Project Management Plan (PMP)

## **Project Overview**

* **Project Title**: Online Mobile Portal
* **Project Description**: Web based platform allowing:

1. Clients to buy phones / accessories
2. Suppliers to list and sell phones / accessories

* **Approach**: Water Fall Model, and this includes:
* **Testing**

Goal: ensure everything is ok

Activity: (End to End – Validation) Testing

* **Deployment**

Goal: produce website

Activity: review by checklist everything is ok

* **Requirement gathering**

Goal: understand what the system should do

Activity: meet customer and know their needs and create SRS, CRS, Questions

* **System design**

Goal: plan how system will work

Activity: design UI/UX and architecture of system

* **Development**

Goal: build system

Activity: create front end and Create backend System

* **Objective**: Provide full functionality for clients and suppliers – Website Deployment

## **Project Organization**

* Customer: Youssef Shabaan
* Supervisor: Mohamed Hassan
* Coach: Amr Mokhtar
* Project Manager: Abdulah El-Karamany
* Dev team: Mohamed Mumtaz, Youstina Atef, Mariam Abdelmoniem, Rana Saad

## **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

## **Assumptions and constraints:**

* Must be web-based / PC compatible
* Unique user IDs for all accounts
* Requires admin and supplier-specific features

## **Task Planning:**

* Each delivery breaks down into small tasks
* Tasks Assigned in (Management tool: Trello)
* Each task has at least **2** activities:

1. Assigned to developer
2. Assigned to a Reviewer

* Each assignee will be notified by his tasks via email
* Each delivery has **1** activity: Assigned for generating a Base Line

## **Communication Plan**

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

To Discuss:

1. Clear objective of what the team will achieve
2. Break down the delivery into small tasks
3. Assign tasks to Development Team and reviewers

* **Standup meeting:**
* **Days**: Monday-Thursday
* **Status**: Online
* **leader**: PM

To Discuss:

1. What is done
2. What will we do today
3. Any blocking issues

## **Review Strategy:**

* Internal Reviews: Weekly internal review meetings.
  + To Identify if there are any issues.
  + Each task is assigned to a team member to review it on (Trello).
  + They will be notified via email.
* Customer Reviews: At the end of each Release.

## **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.

## **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**

* **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.

## 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

└── 📄 README.md

* Control changes using a Configuration Change Request (CCR).
* Approve/reject changes via Change Control Board (CCB).
* Maintain version history and documentation.

**Tools Used:**

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

**Configuration Item List (CIL):**

* Requirements documents
* Source code
* Design files
* Test scripts
* Deployment scripts
* User manuals.
* **Baseline Strategy**
* Set baselines each Delivery:
  + Requirements
  + Design
  + Development
  + Testing
* Changes after baselines need formal approval via CCR.

## **Naming Conventions:**

**GitHub commits Format:**

* **Abbreviations:**
* **Feat 🡪 Feature**
* **Docs 🡪 Document**
* **How:**
* **Every change in files must be committed and the commit message must follow format.**
* **[Type]: [Action made on the file and then descriptive message that explains what was changed]**
* **Example:**
* Feat: added user login functionality
* Fix: resolved crash when clicking checkout button
* Docs: updated README with setup instructions
* Refactor: simplified order validation logic
* Test: added unit tests for shopping cart

**Document Naming Format:**  
[Type]-[Module]  
*Example:* REQ-Login-.docx

**ID Naming Format:**

[Document]\_[Module]\_[Number]

*Example:* CRS\_Login\_001