1.Introduction

1.1 Purpose

This Software Configuration Management Plan (SCMP) defines the comprehensive configuration management framework that will be applied throughout the development lifecycle of the Simple Event Registration System, a mini-project undertaken for the Software Configuration Management course. The plan establishes the policies, roles, responsibilities, procedures, tools, schedules, surveillance mechanisms, metrics, and continuous improvement processes required to ensure controlled, traceable, and auditable evolution of all project deliverables. It is fully aligned with IEEE Std 828-2012 and the detailed guidance provided by the project

1.2 Scope

This SCMP applies to all phases of the project from initial planning through final submission and covers every work to be done the team,which ranges from requirements specification , design , coding , testing, baseline records etc. and the complete Github repo structure

1.3 Organizational Relationships

The project is to be done by a student team of 6 members from the software Department in AASTU. The team will do this project under the course Software Configuration Management. For the project all will communicate with the instructor but to keep up with the theme of the course the SCM Manager (Leul) will communicate with the course instructor on SCM compliance.

1.4 References

- IEEE Std 828-2012: Standard for configuration management in systems and software engineering
- Lecture 5 slide on SCMP
- The project description

2. Criteria for identification of configuration Items

All artifacts that require version control, traceability, or formal release are designated as Configuration Items (CIs) and placed under strict configuration management. The following categories are explicitly included:

- **Requirements and planning documents:** in this case due to the project having simple requirements we will prepare a simple Project README for it
- **Design artifacts**: Simple UI mockups of the pages
- **Source Code and configuration files**
- **Test artifacts :** simple files to test system functionalities
- **SCM supporting records :** like CI register , change request forms, changes logs , baseline records, release notes and audit reports

2.1 Naming Conventions

All items must follow structured naming to ensure clarity and consistency:

- **Documents:** `DOC-<Name>-v<Version>.md/pdf`
- **Source Code:** `SRC-<Component>-v<Version>.<ext>`
- **Data Files:** `DATA-Events-v<Version>.json`
- **CR Forms:** `CR-00X.pdf`
- **Baseline Records:** `BL1-Record.pdf`, `BL2-Record.pdf`
- **Releases:** `REL-v1.0.zip`, `REL-v1.1.zip`

3. Limitations and Assumptions

This SCMP has been tailored to the scale and scope of the Simple Event Registration System, which is a coursework mini-project rather than a full production system. As such, several assumptions guide the development and planning of configuration management activities. First, the project assumes that all team members will consistently use GitHub for version control, branching, merging, and release tagging. It also assumes full participation and adherence to the SCM procedures described in this plan, as the effectiveness of configuration management depends on disciplined compliance.

Limitations include the absence of automated CI/CD pipelines, enterprise-level CM tools, or dedicated configuration management software. Since the project is intentionally simple, certain industrial practices such as controlled workspaces, automated audits, or formalized build management are kept lightweight. The project timeframe is short, and the codebase is small, so all procedures are scaled accordingly while still maintaining compliance with the principles of IEEE 828-2012.

4. Roles and Responsibilities

SCM Manager (Leul)

The SCM Manager oversees all SCM activities and ensures the team follows the SCMP. Responsibilities include maintaining the CI Register, verifying versioning and naming conventions, approving baselines, reviewing pull requests, and communicating SCM-related updates to the course instructor. The SCM Manager is the central authority for all configuration decisions.

Project Developers (Team Members)

Developers are responsible for implementing features assigned to them, creating feature branches, performing commits with proper messages, ensuring code consistency, and updating documentation relevant to their work. They must not modify any baselined item without an approved Change Request.

Documentation Officer

Responsible for maintaining the accuracy and currency of documentation, including the SCMP, CI Register, Change Log, Baseline Records, Release Notes, and project README. Ensures proper document versioning and organization within the /docs directory.

Change Control Board (CCB)

The CCB consists of all team members and is chaired by the SCM Manager. It reviews all submitted CRs, evaluates their necessity and impact, and approves or rejects changes. All approved CRs are scheduled for implementation and tracked in the Change Log.

Configuration Auditor

Conducts the Physical Configuration Audit (PCA) and Functional Configuration Audit (FCA). Confirms that repository items match documented CIs, verifies that naming conventions and version numbers are correct, and ensures all approved CRs have been implemented before release.

5. Versioning Rules

Version control for all Configuration Items (CIs) in this project will follow a structured and consistent versioning scheme based on *Semantic Versioning (SemVer)* adapted for academic project scale. Version identifiers will follow the format:

vMAJOR.MINOR.PATCH

- **MAJOR version** changes occur when a baseline is established, when significant functionality is added, or when backward-incompatible modifications are introduced.
- **MINOR version** increments occur when new features or CR-approved enhancements are implemented without altering the overall system architecture.
- **PATCH version** increments are reserved for minor corrections such as documentation updates, formatting revisions, and bug fixes that do not affect system behavior.

All documents, code files, and SCM artifacts must include the version in the file name or metadata section. Only the SCM Manager is authorized to increment the MAJOR version number, and all version updates must be recorded in the Change Log. Baselines and releases must be tagged in GitHub using the format **BL1**, **BL2**, **v1.0**, and **v1.1** as defined in the assignment guidelines.

6. Branching Model

The project adopts a simplified but structured branching strategy to ensure controlled development and prevent conflicts. The model is based on Git Feature Branch Workflow and consists of:

6.1 Main Branch

The **main** branch contains the most stable version of the project. Only approved and reviewed changes from feature branches are merged into main. Baselined versions and releases are tagged on this branch.

6.2 Feature Branches

Each new feature, bug fix, documentation update, or CR implementation is developed in a dedicated feature branch. Naming follows the structure:`feature/<feature-name>`, `fix/<issue-name>`, `docs/<document-update>`, `CR/<CR-number>`

6.3 Pull Request Workflow

All feature branches must be integrated into main through a Pull Request (PR). PRs must include: summary of changes, related CR number ,reviewer assignment (SCM Manager + at least one developer)

7. Change Control Process

A formal change control procedure regulates all modifications to baselined Configuration Items. No change may be executed or merged without an approved Change Request (CR).

7.1 Change Request Submission

Any team member may initiate a CR by completing the standardized CR form. Each CR must include:

- CR ID (e.g., CR-001)
- Description of proposed change
- Affected CI(s)
- Justification
- Impact analysis
- Suggested priority level

7.2 Evaluation and Approval

The Change Control Board (CCB), chaired by the SCM Manager, evaluates the CR based on technical impact, effort required, and alignment with project objectives. The CCB may:

approve , reject , request modification or clarification . and they must be registered in change log

7.3 Implementation

Once approved, the developer assigned to the CR creates a feature branch named:

`CR/<CR-number>-implementation`,The developer implements the change, documents updates, and submits a Pull Request referencing the CR.

7.4 Verification

The SCM Manager verifies the correctness and completeness of implemented changes. Approved PRs are merged into main and reflected in the Change Log.

8. Baseline Management

8.1 Baseline 1 (BL1) – Initial Documentation Baseline

Includes:

- Repository structure
- Initial project README
- SCMP
- CI Register
- UI mockups
- First version of source files (skeleton)

8.2 Baseline 2 (BL2) – Working Prototype Baseline

Includes:

- Implemented login page
- Main module/service list page
- One core functional action
- CR fixes completed after BL1
- Updated documentation

8.3 Baseline Control

After a baseline is established:

- Items within the baseline cannot be modified without an approved CR
- All changes must follow the Change Control Process
- The SCM Manager monitors changes to ensure baseline integrity

9.Tools used

- Git/Github
- Vs code
- Github issues
- Google docs
- Markdown tools for CI register, baseline records , release note and audit reports