ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY
UNIVERSITY FOR INDUSTRY

College of Engineering

Department of Software Engineering

Software Configuration Management (SWEG5103)

Software Configuration Management Plan (SCMP)

Student Login Module, Version: 1.0, Date: Dec. 28, 2025

1) Abduselam Sultan ———— ETS0024/14
2) Abel Tessema ———— ETS0027/14
3) Abel Maireg ———— ETS0028/14
4) Abel Mulat ———— ETS0029/14
5) Abel Seyoum ———— ETS0032/14
6) Abel Bogale ———— ETS0035/14

Submitted to – Instructor Yibrah G.

Table of Contents

1. Introduction....1

1.1. Purpose.... 1

1.2. Scope....1

1.3. Definition and Acronyms....1

2. SCM Management....2

2.1. Organization and Roles.... 2

2.2. Tools and Environment.... 2

3. Configuration Identification....3

3.1. Naming Conventions.... 3

3.2. Configuration Items (CIs)....3

4. Configuration Control.... 4

4.1. Change Request (CR) Process.... 4

4.2. Branching Model....4

5. Status Accounting and Audits.... 5

5.1. Baseline Management....5

5.2. Configuration Audits.... 5

1. Introduction

1.1. Purpose

The purpose of this Software Configuration Management Plan (SCMP) is to establish the SCM activities for the **Student Login Module** project. This plan defines the processes for identifying, controlling, and tracking the software configuration items (CIs) throughout the project lifecycle. It ensures that the integrity of the software artifacts—including the Next.js source code, Prisma database schemas, and documentation—is maintained.

1.2. Scope

This SCMP applies to all phases of the mini-project development, from initial setup to the final release (v1.1). It covers:

- **Source Code:** All `.tsx`, `.ts`, and `.js` files within the `src/` directory.
- **Database Configuration:** Prisma schema files and migration history.
- **Documentation:** Requirement specifications, SCM plans, and audit reports.
- **Build Scripts:** `package.json` and `next.config.js`.

1.3. Definition and Acronyms

- **SCM:** Software Configuration Management
- **CI:** Configuration Item
- **CR:** Change Request
- **CCB:** Configuration Control Board (Simulated by the project team)
- **Baseline:** A snapshot of the project at a specific point in time (e.g., BL1, BL2).

2. SCM Management

2.1. Organization and Roles

The SCM activities will be managed by the project team. The following roles are assigned:

Role	Responsibilities	Assigned to
Configuration Manager (CM)	SCM planning, creating baselines, and managing the repository.	Abel Tessema, Abel Seyoum
Developer	Implementing features (Next.js/MUI) and fixing bugs identified in CRs.	Abel Mulat, Abel Bogale
Auditor	Conducting the Physical and Functional Configuration Audits (PCA/FCA).	Abel Seyoum, Abel Maireg

2.2. Tools and Environment

The following tools will be used to enforce SCM controls:

- **Version Control:** Git (Distributed Version Control System).
- **Repository Hosting:** GitHub.
- **Editor:** Visual Studio Code.
- **Framework:** Next.js with Material UI (MUI).
- **Database ORM:** Prisma.
- **Documentation:** Google Docs / PDF.

3. Configuration Identification

3.1. Naming Conventions

All Configuration Items (CIs) will follow a standard naming convention to ensure consistency.

- **Files:** `kebab-case` (e.g., `page.tsx`, `scm-plan-v1.0.docx`).
- **Branches:** `type/description`
    - Feature: `feature/login-ui`
    - Bugfix: `fix/button-color`
    - Main: `main`
- **Tags/Baselines:** `BL[#]` (e.g., `BL1`, `BL2`).
- **Releases:** `v[Major].[Minor]` (e.g., `v1.0`, `v1.1`).

3.2. Configuration Items (CIs)

The following categories of CIs are identified for this project:

1. **Project Documentation** (e.g., SCMP, Specs)
2. **Source Code** (e.g., Next.js App Router files)
3. **Database Schemas** (e.g., `schema.prisma`)
4. **Build Configuration** (e.g., `package.json`)

(Refer to the CI Register for the full list of tracked items.)

4. Configuration Control

4.1. Change Request (CR) Process

Changes to baselined items (e.g., changing the UI after Release v1.0) must follow this process:

1. **Request:** A Change Request (CR) Form is completed, detailing the issue and proposed fix.
2. **Assessment:** The Configuration Manager reviews the CR for impact.
3. **Approval:** The Change is approved or rejected.
4. **Implementation:** If approved, a new branch (e.g., `fix/cr-01`) is created.
5. **Verification:** The change is merged via Pull Request (PR) only after verification.

4.2. Branching Model

We will utilize a **Feature Branch Workflow**:

- **`main`**: Contains the stable, baselined production code. Direct commits are restricted; changes come via Merge Requests.
- **`feature/*`**: Used for developing new functionality (e.g., Database setup, UI creation).
- **`fix/*`**: Used for resolving bugs or implementing Change Requests (CRs).

5. Status Accounting and Audits

5.1. Baseline Management

Two formal baselines will be established:

- **Baseline 1 (BL1):** Established after project initialization and documentation approval. Contains the project skeleton and plan.
- **Baseline 2 (BL2):** Established after the implementation of the first Change Request (CR-01). Contains the refined prototype.

5.2. Configuration Audits

- **Physical Configuration Audit (PCA):** Verification that the files in the GitHub repository match the versions listed in the CI Register.
- **Functional Configuration Audit (FCA):** Verification that the software functionality (Login, Button Colors) meets the requirements and CR descriptions.