Software Requirements Specification

for

AnyTimeFoods

Version 1.0 approved

Prepared by Team_909

IIT Kharagpur

29-03-2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Version 0.1	27-03-2022	Initial SRS Document	0.1
Version 1.0	29-03-2022	Updated the SRS Document with diagrams	1.0

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of AnytimeFood. It will explain the objective/purpose of this project and its scopes of application, features of the projects, the interface and the details about interactions through the interface with the database, data management strategies,constraints under which it will operate.

1.2 Document Conventions

This document has been written using Arial font with font size 11 and important points are written in bold. Additionally the headings are written in bold, using 'Times' font with font size 18 and subheadings with font size 14.

1.3 Intended Audience and Reading Suggestions

This document is intended for developers contributing towards the project, the client, users, and testers and anyone who is reviewing the project to easily view descriptions about functional and no-functional requirements and understand product scope.
This document contains six(6) sections:
a. An overall description, which includes a summary of the scope, assumptions and limitations of the project, types and characteristics of users, plan for implementation.
b. A brief description of external interfaces requirements.
c. A description of the system, which includes an explanation of the system's purpose, features and benefits, characteristics, and technical and operational background; and,
d. A list of nonfunctional requirements, which contains information regarding performance,safety, security and business rules.
e. A description of other uncategorised requirements.
f. A list of the appendices.(definition of terms used in the document, analysis model UML and future extension possibilities)

This is a suggested sequence to go through the SRS:
-product scope and perspective
-product function
-user classes and characteristics
-Appendix B
-system features
-other requirements

1.4 Product Scope

It is proposed to design and build a modernized hybrid cloud architecture for

AnytimeFood using new container and cloud native technologies to ensure that their critical operations such as food ordering and delivery can still function normally to provide an acceptable consumer experience even during a downtime at its centralized IT core.

1.5 References

- 830-1998 — IEEE Recommended Practice for Software Requirements Specifications : https://ieeexplore.ieee.org/document/720574

2. Overall Description

2.1 Product Perspective

In the current existing system of Anytime Food ,a major drawback is that there would be total downtime if there be any major disruption in its central IT core.So to counter this it is being planned to design and build a hybrid cloud solution, using new container and cloud native technologies to ensure that their critical operations such as food ordering and delivery can still function normally to provide an acceptable consumer experience even during a downtime at its centralized IT core. The product developed is scalable and provides business continuity in the event of outages. The product allows customers to place orders and track the delivery status, stores to sell food items and provide an interface for delivery staff.

2.2 Product Functions

The software is intended for the use of three types of users - customer, stores and delivery staff. The functions for each type of user provided by this software are listed as follows:

1. **Customer**
   a. Users can place and cancel orders, give ratings and track the status of their delivery.
   b. Query for stores having specific food items.
2. **Store Manager**
   a. The store manager can edit data about their store like their timings.
   b. The store manager can add, edit and delete food items to the menu.
   c. They can also track the status of deliveries from their store.
3. **Delivery Partner**
   a. The delivery partners are assigned orders to deliver, based on their location.

2.3 User Classes and Characteristics

The following classes are used in the software:

1. User : Every user has their personal details like mobile number, password, address, etc. Apart from that, the type of the user (whether a delivery person or a customer) is also stored.

2. Order: Order holds information about which store it is from, the list of food items ordered, total price, user details and a transaction token.
3. Item: It holds detail about a particular food item like their name, price, picture, etc.
4. Store: It is a class to store information about a store - the food items they offer, their timings, address and contact info.
5. Delivery: The class holds information about a delivery, like the delivery partner assigned, store to pick up from, delivery address, status of delivery, etc.

2.4 Operating Environment

We are testing this to work on the following tech stack - Python 3.6+, MySQL. We have tested it to run on reasonably recent modern browsers like Google Chrome 88+, Firefox 86+ on amd64 architecture.

2.5 Design and Implementation Constraints

- In one order, items from only one store can be ordered.

2.6 User Documentation

The front-end of the software will be kept sufficiently intuitive and straightforward for anyone who is familiar with browsing the internet through their computer or smartphones.

2.7 Assumptions and Dependencies

It is assumed that the user has access to:
1. An email account
2. A working contact number(mobile)
3. A desktop or laptop or smartphone to access and run the software

Additionally the users need to have a browser through which they will access this software

3. External Interface Requirements

3.1 User Interfaces

The user-interface is a browser-based GUI. It begins with a *login* page, where the user logs in with his/her username and password. He is then taken to the correct dashboard, depending on the type of user he/she is. The customer can also sign-up from the *sign-up* page.

After a customer has logged in, they are taken to the stores page where stores nearest to the user are shown along with a search bar and other filters. A customer could then select a store and place orders from it. There are also interfaces for users to track their orders and edit their profiles.

The store managers can edit details about their stores and menu, and track delivery status of their orders. The delivery staff can update the status of delivery assigned to them as delivered/picked up.

3.2 Hardware Interfaces

The product is intended to have two components - a backend API server divided into multiple microservices and the frontend which will be accessed by users through their browsers. Multiple instances of the microservices will be hosted and orchestrated using docker and kubernetes. For the backend, it will be handled by powerful computing resources having powerful processors and physical memory. However, the users should have a desktop/laptop(preferably with 4gb ram or more) or a smartphone(preferably with 2gb ram or more) for smoothly using this software through their browser

3.3 Software Interfaces

- The backend of the software will consist of the various databases based on the SQL framework(SQLite), the server being implemented using Django.
- The frontend will consist of a graphical user interface developed on Javascript, HTML and CSS and ReactJs .
- An API gateway redirects the HTTP request to the appropriate microservices.
- The product is containerized using Kubernetes. #todo

3.4 Communications Interfaces

There will be a direct communication between the backend and frontend of this software using standard HTTP(HyperText Transfer Protocol) using REST API.

4. System Features

4.1 Login and Creating an Account

4.1.1 Description and Priority

This is the first task of the customer to create an account. Only after the customer has created an account, can they access the different stores and place orders.

4.1.2 Stimulus/Response Sequences

The customer has to enter their personal details like name, contact details and address, the software will create an account for the customer and store the data.

4.1.3 Functional Requirements

Req 1: Prompt the user to login, and give an option to create an account if they don't have one.

Req 2: Ask the user details for necessary information.
Req 3: Allow the user to modify the details if required.

4.2 Adding Items to Cart and Placing Order by Customer

4.2.1 Description and Priority

The customers can select a store, add items from it to the cart and place an order.

4.2.2 Stimulus/Response Sequences

The customer has to select food items, identify the quantity and then place the order, also performing payment.

4.2.3 Functional Requirements

Req 1: Ask the user for quantity for each item they select.
Req 2: Prompt the user for payment and process the order to the store once payment is confirmed.

4.3 Adding Items to Cart and Placing Order by Store Staff

4.3.1 Description and Priority

The store staff (kiosk manager) places orders on behalf of customers from their staff.

4.3.2 Stimulus/Response Sequences

The store staff selects items and places the order from their customer.

4.3.3 Functional Requirements

Req 1: Display the menu and let the store staff, p
Req 2: Prompt the user for payment and process the order to the store once payment is confirmed.

4.4 Adding, Editing and Deleting Items in Menu

4.4.1 Description and Priority

The stores can add, edit and delete items sold by them.

4.4.2 Stimulus/Response Sequences

The store dashboard will have buttons which can allow the manager to add, edit and delete items from their menu. For addition, the manager will have to give details about the item like name, price, quantity, etc. After the action, the changes will be reflected in the menu.

4.4.3 Functional Requirements

Req 1: Ask the user if they want to add, edit or delete items.

Req 2: Based on their choice, make an appropriate API call and reflect the changes in the menu.

4.5 Fetching nearest free Delivery Partner

4.5.1 Description and Priority

For assigning delivery partners for orders, the nearest delivery partner is fetched.

4.5.2 Stimulus/Response Sequences

Upon placing the order, the system will find the delivery partner nearest to the store who is not assigned any deliveries.

4.5.3 Functional Requirements

Req 1: On placement of order, locate the nearest delivery partner based on their GPS location.

4.6 Send and Verify OTP

4.6.1 Description and Priority

Send OTP to mobile and verify from user. This is required while registering users and also delivery partners verify OTP when delivering orders to customers to make sure it's delivered to the correct person.

4.6.2 Stimulus/Response Sequences

The software prompts from an OTP that was sent to the user. The software then checks if the OTP entered is indeed the one that was sent by it.

4.6.3 Functional Requirements

Req 1: On placement of order, locate the nearest delivery partner based on their GPS location.

4.7 Find Stores closest to customer

4.7.1 Description and Priority

Finds the stores which are closest to the location of the customer.

4.7.2 Stimulus/Response Sequences

When the user opens the app, they are shown the list of stores which are nearest.

4.7.3 Functional Requirements

Req 1: The software should get the GPS location of the user and then sort the stores based on distance from them.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The system response should be fast. Time required for fetching the user details from the database and displaying them on the frontend should ideally be less than 1 second.
- Also the system should immediately relay the issues to the admin.
- When a store adds a new food item, it should be immediately available to the customers.
- The OTP sent for verification should reach the user in a few seconds.

5.2 Safety and Security Requirements

The user will be able to report any bugs they encounter to the admin so that the developers can fix it in the next release. The user details will be kept completely private and will be accessible only to the user himself. Additionally every user will be allotted a unique user ID that will be known only to them and will act as the gateway to access his account. All the data will be protected in order to prevent any possible data theft.

5.3 Software Quality Attributes

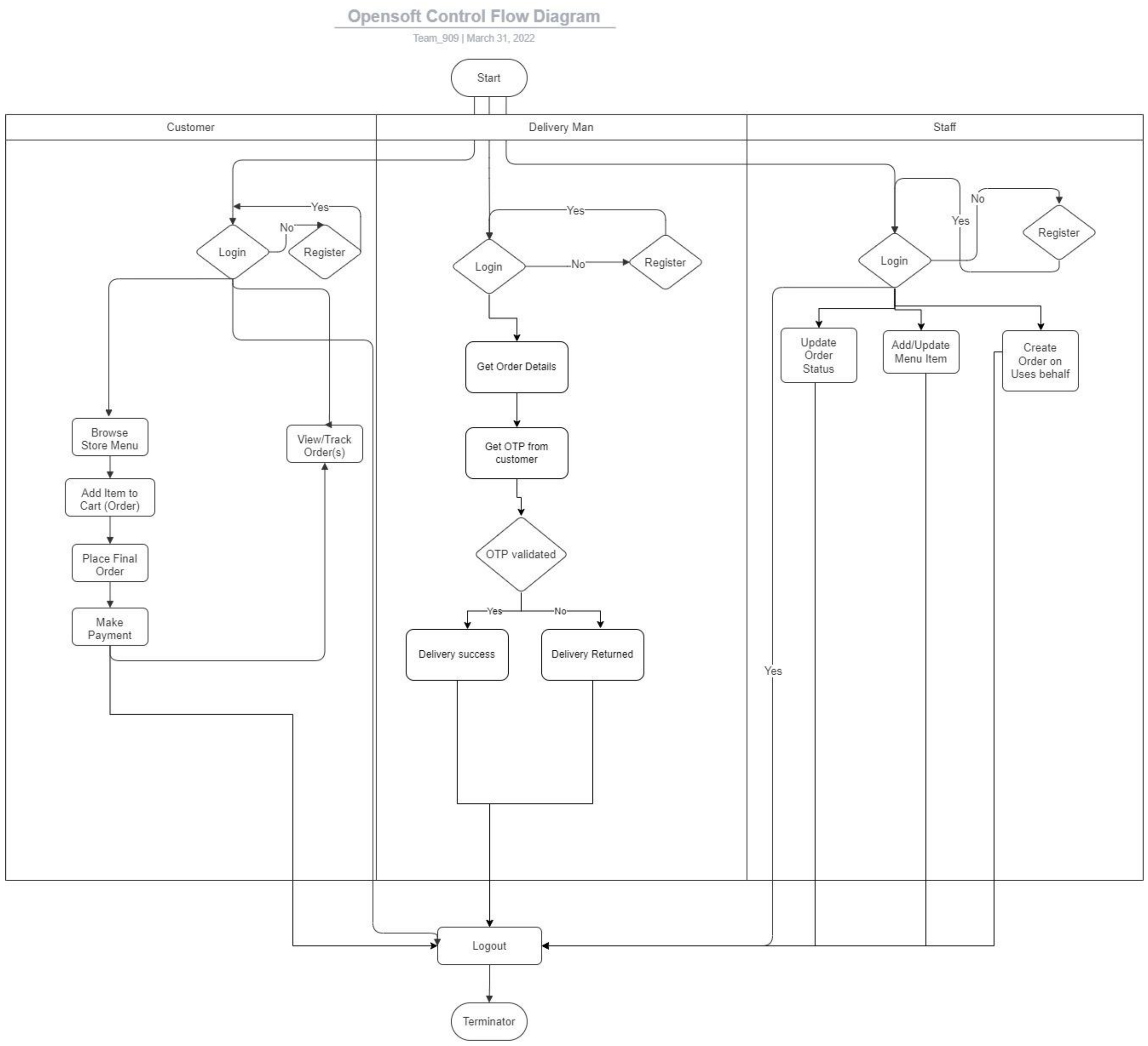
The software provides a well-designed, easy to use interface that allows anyone to use the software.

6. Other Requirements

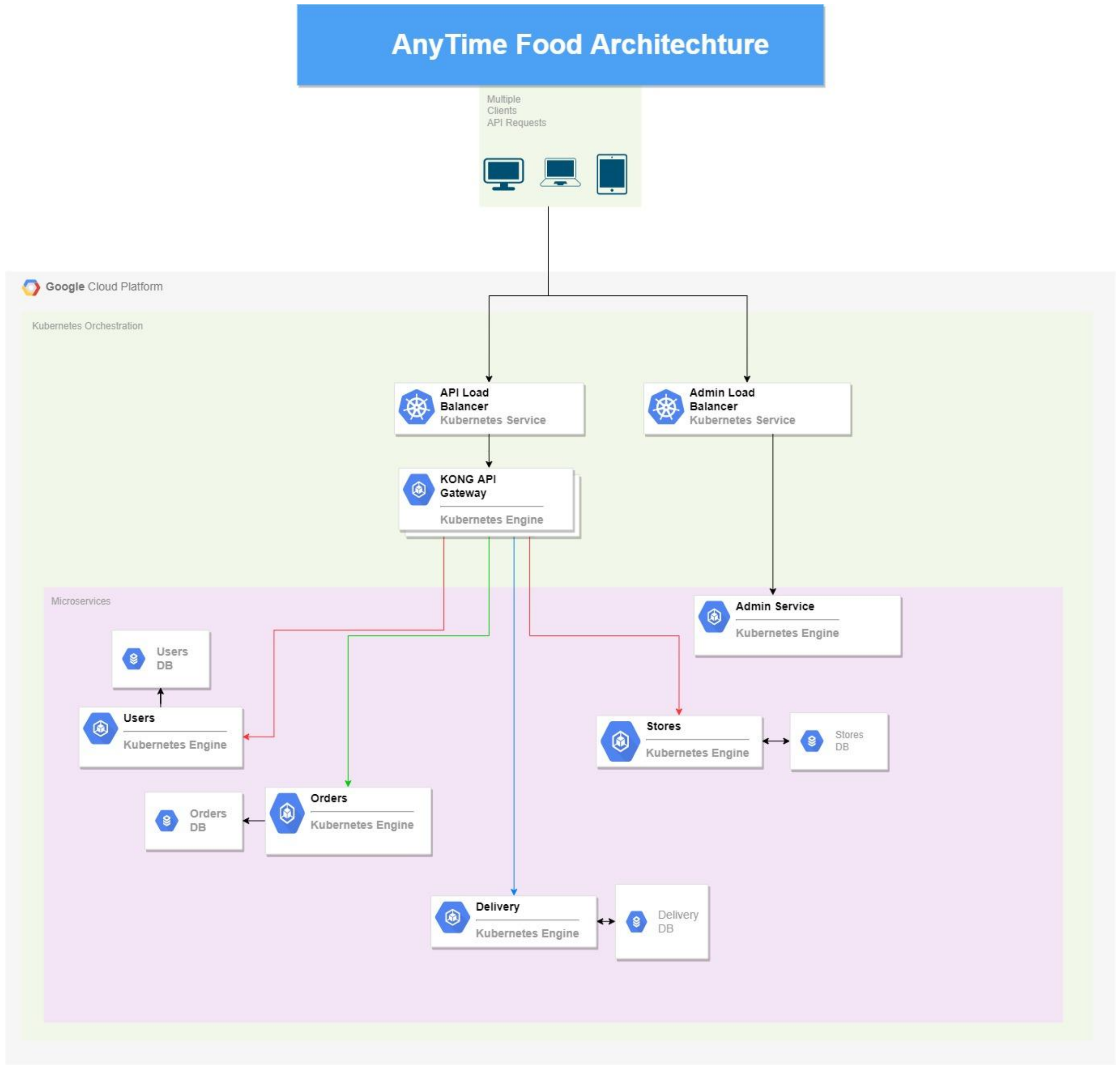
Appendix A: Glossary

Appendix B: Analysis Models

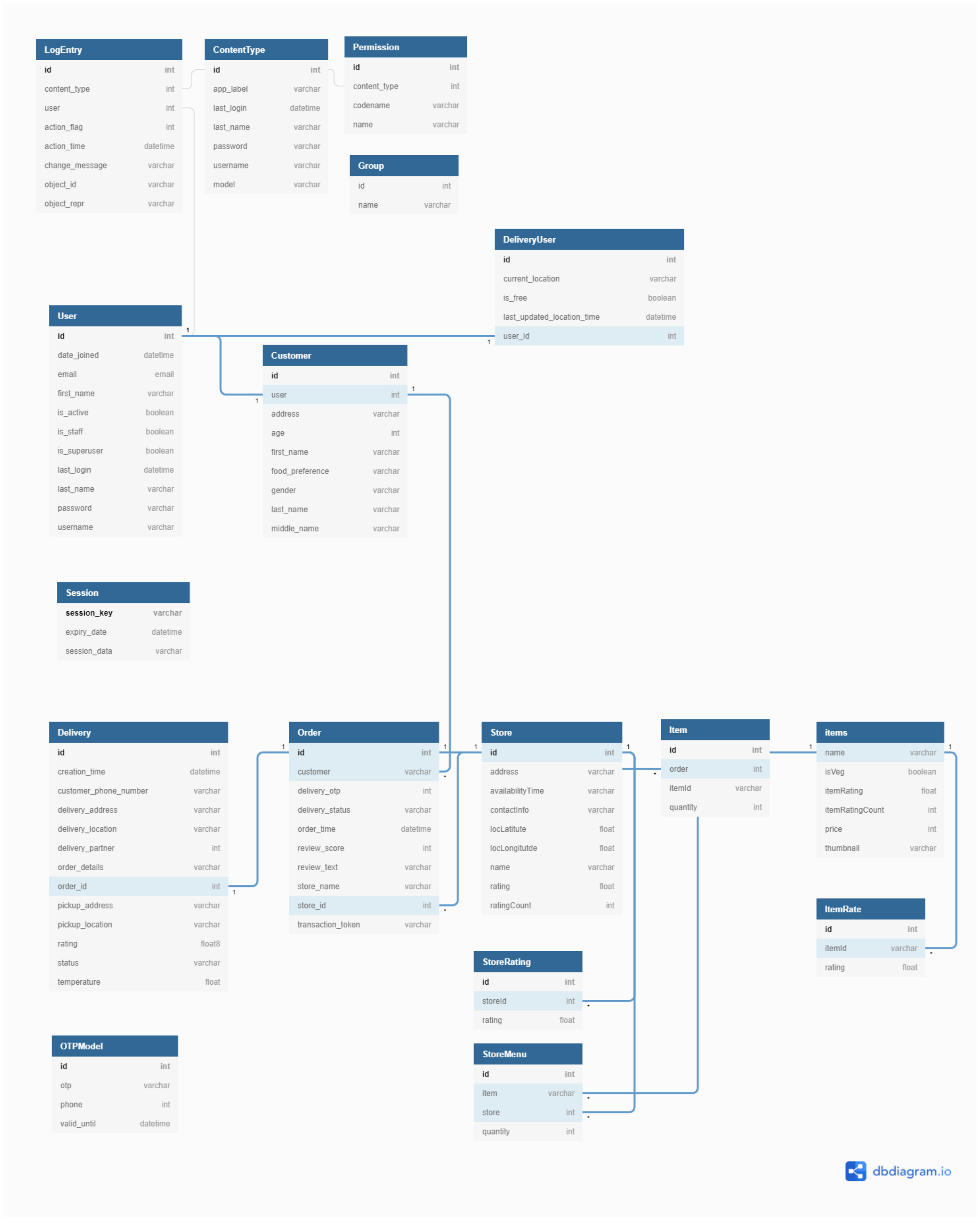
Control Flow Diagram :


Opensoft Control Flow Diagram
Team_909 | March 31, 2022
Start
Customer
Delivery Man
Staff
Login
Register
Yes
No
Browse Store Menu
Add Item to Cart (Order)
Place Final Order
Make Payment
View/Track Order(s)
Login
Register
Yes
No
Get Order Details
Get OTP from customer
OTP validated
Yes
No
Delivery success
Delivery Returned
Login
Register
No
Yes
Update Order Status
Add/Update Menu Item
Create Order on Uses behalf
Yes
Logout
Terminator

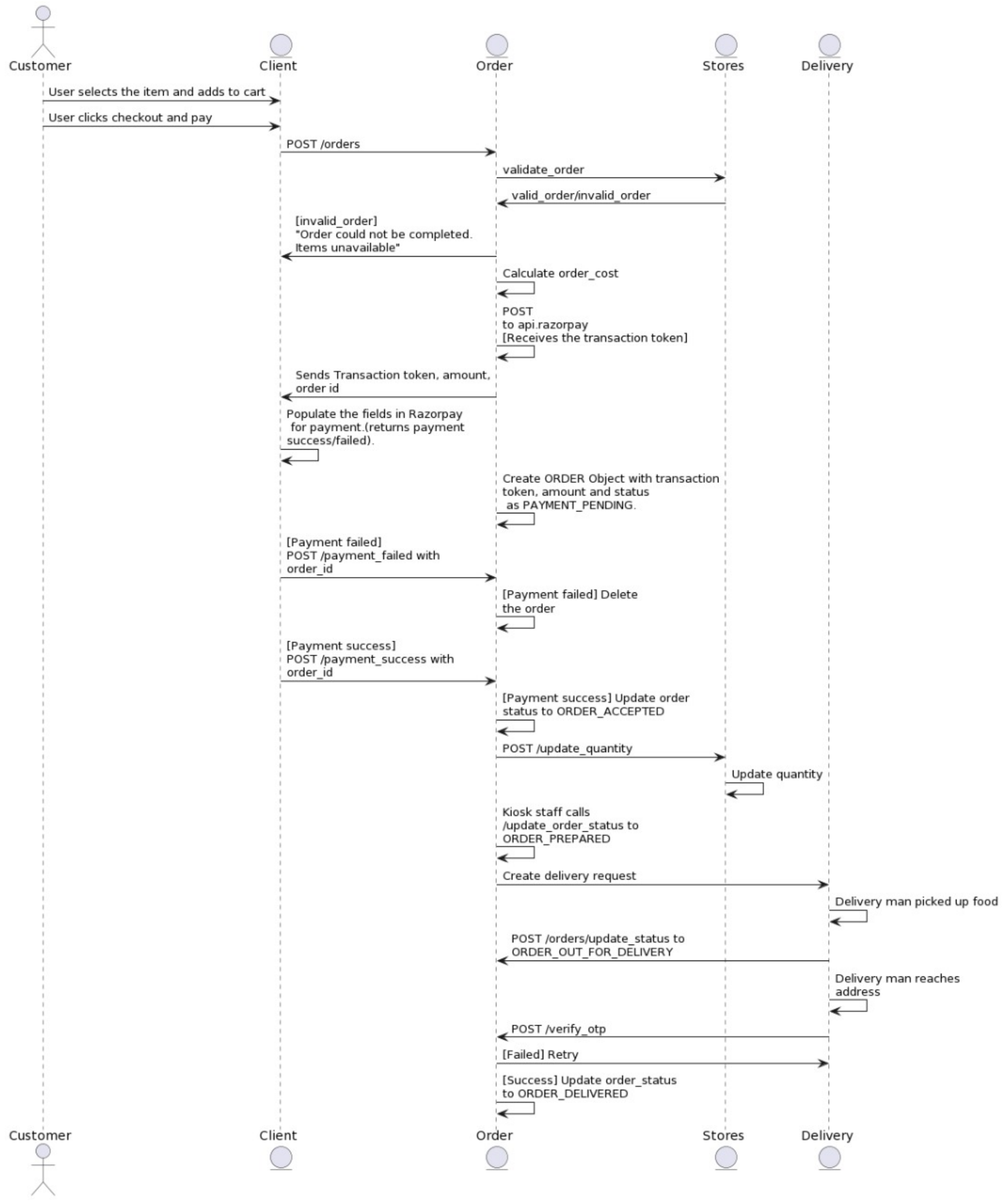
Data Flow Diagram :



Class Diagram :



Create Order Sequence Diagram :


Customer
Client
Order
Stores
Delivery
User selects the item and adds to cart
User clicks checkout and pay
POST /orders
validate_order
valid_order/invalid_order
[invalid_order] "Order could not be completed. Items unavailable"
Calculate order_cost
POST to api.razorpay [Receives the transaction token]
Sends Transaction token, amount, order id
Populate the fields in Razorpay for payment.(returns payment success/failed).
Create ORDER Object with transaction token, amount and status as PAYMENT_PENDING.
[Payment failed] POST /payment_failed with order_id
[Payment failed] Delete the order
[Payment success] POST /payment_success with order_id
[Payment success] Update order status to ORDER_ACCEPTED
POST /update_quantity
Update quantity
Kiosk staff calls /update_order_status to ORDER_PREPARED
Create delivery request
Delivery man picked up food
POST /orders/update_status to ORDER_OUT_FOR_DELIVERY
Delivery man reaches address
POST /verify_otp
[Failed] Retry
[Success] Update order_status to ORDER_DELIVERED
Customer
Client
Order
Stores
Delivery