Software Design Specification (SDS)

Project Name: Recipe to Cart

Prepared By: Youssef Mahmoud Youssef, Seif Ahmed, Amel Emad, Abdelrahman Elsayed

Date: [9/11/2024]

---

1. Introduction

1.1 Purpose

This document aims to clarify the design, architecture and technology behind “Recipe to Cart” application. This e-commerce site gives an opportunity to its customers to search for recipes, put the ingredients needed for them into a cart directly from the recipes, and be notified personally in regard to the offers for perishable goods. The service makes shopping more convenient.

1.2 Scope

This SDS covers the specification as well as implementation aspects of Recipe to Cart application. Major features include:

- Browsing through recipes
- Automatic addition of ingredients to the cart
- Tracking inventory

---

2. System Overview

The Recipe to Cart system consists of the following components:

- **Frontend**: Built with React, providing a user-friendly interface for browsing recipes and managing the shopping cart..
- **Backend**: Powered by Django, handling the business logic, data processing, and API endpoints.
- **Database**: MySQL, used for managing users, products, recipes, carts, and inventory.

---

3. System Architecture

3.1 Architectural Design

This project implements a client-server architecture with clearly defined frontend and backend in order to achieve modular and scalable architecture., where:

- **Frontend**: Interacts with users to display recipes, manage carts, and send requests.

- **Backend**: Processes requests, manages data, and communicates with the MySQL database.
- **Database**: Stores entities like Users, Products, Recipes, and Cart.

3.2 Data Flow

1. **User Interaction**: Users browse recipes and add ingredients directly to their cart.
2. **Request Processing**: it sends API requests to the back-end, through the front-end when a recipe is selected or items are added in the cart.
3. **Data Handling**: The back end receives the recipe data, adds items to the cart, and communicates with the inventory.
4. **Response**: The backend sends responses to update the frontend UI, showing cart status and notifications.

---

4. Database Design

4.1 Database Schema

The system will store data in a **relational database (MySQL)** with the following entities and relationships:

Table 1: Users

- **user_id:** Primary key, unique identifier for each user
- **username:** Username of the user
- **email:** Email address of the user
- **password:** Encrypted password for user authentication **Table 2**: Products

- **product_id:** Primary key, unique identifier for each product
- **name:** Name of the product
- **category:** Category of the product (e.g., dairy, vegetables)
- **price:** Price of the product
- **stock:** Quantity available in inventory

Table 3: Recipes

- **recipe_id: Primary key, unique identifier for each recipe**
- **title: Name of the recipe**

- **instructions: Step-by-step instructions for preparing the recipe**
- **user_id: Foreign key linking to Users table (indicating who added the recipe)**

Table 4: Recipe_Ingredients

- **recipe_ingredient_id: Primary key, unique identifier for each ingredient in a recipe**
- **recipe_id: Foreign key linking to Recipes table**
- **product_id: Foreign key linking to Products table**
- **quantity: Amount of the ingredient required for the recipe**

Table 5: Cart

- **cart_id:** Primary key, unique identifier for each cart
- **user_id:** Foreign key linking to Users table (indicating cart owner)

Table 6: Cart_Items

- **cart_item_id:** Primary key, unique identifier for each item in a cart
- **cart_id:** Foreign key linking to Cart table
- **product_id:** Foreign key linking to Products table
- **quantity:** Quantity of the product added to the cart

Relationships:

1. There is a one-to-many relationship between Users and Recipes because one user can create more than one recipe.
2. The association between Recipe and Recipe_Ingredients is one to many, because one recipe could have more than one ingredient.
3. One can say that Products relate to Recipe_Ingredients in many-to-many relations because one product can be an ingredient in several recipes, and one recipe can include several products as ingredients.
4. Users and Cart are in one-to-one relationships: one user has only one cart.
5. Cart and Cart_Items are in one-to-many relationships since each cart can have many items.

---

5. Technology Stack

- **Frontend:** React
- **Backend:** Django
- **Database:** MySQL

- **Push Notifications:** OneSignal
- **Version Control & CI/CD:** GitHub and GitHub Actions ● **Containerization:** Docker

---

6. Testing Plan

6.1 Unit Testing

Individual modules and functions will be unit-tested for the assurance of desired functionality.

6.2 Integration Testing

Integration tests will be performed to check whether the interaction between different modules, such as frontend and backend or backend and database, is correctly working.

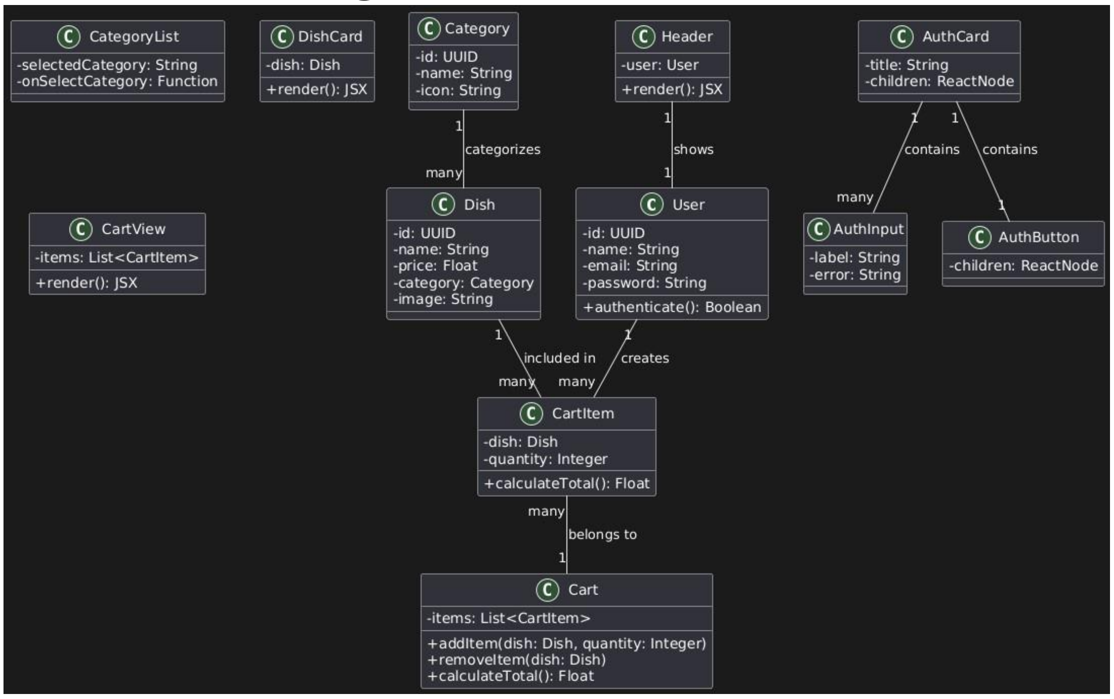
6.3 User Acceptance Testing (UAT)

The end users will be involved in testing the system to satisfy all requirements and expectations of the end users.

6.4 Performance Testing

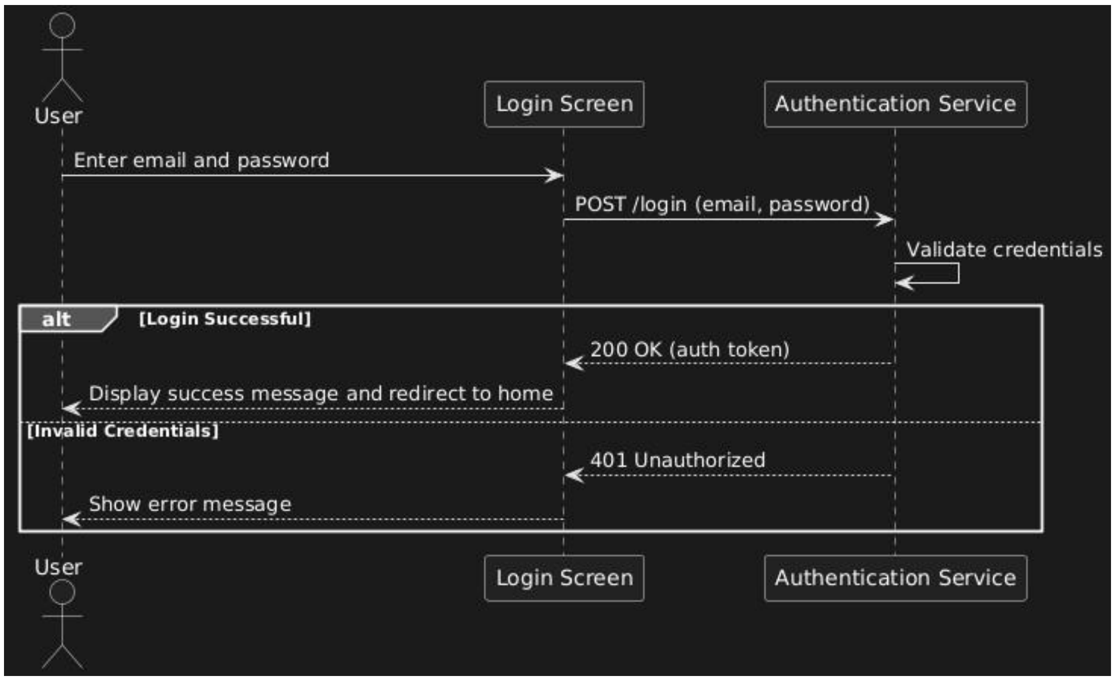
Stress testing and load testing will also be performed, which will ensure that the system is performing well without any degradation with the required number of users and operations.

7. UML Class Diagram

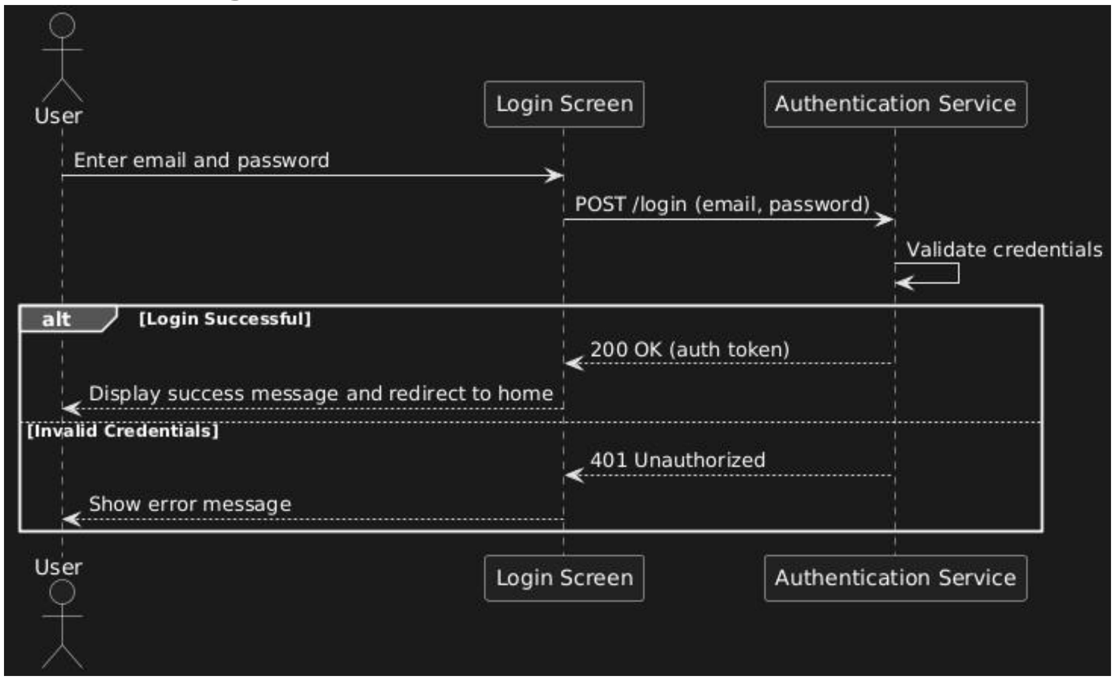


8. Sequence Diagrams

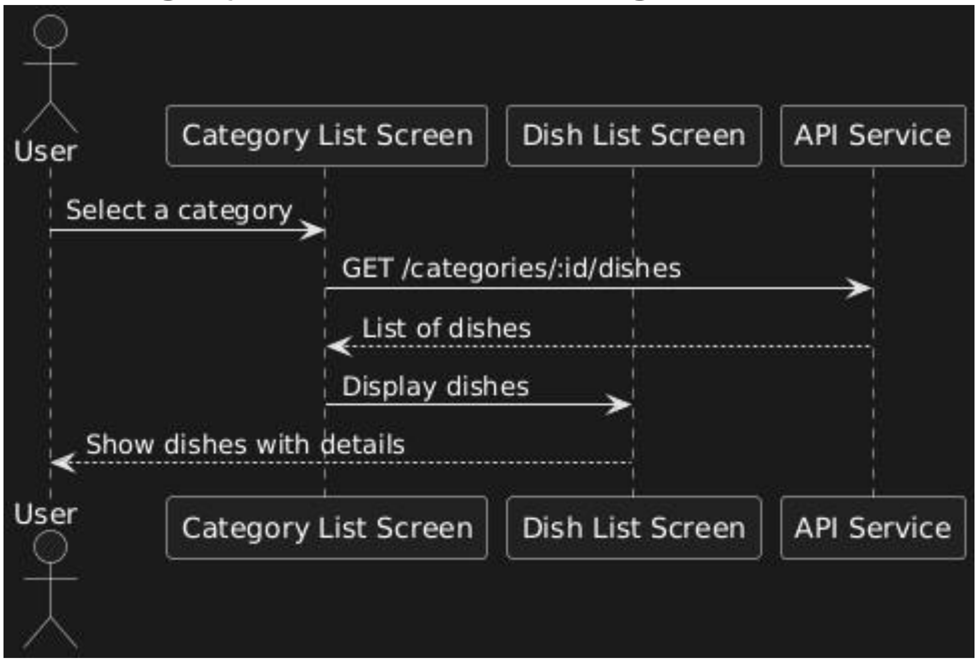
8.1 User Authentication



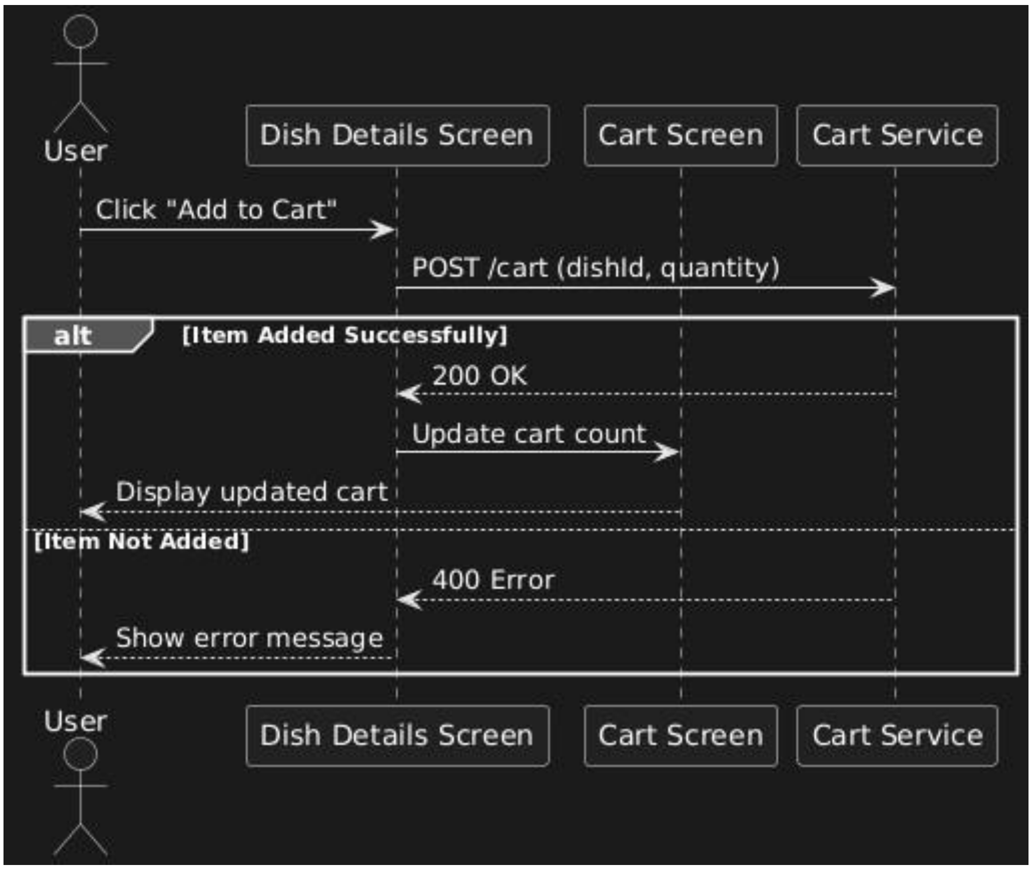
8.2 User Loguot



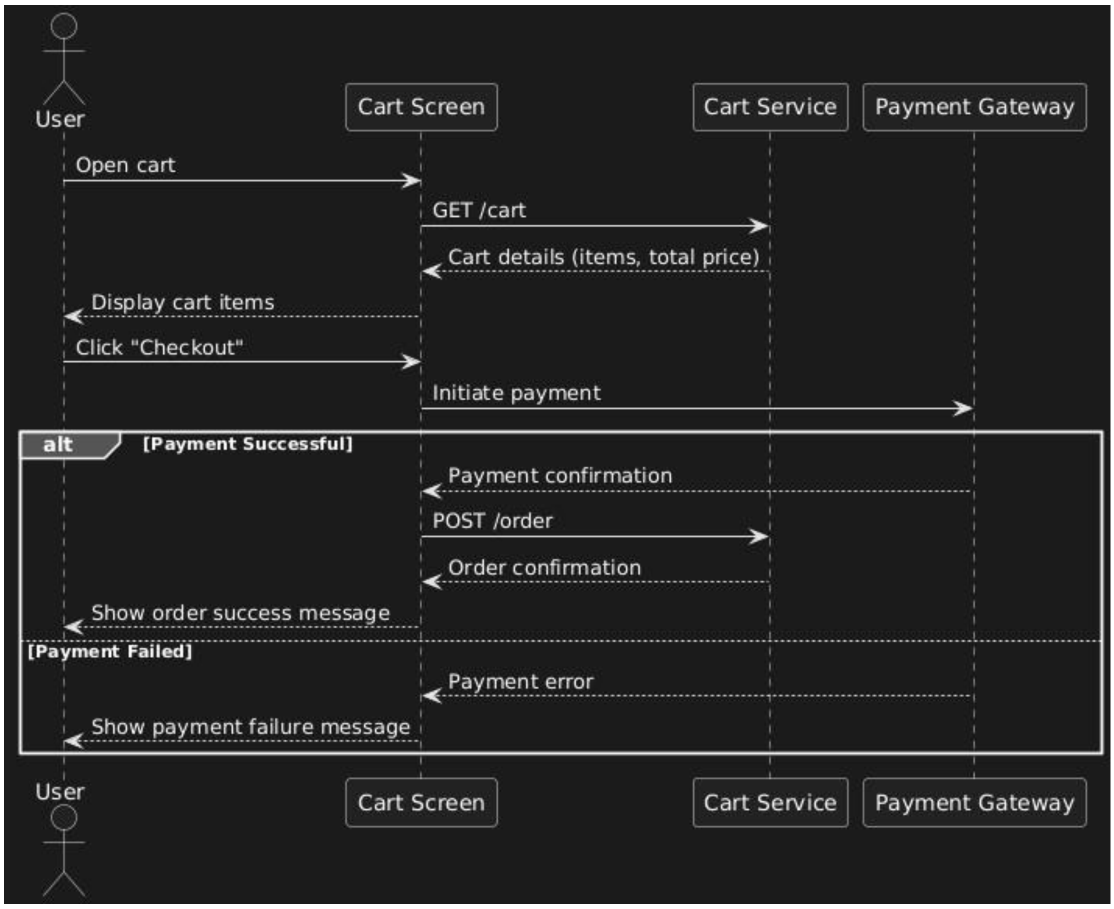
8.3 Category and Dish Browsing



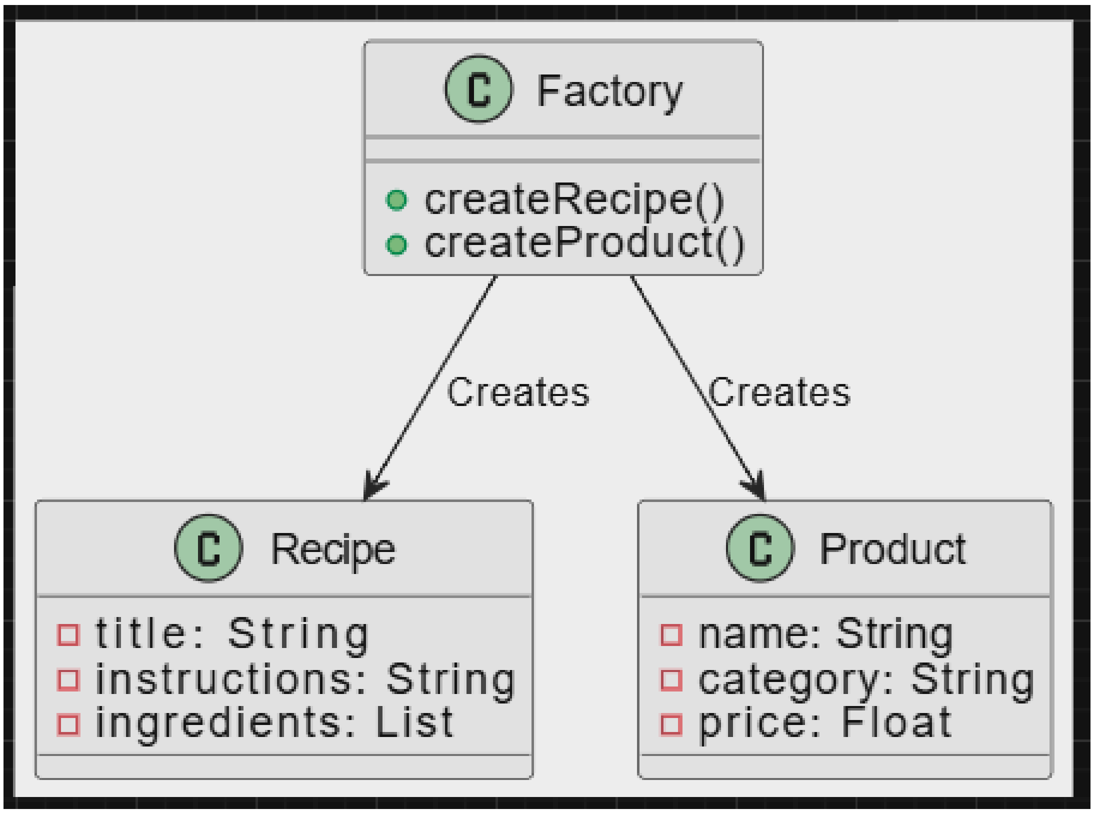
8.4 Add to Cart

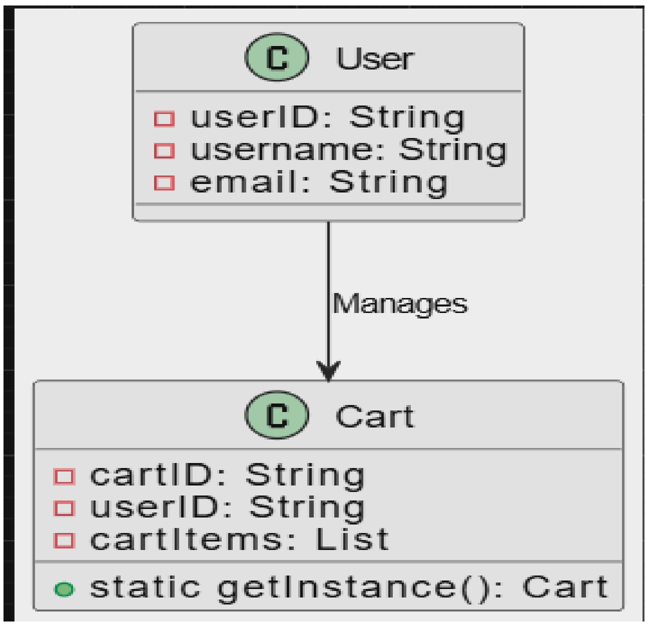


8.5 View Cart and Checkout

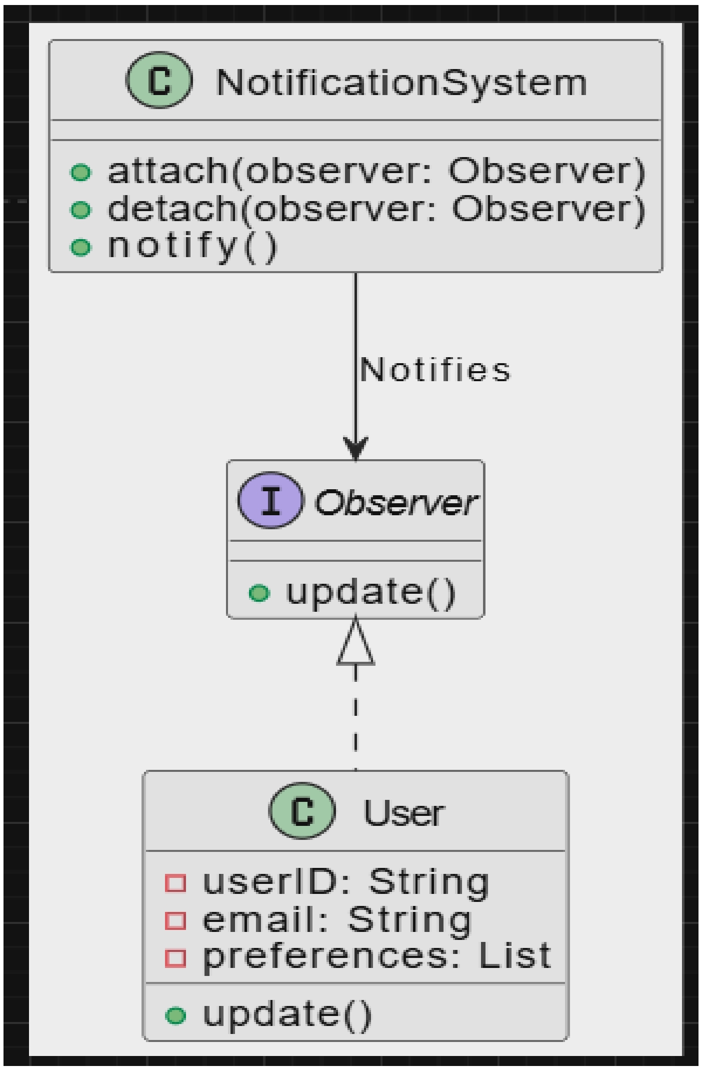


8.6 Creational Pattern

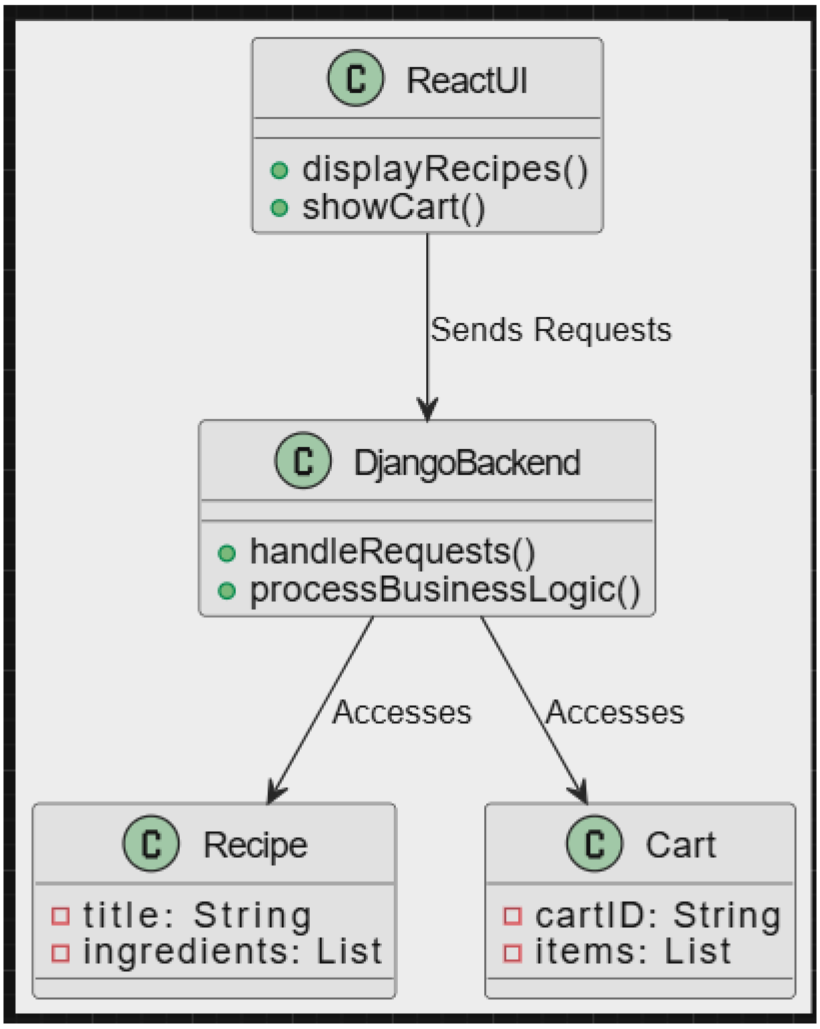




8.7 Behavioral Pattern



8.8 Structural Pattern



---

9. Conclusion

The Recipe to Cart application aims to provide a seamless recipe-based shopping experience. The design ensures scalability, modularity, and user-friendliness, making the shopping process more intuitive and engaging. The detailed specifications in this SDS will guide development to achieve these objectives.