Project Management Plan (PMP)

**1. Project Introduction**

### 1.1 Project Name

**Car Purchasing System [ XCar]**

### 1.2 Purpose

The purpose of this project is to design, develop, and implement a Car Purchasing System that allows users to browse available cars, it’s features, and complete purchases online. The system aims to simplify the car buying process by providing a seamless and user-friendly digital experience.

### 1.3 Objectives

* To develop a fully functional web-based App that works on PC for purchasing new and used cars.
* To enable users to search, filter, make adds for cars based on various parameters (brand, price, etc.).
* To go live with the system within [ 2 months].

### 1.4 Project Scope

#### In-Scope:

* Use V-Model as the SDLC
* SIQ Analysis
* Requirements Analysis and design
* System Design
* System Implementation Using .NetCore Technology
* Validation Testing
* System Delivery
* User register / login through only the provided info that has been approved in the SIQ Document

#### Out-of-Scope:

* Mobile application
* Car maintenance
* Integration with government vehicle registration databases
* Unit Testing
* Integration Testing
* Hosting
* Payment Integration ( It would be implemented through a Third-Party company)
* Maintenance After approved delivery.
* Register / Login Using gmial / facebook or any other platform .

# 2. Constraints & Assumptions

**Constraints:**

* Web App that must work on Pcs / Laptops
* The id of the user would be the email address and it should be unique for registration and login
* Admin must has a specific features which would be only allowed to Admins , and this features would be : Create new user , Delete user ( admin allowed to delete normal user not admin users )

**Assumptions:**

* The system shall be accessible through a modern web browser (e.g., Chrome, Firefox, Edge).
* The system shall support secure access via HTTPS.
* The system shall not require installation on the user’s local machine.
* The system shall allow user authentication through a login page.

# 3. Roles & Responsibilities (R&R)

|  |  |  |
| --- | --- | --- |
| Name | Role | Responsibilities |
| Ebrahem Mostafa | Project Manager | Plan, monitor, and control the project. Facilitate communication, manage risks, and ensure timely delivery. |
| Omar Abdou | Developer | Implement backend and frontend features. Ensure code quality and integration. |
| Arsany Mounir | Developer | Work on frontend/backend tasks. Collaborate with tester and designer for functionality. |
| Abdelaziz Elsisi | Tester | Prepare test cases, perform testing, report bugs, and ensure the product meets requirements. |
| Anas Mohamed | Tester | Support test planning and execution. Perform regression and sprint testing. |
| Ehab Ahmed | Designer | Create UI/UX designs, wireframes, and ensure visual consistency of the website. |

# Section 4: Configuration Management (CM)

## 1. Purpose

The purpose of this Configuration Management Plan is to ensure all project artifacts (software code, documents, configurations, test scripts, etc.) are identified, tracked, and managed throughout the lifecycle of the software project. This ensures consistency, traceability, and integrity of project deliverables.

## 2. Configuration Items (CIs)

• Source code files (e.g., .java, .py, .js)  
• UI/UX design files  
• PMP documents  
• Requirement specs  
• Testing artifacts  
• Review and feedback forms  
  
Each CI will be uniquely identified and version-controlled using standardized naming conventions and version numbers.

## 3. Configuration Item List (CIL)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CID | CI NAME | CI TYPE | DESC | LOC |
| CI-001 | PMP\_Car\_Purchasing\_Website.docx | Documentation | Project Management Plan for the Car Purchasing Website | PMP/PMP\_Car\_Purchasing\_Website.docx(V) |
| CI-002 | CRS.xlsx | Requirements | Customer Requirements Specification document | Requirements/CRS/CRS.xlsx(V) |
| CI-003 | SIQ.xlsx | Requirements | Software Implementation Questions derived from the SRS | Requirements/SIQ/SIQ.xlsx(V) |
| CI-004 | SRS.xlsx | Requirements | Software Requirements Specification document | Requirements/SRS/SRS.xls(V) |
| CI-005 | review\_template.xlsx | Documentation | Template used for reviewing project documents | Review/review\_template.xlsx(V) |
| CI-006 | TestCases\_Login.xlsx | Test Artifact | Test cases designed to verify login functionality | Testing/TestCases\_Login.xlsx(V) |
| CI-007 | TestCases\_Registration.xlsx | Test Artifact | Test cases for user registration scenarios | Testing/TestCases\_Registration.xlsx(V) |
| CI-008 | TestCases\_Reserve.xlsx | Test Artifact | Test cases covering the reservation process | Testing/TestCases\_Reserve.xlsx(V) |
| CI-009 | TestCases\_Search.xlsx | Test Artifact | Test cases related to car search functionality | Testing/TestCase\_Search.xlsx(V) |
| CI-0010 | TestCases\_Advertisment.xlsx | Test Artifact | Test cases for car advertisement features | Testing/TestCase\_Advertisment.xlsx(V) |
| CI-0011 | TestCases\_WebApp.xlsx | Test Artifact | Comprehensive test cases for overall web application functionality | Testing/TestCase\_WebApp.xlsx(V) |
| CI-0012 | TestCases\_Admin.xlsx | Test Artifact | Test cases for admin features and controls | Testing/TestCases\_Admin.xlsx(V) |

## 4. Configuration Identification

Each CI will be named and stored with clear labels and version control. The following rules apply:  
• [Type][Feature]  
• All CIs are managed using Git repositories with branching strategy

## 5. Configuration Control

Purpose: restrict changing the main branch  
Instead:  
• Someone asks for a change (e.g., merge the update of the login feature to main branch)  
• The PM reviews it.  
• If approved, the change is made.

## 6. Roles and Responsibilities

|  |  |
| --- | --- |
| Role | Responsibility |
| Project Manager | Oversees configuration management processes, approves change requests |
| Dev Team | Versions and manages code changes |

## 7. Branching Strategy

Main Branch (main)  
• Used for stable, production-ready versions of the project.  
• Baselines and official releases are created from this branch.  
• A release is created at the end of each sprint or after final approval by merging tested changes from the develop branch into main.  
• Version tags are applied using semantic versioning (e.g., v1.0, v1.1-sprint2) to mark each release.  
  
Development Branch  
• Used for ongoing work, including feature development, bug fixes, and integration testing.

## 8. Commit Message Convention

Each commit message must follow the structure: <Ticket-ID> <Action> <Component/Feature>  
  
Example Commit Messages:  
• SCRUM-26 Upload testcases for login feature

## 9. Change Tracking

- Changes are tracked using commits with clear messages.  
- Team members contribute via pull requests or direct commits (based on internal agreements).  
- History of changes is maintained and reviewed via Git logs.

## 10. Folder Structure

Car-Purchasing/  
├── Design/  
├── PMP/  
│ └── Release Planning/  
├── Requirements/  
│ ├── CRS/  
│ ├── SIQ/  
│ └── SRS/  
├── Review/  
├── Testing/

Tools to be used:  
- Git / GitHub  
- Jira  
- Excel

# 5. Phases (Sprints Breakdown)

|  |  |  |  |
| --- | --- | --- | --- |
| Sprint | Weeks | Duration | Focus |
| Sprint 1 | Week 1 | 05-04-2025 – 11-04-2025 | * SIQ * CRS * PMP |
| Sprint 2-3 | Week 2-3 | 12-04-2025 – 25-04-2025 | * SRS * project timeline definition * RTM * Initial Wireframe * Test Cases |
| Sprint 4 | Week 4 | 26-04-2025 – 02-05-2025 | * UI/UX Design * User flow creation * Frontend & Backend development (Phase 1):   + Registration   + Login   + Car Listing |
| Sprint 5 | Week 5 | 03-05-2025 – 09-05-2025 | * Development (Phase 2):   + Search   + Reservation   + Admin features |
| Sprint 6 | Week 6 | 10-05-2025 – 17-05-2025 | * Testing   + Functional   + Usability   + Regression * Bug fixing * Final delivery |

# 6. Task Management (Jira)

The team uses **Jira** to manage, assign and track all project tasks. Each work item such as creating documents, developing features, or testing

is logged as a separate issue.

Step-by-step workflow followed:

**1-Task Creation:**

* A new Jira issue is created for each individual task (e.g., document writing, feature development, test case creation).
* A clear, concise description is written to explain the task's purpose and scope.
* The description includes:
  + **Owner**: The person responsible for executing the task (mentioned using @).
  + **Reviewer**: The person responsible for reviewing and validating the task (also mentioned using @).

**2-Assignment:**

* The issue is initially assigned to the **Owner** (the person who will work on it).

**3-Task Workflow:**

Tasks follow a structured flow: **To Do → In Progress→ In Review → Done**.

· **To Do**  
–The task is logged and ready to be started.

· **In Progress**  
– The owner begins working on the task.

· **In Review**  
– Once the owner completes the task:

* + They **change the status to "In Review"**.
  + They **reassign the task to the Reviewer**.

· **Review Feedback**

The review document is uploaded to that specific issue.

* If the **review passes** with no issues:
  + The Reviewer **changes the status to "Done"**.
* If the **review identifies gaps or issues**:
  + The Reviewer **changes the status back to "In Progress"**.
  + The task is **reassigned to the Owner** for corrections.

**7. Change Requests (CR)**

**Change Request Process**

1. **Submission**:
   * CRs must be submitted via a **formal CR Form** or documented email/meeting notes.
2. **Review**:
   * **Project Manager (Ebrahem Mostafa)** assesses impact on scope, cost, and schedule.
   * Stakeholders/technical team may be consulted for feasibility analysis.
3. **Approval/Rejection**:
   * Decision communicated to stakeholders.
   * Rejected CRs require a documented rationale.
4. **Implementation**:
   * Approved changes are added to the **product backlog** for prioritization in future sprints.

**Example CRs for This Project**

| Request | Impact | Decision |
| --- | --- | --- |
| Add "Price Comparison" feature for buyers. | +2 weeks development; $5k cost. | Approved for Phase 2. |
| Allow sellers to upload video reviews of cars. | +1 week; minor backend adjustments. | Rejected (Out of Scope). |
| Integrate a chat system for buyer-seller communication. | +3 weeks; $8k cost. | Pending stakeholder review. |

**8. Risk Management**

Identified risks and mitigation strategies for the project:

**Risk Register:**

| Risk | Likelihood | Impact | Mitigation Strategy |
| --- | --- | --- | --- |
| Developer Absence (e.g., due to illness) | Medium | Medium | Assign 2 backup team members. |
| Scalability Issues: Database struggles with large car listings. | Medium | High | Perform load testing early, and use cloud scaling (e.g., AWS/Azure). |
| Third-Party Delays: Payment gateway integration takes longer than expected. | Medium | High | Identify backup providers (e.g., PayPal). |
| Low User Registration Rates | Low | High | Simplify UI/UX. |
| Scope Creep: delay in timelines. | High | Medium | Prioritize "most important" features. |
| Performance Issues: Slow search results with complex filters. | Medium | Medium | Optimize database indexing. |

**Priority:** Combine the likelihood and impact to calculate priority using a matrix approach

| Likelihood | Impact | Priority Level | Description |
| --- | --- | --- | --- |
| Low | Low | Low Priority | The risk has a minimal likelihood of occurring and would cause a negligible impact. |
| Low | Medium | Low Priority | The risk is unlikely to occur but could cause a moderate level of disruption. |
| Low | High | Medium Priority | While the risk is unlikely, its impact would be significant if it did occur. |
| Medium | Low | Low Priority | The risk has a fair chance of occurring but would cause minimal disruption. |
| Medium | Medium | Medium Priority | The risk has an equal likelihood and impact, requiring balanced attention. |
| Medium | High | High Priority | The risk is somewhat likely to occur and could cause serious disruptions. |
| High | Low | Medium Priority | The risk is highly likely but would result in minimal disruption if it occurs. |
| High | Medium | High Priority | The risk is likely to occur and could significantly affect the project. |
| High | High | Critical Priority | The risk is both highly likely and has severe consequences, requiring immediate action. |

**9.Review Process:**

The review process ensures quality control and proper validation of tasks before they are marked as complete. The workflow is as follows:

1. **Task Assignment**
   * The **Project Manager (PM)** assigns a task with:
     + A **clear description** of the work.
     + An **@Owner** (responsible for execution).
     + An **@Reviewer** (responsible for validation).
2. **Task Execution (Owner)**
   * The **Owner** swipes the task to **"In Progress"** when working on it.
   * Once completed, the Owner:
     + Swipes the task to **"In Review"**.
     + **Reassigns the task to the @Reviewer** for validation.
3. **Task Validation (Reviewer)**
   * The **Reviewer** checks the task for completeness and accuracy.
   * **If NO gaps/issues are found**:
     + Comments **"No gaps"** on the task.
     + Moves the task to **"Done"**.
   * **If issues are found**:
     + Documents the issues in the **Review Template (Excel Sheet)**.
     + **Reassigns the task to the Owner** for corrections.
4. **Revisions & Completion (Owner)**
   * The **Owner** addresses the documented issues.
   * Once resolved, the Owner moves the task back to **"In Review"** and reassigns it to the Reviewer.
   * The Reviewer confirms fixes and moves the task to **"Done"**.

**Flow chart of Review Process:**

A screenshot of a computer screen

AI-generated content may be incorrect.

# 10. Communication Plan (Meetings & Tools)

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Tool / Platform | Frequency | Purpose |
| Daily Follow-Up | WhatsApp | Daily | Quick status updates, blockers, coordination |
| Team Sync Meeting | Discord | Every 2 days | Discuss progress, align tasks, clarify blockers |
| Sprint Review Meeting | Discord / In-person | End of each sprint | Review deliverables, gather feedback, plan next sprint |
| Customer Communication | Email | Weekly (or as needed) | Share progress, review documents, collect feedback |
| Documentation Sharing | GitHub | As needed | Share and collaborate on project documents |

# 11. Quality Management Process

Goal: Make sure the car-buying website works smoothly, with no major bugs, and gives users good experience.

1. Quality Planning

What we’ll check:

Car listings show correct info (price, model, photos).

Search filters (brand, price range) work properly.

Payment and trade-in calculators give accurate results.

How we’ll test:

Write simple test cases (e.g., "Test if ‘Used Toyota Camry’ filter shows correct cars").

Agree on what "done" looks like (e.g., "No errors during checkout").

2. Quality Checks During Development

Code Reviews: Developers check each other’s work for mistakes.

Early Testing: Test features as they’re built (e.g., try adding a car to the cart).

Tools: Use JIRA to track bugs, Excel for test cases.

3. Final Testing (Before Launch)

Test Everything:

Can users find, customize, and buy a car without issues?

Do payment/trade-in tools calculate correctly?

Fix Bugs: Log issues in JIRA, assign to developers, retest after fixes.

User Feedback: Let a few real users test the site and report problems.

4. Roles

Project Manager: Makes sure testing happens on time.

Developers: Fix bugs and review code.

Tester: Runs tests, reports bugs.

Users (UAT): Try out the site before launch.

5. Keep Improving

After launch, check for new bugs and fix them fast.

Learn from mistakes (e.g., "We missed testing trade-ins—add more tests next time").

# 12. Document Naming Conventions

To maintain consistency, traceability, and clarity across all project artifacts, the following naming conventions will be used for project documents:

|  |  |  |
| --- | --- | --- |
| Document Type | Naming Convention | Example |
| Project Management Plan | PMP\_<ProjectName>\_<Version>.docx | PMP\_CarPurchasing\_1.0.docx |
| Requirements Document | REQ\_<ModuleName>\_<Version>.docx | REQ\_CRS\_1.0.docx |
| Software Design Document | SDD\_<ModuleName>\_<Version>.docx | SDD\_UserLogin\_1.0.docx |
| Test Plan | TestPlan\_<Sprint/Module>.docx | TestPlan\_Sprint4\_1.0.docx |
| Test Case Document | TestCases\_<Function/Module>.xlsx | TestCases\_Reservation\_1.0.xlsx |
| Review Documents | Review\_<DocType>\_<Module>.docx | Review\_CR\_Reservation.docx |
| Change Request | CR\_<ChangeTitle>\_<Date>.docx | CR\_AddAdminPanel\_2025-04-10.docx |
| Meeting Minutes | Minutes\_<Topic>\_<Date>.docx | Minutes\_Sprint1Kickoff\_2025-04-01.docx |
| UI Design Files | UI\_<ScreenName>\_<Version>.fig | UI\_Homepage\_1.0.fig |

Versioning should follow the format, Major.Minor (e.g., 1.0, 1.1), and documents should be updated and baselined after every approved review cycle.