CSE 201 Configuration Management Plan Document

## Team Member Names

**Team 9:**

Feiyu Wang, Nicole Matthews, Josh Overbeck, Jacob Sandy

## Team Project Title

**Abacus**

### Project Description

Abacus is a JavaFx desktop application. It is designed to help clothing stores log their inventory. Security and integrity is maintained by limiting read and write access of users. Managers will have read and write rights. And they will also be able to manage the accounts of other employees. Employees will only have read and write rights. Guests will have read rights only. At this time, the inventory will be stored locally on a single machine. Abacus will be wrapped in a easy to use graphical user interface that is event driven.

## Team Responsibilities

|  |  |  |
| --- | --- | --- |
| Member Name | Git User ID | Responsibility |
| Feiyu Wang | Feiyuwang1998 | Testing, both front-end and back-end |
| Nicole Matthews | matthenm | Software architecture |
| Josh Overbeck | overbejt | Developing front-end and back-end |
| Jacob Sandy | jacobsandy | ? |

## Project Repository

This project will be stored remotely on Github. It can be found at <https://github.com/overbejt/Group9.git>

## Branching Strategy

The branching strategy for the remote repository will be the version branching strategy. We will only create a new branch on the remote repository when we move on to the next version. On the local repository, any branching strategy can be used. So long as the new additions are merged with the remote branch and pushed to the remote repository.

## Version Release Plan

What release plan do you have? Are you making one major release per iteration? If so use the table below to record it and also include more rows if you are planning more releases.

|  |  |  |
| --- | --- | --- |
| Version Number or Label | Features /Bug fixes Included in version | Remarks |
| Version 1.x.x | Features will include a full screen desktop application. It will allow visitors to log in and view the inventory, but not make any changes to the inventory. Employees will be able to login to view the inventory and make changes to it. Managers will be able to login to view the inventory, make changes to the inventory, and create and delete employees. Users with write access will be able to add items to the inventory. They will be able to specify attributes about the items. They will be able to specify the quantity of the item in the inventory. They will also be able to delete items from the inventory. | This will serve as the baseline and the working prototype. The 1 in the version will specify the which version we are on. The first “x” will specify which iteration we are on. The third “x” will specify which bug fix has been worked on. |
| Version 2.x.x | Users will be able to sort the inventory based on the type. Users will be able to sort the inventory based on the date added. Users will be able to sort the inventory based on whether the item has “male” or “female” attributes. | This will be the baseline for the low priority user stories. The 2 in the version will specify the which version we are on. The first “x” will specify which iteration we are on. The third “x” will specify which bug fix has been worked on. |
| Version 3.x.x | The inventory will be stored in a database. The system will be deployed as a website. It will have the same functionality as the desktop application. But it will be wrapped as a .war project. | This will be the baseline for the blue sky priority user stories. The 3 in the version will specify the which version we are on. The first “x” will specify which iteration we are on. The third “x” will specify which bug fix has been worked on. |
|  |  |  |

## Code Check in Plan

On the local repositories, code can be branched and managed in whatever way the developer wants. When it is time to check the code in, the developer will have to:

* Go onto Trello and label the user story as “In Progress” (It will be a blue label).
* Checkout the current branch based on the first digit in the version number.
* Pull the origin/<current-branch> from the remote repository.
* Compare the current branch from the remote repository to the local branch that the additional code is on.
* If there are no conflicts, they can merge their branch to the current branch of the remote repository.
* They then have to notify Josh Overbeck of the new commit to the remote repository on Trello.
* Josh Overbeck will have to verify that the new commit does not conflict with the previous code on the remote repository.
* If there are no conflicts to be found, Josh Overbeck will have to go onto Trello and mark the user story as “Added” (It will be a pink label).
* After that, Feiyu Wang will have to test the new code.
* If Feiyu Wang finds any bugs, he will have to go onto Trello and label it with a “Bug” tag (It will be a black label).
* Once the bug is resolved, the “Bug” tag can be removed the the “Completed” tag can be added (It will be a Green label).