**CS673 Software Engineering** 

**Team 4 - Rxcellent**

**Software Design Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Ignacio Moral | Requirement Leader and Configuration Leader | *Ignacio Joaquin Moral* | 10-15-2022 |
| Ryan Burns | Team Leader | *Ryan Burns* | 12/10/2022 |
| Youqing Shao(Tsing) | Security Leader | *Youqing Shao* | 10/17/2022 |
| Zahit Odabas | QA Leader | *Zahit Odabas* | 12/10/2022 |
| Nivan Hanjura | Implementation Leader | *Nivan Hanjura* | 10/18/2022 |
| Chenfei Yu | Design Leader | *Chenfei Yu* | 10/18/2022 |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **1.0.1** | **Ignacio Moral** | **15-Oct-2022** | **Added Class Diagram, Started working on Business Logic** |
| **1.0.2** | **Youqing shao(Tsing)** | **16-Oct-2022** | **Add security points** |
| **1.0.3** | **Zahit Odabas** | **17-Oct-2022** | **Architecture overview, backend structure, database schema** |

| **1.0.4** | **Ryan** | **17-Oct-2022** | **Introduction, design pattern and cleaned up the doc.** |
| --- | --- | --- | --- |
| **1.0.5** | **Ignacio** | **18-Oct-2022** | **Added Class Diagram description, included Business Logic Diagrams for Search functionality and Medicine Page functionality** |
| **1.0.6** | **Christol** | **19-Oct-2022** | **Added frontend design and design patterns** |
| **1.0.7** | **Zahit** | **20-Oct-2022** | **Updated Security Design section** |
| **1.0.8** | **Ignacio Moral** | **06-Dec-2022** | **Updated Database Design** |

[Introduction](#_87t9hln2vjz0)

[Software Architecture](#_3ipvmjgn6clp)

[Class Diagram](#_ky60nv8suxxm)

[UI Design (if applicable)](#_7ucksmkf6rzx)

[Database Design (if applicable)](#_tcmuor4nl1kz)

[Security Design](#_x18fj36s1121)

[Business Logic and/or Key Algorithms](#_mtfbusfb0eq3)

[Design Patterns](#_9zvwkmc4luo5)

[Any Additional Topics you would like to include.](#_15tmymhipvdv)

[References](#_50ojo9i46ytq)

[Glossary](#_8n34lvocupub)

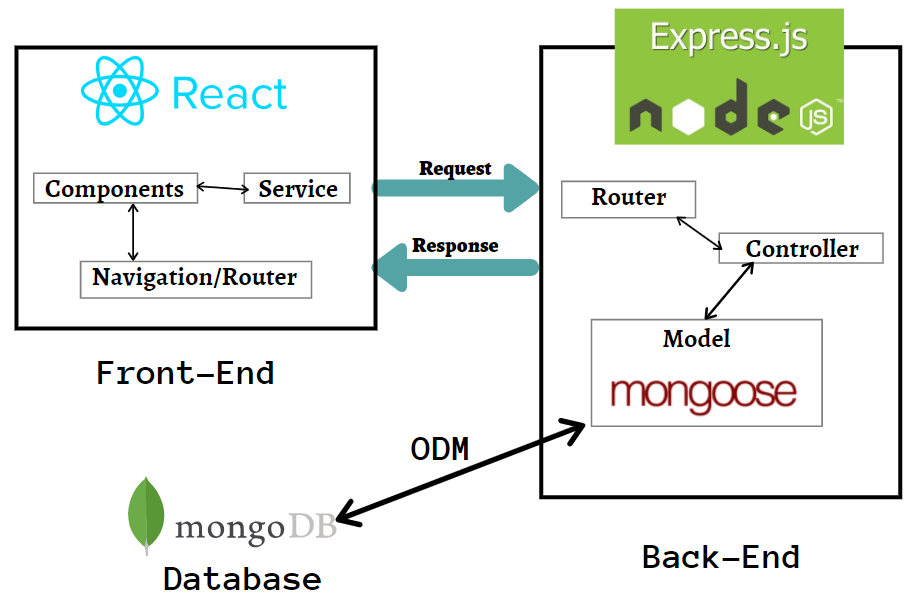
# Introduction

This document details the Software Design of our prescription e-commerce website, Rxcellent. We are using Javascript, the React JS library, and some TypeScript to build our application. Our team is leveraging each individual's professional skills and coding experience. Our goal is to develop a web application that implements REST APIs. Once our application is ready for production we will deploy it using Vercel..

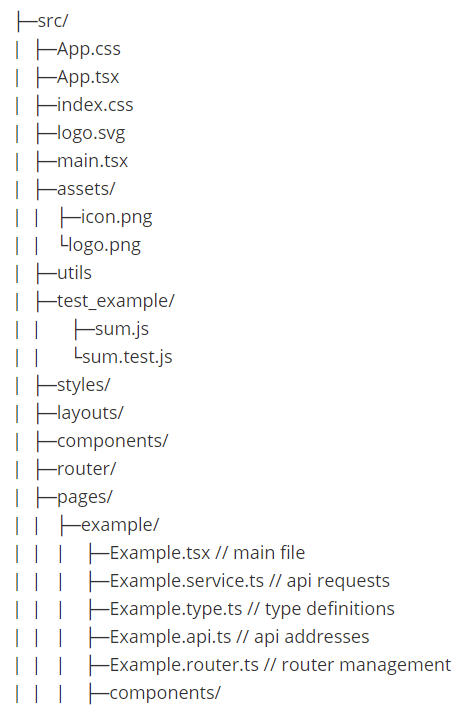
# Software Architecture

*In this section, you will describe the decomposition of your software system, which includes each component (which may be in terms of package or folder) and the relationship between components. You shall have at least one diagram to show the whole architecture of . The interface of each component and dependency between components should also be described. If any framework is used, it shall be defined here too.*

We’re developing a web app in a client-server architecture. The client is the web-page which is built by the React.js and many other libraries. The server side is powered by Node.js and will be deployed to Vercel. We plan to use mongoDB as our database.

****

**Frontend folder structure**



The main purpose of this frontend folder structure design is to separate the UI from business logic and encapsulate components. For every page, its UI, http requests, api and router are defined in separate files. Components that will be frequently reused are extracted. AJAX related operations are encapsulated in utils folder.

**Main libraries:**

React.js

React-router: router management

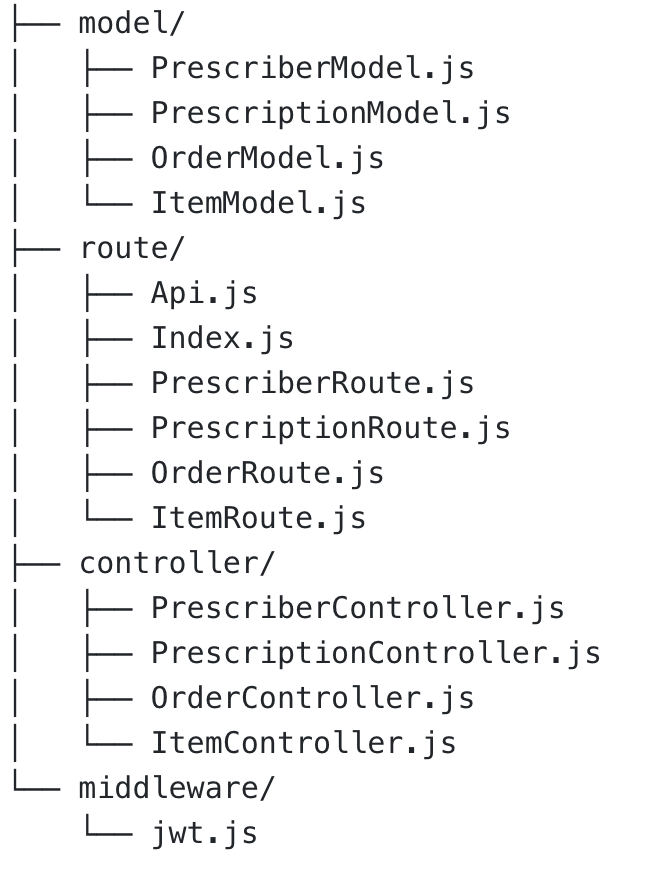
React-query: asynchronous states management

Axios: http requests

MUI: UI component library

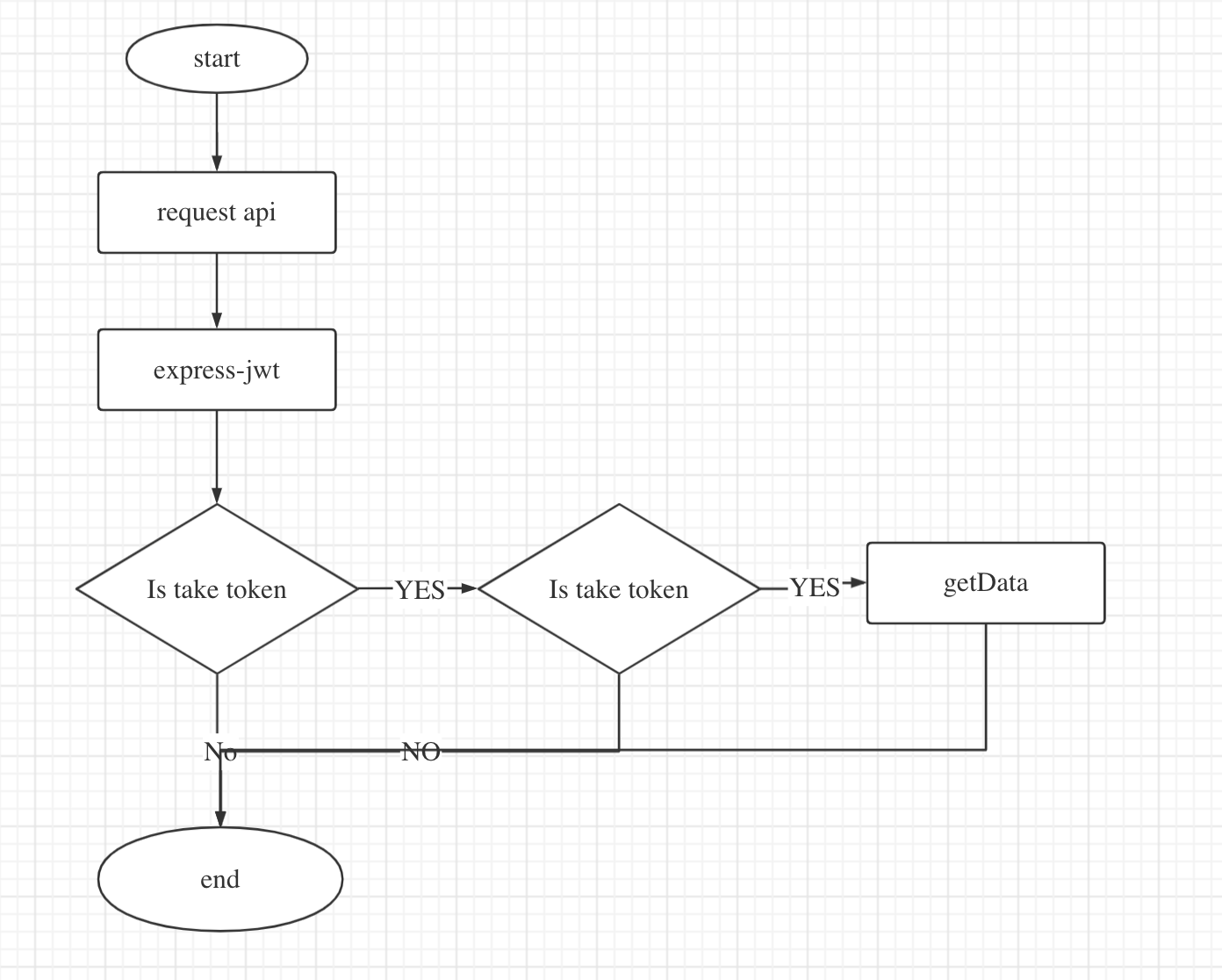
Sass: CSS pre-processor

**Backend folder structure**



In the backend, we will have our models and some application logic in controllers. Our models will be created by Mongoose object data modeling tool which will map our models to our database tables. Routes will define the API’s we will use for each model. Controllers will basically be used for data validation and sanitation, and they will call the related API. We plan to use json web token as middleware for prescriber login

Jwt work flow:



# Class Diagram

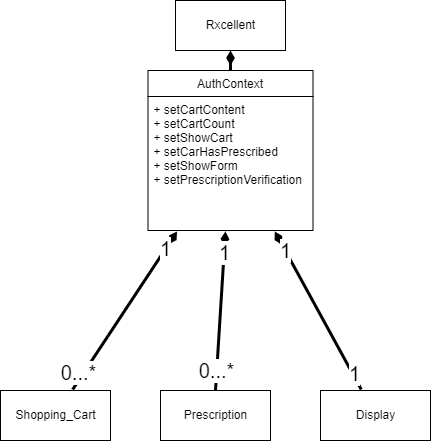
In this section, you will provide a detailed description of each component (or package) and use one or multiple class diagrams to show the main classes and their relationships in each component.

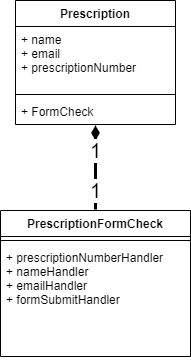
The Class Diagram is sectioned into multiple components. React is a component-based library, and so many of these classes will behave like components. The main component is the AuthContext component. This class is in charge of handling what can show up, what needs to be visualized, what will need to be sent to databases, etc.

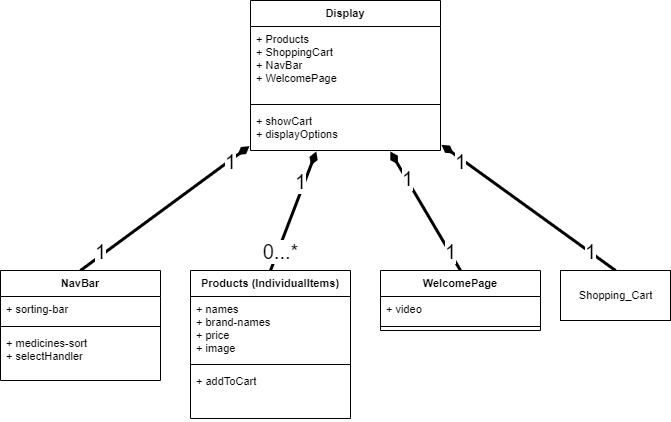
From there, there’s the Shopping\_Cart component, which is in charge of handling all transactions, products, and the sells. This Shopping\_Cart class has a component subclass called CartItem, which is each product that will be sold.

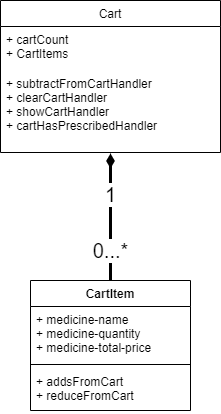
AuthContext also has the Prescription component. This class handles all prescription inputs by calling it’s FormCheck subclass, but it does not verify the prescription. That is done in AuthContext.

Finally, there’s the Display component. This component is the View in our architecture. Depending on what the AuthContext allows/verifies, Display will show. It has multiple subcomponents, mostly in charge of the visualization of each thing. NavBar shows the navigation bar, Products are the medicines for sale, we have a welcome page, and it will also display or hide the Shopping Cart.



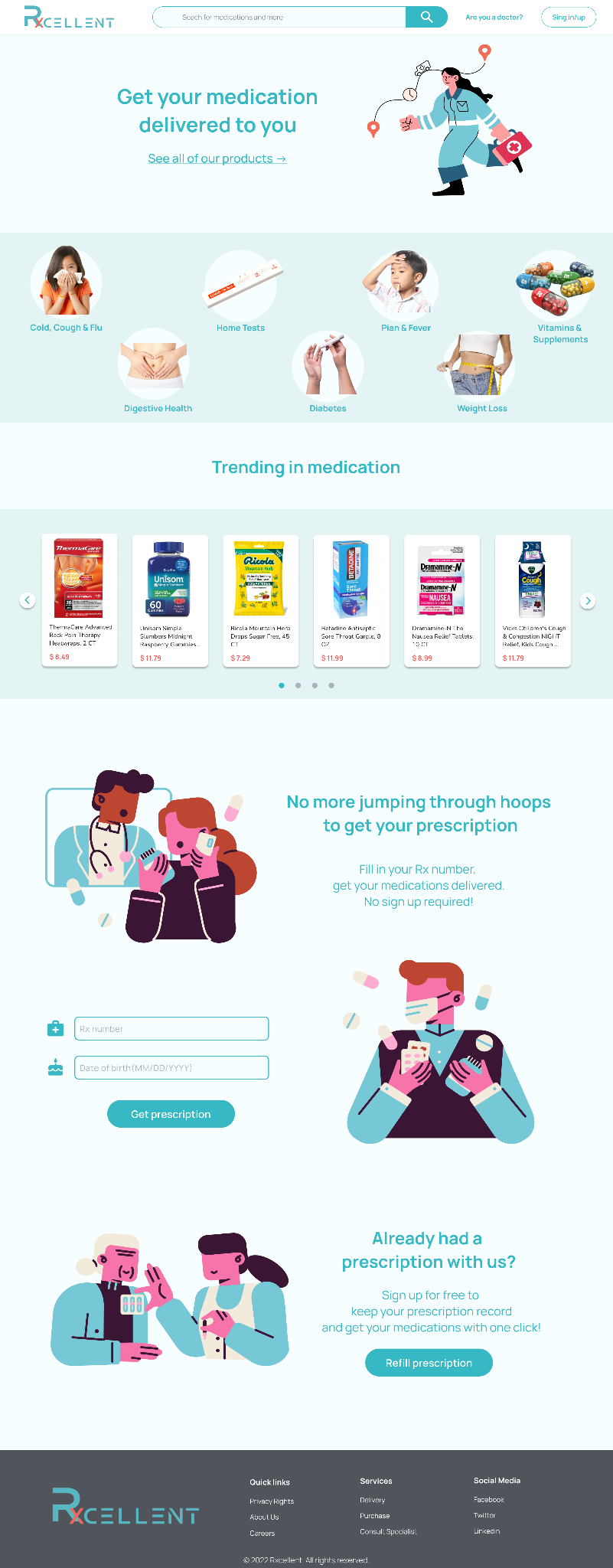


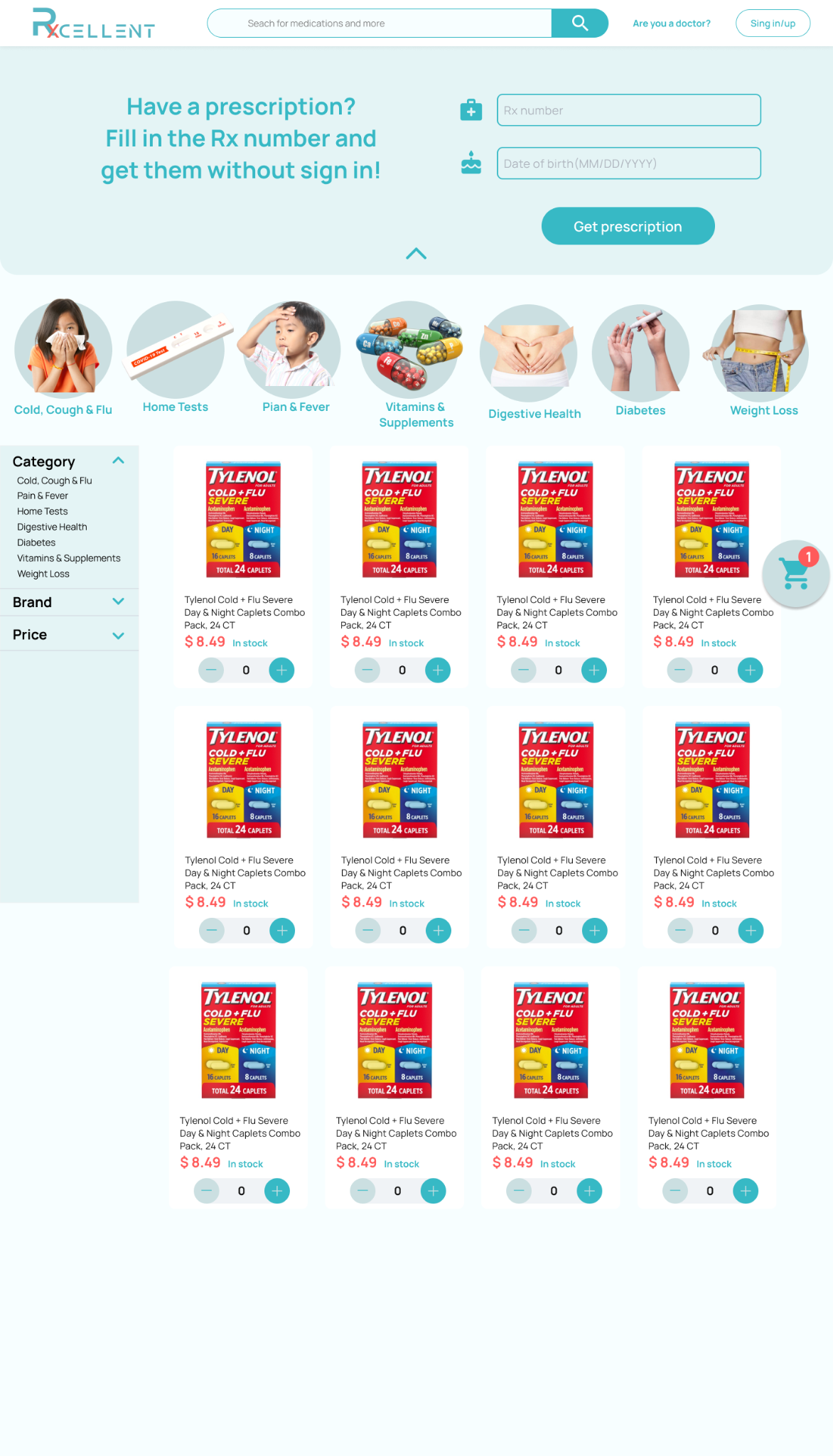


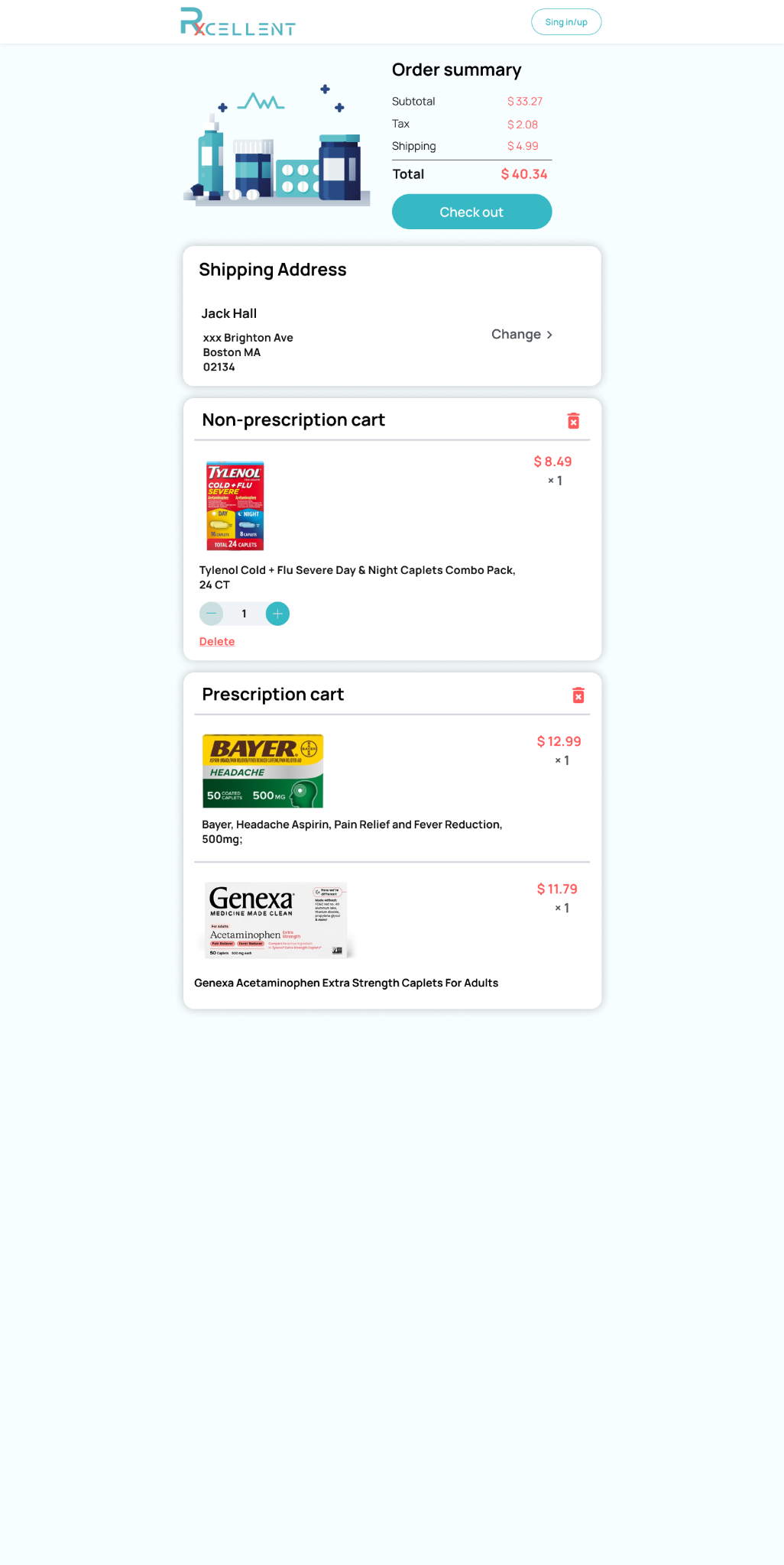


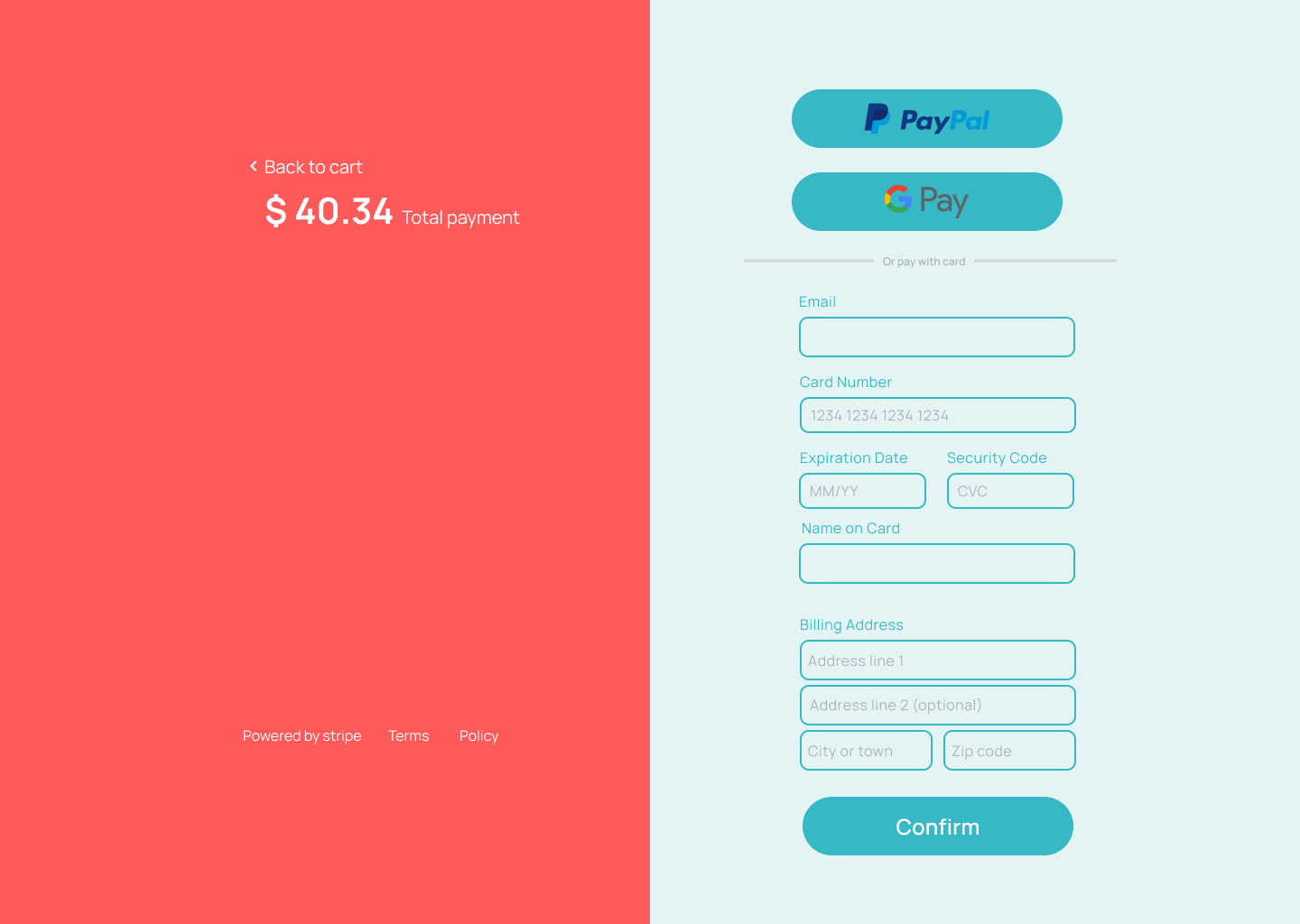
# UI Design (if applicable)

We utilized figma to create an appealing and sleek user interface.



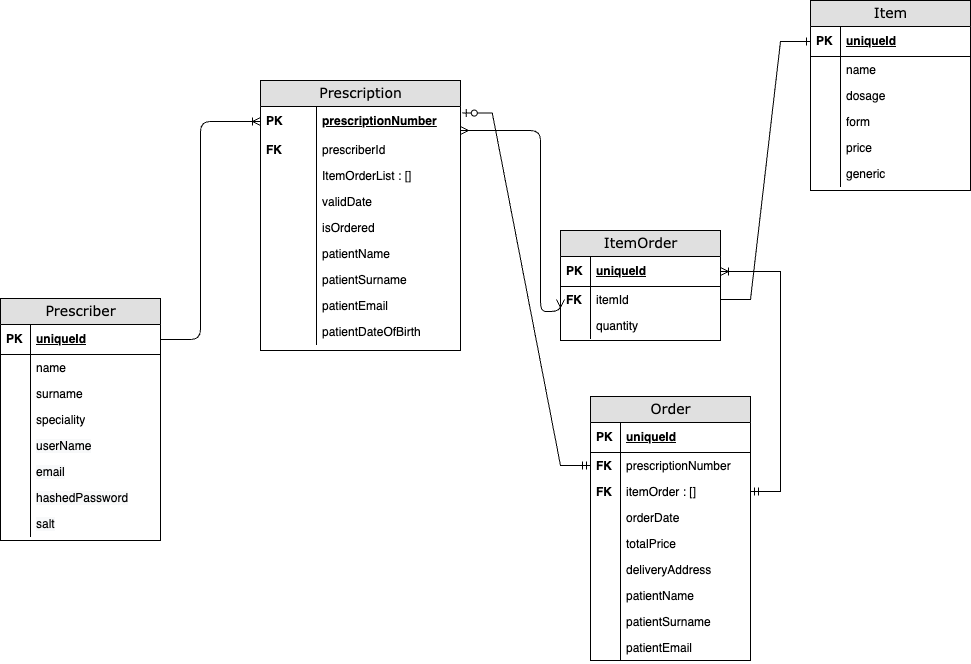






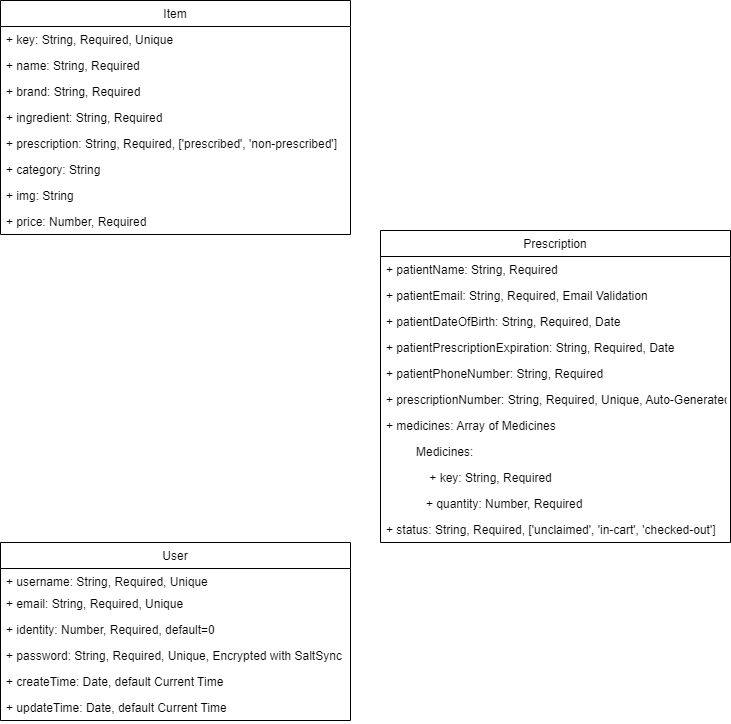
# Database Design (if applicable)

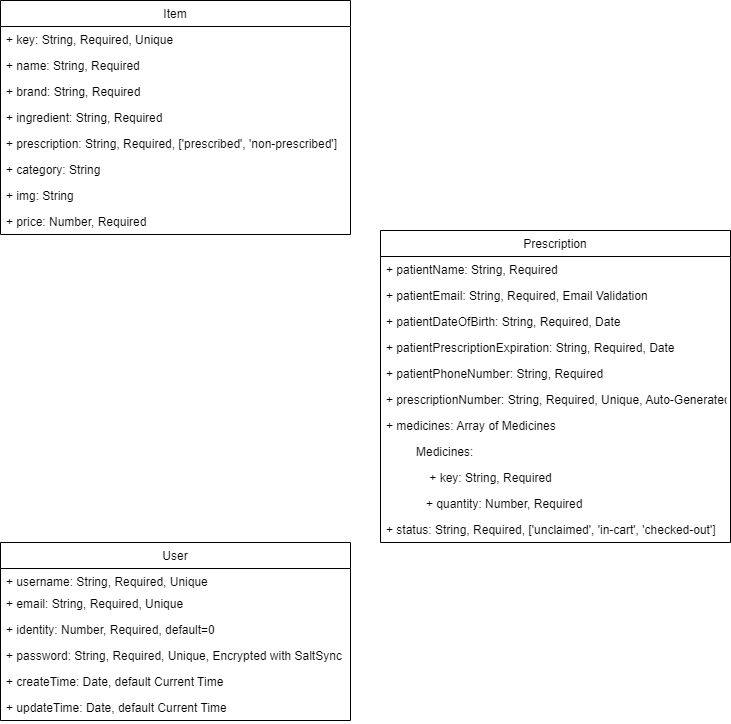
In this section, you shall describe any database schema if used in your software system.

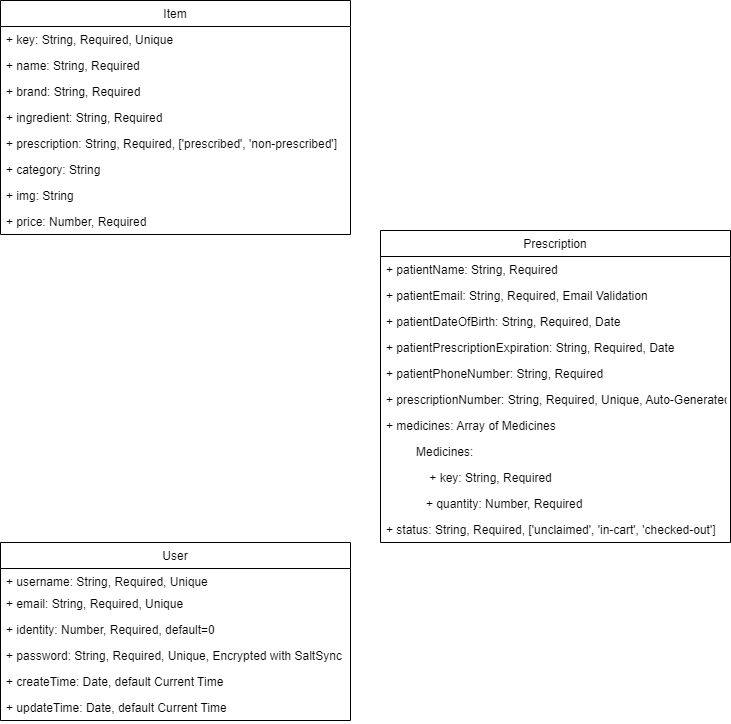


This is the ER Diagram of our db. Since mongodb uses a non-relational database model, there’s no real primary or foreign keys. But data in the tables(documents in mongoDB) itself has relation. The relation is created via manual references in mongodb. Pks and Fks are added to the diagram to make the object references more understandable.

## Updated For Iteration 3







For Iteration 3, some updates were done in the database models. We kept using MongoDB, so the Database Diagram is mostly Document Oriented.

The Item Schema focuses on the items Rxcellent has for sale. It is always identified by a key for easy access and for sorting in the Prescription Schema. They also contain name, brand, ingredients, whether or not it’s a prescription medicine, and the price. It can also contain an image to show the product, and a category for user search purposes.

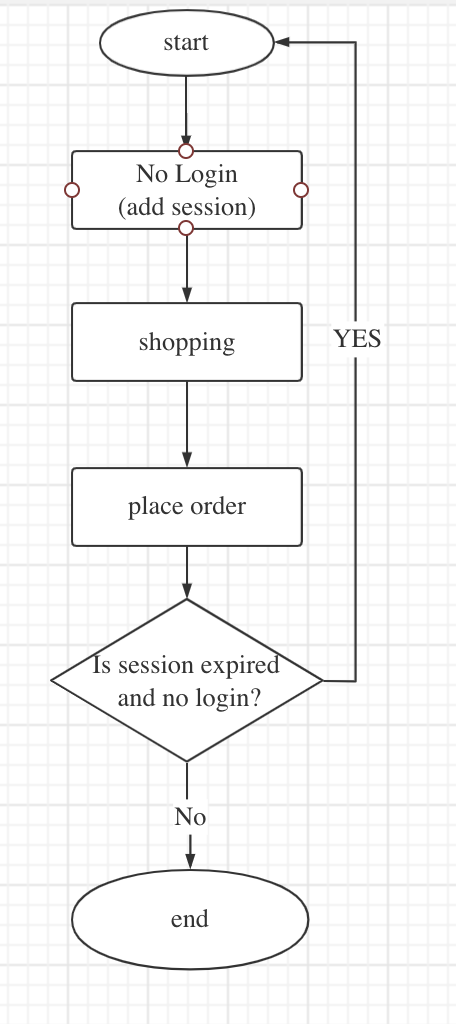
Then the Prescription Schema contains all the information a doctor might need to write a prescription for the patients. It contains the patient’s name, email, date of birth, and phone number. It also contains information about the prescriptions, like a subdocument of the medicines required, sorted by a key, the expiration of the prescription, the number, and status for shopping purposes. The patients will receive an email with their prescription number, which they can input alongside their date of birth to the website to easily shop for their medicines.

Finally, there’s the User Schema. This is mainly used by doctors, but regular customers will be able to register in the future. It contains a username, email, password, and time of creation and update. It will also contain an identity that tells the system whether the user is a doctor or a regular customer.

# Security Design

In this section, you shall describe any security design in your software system.

* Session
  + When the user is not logged in, it is permitted, but finishing their own order in 30 mins. Prevent user information(prescription number, order info etc) leakage caused by long-time online.

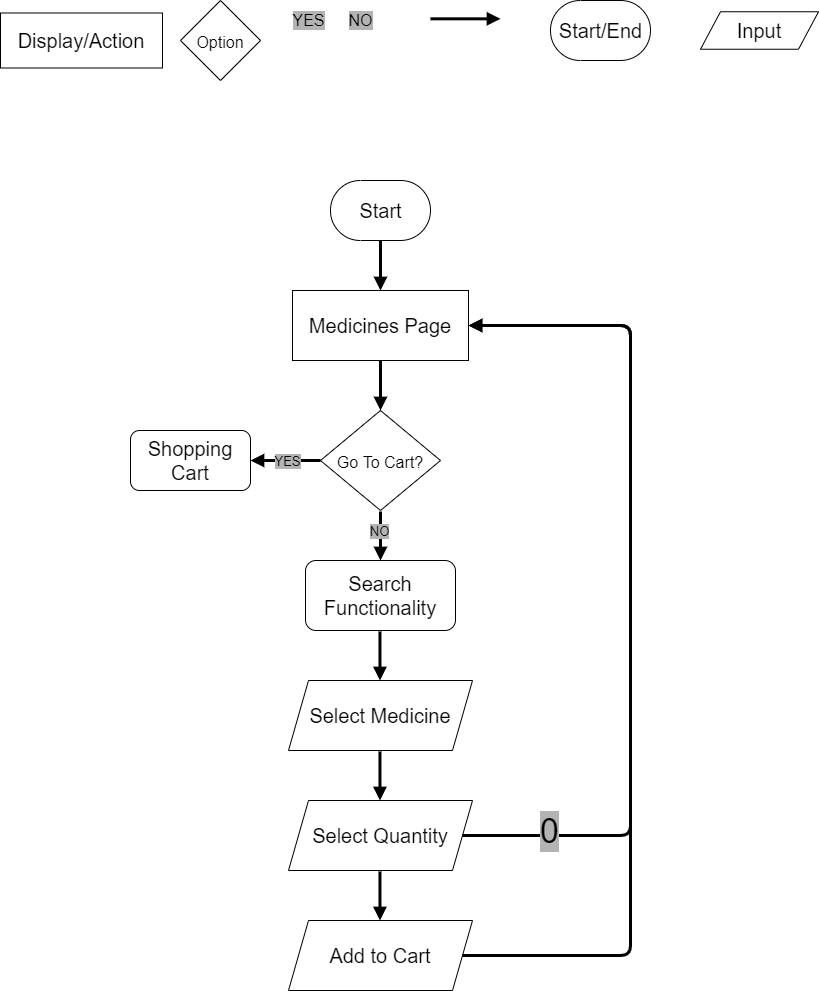


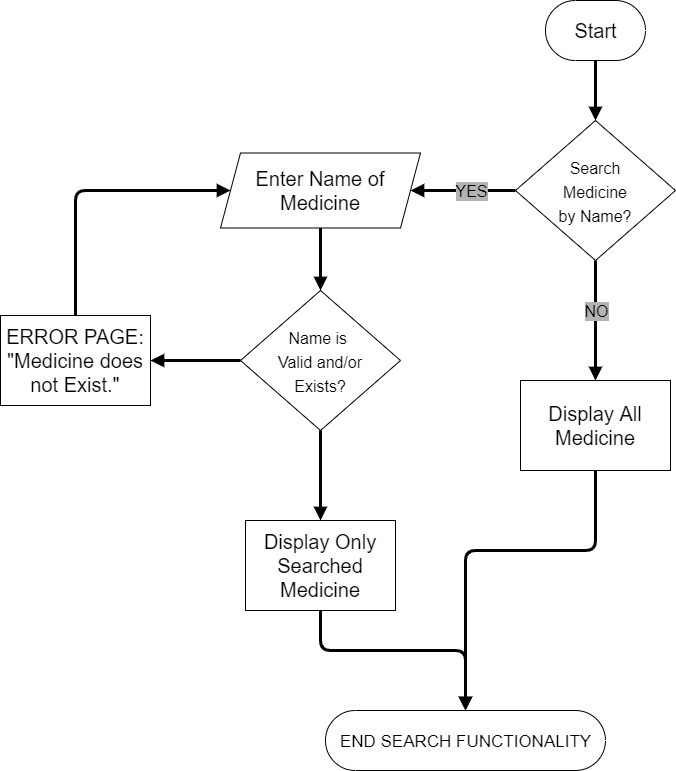
A timestamp will be assigned to the user when he visits our web page for the first time. In every 30 mins, timestamp will be reset, all react state and cart data will be deleted and the user will be forwarded to the home page.

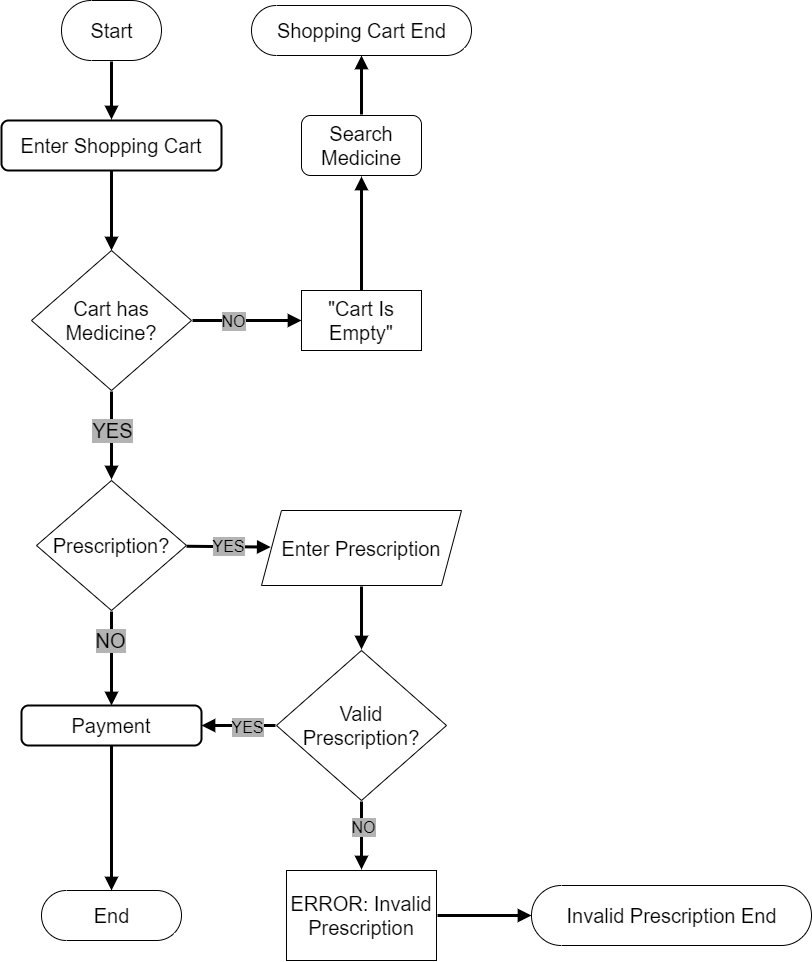
* Jwt authentication
  + A jwt will be generated when a prescriber logs in. There will be a 30 min timeout for prescribers as well and if the time exceeds, the system will automatically invalidate the jwt and log out the prescriber
  + If an optional feature of user login is implemented, the logged in user will generate a jwt, Token authentication is required for saving users’ information and orders.
* Script attack
  + React has a built-in security measure to auto-escape any value embedded in jsx files. It prevents XSS attacks as long as the DOM is not mutated directly (Eg: innerHtml, findDOMNode), following React implementation recommendations.
  + We will be using jwt for user authentication. It will mitigate most CSRF attacks as long as it’s transmitted in the Authorization header and not in the cookie. We will not allow CORS requests as an additional measure.
* Sensitive data
  + Sensitive data will be encrypted (Eg: date of birth, prescription number, delivery address)
* SQL command execution
  + We will prevent sql injection, by sanitizing client side data and using parameterized SQL queries.
* Password & Login security
  + We will enforce at least 8 characters, with an uppercase, a lowercase and a numeric character for passwords. We will store hashed passwords in our DB. We will use salt with the hashing algorithm to improve the security of passwords.

# Business Logic and/or Key Algorithms

In this section, you shall describe any key algorithms used in your software system, either in terms of pseudocode or flowchart, or sequence diagrams.





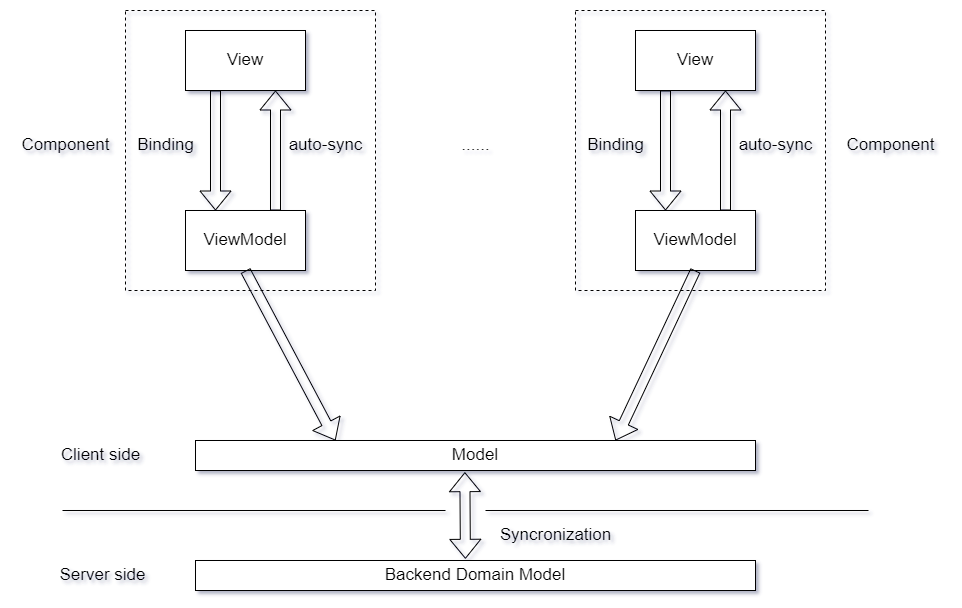


# Design Patterns

*In this section, you shall describe any design patterns used in your software system.*

Our application is designed based on a MVVM pattern. On the frontend, we write components to assemble the user interface, and each component consists of a view layer and a view-model layer. The view layer is the jsx code which defines component styles, and the view-model layer defines component logic, exposes data needed by view layer, and is triggered to render the view when data is updated. The models of view-model layers come from the model layer, and the model layer consists of a client side model and a server side model. When a request is initiated, the client side model layer will send the request to backend domain model, get the data synchronized, and handles the response (if there is).

React.js itself does not form any design pattern. In real implementation, we use many other libraries and tools. In client side model layer, there basically are AJAX operations, and Axios is be used to encapsulate these operations. In view-model layer, react-query is used for asynchronous state management, and react-router is used for navigation.



# Any Additional Topics you would like to include.

# References

# Glossary

* + JWT: JSON Web Token