**Project Planning & Management**

**1. Project Proposal**

* **Overview:**  
  Crave is a dynamic food delivery web platform designed to connect customers with local restaurants seamlessly. The platform aims to provide an intuitive ordering experience and secure payment processing—all while ensuring ease of management for restaurant partners through a dedicated dashboard.
* **Objectives:**
  + Deliver a responsive, user-friendly interface for ordering food.
  + Enable restaurant partners to manage menus, orders, and promotional offers through an administrative portal.
  + Integrate secure, reliable payment gateways for smooth transactions.
  + Implement real-time order tracking and notification systems to enhance user satisfaction.
* **Scope:**  
  The project encompasses the development of both customer-facing and restaurant management modules. It includes designing the UI/UX, building a robust backend for handling orders and user data, integrating third-party services (e.g., payment and mapping APIs), and ensuring high availability and security throughout the platform.

**2. Project Plan**

* **Timeline & Milestones:**
  + **Weeks 1-2:**
    - Requirements gathering
    - Initial project proposal and design brainstorming
  + **Weeks 3-4:**
    - UI/UX design creation (wireframes, mockups)
    - Design approval and refinement
  + **Weeks 5-8:**
    - Development of frontend interfaces using modern web technologies
    - Backend development with API integrations (payment gateways, mapping services)
  + **Week 9:**
    - Integration testing and bug fixing
  + **Week 10:**
    - Beta launch for early feedback collection
  + **Week 11:**
    - Final adjustments and performance tuning
  + **Week 12:**
    - Official deployment and launch
* **Key Deliverables:**
  + Comprehensive project proposal and design documents
  + Wireframes, mockups, and interactive prototypes
  + Fully developed source code with inline documentation
  + Detailed testing reports and user feedback analysis
  + Deployed web application with continuous integration and deployment pipelines

**Task Assignment & Roles**

* **Student 1: Project Manager & QA Engineer**
  + **Project Manager:** Oversees overall project progress, timeline management, and stakeholder communications, ensuring that objectives are met on schedule.
  + **QA Engineer:** Conducts thorough testing (unit, integration, and usability) to verify functionality, performance, and security across the application.
* **Student 2: UI/UX Designer & Frontend Developer**
  + **UI/UX Designer:** Crafts engaging, intuitive designs, wireframes, and mockups to define the visual and interactive aspects of Crave.
  + **Frontend Developer:** Implements the customer-facing interface using responsive web technologies (e.g., React, Angular, or Vue) to ensure a smooth user experience.
* **Student 3: Backend Developer**
  + Develops and maintains the server-side application, manages database interactions, and integrates essential third-party APIs for payment processing and geolocation services.
* **Student 4: DevOps Engineer**
  + Manages deployment processes, sets up continuous integration/continuous deployment (CI/CD) pipelines, and ensures the scalability and reliability of the production environment.

**4. Risk Assessment & Mitigation Plan**

* **Identified Risks:**
  + Technical Integration Issues: Challenges with integrating payment gateways or mapping APIs.
  + Scalability Concerns: Handling increased traffic and order volume during peak times.
  + Security Vulnerabilities: Risks related to data breaches or transaction security.
* **Mitigation Strategies:**
  + Develop contingency plans with alternative API options and maintain close communication with third-party providers.
  + Design a scalable architecture with load balancing, auto-scaling, and performance monitoring.
  + Implement rigorous security protocols, regular vulnerability assessments, and adhere to industry best practices for data protection.

**5. Key Performance Indicators (KPIs)**

* **System Performance:**
  + **Response Time:** Target a maximum load time of under 2 seconds per page.
  + **Uptime:** Achieve a minimum of 99.9% system availability.
* **Order Efficiency:**
  + **Order Processing Time:** Ensure orders are processed within 5 minutes from placement to confirmation.
* **User Engagement & Satisfaction:**
  + **Active User Growth:** Monitor monthly active users and order frequency.
  + **Customer Satisfaction:** Maintain a satisfaction rating of at least 90% based on user surveys and feedback.
* **Revenue & Partner Metrics:**
  + **Average Order Value:** Track and aim to increase the average transaction value over time.
  + **Restaurant Partner Engagement:** Monitor the number of active restaurant partners and their order volumes to ensure platform viability.

**Requirements Gathering**

**1.Stakeholder Analysis**

* **Customers:**  
  Identify individuals who will order food using Crave. Their needs include an intuitive interface, fast order placement, secure payment processing, and reliable customer support.
* **Restaurant Partners:**

Understand the needs of local restaurants that will use the platform to manage their menus, orders, and promotions. Their requirements include a simple dashboard for updating offerings, order management, and analytics.

* **Delivery Personnel:**  
  Consider the needs of delivery drivers who require efficient route planning, timely notifications, and clear instructions for pickups and drop-offs.
* **Administrators & System Managers:**  
  Define the needs of those overseeing the platform’s operation—ensuring system reliability, security, performance monitoring, and support for troubleshooting issues.

**2.User Stories & Use Cases**

* **Customer User Story:**  
  "As a customer, I want to easily browse available food options and track my order State so that I can have a smooth ordering experience."
* **Restaurant Partner Use Case:**  
  "A restaurant logs to update their menu items manage incoming orders efficiently."
* **Delivery Personnel User Story:**  
  "As a delivery person, I need to receive timely notifications and update order state”
* **Administrator Use Case:**  
  "An administrator monitors system performance and resolves issues, ensuring that all components—from order processing to payment handling—function reliably."

**Functional Requirements**

* **User Interface:**
  + Responsive design for both desktop and mobile devices.
  + Intuitive navigation with clear categories for food items.
* **Authentication & Authorization:**
  + Secure registration and login for customers, restaurants, and staff.
  + Role-based access control for different user types.
* **Order Management:**
  + Ability for customers to add items to a cart, place orders, and view order history.
  + Order status updates.
* **Payment Processing:**
  + Integration with secure payment gateways to handle transactions.
  + Support for multiple payment methods (credit/debit cards, digital wallets).
* **Restaurant Dashboard:(x)**
  + Tools for managing menu items, prices, and order statuses.
  + Analytics on sales and customer feedback.
* **Notification System:**
  + Email and push notifications for order confirmations, status changes, and promotions.
* **Administrative Tools:**
  + User management, and reporting capabilities.

**Non-functional Requirements**

* **Performance:**
  + Fast page load times
  + Ability to handle high traffic and concurrent users, especially during peak hours.
* **Security:**
  + Implementation of HTTPS, data encryption, and secure payment processing.
  + Robust authentication mechanisms to protect user data.
* **Usability:**
  + An intuitive user experience with accessible design standards.
  + Clear, consistent layouts and easy-to-use interfaces for all user roles.
* **Reliability & Scalability:**
  + System uptime of at least 99.9%.
  + Scalable architecture to accommodate future growth in users and transactions.
* **Maintainability:**
  + Modular codebase with thorough documentation to facilitate future updates and maintenance.
  + Regular testing and monitoring for ongoing quality assurance.

**System Analysis & Design**

**1. Problem Statement:**  
 In today’s fast-paced environment, ordering food can often be cumbersome—customers face delayed order confirmations, poor tracking, and inconsistent service. Local restaurants, on the other hand, struggle to efficiently manage online orders and menu updates. Crave aims to bridge this gap by providing a unified, intuitive platform that streamlines the food ordering process while offering restaurants a reliable dashboard for order and menu management.

**Project Objectives:**

* **Enhance User Experience:** Deliver a responsive, user-friendly interface that simplifies the food ordering process from browsing to delivery tracking.
* **Improve Operational Efficiency:** Enable restaurants to update menus, manage orders, and track sales in real time through a dedicated dashboard.
* **Ensure Reliability and Security:** Implement robust authentication, secure payment processing, and real-time order tracking, ensuring a safe and efficient transaction experience.
* **Scalability:** Build an architecture that can handle peak loads and can be easily scaled as the user base grows.

**2.** **Use Case Description**

**Primary Actors:**

* **Customer:** Uses the platform to browse restaurants, order food, and track delivery.
* **Restaurant Partner:** Manages menu items, processes orders, and monitors sales.
* **Delivery Personnel:** Receives order assignments and updates delivery status.
* **Administrator:** Oversees system operations, manages users, and handles troubleshooting.

**Key Use Cases & Descriptions:**

* **Customer Use Cases:**
  + *Browse & Search Menu:* Customers view available food items, filter by category or cuisine, and search for specific dishes.
  + *Place Order:* Select food items, add to cart, and complete payment.
  + *Track Order:* Receive real-time updates on order status from preparation to delivery.
  + *Manage Profile:* Update personal details, view order history, and manage payment methods.
* **Restaurant Partner Use Cases:**
  + *Manage Menu:* Add, update, or remove menu items, set prices, and mark items as available/unavailable.
  + *Process Orders:* Receive notifications for new orders, update order status, and communicate with delivery personnel.
* **Delivery Personnel Use Cases:**
  + *Receive Delivery Assignment:* Get notified of new delivery tasks along with order details.
  + *Update Delivery Status:* Mark orders as picked up, in-transit, or delivered.
* **Administrator Use Cases:**
  + *User Management:* Manage user accounts for customers, restaurants, and delivery personnel.
  + *Issue Resolution:* Address reported issues, perform routine maintenance, and ensure data integrity.

**3. High-Level Architecture:**  
Crave follows a layered architecture model, promoting separation of concerns and scalability. The system is designed using the Model-View-Controller (MVC) architecture pattern, with clear divisions between the user interface, business logic, and data access layers.

* **Presentation Layer (View):**
  + Implements the frontend using modern web frameworks to deliver responsive and engaging user interfaces.
  + Handles all user interactions and displays dynamic content based on user inputs and system responses.
* **Business Logic Layer (Controller):**
  + Manages application workflows, processes orders, handles authentication, and enforces business rules.
  + Coordinates with the presentation layer to validate and process user inputs, and communicates with the data access layer for database operations.
* **Data Access Layer (Model):**
  + Utilizes an ORM (Object-Relational Mapping) such as Entity Framework Core to abstract database interactions.
  + Manages CRUD operations for entities like Users, Orders, Menu Items, and Delivery Data through repository and unit-of-work patterns.
* **Integration Layer:**
  + Interfaces with third-party APIs (e.g., payment gateways, mapping services) to handle secure transactions and real-time order tracking.
* **Infrastructure & Deployment:**
  + Emphasizes scalability through cloud deployment (e.g., Microsoft Azure, AWS) with CI/CD pipelines for continuous integration and deployment.
  + Implements robust monitoring, logging, and security measures to ensure high availability and performance.