**[Table of Contents]**

# Introduction

1.1 Purpose

1.2 Scope

1.3 Key Terms

1.4 References

# CCM Management

2.1 Organization

2.2 Responsibilities

2.3 Applicable policies, directives, and procedures

1. **CCM Activities**

3.1 Configuration Identification

3.2 Configuration Control

3.3 Configuration Status Accounting

3.4 Configuration Evaluations and Reviews

3.5 Interface Control

3.6 Subcontractor/ Vendor Control

3.7 Release Management and Delivery

1. **CCM Schedules**

4.1 **S**equence and coordination of CCM activities

4.2 Relationship of key CCM activities to project milestones or events, such as:

4.3 Schedule either as absolute dates, relative to CCM or project milestones or as sequence of events.

4.4 Graphical representations can be used here.

1. **CCM Resources**

5.1 Identifies environment, infrastructure, software tools, techniques, equipment, personnel, and training.

5.2 Key factors for infrastructure

5.3 Identify which tools are used in which activity.

1. **CCM Plan Maintenance**

6.1 Who monitors the plan?

6.2 How frequently updates are to be performed?

6.3 How changes to the Plan are to be evaluated and approved?

6.4 How changes to the Plan are to be made and communicated?

6.5 Also includes history of changes made to the plan.

# Introduction

The CSwap Configuration Management Plan describes the processes for documenting the software configuration management requirement for the CSwap web application. The CSwap Configuration Management Plan addresses configuration management for our application. Configuration management for hardware, change management, development, and database. The CSwap Configuration Management Plan describes the best practices for the CSwap web application. This document contains:

1. Process description
2. Process roles and responsibilities
3. Process flows and activities
4. Process scheduling
5. Process resources and
6. Process planning

# Purpose

A CI is any particular entity for configuration management, but can also represent multiple CIs. Grouping them can help ease identification. The purpose of a Software Configuration Management Plan, is to establish and maintain integrity for our application using:

Configuration Identification

Defines the functional and physical characteristics of a Configuration Identity in enough detail so that it may be planned, evaluated, created, maintained, and supported.

Configuration Control

The process of approving or disapproving changes made to CI. This includes configuration for Configuration Identifications, and allows the baseline to be updated.

Configuration Status Accounting

The process of creating & organizing information to find how well the Configuration Management Plan performs. A single element of configuration management includes recording and reporting information needed to manage the configuration.

Configuration Audit

Audits are performed to verify that an individual CI, or CIs that create a baseline, follows a specified standard or requirement. This can include functional or physical audits.

# Scope

The scope of this plan is all CI that are created for the system’s software development life cycle. All members of the group will use this document to provide processes and procedures that will be used to verify control over the applications components. The CSwap Configuration Management Plan exists to create integrity in testing, and reduce the creation of bugs in the CSwap production environment.

# 

# Key Terms

CSwap - College Swap

CI - Configuration Item

# References

<https://www.guru99.com/software-configuration-management-tutorial.html#5>

<https://www.stakeholdermap.com/project-management/project-milestones.html>

# CCM Management

# OrganizationThe Configuration Management organization consists of members of the CSwap team. The project is managed under the context of scrum meetings, led by the scrum master. In the scrum meetings decisions regarding configuration management are made. Managerial organization is primarily held responsible by the scrum master and product owner, whereas technical organization is held responsible by the development team.

# Responsibilities

Configuration Management responsibilities are listed and detailed in the following table under each organizational role. For each role, their purpose and objectives, scope of authority, and operational procedures are discussed. All roles are affiliated with the CSwap team and their responsibilities are effective throughout the entirety of the project lifecycle or the duration of their role.

| **Roles** | **Required Responsibilities** |
| --- | --- |
| Scrum Master | * Makes sure that development is on track * Works on Documentation |
| Product Owner | * Works on Documentation * Makes sure that all parts of the deliverable are getting completed |
| Developer | * Works on the application by developing the required story tasks. * Works on Documentation |

# 

# Applicable Policies, Directives, and Procedures

# Currently, there are no known applicable policies, directives, or procedures that hold constraints to the development of the Configuration Management plan.

# CCM Activities

# Configuration Identification

# Identifying Configuration Items

# Activities identified for the CCM include the CSwap baseline, the system CIs (expanding to the sublevel CIs of database, source, user interface and methods), and the configuration identification documents (expanding to specifications, plans, and procedures). These are decided by the CSwap team as items that may require the need of modifications over a timeline.

# Naming Configuration Items

The naming convention entails unique identifiers for each item and their versions. The naming of items should be based on the file type, CI title, or the combination of both. Words in the name should be separated by underscores, not spaces, and ending with file type. If the item is a CI document and is required for a deliverable, the leading words should be the team name, followed by the deliverable and its number. The following is an example:

| **File Type/CI Title** | **Name Identifiers** |
| --- | --- |
| Configuration Management Plan | CSwap\_Deliverable\_4\_Configuration\_Management.pdf |

Versions should be labeled as “vX.X” where X.X defines the version number (e.g. v1.0, v1.1, etc.).

# Acquiring Configuration Items

The configuration management library is maintained by the CSwap team. CIs are stored and to be accessed from the remote Github repository. This will contain all software and documentation CIs. Access to items is given to authorized users and non-authorized users will need to request access.

# Configuration Control

# Requesting changes

Currently, changes to CIs may only be requested through the remote Github repository (https://github.com/JacquesLJT/COS420\_Project/issues) as an issue or in the form of a pull request. To first gain access to requesting changes, contact must be made with one of the members of the development team to gain access to the repository.

# Evaluating changes

The evaluation of changes are to be done during meetings or in our discord. The development team is to discuss and weigh the merit of change requests in order of requests received chronologically.

# Approving or disapproving changes

Changes can be approved or disapproved either during a scrum meeting or under pull requests/issues in a conversation or in the discord. In the advent of differing opinions, action taken regarding changes should be done through a voting process by the development team. Disapproved changes shall be archived and can be referenced again in the future.

# Implementing changes

Depending on the size, complexity, and urgency of the request, changes will be added to the backlog and marked as critical or non-critical. Critical changes will be developed and released in the upcoming major version, whereas non-critical changes will be rolled out in the next minor version.

# Configuration Status Accounting

# Metrics to be tracked and reported and type of report

CIs are to be tracked on a Kanban board. Items may be linked to Github Issues and progress is to be reported in the conversation of Github Issues. Following the release of a version, there should be a report made detailing all the changes made. Change Requests that are submitted under Issues should have its progress reported by marking its status with the following: Submitted, Reviewed, Assigned, Opened, Closed..

# Storage and access control of status data.

Storage and access of status data can be viewed and changed by authorized users, such as the CSwap team. This information can be found under the project’s Github.

# Configuration Evaluation and Reviews

Configuration evaluations and reviews are to be performed by the development team during scrum meetings. This is to ensure that CIs have been identified accurately and created accordingly. Evaluations and reviews are to be done on the same schedule as the team’s weekly scrum meetings with the participants being the development team.

# Interface Control

Coordination of changes to CIs with changes to interfacing items outside of the scope of the plan will be captured and detailed in the near future under the Interface Control Document. Under this document, critical interfaces will be identified and specified with requirements for the system.

# Subcontractor/Vendor Control

# Due to the fact that the application is built with the use of Firebase, the development team will need to monitor changes to such services and adjust accordingly if changes interfere with the functionality of the application.

# Release Management and Delivery

The build of the application is automatically done through continuous integration as the development team pushes progress to the remote repository. The time of releases and deliveries will be decided upon by the development team.

# CCM Schedules

# Sequence and coordination of CCM activities

Coordination of CCM activities can be found in our Sprint backlog which we update weekly keeping track of the progress of the activities.

# Relationship of key CCM activities to project milestones or events, such as:

# As CIs are identified, specific milestones will be identified in conjunction. As the project progresses, this section of the configuration management plan will be updated accordingly. Minor releases will be related to major releases in such a way that minor releases will occur concurrent to the development of major releases. Depending on the timing of minor releases, planned minor releases may be merged with major releases. The establishment of the configuration baseline is planned for when the development team has control of the application design and is ready for application testing.

# Schedule either as absolute dates, relative to CCM or project milestones or as sequence of events.

The graphical representation in Section 4.4 displays an idealized sample of the project’s release cycle and is subject to changes as the development team progresses. For this project, releases will be planned around a weekly lifecycle which will be concurrent with project milestones. Minor releases will occur weekly with the updates we made that week. After W4 the process repeats.

# Graphical Representations

| **W1** | | **W2** | | **W3** | | **W4** | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Minor  Development | Minor Release |  | |  | |  | |
|  | | Minor  Development | Minor Release |  | |  | |
|  | |  | | Minor  Development | Minor Release |  | |
|  | |  | |  | | Minor  Development | Minor Release |

# CCM Resources

# Identifies environment, infrastructure, software tools, techniques, equipment, personnel, and training.

Development Tools: VSCode

Database Tools: Firebase

Environment Tools: JavaScript version ES6, React 17, Firebase API version 9

VCS Tools: Git, GitHub, Zenhub

Notable Libraries(API): Firebase

# Key factors for infrastructure:

Our development tools were chosen due to being industry standard for developing JavaScript code and web applications, respectively.

Firebase was chosen as our backend database due to its ease of use and massive support from Google which meant there is ample documentation.

We chose to use version 9 of Firebase as this was the latest version of Firebase and offered increased performance due to the change from namespaced to modular architecture for web applications.

GitHub was chosen as our VCS tool due to its great security and due to it being a contemporary industry standard for version control.

Firebase was used for everything from application hosting to being the backend database for our application. Firebase was chosen thanks to its creation by Google and massive amounts of documentation.

# Identify which tools are used in which activity.

* Visual Studio Code is used to edit our JavaScript files
* GitHub is an application where we can share files with each other and keep up with changes made to the application
* Zenhub is a web application where we can keep track of what needs to be completed and who needs to work on it
* Firebase is used to host our application and to handle database management

# CCM Plan Maintenance

# Who monitors the plan?

Scrum Master

# How frequently updates are to be performed?

Weekly

# How changes to the Plan are to be evaluated and approved?

The group will evaluate all changes made to the Plan and decide as a whole whether to approve or not.

# How changes to the Plan are to be made and communicated?

We will maintain a Kanban board using Zenhub that contains cards that indicate that changes to the CCM need to be made.

# Also includes history of changes made to the plan.

| Version History | Name | Date |
| --- | --- | --- |
| 1.0 | First Release | 4/3/2022 |
|  |  |  |