1. Introduction

Software Configuration Management (SCM) is a critical discipline in software engineering that ensures software products are developed in a controlled, organized, and traceable manner. This Software Configuration Management Plan (SCMP) defines the configuration management activities, procedures, roles, and tools used during the development of the *Student Portal System*.

The Student Portal System is a small-scale web application developed as part of the Software Configuration Management mini project. Although the system itself is intentionally simple, it provides a practical environment for applying SCM principles such as configuration identification, version control, change management, baselining, release management, and configuration auditing.

This plan serves as a reference document for all team members and ensures that every artifact produced during the project lifecycle is properly managed and controlled.

2. Purpose and Scope

2.1 Purpose

The purpose of this SCMP is to establish a structured and systematic approach to managing the evolution of the Student Portal System. The plan defines how configuration items are identified, versioned, modified, and released while maintaining consistency and traceability.

By following this plan, the project team aims to:

- Prevent uncontrolled changes
- Maintain integrity of project artifacts
- Ensure accountability and transparency
- Support collaboration among team members
- Provide evidence of proper SCM practices for evaluation purposes

2.2 Scope

This SCMP applies to all artifacts produced during the Student Portal System project, including but not limited to:

- Project documentation

- Source code and configuration files
- Change request records
- Baseline records
- Release notes and audit reports

The plan covers SCM activities from initial repository creation through final system release and audit.

3. Project Overview

The Student Portal System is a simple web-based application developed using the Next.js framework. The system allows students to authenticate using a login page and access a dashboard that displays basic academic services. One core functional feature—viewing registered courses—is implemented to demonstrate system behavior.

Data used by the system is stored in a simple JSON file. The application does not aim to provide full production-level functionality. Instead, it focuses on demonstrating how software artifacts evolve under controlled SCM processes.

This project is developed by a group of six students and emphasizes disciplined use of Git and GitHub for configuration management.

4. SCM Organization and Responsibilities

4.1 SCM Organization

The project is developed by a team of six members. To ensure effective SCM implementation, specific roles are assigned. While team members may collaborate across tasks, each role has defined responsibilities to maintain accountability.

4.2 Roles and Responsibilities

Role	**Responsibility**
Project Leader	Coordinates project activities and ensures deadlines are met
SCM Manager	Oversees configuration identification, versioning, baselines, and audits

Developer 1	Implements login functionality and related features
Developer 2	Implements dashboard and core system functionality
Tester	Tests features and verifies change implementations
Documentation Manager	Maintains SCM documents and project records

The SCM Manager is responsible for enforcing this SCMP and ensuring compliance with SCM procedures.

5. Configuration Item (CI) Identification

5.1 Definition of Configuration Items

A Configuration Item (CI) is any artifact that must be controlled to ensure consistency and traceability throughout the project lifecycle. CIs are formally identified, versioned, and recorded in the CI Register.

5.2 Identified Configuration Items

The following artifacts are identified as CIs in this project:

- Software Configuration Management Plan (SCMP)
- Configuration Item (CI) Register
- Source code files (Next.js pages and components)
- JSON data files
- README documentation
- Change Request (CR) forms
- Change Log

- Baseline records
- Release notes
- Configuration Audit Report

Each CI is assigned a unique name, version number, owner, category, and status.

6. CI Naming and Versioning Conventions

6.1 Naming Conventions

To maintain consistency and clarity:

- Documentation files use descriptive names (e.g., `SCMP.docx`, `CI_Register.xlsx`)
- Source code files follow Next.js naming standards
- Baseline records include baseline identifiers (e.g., `Baseline_Record_BL1.docx`)
- Release documents include release version numbers

6.2 Versioning Conventions

Versioning is applied as follows:

- Documents use incremental version numbers (v1.0, v1.1)
- Git tags are used to mark baselines (BL1, BL2)
- GitHub Releases use semantic versioning (v1.0, v1.1)

These conventions allow clear identification of artifact evolution.

7. Version Control and Branching Strategy

Git is used as the version control system, with GitHub serving as the centralized repository.

The branching strategy includes:

- **main branch:** Contains stable and approved code

- **feature branches:** Used for developing new features or changes (e.g., `feature/login`, `feature/dashboard`)
- Changes are merged into the main branch using Pull Requests (PRs)

This approach ensures that all changes are reviewed and documented before integration.

8. Change Control Management

8.1 Change Request Process

All changes to configuration items must follow a formal Change Request (CR) process. This ensures that changes are evaluated before implementation.

8.2 Change Control Workflow

1. A Change Request is submitted using the CR form
2. The request is reviewed by the SCM Manager and Project Leader
3. Approved changes are assigned for implementation
4. Changes are implemented in a feature branch
5. Changes are tested and verified
6. Approved changes are merged into the main branch
7. The Change Log is updated

This process prevents unauthorized or undocumented modifications.

9. Baseline Management

Baselines represent formally approved snapshots of the project at specific points in time.

The following baselines are defined:

- **Baseline 1 (BL1):** Repository structure, SCM documents, and CI Register
- **Baseline 2 (BL2):** Working prototype after approved change requests

Each baseline is:

- Tagged in GitHub
- Documented using a Baseline Record
- Used as a reference point for audits and releases

10. Release Management

Release management ensures that stable versions of the system are delivered in a controlled manner.

The project includes two releases:

- **Release v1.0:** Initial working version of the Student Portal System
- **Release v1.1:** Enhanced version after implementing change requests

Each release includes release notes describing features, changes, and fixes.

11. Configuration Audits

Configuration audits verify that SCM procedures are followed correctly.

11.1 Physical Configuration Audit (PCA)

The PCA verifies that:

- Documents match repository contents
- CI names and versions are consistent
- Baselines and tags are correctly applied

11.2 Functional Configuration Audit (FCA)

The FCA verifies that:

- Approved change requests are implemented
- System functionality matches documented requirements

Audit findings are recorded in the Configuration Audit Report.

12. Tools and Environment

The following tools are used throughout the project:

- Git for version control
- GitHub for collaboration and repository management
- Next.js for application development
- Visual Studio Code / Cursor for coding
- Microsoft Word and Excel for documentation

13. Risks and Mitigation

Potential risks include:

- Inconsistent documentation
- Uncontrolled changes
- Merge conflicts

Mitigation strategies:

- Strict adherence to SCM procedures
- Regular commits and reviews
- Clear role assignments

14. Conclusion

This Software Configuration Management Plan provides a comprehensive framework for managing the Student Portal System project. By applying structured SCM practices, the project ensures controlled evolution, traceability, and quality of all artifacts. The plan supports collaboration, accountability, and successful project completion.