**VISUALIZING MICE BEHAVIOR WITH HEAT AND VECTOR MAP GENERATION**

**TEST PLAN DOCUMENT**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Compiled by: Josh Mwandu, Alex Brown, Parker Rowland

Version 1.0

03/31/2017

**TABLE OF CONTENTS**

1. **INTRODUCTION**
   1. Purpose of the Test Plan Document
   2. Statement of Scope
   3. Overview of the Remainder of the Document
2. **SYSTEM OVERVIEW** 
   1. Project Description
   2. Requirements List
3. **TEST PLAN**
   1. Testing Strategy
   2. Testing resources and staffing
   3. Test work products
   4. Test record keeping
   5. Test schedule

4**. Test Procedure**

4.1. Unit Tests

4.2. Usability Tests

4.3. Stability Tests

4.4. Performance Tests

4.5. Cross-Platform Tests

4.6. Implementation team Tests

5. **APPENDICES**

5.1. Glossary

5.2. Team Member Contributions

5.3. Additional Information

1. **INTRODUCTION**
   1. **Purpose of the Test Requirements Document**

The purpose of this document is to convey information about the testing processes for the mouse visualization software. Most of this project will be conducted under grey box testing so that testers have some access to the code with limited privileges. The implementation team will be the only party that will be able to do white box testing for the purposes for expediency in being able to produce a complete deliverable by the shipping date.

* 1. **Statement of Scope**

The System involves several key subsystems that must interoperate to produce the visualizations. A Graphical User Interface (GUI) provides the User with the means to operate the System. A data management subsystem enables the User to load datasets into the System for the purpose of producing visualizations. The heat and vector mapping component utilizes the necessary mathematical formulae for rendering images and animations of the loaded data set. Finally, an export manager gives the User the ability to save the rendered heat and vector map graphics as image files for use outside of the System. The Testing plan within this document originates from the discussions of the implementation team.

* 1. **References**

Dr. Polack provided the necessary descriptions of functionality that the System should possess at its implementation. Dr. Waters provided a clearer understanding of how the data sets will be used.

* 1. **Overview of the Remainder of the Document**

The remainder of this document is roughly divided into four sections of information. The system overview section gives a snapshot of the core features that should be completed when the final deliverable is created. The Testing plan section gives a break down into the nature of testing for this project and the kinds of tests that will occur during this development cycle. The testing procedure section contains a chart displaying the different kinds of tests happening in the projects various subsystems. The final section is an appendix which gives information about the glossary, and what the different members of the implementation team contributed to this testing plan document.

1. **System Overview**

**2.1. Project Description**

The purpose of this project is to read in large data set files that contain several mouse parameters and data that is then used to create different visualization tools to see patterns between each mouse. The program first starts with a GUI with limited options available. Upon loading a data file with the .CSV extension or a session file using .JSON, more options will become available upon a successful file opening.

The user can then select visualization options, such as a heat or vector map, along with the options to add grid lines or numbers to the image as well. With all that data together and any number of mice selected from the data set, you can then click the generate button to produce the visualization to the screen. Lastly, you can also export this visualization to your computer that can be saved as a .png or .jpeg file.

**2.2. Requirements List**

The main functions of the System allow the user to open two different file types, create different visualization tools from the loaded file, along with the ability to save that data as an image file type. The User interacts with the System through the graphical user interface to access its functions. The following use cases are descriptions of how particular core functions of the System work:

**2.2.1. Load Dataset**

**Description:** The User wants to load a new data set into the

System.

**Main Flow:** The User indicates that they want to load a new data set. The System opens the operating system’s file explorer, prompting the User to select a CSV or .JSON file to load. If the file selected is in CSV or .JSON format and the data within the file passes a validation check, the System prompts the User to enter the grid size associated with the experiment data in the file. The System then prompts the User for a name to identify the current session and saves the file path and session information to its internal database for quick use in future sessions.

**Alternate Flow A:** If the User selects a file that is not a CSV or .JSON file, then the System displays message to the User stating that the selected file type is invalid and that only CSV files can be accepted.

**Alternate Flow B:** If the data within the selected CSV or .JSON file fails to pass the validation check, then the System displays a message to the User stating that the selected file type is invalid and that only CSV or .JSON files with experiment data will be accepted.

**2.2.2. Delete Session**

**Description:** A User wants to delete a previously saved System Session.

**Main Flow:** User indicates that they want to delete a Session from the System’s list of previous Sessions. The System displays the Session Manager section, which contains a list of previous Sessions. If the User selects to delete the GUI control next to a particular Session, the System will prompt the User with a confirmation dialog asking if they would really like to delete the Session. If the User selects the affirmative choice, the System deletes the file association of the Session from its internal database and displays a success message to the User.

**Alternate Flow A:** If the User selects either the negative or cancel choice in the prompt, then the System cancels the delete Session operation.

**2.2.3. Generate Visualization**

**Description:** A User wants to generate a heat map or vector map visualization.

**Main Flow:** User indicates that they want to generate a visualization graphic. The System allows the User to specify certain parameters influencing the visualization, including map type, number of mice to graph, and static / dynamic graphic generation (“at once” vs. animated over time). Once the User has selected their parameters, the User activates the GUI control that begins the visualization generation process. The System generates the visualization, and the ability to export the visualization becomes available to the User.

**2.2.4. Export Visualization**

**Description:** User wants to save a visualization in an image file.

**Main Flow:** The User indicates that they want to export the current or selected visualization as an image file. The System opens the operating system's file navigation window asking the User what to name and has options to save as a .png or .jpeg extension. The System generates the image, saves it to the specified location given by the User, and displays a success message to the User.

**Alternate Flow A:** If the User closes the operating system's file navigation window, then the System cancels the image export operation.

**3. Test Plan**

This section describes the overall strategy for testing the System, the coordination required of the testing and integration teams for the purpose of correctly executing System tests, the types and formats of data records produced from the test team's efforts, and the testing schedule.

**3.1 Testing Strategy**

The types of tests that will be conducted on the System consist of Gray Box and Black Box tests devised to ascertain the System's quality of functionality, performance, stability, and usability. In order to provide the testing team both a realistic testing schedule and the ability to thoroughly investigate any code flaws, a majority of tests performed on the System will be Gray Box tests in which the testing team will use the NetBeans Integrated Development Environment (IDE) to run and examine the console output of the System.

The testing team is not permitted to modify the source code, but may examine any portion of the code they suspect to be the cause of any System errors discovered during the course of testing. Certain tests, such as the verification of the System's session handling capabilities, can not be performed in the NetBeans IDE, so Black Box tests will be conducted in those instances and also for any targeted System capabilities not covered by Gray Box tests. It is the integration team's decision to disallow any White Box testing by the test team for the sole purpose of achieving a reduction in the amount of time required by the test team to perform each test; this is necessary in order to provide the testing team a quantity of tests large enough to ensure that the System is thoroughly tested in all aspects identified by the integration team.

Tests will be executed to determine the System's quality in four key areas: functionality, performance, stability, and usability. System components that will be tested for functionality include all features specified as functional requirements by the Client and any functional dependencies of these features. For instance, the Client specified as a requirement that the User must be able to save and restore the current state of the System; in order to fully test this concept, tests must be performed on the mechanisms that both directly and indirectly impact the System's ability to save and restore its state, including the validity of data generated by Graphical User Interface (GUI) controls influencing the System's state.

The performance of the System will also be tested to ascertain the maximum amount of data that the System can process in a reasonable time, quantified by examining the length of data processing operations and periods of System unresponsiveness. Since consistent performance of the System is a critical goal of the implementation team, the testing team will perform stability tests to ensure that the System is consistent in its average processing speed and usability over the course of several independent sessions. The testing team will also be responsible for studying the effectiveness of the user interface by executing usability tests, ensuring that the form of the System's interface follows its function in a User-friendly manner.

Non-requirements of the System, interpreted by the implementation team as any aspects of the System not explicitly expressed by the Client as functional or non-functional requirements, will not be tested.

In order to verify the correctness of System components under development, the implementation team will continuously perform general testing of the System throughout the development period. The testing team will perform and document a subset of general testing considered to be critical to the overall quality of the System; the feedback as a result

The following is a summary of System components to be tested and their classification based on test type ([GB] indicates Gray Box testing, [BB] indicates Black Box testing):

**Functional Testing:**

* The opening of files [GB]
* The parsing of CSV files containing experiment data [GB]
* The parsing of JSON files containing System state information [GB]
* The ability to save the current session [GB / BB]
* The ability to delete a session in the session manager [GB / BB]
* The functionality of, data accepted by, and data input into GUI controls [GB]
* The visualization generation for all types of supported maps [GB]
* The exporting of visualizations as image files [GB]

**Performance Testing:**

* The duration of file load operations for both CSV data sets and JSON session files [GB]
* The duration of data processing operations for all types of supported maps [GB]
* The duration of time the System is unresponsive during file loading and data processing operations [GB]

**Stability Testing:**

* The consistency of general System performance and quality of functionality over the course of several independent sessions [GB]
* Ensuring the System works as an executable, independent of the NetBeans IDE [BB]
* Ensuring the System works on computers that run either Windows or Mac operating systems [GB / BB]

**Usability Testing:**

* The effectiveness of the user interface [BB]

**3.2 Testing Resources and Staffing**

In order to test the System, the testing team will need access to a set of specific hardware and software resources. Since it is a requirement that the System be able to operate on both Windows and Mac operating systems, the testing team will need access to at least one Windows computer and one Mac computer. The testing team may not use any dual-boot configuration or operating system emulation software to satisfy this condition.

Computers suitable for testing the System include the Windows computers residing in the Trinkle computer lab and the Mac computers in the computer lab on the first floor of the Hurley Convergence Center. Since NetBeans will be required for the Gray Box testing, the testing team will need a flash drive to contain an installation of NetBeans.

The test team must download the most recent stable release of the Java SE NetBeans bundle for the OS Independent platform available at<https://netbeans.org/downloads/>. Once downloaded, NetBeans must be installed to the flash drive that will be used for testing. Next, the test team will need to clone the GitHub repository containing the System's source code, available at<https://github.com/parkercode/mice-visualization>.

The easiest way to do this is to use NetBeans' built-in Git features, which allow the user to clone a repository from inside NetBeans and start a new project based on that cloned repository; instructions on how to use NetBeans' Git features are at<https://netbeans.org/kb/docs/ide/git.html>. Access to the data set file provided to the implementation team by the Client and an additional file containing aggregations of the data set’s values will be shared with the testing team. The file containing aggregated values of the original data set will be used for the purpose of verifying that correct ratios exist between the colors of different grid sectors in the heat map visualizations. The testing team must also have access to a text editor for the purpose of inspecting and modifying the contents of the data set files.

The following is a summary of the steps necessary to setup the testing environment ([IT] indicates the item is a responsibility of the integration team, [TT] indicates the item is a responsibility of the testing team:

1. Obtain a flash drive with enough storage capacity to contain a full installation of NetBeans and several copies of the main data set file [TT]
2. Download the Java SE NetBeans bundle for the OS Independent platform available at <https://netbeans.org/downloads/> [TT]
3. Use the Git features of NetBeans (described at <https://netbeans.org/kb/docs/ide/git.html>) to clone a copy of the System's source code from GitHub (link to the System's source code repository: <https://github.com/parkercode/mice-visualization>) into a new NetBeans project [TT]
4. Obtain a copy of the data set files from the implementation team [IT]
5. Test that the installation of NetBeans residing on the flash drive works on both Windows and Mac computers [TT]

**3.3 Test Work Products**

The testing team will use the NetBeans IDE as a source of information regarding the causes of any System errors, which are reported by NetBeans as exception messages printed to the NetBeans console during the execution of the System. A text copy of the NetBeans console output from each System session involving functional tests, along with the necessary annotations and notes provided by the testing team, will be the primary source of records generated for testing the functionality of the System's components; reports of Black Box functional testing will consist of information regarding the pretest expected functionality, the actual functionality experienced during and immediately after the test, and any noticeable changes in System state following a bug or System error.

Since the performance testing of the System will heavily rely upon timed durations of System events, the testing team will employ a digital timer with a minimum precision down to the millisecond and will record, in a text summary, any timing data gathered during testing. The testing team will utilize a checklist-type tool during the stability testing phase, for the purpose of checking System stability over repeated sessions. During the usability phase of testing, the test team will record their general comments about the user interface as a text summary.

Any bugs discovered during the course of testing will be tracked by recording the following information in a text report:

1. the name of the test that the bug occurred in
2. the timestamp of the bug occurrence
3. any NetBeans console output associated with the bug plus ten lines prior to the exception (if applicable)
4. any relevant comments about how the System's state changed as a result of the bug.

**3.4 Test Record Keeping**

In order to report the results of the testing to the implementation team, the testing team will record its testing activity by identifying the current test, expressing relevant information about the current test, logging System events, recording the location of any source code findings, and recording any relevant System output.

Any tests that fail must be repeated at least once, for the sake of determining the error’s replicability. Tests that pass the first time do not need to be repeated.

For the sake of maintaining a consistent format of test results, the testing team will record their tests in the following structure (each item, where applicable, may be recorded in a sentence or less):

1. Test Title
2. Name of tester
3. Date and time of test in (MM/dd/yyyy hh:mm) format
4. Describe the actions taken in the System taken to setup the test. This should be a description of the state of the System prior to beginning the test.
5. Describe the test itself, including the state of the System during the test.
   1. If the test is a Gray Box test involving the NetBeans IDE, additionally include any console output generated during the test by copying and pasting it into this section.
   2. If the test involves verifying any graphic component of the System, include any screenshots of the relevant System components.
6. Describe the state of the System after the test
   1. If the test triggers a bug in the System, record the System’s state after the bug and whether or not it appears that the System can continue to function.
7. Number of times the test was performed
8. Summarize the test result with one of the following status words: PASSING, FAILING, or INCONCLUSIVE
9. Summarize the ability of the test result to be replicated: REPLICABLE, IRREPLICABLE, INDETERMINATE

**3.5 Test Schedule**

The window of time that the testing team has to perform tests on the System will be scheduled by Dr. Anewalt. The testing team will perform all tests listed in section 4.1 during this time frame. The order of test execution is not important, but it is critical that all test results are documented according to the standard format set forth in section 3.4.

General testing of the System will be performed by the integration team continuously throughout the development period, due to the chance that the evolution of System features and capabilities over time may introduce or reveal bugs in the System.

**4. Test Procedure**

**4.1. Functional Tests**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Number** | **Related to Req** | **SubSystem** | **Purpose** | **Test Case Data** | **Expected Results** | **Verify Results** |
| **1** | **2.2.1** | **Load Dataset** | **Check for bad or incomplete data** | **“Mickey Mouse” placed in the IdRFID column of a random data row in the Hannah mouse position data set** | **After loading the data set file containing the bad data, the System will either warn the User that bad data is present in the data set OR the System gracefully handles the bad data by throwing out (skipping) its associated data row** | **Check if the IdRFID containing the bad information is displayed in the Selected Mice list contained within the Visualization Options** |
| **2** | **2.2.3** | **Visualization** | **Check if visualization properly represents the data** | **Hannah mouse position data set,**  **Aggregate position data set** | **Once the User selects the necessary options for a static heat map of 6 mice and activates the “Generate” button, the program should display a visualization of the data** | **Compare the results of the graph with the aggregate position data totals and judge if the visualization is an accurate representation of the data** |
| **3** | **2.2.4** | **Export** | **Check if visualization is properly formatted for exporting** | **Hannah mouse position dataset** | **After generating a visualization, selecting the “Export” menu option, and saving the file, a jpeg or png of the expected graphic should be created** | **Locate and attempt to open the image file** |

**4.2. Usability Tests**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | **3.2** | **GUI** | **Check if the GUI properly works while in use** | **Not applicable** | **The System should respond to user interactions with its GUI controls in a way that roughly matches the tester’s user model** | **Not applicable** |

**4.3 Stability Tests**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | **3.2** | **sessions** | **To check if sessions can be saved persistently** | **Not applicable** | **When the user first opens the System and imports a CSV file containing a data set, the System should prompt the User to create a new session file.** | **After following the System prompts for creating a new session file, locate the session file on disk.** |

**4.4 Performance Tests**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | **2.21** | **Load dataset** | **Check the amount of time the program takes to load a normal dataset** | **Hannah mouse positions dataset** | **After importing the Hannah mouse positions dataset file into the program, the System should output the amount of time taken to load the file in the lower left status area** | **Use a physical or digital timer to confirm the amount of time it takes the System to load the data set** |
| **2** | **2.23** | **Visualization** | **Check amount of time it takes to process larger datasets** | **Hannah mouse positions dataset** | **After visualizing the static vector map for 6 mice, the System should output the amount of time taken to process the data in the lower left status area** | **Use a physical or digital timer to confirm the amount of time it takes the System to load the data set** |

**4.5 Cross platform testing**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | **4.5** | **Operating System** | **Test the program on multiple operating system ranging from windows to mac** | **The operating system running to program** | **The program should run without unplanned for behavior** | **Any unique errors being created in the netbeans log created by the operating system being used** |

**4.6 Implementation team testing**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | **2.2.1** | **Save new session** | **Ensure the user does not enter an invalid session file name** | **Hannah mouse positions dataset** | **When loading a data set file into the System if there is no session file for the current session, the System will prompt the User to enter a name for the current session’s file.** | **If the User enters a name that contains invalid characters, the System will alert the User and ask them to enter a new name.** |
| **2** | **2.23** | **Heat map** | **Ensure each mouse in the dataset, passed in properly, renders when selected** | **Hannah mouse positions dataset** | **No error appears during the generation of a heat map visualization** | **A graphic of the mouse's behavior should be produced, and no errors appear in the NetBeans IDE** |
| **3** | **2.23** | **Vector Map** | **Ensure each mouse in the dataset, passed in properly, renders when selected** | **Hannah mouse positions dataset** | **No error appears during the generation of avector map visualization** | **A graphic of the mouse's behavior should be produced, and no errors appear in the NetBeans IDE** |
| **4** | **2.2.2** | **Session manager** | **Ensure that if the user tries to delete the current session’s file, the System will gracefully reset its state** | **Hannah mouse positions dataset** | **After loading the Hannah mouse positions dataset and creating a new session, the session file of the current session should appear in the recent sessions list. If the user selects the current session in the manager and activates the Delete button, the System should alert the user that proceeding will reset the state of the System.** | **After deleting the current System’s session file via the Sessions Manager, the System’s state, including its GUI, should be reset to default.** |
| **5** | **2.2.1** | **Load dataset** | **Ensure the dataset can be parsed into a mice object** | **Hannah mouse positions dataset** | **After a data set file is loaded, the System will parse it into a mice object** | **Only a single mice class object instance should exist per session** |

**5. Appendices**

**5.1. Glossary**

**Client:** The client is an interdisciplinary team of biology, psychology, and computer science researchers investigating the feasibility of developing an artificial intelligence model for group mouse behavior.

**Comma Separated Value (CSV):** A type of data format associated with text that consists of rows of data separated by line breaks. Within the rows of data, individual values are delimited by a value separator, which can be any character so long as the character does not appear within the values of the row data and only if that delimiter is used consistently throughout the file.

**JavaScript Object Notation (JSON):** A type of data format

associated with storing session data that was previously saved

from using a CSV file type. It is just a long line of code saving

different parameters from the program that is used to load that

same data back into the program at any time.

**Cascading Style Sheet (CSS):** A type of data format associated with identifiers that correspond to sets of specific styling properties, typically used with HTML or XML documents.

**Data Set:** A collection of related sets of information that is composed of separate elements but can be manipulated as a unit by a computer (Google definition).

**Graphical User Interface (GUI):** The visual component of a software program that allows Users to interact with the software.

**Heat Map:** A graphical representation of the aggregate intensity of specific data parameters projected onto a data matrix, where the axes are particular parameters. In this project, the axes of the heat map reflect the dimensions of the mouse enclosure used in the experiment.

**Integrated Development Environment (IDE):** A program or set of synchronized programs that provide an interface through which a User can write, test, debug, and ship code.

**JavaFX**: A programming framework that implements several languages to achieve a graphical user interface.

**Radio Frequency IDentification (RFID):** RFID is a technology that implements tags (transponders) containing information and readers that capture that data from the tags (transceivers). A reader sends a radio signal broadcast specific to the RFID application, and if there are any tags in the vicinity of the broadcast, the tags send their information back to the reader in the form of passive backscatter or active radio transmission. RFID is commonly used in asset tracking applications. In this project, RFID is responsible for generating the data of mouse location and duration of time spent within the sectors of the enclosure.

**Session:** The current state of the System, including its current data set and visualization parameters.

**System:** The system is the software program being built for the Client and interacted with by the User.

**User:** The user is the person using the System.

**Vector**: A quantity having direction as well as magnitude, especially as determining the position of one point in space relative to another (Google definition).

**Vector Map:** A graphical representation of the path vectors traveled over time. In this project, the vector map will show the paths traveled by individual mice within the enclosure.

**eXtensible Markup Language (XML):** A structured document format that consists of tags and elements meant to be interpreted both by humans and computers.

**Blackbox Testing (BB):** Testing the program without access to the source code.

**Greybox Testing (GB):** Testing the program with some access to the source code.

**5.2. Contributions**

All team members worked together to create this test plan document.

Specific contributions are listed below:

**Parker:** Test plan sections 3.1-3.5, the “Verify Results” column in the tests of section 4

**Josh:** Introduction sections 1.1-1.3

**Alex:** System overview sections 2.1-2.2

**All:** Test procedure 4.1-4.4, Appendices section 5.1-5.3

**5.3. Additional Information**