**1. Introduction and Project Overview**

**Project Name:** Food Delivery App (Web & Mobile)  
**Objective:**  
To build a food delivery platform that operates via both web and mobile applications. Users can browse restaurants, add items to their cart, place orders, and track order statuses. The platform will feature a restaurant management dashboard and real-time order tracking.

**2. Roles and Responsibilities**

| **Role** | **Responsibility** |
| --- | --- |
| **Project Manager** | Oversees the project, manages timelines, and ensures communication between stakeholders. |
| **Backend Developer** | Develops the server-side logic, API routes, and manages the database. |
| **Frontend Developer** | Builds the user interface for both web and mobile platforms. |
| **UI/UX Designer** | Designs wireframes and ensures a user-friendly interface. |
| **Database Administrator** | Designs and maintains the database schemas. |
| **QA Tester** | Conducts testing to ensure functionality and performance. |
| **Restaurant Owners** | Provide feedback on the dashboard and order management features. |
| **Users (Testers)** | Test the app from the perspective of a customer for feedback and user experience improvements. |

**3. Development Methodology**

**Agile Development Process**  
The project will follow an **Agile** methodology, breaking down development into iterative sprints (usually 2 weeks). Each sprint will deliver specific functionalities and refinements, with regular updates and communication among stakeholders.

**4. Feature Breakdown and Requirements**

**Core Features (MVP)**

1. **User Authentication**
   * Secure user registration and login using JWT.
2. **Restaurant Browsing**
   * Users can view a list of restaurants and filter by location or cuisine.
3. **Menu Viewing**
   * Each restaurant displays its menu with item details.
4. **Cart Management**
   * Users can add items to the cart and update quantities.
5. **Order Placement**
   * Users can review and confirm their order at checkout.
6. **Order Tracking**
   * Users can track the status of their order (Pending, Preparing, Delivered).
7. **Restaurant Dashboard**
   * Restaurant owners can manage their menu and view incoming orders.

**Optional Features (Post-MVP)**

1. **Push Notifications (Mobile)**
   * Customers receive order status updates via push notifications.
2. **Real-Time Order Tracking**
   * Use WebSockets or polling to display real-time order status updates.
3. **Payment Integration**
   * Integrate with payment gateways like Stripe or PayPal for online payments.

**5. Project Milestones and Timeline**

**Phase 1: Planning & Design**

**Duration:** 2 Weeks

* Define the project scope and features.
* Create wireframes and mockups for both the web and mobile versions.
* Finalize database and API design.

**Phase 2: Backend API Development**

**Duration:** 4 Weeks

* Develop authentication, restaurant listings, menu management, and order-related APIs.
* Set up and configure the database.

**Phase 3: Frontend Web Development**

**Duration:** 4 Weeks

* Develop the user interface for web users (React/Vue.js).
* Implement functionality for restaurant browsing, cart management, and order placement.

**Phase 4: Mobile App Development**

**Duration:** 4 Weeks

* Develop the mobile app using React Native or Flutter.
* Ensure seamless interaction with the backend API.

**Phase 5: Testing**

**Duration:** 2 Weeks

* Conduct unit, integration, and user acceptance testing (UAT).

**Phase 6: Deployment**

**Duration:** 1 Week

* Deploy the web app (Netlify/Vercel).
* Deploy the mobile app to the Google Play Store and Apple App Store.

**Phase 7: Post-Launch and Maintenance**

* Regular updates, bug fixes, and optional feature implementation.

**6. Wireframes and Design Prototypes**

**Tool Used:** Figma

* Attach all wireframes for each key page: restaurant listings, menu views, cart, order tracking, and restaurant dashboard.
* Ensure designs for both mobile and web interfaces are user-friendly and consistent.

**7. Database and API Design**

**Database Schema (MongoDB or PostgreSQL)**

* **Users Table**: Stores user data (customer, restaurant owner, admin).
* **Restaurants Table**: Stores restaurant details.
* **MenuItems Table**: Stores menu items and their respective prices.
* **Orders Table**: Stores order data, including status and related restaurant and customer IDs.

**API Design**

* **Authentication API**: /api/auth/register, /api/auth/login
* **Restaurant API**: /api/restaurants, /api/restaurants/:id/menu
* **Order API**: /api/orders, /api/orders/:orderId

**8. Testing Strategy**

**Unit Testing**

Test individual components and APIs to ensure functionality works as expected.

**Integration Testing**

Ensure that the web and mobile frontends communicate properly with the backend API.

**User Acceptance Testing (UAT)**

Conduct testing with a group of users (customers and restaurant owners) to validate the user experience.

**Performance Testing**

Evaluate the system’s performance under load (e.g., concurrent users).

**9. Deployment and Maintenance Plan**

**Web Deployment**

* **Platform:** Netlify, Vercel, or AWS
* **Deployment Process:** Set up CI/CD pipelines to automatically deploy the latest changes after testing.

**Mobile App Deployment**

* **Platform:** Google Play Store (Android), Apple App Store (iOS)
* **Process:** Build the mobile app with React Native/Flutter and submit it to app stores.

**Backend Deployment**

* **Platform:** Heroku, AWS, or DigitalOcean.
* **Process:** Deploy the backend APIs and ensure the database is configured.

**Post-Launch Maintenance**

* Bug fixes, system updates, and regular monitoring using tools like Google Cloud Monitoring or New Relic.
* Gather user feedback and implement requested features or improvements.

**10. Risk Management and Mitigation**

| **Risk** | **Mitigation Plan** |
| --- | --- |
| **Timeline Delays** | Break down tasks into smaller, manageable parts, and prioritize features. |
| **Technical Complexity** | Allocate additional time for researching complex features like real-time updates or payment integration. |
| **App Store Rejection** | Follow the respective store guidelines strictly and test the mobile app thoroughly before submission. |
| **Scalability Issues** | Use scalable services (like AWS, Heroku) to handle growing traffic and implement caching strategies (Redis) if needed. |

**11. Communication Plan**

* **Weekly Stand-ups**: All team members meet virtually to review progress, discuss blockers, and plan upcoming tasks.
* **Task Management**: Use **Jira** or **Trello** for task assignment, tracking, and sprint planning.
* **Stakeholder Updates**: Weekly email updates or meetings with key stakeholders to report progress and gather feedback.
* **Tools**: Use Slack for daily communication, GitHub for version control, and Google Drive for shared documents.