Table of Contents: 1

1. Introduction

1.1. Purpose

1.2. The purpose is to define and maintain the integrity of Squared's outputs and processes, making them accessible to all concerned parties.

1.3. Scope

1.4. The scope entails the details of the processes to handle released systems and the distributed system since each member is in a different location. Lastly, manage the multiple versions of the software that is supported through our various Git branches.

1.5. Key Terms

Table 1.5.1, Plan Acronyms, outlines and defines terminology that is to be used throughout this plan. **Table 1.5.2, Plan Definitions,** defines acronyms that are to be used throughout this plan.

ACRONYM	**MEANING ID**
API	Application Programming Interface 1.5.1.1
COPPA	Children's Online Privacy Protection Rule 1.5.1.2
CM	Configuration Management 1.5.1.3
CMP	Configuration Management Plan 1.5.1.4
CSS	Cascading Style Sheets 1.5.1.5
DB	Database 1.5.1.6
GDPR	General Data Protection Regulation 1.5.1.7
GUI	Graphical User Interface 1.5.1.8 Hypertext Markup
HTML	Language 1.5.1.9 Javascript 1.5.1.10
JS	
JSX	Javascript XML 1.5.1.11
SCM	Software Configuration Management 1.5.1.12
SOP	Standard Operating Procedures 1.5.1.13
UI	User Interface 1.5.1.14
XML	Extensible Markup Language 1.5.1.15

Table 1.1: Plan Acronyms

TERM	**DEFINITION**	**ID**
Back-End *(adj.)*	Descriptor for any component, algorithm or interface allocated for performing application tasks that cannot be directly accessed by an end user, such as the manipulation of data and transferral of data to or from databases.	1.5.2.1
CSS *(n.)*	Cascading Style Sheets. A markup language designed to apply decorations and graphics to HTML components and webpages, such as colors, background images and custom text fonts.	1.5.2.2
Developer *(n.)*	A member of the *Squared* configuration management team and Team Squared development team, responsible for: ● The creation and software implementation of user stories within the *Squared* application; ● Consistently and frequently checking for and resolving any conflicts related to pushes made towards the *Squared* repository, including merge conflicts and software configuration issues; ● Pushing at least one to two significant component or functionality updates to the *Squared* repository (the required significance of these pushes of which are determined by Team Squared in accordance with recent demands and deadlines) per scrum period; ● Maintaining the configuration of the source code of *Squared*, and; ● Contributing to the production and publication of documentation and deliverable materials pertaining to specific components of *Squared*, specific contributions made to the *Squared* repository, or the manner in which Team Squared functions and operates.	1.5.2.3
Front-End *(adj.)*	Descriptor for any component, algorithm or interface allocated for performing application tasks that are directly accessed by an end user, such as the display of an interactive graphical user interface.	1.5.2.4
HTML *(n.)*	Hypertext Markup Language. A markup language used for displaying text, forms, images and other basic web components onto a webpage.	1.5.2.5
JavaScript *(n.)*	A scripting and programming language that is frequently used in websites to apply additional functionality to web pages, such as the processing of input given by users through forms.	1.5.2.6
Open-Source *(adj.)*	Descriptor for any software distributed under a license which grants other users the ability to use, modify and distribute the software in any way and for any purpose.	1.5.2.7

Scrum Period *(n.)*	A period of time (typically consisting of one or two weeks within Team Squared) in which a member of Team Squared is designated the team's scrum master, and within which all members of Team Squared are expected to make at least one or two relatively significant contributions to the *Squared* repository.	1.5.2.8
Scrum Master *(n.)*	A member of Team Squared who is designated immediately after a scrum period to perform a variety of team-related tasks relating to team leadership and the running of a weekly team scrum meeting.	1.5.2.9
Scrum Meeting *(n.)*	A weekly development meeting within Team Squared led and coordinated by the team's designated scrum master for the purposes of establishing what development tasks have been completed during the previous scrum period, the development team's priorities should be for the scrum period, and discussing any user feedback given towards *Squared.*	1.5.2.10
ReactJS *(n.)*	An open-source, front-end Javascript library used for building user interfaces using custom defined React components.	1.5.2.11
Javascript XML *(n.)*	Javascript Extensible Markup Language. A markup language used within ReactJS that allows the easy integration of Javascript and HTML by converting HTML tags into React elements.	1.5.2.12
User Interface *(n.)*	A front-end display designed to neatly convey information or provide intuitive functionality to an end user.	1.5.2.13

Table 1.2: Plan Definition

2. SCM Management

2.1. Organization

Team Squared's configuration management team is composed of each member. The current scrum master manages our team. The configuration management is achieved via weekly meetings wherein all members can voice concerns and suggestions on how development occurs going forward. As for the technical management, every team member contributes, including the current scrum master. In short, everyone develops for the application, and one member doubles as a manager, with this role switching once every two weeks.

2.2. Responsibilities

Due to the size and decentralized organization of Team Squared, the roles that a member of Team Squared may be tasked with fulfilling during any given scrum period can largely vary and change depending on the immediate needs of the *Squared* application during that particular scrum period. **Section 2.2.1** outlines the most common roles that a member of Team Squared or a member of the *Squared* configuration management team may be tasked with playing during the course of a scrum period, while **Section 2.2.2** specifies which of these common roles each member of Team Squared and / or the *Squared* configuration management team have been tasked with fulfilling most often.

2.2.1. Development and Team Roles

The following descriptions outline common roles that members of the *Squared* development team and configuration management team may play within their teams during the course of a scrum period. These roles have been described and defined using the following aspects of each role:

- **Purpose and objectives:** The purpose of the specified role with regards to the continued development of the *Squared* application or the organization of Team Squared or the *Squared* configuration management team.
- **Period of effectivity:** How long a team member designated with the specified role will likely be expected to maintain the role for, or how long a team member fulfilling that role may effectively be expected to fulfill it for.
- **Scope of authority:** Which teams, sections of the *Squared* development team, or individuals responsible for the development or documentation of *Squared* anyone with the specified role will have administrative or development influence or authority over.
- **Operational procedures:** A brief description of the standard operating procedures that a team member designated with the specified role will be required to uphold and perform throughout the period of effectivity in which they are designated that role.

Some common roles that *Squared* development team and configuration team members may play during one or more scrum periods include:

1. **Front-End Designer**
   - ***a.*** **Purpose and objectives:** A front-end designer is tasked with creating user interfaces that are visually appealing, easy to navigate and understand, thematic to the appearance of *Squared*, and designed in accordance with player feedback and recommended web design practices. Specific objectives that a front-end designer may be tasked with include the pushing of HTML, CSS, and ReactJS-based interface documents to the *Squared* repository, and communicating with front-end and back-end developers to ensure that interfaces are designed with the functionality of *Squared* in mind.
   - ***b.*** **Period of effectivity:** Varies; typically assigned to specific team members for one or more scrum periods in a row, but the front-end designer role will likely be on-going as the layout and appearance of *Squared* further modernizes and is changed in accordance with user feedback over time.
   - ***c.*** **Scope of authority:** Team Squared development team members designated for front-end design and development projects, and *Squared* configuration management team.
   - ***d.*** **Operational procedures:** Standard operating procedures for front-end designers include conducting research on recommended web design styles and practices, working with front-end and back-end developers and components in designing user interfaces with desired functionalities, designing themes and style sheets for specific pages of the *Squared* website and the *Squared* website as a whole, and updating user interfaces that have been pushed to the *Squared* repository in accordance with user and developer feedback and suggestions.

2. **Front-End Developer**
   - ***a.*** **Purpose and objectives:** A front-end developer is tasked with implementing functionality to *Squared* user interfaces that allows those interfaces to change depending on user input or the state of the device that *Squared* is being played within. Specific objectives that a front-end developer may be tasked with include the pushing of Javascript and ReactJS-based algorithms and functions to the *Squared* repository, communicating prolifically with front-end designers to ensure that functionality is properly applied to *Squared* interfaces, and communicating with back-end developers to ensure that information sent by users through the *Squared* front-end can be properly transferred to the *Squared* back-end.
   - ***b.*** **Period of effectivity:** Varies; typically assigned to specific team members for one or more scrum periods in a row, but the front-end developer role will likely be on-going as the functionality of *Squared* is further improved and changed in accordance with user feedback over time.
   - ***c.*** **Scope of authority:** Team Squared development team members designated for front-end design and development projects, and *Squared* configuration management team.
   - ***d.*** **Operational procedures:** Standard operating procedures for front-end developers include conducting research on programming strategies and functions that may prove useful for specific Javascript and ReactJS front-end development tasks, working with front-end designers to implement Javascript and ReactJS-based functionality into HTML, CSS and ReactJS-based interfaces, working with back-end designers to ensure that data sent through user input can

sufficiently be used in databases, and continuously implementing new functionality to the game of *Squared* and the *Squared* website as a whole in accordance with user and developer feedback and suggestions.

3. **Back-End Developer**
    - ***a.*** **Purpose and objectives:** A back-end developer is tasked with implementing functionality to *Squared* as it pertains to the transferral of user input, data and
information to, from and between back-end *Squared* databases. Specific objectives that a back-end developer may be tasked with include the pushing of Python and MongoDB-based functions and algorithms designed for transferring, saving and fetching user-defined data between *Squared* databases, implementing back-end functions that processes data pertaining to active *Squared* games, and communicating with front-end developers and database administrators to ensure that proper, secure and efficient communications can be made between the front-end and back-end components of *Squared.*
    - ***b.*** **Period of effectivity:** Varies; typically assigned to specific team members for one or more scrum periods in a row, but the back-end developer role will likely be on-going as the databases and systems used for connecting the front- and back-end components of *Squared* evolve alongside the continued development of the *Squared* website over time.
    - ***c.*** **Scope of authority:** Team Squared development team members designated for back-end design and database administration projects, and *Squared* configuration management team.
    - ***d.*** **Operational procedures:** Standard operating procedures for back-end developers include conducting research on safe, secure and efficient Python and MongoDB scripting practices in accordance with the desired functionality of back-end *Squared* components, working with database administrators to design appropriate *Squared* databases and connect them with back-end processes, working with front-end developers and designers to ensure that secure and efficient communication can be maintained between the *Squared* front-end and back-end, and pushing frequent security and functionality updates to back-end and database-related queries and functions in accordance with user and developer concerns, feedback and suggestions.

4. **Database Administrator**
    - ***a.*** **Purpose and objectives:** A database administrator is responsible for the safe and secure construction, maintenance and integration of back-end databases to other back-end and front-end components of the *Squared* application. Specific objectives that a database administrator may be tasked with include the definition and implementation of MongoDB-based databases in accordance with desired back-end *Squared* functionality, working with back-end developers to ensure that back-end functions are created with respect to the format of *Squared* databases, working with front-end developers to ensure that front-end user input can be received by databases and that database information can be fetched out to other back-end and front-end components, and constantly maintaining *Squared* databases by improving safety measures and designing systems to prevent malicious actors from using data within a database that is to remain off-access to *Squared* front-end components.
    - ***b.*** **Period of effectivity:** Varies; typically assigned to specific team members for

one or more scrum periods in a row, but the database administrator role will likely be on-going as potential security, efficiency, and functionality concerns related to implemented databases are brought to the attention of the *Squared* development team over time.

c. **Scope of authority:** Team Squared development team members designated for back-end design and database administration projects, and *Squared* configuration management team.

d. **Operational procedures:** Standard operating procedures for database administrators include conducting research on safe and secure development practices for defining MongoDB-based databases and queries, maintaining constant communication between oneself and back-end developers to ensure that databases are implemented as safely, securely and efficiently as possible, and pushing frequent database security and performance improvement updates to the *Squared* repository in accordance with user and developer concerns, feedback and suggestions.

5. Server Administrator

a. **Purpose and objectives:** A server administrator is responsible for overseeing and contributing to the integration, security and development of the front-end ReactJS and back-end MongoDB and Python servers of *Squared* through the implementation of functions that go between both servers and permit integration between the two systems. Specific objectives that a server administrator may be tasked with include the implementation of Python and ReactJS functions that transfer data between front-end ReactJS components and back-end Python and MongoDB components, communicating with front- and back-end developers to ensure that functions are designed with other components of *Squared* in mind, and promptly addressing and resolving any issues pertaining to front-end or back-end security.

b. **Period of effectivity:** Varies; typically assigned to specific team members for one or more scrum periods in a row, but the server administrator role will likely be on-going as potential security, efficiency, and functionality concerns related to the front-end and back-end servers of *Squared* are brought to the attention of the *Squared* development team over time.

c. **Scope of authority:** Team Squared development team members designated for front-end design, back-end design and database administration projects, and *Squared* configuration management team.

d. **Operational procedures:** Standard operating procedures for server administrators include conducting research on safe and secure programming and scripting practices for both front-end MongoDB and Python-based back-end components and front-end server components, assurring and personally maintaining constant contact between front-end and back-end developers and designers to ensure that functionality implemented in both ends of *Squared* are done with complete respect to the rest of the components of *Squared*, and consistently addressing issues pertaining to, and subsequently pushing, security and performance updates to both front-end and back-end functionality in accordance with user and developer concerns, feedback and suggestions.

6. Team Scrum Master

a. **Purpose and objectives:** A team scrum master is responsible for overseeing the overall progress, development and documentation of the *Squared* project and contributions made to the *Squared* repository, as well as coordinating and leading a brief weekly scrum meeting for the purposes of maintaining communication between all members of Team Squared and allowing team members to share and be given feedback on their personal *Squared* contributions. Specific objectives that a server administrator may be tasked with include successfully and productively coordinating, planning and leading at least one scrum meeting throughout the course of a scrum period, communicating with all members of Team Squared to ensure that desired *Squared* documentation and *Squared* repository contributions are provided efficiently and in a timely manner, and maintaining an overall leadership position within Team Squared throughout the course of one to two scrum periods.

***b.* Period of effectivity:** One to two scrum periods.

***c.* Scope of authority:** All of Team Squared and the *Squared* configuration management team.

***d.* Operational procedures:** Standard operating procedures for team scrum masters include the coordination, planning and leading of at least one scrum meeting throughout the course of one to two scrum periods, maintaining and encouraging transparency and communication between all members of Team Squared, acknowledging user and developer feedback and concerns and directing the course of Team Squared with respect to that feedback, and playing an overall leadership role in the implementation of new *Squared* repository contributions and documentation throughout the course of one to two scrum periods.

2.2.2. Specification of Team Member Roles

The following statements, based on the roles outlined in section 2.2.1 of this plan, distinguish and describe the roles of the members of Team Squared based on the roles that each member has fulfilled or intends to fulfill the most frequently throughout the course of the development of *Squared.* Note again that the specifications provided below are not immutable, and can largely vary depending on the specific needs and demands required of *Squared* or the Squared Team during any given scrum period.

- **Keijaoh Campbell** is a front-end designer, front-end developer, back-end developer, database administrator, server administrator and former team scrum master.
- **Jeffrey Fosgate** is a front-end designer and front-end developer.
- **Shea Keegan** is a back-end developer and database administrator.
- **Jacob Lorenzo** is a front-end developer, back-end developer, database administrator and current team scrum master.
- **Tyler Walker** is a front-end developer, back-end developer, database administrator, server administrator and former team scrum master.

2.3. Applicable Policies, Directives, and Procedures

As a multi-national game, we abide by the General Data Protection Regulation (GDPR) for our European users, as well as Children's Online Privacy Protection Rule (COPPA) for our users. This guideline is covered in our privacy policy which all users agree to. These laws essentially say what we can and can't do with certain demographics of our user base.

3. SCM Activities

3.1. Configuration Identification

3.1.1. Name Configuration Items

3.1.1.1. ReactJS page names should be written in lowercase and have the extension .jsx. For example, “**login.jsx”** indicates that the page contains JSX source code.

3.1.1.2. All Javascript classes should have the .js extension at the end of the lowercase name. For example, “**board.js”.**

3.1.1.3. In the website and API project’s folder, names should be lowercase, for example, **constants**.

3.1.1.4. At the top of each file, an author's name should be present. Also, the version and what task was completed in the project's current iteration. This information should be in the form of a multi-level comment.

3.1.1.4.1. The comments header should be the class name with an underscore version number. For example, “Game_v1.2” identifies the file as Game file version 1.2.

3.2.1. Acquiring Configuration Items requires all of the past system requirements we wrote along with the user stories. They may also be chosen from discussion in scrum meetings and added as a story. 3.3.1. All CI’s must be easily referenced by their ID and name. If needed the description can also be needed.

3.2. Configuration Control

3.2.1. Requesting changes would usually be just a discussion in the discord or during our scrum meeting. As a group we will work together to confirm if this is a change we would like to see or not. **3.2.2**. Evaluating these changes will be discussed in the same way, but probably more often in the scrum meeting.

3.2.3. Implementing the change will happen by assigning the task to someone and then discussing when it will be pushed into the github. when it is, we will record it in our backlogs.

Configuration Status Accounting

- *Metrics to be tracked and reported and type of report.*
- *Storage and access control of status data.*

The commits will be tracked automatically by the github and manually tracked through our backlogs. You should track the amount of work you did on a certain CI and record it in the backlog when you update it.

3.3. Configuration Evaluation and Reviews

- *At minimum an audit on a CI prior to its release.*
- *Defines objective, schedule, procedures, participants, approval criteria etc.*

Before a CI can be released it must be tested, tweaked, and approved. New versions can’t be released if the code is incomplete and will hinder the development of people that pull from that version. Any problems interacting with other parts should be discussed with the person responsible for developing that portion.

3.4. Interface Control

- *Coordination of changes to CIs with changes to interfacing items outside of the scope of the Plan.*

The front end designers will make changes to the interface and should be spoken about between the people working on it.

3.5. Subcontractor/Vendor Control

● *Incorporation of items developed outside the project environment into the project CIs.* Anything developed outside of the environment will go through the same testing process as the rest. The specifics may be chosen by the person developing the portion. **3.6. Release Management and Delivery**

● *Description of the formal control of build, release and delivery of software products.* Minor releases will be the releases that occur throughout the sprint weeks and are relatively small. Major releases will be ones that include a lot of new additions and will usually be released at the end of a sprint.

4. SCM Schedules

4.1. Sequence and coordination of SCM activities

Our sequence of activities for developing our application is regularly updated in our sprint backlog. This presents which features are either complete or in progress and which users were responsible for their creation.

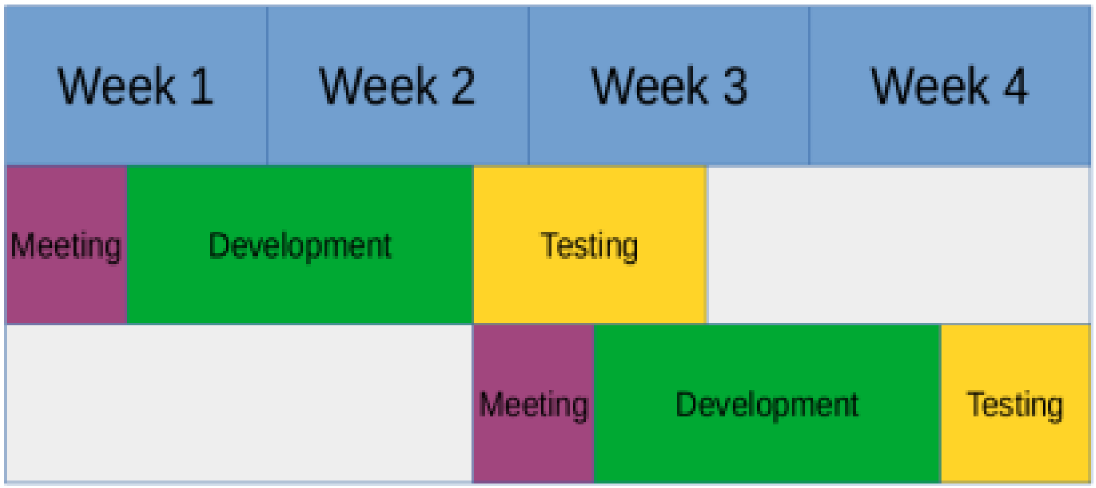
4.2. Relationship of key SCM activities to project milestones or events:

In terms of agile, we're deciding our next milestone (the next version of our project) and then breaking it down into smaller events that need to be done prior. Essentially, by the end of our sprint, these small goals will converge into one major milestone. This sequence continues throughout development until our entire application and user stories are completed.

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

Every two weeks, we, as a team, hold a meeting to create a vision of where we see our project in the coming two weeks. The change can range from simple UI changes to feature implementation. These are usually not absolute dates, as some features require more work than initially foreseen. However, this doesn't necessarily correlate with live updates, as many features are minor changes to the codebase that aren't immediately visible to the users. Rather, live updates will be released to the users on a monthly basis instead.

4.4. Graphical representations can be used here.



In the above, meetings consist of planning the next features in development. Following that, the features are developed in the slot labeled “Development”. After development, these new features are tested to ensure they don’t exhibit unintentional issues. This testing is done concurrently with the first portion of week 3 and 4’s development. This two week development cycle continues until the application is done.

5. SCM Resources

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

- React
- Flask
- Github

5.2. Key factors for infrastructure:

- *Functionality, performance, safety, security, availability, space requirements, equipment, costs, and time constraints.*

5.3. Identify which tools are used in which activity.

5.4. Version Control (Git and GitHub)

5.4.1. This is handled by using Git in each developer's local machine environment. However, changes are pushed to Github, which actions as the HUB for all version control for the team. It facilitates rollbacks in case of issues and branching when key changes by the team need to be tested prior to merging into the main branch.

5.5. Concurrency Management

5.5.1. The framework has been facilitated through rules established by the team that handles when two or more team members are working in the same file. In this situation, the following is done.

5.5.1.1. The developers are assigned the task inside the Kanban board, which has tags to indicate the application work area.

5.5.1.2. Before a task is started, each developer reviews their task inside the Kanban and will be able to see other related tasks to establish if their task overlaps with another. Furthermore, in scrum meetings, tasks are assigned, which makes all members aware of what will be worked on and what task is dependent on the other; to reduce duplication of efforts.

5.5.1.3. Each developer containerizes their source code so that although working in the same file, functionalities. It can be defined into separate functions within the class and inherited from super-classes that are common to both and keeping the functionalities componentized so that they won't clash with each other tasks.

6. SCM Plan Maintenance

6.1. Who is responsible for monitoring the plan?

It is the product owner's responsibility to monitor the plan, and it’s the responsibility of all team members to maintain the plan.

6.2. How frequently updates are to be performed?

There is currently no definitive update timeline. The team shall revise the plan before any major release. Any developer should make changes proactively if they think a change is necessary.

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

Changes to the plan shall be evaluated and approved by the product owner.

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

Changes shall be proposed and performed as stated in section 3.2.

6.5. Also includes history of changes made to the plan.

Name	Date Change
	4/3/22 Creation of the CMP
Keijaoh Campbell	4/3/22 Addition of Scope, Purpose, Configuration Identification, Tools Used, grammar edits on various parts of the document.
Jeffrey Fosgate	4/3/22 Tabulation of plan descriptions and acronyms; addition of Responsibilities.
Jacob Lorenzo	4/16/22 Updated document based on feedback given

Jeffrey Fosgate	4/17/22 Removal of empty “Resources” section