Software Configuration Management Plan

Student Portal Management System

Document Control

Document Title: Software Configuration Management Plan (SCMP) – Student Portal Management System
Document ID: SCMP-SPMS-001
Version: 1.0
Status: Approved
Date: 27 December 2025

Prepared By: _____ (SCM Manager)
Reviewed and Approved By: _____ (Instructor / Team Lead)

1. Purpose

The purpose of this Software Configuration Management Plan (SCMP) is to define the policies, procedures, and responsibilities used to manage configuration items for the Student Portal Management System mini project. This plan establishes a structured approach for identifying, organizing, controlling, and tracking all project artifacts throughout the project lifecycle.

The SCMP ensures that all documents, source code, data files, test artifacts, and releases are properly versioned, traceable, and protected from unauthorized or uncontrolled changes. By following this plan, the project team aims to maintain consistency, transparency, and accountability while working collaboratively using GitHub.

2. Scope

This SCMP applies to all artifacts created, modified, or maintained during the Student Portal Management System project. These artifacts include project documentation, source code files, data files, test cases, baseline records, release notes, and audit reports.

The scope also includes repository-level items such as branches, commits, pull requests, tags, and releases managed through GitHub. This project focuses primarily on demonstrating effective Software Configuration Management practices rather than delivering a production-level software system.

3. SCM Organization and Responsibilities

The project is carried out by a team of six members. Each member is assigned specific SCM-related responsibilities to ensure proper control and coordination throughout the project.

The SCM Manager is responsible for preparing and maintaining this SCMP, coordinating baseline creation, verifying SCM compliance, and confirming readiness for releases.

The Repository Administrator manages the GitHub repository, ensures proper branch usage, oversees the pull request process, and maintains repository structure and access controls.

The Configuration Librarian is responsible for creating and updating the Configuration Item (CI) Register, ensuring that all configuration items are properly identified, versioned, and tracked.

The Change Control Lead manages the Change Request (CR) process, ensures that all change requests are formally documented, reviewed, approved or rejected, and properly logged in the Change Log.

Developer A is responsible for implementing the login and authentication functionality and supporting documentation updates related to that feature.

Developer B also acts as the Tester and is responsible for implementing the dashboard and core functionality, preparing test cases, executing tests, collecting evidence, and supporting configuration audits.

4. Configuration Item Identification

A Configuration Item (CI) is any project artifact that must be controlled because it impacts the functionality, documentation, testing, or traceability of the system. Once an artifact is identified as a CI, it must be version-controlled, recorded in the CI Register, and modified only through approved processes.

Configuration items in this project include documents such as the SCMP and requirements specification, source code files written in HTML, CSS, and JavaScript, JSON data files used by the system, test cases, baseline records, release notes, and audit reports.

All configuration items are documented and tracked in the CI Register located in the project documentation folder.

5. Naming and Versioning Standards

To maintain clarity and consistency, standard naming conventions are used across the project. Change Request files follow the format “CR-number-short-title”. Baseline records are named according to their baseline identifiers, such as “Baseline-Record-BL1” and “Baseline-Record-BL2”. Release notes are named using the release version, such as “v1.0-release-notes” and “v1.1-release-notes”.

Document versions follow a Major.Minor format. A major version update indicates a significant revision, while a minor version update reflects small improvements or clarifications. Source code versions are tracked using Git commits and repository tags rather than manual numbering inside files.

Baseline tags are identified as BL1 and BL2, while release tags are identified as v1.0 and v1.1.

6. Repository Structure and Control

The GitHub repository follows a fixed structure to ensure consistency and ease of navigation. The documentation folder contains all SCM and project documents. The source folder contains the prototype source code and data files. The tests folder stores test cases and verification artifacts. The releases folder contains release notes for each published version.

No project deliverables should be stored outside these folders unless approved by the SCM Manager.

7. Branching and Development Model

The project uses a simple and controlled branching model suitable for a small team. The main branch represents the stable and baseline-ready version of the project. Feature branches are created for major development tasks such as login functionality and dashboard implementation.

When a change request is approved, a dedicated branch is created to implement that change. All changes are merged into the main branch through pull requests. Direct commits to the main branch are discouraged to ensure review and traceability.

Each pull request should clearly describe the change being made and, where applicable, reference the corresponding change request.

8. Change Control Process

Change control ensures that all modifications to the project are intentional, reviewed, and traceable. Any proposed change must be documented using a Change Request form and recorded in the Change Log.

Once a change request is submitted, it is reviewed by the Change Control Lead and the SCM Manager. If approved, a dedicated branch is created to implement the change. After implementation, the change is reviewed through a pull request and merged into the main branch. The Change Log is then updated to reflect the final status of the change.

This process ensures that no uncontrolled changes are introduced into the system.

9. Baseline Management

A baseline represents a formally approved snapshot of the project at a specific point in time. Once a baseline is established, it cannot be modified directly.

Baseline 1 represents the initial SCM readiness of the project and includes the repository structure, SCMP, requirements document, CI Register, and change control templates.

Baseline 2 represents a stable working version of the system that includes the implemented prototype, approved change requests, updated documentation, test cases, and audit materials.

Each baseline is identified by a Git tag and documented using a one-page baseline record stored in the documentation folder.

10. Release Management

A release is a packaged version of the system made available through GitHub Releases. Each release includes a release tag and release notes describing the features and changes included.

Release v1.0 represents the initial working prototype of the Student Portal Management System. Release v1.1 represents an updated version of the system after approved change requests have been implemented.

Release notes are stored in the releases folder and are also published through GitHub.

11. Configuration Audits

Configuration audits are conducted to verify that the project complies with SCM requirements.

The Physical Configuration Audit (PCA) confirms that all required artifacts exist, are properly named, correctly versioned, and consistent with the CI Register. It also verifies the existence of baseline tags and releases.

The Functional Configuration Audit (FCA) verifies that system functionality matches the documented requirements and that all approved change requests have been correctly implemented.

Audit findings and evidence are documented in the Configuration Audit Report.

12. Approval and Sign-off

By signing below, the project team confirms that this Software Configuration Management Plan will be followed throughout the project lifecycle.

SCM Manager: _____ Date: _____

Repository Administrator: _____ Date: _____

Instructor / Team Lead: _____ Date: _____