Group 3

COS 420

April 14,2019

**Configuration Management Plan Document**

**Introduction**

Overview description of software project.

* Our software project is a dating app. However, it differs itself from other dating apps by not letting users use pictures of each other in order to “like” or “dislike.” Instead, users are encouraged to like someone based off of their customizable profiles and our ice breaker system. Our app will still have features that other dating apps have, like blocking others and setting an age range, but will focus more on your personality rather than your looks. Profiles can either be made to be unique and completely customized, or users can choose to fill out a basic profile. In place of profile pictures, users can either choose to display:
  + Their answer to a chosen ice breaker type question. These can be fun get-to-know you questions, as a way to introduce yourself in a fun way. These can be things like “would you rather?”, “two-truths and a lie.”
  + Their created user profile, which lists things including likes, dislikes

Identification of the software configuration items (CIs) to which SCM will be Applied.

* There are a variety of software configuration items to which our software configuration management plan will be applied to. These include documents like our Software Requirement Specification and Software Architecture documents, and models like our Use Case and Sequence Diagrams. Other software configuration items include the software itself, which is constantly being updated on our GitHub repository.

Identification of other software to be included as part of the plan (e.g., support or test software).

* Support software that is included as part of the plan, include React Native with Expo to help us develop the app’s front end easily, and Flask as our backend. At this time, there is no test software as part of our plan.

Relationship of SCM to the hardware or system configuration management activities for the project.

* In order to keep track of our different project versions, different software components, we are using the common distributed system, Git. This ensures that changes made by our developers to these varying versions do not interfere with each other. This also allows us to manage our codelines and baselines.
* Our project uses Expo in order to help make our app cross platform using JavaScript. With our one codebase, we are able to test our app on multiple platforms. Using Expo and React Native allows use to take advantage of powerful developer tools like Remote Debugging, and publish project updates instantly.
* As of right now, our change management system is just using our Zenhub board in order to keep track of things in progress, things that need to be worked on, and bugs that need to be fixed.
* As of right now, we don’t have release management plan, but we will update this document when one is settled on.

The degree of formality, depth of control, and portion of the software life cycle for applying SCM on this project.

* Our software configuration management plan will be applied to most of the project. All of our documentation and software configuration items are updated and edited based on feedback every deliverable. Similarly, the code part of our project updated and improved every deliverable as well. Both the backend and frontend are changed during every sprint, and these changes are can be seen on our project’s GitHub repository. Whenever documentation is completed, it is added to the project repository as well.

Limitations, such as time constraints, that apply to the plan.

* We will not be able to fully implement all of the features that we wish to add to the app because of time constraints. This project is to be completed within the time given for a software engineering course so we do not have time to create all of the features that we want. We will start by creating only the features that are absolutely necessary for the project, then work on those that we would like, but not necessarily need.

Assumptions that might have an impact on the cost, schedule, or ability to

perform defined SCM activities (e.g., assumptions of the degree of customer

participation in SCM activities or the availability of automated aids).

* For this project, we are assuming that due date of everything is going to be on May 5th. Additionally, we assume that we aren’t going to run into anymore problems with development. This means that the app’s main features (building profile, basic matching with other users, chatting) will be completed within the given time frame.

**SCM Management (Who?)-**

* + 2.1 Organization
    - We are implementing the configuration management activities within the context of a University of Maine Software Engineering course. The use of Git not only helps the contributors of the project, but helps the professor to determine the work done by each student.
  + 2.2. Responsibilities
    - Noah Monto, University of Maine Student
      * Upload any changes made to the software onto GitHub.
      * Make changes to the software in any way they see fit.
      * Update ZenHub board to notify team members of what is being worked on and what has been completed.
    - Kody Moseley, University of Maine Student
      * Upload any changes made to the software onto GitHub.
      * Make changes to the software in any way they see fit.
      * Update ZenHub board to notify team members of what is being worked on and what has been completed.
    - Enoch Lin, University of Maine Student
      * Upload any changes made to the software onto GitHub.
      * Make changes to the software in any way they see fit.
      * Update ZenHub board to notify team members of what is being worked on and what has been completed.
    - Matthew Loewen, University of Maine Student
      * Upload any changes made to the software onto GitHub.
      * Make changes to the software in any way they see fit.
      * Update ZenHub board to notify team members of what is being worked on and what has been completed.
    - Andrew Piccirillo, University of Maine Student
      * Upload any changes made to the software onto GitHub.
      * Make changes to the software in any way they see fit.
      * Update ZenHub board to notify team members of what is being worked on and what has been completed.
    - Taidgh Robinson, University of Maine Student
      * Upload any changes made to the software onto GitHub.
      * Make changes to the software in any way they see fit.
      * Update ZenHub board to notify team members of what is being worked on and what has been completed.
    - Sepideh Ghanavati, Software Engineering Professor
      * View changes made to the software using software configuration management activities.
    - Sanonda Gupta, Software Engineering TA
      * View changes made to the software using software configuration management activities.
* 2.3. Applicable policies, directives and procedures
  + All changes must be uploaded to GitHub by the person who made them. Also the ZenHub must be updated to track the progress of the software development process by the appropriate person. This same policy applies to all of the documents created during the duration of this project. We also have a time constraint that we must adhere to in completing all of the documents.

**SCM Activities(What?)**–

* + 3.1 Configuration identification
    - The configuration items include documents that we have created. This includes: Project Description, User Story Document, Product Backlogs, Sprint Backlogs, Use Case Model, Sprint Review, SRS, Team Member Reports, Focus Group, Sequence Diagram, Domain Model, Architecture Design, Configuration Management Plan, Test Plan, Usability Study, and Remaining Tasks. Our configuration items also include the software itself which is updated and uploaded to GitHub by the team member that worked on it. All of these configuration items can be found in our public GitHub repository. There is a separate folder for the documents that we needed to construct for this project, and other folders that contain the code for this app.
  + 3.2. Configuration control
    - Requesting changes can be done by creating a public issue on our GitHub repository. These changes will be seen by all members of the team and then someone can get assigned to work on the change. The changes are evaluated through the use of documentation. All changes will be reflected by updating documentation detailing the specifics of the change. Approval and disapproval of changes is decided on by the team as a whole. We will figure out what we as a whole believe to be an appropriate change. The change will be implemented by the team members that get assigned to work on it, and will be uploaded to GitHub to merge with our previous code and fully implement the change.
* 3.3. Configuration status accounting:
  + Metrics to be tracked include the number of bugs that are known to the team as well as features that the team wants to implement, but hasn’t yet. These metrics can be tracked in our GitHub repository by looking at the issues section. The issues may include bugs, or a feature that a team member wants to implement in the future. We can determine how complete of an app we have by looking at the number of issues in our GitHub repository.
* 3.4. Configuration evaluation and reviews:
  + For this I chose to do an audit on the Detailed Design Model and Design Pattern Document, which will be turned in on 4/14 for Deliverable 3. For this document, we must include UML class diagrams for detailed design. Our UML classes have their attributes, and methods that represent the system under development, our Sprout dating app. This are the exact things that are required from this document. Our grades for the project reflect the fact that all of our CIs so far have been approved, as they meet the criteria. This criteria is basically making sure that the documents match the requirements as listed on the Project Details document on BlackBoard. The participants of all of these CIs are the Sprout Inc members listed above.
* 3.5. Interface control:
  + Any changes to the CIs are made after getting feedback from each deliverable. These changes are then reflected to the application itself. Then, the following deliverable includes updated CIs based on feedback from the previous deliverable. Deliverable four will include all of finalized CIs and software.
* 3.6. Subcontractor/vendor control:
  + Since we aren’t using any subcontractor or vendor control items, there is no need to detail the incorporation of items developed outside of the project environment.
* 3.7. Release Management and Delivery:
  + Our project is a free and open source application. Anyone can help develop it by going to our GitHub repository. The members of the group have been making incremental progress on the application every Sprint. The goals for each Sprint, as well of the progress on tasks needed to complete the Sprint, can be found on our ZenHub board. The progress of the application is split up into deliverables that we turn in over time. Each deliverable includes multiple CIs as well as one to two Sprint cycles. The final product will be turned in on May 5th, 2019. During this final deliverable, all CIs will be completed, as well as all progress on the application.
* **SCM Schedule (When?)**
* Establishes required coordination of SCM activities with other activities in the project.
* Sequence and coordination of SCM activities.
  + Everybody on the team is expected to take part in the coordination of SCM activities. To start, the sequence of SCM activities must be completed in a certain order, with configuration identification being first. This activity is necessary to complete before any other activity can be completed. Identifying CIs themselves, naming them, and acquiring them has to be completed before the next activity, configuration control. While we are requesting changes, evaluating changes, and implementing changes, we will be completing the next activity, configuration status accounting, in parallel. Configuration evaluation and reviews is the next activity, and this is important for all project CIs, as we need to figure out what is being asked for in each CI. Since we don’t really have any subcontractor/vendor with the project, the step of incorporating their items into our CIs is not necessary. Finally, the last activity is release management and delivery. This lines up with our sprints and deliverable schedules, with the entire project due at the end of the semester.
* Relationship of key SCM activities to project milestones or events, such as:

|  |  |  |
| --- | --- | --- |
| **Date** | **Activity/Milestone** | **Notes** |
| 2/6 | Deliverable 0 | * CIs included:   + Project Description   + User Story Document |
| 2/26 | Deliverable 1 | Sprint 1   * CIs Included:   + Product Backlog   + Sprint Backlog   + User Story   + Use Case Model   + Sprint Review   + Project Description   + SRS   + Team Member Report |
| 3/18 | Deliverable 2 | * Sprint 2 and 3 * CIs Included:   + Project Description   + User Story   + Product Backlog   + Sprint Backlog   + Team Member Report   + SRS   + Use Case Model and Description   + Focus Group   + Sequence Diagram   + Domain Model   + Architecture Design   + Sprint Review |
| 3/20 | Configuration Identification |  |
| 4/14 | Deliverable 3 | * Sprint 4 and 5 * CIs Included:   + Product Backlog   + Sprint Backlog   + Team Member Report   + SRS   + Use Case Model and Description   + Focus Group   + Sequence Diagram   + Domain Model   + Architecture Design   + Sprint Review   + Design Design Document   + Configuration Management Plan Document   + Test Plan Document |
| 4/14 | Configuration Identification | * Identify Configuration Items * Name Configuration Items * Acquiring Configuration Items |
| 4/14 | Configuration Control | * Requesting Changes * Evaluating Changes * Approving or Disapproving changes * Implementing Changes |
| 4/18 | Configuration Status Accounting | * Metrics to be tracked and reported and type of report * Storage and access control of status data |
| 4/22 | Configuration Evaluation and Review | * At minimum an audit on a CI prior to its release * Defines objective, schedule, procedures, participants, approval criteria etc. |
| 4/26 | Interface Control | * Coordination of changes to CIs with changes to interfacing items outside of the scope of the plan |
| 5/1 | Release Management and Delivery | * Description of the formal control of build, release and delivery of software products |
| 5/5 | Deliverable 4 | * Sprint 6: Final Project Turned in * CIs: Included   + Product Backlog   + Sprint Backlog   + Team Member Report   + SRS   + Use Case Model and Description   + Focus Group   + Sequence Diagram   + Domain Model   + Architecture Design   + Sprint Review   + Design Design Document   + Configuration Management Plan Document   + Test Plan Document   + Usability Study Documents   + Remaining Tasks Document |

**SCM Resources (How?) –**

* The plan requires tools such as Git and GitHub to successfully execute our plan. All team members must allot enough time to complete the tasks that were given to them during the sprint. If the tasks are not completed in a timely manner we may fall behind in the development of the app. The app must also perform to adequate performance, safety, and security guidelines. These guidelines are laid out in our documents and will be updated along the way if anything changes. All team members will report what has been done and what still needs to be done at the end of every sprint. This will be documented and updated on GitHub to reflect the current state of the app.
* **SCM Plan Maintenance –**
* All the members of the team will update the document as any changes in the plan are made. Updates will be performed every sprint and will be finalized by the May 5th deadline. Changes will need to be approved by all members of the group as to not have any conflict. This approval will be communicated to every member of the group through a group message.