Rehnuma Specifications Document

1. Introduction

1.1 Purpose

The purpose of this web application is to build an online system that enables users to generate their optimized grocery list which is cost and fuel-efficient.

1.2 Scope

The purpose of Rehnuma is to optimize grocery shopping by finding cost-effective routes based on supermarket prices, distances, and fuel costs. The system aims to improve convenience and savings for shoppers while maintaining an easy-to-use interface for people from all walks of life.

1.3 Document Conventions

- **API:** Application Programming Interface
- **UI/UX:** User Interface/User Experience
- **JWT:** JSON Web Token

1.4 References

- Project Proposal Document
- Technical Constraints and Non-Functional Requirements for Rehnuma
- Rehnuma Architecture Document
- Rehnuma API Documentation

2. Overall Description

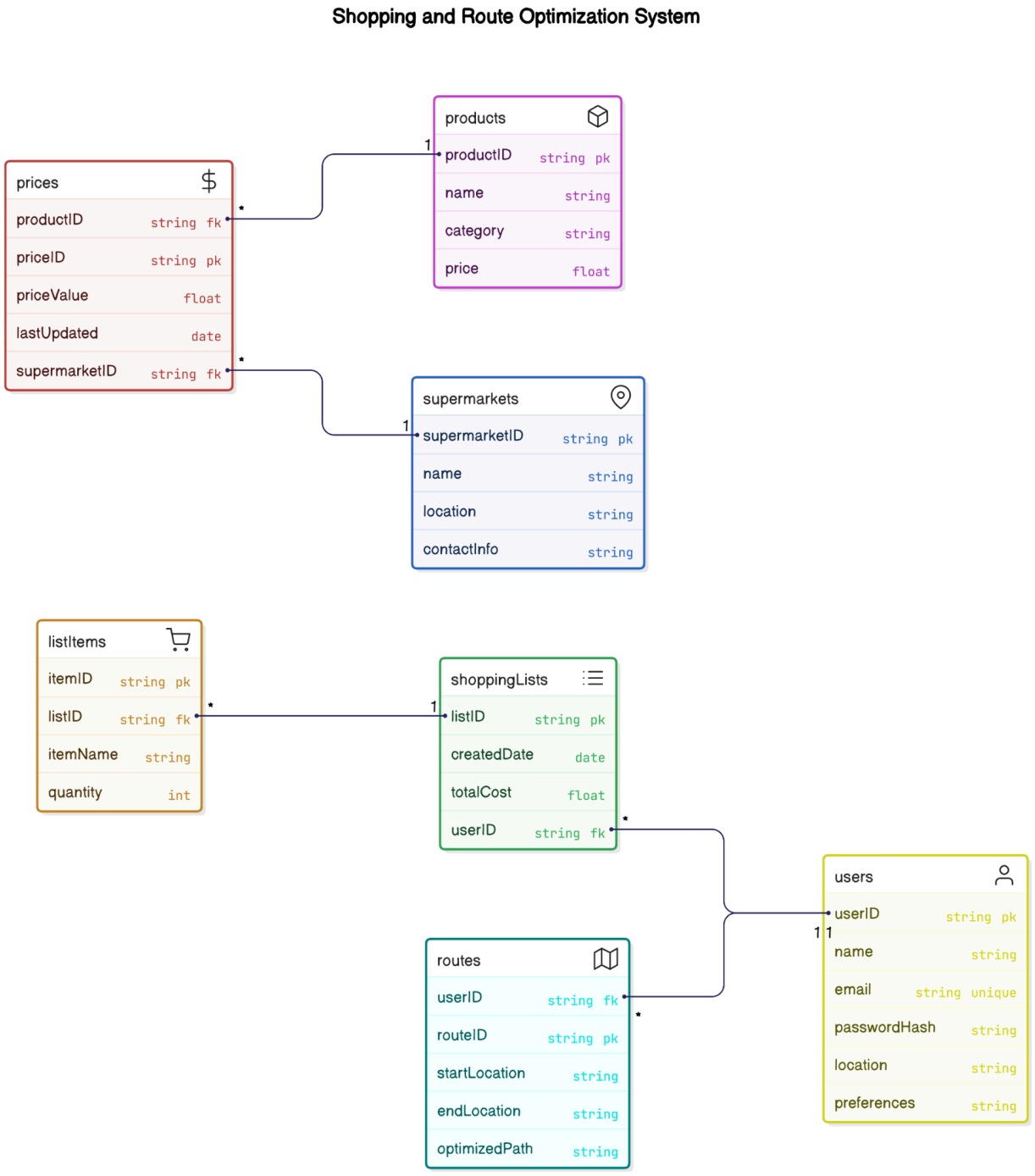
2.1 Product Perspective

Rehnuma scrapes various supermarket websites to fetch price and inventory data. The backend uses Node.js and processes this data, while the frontend (React.js) provides a seamless UI for users to interact with their shopping lists. The Python-based optimization algorithm optimizes routes and compares prices.

2.2 Product Features

- **User Registration and Login**: The product allows users to create and manage their accounts without any security hazards.
- **Shopping List Management**: The product will allow users to add, modify, or delete items from their grocery list.
- **Price Comparison**: The product will fetch supermarket data for prices to recommend the most low-cost items.
- **Route Optimization**: The product will suggest optimal shopping routes based on distance, fuel costs, and item prices.
- **History Lookup**: The product will allow users to store and view their previously generated itineraries.
- **Location Based SuperMarket Selection**: The supermarkets which are closer to the user should be considered.
- **SuperMarket Inventory Update**: Timely update of supermarket inventory and prices.

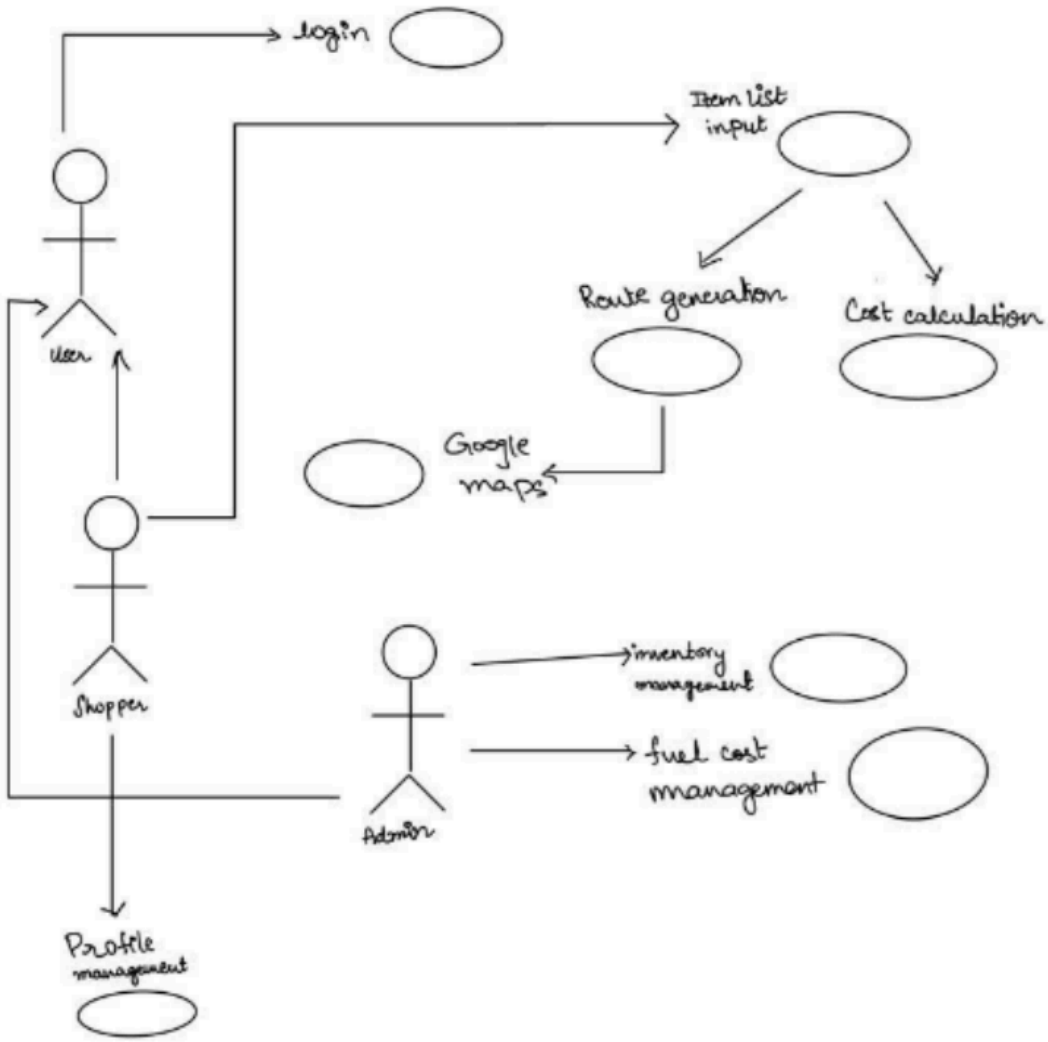
The major features of the Rehnuma database are shown below in the ER diagram.



2.3 User Classes and Characteristics

Shoppers using the system will be able to create their own profiles. Once they enter the system they'll be able to enter all the items they want to purchase and the application will find the most optimised itinerary for them based on cost and distance. Another form of users will be the administrators, they'll be responsible for managing and updating the information on the application.

- **Shoppers:** Primary users seeking to minimize grocery expenses.
- **Administrators:** Manage database updates and ensure data integrity.



2.4 Operating Environment

- Web browser
- Database: MongoDB
- Centralised Database
- Windows Operating system
- VS code

2.5 Design and Implementation Constraints

- Integration with Google Maps API
- Real-time data fetching from supermarket websites.
- SQL commands for data fetching.
- Implement the database using a centralized database management system.

2.6 Assumptions and Dependencies

- A reliable internet connection is required for users to access the application.
- Accurate data (Fuel and grocery pricing) for optimal functionality.
- The user knows the brands of products they wish to buy.
- The user approves for location tracking by the web application.

3. System Features

3.1 Functional Requirements

- **User Authentication:** The user should be able to sign up and log in smoothly.
- **Shopping Cart Management:** The user should be able to create, modify, and delete their shopping items.
- **Price Comparison:** The system should use the data from the database regarding the items and their prices and generate the most optimal cost-effective itinerary.
- **Route Optimization:** The application should generate optimized routes based on prices and distance.
- **Supermarket Data Updation:** The system will keep the data up to date for accuracy.
- **Error Handling:** Clear error messages will be generated for every invalid input and request.
- **History LookUp**: The user should be able to view their previously generate shopping plans
- **User Profile Management**: The user should be able to view and edit their personal information.
- **Customer Support**: The user should be able to send their queries to us via the web application and expect a response on their email.

4. EXTERNAL INTERFACE REQUIREMENTS

4.1 USER INTERFACES

- **Front-end software:** React.js (Web Application)
- **Back-end software:** Node.js for business logic and MongoDB for data storage

4.2 HARDWARE INTERFACES

- Windows operating systems
- A browser that supports HTML5, CSS3, and JavaScript

4.3 SOFTWARE INTERFACES

Software Used	**Description**
Operating system	Windows
Database	MongoDB
React.js	Frontend
Node.js	Backend
Python	Optimization algorithm

4.4 COMMUNICATION INTERFACES

- We will be using text boxes for taking grocery lists as input from users.
- Emails will be used to respond to customer queries and questions.

5. NON-FUNCTIONAL REQUIREMENTS

5.1 Software Quality Attributes

- **Performance:** Optimized routes should be generated within 8-10 seconds.
- **Scalability:** The system should be able to support up to 500 concurrent users.
- **Security:** The system should use OAuth/JWT-based authentication.
- **Availability:** The system should achieve 85-90% uptime.
- **Maintainability:** The code should maintain a modular structure for easier updates and scalability.