**Food Delivery Website Documentation**

**Table of Contents**

1. [Introduction](#introduction)
2. [Technologies Used](#technologies-used)
3. [Installation Guide](#installation-guide)
4. [Project Structure](#project-structure)
5. [Frontend](#frontend)
6. [Backend](#backend)
   * [Environment Variables](#environment-variables)
   * [API Endpoints](#api-endpoints)
7. [Admin Dashboard](#admin-dashboard)
8. [Features](#features)
9. [Screenshots](#screenshots)
10. [Future Improvements](#future-improvements)

**1. Introduction**

The **Food Delivery Website** is a full-stack project that allows users to browse through food items, add them to their cart, and place an order. The application supports user authentication and includes an admin panel to manage food items and track customer orders.

The project consists of two main parts:

* **Frontend**: Built using **React.js**, allows users to interact with the system.
* **Backend**: Built using **Node.js**, **Express.js**, and **MongoDB**, handles all the business logic, data management, and authentication.

**2. Technologies Used**

**Frontend:**

* **React.js**: Main framework for building the user interface.
* **Axios**: To handle HTTP requests.
* **React Router**: For navigation.
* **CSS**: For styling the frontend.

**Backend:**

* **Node.js**: JavaScript runtime environment for the server.
* **Express.js**: Web framework for Node.js to handle API requests.
* **MongoDB**: NoSQL database for storing user and order data.
* **Mongoose**: ODM for MongoDB to structure schemas.
* **JWT (JSON Web Tokens)**: For secure user authentication.
* **BCrypt**: For password encryption.

**3. Installation Guide**

Follow these steps to install and run the project locally.

**Clone the Repository**

bash

Copy code

git clone https://github.com/Dipesh-Arwat/Food-Delivery-Website.git

cd Food-Delivery-Website

**Frontend Setup:**

Navigate to the frontend directory, install dependencies, and start the development server.

bash

Copy code

cd frontend

npm install

npm run dev

The frontend will run at http://localhost:3000.

**Backend Setup:**

Navigate to the backend directory, install dependencies, configure environment variables, and start the server.

bash

Copy code

cd backend

npm run server

Create a .env file in the backend directory to store environment variables:

bash

Copy code

MONGO\_URI=<Your MongoDB URI>

JWT\_SECRET=<Your JWT Secret>

PORT=4000

Start the backend server:

bash

Copy code

npm run server

The backend will run at http://localhost:4000.

**4. Project Structure**

bash

Copy code

Food-Delivery-Website/

│

├── backend/ # Backend server files

│ ├── models/ # Mongoose models

│ ├── routes/ # Express routes for API endpoints

│ ├── controllers/ # Controller functions for API

│ ├── middlewares/ # Authentication middleware (JWT)

│ ├── server.js # Main server file

│ └── .env # Environment variables (not in repo)

│

├── frontend/ # Frontend client files

│ ├── src/

│ │ ├── components/ # React components

│ │ ├── pages/ # Different pages (Home, Cart, Admin)

│ │ ├── App.js # Main app file

│ └── public/ # Public files (HTML, images, etc.)

│

└── README.md # Project documentation

**5. Frontend**

The frontend is built with **React.js** and manages the user interaction. It includes multiple components like:

* **Home Page**: Displays all available food items.
* **Cart**: Allows users to view items in their cart.
* **Authentication Pages**: For user signup and login.
* **Order Page**: Users can place and track their orders.

**Key Components:**

* **Header**: Contains navigation links.
* **FoodList**: Displays all food items fetched from the backend.
* **Cart**: Manages the user’s order summary.
* **Order**: Displays order details and status updates.

**Dependencies:**

* React Router for navigation between pages.
* Axios for API requests to the backend.

**6. Backend**

The backend is built with **Node.js** and **Express.js**. It handles the following:

* User authentication.
* Managing food items and orders.
* Secure JWT-based authentication.
* Connecting with MongoDB for data storage.

**6.1 Environment Variables**

The backend uses the following environment variables:

plaintext

Copy code

MONGO\_URI=<Your MongoDB connection string>

JWT\_SECRET=<A secure secret key for JWT tokens>

PORT=5000 (or any preferred port)

**6.2 API Endpoints**

**User Authentication**

* **POST /api/auth/register**: Register a new user.
  + Request body: { "name", "email", "password" }
  + Response: User object and JWT token.
* **POST /api/auth/login**: Log in an existing user.
  + Request body: { "email", "password" }
  + Response: JWT token.

**Food Items**

* **GET /api/food**: Get all available food items.
  + Response: List of food items.
* **POST /api/food** (Admin only): Add a new food item.
  + Request body: { "name", "price", "category", "image" }
  + Response: Created food item.
* **PUT /api/food/**

(Admin only): Update food item details.

* + Request body: Updated food object.
* **DELETE /api/food/**

(Admin only): Delete a food item.

**Orders**

* **POST /api/orders**: Place a new order.
  + Request body: Order details.
  + Response: Created order.
* **GET /api/orders/**

: Get details of a specific order.

* **GET /api/orders/user/**

: Get all orders for a specific user.

**7. Admin Dashboard**

The admin dashboard allows the admin to:

* Add, edit, and delete food items.
* View all customer orders.
* Update the delivery status of orders.

**Admin Features:**

* Secure login.
* Manage products (CRUD operations).
* View and update orders.

**8. Features**

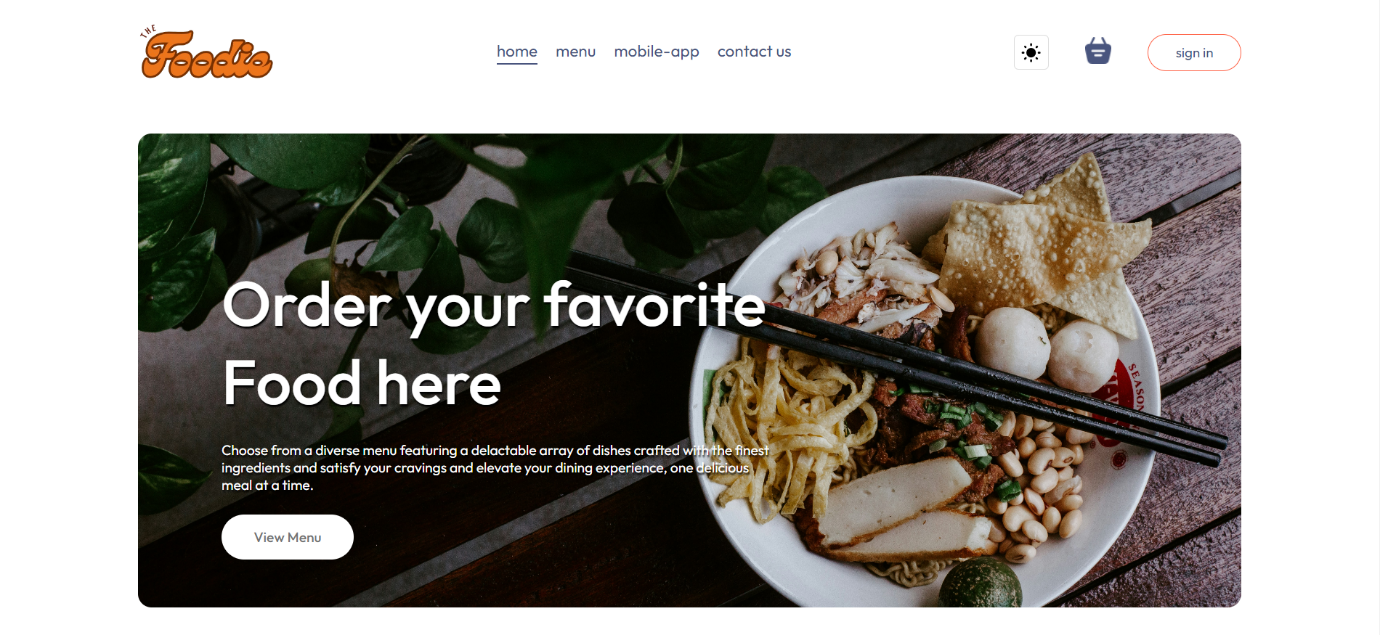
**For Users:**

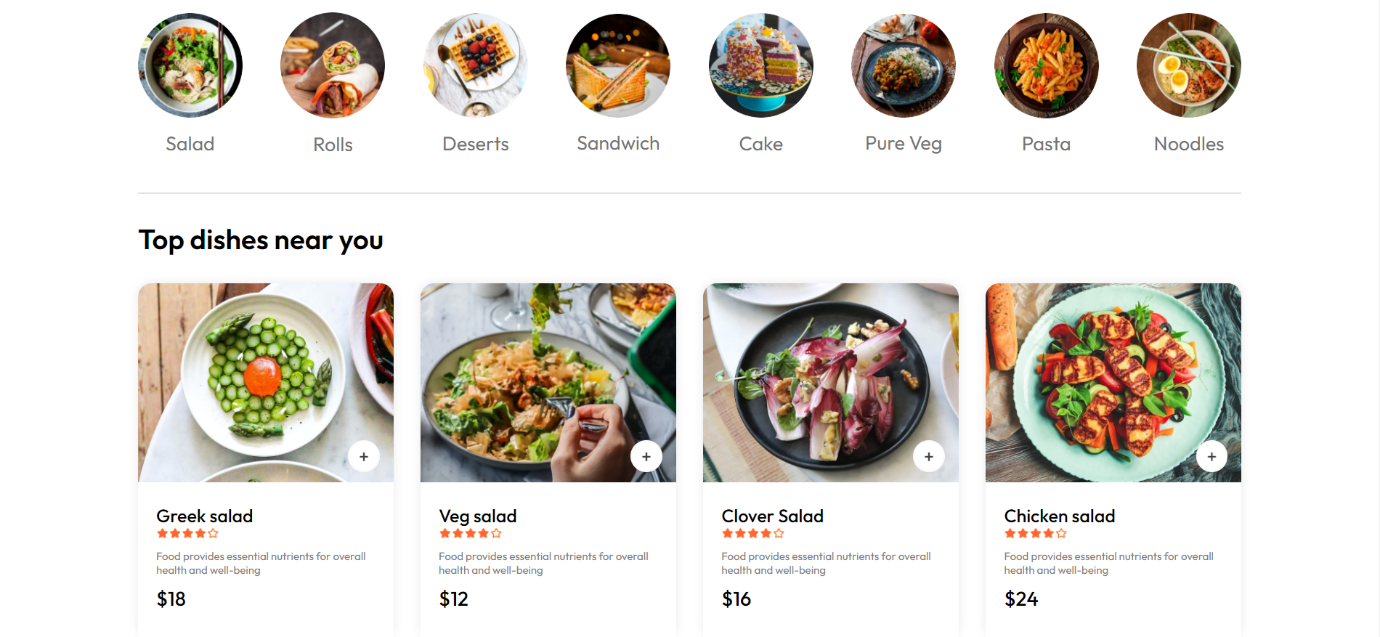
* Secure registration and login.
* Browse available food items.
* Add items to the cart and place orders.
* View order summary and order status.

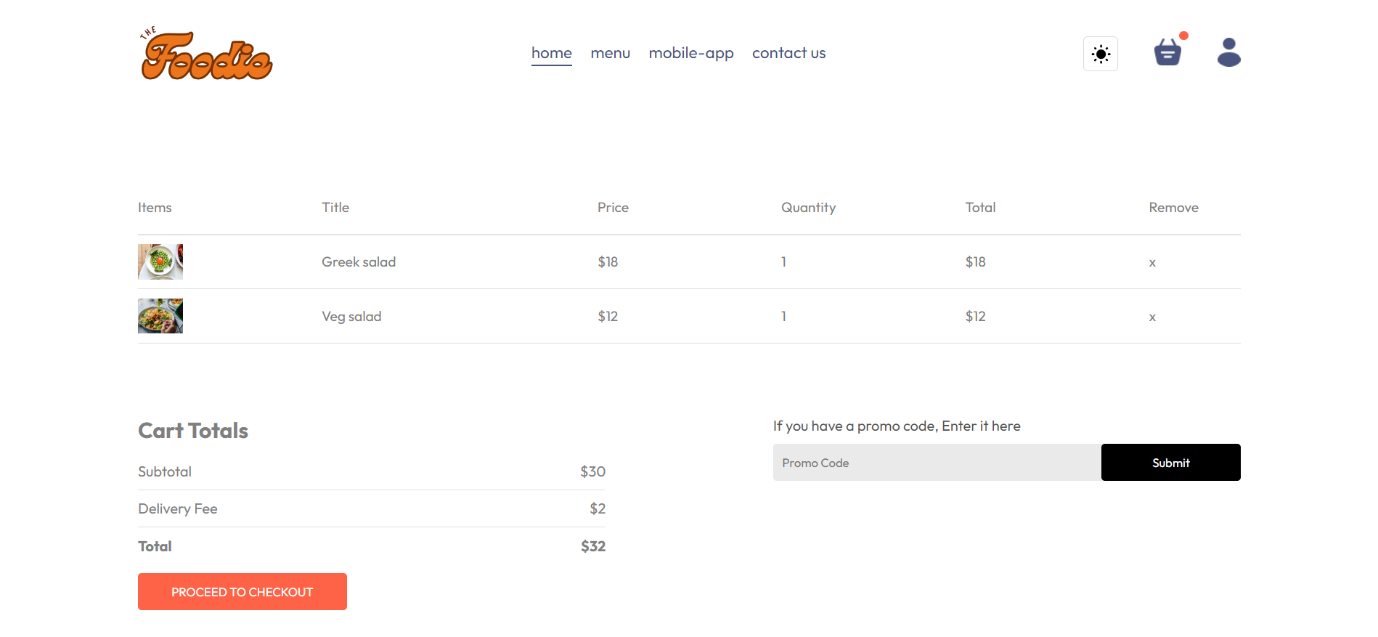
**For Admin:**

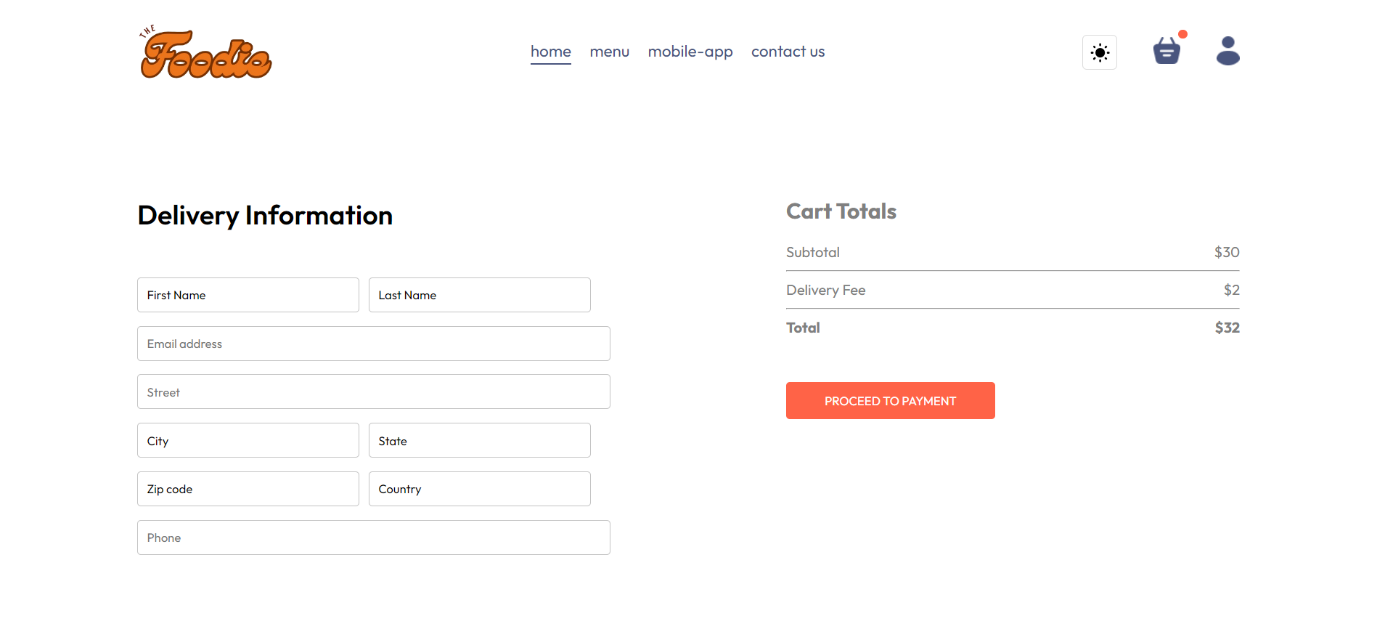
* Manage (add, edit, delete) food items.
* View all customer orders.
* Update order delivery status.

**9. Screenshots**

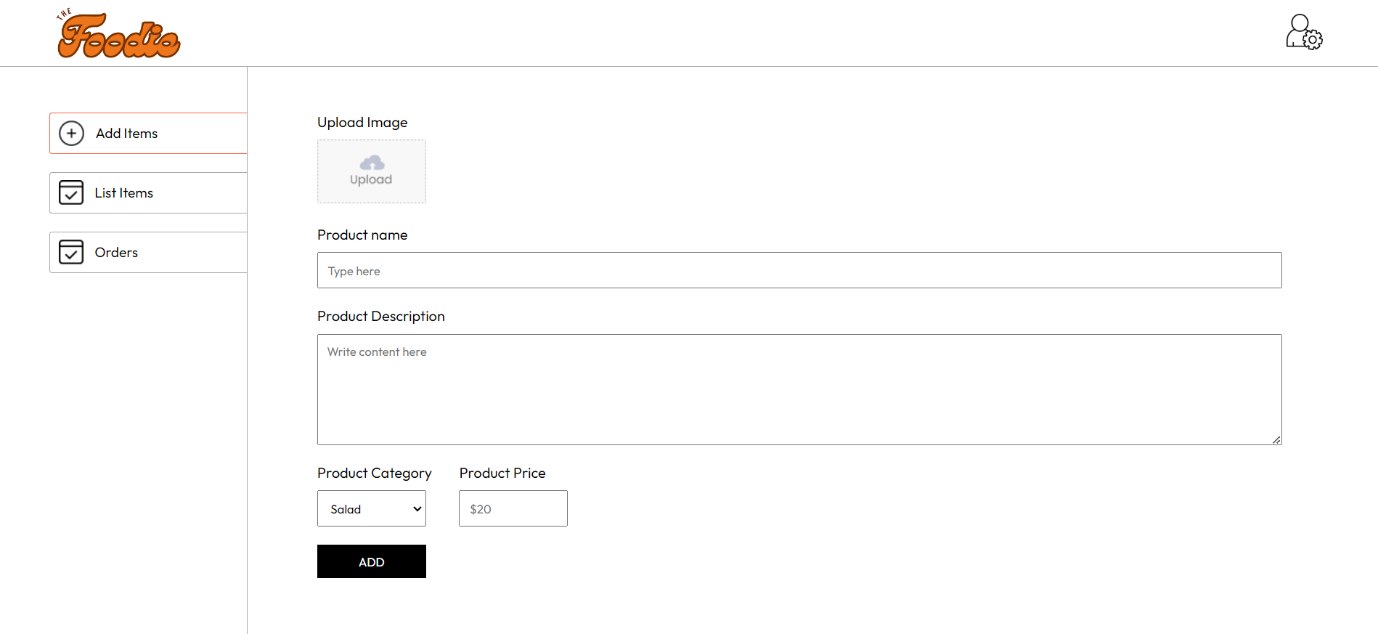
****

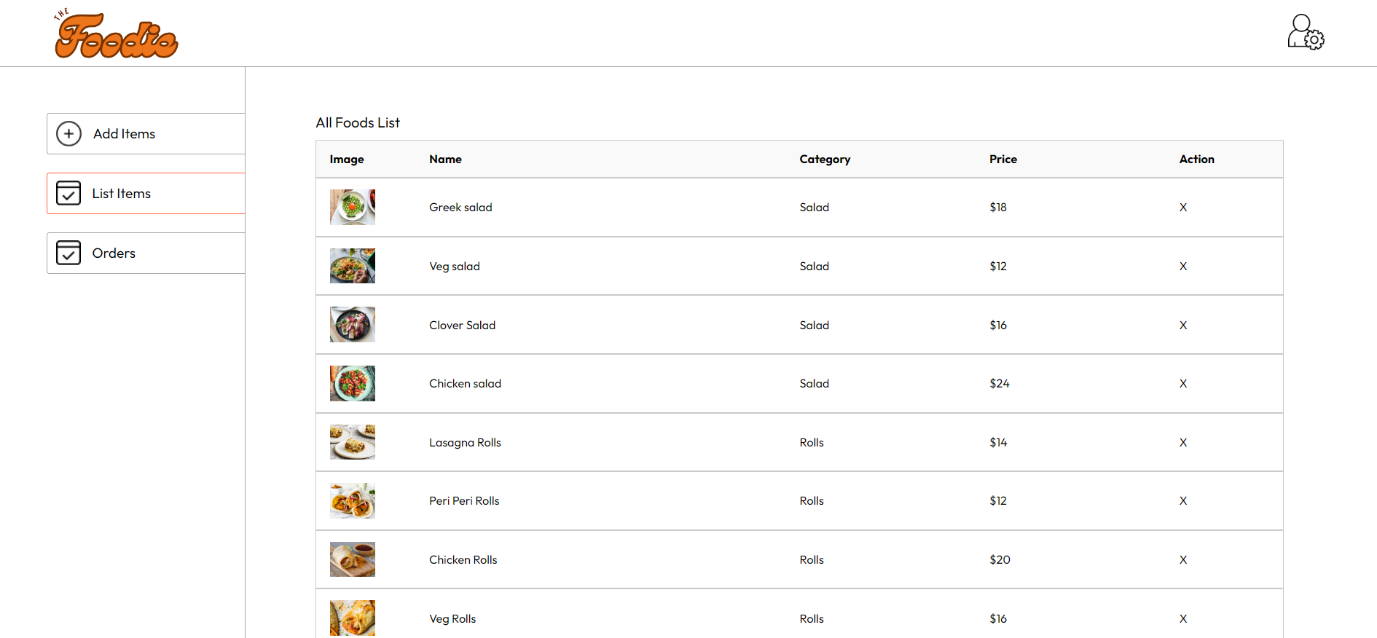
****

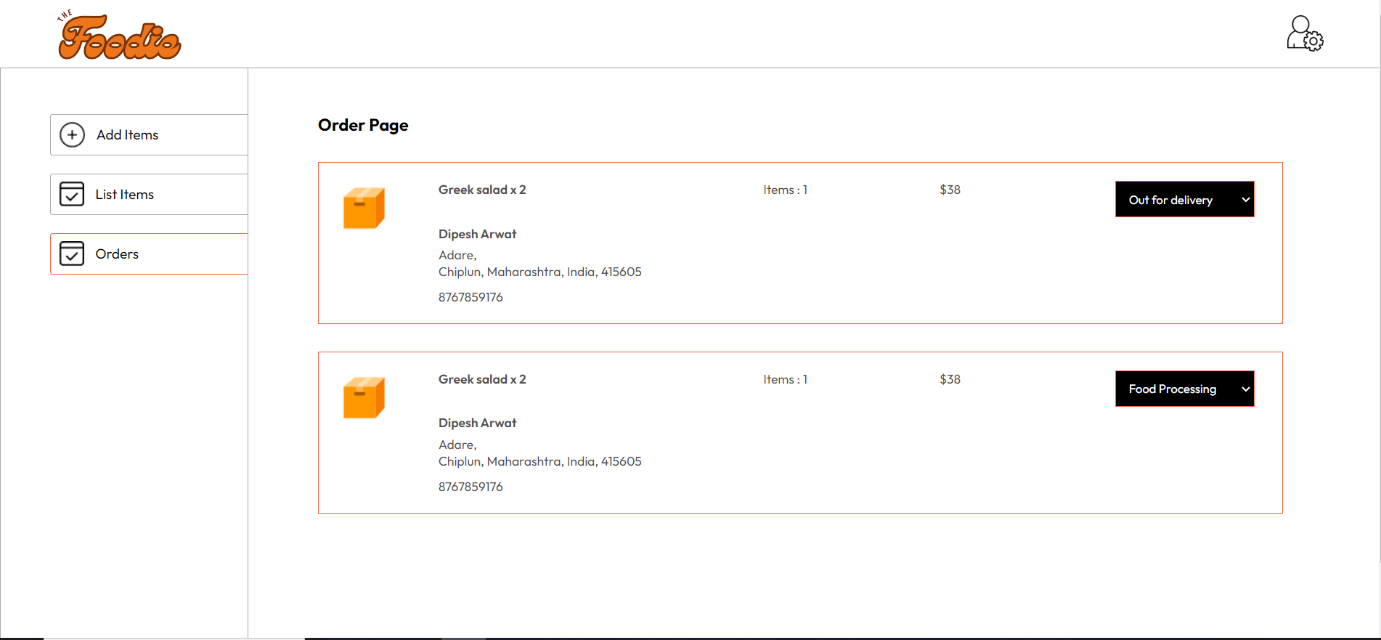
****

****

**Admin Dashboard :**

****

****

****

**10. Future Improvements**

* Implement real-time order tracking using WebSockets.
* Add payment gateway integration (e.g., Stripe).
* Provide notifications for order status updates.
* Improve the admin dashboard with better analytics.