[Table of Content]

1. Introduction

1.1. Purpose

This configuration management plan covers only the high level administrative tasks for approving, recording, verifying, and releasing TravelGavel changes. This plan assigns and describes all software configuration management details and responsibilities to developers for maintenance and travelers for use as they are the main audience for TravelGavel.

1.2. Scope

Updates and management for this app requires Android Studio in Java. When configuring the project, the minimum SDK level is currently Android 9. Management of updates, revisions, and additions to this app has currently been managed in GitHub. Developers should have a GitHub account so that they can work with the most up to date version of the code.

1.3. Key Terms

1.3.1. Configuration Management Terminology

1.3.1.1. baseline - (1) A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures. (2) A configuration identification document or a set of such documents formally designated and fixed at a specific time during a configuration item's lifecycle.

1.3.1.2. configuration - (1) The arrangement of a computer system or component as defined by the number, nature, and interconnections of its constituent parts. (2) In configuration management, the functional and physical characteristics of hardware or software as set forth in technical documentation or achieved in a product.

1.3.1.3. configuration item (CI) - An aggregation of hardware, software or both, that is designated for configuration management and treated as a single entity in the configuration management process

1.3.1.4. configuration management (CM) - A discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements.

1.3.1.5. graphical user interface (GUI) - a visual way of interacting with a computer using items such as windows, icons, and menus, used by most modern operating systems

1.3.1.6. pull request - a request to a target repository to pick up their local changes and merge them into the existing code

1.3.1.7. software development kit (SDK) - a set of tools for third-party developers to use in producing applications using a particular framework or platform

1.3.2. Project-related Terminology

1.3.2.1. user - an individual or group that uses the system for its intended use when deployed in its environment

1.3.3. Abbreviations

1.3.3.1. CI - configuration item
1.3.3.2. CM - configuration management
1.3.3.3. GUI - graphical user interface
1.3.3.4. SDK - software development kit

1.4. References

1.4.1. IEEE Std 610.12-1990, “IEEE Standard Glossary of Software Engineering Terminology”
1.4.2. NASA “Configuration Management Plan Template”

2. SCM Management

2.1. Organization

2.1.1. Development Organization

This section outlines the basic organizational structure for all groups working on future development of the product. Each development session has a project manager, and at least two additional developers working on each baseline iteration of the product.

2.1.2. Project Organization

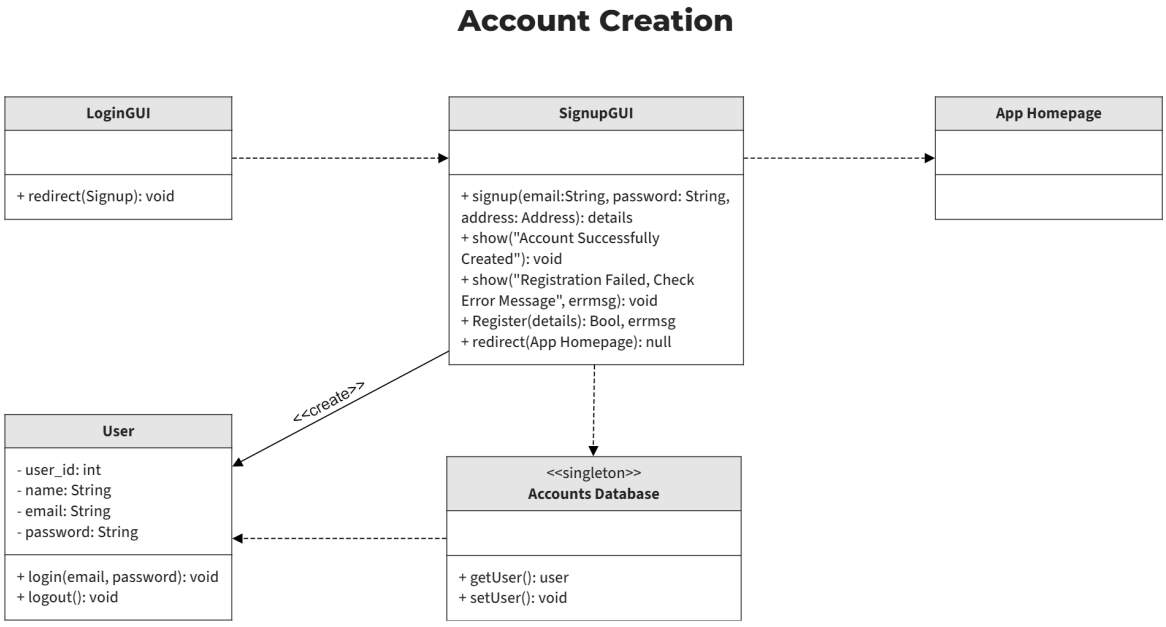

Figure 1 Account Creation Organization

Figure 1 demonstrates the organization of the account creation system within the project. This process is executed through the SignupGUI class' dependent relationship with the LoginGUI class, as SignupGUI would be inaccessible without this relationship. The SignupGUI class, once provided with name, email, and password information from the currently-anonymous user, establishes a dependent relation with the Accounts Database, which can furthermore create a new User instantiation only after having been passed the inputted information. Therefore, a special relationship directly between the SignupGUI class and the User class is established to signify that SignupGUI initializes the creation of a new User account, utilizing its dependency on Accounts Database in the process. Finally, a relationship exists where the App Homepage user interface is dependent on the SignupGUI class, as no transition between Account Creation and the App Homepage would occur without the redirect() method from SignupGUI. Trivial relationships not displayed within this diagram only include messages displayed to the user dependent on the state of the Boolean variable returned from the SignupGUI's Register() method.

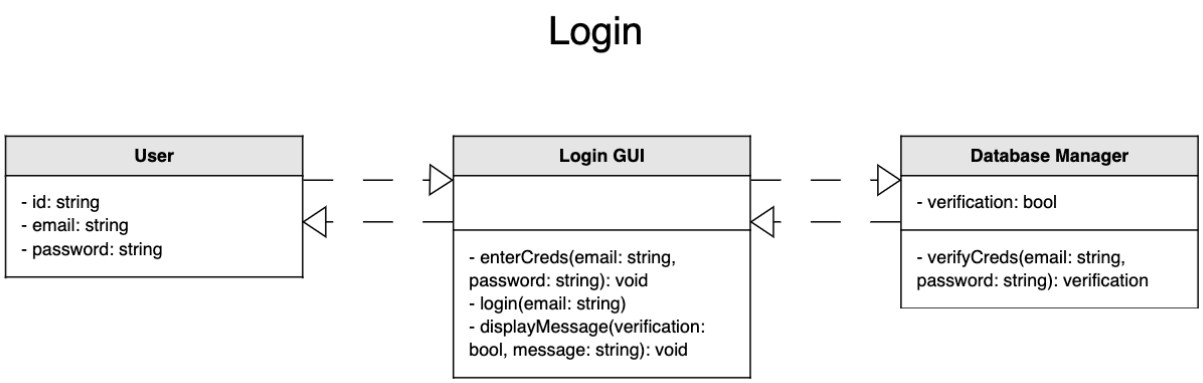

Figure 2 Login Organization

Figure 2 demonstrates the organization of the login system for the project. The User must be able to send information to the Login GUI, so that a user can enter an email and password to verify their credentials. The Login GUI is connected to the Database Manager, so that it can request the user's stored credentials. The Database Manager is connected to the Login GUI, so that it can be sent the user's stored credentials. The Login GUI checks the entered information from the User against the user's stored credentials from the Database Manager. Depending on the result, the user will either be logged in and receive a success message or they will receive a failure message.

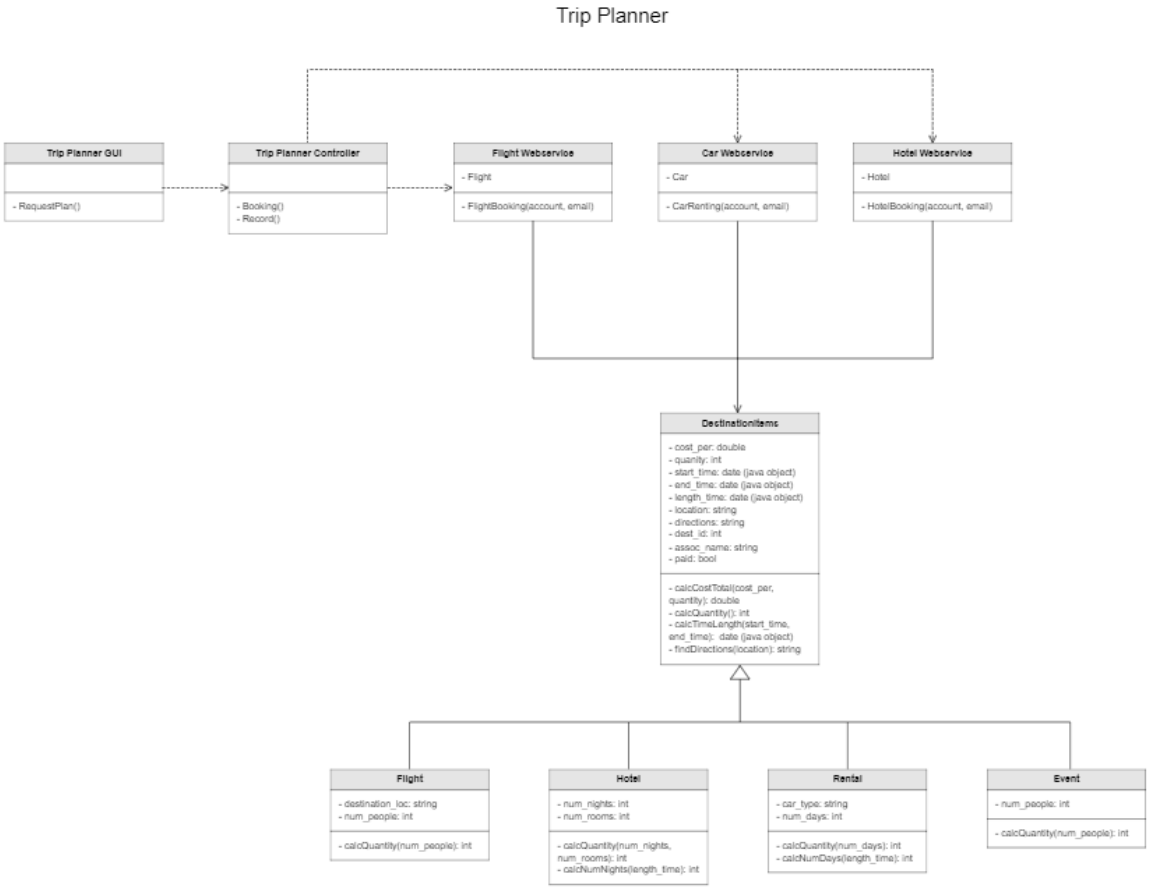

Figure 3 Organization of a Trip

Figure 3 illustrates the inner workings of planning a trip and its components. For this process the User requests to plan a trip through the Trip Planner GUI. The Trip Planner GUI is connected to the Trip Planner Controller, which connects to the Flight Webservice, Car Webservice, and Hotel Webservice to book flights, cars, and hotels per the User's request and record those bookings. The webservice classes connect to the Destination class, which in turn connects to the appropriate subclass, depending on the webservice class that connected to it, and retrieves the appropriate information and attributes for the intended object. This information is then sent back along the proper pathways to be recorded in the trip. Depending on the nature of the request, the user will be able to book and record a hotel, flight, or rental for their vacation.

2.2. Responsibilities

Roles	*Purpose and Objectives*	*Membership and Affiliations*	*Period of Effectivity*	*Scope of Authority*	*Operational Procedures*

Project Manager	*Manage workflow of new code or features* *Review current version of app and plan future improvements or changes*	*•Management* *•Development* *•Q&A*	*Duration of the development and Q&A of the newest version or patch of the product*	*All aspects of development of the current version of the product*	*Plan workflow, tasks, and new features to implement* *Note any outstanding issues, flaws, or changes that need to be made to ensure functionality of the product* *Collaboration on use case, analysis, and domain models*
Developer	*Implement new features or fixes into the product* *Review the changes of others to ensure the product functions properly with no new issues*	*•Development* *•Q&A*	*Duration of the development and Q&A of the newest version or patch of the product*	*Development of assigned feature for implementation and assigned implementation of others to review*	*Collaboration on use case, analysis, and domain models* *Implement needed changes in the product* *Review updates to the product to maintain its quality* *Quality testing (use case and unit) to ensure the functionality of the product*

2.3. Applicable Policies, Directives, and Procedures

2.3.1. Maine Privacy Law (LD 946) details the requirements of Internet service providers to obtain consent before being allowed to use, disclose or provide access to customer information.

3. SCM Activities (Ideally you will have subsections 3.1.1, 3.1.2, …)

3.1. Configuration Identification

Seeing as this project has been developed using GitHub, this project utilizes a distributed version control system. For baselines, we use a 3-digit naming scheme of 1.2.3a. 1 - indicates the major, external release version. 2 - indicates the minor, internal release version. 3 - is the small, internal revision, and a - is the patch version for the major release.

Baseline Version History:

- 0.3.0 - implementation off the search feature
- 0.2.0 - implementation of the account creation feature
- 0.1.0 - implementation of the login feature

Implementation of each baseline takes place on individual developer's devices through various codelines. Each codeline is then pushed to the repository each day and is reviewed and approved by at least one other developer prior to commitment to the main branch of the code.

3.2. Configuration Control

3.2.1. Requesting Changes

To request a change, a change must first be submitted by creating a pull request in GitHub with their proposed changes as well as an explanation or description of the purpose and effect of the proposed changes.

3.2.2. Evaluating Changes

At least one other developer needs to review the pull request changes to ensure that it will not cause any damage to the primary baseline or current version of the product.

3.2.3. Approving, Disapproving, and Implementing Changes

If the proposed changes in the pull request are found to not cause any conflicts or issues, then the reviewer may approve the pull request through GitHub and merge the codeline connected via the pull request to the baseline. If the proposed changes do create an issue or conflict, the reviewer may request changes to the pull request with comments as to what needs to change and why before the codeline can be approved and merged with the baseline.

3.3. Configuration Status Accounting

For each file changed or created in the repository, the following data is documented: who made the change, the date the change was made, the reason for the change, and who approved the change. There is also the README file which should contain a version identifier of the current version release and the primary contributions or changes for the version.

3.4. Configuration Evaluation and Reviews

This plan defines a formal review that takes place as a final check on the software CIs. This begins with a code review as part of the process for accepting developer changes. It verifies

that the changes in the code do not propose significant security or functionality risks to the current version of the product. During this process, approval or acceptance of the current baseline can be delayed if the baseline doesn't properly implement a functional requirement, if proposed changes in the baseline are found to compromise the security or functionality of the product, or if a developer has made a last minute requirement change that must be in place before the next product release. The code should be reviewed by two or more developers and receive final approval from the project manager.

3.5. Interface Control

This section discusses the other organizations that the project development team interfaces with both on a software configuration management level and a design level. If the project configuration management plan adds an organization to section 2.1, it will use this section to describe the interfaces with these organizations and how they are to be managed. In this particular instance, each development cycle of the product is managed by a project manager with at least two other developers working on the team.

Project software and baseline contributions are built using Android Studio in Java, unless specified or warranted to be done otherwise. Each developer is assigned either individually or in pairs to work on a primary feature to implement. Codelines are pushed to the GitHub repository each day and reviewed by at least one other developer to ensure that the changes do not cause security or functionality issues with the current version of the code. Prior to approval for merging to the baseline, at least two developers must review the code for any security or functionality conflicts that could arise from the new changes. THe project manager must give final approval to the changes for the code to be merged with the baseline.

3.6. Subcontractor/Vendor Control

This section does not apply to versions of the product up to and including version 0.3.0. However, should this project receive source code or executable programs from a third party or subcontractor, the control the current development team and project manager has over the changes to the supplied software will be specified here.

3.7. Release Management and Delivery

This plan defines a specific method for integrating changes from different branches of the software. Individual components are submitted to the version control repository each day and reviewed by at least one other developer. Changes are given a description for the branch and reviewed by at least two developers for any functional or security conflicts. If the changes are approved by the developers and project manager they are merged to the main branch for release.

4. SCM Schedules

4.1. Sequence and coordination of SCM activities

Implementation of a development cycle will not begin until the project manager has approved it.

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

The minimum events in the development process are as follows:

- Select the primary functional requirements, features, and security patches that are needed for this major release
- Develop use case models, analysis diagrams, and domain models for implementation of the new requirements
- Assign a functional requirement or primary feature to each developer or pair of developers for implementation
- Codeline changes are pushed at the end of each day and reviewed by at least one other developer for any potential issues caused by the changes
- This development process repeats until all required changes have been implemented
- Code review begins by at least two developers with final approval from the project manager
- The code is merged to the new baseline after final approval

Any functional requirements or new features should be completed and formally requested prior to the start of the design process. The only exception to this are security patches needed due to urgent security concerns from the current version of the product.

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

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
1	2 · Project manager approves new development cycle	3 · begin selection process for feature, functional requirements, and security patches	4 · Finalize selection for features and functional requirements	5 · Begin development of use case models, analysis diagrams, and domain models for selected features and requirements	6	7
8	9 · Finalize models for selected	10 · begin assigning features and requirements to each	11	12	13 · Testing and review of implemented	14

	features and requirements	developer (or pair) for implementation			requirements and features	
15	16 · Assign next set of features and requirements to each developer (or pair) for implementation	17	18	19	20 · testing and review of implemented requirements and features	21
22	23 · Code freeze	24 · Begin review of code changes by at least two developers	25	26	27 · Project manager reviews code for any conflicts	28
29	30	31 · Project manager finalizes code for approval and code is merged to the baseline				

Table 2 General CM Activities for a One Month Cycle

5. SCM Resources

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

At the time of development for version 0.3.0, Android Studio has been selected for software development, and GitHub has been selected as a means of storing the repository and for version control management. For these software requirements, the following hardware

requirements are needed: a computer running a 64-bit version of Windows (8, 10, or 11), Linux, macOS (10.14 Mojave or later), or ChromeOS.

The project manager must have experience working with GitHub and version control. Ideally, the other developers have experience with Android Studio and Java or the team has at least one developer with this experience ot guide, monitor, or train others.

5.2. Key factors for infrastructure:

To ensure efficient, secure, and timely development, any functional requirements or new features should be completed and formally requested prior to the start of the design process. The only exception to this are security patches needed due to urgent security concerns from the current version of the product. Additionally, no pull request should be approved without being reviewed by another developer for any conflicts (security or functional) with the current version of the product. Finally, prior to official release, the code should be thoroughly tested and reviewed by at least two developers with final approval from the project manager to ensure there are no significant security or functionality concerns with the new changes to the baseline.

5.3. Identify which tools are used in which activity.

At the time of development for version 0.3.0, Android Studio in Java has been selected for software development, and GitHub has been selected as a means of storing the repository and for version control management. For these software requirements, the following hardware requirements are needed: a computer running a 64-bit version of Windows (8, 10, or 11), Linux, macOS (10.14 Mojave or later), or ChromeOS.

6. SCM Plan Maintenance

6.1. Who is responsible for monitoring the plan?

The plan is to be monitored by the project manager to ensure that the development team is functioning on schedule. All members of the development team have a general responsibility for ensuring that they are meeting all software, hardware, security, and best practice requirements, and for checking that others are doing the same.

6.2. How frequently updates are to be performed?

Updates to the product are to be performed at least once every three months to keep up with changing security requirements and concerns.

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

Changes to the configuration management plan should happen within one week after each three month development cycle to ensure the guide and information is up to date. These changes should be proposed by the members of the development team for this product release version. They should be approved by at least two other developers or project managers outside of the development team for that release.

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

A proposed change to the plan should include the following: the proposed change, who proposed the change, a clear description of the change and what it would include, an explanation for why the change is needed, and an explanation as to how the proposed change will resolve the issue. This is brought to the attention of al lof the members of the development team of the most recent version release. This team reviews the proposed change and its components and may request and changes or further explanation. If a majority of the team members of the development team approve of the change, it goes to two external developers or project managers for final review and approval.

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

A history of changes made to the plan can be found in different versions of this document, named to reflect the different version release in which the document was updated (i.e. Configuration Management Plan 0.3.0 is the plan that has been modified to include changes made as a result of the development of version 0.3.0). The changes made to the document are noted at the bottom of the most recent document version along with who proposed the changes, who approved them, and why they were implemented.