# **Overview**

The Online Mobile Store project aims to establish a robust framework for managing the changes and variations inherent in software development and implementation. This document serves as a roadmap throughout the project's lifecycle, ensuring the integrity and traceability of configuration items.

Objective: The main goal of this plan is to ensure that all modifications to the system are conducted in a coordinated and controlled manner.

Scope: The plan encompasses all functionalities of the Online Mobile Store web portal, including user account management, client and supplier interfaces, and transaction history features.

Objectives: Our objectives are to ensure the reliability of the online platform, maintain system consistency with project requirements, and systematically track modifications.

## **Configuration Management Strategy Overview**

Our configuration management strategy defines a clear path to maintaining consistency and control of project configurations across two primary branches:

1. The **‘DIV'** branch:

· Team members should upload their work when finished locally and create a pull request to the "Pre-Merge" branch.

2. The **'Pre-Merge'** branch:

· Pre-production branch.

· To review features and fix any issues before merging them into the main branch. This practice ensures that the code is stable and meets the requirements before deployment to production, and involving the entire team in this process.

3. The **'Main'** branch:

· production branch.

· At the end of the pre-merge branch, a pull request is created, and changes are validated and approved by the configuration manager.

**Configuration Management System**

* **System Name**: GitHub
* **Purpose:** GitHub acts as the primary repository for all project documentation, Its primary purpose is to ensure version control, access control, and traceability of project documents throughout the project lifecycle.
* **Naming convention**:
  + File Names:
    - Use descriptive names that reflect the content of the file..
    - Using a combination of {Project Name}\_{Document Name}.
    - Separate each section of the name with dashes or underscores for better readability.
    - Use a camel case for each part if it contains more than one word.
      * Example: OnlineMobileStore\_SRS, OnlineMobileStore\_user\_profile.js
  + Commit Messages:
    - Using a combination of Task ID in Trello and verb in the imperative mood (e.g., "Add", "Fix", "Update").
    - Keep messages concise but descriptive.
    - Example: Task id → Add login functionality, Fix typo in README, Update dependencies
  + Tag Names:
    - Use tags to mark specific releases or versions of your software.
    - For versioning, adopt a format that includes the {sprint or week number}\_{a number representing any changes or updates}
    - After all, changes have been created, handled, and approved, a new baseline is created
    - Example: start with (e.g., v1.0) for the first submission. Any changes made, reviewed, and approved would increment the version to (e.g., v1.1).
* **File structure**:
  + /Requirements: Contains documents related to project requirements.
  + /Design: Contains design documents and diagrams.
  + /Implementation: Contains folders for /code and /tests.
  + /Project Management (PM): Contains files related to project management such as :
    - * /CIL (Configuration Item List)
      * /Change (Change management documents)
      * /Problem (Problem management documents)
      * /Review (Review documents)
      * /PMP (Project Management Plan document)

**Configuration Items:**

The purpose of identifying and managing CIs is to ensure that changes to these items are controlled and documented throughout their lifecycle. This helps in maintaining the integrity and consistency of the configuration items, which is crucial for ensuring the reliability and quality of the system or project These items can include software, documentation, firmware, and other items that are relevant to the project or system being managed.

* Each item owner is responsible for updating the Configuration Item List (CIL) document with the appropriate information
* Each Configuration Item (CI) is uniquely identified starting with {CI\_00} and increments for each item added
* Each CI must adhere to a specific naming convention.
* The CIL must include a direct link to the GitHub file
* In cases where the link is not functional, a path must be provided. Additionally.
* Each CI must be added when created and updated with the configuration level when reviewed or approved according to review process, If the item not need to update frequent only once then it will be “Constant || Steady” OR if it need to update many time so it will be “Variable || Unsteady” .

**Baseline:**

Refers to a specific version of a configuration item (CI) or a set of CIs that has been formally approved and is used as a basis for further development, changes, or configuration management activities. They provide a snapshot of the project's configuration at a particular point in time, ensuring that changes are made in a controlled and documented manner

* When a pull request is initiated from the pre-merge branch, the Configuration Manager must review the request thoroughly. He should ensure there are no conflicts in the merge and anticipate and address any potential issues before accepting the final merge.
* Baselines should be taken at the end of each sprint, after all changes have been made, reviewed, and approved for implementation in the next sprint
* The Configuration Manager is responsible for determining the necessity of additional baselines to be taken within the sprint.
* Baselines should be named according to the format {Baseline keyword + baseline num}\_{Sprint or Week Number}.
* Each baseline must undergo a review process, similar to other project components.