**Software Implementation and Testing Document**

**For**

**Group 4**

Version 2.0

**Authors**:

Brenden G

Stephen J

Benjamin W

Logan L

# Programming Languages

* Java
  + Java was chosen because most of the members of our group are familiar with it and it’s graphics and cross-platform features are perfect for our game. We also would like to expand our knowledge on the language and become more familiar with the graphics interface. This is being used for the game portion of the project.
* Javascript
  + Javascript was chosen to add parallax open source libraries to our website. Parallax is motion effects that make the website more interactive to the user. As of now, we have only used it to initialize the motion effects for our website in the Main.js file. Inline Javascript has also been used to include the Bootstrap containers. This was the easiest way of implementing the Bootstrap. More javascript may be added in the future to make the website more interesting.
* CSS/HTML
  + While not “programming languages” we have used them to build the framework of the website. Along with imported bootstrap classes, these languages are located within our several html files. The CSS has its own main file to be shared with all the web pages. We chose them because they are a standard for website development. Bootstrap CSS classes were used to make the website look nicer and more professional.

# Platforms, APIs, Databases, and other technologies used

* Java Swing
  + We used the Java Swing toolkit for implementing the graphics in the game portion of the project. This is used in the main game class to set up the window and draw the character and walls in the displayed room.
* AWS
  + We have used the AWS S3 bucket to host our static website. This located on Stephen Johnson’s personal AWS account. The website code is located within the website branch on github.
* Bootstrap
  + We have used Bootstraps open source libraries to help create some of the websites container classes. The container classes are in the divs of the html files. The places we’ve included the libraries in are linked at the top and bottom of the html files.

# Execution-based Functional Testing

In terms of the game, we ran the game many times to test our functional requirements. We checked to make sure the KeyListeners worked by using the arrows on the keyboard. We checked that the menu worked by clicking the buttons and making sure they all worked appropriately.

In terms of website, we made sure the user can click different pages and can click the download button, as well as many other buttons (and that they led to the right place). We checked to make sure when the user minimizes the screen, it adjusts appropriately. We also used a unit test to test the floor layout generation, drawing a representation of it in the terminal to make sure it worked. Have tested the download link to make sure that the game can be downloaded correctly.

# Execution-based Non-Functional Testing

* Target Platform
  + We tested the runnable jar file on both Mac and Windows to see if it worked when downloaded from the website to either platforms.
* Random Floor Layout
  + We tested the floor generations code by writing out the layout to the terminal. This let us visually see the floors we were testing.
* Game Lag and Character Movement
  + We held down the arrow buttons to make sure the character moves smoothly around the spaces.
* Website Lag
  + We tested the lag of the website by running it through a website domain speed test GTMetrix.

# Non-Execution-based Testing

For the Non-Execution based testing, all code pushed to the Git repository was reviewed by at least one other person, and we discussed optimization and errors in our Discord chat.