This chat history is maintained in firebase using accounts.

Separate machine learning models are deployed on the cloud for a recommendation system that recommends similar products to the user, this model selects the most similar products to the one the user has bought and then forwards the information to the user interface to display. Products that support Virtual try on and AR view are displayed in a separate section of the user interface, these products request the machine learning model for Virtual try on or AR view when the user prompts for it using the user interface.

User transactions are made using a payment API, user prompts for payment options using the user interface and selects most suitable one, then a request is made to the payment API and transaction info is store in the backend database.

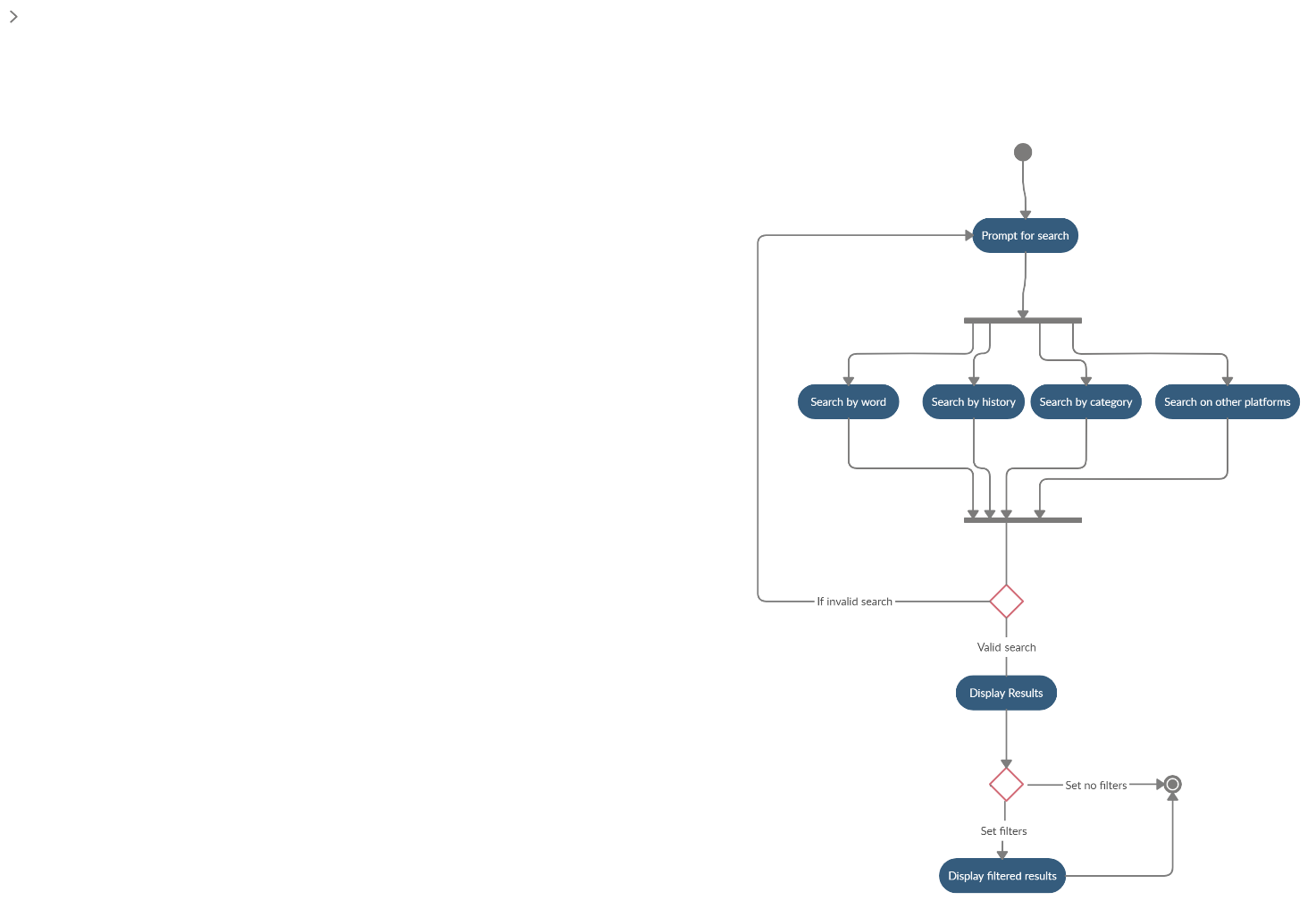
Interactions between the subsystems of VARDROBE are shown below:



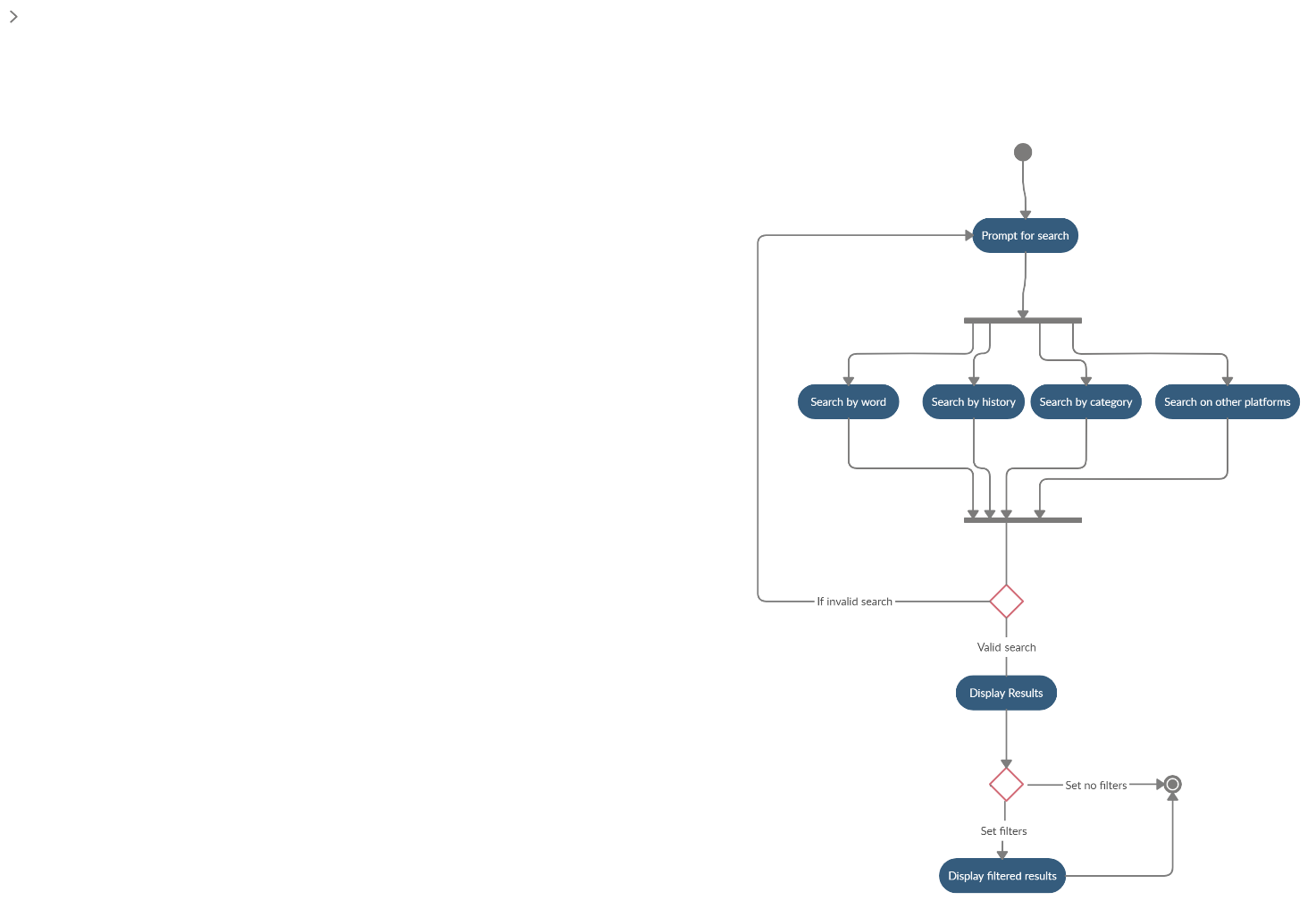
## Process flow/Representation

### Login:

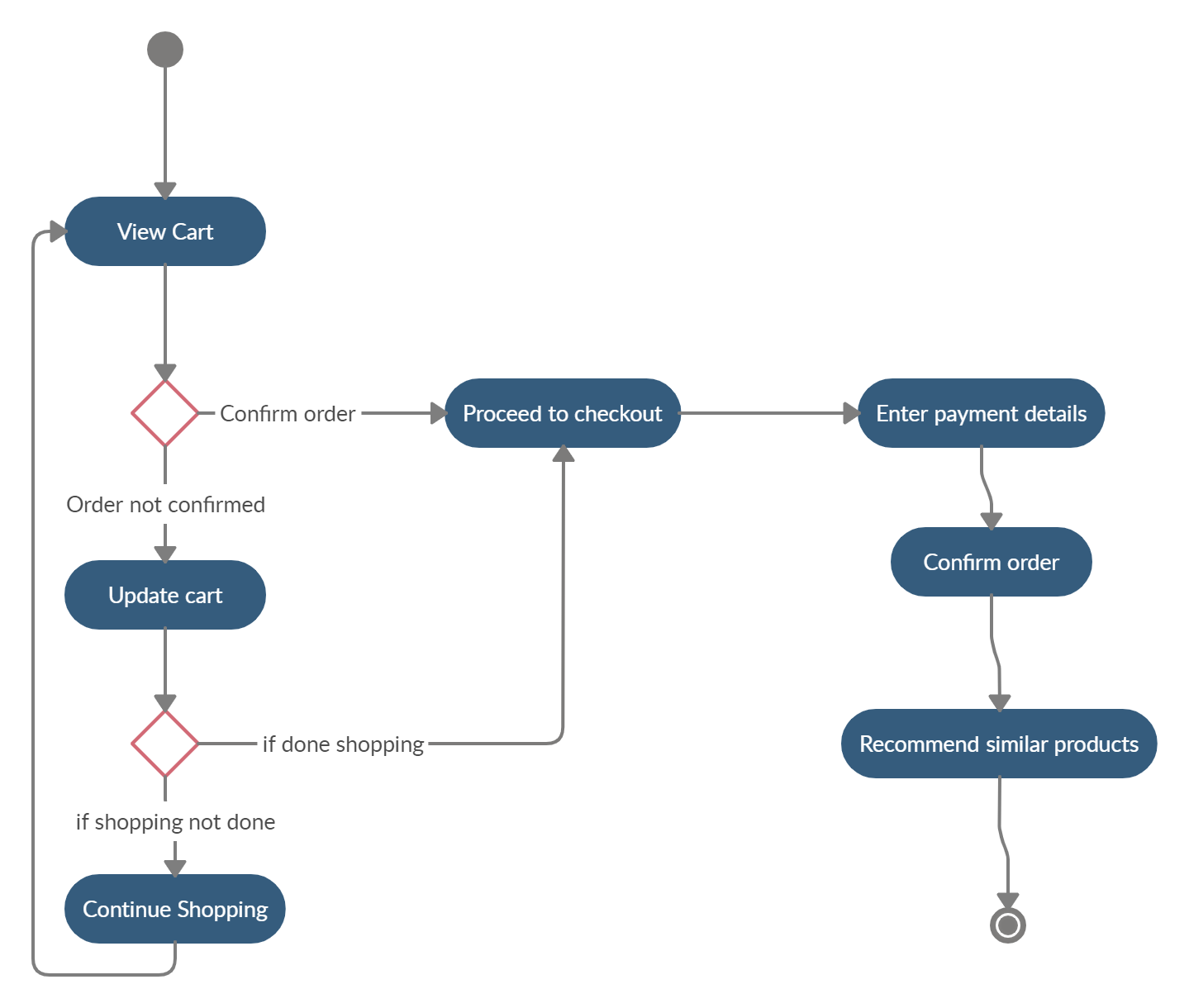
### Search Item:



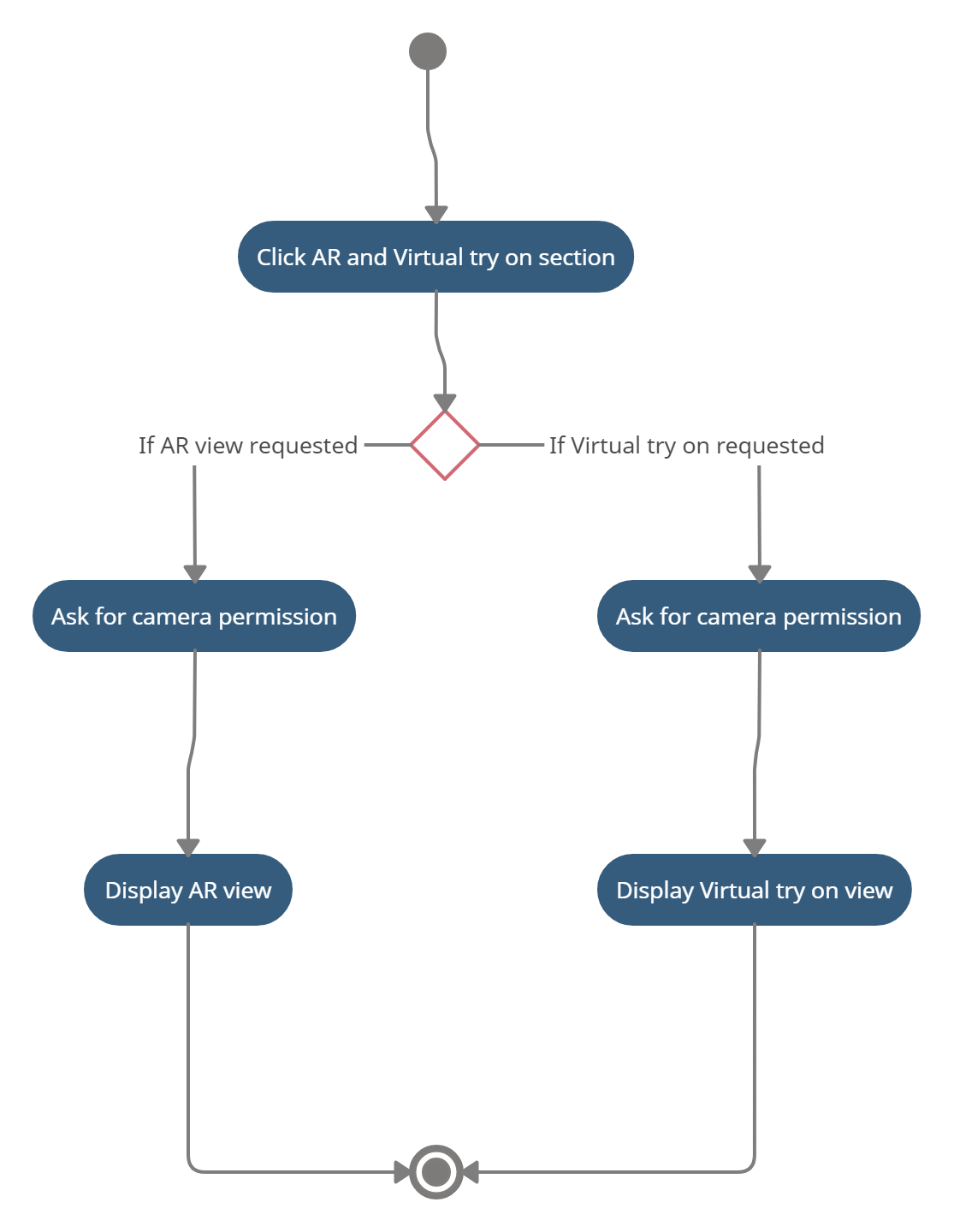
### See item details:



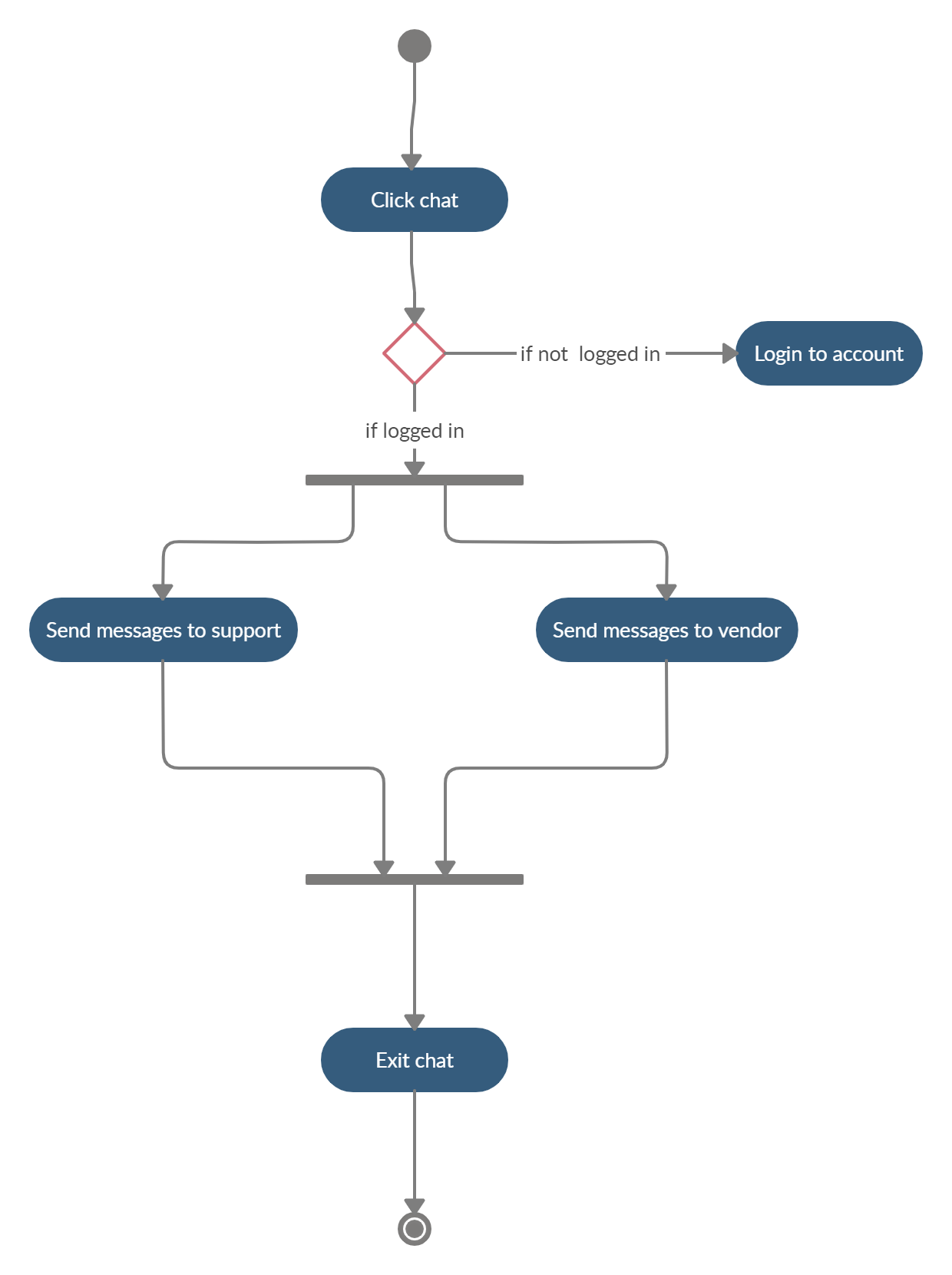
### View Cart and Checkout:



### AR and VR view:

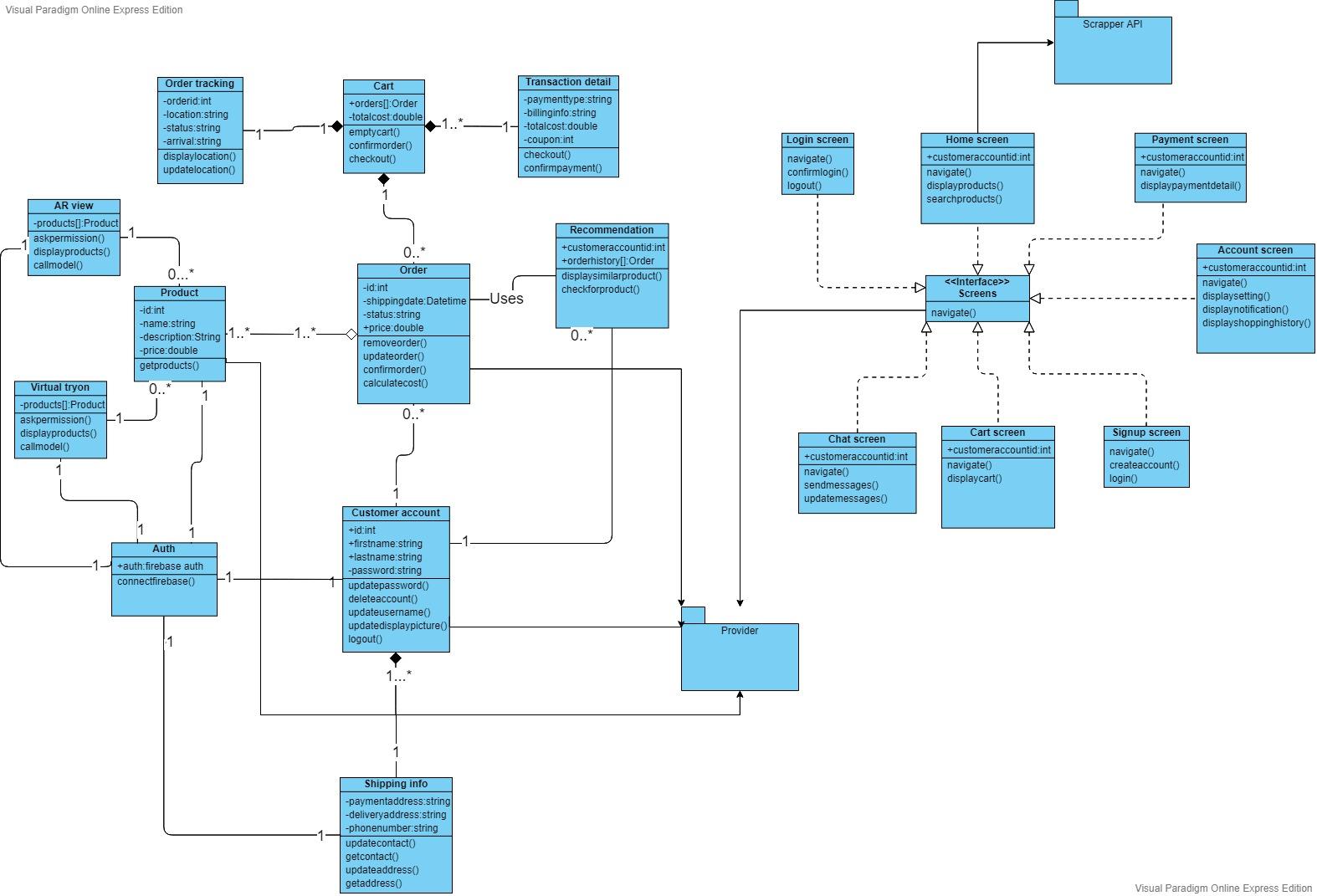


### Customer support:



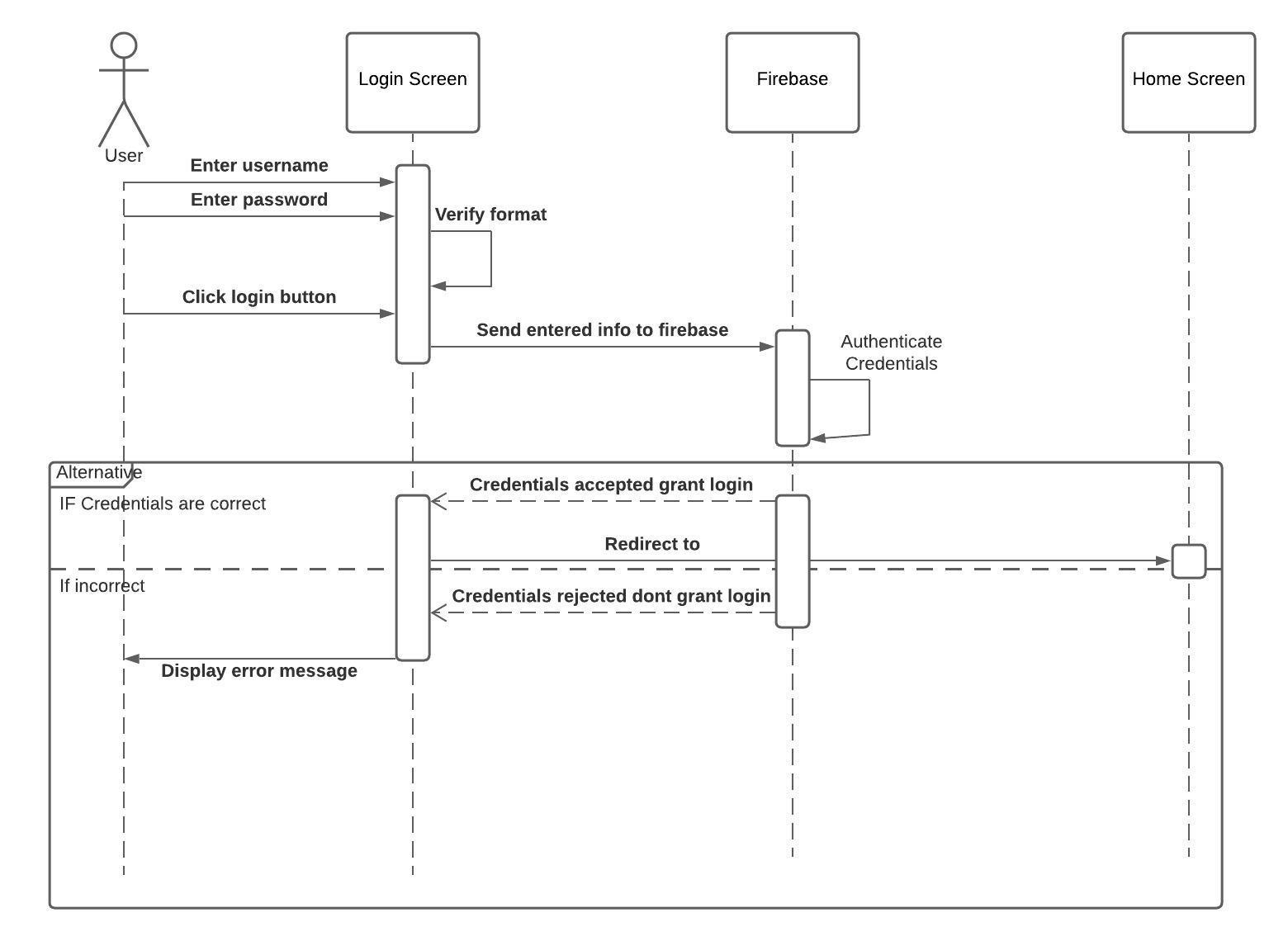
# Design models

## Class Diagram:

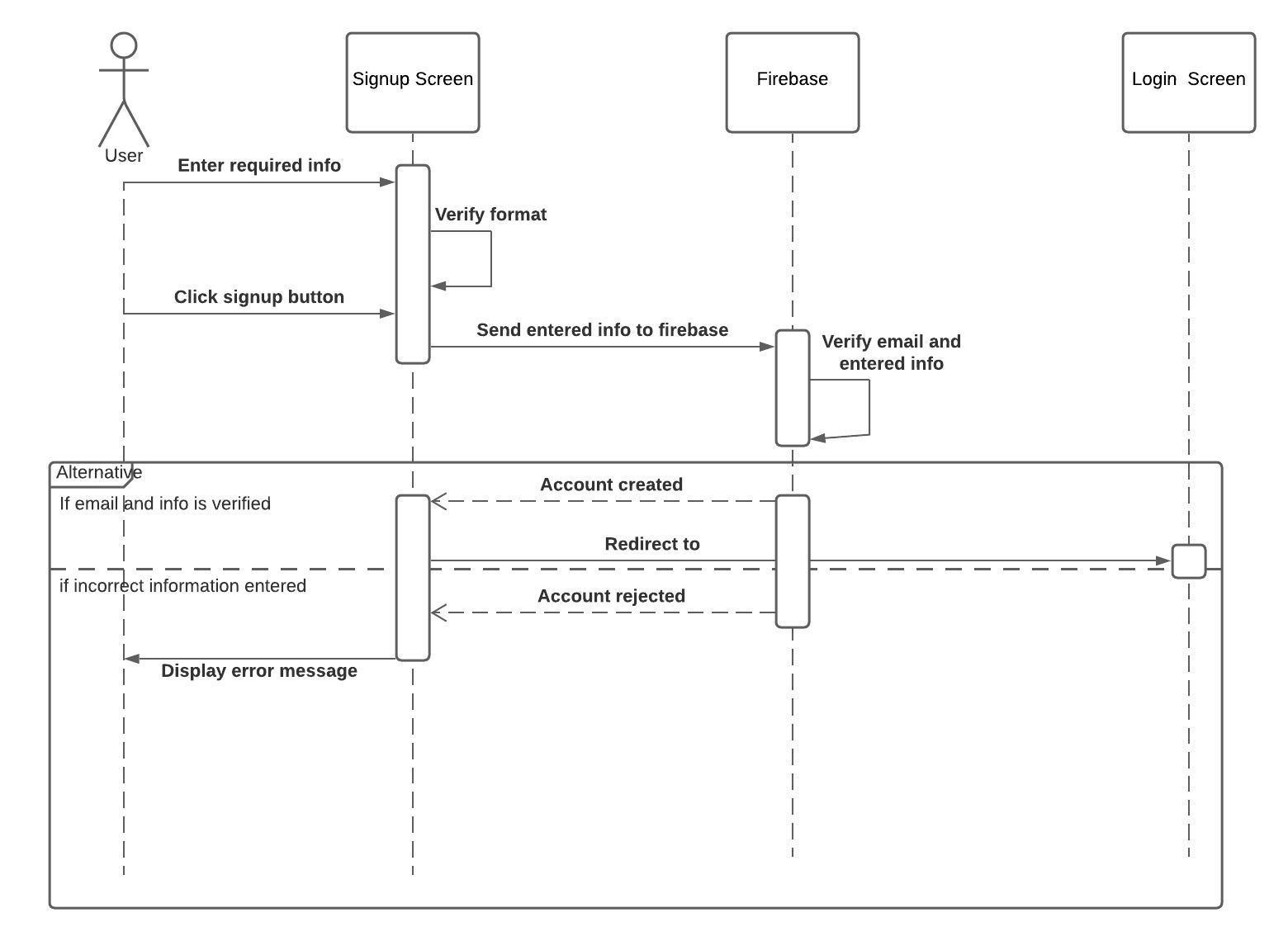


## Sequence Diagram:

### Login:

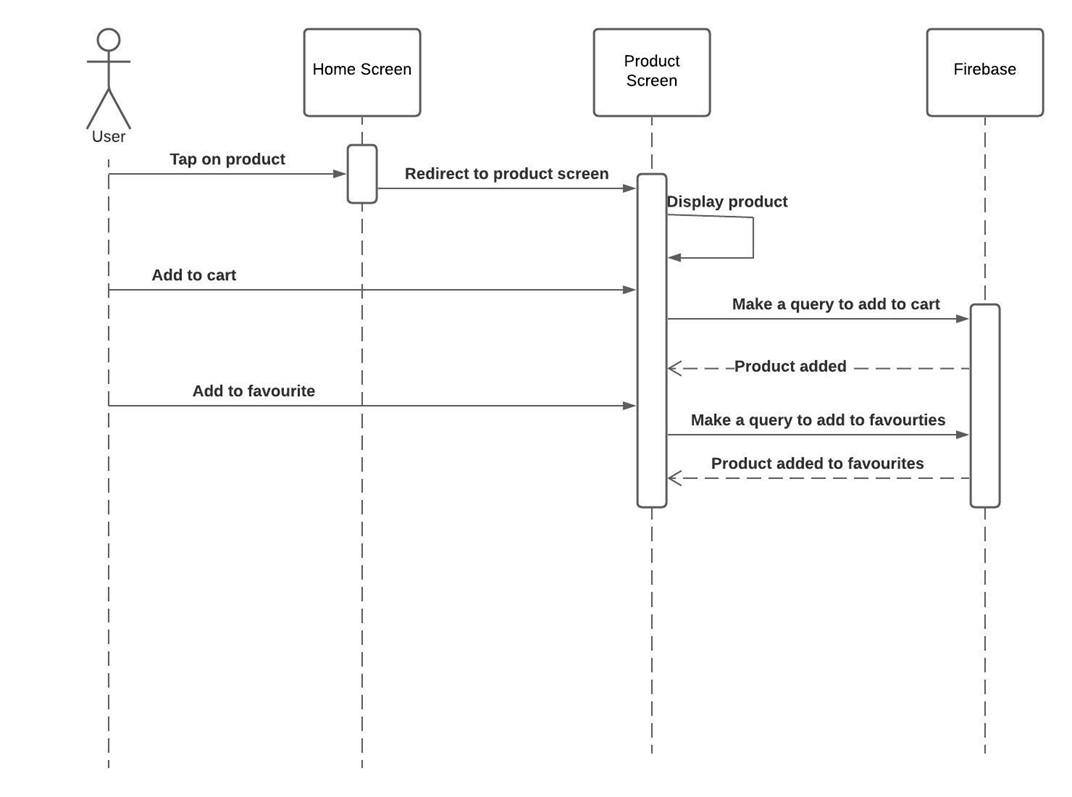


### Signup:

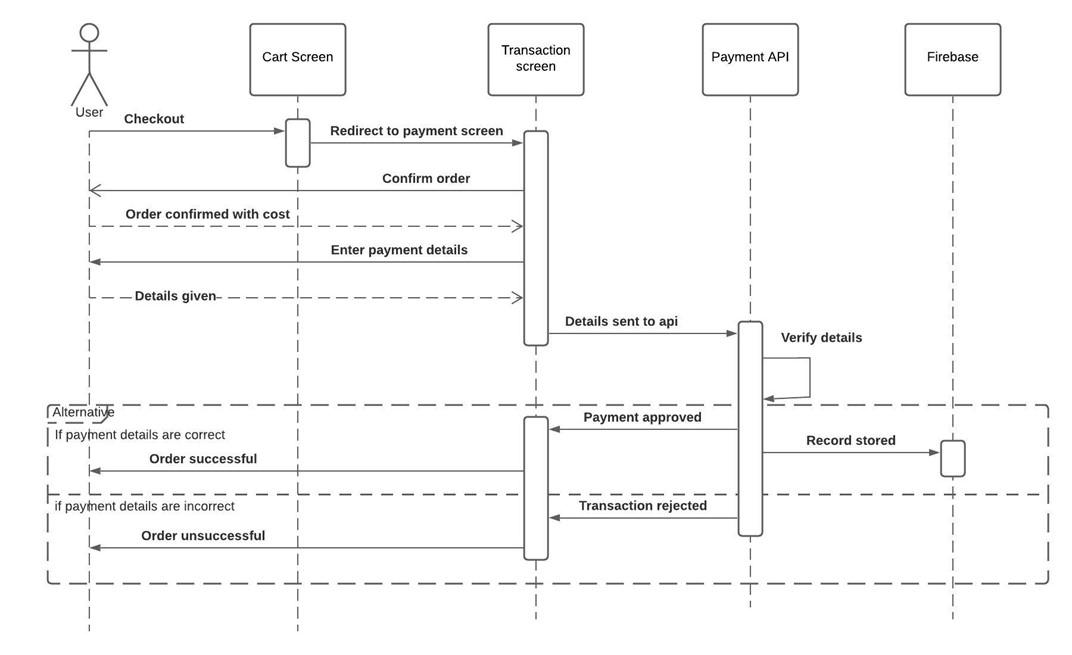


### Search Product:

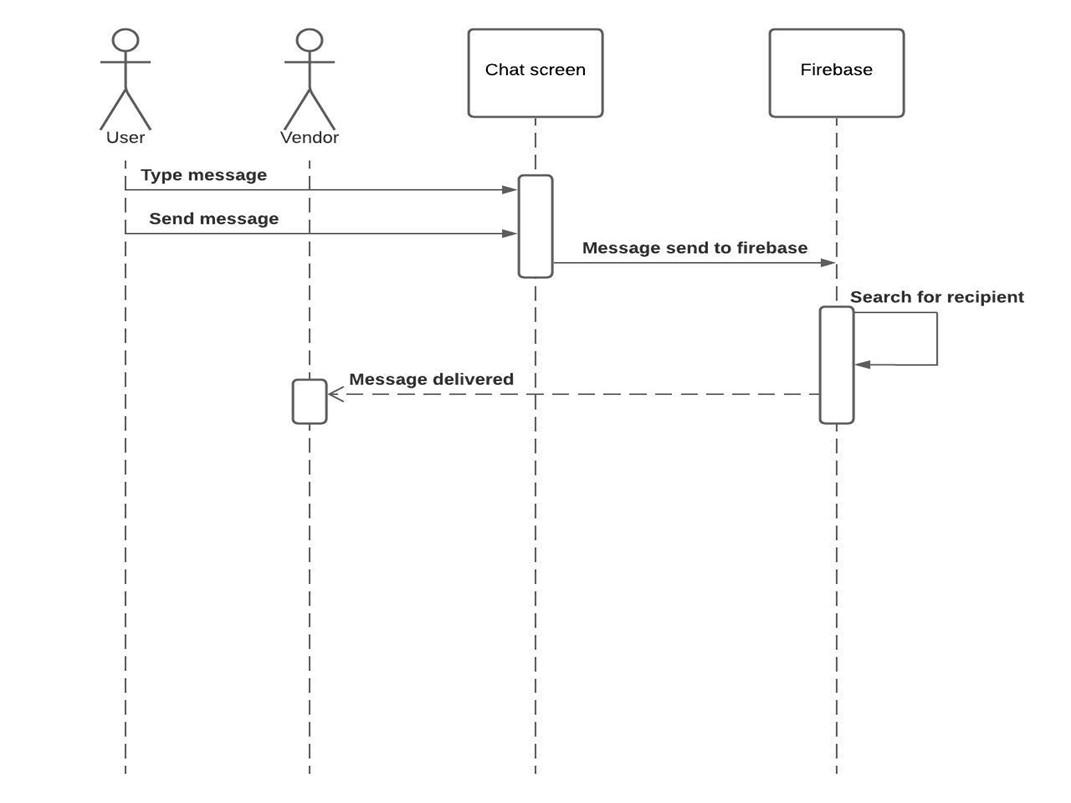
### View Product:

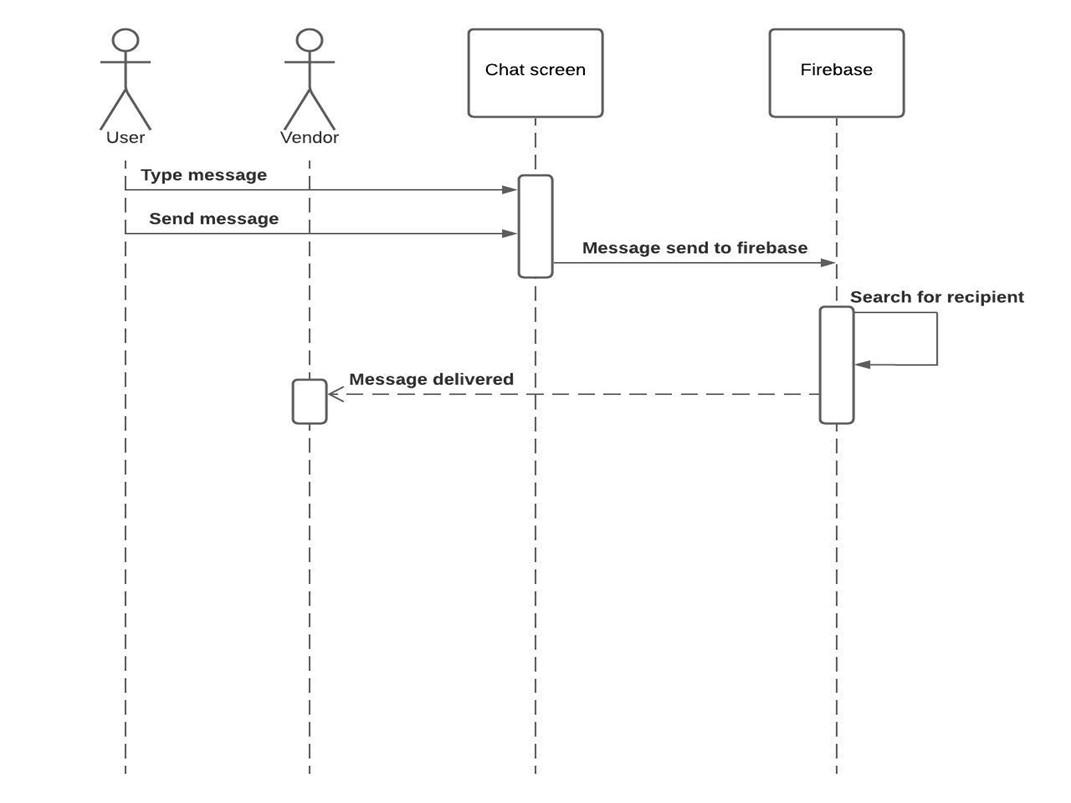


### Checkout:



### Chat:





# Data design

The application is very rich in the terms of the data, the application is dependent upon various data sources, there is versatile kind of data coming from all dimensions from user logins, choices shopping experience, shopping habits to complex data structures like JSON objects to retrieve the data from the external API that would provide access to the recommendation of products, scraped data from a different website, the image data to give in to the model and return an object fitted on to the human pose. The data would be stored in a backend database using Google’s powerful cloud structure of firebase and firestore. The recommendation system, scraper, and Virtual try-on models would reside on the cloud services like Heroku, DataBricks, and AWS Sagemaker and the data would be fetched through a series of API calls to the various platforms to gather the versatile data.

The images would be stored locally on the user's mobile phone because the local storage is always available, the files saved on the local storage are only accessible to the applications.

The user data that includes the text-based data would be stored on the Firebase cloud whereas the images would be stored locally, the interfaces of the application would have the data coming from the cloud API calls, all of the data would be gathered in the application and would be used in the required places throughout the application. The ListArrrays would be used locally to store the data within the application this goes both for Dart and python-based programs, other complex data structures like queue would also be used to process the user request to the server. The dictionary data structure would also be used to store the key-value pair of the data and the results would be retrieved through the use of API.

## Data dictionary

**Customer Support**

This part of the application would be used to store the queries/ complaints of the platform user and a response from the admin of the application would also be stored on the same page. This module would encapsulate the information and send it over to the server until it reaches the required destination and is stored on the receiver end.

**Contact information**

The contact information of the user would also be stored so that any issues, queries, or order confirmation should be made on the contact information provided by the user.

**Delivery Address**

The user needs to give in their residential address details with complete information like house no, street no, postal code, district, and city. The delivery details would be encapsulated so that the external part of the application cannot access or modify the information directly.

**Image data**

This part of the system requires the image to be sent to the GAN to generate an image that fits the product unto the customer and send back the response to show it in the application all of these would be done via an API call to the cloud where the model is stored.

**Favorites**

The part of the application stores the favorite product of the user, the record would be stored on the server under the section where all other user-related data is stored on the server.

**Log in**

This part of the application deals with that when the application is installed on the mobile device the user is prompted to log in to their account that is already registered and holds the record in the system database.

**Order history**

This part of the application deals with keeping track of the order history of the user, this record is kept as a field in the NoSQL database on the cloud server.

**Profile**

Every user needs to have a profile before using the application and its services, the profile information is stored on the database, every user is identified by a unique username and email.

**Recommendations**

The record of the similar product recommendation would be fetched via API stored in the cloud server, python would be used to create a recommender system and the results would be retrieved by making the API call to Heroku or some other cloud service.

**Search**

The application has several search options to select from, the user can search the local products offered by the platform and apply various filtering and sorting techniques to rearrange the data. The user can also search for the products on the other websites within the application the data would be retrieved, through the API call to the scraper script running on the cloud. The scraped data won’t be stored in the firebase database.

**Sign up**

Before using the application and its services the user needs to be registered into the application by providing either their email or associating their vardrobe account with their existing Facebook or Google accounts, the email would be verified to make sure that the user is legitimate.

# Algorithm & Implementation

**Algorithm 1: Login/ Sign up**

**Step 1:** The user would install the application on their mobile devices.

**Step 2:** The user would be prompted to sign up if their account does not exist already, basic information would be gathered and the data would be forwarded to the database for safekeeping and later use, the user would be redirected to the login page after that.

**Step 3:** After verification from the database the user would be able to see and use the services offered by VARDROBE.

**Algorithm 2: Search Products on the local system**

**Step 1:** The query is retrieved from the search bar from the application.

**Step 2:** It would be forwarded to the firebase data.

**Step 3:** The database would return the relevant data against the required results

**Step 4:** The data would be displayed to the screen for the user.

**Algorithm 3: Search products on other websites**

**Step 1:** The query is retrieved from the search bar of the application.

**Step 2:** The query would be forwarded to the scraper running in the Heroku cloud through API call.

**Step 3:** An async request to the server would be made.

**Step 4:**  The server would return the data from the API as a response to the HTTP get request.

**Step 5:** The data would be displayed on the screen directly without storing it in the database by a JSON object.

**Algorithm 4: Filtering the results**

**Step 1:** The user-selected filter would be fetched through the selection menu.

**Step 2:** The required filters would be applied to the records by the use of a firebase realtime database function call.

**Step 3:** The results would be returned and shown on the screen.

**Algorithm 5: Sorting the results**

**Step 1:** The user-selected sorting would be fetched through the selection menu.

**Step 2:** The required sorting would be applied to the records by the use of a firebase realtime database function call.

**Step 3:** The results would be returned by the API call and shown on the screen.

**Algorithm 6: Products Recommendation**

**Step 1:** The available data for the products are gathered and stored by fetching it from the database.

**Step 2:** The machine learning model trained to predict the recommendations that need to be made to the user would be deployed on the cloud and the API call would be created to fetch the recommended products.

**Step 3:** In case there are no ratings by the user, hence the classical cold start problem, the random products would be recommended to the user.

**Algorithm 7: Virtual Try-On**

The Virtual try-on module is based on GAN, GAN consists of two Deep Learning model competing against each other, one input is from the real sample and the other is from A generative model that generates an output in our case an image, by looking at the training data but a noise (bias) is added to its generation and a judge model called discriminator finalizes that which model predictions are true or not by classifying whether the input is generated is from the real dataset or It is from the Generative model by the help of sigmoid activation function, so basically AI vs AI, the steps included in a GAN is listed below

**Step 1:** The discriminator is trained on the real data and freezes the generator, in this phase only forward pass is done and no backpropagation (that is used to reduce the loss) is done, the discriminator is trained with real data.

**Step 2:** The generator is trained to produce fake images and the generator is frozen, the results from step 1 are used to feed into the generator to make it better from the previous state.

**Step 3:** Train the discriminator again on the fake data generated by the generator, to train it to classify the data generated as fake.

**Step 4:** Agin trains the generator by giving in the output from the discriminator, thus training the generator to fool the discriminator.

Now the model that would be used to generate the virtual try-on images is discussed below.

**Step 1:** The pose estimator is used to estimate the pose of the human being , this would be done by the use of Open Pose Look into the person that is caffemodel by which the features of the person that are

* Background
* Hat
* Hair
* Glove
* Sunglasses
* Upper-clothes
* Dress
* Coat
* Socks
* Pants
* JumpSuits
* Scarf
* Skirt
* Face
* Left-arm
* Right-arm
* Left-leg
* Right-leg
* Left-shoe
* Right-shoe

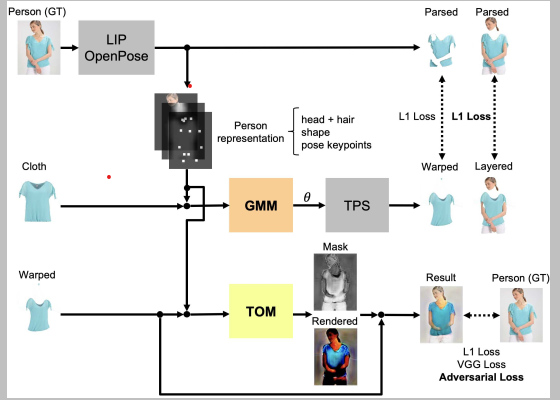
are extracted in the form of a matrix, these features include the head, hair shape and pose key points around the body, the features are parsed to get the only cloth image and the face with the cloth image to get both aspects of the image.

**Step 2:** The human pose along with the image of cloth presented as a matrix is given to the GMM (Geometric Matching module) that find outs the geometric points to which the piece of clothing would be fitted on depending upon the features generated by the Open Pose, the features are passed to TPS (Thin Plate Spline) that smooths the image and produces the wrapped image to an article of clothing.

**Step 3:** The L1 regularized loss that prevents overfitting is used as a loss minimizer is back propagated to the piece of clothing image to separate the human body from the clothing that improves wrapping

**Step 4:** The features also forwarded to the TOM (Try on module) the adversarial loss   
from the GAN is also added to the TOM to make the predictions better that first marks the output to the grayscale that reduces the amount of data to work on this by turning off the color so that fewer resources are required for the computation, the grayscale image is rendered to produce the output result that is a new article of clothing fitted to the human pose.

**Step 5:** The data is augmented by random horizontal flipping.



# Human interface design

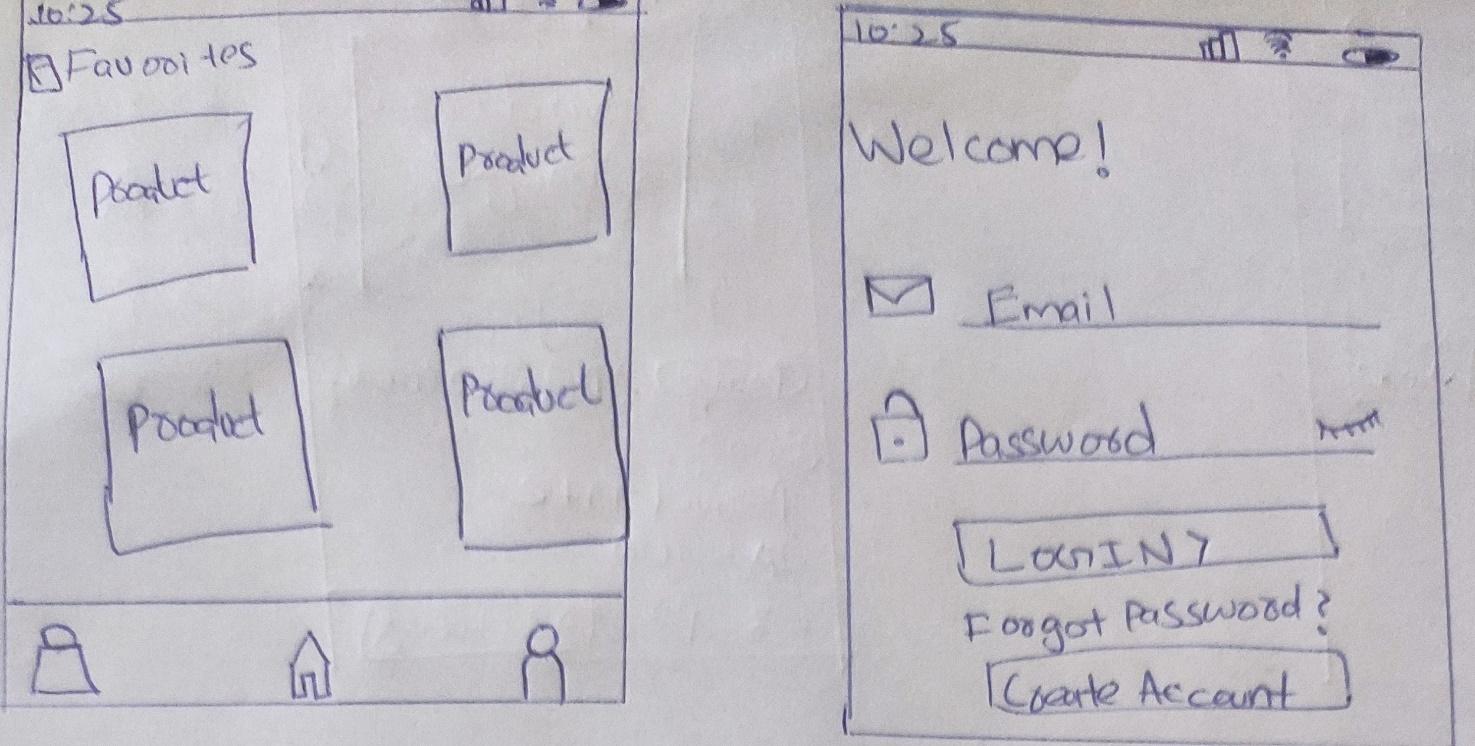
Every application should follow some standard guidelines so that the user experience can be as smooth as possible, just like that VARDROBE focuses on the ease of the user by providing them with easy to use and navigate UI in which they do not feel like that they are stuck at using some feature.

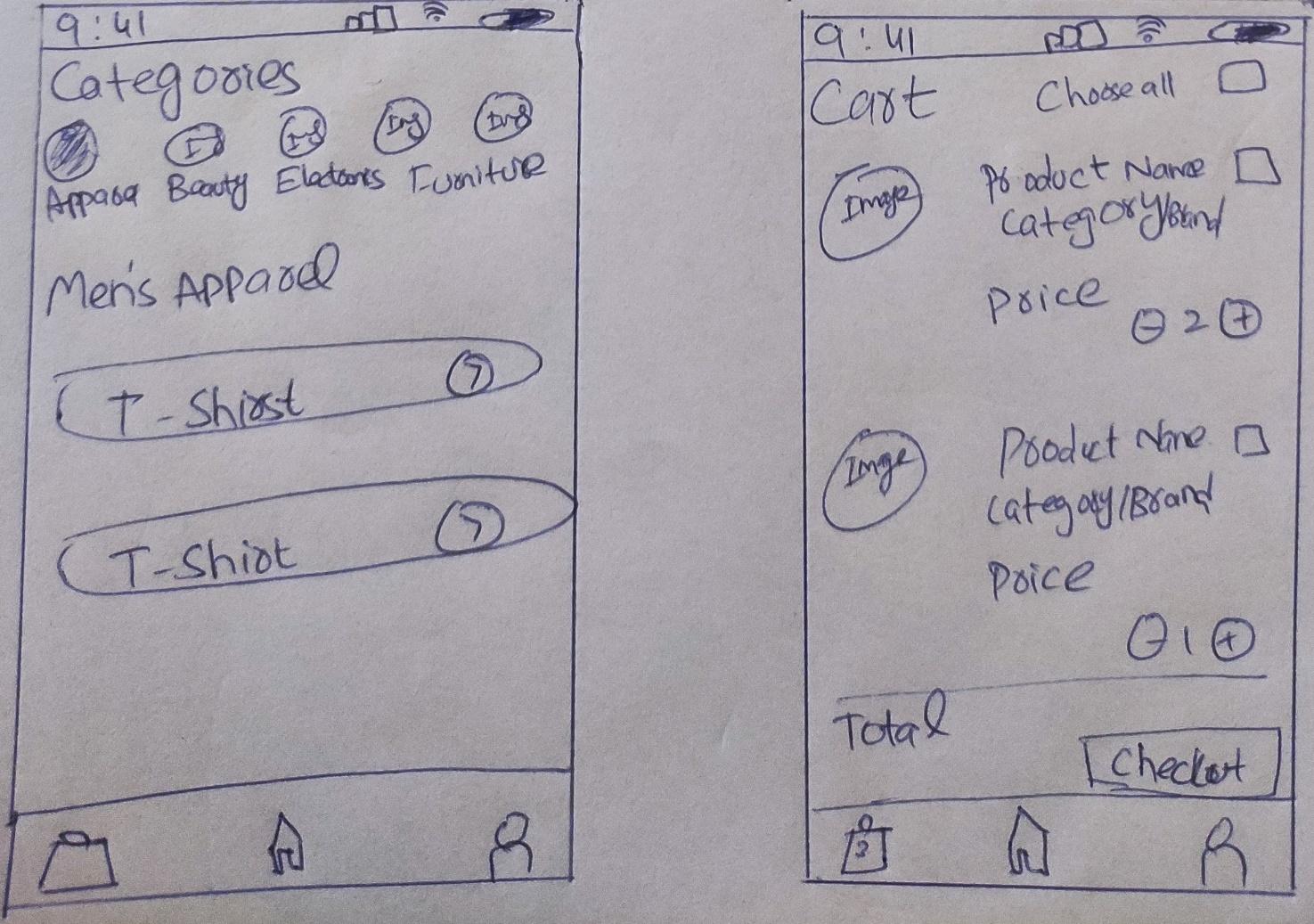
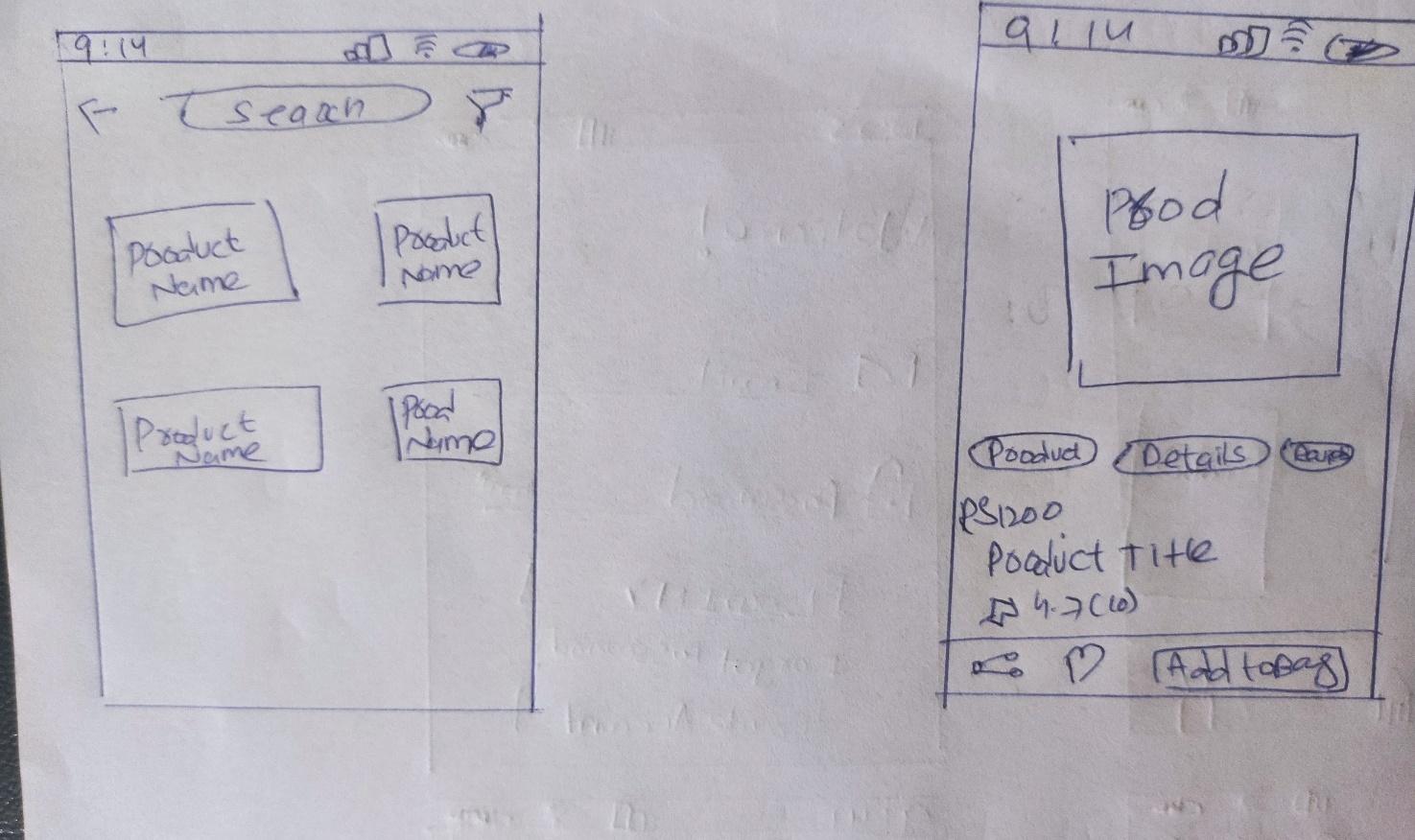
Few of the guidelines in this regard that needs to be followed are

* The interface should be intuitive.
* The icons used should be self-explanatory that by looking at the icon the user can judge what they would do, so standard icons that are used in the other application should be used.
* The font size, color, and family should not be too overwhelming.
* The UI should be consistent throughout the application.
* The color scheme should match the dark theme of the application.

VARDROBE comes with a built-in dark mode so to cause less strain on the user's eyes, the future updates may include the light mode as well, right now our aim is to fit with the trends of the moving world that is going towards the dark mode of the application to preserve the eyes of the

## Screen images





## Screen objects and actions

|  |  |
| --- | --- |
| **Screen object** | **Action** |
|  | Users can view their profile, order history, and other related information by clicking on this icon. |
|  | Users can go to the homepage of the application by clicking on the home button. |
|  | Users can navigate directly to their cart by clicking on the cart button. |
|  | Users can click these plus and minus buttons to increase or decrease the number of products in their cart. |
|  | Users can click this check button to add the item to their checkout total, when the item is clicked the price would be added and it would be visible below the total label. |
|  | Users can search for a product by entering their query in the search bar. |
|  | Users can see various filters and sorting options by clicking on the funnel icon on the screen. |
|  | Users can share the link to the product by pressing the share button. |
|  | Users can add items to their favorites by clicking the heart icon on the product description screen. |

# Appendix I

* How to design using UML (OOP): For guidance please follow the instructions mentioned in the link: http://agilemodeling.com/artifacts/
* How and when to design ER diagrams: For guidance please follow the instructions mentioned in the link:

<http://people.inf.elte.hu/nikovits/DB2/Ullman_The_Complete_Book.pdf>

* Data flow diagrams: For guidance please follow the instructions mentioned in the link and book:
  + http://www.agilemodeling.com/artifacts/dataFlowDiagram.htm
  + Software Engineering –A Practitioner’s Approach by Roger Pressman
* Architecture diagram: For guidance please follow the instructions mentioned in the link and book:
  + Ian Sommerville – Software Engineering 9th Edition– Chapter 6