Food Delivery App

Contents

[1.Project Overview 2](#_Toc176709352)

[2.System Requirements 2](#_Toc176709353)

[3.Architecture 3](#_Toc176709354)

[4.User Roles and Permissions 4](#_Toc176709355)

[5.User Interface Design 4](#_Toc176709356)

[6. API Endpoints 5](#_Toc176709357)

[7.Milestones and Timeline 5](#_Toc176709358)

[8.Testing and Validation 6](#_Toc176709359)

[9.Deployment and Maintenance 6](#_Toc176709360)

# 1.Project Overview

Project Name: Food Delivery App (Web & Mobile)

Objective:

To develop a food delivery platform available via web and mobile, where users can browse restaurants, add items to their cart, place orders, and track the order status. The app will have a restaurant management dashboard and real-time order tracking.

# 2.System Requirements

**Functional Requirements:**

* **User Authentication** (common across web and mobile):
  + Users can sign up, log in, and log out securely.
  + Passwords should be encrypted, and JWT (JSON Web Tokens) will be used for authentication.
* **Customer Features (Web & Mobile):**
  + **Browse Restaurants:** Users can search and view restaurants and their menu.
  + **View Menu:** Users can view detailed information about the restaurant's menu.
  + **Add to Cart:** Users can add food items to the cart and update the cart.
  + **Place Order:** Users can place an order by checking out with the items in the cart.
  + **Track Order:** Users can track their order status (Pending, Preparing, Delivered) in real time.
* **Restaurant Features (Web-Based Dashboard):**
  + **Restaurant Dashboard:** Restaurants can log in to view and manage incoming orders.
  + **Menu Management:** Restaurants can add, update, or delete items from their menu.
  + **Order Management:** Restaurants can view and update the status of customer orders (Pending, Preparing, Delivered).
* **Admin Features (Optional, Web):**
  + Admins can manage all users, restaurants, and orders via an admin dashboard.

**Non-Functional Requirements:**

* **Security:**
  + HTTPS should be used to ensure secure data transmission.
  + Sensitive data (passwords, etc.) should be encrypted and stored securely.
  + Use JWT for secure authentication.
* **Performance:**
  + The system should handle at least 100 concurrent user sessions without performance degradation.
* **Scalability:**
  + Both web and mobile platforms should be scalable, allowing future expansions with additional features or an increased number of users.
* **Usability:**
  + The app should have a user-friendly UI/UX, providing a seamless experience for users on both web and mobile platforms.
* **Availability:**
  + The app should be available 24/7 with minimal downtime.

# 3.Architecture

**Web-Based Version:**

* **Frontend Tech Stack:**
  + **React** or **Vue.js** for building dynamic, responsive user interfaces.
  + **CSS Framework:** Bootstrap or Tailwind CSS for responsive and mobile-first design.
* **Backend Tech Stack:**
  + **Node.js with Express** or **Python with Flask/Django** for the backend, handling API requests and database queries.
* **Database:**
  + **MongoDB** (NoSQL) or **PostgreSQL/MySQL** (SQL) to store:
    - Users (Customers, Restaurant Owners, Admins)
    - Restaurants
    - Menu Items
    - Orders
    - Order Statuses

**Mobile App Version:**

* **Mobile Tech Stack:**
  + **React Native** (recommended for leveraging React skills) or **Flutter** to build cross-platform mobile apps (iOS and Android).
* **Mobile-Specific Features:**
  + Push notifications for real-time updates on orders.
  + Offline capabilities using local storage for browsing menus.

**Shared Backend API:**

* The backend APIs will serve both the web and mobile platforms.
* RESTful APIs will provide access to functionalities such as restaurant listings, placing orders, and updating order status.

# 4.User Roles and Permissions

 **Customer (Web & Mobile):**

* Can register, log in, browse restaurants, view menus, add items to the cart, place orders, and track orders.

 **Restaurant Owner (Web-Based Dashboard):**

* Can log in to the dashboard, manage their menu, view incoming orders, and update order statuses.

 **Admin (Web-Based, Optional):**

* Can manage users, restaurants, and orders.

# 5.User Interface Design

**Web App:**

* **Landing Page**: Displays a list of restaurants.
* **Restaurant Menu Page**: Displays a restaurant’s available menu items.
* **Cart Page**: Allows users to review and manage their selected items.
* **Order Status Page**: Displays the status of a customer’s order.
* **Restaurant Dashboard**: Allows restaurant owners to manage menus and orders.

**Mobile App:**

* The UI should be simplified for mobile use:
  + **Restaurant List**: Similar to web, but optimized for mobile screens.
  + **Menu Pages**: Mobile-friendly versions of restaurant menus.
  + **Cart & Order Summary**: Streamlined for quick interactions.
  + **Order Status**: Push notifications for real-time updates on orders.

# 6. API Endpoints

* **Authentication API:**
  1. POST /api/register: Register a new user (customer or restaurant owner).
  2. POST /api/login: Log in a user.
* **Restaurant API:**
  1. GET /api/restaurants: Retrieve the list of restaurants.
  2. GET /api/restaurants/:id/menu: Retrieve the menu for a specific restaurant.
* **Order API:**
  1. POST /api/orders: Place a new order.
  2. GET /api/orders/:orderId: Retrieve order details and status.
  3. PATCH /api/orders/:orderId: Update order status (for restaurant owners).
* **Restaurant Management API (Dashboard):**
  1. GET /api/dashboard/orders: Retrieve all incoming orders for a restaurant.
  2. POST /api/dashboard/menu: Add a new menu item.
  3. PATCH /api/dashboard/menu/:menuItemId: Update or delete menu items.

# 7.Milestones and Timeline

* Wireframes **and UI Design**: 1-2 weeks.
* **Web App Frontend Development**: 3-4 weeks.
* **Mobile App Development (React Native/Flutter)**: 3-4 weeks.
* **Backend API Development**: 4-5 weeks.
* **Database Setup**: 1-2 weeks.
* **Testing (Unit, Integration, UAT)**: 2-3 weeks.
* **Deployment (Web & Mobile)**: 1-2 weeks.

# 8.Testing and Validation

* **Unit Testing:** For individual components, APIs, and backend logic.
* **Integration Testing:** Ensuring frontend and backend work together smoothly.
* **User Acceptance Testing (UAT):** Gather feedback from potential users to refine UX.
* **Performance Testing:** Ensuring the system can handle concurrent users, especially during high traffic times.

# 9.Deployment and Maintenance

**Web App Deployment:**

* Host the web app on **Netlify**, **Vercel**, or **AWS**.

**Mobile App Deployment:**

* Use **React Native** or **Flutter** to create iOS and Android apps, deploy them on the **App Store** and **Google Play Store**.

**Backend Deployment:**

* Use **Heroku**, **AWS**, or **DigitalOcean** to deploy the backend APIs.

**Monitoring and Maintenance:**

* Set up monitoring tools like **New Relic** or **Google Cloud Monitoring** to track app performance, user activities, and bugs.
* Plan for regular updates and feature additions based on user feedback.