FOOD ORDER DELIVERY FULL STACK DEVELOPMENT WITH MERN

PROJECT DOCUMENTATION

1.Introduction

- **Project Title: FOOD ORDER DELIVERY APPLICATION**
- Team Members:

1.Harsha vardhini V – 2021503512

2.Vithya S – 2021503572

3.Preetha Raai RS – 2021503320

4.Puja Sree A - 2021503540

2.Project Overview

Purpose:

The Food Order Delivery Application is designed to facilitate the ordering of food from various restaurants in an organized and user-friendly environment. Users have access to a variety of features that enhance their ordering experience.

Features:

- Show Restaurants: Users can browse a collection of restaurants presented in visually appealing cards and tables, making it easy to find and explore various dining options.
- Create New Menu Items: Admin users have the ability to add new food items to the restaurant menus, ensuring a continuous and diverse selection for users.
- Edit Existing Menu Items: Authorized users can modify details of existing food items, such as name, ingredients, price, ensuring accurate and up-to-date information.
- Delete Menu Items: Users with appropriate permissions can remove food items from the menus, helping maintain the relevance of the offerings.
- Show Description: Each food item displays a detailed description, including ingredients, reviews, and other pertinent information to assist users in their decisions.

3.Architecture

Frontend:

Built with React, the frontend architecture utilizes a component-based structure, enabling reusable components for food cards, forms, and navigation. The application state is managed with hooks and context, while Axios handles API requests to the backend.

Backend:

The backend is developed using Node.js and Express.js, structured to provide RESTful API endpoints that facilitate CRUD operations for menu items and user management. Middleware is implemented for handling authentication and validation of requests.

Database:

The application uses MongoDB as the database, with a schema that includes collections for users and menu items. Mongoose is used to model the data and simplify interactions with the database, providing a clear structure for data retrieval and manipulation.

4.Setup Instructions

Prerequisites:

Node.js (v16 or higher)

MongoDB (local installation or MongoDB Atlas)

npm (Node package manager)

Installation:

i. Navigate to the project directory:

cd food-order-delivery

ii. Install dependencies for both the client and server:

cd client

npm install

cd ../server

npm install

iii. Set up environment variables:

Create a .env file in the server directory and add necessary variables such as:

MONGO_URI=your_mongo_uri

JWT_SECRET=your_jwt_secret

Folder Structure

Client:

/client

├── /public

├── /src

│ ├── /components // Reusable UI components (FoodCard, Header, etc.)

│ ├── /pages // Main application pages (Home, Menu, etc.)

│ ├── /context // Context and hooks for state management

│ ├── App.js

│ └── index.js

└── package.json

Server:

/server

├── /models // Mongoose models for users and menu items

├── /routes // Routes for API endpoints

├── /config // Configuration files, e.g., database connection

├── /controllers // Controllers for handling business logic

├── server.js // Main entry point for the server

└── package.json

6.Running the Application

Frontend:

Navigate to the client directory and run:

npm start

Backend:

Navigate to the server directory and run:

npm start

7.API Documentation

Endpoints:

- **GET /api/menu**
  Description: Retrieve all menu items
  Response: JSON array of menu item objects
- **POST /api/menu**
  Description: Add a new food item to the menu
  Request Body: { name, ingredients, price, description }
  Response: Confirmation message and added menu item object
- **PUT /api/menu/:id**
  Description: Update an existing menu item's details
  Request Body: { name, ingredients, price, description }
  Response: Confirmation message and updated menu item object
- **DELETE /api/menu/:id**
  Description: Remove a food item from the menu
  Response: Confirmation message

8.Authentication

Authentication is managed through JWT (JSON Web Tokens). After successful login, a token is created and stored in local storage. This token is sent with subsequent requests for

protected routes to verify user identity and permissions, allowing access to CRUD operations on menu items.

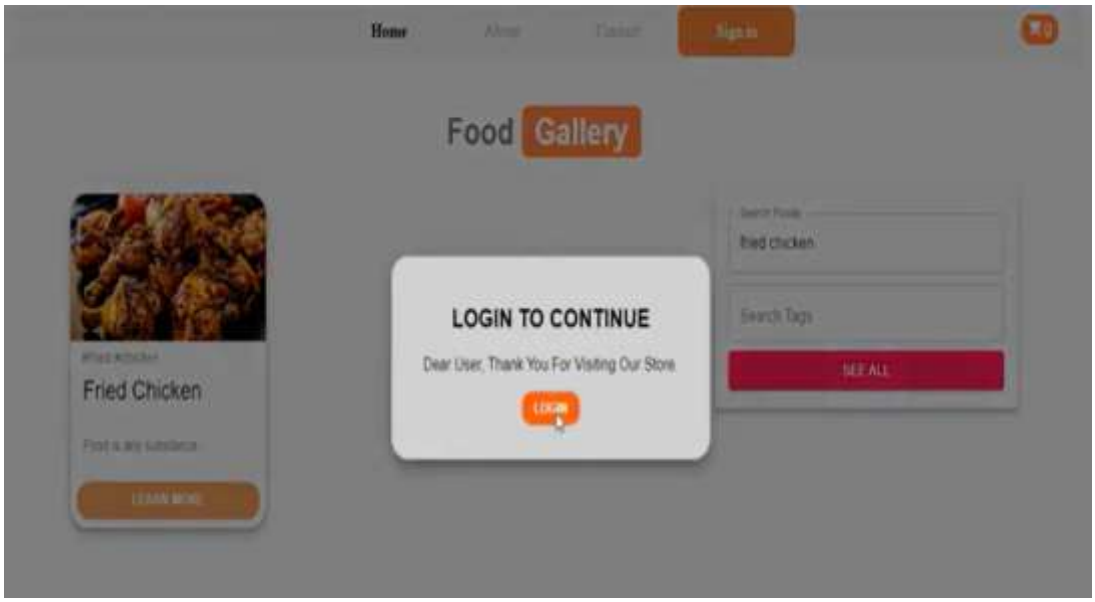
9.User Interface

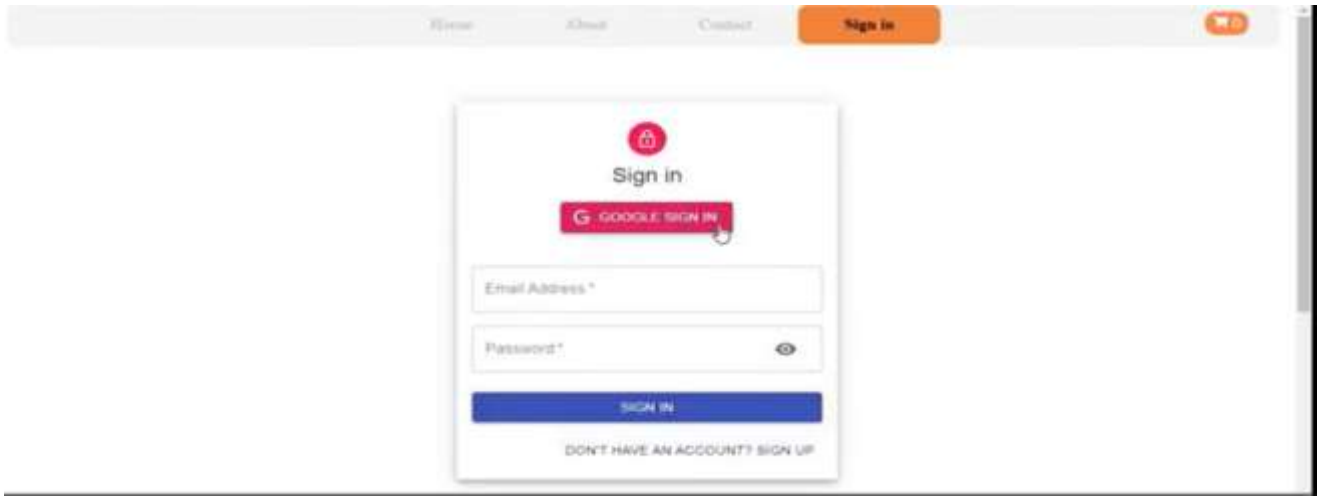
Screenshots or Demo

Demo Link: MERN-DEMO-FOOD ORDER DELIVERY -

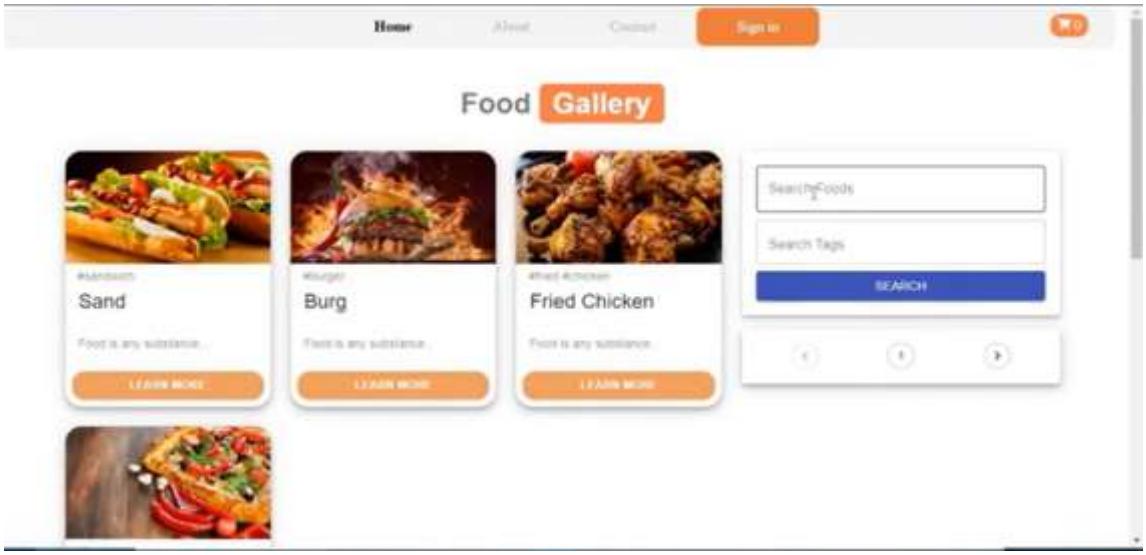
Screenshots:

SIGN IN/SIGN UP:

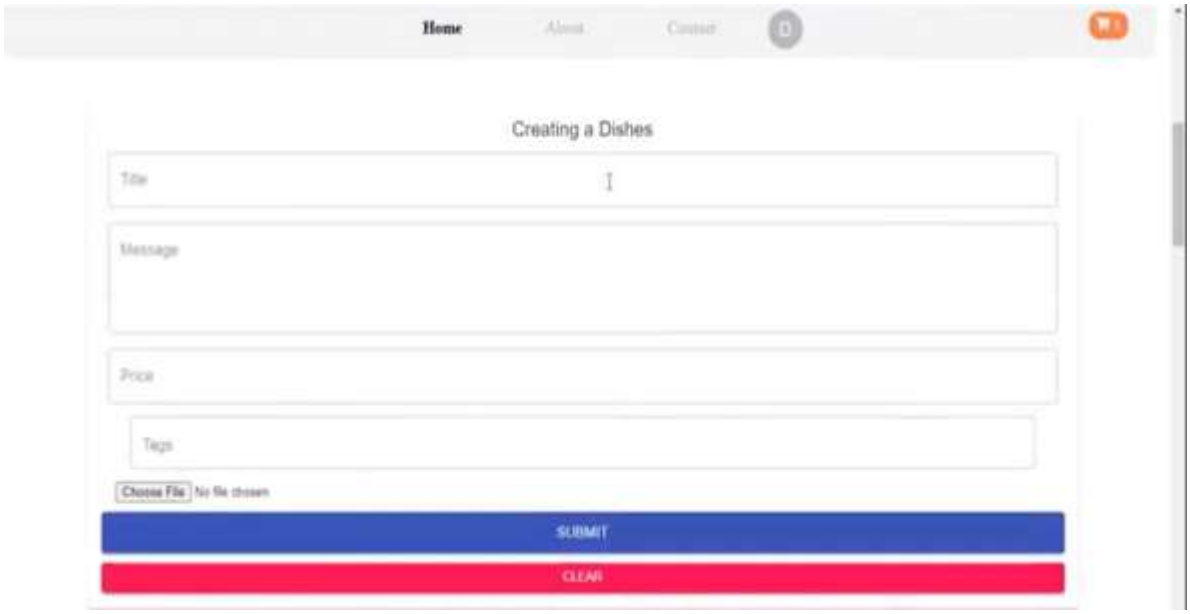




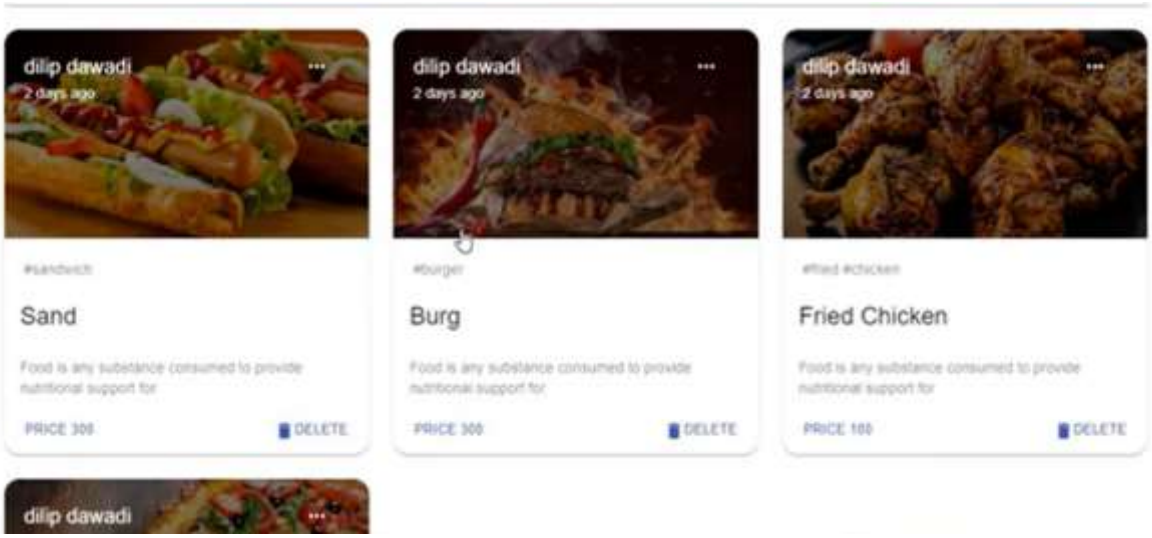
HOME PAGE:



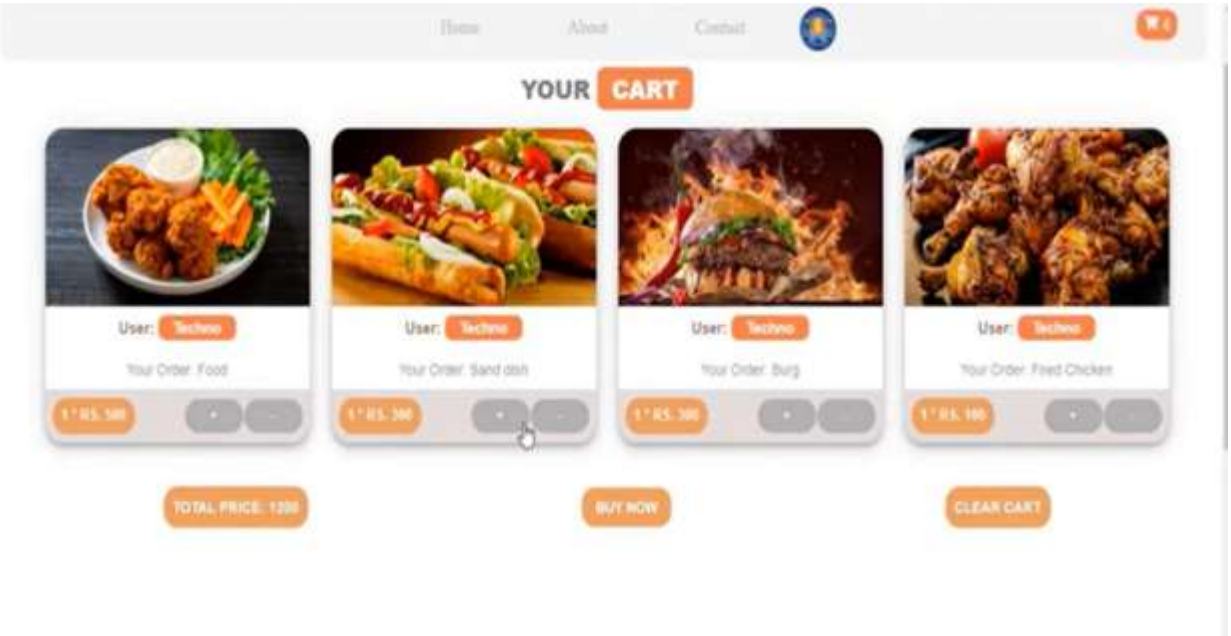
ADDING FOOD ITEMS IN APP:



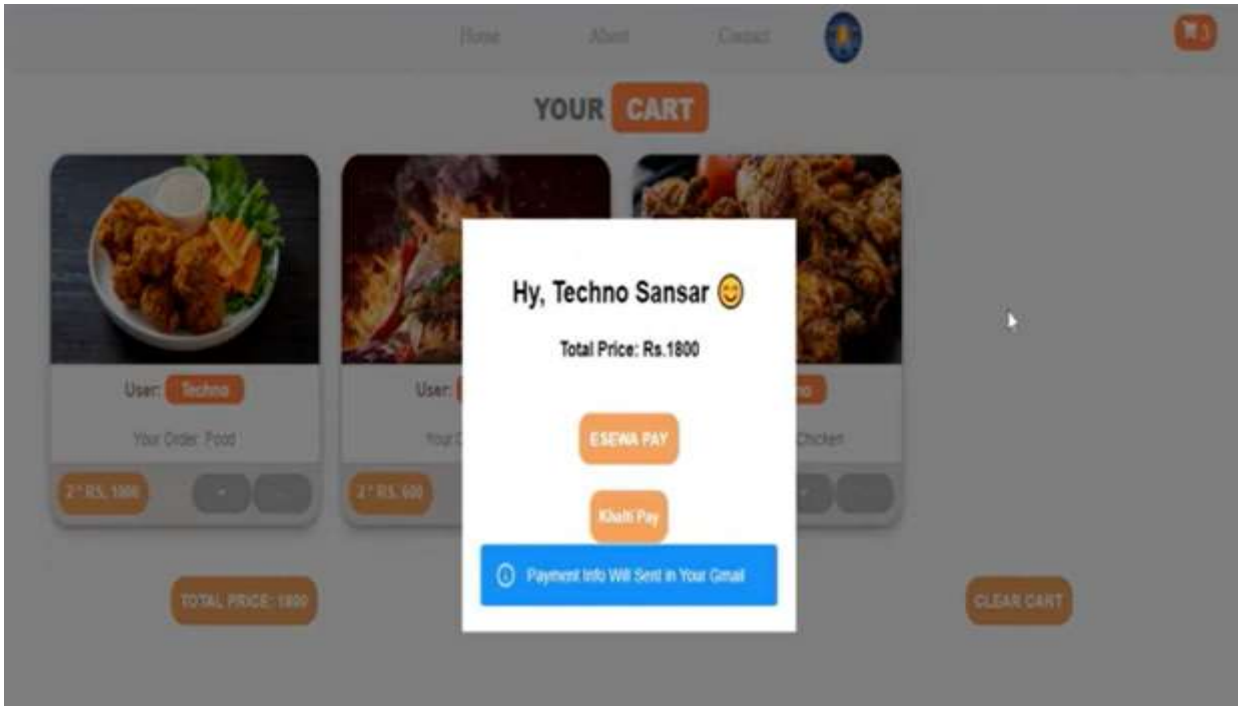
FOOD ITEM BY SEARCH:



VIEWING THE FOOD ITEM DETAILS IN CART:



AFTER PAYMENT:



Known Issues:

Users may occasionally encounter errors during the deletion of food items, which will be monitored for resolution in future updates.

Future Enhancements:

- Potential future features could include:
- Implementing user roles and permissions for advanced access control
- Integration of social media login options
- Adding advanced search and filtering capabilities
- Recommendations based on user preferences.