Project Management Plan (PMP)

## **Project Overview**

The Foodies Web App is a web-based application that connects users with nearby restaurants and allows them to place food orders online. It is designed to provide a smooth and engaging experience for discovering and ordering food, while also helping restaurants increase their visibility and promote their services digitally.

Users can create accounts, browse a list of nearby restaurants, apply search and filter options, view current promotions, and earn loyalty points through a credit-based system. Restaurant owners can manage their menus and promotional offers through an admin interface. The system includes administrative functionalities, utilizes unique user IDs, and is designed for PC/web use.

The project is being developed by a team of six members over a planned duration of at least six weeks. It follows the Waterfall model with phase-based collaboration, meaning all team members are involved in each phase and rotate responsibilities throughout the process.

## **Project Organization**

* Customer: Omar
* Supervisor: Mohamed Hassan
* Coach: Amr Mokhtar
* Project Manager: Saged Wael

## **Timeline – Milestones**

|  |  |  |  |
| --- | --- | --- | --- |
| Phase | Duration | Start Date | End Date |
| Requirement Analysis | 1 week | [04/05/2025] | [04/11/2025] |
| Design | 1 week | [04/12/2025] | [04/18/2025] |
| Development | 2 weeks | [04/19/2025] | [05/02/2025] |
| Testing | 1 week | [05/03/2025] | [05/09/2025] |
| Deployment | 1 week | [05/10/2025] | [05/16/2025] |

## **Project Scope:**

**In scope:**

* **Requirements**
* **System Design**
* **Development**
* **Validation Testing**

**Out scope:**

* Marketing strategy planning
* Unit Testing
* Integration testing

## **Task Planning:**

Who:

• The Project Manager oversees the task distribution.

How:

• Every delivery is divided into smaller, manageable tasks.

• Tasks are organized and assigned through the project management tool, Trello.

• Each task involves at least two key activities:

 1. Assignment to a developer

 2. Assignment to a reviewer

• Task notifications are sent to each assignee via email.

• Additionally, each delivery includes an activity dedicated to generating a baseline.

What:

• All team members are clearly informed of their individual responsibilities.

## **Communication Plan**

* **Meetings:**
* **Weekly meeting**: (Delivery Planning)
* **Day**: Tuesday
* **Status**: Face to Face
* **leader**: PM

**Standup meeting:**

* **Days**: Thursday
* **Status**: Online
* **leader**: PM

To Discuss:

1. What is done
2. What will we do today

## **Review Strategy:**

* **Task Assignment**

Each main task (e.g., writing a document, creating test cases) is assigned to a responsible team member (e.g., Member1).

A corresponding review task is created and assigned to a reviewer (e.g., Member2). The review task is clearly linked to the original task.

* **Review Execution**

The reviewer (Member2) uses a Review Template to document feedback.

If the task includes multiple parts (e.g., several requirements, test cases, or components), each part is reviewed individually.

For each item, the reviewer provides:

Comments or suggestions

Issue details (if any)

A specific status: Approved, Pending, In Progress, or Needs Changes

The reviewer places his review document in Trello card so member1 can see the issues and comments.

* **Feedback Loop**

The review file is attached to the review task.

The reviewer notifies the task owner (Member1) after completing the review.

Member1 reviews the comments and addresses them accordingly.

Once updates are made, the task owner informs the reviewer for a second round of review.

The cycle repeats until all items are marked Approved.

**Once all parts are approved, the review task is marked complete.**

## **Change request Management:**

1. **Identify Change Need**

* **Who:** Any stakeholder (project team members, clients, sponsors)
* **How**: Raise a concern regarding scope, features, schedule, or risks
* **Tool**: Email, issue tracking system, or meeting discussion

1. **Submit Change Request**

* **Who:** Request originator (stakeholder)
* **How:** Fill out a Change Request Form, including details such as description, priority, justification, and affected areas
* **Tool:** Formal templates (Microsoft Word or Google Docs) or project management software (Trello)

1. **Review & Approval**

* **Who**: Change Control Board (CCB) or Project Manager
* **How:** Evaluate the analysis and determine whether to approve, reject, or defer the change request
* **Tool:** Meeting minutes, voting tools, or dedicated approval workflow systems (Trello)

1. **Plan the Change**

* **Who:** Project Manager and Planning Team
* **How:** Update tasks, schedules, resources, and communicate with stakeholders
* **Tool:** Project schedule tools (Microsoft Project, Gantt charts)

1. **Implement the Change**

* **Who:** Developers, Designers, or Relevant Execution Team Members
* **How:** Make required modifications in code, documentation, or system design
* **Tool:** Version control systems (GitHub) or relevant software development tools

1. **Verify & Close**

* **Who**: QA Team or Quality Assurance Specialists
* **How:** Test and validate the implemented change; close the request once verified
* **Tool:** Testing frameworks, checklist templates, or bug tracking tools

1. **Document & Communicate**

* **Who:** Project Manager or Documentation Team
* **How:** Update change logs, project documentation, and share updates with all stakeholders via email
* **Tool:** Shared documentation platforms (Google Drive) or communication tools (Google meet)

## **Problem Resolution Management:**

* Issues are logged into an issue-tracking system (Trello, GitHub).
* Each issue is assigned to a responsible team member using (Trello).
* Each assignee gets a notified via email
* Urgent issues are prioritized for quick resolution.
* Root cause analysis is done for recurring issues.
* Resolutions are documented and shared with relevant parties.

1. **Issue Tracking and Management**

**Who**: Project Manager  
**How**: All project issues should be documented and managed through the designated issue-tracking system (Trello/GitHub).  
**What**: documentation includes issue description, severity level, date reported, and initial analysis.

1. **Assignment Strategy**

**Who**: Project Manager

**How**:

1. The Project Manager reviews new issues daily and assigns them to appropriate team members via Trello.
2. System automatically generates email notifications to assignees.
3. The assignment includes defined resolution expectations and deadlines.

**What**: Assignment contains issue ID, description, priority level, expected resolution date, and required documentation deliverables.

1. **Root Cause Analysis**

**Who**: Assigned developers  
**How**: Structured analysis using the 5-Why methodology or Fishbone diagram for recurring or significant issues.  
**What**: Analysis document detailing:

* Problem statement
* Investigation findings
* Identified root cause
* Recommended preventative actions
* Implementation plan for preventative measures

## **Risk Management:**

**1. Identify Risks**

* **Who:** Project Manager, Team Leads, Stakeholders
* **How:**
  + Brainstorming sessions
  + SWOT analysis (identify strengths, weaknesses, opportunities, and threats)
  + Review historical project data
* **What:**
  + List of potential risks (e.g., technical, schedule, security)

**2. Assess Risks**

* **Who:** Project Manager, Risk Analyst
* **How:**
  + Likelihood: Probability of risk to happen (High/Medium/Low)
  + Impact: If the risk happens how, it will affect the product (High/Medium/Low)
  + Likelihood and Impact are scored according to

|  |  |  |
| --- | --- | --- |
| **Level** | Likelihood | Impact |
| High | Risk occurrence has a high probability according to previous projects (>70%) | Major delays, budget overruns, scope change, reputation damage |
| Medium | Risk might occur occasionally according to previous projects (30%-70%) | delays or rework, temporary drawbacks that can be handled in the next release only |
| Low | Risk is probability according to previous projects (<30%) | Risk that has negligible effect on the project |

* + Scoring (Risk Level = Likelihood × Impact)

|  |  |  |
| --- | --- | --- |
| Likelihood | Impact | Risk Level |
| Low | Low | Low |
| Low | Medium | Medium |
| Medium | Medium | Medium |
| High | Medium | High |
| High | High | High |

* **What:**
  + Prioritized risk register (High/Medium/Low)

**3. Mitigate Risks Plan**

* **Who:** Project Manager, Dev Team, QA
* **How:**
  + Avoid: Change the project plan to eliminate the risk or its impact.
  + Transfer: Shift the risk to someone else (e.g., using insurance, outsourcing, or contracts)
  + Mitigate: Take actions to reduce the likelihood or the impact of the risk if it happens.
  + Accept: Acknowledge the risk and prepare a contingency plan for how to handle it if it happens.
* **What:**
  + Risk response plan

4. Contingency Plan

* Who: Project Manager, Risk Owner
* How: Back up plan in case the risk happens
* What: Risk Response Plane

**4. Continuous Monitoring**

* **Who:** Project Manager, Risk Owners
* **How:**
  + Weekly reviews: to make sure all identified risks, and their responses are still relevant and up to date and make sure the mitigation plan is effective.
* **What:**
  + Updated risk status reports

**5. Risk Log Creation**

* **Who:** Project Manager
* **How:**
  + Maintain in Excel
* **What:**
  + Live Risk Register

**6. Risk Log Template**

**Who: Project Manager**

**How:**

* **Risk ID** → Unique identifier for tracking the risk.
* **Risk Summary** → Brief description of the identified risk.
* **Risk Category** → Classification (Product Risk-Project Risk).
* **Impact** → Level of negative effect on the project (Low/Medium/High).
* **Likelihood** → Probability of the risk occurring (Low/Medium/High).
* **Severity** → Overall importance (Impact × Likelihood).
* **Mitigation Type** → Action type:
  + **Acceptance Risk**
  + **Avoid Risk**
  + **Transfer Risk**
  + **Mitigate Risk**
* **Mitigation Strategy** → Proactive plan to reduce/avoid the risk.
* **Owner** → Person responsible for monitoring/managing the risk.

What:

* Systematic identification and documentation of all project risks
* Objective assessment and prioritization based on quantifiable metrics
* Clear accountability for risk management activities
* Tracking of mitigation progress and effectiveness
* Historical record for lessons learned and future project planning

## Configuration Management:

* **Configuration Management Plan**

**Who: Project Manager**

**How:**

* Identify and list all Configuration Items (CIs) via Repository structure.

📂 Online-Mobile-Store-Website

│

├── 📂 Project Management Plan

├── 📄 PMP.docx

├── 📄 CIL.xlsx

└── 📄 Risk Register.xlsx

│├── 📂 RTM

├── 📄Traceability-Matrix.xlsx

│├── 📂 Requirements

├── 📄CRS-SIQ.xlsx

├── 📄CRS.xlsx

├── 📄SRS.xlsx

└── 📄 Review Template.xlsx

**Tools Used:**

* Git (GitHub) – version control
* Trello – tracking changes and issues

**Configuration Item List (CIL):**

**Who:** Maintained By Project Manager

**How:** Implementation of version control and change tracking processes for:

**Documentation Items**

* **Requirements Documents**
  + Customer Requirements Specification (CRS)
  + Software Requirements Specification (SRS)
  + Requirements Traceability Matrix (RTM)

**Development**

* **Source Code**
  + Frontend components (HTML, CSS, JavaScript)
  + Backend services
  + Database schemas
* **Design Files**
  + System Architecture Diagrams
  + UI/UX Design Mockups and Wireframes
  + Sequence and Flow Diagrams
* **Test Files**
  + Test Cases
  + Test Results and Reports

**Project Management**

* **Project Control Documents**
  + Project Management Plan
  + Risk Register
  + Status Reports
  + Release Planning

**What:** inventory of all project artifacts requiring formal management

* **Baseline Strategy**

**Who: Project Manager**

**How:**

* Set baselines each Delivery:
  + Requirements
  + Design
  + Development
  + Testing
* Changes after baselines need formal approval via CCR.

What: Baseline for Each Delivery

## **Naming Conventions:**

**1. Design Artifacts**

Naming format:

DESIGN\_FEATXXX\_DESIGNTYPE\_XXX

• DESIGNTYPE options:

- CLASSDIAGRAM  
 - SEQUENCEDIAGRAM  
 - LOWLEVEL

• The final XXX is a serial number (e.g., 001, 002, ...).

Example: DESIGN\_FEAT001\_CLASSDIAGRAM\_001

**2. Software Requirements Specification (SRS)**

Naming format:

SRS\_FEATXXX\_XXX

• The final XXX is a serial number (e.g., 001, 002, ...).

Example: SRS\_FEAT002\_001

**3. Customer Requirements Specification (CRS)**

Naming format:

CRS\_FEATXXX\_XXX

• The final XXX is a serial number (e.g., 001, 002, ...).

Example: CRS\_FEAT005\_002

**4. Test Cases**

Naming format:

TC\_FEATXXX\_XXX

• The final XXX is a serial number (e.g., 001, 002, ...).

Example: TC\_FEAT007\_003

**5. Feature Codes**

|  |  |
| --- | --- |
| Feature Code | Feature Description |
| FEAT001 | Registration |
| FEAT002 | Login |
| FEAT003 | Homepage |
| FEAT004 | Restaurant Page |
| FEAT005 | Offers |
| FEAT006 | Loyalty Points |
| FEAT007 | Checkout |

Note:  
- The serial number (XXX) is incremented for each new file related to the same feature and type (e.g., DESIGN\_FEAT001\_CLASSDIAGRAM\_002 for the second class diagram of Registration).