Online Food Ordering System

Objective:

The main objective of this project is to create a simple and user-friendly web app where users can order food from their favorite restaurants online. This will simplify the restaurants order management- and save customers time as they can order and pay for food from the comfort of their homes.

Features and Functions:

1. User (Customer) Features:
    - Account creation and login system.
    - View food menu.
    - Add food to cart.
    - Online payment / cash on delivery option.
    - View order tracking and delivery status.
    - Give reviews and ratings.
2. Restaurant Admin Features:
    - Add, update, or delete new food.
    - View orders and change status.
    - View sales reports.
    - View customer feedback.
3. System Admin Features:
    - Restaurant and user management.
    - Security and data backup.
    - Report generation.

Stakeholders:

1. Customer
2. Restaurant owner
3. Delivery man
4. System Administrator

Project goal:

Creating a web-based system that will allow users to easily order and pay for food and for the management of restaurant digitalization.

Project planning:

We need 5 members for this project, which will last for a total of 3 months and is estimated to cost around $2500.

1. Task initiation:
    - Requirement Analysis
    - Feasibility Study
    - Project Charter
2. System Design:
    - UI/UX Design
    - Database design
    - System architecture
3. Development:
    - Frontend
    - Backend
    - Integration
4. Testing:
    - Unit testing
    - System testing
    - Bug testing
5. Deployment:
    - Server setup
    - Domain and hosting
    - Go live
6. Maintenance:
    - Performance monitoring
    - Regular updates
    - User support

Project Scheduling:

Phase	Task	Duration	Timeline
Week 1–2	Requirement Analysis & Feasibility Study	2 weeks	1 Jan – 14 Jan
Week 3–4	System & UI Design	2 weeks	15 Jan – 28 Jan
Week 5–8	Development (Frontend + Backend)	4 weeks	29 Jan – 25 Feb
Week 9–10	Integration & Testing	2 weeks	26 Feb – 10 Mar
Week 11	Deployment	1 week	11 Mar – 17 Mar
Week 12	Maintenance & Review	1 week	18 Mar – 24 Mar

Budget Estimation:

Category	Description	Estimated Cost (USD)
Human Resources	1 Project Manager ($500), 2 Developers ($400 × 2), 1 Designer ($300), 1 Tester ($200)	$1,800
Software Tools	VS Code, GitHub (Free), Hosting Control Panel, Database Tools	$100
Hardware / Server	Domain ($30), Hosting Server ($120/year)	$150
Miscellaneous	Internet, Testing devices, Backup drives	$100
Total Estimated Budget	—	$2,150

Team Formation:

To successfully complete every project, an organized team is needed, where everyone has specific roles and responsibilities.

- **Team Members Responsibilities:**

  First of all the project manager or the team leader planning the whole project, manage time and budget complexity, creates a risk assessment and report it.

  The system analyst (the requirement and design expert) analyzes the client's needs, create system design modeling and user flow documentation.

  The visible portion of the website or application that users view and interact with is created by the front-end developer.

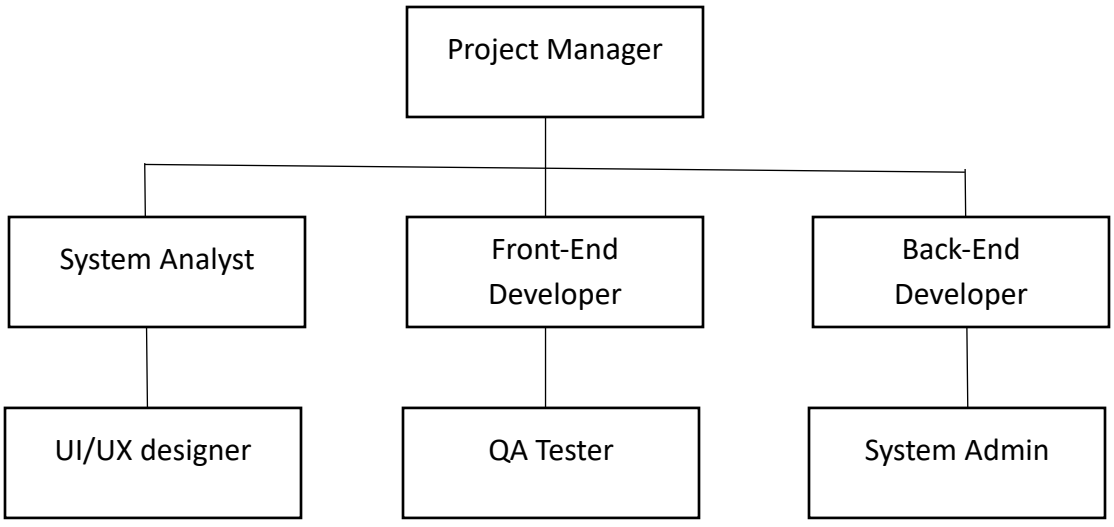
  The back-end developer then constructs the back-end, overseeing the database, server, and business logic.

  A UI/UX designer creates the software's appearance and feel to make it simple and pleasurable to use.

  Before the software is released, QA testers test it to identify and report any errors, bugs, or issues.

  Finally, through servers, security, and data backups, the system administrator maintains the functionality of the live software.

- **Organizational Chart:**



Risk Management:

There are risks associated with every software project, including team issues, data loss, and delays. Each risk's likelihood, impact, and mitigation strategy are broken down below.

- A prerequisite Change: We will use a formal "Change Request" process for any new client ideas and clearly define all needs at the outset.
- Delay in Development: We'll hold regular progress meetings, track our schedule closely, and add more developers if we fall behind.
- Technical Failure: We'll keep regular backups, use a strong server, and have a backup (failover) system ready to go.
- Security Breach: We'll test for security flaws, employ secure connections, create strong passwords, and defend against frequent attacks.
- Budget Overrun: We'll keep a close eye on all expenditures, eliminate any features that aren't absolutely necessary, and update cost estimates frequently.
- Unavailable Team Member: We'll make sure that everyone keeps a record of their activities and has a backup plan in place.
- Ineffective Communication: To keep everyone in agreement, we'll hold frequent meetings and make use of online resources.
- Testing Delays: To expedite the process, we'll begin testing earlier (during coding) and make use of automated tools.
- Delay in Client Feedback: We'll give them polite reminders and establish precise due dates for client feedback.
- Data Loss or Corruption: We'll use cloud storage, create a daily backup, and have a backup database available.

Mitigation Strategy Summary:

Risk Type	**Prevention Method**
Technical Risks	Regular backups, secured server setup, automated testing
Managerial Risks	Weekly meetings, updated progress reports, milestone tracking
Human Risks	Cross-training team members, keeping backup personnel
Financial Risks	Expense monitoring, early estimation updates
Security Risks	Authentication, encryption, regular audits

Development:

Development Methodology

Method Chosen: Agile.

Agile encourages quick, iterative development through brief work cycles called sprints and provides flexibility to easily handle client changes, which is why I chose it. This strategy is great for lowering risk and making sure you receive regular, early feedback to keep the project moving forward.

Steps in the Agile Development Process

The entire project is divided into sprints, which are brief, recurring stages. A functional portion of the system is delivered following each sprint. First, we gather the requirements.

- Then comes planning and design.
- Next are the main development sprints (Sprint 1, 2, 3, 4).
- Followed by testing and feedback.
- Then, the product goes live (deployment).
- Finally, we do review and maintenance.

Sprint Breakdown

- Sprint 1 (Weeks 1–2): System Setup & Basic UI

  The goal is to finalize requirements, design the database, and build the basic homepage and login screens. After that result is a working user login and homepage demonstration.

- Sprint 2 (Weeks 3–4): Core Functionality

  Implement the main features like displaying menus, "add to cart," and creating an order. And then we achieve a basic, functional order system.

- Sprint 3 (Weeks 5–6): Payment & Admin Module

  Integrate the payment system and build the admin control panel for the restaurant menu. A working admin panel.

- Sprint 4 (Weeks 7–8): Testing & Deployment

Complete all system testing, fix any remaining bugs, and host the application live. The final, live, and fully functional product.

Tools & Technologies We Will Use

- Frontend (What you see): HTML, CSS, JavaScript, and React.js.
- Backend (The engine): Node.js / Express.js (or PHP) for the server logic.
- Database (Data storage): MySQL / MongoDB.
- Code Management: Git and GitHub (to track changes in the code).
- Project Management: Trello / Jira (to track sprints and tasks).
- Testing: Postman or Selenium (to test APIs and functionality).
- Deployment (Hosting): AWS / cPanel Hosting.

Conclusion:

The entire plan for creating an online food ordering system was laid out in this document. To guarantee that the system is finished on schedule, within budget, and to a high standard of quality, the entire project will be managed step-by-step with precise plans and designated roles.

- **Summary of the Plan**
  - The project's objective is to develop a straightforward and user-friendly online ordering and payment system for restaurants.
  - Development Approach: The project is being divided into four brief work cycles, or sprints, using the Agile (Scrum) method. After each sprint, client input will be gathered to enable any necessary adjustments.
  - Planning & Management: Twelve weeks are allotted for the project's completion. The plan addresses risk management, team roles, the budget ($2,150), the timeline (Gantt chart), and the work breakdown structure (WBS).
  - Risk and Quality Control: Every possible risk, such as delays, technical malfunctions, or budget overruns, has been identified, and a clear solution (mitigation plan) is prepared for each. To maintain a high level of system quality, we will employ routine testing and review procedures.

- **What We Learned from This Plan**
    - We recognized how crucial organization, communication, and planning are to the success of a software project.
    - We discovered how the Agile Methodology enhances responsiveness to feedback, teamwork, and speed.
    - Project failure is greatly decreased by risk management, and user satisfaction is ensured by quality assurance.
    - A clear project roadmap is provided by having an accurate Work Breakdown Structure (WBS), Schedule, and Budget Estimate.

- **Final Remark**

  This plan demonstrates that a well-organized software project can easily succeed. If the team maintains regular communication, completes tasks on time, and uses client feedback, this Online Food Ordering System will become an effective and reliable web solution.