Food Delivery Web Application

Front-end:

To design the front-end for a food delivery platform, I'll outline the key features and provide a high-level overview of the implementation using React, advanced CSS, and JavaScript.

### Features:

1. Restaurant Listings: Display a list of available restaurants with their names, addresses, and cuisines.
2. Menus: Show the menu items for each restaurant, including prices and descriptions.
3. Order Placement: Allow users to select menu items and place orders with options for pickup or delivery.
4. Tracking: Provide real-time order tracking, including the status of the order (e.g., "Preparing", "On the way", "Delivered").
5. Reviews: Display user reviews and ratings for each restaurant.
6. User Notifications: Send notifications to users about their order status, promotions, and other relevant information.

### Implementation:

* React: Use React to build reusable UI components, manage state, and handle user interactions.
* Advanced CSS: Utilize CSS Grid, Flexbox, and media queries to create a responsive and visually appealing design.
* JavaScript: Write JavaScript code to handle business logic, API calls, and

// RestaurantList.js

import React from 'eact';

import RestaurantCard from './RestaurantCard';

const RestaurantList = ({ restaurants }) => {

return (

<div className="restaurant-list">

{restaurants.map((restaurant) => (

<RestaurantCard key={restaurant.id} restaurant={restaurant} />

))}

</div>

);

};

export default RestaurantList;

CSS

/\* styles.css \*/

.restaurant-list {

display: grid;

grid-template-columns: repeat(3, 1fr);

gap: 20px;

}

.restaurant-card {

background-color: #fff;

padding: 20px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

Back-end:

To build a full-stack food delivery platform, I'll outline the key features and provide a high-level overview of the implementation using Node.js, Express.js, and a database.

Features:

User Authentication: Implement user registration, login, and authentication using JSON Web Tokens (JWT).

Final Task : Food Delivery Web Application Front-end: Design the front-end for a food delivery platform with features like restaurant listings, menus, order placement, tracking, reviews, and user notifications using advanced CSS, JavaScript, and a front-end framework like React. Back-end: Build a full-stack food delivery platform using React for the front end and Node.js/Express.js for the back end. Implement user authentication, restaurant management, order processing, geolocation services, and database integration for user data, restaurants, orders, and reviews.

Food Delivery Web Application

Front-end:

To design the front-end for a food delivery platform, I'll outline the key features and provide a high-level overview of the implementation using React, advanced CSS, and JavaScript.

Features:

Restaurant Listings: Display a list of available restaurants with their names, addresses, and cuisines.

Menus: Show the menu items for each restaurant, including prices and descriptions.

Order Placement: Allow users to select menu items and place orders with options for pickup or delivery.

Tracking: Provide real-time order tracking, including the status of the order (e.g., "Preparing", "On the way", "Delivered").

Reviews: Display user reviews and ratings for each restaurant.

User Notifications: Send notifications to users about their order status, promotions, and other relevant information.

Implementation:

React: Use React to build reusable UI components, manage state, and handle user interactions.

Advanced CSS: Utilize CSS Grid, Flexbox, and media queries to create a responsive and visually appealing design.

JavaScript: Write JavaScript code to handle business logic, API calls, and state management.

Sample Code:

jsx

Edit

Copy code

// RestaurantList.js

import React from 'eact';

import RestaurantCard from './RestaurantCard';

const RestaurantList = ({ restaurants }) => {

return (

<div className="restaurant-list">

{restaurants.map((restaurant) => (

<RestaurantCard key={restaurant.id} restaurant={restaurant} />

))}

</div>

);

};

export default RestaurantList;

css

Edit

Copy code

/\* styles.css \*/

.restaurant-list {

display: grid;

grid-template-columns: repeat(3, 1fr);

gap: 20px;

}

.restaurant-card {

background-color: #fff;

padding: 20px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

Back-end:

To build a full-stack food delivery platform, I'll outline the key features and provide a high-level overview of the implementation using Node.js, Express.js, and a database.

Features:

User Authentication: Implement user registration, login, and authentication using JSON Web Tokens (JWT).

Restaurant Management: Allow restaurant owners to manage their menus, prices, and availability.

Order Processing: Handle order placement, payment processing, and order status updates.

Geolocation Services: Integrate with a geolocation service (e.g., Google Maps) to provide distance-based restaurant listings and order tracking.

Database Integration: Design a database schema to store user data, restaurants, orders, and reviews.

### Implementation:

* Node.js: Use Node.js as the server-side runtime environment.
* Express.js: Build a RESTful API using Express.js to handle HTTP requests and responses.
* Database: Choose a suitable database management system (e.g., MongoDB, PostgreSQL) to store data.

Sample Code:

// server.js

const express = require('express');

const app = express();

const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost/food-delivery-db', { useNewUrlParser: true, useUnifiedTopology: true });

app.use(express.json());

// Define API routes

app.post('/register', registerUser);

app.post('/login', loginUser);

app.get('/restaurants', getRestaurants);

app.post('/orders', placeOrder);

app.listen(3000, () => {

console.log('Server listening on port 3000');

});

JS

// models/Restaurant.js

const mongoose = require('mongoose');

const restaurantSchema = new mongoose.Schema({

name: String,

address: String,

cuisine: String,

menu: [{ type: mongoose.Schema.Types.ObjectId, ref: 'MenuItem' }]

});

module.exports = mongoose.model('Restaurant', restaurantSchema);

This is a high-level overview of the food delivery web application. The implementation details will vary depending on the specific requirements and technologies chosen