**Project Report Format**

1. **INTRODUCTION** 
   1. Project Overview
   2. Purpose
2. **IDEATION PHASE**
   1. Problem Statement
   2. Empathy Map Canvas
   3. Brainstorming
3. **REQUIREMENT ANALYSIS**
   1. Customer Journey map
   2. Solution Requirement
   3. Data Flow Diagram
   4. Technology Stack
4. **PROJECT DESIGN** 
   1. Problem Solution Fit
   2. Proposed Solution
   3. Solution Architecture
5. **PROJECT PLANNING & SCHEDULING** 
   1. Project Planning
6. **FUNCTIONAL AND PERFORMANCE TESTING** 
   1. Performance Testing
7. **RESULTS** 
   1. Output Screenshots
8. **ADVANTAGES & DISADVANTAGES**
9. **CONCLUSION**
10. **FUTURE SCOPE 11. APPENDIX**

Source Code (if any)

Dataset Link

GitHub & Project Demo Link

**1. INTRODUCTION**

**1.1 Project Overview**

This project involved the development of a comprehensive web-based food delivery system. The primary goal was to create a platform that efficiently connects customers with local restaurants, facilitating easy food ordering, real-time tracking, and effective management for all stakeholders. The system is designed to enhance user convenience while empowering restaurant owners with tools to streamline their operations.

**1.2 Purpose**

The purpose of this report is to document the various phases of the Food Delivery System project, from ideation and requirements analysis through design, planning, and testing. It provides a detailed account of the problem addressed, the proposed solution, the technologies utilized, the project's execution, and its outcomes.

**2. IDEATION PHASE**

**2.1 Problem Statement**

The project addresses the challenge faced by **busy individuals/families** who struggle to find a convenient, diverse, and reliable way to order food from local restaurants. Simultaneously, it tackles the difficulties experienced by **restaurant owners** in expanding their customer reach and efficiently managing online orders without a robust, integrated system.

* **PS-1 (Customer):** "I am a busy individual/family member trying to order food conveniently from local restaurants, but existing options are limited, or ordering processes are cumbersome and slow, because there isn't a centralized, user-friendly platform with diverse restaurant choices, which makes me feel frustrated and hungry."
* **PS-2 (Restaurant Owner):** "I am a restaurant owner trying to expand my customer base and streamline order management, but managing online orders manually is inefficient, and reaching new customers is difficult, because there's no integrated system to showcase my menu and automate order processing, which makes me feel overwhelmed and losing potential business."

**2.2 Empathy Map Canvas**

The Empathy Map for the customer revealed key insights:

* **Sees:** People using food delivery apps, diverse food promotions, app interfaces with varied menu presentations.
* **Hears:** Friends discussing good delivery apps, social media ads, occasional frustrations about payment methods.
* **Thinks & Feels:** "What should I eat today?", "Will the food be on time/cold?", desire for food matching interests, frustration with manual payments.
* **Says & Does:** Scrolls through menus, places orders, tracks status, expresses a desire for digital payments, seeks convenience.
* **Pains:** Limited selection, long wait times, cold/incorrect food, confusing interfaces, cash payment hassles.
* **Gains:** Wide variety, efficient ordering, real-time tracking, card payment options, direct delivery.

**2.3 Brainstorming**

Brainstorming focused on addressing the problem statement, leading to ideas categorized into customer-centric, restaurant-centric, and admin/platform-centric features. Key ideas included:

* **Customer:** User-friendly interface, real-time tracking, multiple payment options, personalized recommendations, easy re-ordering, rating/review system.
* **Restaurant:** Owner dashboard for order/menu management, new order notifications, performance analytics.
* **Admin:** System oversight dashboard, user/restaurant approval, global category management, comprehensive reporting, announcements.

Prioritization focused on core functionalities first (e.g., user authentication, order placement, basic management) with more advanced features considered for future iterations.

**3. REQUIREMENT ANALYSIS**

**3.1 Customer Journey Map**

The customer journey map outlines the typical path a customer takes:

* **Entice:** Awareness through ads or referrals.
* **Enter:** Registration and login to access the app.
* **Engage:** Browsing menus, adding to cart, placing an order.
* **Exit:** Order preparation, delivery, and receipt of food.
* **Extend:** Providing feedback, re-ordering, or receiving post-delivery promotions.

Key pain points identified in the journey include confusing registration, app crashes during checkout, delayed or incorrect orders, and complex feedback processes, which the proposed solution aims to mitigate.

**3.2 Solution Requirement**

The solution adheres to a set of detailed functional and non-functional requirements:

* **Functional:**
  + **User Management:** Registration (Form-based), Login, Profile Management (view, update, change password), Role-based access (Customer, Restaurant, Admin).
  + **Restaurant & Menu Management:** Restaurant owner registration/approval, add/edit restaurant details, add/edit/delete food items.
  + **Order Management:** Cart functionality, order creation, order history, order status updates.
  + **Search & Filtering:** Search restaurants, filter food items by category.
  + **Admin Specifics:** User/restaurant approval, category management, dashboard, promoted restaurants, feedback management, announcements, reports.
  + **Feedback:** Customer-to-Restaurant/Food, Restaurant-to-Admin, User-to-Admin feedback.
* **Non-functional:**
  + **Usability:** Intuitive UI, responsive design.
  + **Security:** Hashed passwords, JWT authentication/authorization, protection against common vulnerabilities.
  + **Reliability:** Graceful error handling, data integrity.
  + **Performance:** Optimized API response times, efficient database queries.
  + **Availability:** Minimized downtime, support for load balancing.
  + **Scalability:** Ability to handle increasing user/order volume through architectural design.

**3.3 Data Flow Diagram**

The Data Flow Diagram (DFD) illustrates how information moves through the system.

* **DFD Level 0 (Context Diagram):** Shows external entities (Customer, Restaurant Owner, Admin, Payment Gateway) interacting with the central "Food Delivery System" process, exchanging data like registration info, order requests, and reports.
* **DFD Level 1 (Decomposition):** Breaks down the system into core processes: User & Authentication Management, Restaurant & Menu Management, Order Processing & Delivery, Communication & Feedback, and Admin & Reporting. It details data flows between these processes and internal data stores (Users, Restaurants, Food Items, Orders, Categories, Feedback).

**3.4 Technology Stack**

The project is built on a robust and modern technology stack:

* **Frontend:** HTML, CSS, JavaScript for the static web pages and client-side logic.
* **Backend:** Node.js runtime environment using the Express.js framework for RESTful API development.
* **Database:** MongoDB as the NoSQL database for flexible data storage, managed with Mongoose ODM.
* **Authentication:** JSON Web Tokens (JWT) for secure, stateless authentication and authorization.
* **Security:** bcrypt.js for password hashing.
* **Infrastructure:** Developed on a local server environment, with design considerations for cloud deployment (e.g., Docker for containerization).

**4. PROJECT DESIGN**

**4.1 Problem Solution Fit**

The solution directly addresses the identified problems:

* **For Customers:** It provides a centralized, easy-to-use platform with diverse restaurant options, real-time tracking, and secure payment methods, alleviating the frustration of limited choices and cumbersome ordering.
* **For Restaurant Owners:** It offers an integrated system for menu presentation, automated order processing, and customer reach, reducing manual overhead and increasing business potential.

**4.2 Proposed Solution**

The proposed solution is a full-stack web application supporting multiple user roles (Customer, Restaurant Owner, Admin). It features a user-friendly interface for customers to browse and order food, a dedicated portal for restaurant owners to manage their businesses, and a powerful admin dashboard for system-wide control and reporting. The solution emphasizes efficiency, user satisfaction, and operational streamlining for restaurants.

**4.3 Solution Architecture**

The system employs a 3-tier architecture:

* **Presentation Layer:** HTML, CSS, and client-side JavaScript for user interaction.
* **Application Layer:** Node.js with Express.js handles all business logic, API requests, and serves the HTML pages.
* **Data Layer:** MongoDB stores all application data, accessed via Mongoose.

Authentication (JWT) and authorization middleware’s secure API endpoints. The architecture is designed for scalability (horizontal scaling of backend, MongoDB sharding capabilities) and availability (load balancing for future cloud deployments).

**5. PROJECT PLANNING & SCHEDULING**

**5.1 Project Planning**

The project was executed using an Agile methodology, structured into two aggressive 5-day sprints.

* **Project Dates:** June 17, 2025 - June 25, 2025 (Total 9 days)
* **Total Story Points:** 54 (distributed across two sprints)
* **Sprint 1 (June 17 - June 20):** Focused on core user authentication, basic profile management, initial restaurant owner/admin setup, and basic browsing/ordering. (24 Story Points)
* **Sprint 2 (June 21 - June 25):** Focused on advanced order features, comprehensive restaurant/menu management, feedback systems, comprehensive admin reporting, and final refinements. (30 Story Points)
* **Team Velocity:** Approximately 6 Story Points per day (54 total points / 9 days).
* **Burndown Chart:** A visual representation tracking remaining work versus time was used to monitor progress, starting at 70 points and aiming for 0 by June 26th.

**6. FUNCTIONAL AND PERFORMANCE TESTING**

**6.1 Performance Testing**

Performance was considered throughout development with an emphasis on efficient API design and optimized database queries. While formal load testing tools were not used in this phase, the architecture supports future performance enhancements like caching (Redis) and Content Delivery Networks (CDNs). The goal was to ensure critical operations (e.g., login, order placement, dashboard loading) had quick response times.

**7. RESULTS**

**7.1 Output Screenshot**

* Customer Login Page
* Customer Dashboard/Restaurant Listing
* Restaurant Menu View with Add to Cart
* Customer Order History
* Restaurant Owner Dashboard with New Order Notification
* Restaurant Owner Menu Management
* Admin Dashboard with Metrics
* Admin User/Restaurant Approval List
* Admin Reports (e.g., Order Trend Chart)

are present in the **Project Documentation format**

**8. ADVANTAGES & DISADVANTAGES**

**Advantages:**

* **Comprehensive Functionality:** Caters to three distinct user roles (Customer, Restaurant, Admin) with tailored features.
* **User-Friendly Interface:** Designed for intuitive navigation and ease of use.
* **Efficient Order Management:** Streamlines the process for both customers (ordering, tracking) and restaurants (receiving, updating).
* **Robust Admin Control:** Powerful dashboard for system oversight, content management, and analytical reporting.
* **Scalable Architecture:** Built on a modular, multi-tier structure (Node.js, MongoDB) that supports future growth and high traffic.
* **Secure Authentication:** Utilizes JWT and password hashing for strong security.
* **Open-Source Technologies:** Leverages well-supported open-source frameworks, facilitating development and potential future contributions.

**Disadvantages:**

* **Limited Real-time Delivery Tracking:** Currently lacks advanced GPS-based real-time tracking integration with external map APIs.
* **Basic Payment Gateway:** Only supports COD and conceptual card payments without a live payment gateway integration.
* **No Push Notifications:** Lacks real-time push notifications for order updates (e.g., to customer's phone).
* **Initial Setup Complexity:** Setting up the local environment with Node.js and MongoDB might require some technical knowledge.
* **UI Polish:** While functional, further UI/UX refinements could enhance the aesthetic appeal and user experience.

**9. CONCLUSION**

The Food Delivery System project successfully developed a functional web application addressing core problems for customers and restaurant owners. It provides a solid foundation for online food ordering, menu management, and administrative control. The system demonstrates the effective integration of modern web technologies, secure authentication, and a scalable architectural design, proving its viability as a valuable solution in the food delivery domain.

**10. FUTURE SCOPE**

* **Advanced Online Payment Integration:** Integrate with popular payment gateways (e.g., Stripe, PayPal) for seamless online transactions.
* **Live Delivery Tracking:** Implement real-time GPS tracking for delivery drivers, visible on the customer's order tracking page using mapping APIs.
* **Push Notifications:** Integrate WebSocket technology for real-time push notifications to users for order status changes and announcements.
* **Rating and Review System for Food Items:** Allow customers to rate and review individual food items, not just restaurants.
* **Promotional Campaign Management:** Enable admins and restaurant owners to create and manage discounts, loyalty programs, and special offers within the platform.
* **Enhanced Search and Filtering:** Implement more advanced search capabilities, including filters for dietary preferences, cuisine types, and delivery time estimates.
* **Mobile Application Development:** Extend the service to native mobile applications (iOS/Android) for a richer mobile experience.
* **Cloud Deployment & DevOps:** Implement CI/CD pipelines and deploy the application to a robust cloud infrastructure (e.g., AWS, GCP, Azure) using containerization (Docker) and orchestration (Kubernetes).

**11. APPENDIX**

**Source Code (if any)**

* <https://github.com/NaiduManasa/food-delivery-app>

**Dataset Link**

* <https://github.com/NaiduManasa/food-delivery-app/blob/main/models/schema.js>

**GitHub & Project Demo Link**

* **GitHub Repository:** [**https://github.com/NaiduManasa/food-delivery-app**](https://github.com/NaiduManasa/food-delivery-app)
* **Project Demo:** http://localhost:5000