Product Design

Team 32 - Vedanivas, Srihitha, Prakul, Arnav Negi

Introduction

A Project Design Document is a document that outlines the design and scope of a software project. It is an essential document that guides the development team throughout the project and helps the client to understand the system and its design.
The purpose of this document is to provide a detailed description of the system's functionality, design, and architecture.It serves as a reference document throughout the project's lifecycle.It provides a clear understanding of the project's goals, features, and scope, helping to ensure that everyone involved in the project has a shared understanding of what needs to be developed.
This is a valuable document for future reference, as it provides a detailed record of the system's design and functionality.

System Overview

This is a MERN based web application designed to assist dairy farmers in their bookkeeping activities. It provides a digital database to store and manage data related to their products and customers. The software system aims to replace the traditional paper or excel sheets with a user-friendly interface that simplifies data entry and analysis.

The software system provides the following functionality:

1. Inventory Management - The dairy farmers can manage their product inventory, track the quantity of each product produced, and record the price of each product.
2. Subscriptions Management - The farmers can manage their subscriptions and generate delivery sheets for each day. Customers can subscribe to products and edit their subscriptions later on.
3. Shopping Cart - This feature allows customers to browse and select products they want to subscribe to, add them to their cart, edit them, and proceed to checkout.

Some potential real-life scenarios include:

1. A dairy farm that produces different types of dairy products such as cheese, and butter can use the software system to manage their inventory of different products and track the production. This will help them optimise their product mix.
2. A dairy farmer who wants to sell their products online can use the software system to create an online store that integrates with their inventory and customer data. This will help them expand their customer base and increase their sales revenue.

3. A dairy farm that offers a subscription service to customers who receive regular deliveries of milk and other dairy products. They can use the software system to manage their subscription service, including customer subscriptions, delivery schedules, and billing. Customers can subscribe to the service and select their preferred delivery schedule and products. The software system will automatically generate invoices and process payments based on the customer's subscription. The dairy farm can use the system to track subscription revenue and customer churn, and make adjustments to their subscription service based on customer feedback.

In all of these scenarios, the software system would solve the problem of manual record-keeping, which can be time-consuming, error-prone, and difficult to manage. The software system provides a user-friendly interface that simplifies data entry and analysis, improves operational efficiency, and helps dairy farmers make better decisions based on accurate data.

Design Overview

Architectural Design

User Authentication

The app is used by two types of users: customers and vendors. Users can register for a specific role, and then log in through the login portal to access dashboards tailored to their role.

Inventory Management

Vendors can add the names, images, unit prices, and discounted prices (if any) for all the products they sell. This information is visible to customers.

Vendors also have the option to offer discounts on the items they sell.

Vendors can view, add, edit, delete, activate, and deactivate items at any time using the edit options.

Customers are able to view all items from all vendors.

Any changes made to products are reflected on the customer side as well.

Subscription management

The customer can choose their preferred subscription type for the item and make edits before the cut-off time.

The vendor can generate a delivery sheet listing all the deliveries they need to make the next day after the cut-off time.

Shopping Cart

Each item has an "add" button. You can add multiple items to the cart and then edit them.

When you click the "view cart" button, you will see the name, unit price, quantity for each item, and the total price of the cart.

You can edit the shopping cart by changing the quantity of an item, removing products, or modifying the subscription.

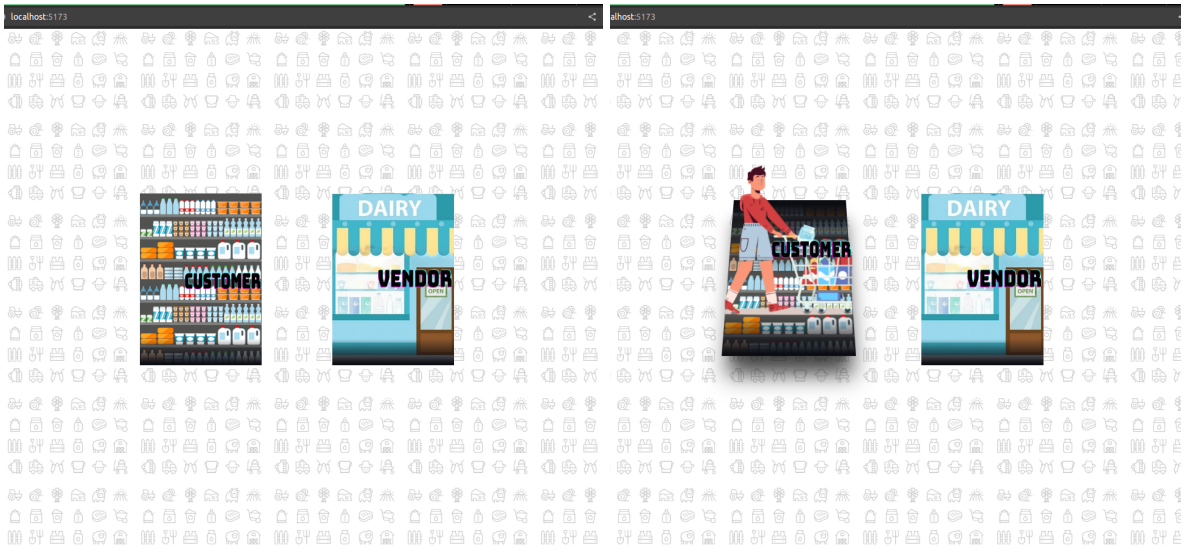
Clicking on the "add promo code" button allows you to apply discounts and other promotional features.

Inside the shopping cart, there is a "continue shopping" button that takes you back to the home page to add more items to the cart.

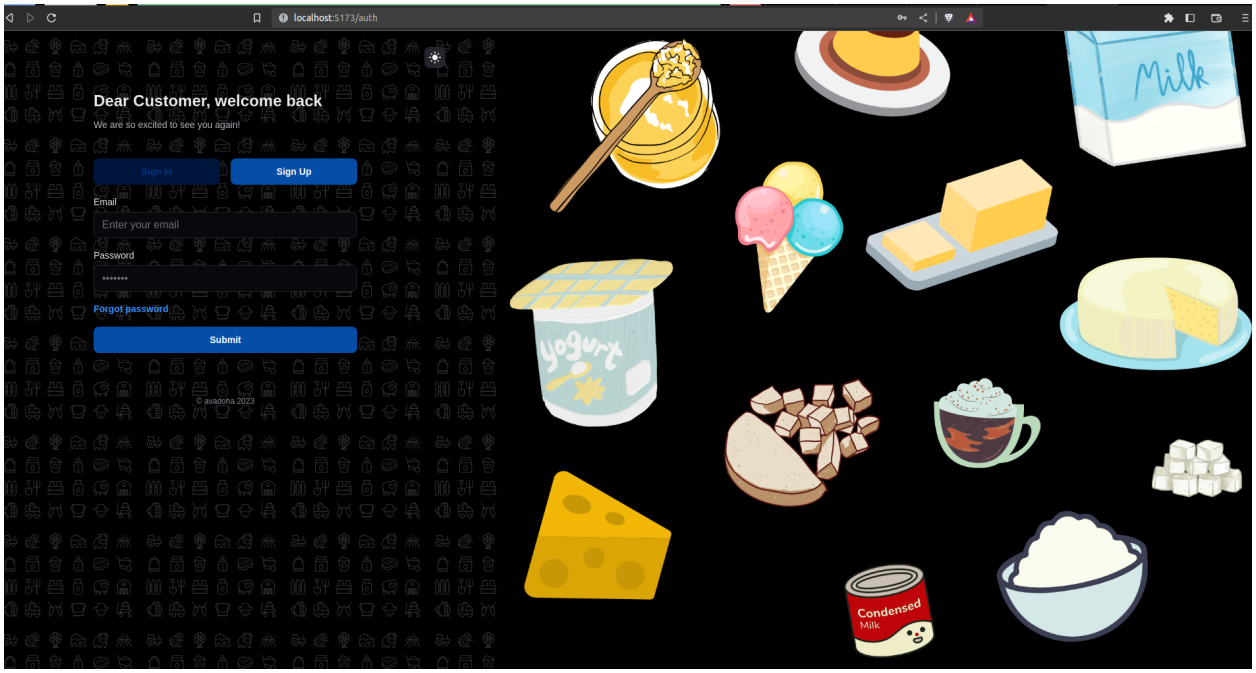
Clicking on the "checkout" button takes you to the checkout page, where you need to add your shipping address, landmark, and notes for any specific delivery details if needed.

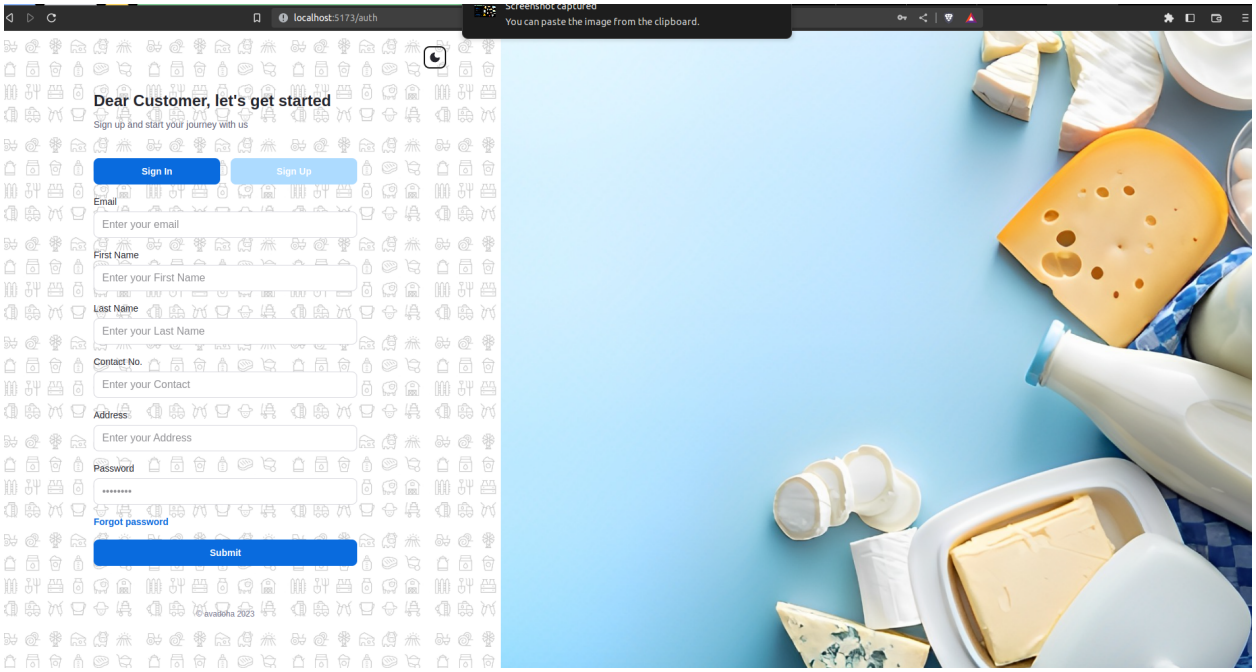
System Interfaces

User Interface

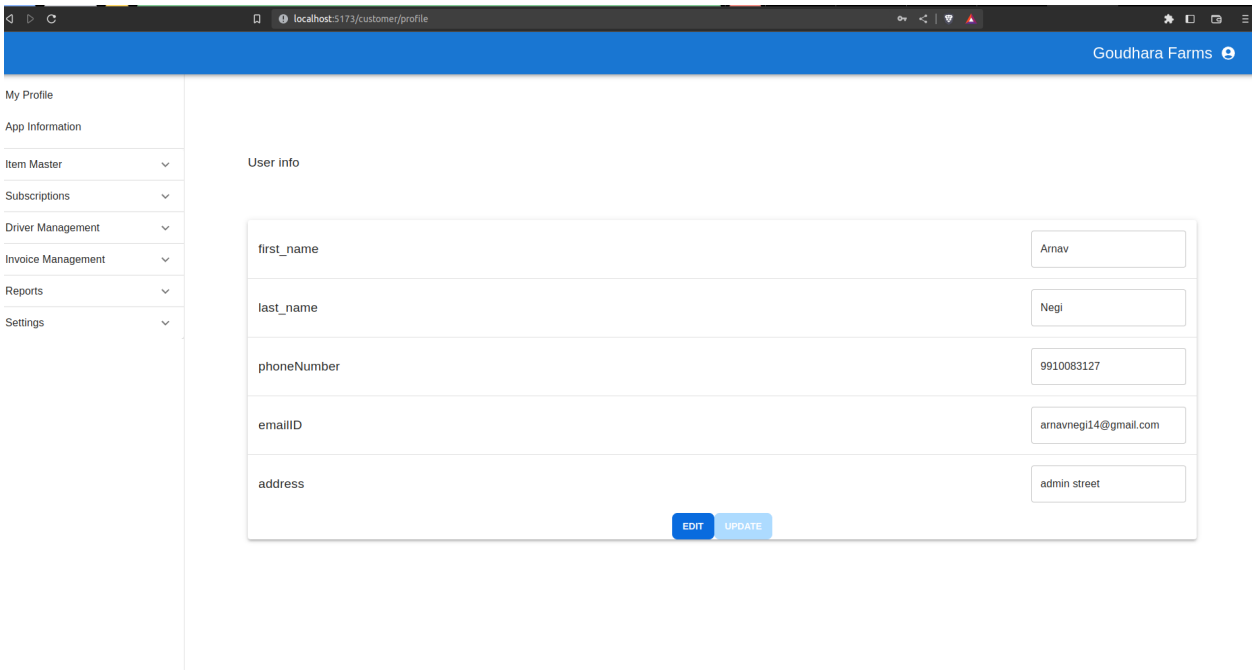


- This is the user type selection screen. The user can choose from either vendor or customer. This acts as a splash screen for the web-app and has an animation portraying the two types of users.

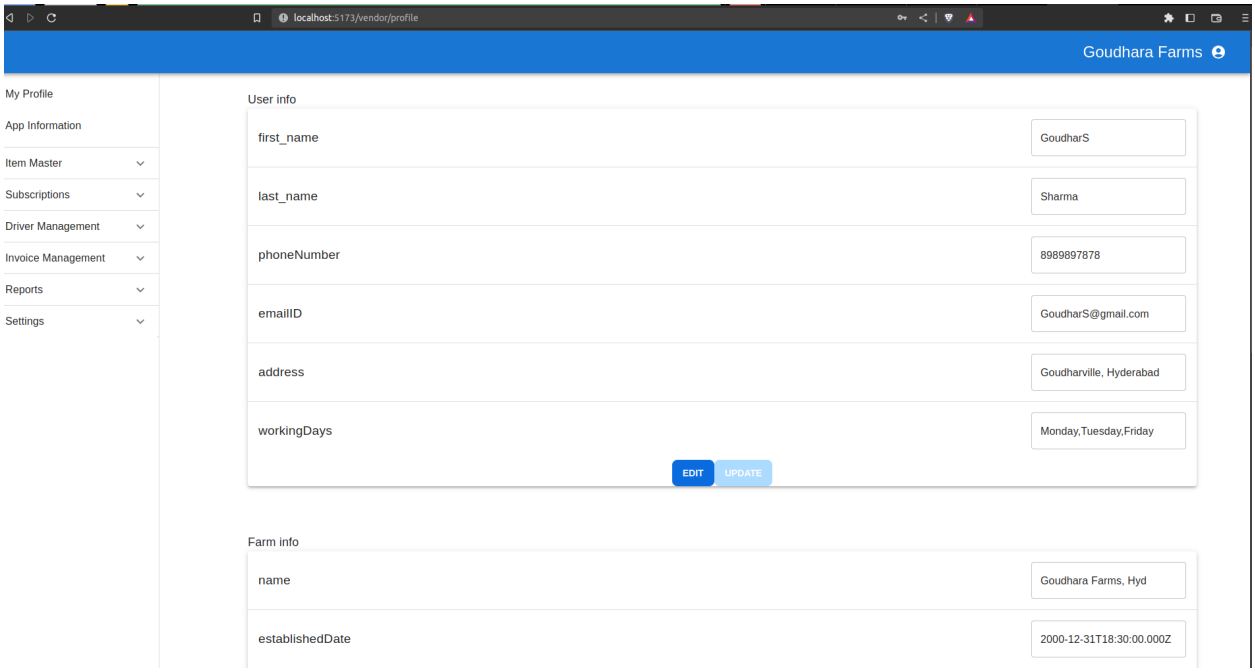


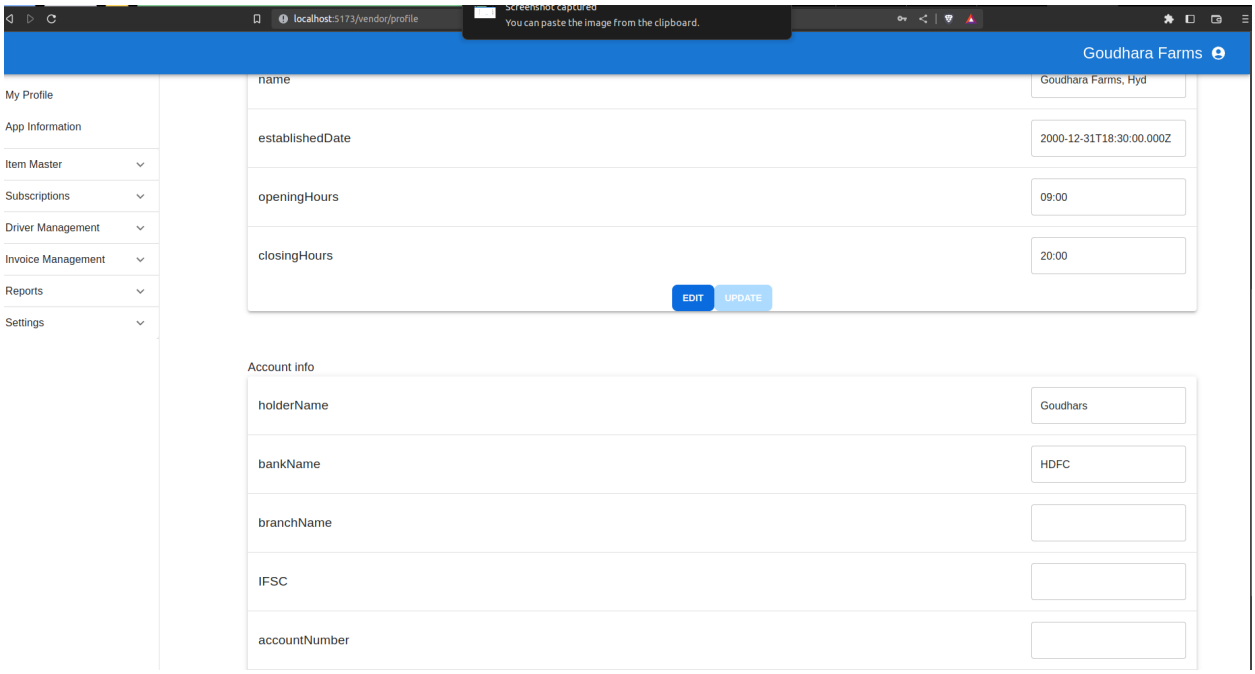


- The login and registration screens, in both dark mode and light mode.
- There are tabs to toggle between login and signup. The type of account depends on whether you chose vendor or customer during the user select screen.

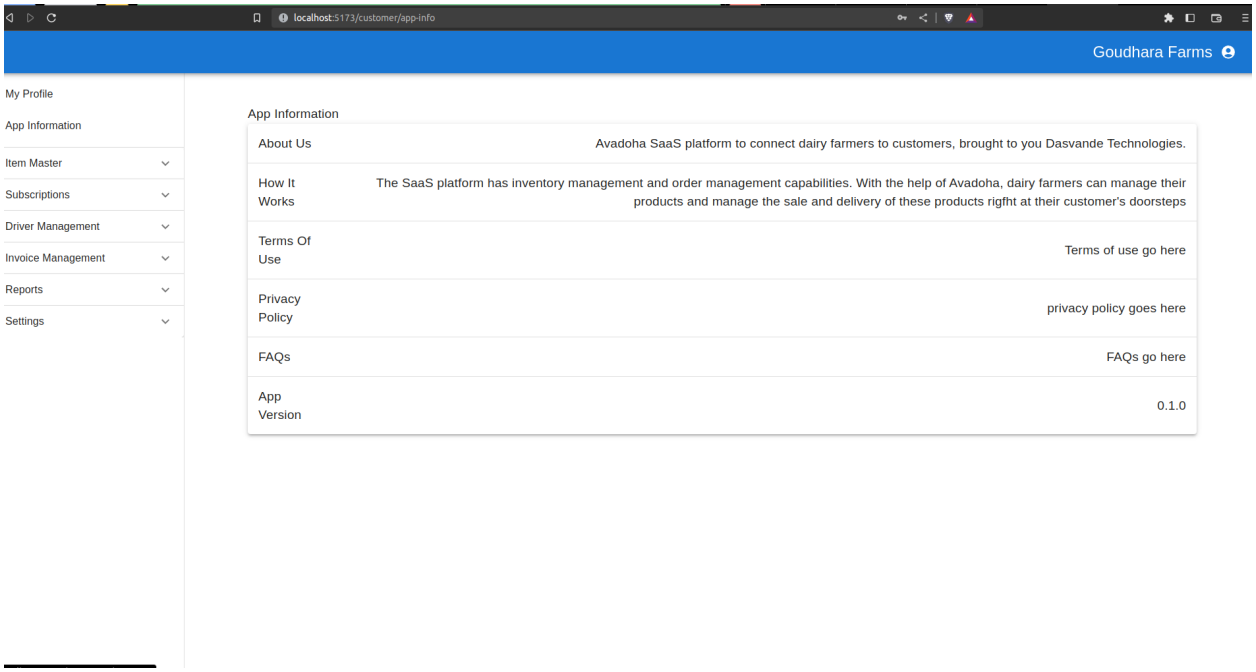


- The dashboard of a customer.
- The current screen is a customer detail screen with an edit feature.
- The top navbar has a profile logo with a logout option.





- Vendor detail screen with edit feature. This screen has more information than the customer screen including farm and bank account information.



- App-info screen with details about the web-app eg. version no., FAQs, privacy policy, etc.

APIs

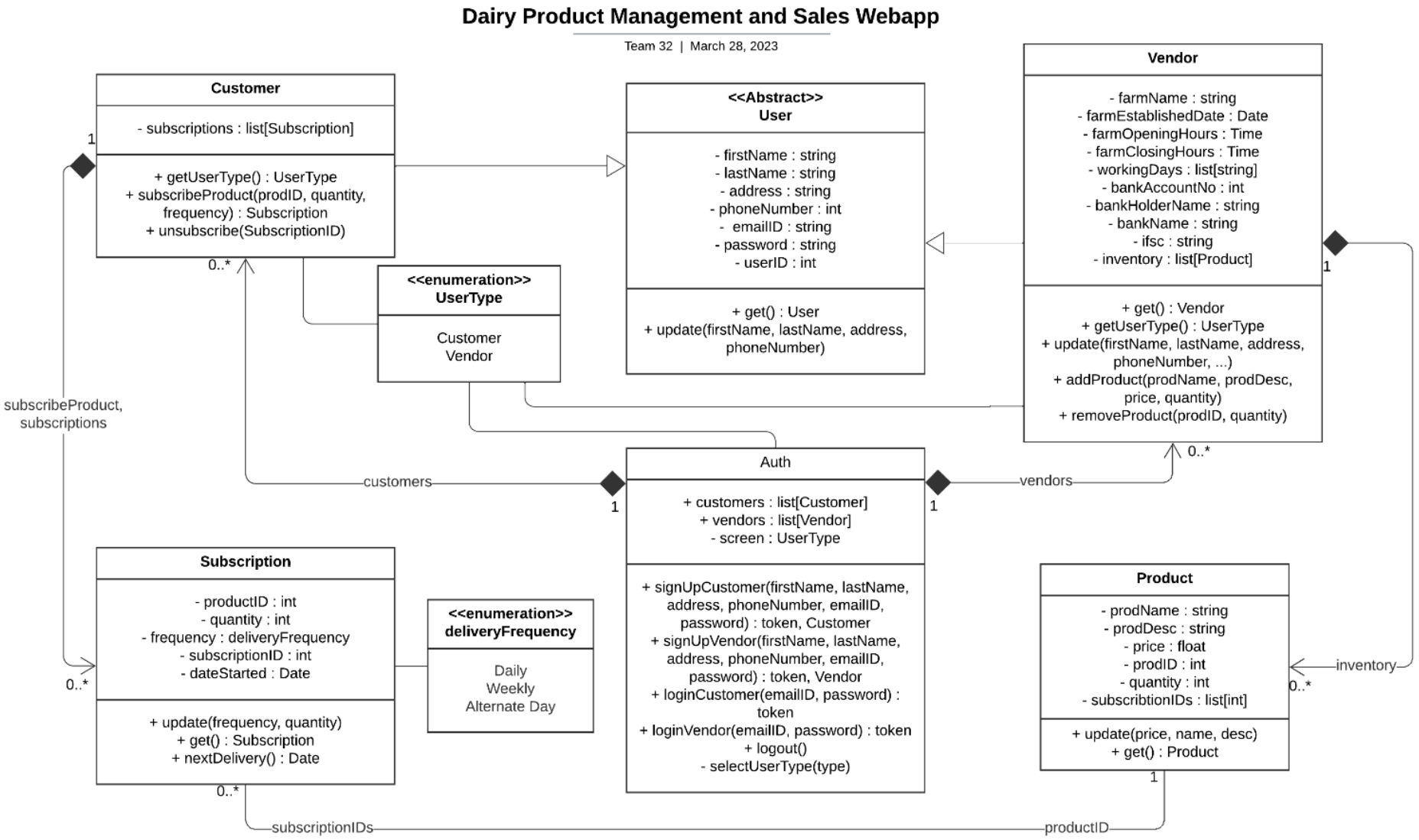
The following API are the ones that will be exposed to enable the users to interact with this web-app:

- `**General**` purpose API - Handles common functionalities like getting the type of user etc. It is responsible for calls that are not dependent on who the user is.

- `**Vendor**` API - Handles calls made by different vendors. It involves calls to manage the inventory, track subscriptions of customers and know what to deliver.

- `**Customer**` API - Handles calls made by the customers. These calls are used to browse the available vendors and their respective products, and then choose and subscribe to these.

Model

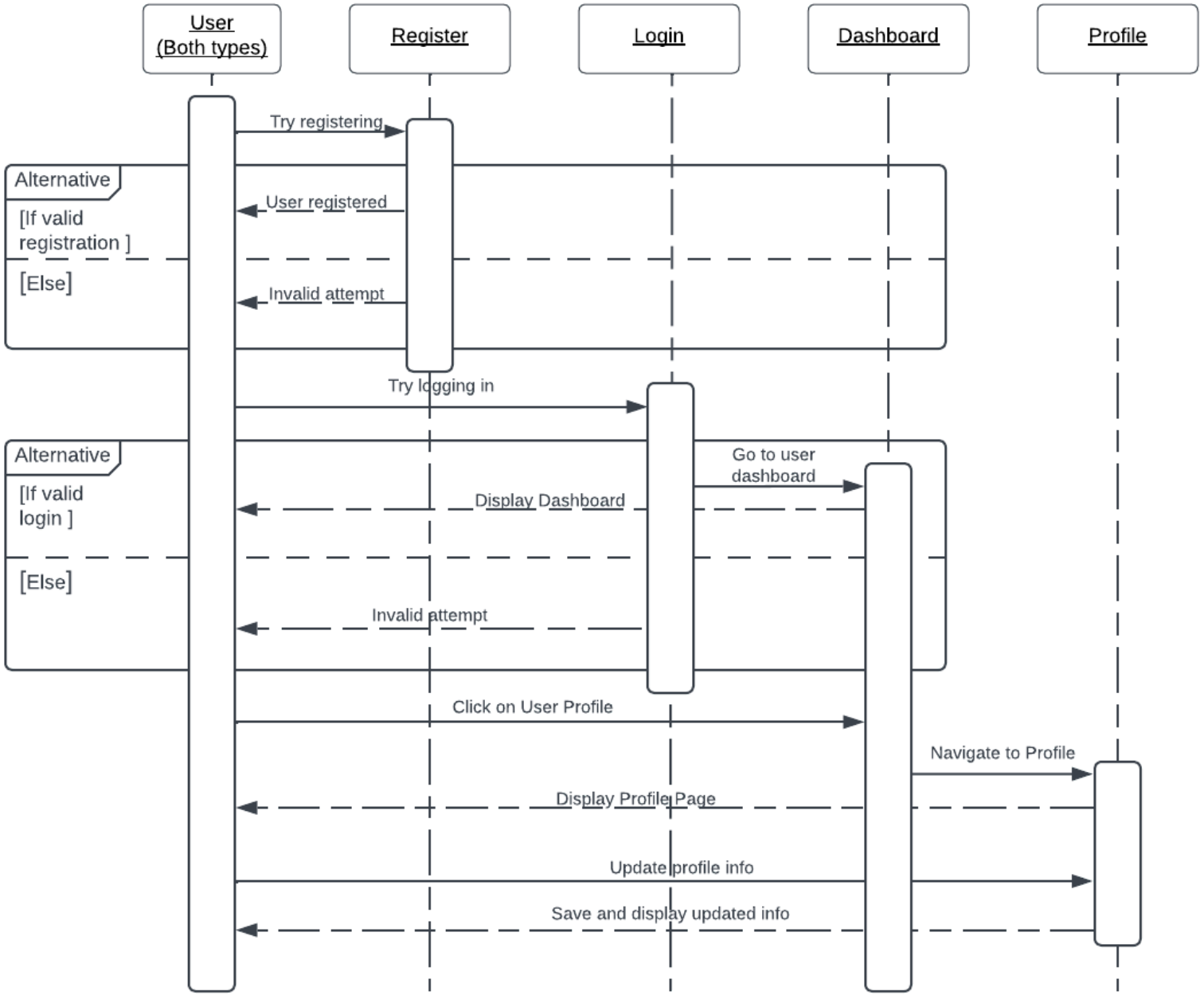

Dairy Product Management and Sales Webapp
Team 32 | March 28, 2023
Customer
- subscriptions : list[Subscription]
+ getUserType() : UserType
+ subscribeProduct(prodID, quantity, frequency) : Subscription
+ unsubscribe(SubscriptionID)
<<Abstract>>
User
- firstName : string
- lastName : string
- address : string
- phoneNumber : int
- emailID : string
- password : string
- userID : int
+ get() : User
+ update(firstName, lastName, address, phoneNumber)
Vendor
- farmName : string
- farmEstablishedDate : Date
- farmOpeningHours : Time
- farmClosingHours : Time
- workingDays : list[string]
- bankAccountNo : int
- bankHolderName : string
- bankName : string
- ifsc : string
- inventory : list[Product]
+ get() : Vendor
+ getUserType() : UserType
+ update(firstName, lastName, address, phoneNumber, ...)
+ addProduct(prodName, prodDesc, price, quantity)
+ removeProduct(prodID, quantity)
<<enumeration>>
UserType
Customer
Vendor
Auth
+ customers : list[Customer]
+ vendors : list[Vendor]
- screen : UserType
+ signUpCustomer(firstName, lastName, address, phoneNumber, emailID, password) : token, Customer
+ signUpVendor(firstName, lastName, address, phoneNumber, emailID, password) : token, Vendor
+ loginCustomer(emailID, password) : token
+ loginVendor(emailID, password) : token
+ logout()
- selectUserType(type)
Subscription
- productID : int
- quantity : int
- frequency : deliveryFrequency
- subscriptionID : int
- dateStarted : Date
+ update(frequency, quantity)
+ get() : Subscription
+ nextDelivery() : Date
<<enumeration>>
deliveryFrequency
Daily
Weekly
Alternate Day
Product
- prodName : string
- prodDesc : string
- price : float
- prodID : int
- quantity : int
- subscribtionIDs : list[int]
+ update(price, name, desc)
+ get() : Product
1
0..*
subscribeProduct,
subscriptions
customers
vendors
inventory
subscriptionIDs
productID

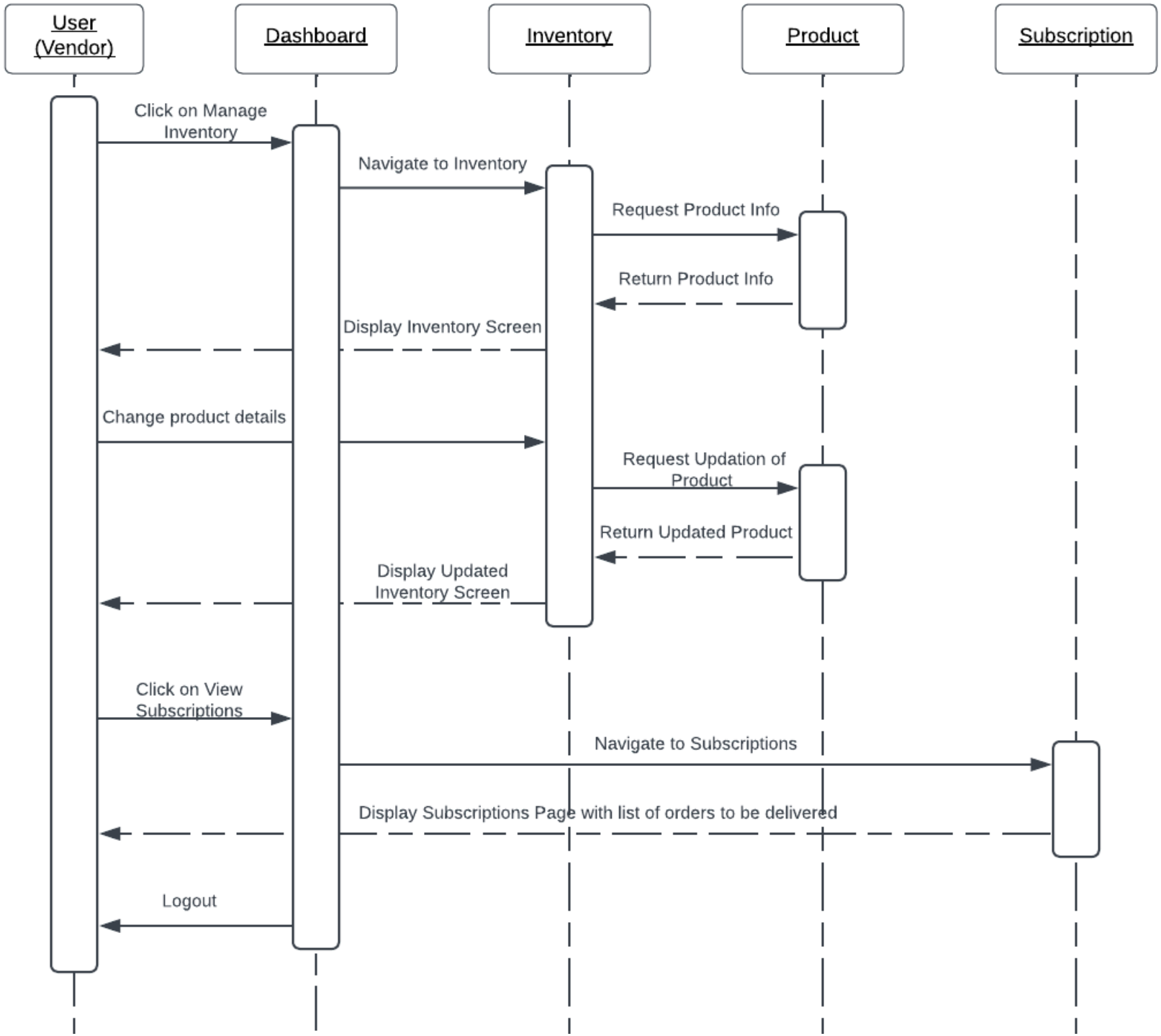
Auth	**Class State** ● screen : type of authentication screen ● vendors : list of Vendors ● customers : list of Customers **Class Behavior** ● signUpVendor() / signUpCustomer() : create vendor / customer account. ● loginVendor() / loginCustomer() : login to existing vendor / customer account ● logout() : logout of current account ● selectUserType() : change type of authentication screen
User **(abstract class)**	**Class State** ● firstName, lastName ● address ● phoneNumber ● emailID ● password : hashed password set by user ● userID ● type : Customer / Vendor **Class Behavior** ● get() : fetch user info ● update() : update user info

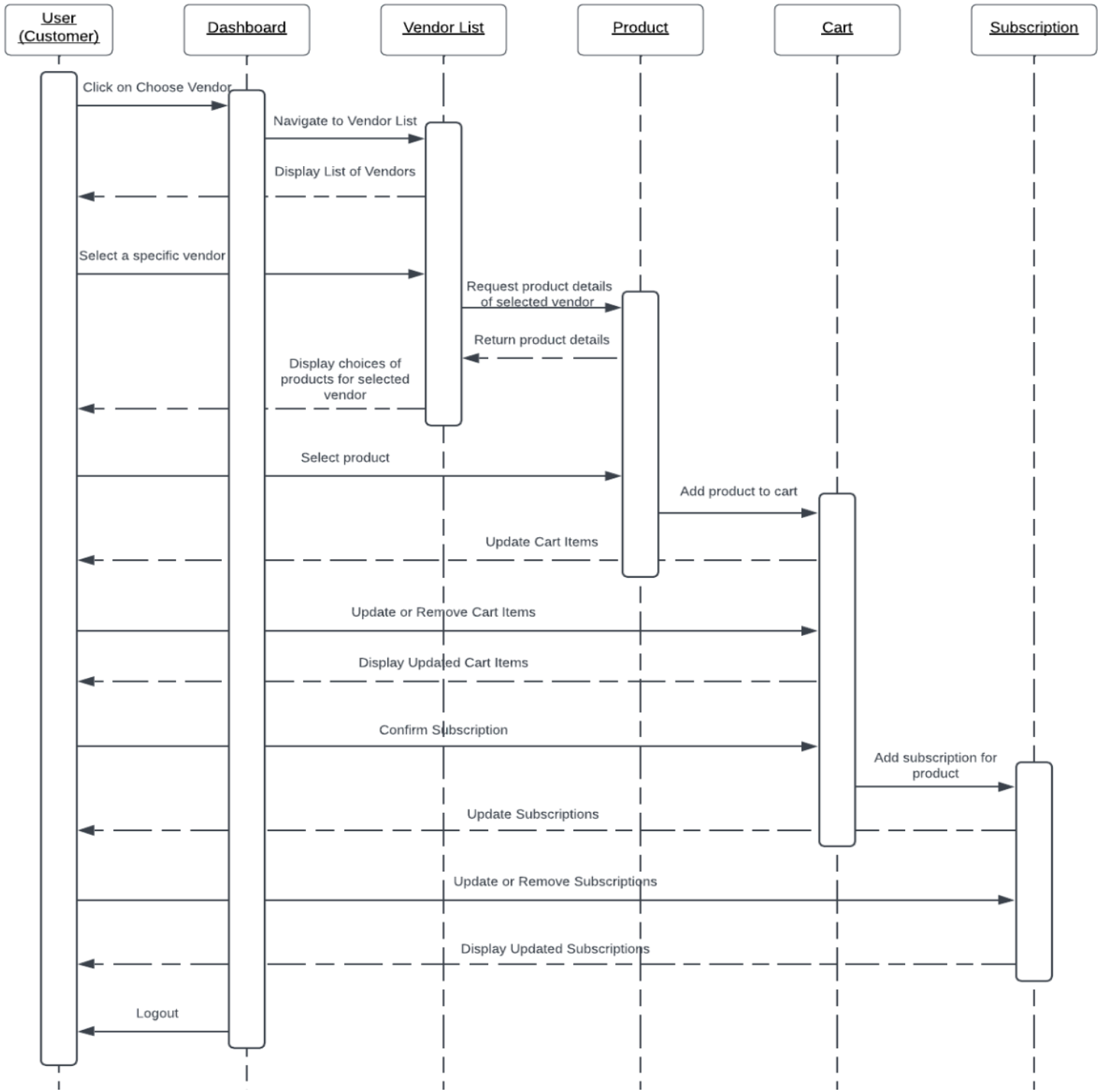
Customer **(child of User)**	**Class State** ● subscriptions : list of subscriptions of the customer **Class Behavior** ● subscribeProduct() : subscribe for a given quantity of a product for some frequency ● unsubscribe() : discontinue a subscription ● getUserType()
Vendor **(child of User)**	**Class State** ● farmName, farmEstablishedDate, farmOpeningHours, farmClosingHours : info about the dairy farm ● workingDays : list of days when farm is open ● bankAccountNo, bankName, bankHolderName, ifsc : bank details of vendor ● Inventory : list of products along in the current inventory. **Class Behavior** ● update() : overridden update method with more fields ● get() : fetch vendor info ● getUserType() ● addProduct() : add product to inventory ● removeProduct() : remove product(s) from inventory after delivery

Product	**Class State** ● prodName, prodDesc ● price ● quantity ● vendor : the vendor selling this product ● prodID ● subscriptionIDs : list of IDs of subscriptions to this product **Class Behavior** ● update() : update product info ● get() : fetch product info
Subscriptions	**Class State** ● product ● quantity : quantity of product subscribed ● frequency : frequency of delivery, daily / alternate day / weekly ● subscriber : customer who has subscribed ● startDate : Date at which subscription started ● subscriptionID **Class Behavior** ● update() : update frequency or quantity of subscription ● get() : fetch subscription info ● nextDelivery() : give date of next delivery

Sequence Diagram(s)




User (Vendor)
Dashboard
Inventory
Product
Subscription
Click on Manage Inventory
Navigate to Inventory
Request Product Info
Return Product Info
Display Inventory Screen
Change product details
Request Updation of Product
Return Updated Product
Display Updated Inventory Screen
Click on View Subscriptions
Navigate to Subscriptions
Display Subscriptions Page with list of orders to be delivered
Logout


User (Customer)
Dashboard
Vendor List
Product
Cart
Subscription
Click on Choose Vendor
Navigate to Vendor List
Display List of Vendors
Select a specific vendor
Request product details of selected vendor
Return product details
Display choices of products for selected vendor
Select product
Add product to cart
Update Cart Items
Update or Remove Cart Items
Display Updated Cart Items
Confirm Subscription
Add subscription for product
Update Subscriptions
Update or Remove Subscriptions
Display Updated Subscriptions
Logout

Design Rationale

The login/register page

Initially, we designed a user login page with a toggle button for different user-types that would trigger a card flip animation. However, we faced some issues with Vite rendering and CSS constraints during implementation.

We then pivoted to a sign-up page with a card taking up half of the screen, featuring large icon buttons for different user types, and a background on the other half of the page. While searching for appropriate icons, we came up with another idea: using a separate page for choosing the user type. After selecting the type of user, the user will be redirected to the corresponding login/register page. To improve the page's visual appeal, we added a hover effect. When the user hovers over their user type, an image related to that type pops up, giving the website users a 3D feeling.

The profile page

We were given a demo website to evaluate its functionality from the vendor's end.

For the profile page, we initially attempted to implement a design similar to the demo website provided. This design included a side navigation bar for different modules and a top navigation bar using Material UI components. However, after further consideration and exploration, we decided to use MUI Joy and their templates instead.