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

**PROJECT PLAN DOCUMENT**

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

Compiled by: Josh Mwandu, Alex Brown, Parker Rowland

Version 1.0

03/1/2017

**TABLE OF CONTENTS**

1. **INTRODUCTION**
   1. Purpose of the Software Requirements Document
   2. Scope of the System and Subsystems
   3. References
   4. Overview of Remainder of Document
2. **PROJECT DESCRIPTION**
   1. Client Characteristics
   2. User Characteristics
   3. Data Source Characteristics
   4. Functional Requirements
      1. Load Dataset
      2. Delete Session
      3. Generate Visualization
      4. Export Visualization
   5. General Constraints
3. **SCHEDULE**
   1. Approach
   2. Milestones and Deliverables
   3. Work Breakdown Structure
   4. Gantt Chart
   5. Task Dependency Diagram
4. **APPENDICES**
   1. Glossary
   2. Team Member Contributions
   3. References
5. **INTRODUCTION**
   1. **Purpose of the Software Requirements Document**

The project plan document details all information pertinent to the strategy of System development devised by the implementation team. The information within this document provides an in-depth description of the project background, project plan, stipulations, and other details in order to give the reader a high level overview of the product. The intended audience of this document is comprised of the Clients, stakeholders, and developers of the System.

* 1. **Scope of the System and Subsystems**

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 project 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 Remainder of Document**

The remainder of this document is roughly divided into four sections of information. The first is an overall description of the System that includes background information, descriptions of the Client and User, as well as a list of use cases that demonstrate the System's intended usage. The second section is devoted to the requirements and non-requirements of the System, which is an analysis of the System's capabilities in closer detail. The penultimate section provides details about the development schedule of the System. The last section contains an appendix of information containing a glossary of terms and a list of contributions to this documentation made by each team member.

1. **PROJECT DESCRIPTION**

**2.1. Project Scope**

The purpose of the System is to visualize data collected from a series of research experiments. The System will be capable of running on both Mac and Windows operating systems. The data visualizations generated by the System will be able to be exported to image formats for use outside of the System. In addition, Users will be able to save their Sessions with the System, encompassing the data set and visualization settings used, to allow collaboration between Users.

**2.2.** **Client Characteristics**

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. To help determine the presence of any significant patterns in the experiment data, the Client needs a tool that can generate heat maps and vector maps (line paths) of mouse activity over time. The Client, as referenced in this document, is defined in section 4.1.

**2.3. User Characteristics**

The User of the System is defined as one of the researchers that comprise the Client group. The User is characterized as being a professional academic in their respective field of study with a basic understanding of computer operation. The User, as defined in this document, is defined in section 4.1.

**2.4. Data Source Characteristics**

The data sets that the System is capable of processing contain the raw data captured from experiments conducted by the researchers. The experiment involves a fixed enclosure divided into grid sections, each with an Radio Frequency Identification (RFID) component, in which a group of one to eight mice traverses for a period of time. If a particular mouse remains stationary over a grid section for a period of time greater than a specific minimum threshold duration (such as 100 milliseconds, for example), a data entry of the amount of time the mouse remains in that grid section begins and is committed to the data set once the mouse leaves that grid section. If a particular mouse moves faster than the minimum threshold duration through a grid section, a data entry with a zero value is committed to the data set once the mouse leaves that grid section. In this way, the data sets are comprised of the stationary activity and transient movements of each individual mouse over a period of time. The format of the data sets is Comma Separated Value (CSV).The sizes of the data sets are not upper bounded, and include files as large as 40 megabytes (approximately 530,000 data entries).

**2.5. Functional Requirements**

The product functions of the System allow the User to load and visualize input data as well as create export images of the generated visualizations. 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:

**2.5.1. 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 file to load. If the file selected is in CSV format and if 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 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 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 files with experiment data will be accepted.

**2.5.2. 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 the delete 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:**

The System has no previously loaded Sessions.

**Alternate Flow B:**

If the User selects either the negative or cancel choice in the prompt, then the System cancels the delete Session operation.

**2.5.3. 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.

**2.5.4. Export Visualization**

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

**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 where to save the image. 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.

**2.6. General Constraints**

The System is to be implemented in Java at the request of the Client. There are no time constraints regarding the generation of visualizations, nor in the loading and processing of data sets. Additionally, there is no upper bound on the amount of data that the System must be able to process. All data sets that the System is able to process are in CSV format. The System must be able to run on both Windows and Mac computers.

**3. PROJECT SCHEDULE**

**3.1 Approach**

Since the System is required to be implemented through a GUI, the development schedule sequences requirements-implementation tasks that correspond to stages of user interface construction. The rationale behind this approach is that the sequential implementation of certain GUI elements leads to the functionality of other GUI elements, each of which corresponds to one or more project requirements. For example, in order for the Client to graphically access file load and save operations, a menu bar with options must be created in the user interface; once the menu bar options are implemented, the functionality they provide can then be used to begin implementing data parsing and visualization tasks.

The code will consist of files and their associated technologies required for running a JavaFX application, including Java, CSS, and XML. Since JavaFX applications allow the separation of user interface scaffolding, style, and event handling logic, the System’s code will consist of several files that contain functionality corresponding to this separation. The code will be written in the NetBeans IDE, and the development team will handle version control via the Git integration built into NetBeans.

The development team will use an iterative approach during the construction of the System. Iterations of development activity will be focused on implementing major functions of the System, which will be demonstrated to the Client in the form of deliverables.

**3.2 Milestones and Deliverables**

The following is a list of project milestones and deliverables. **Deliverables** are bolded, underlined, and noted as parent items in the **3.2.x.** list hierarchy below and contain descriptions of which System features will be demonstrated to the Client at feedback meetings involving new System features. Every deliverable contains a number of milestones, represented as child elements within the below list, each of which are described in terms of implementation details, desired outcomes, and time to completion estimates. In the deliverables / milestones hierarchy below, time estimates of parent elements are the sum of the time estimates of their child elements.

1. **Gui scaffold** : 2 hours

Description: The GUI scaffold is the basic UI framework used for development purposes when building the software. At the end of the implementation phase, the GUI scaffold will contain the logic necessary for linking System functions to their corresponding GUI elements. The development team is utilizing a visual GUI builder called Scene Builder, which allows us to rapidly prototype designs and integrate the corresponding XML code into NetBeans.

* 1. **Integrate scaffold into Netbeans**: 1 hour

Description: Export the XML code generated by Scene Builder and integrate it into the Netbeans IDE.

* 1. **Set up git, test push/pull the GUI scaffold code**: 1 hour

Description: Perform an initial commit of the GUI scaffold to a Git repository on GitHub, for the purposes of implementing version control on each team member’s development machine. Utilize Netbeans’ built-in Git features to add, commit, push, and pull code.

1. **Data Set Functions**: 20 hours

Description: The System’s data set functions comprise the User’s ability to load, save, rename, and delete data used by the System. Once fully implemented, the System will be able to load a data set from a file, validate the format of the selected file and the data contained within, convert the data within that file to data structures for internal use by the System, save the loaded data set and the visualization option settings selected by the User as a Session, display previous Sessions, load Sessions, rename Sessions, and delete Sessions.

* 1. **Display a list of previous Sessions**: 1 hour

Description: This option displays a list of previously saved Sessions. The User will have the ability to select a Session in this list and restore it as the current Session.

* 1. **Call OS file system to allow User to select file:** 1 hour

Description: In order for the user to graphically choose which data files to use with the System, the System will provide a way for the User to access the underlying OS’s file system. The built in OS file system will select the requested User file and pass it to the System for processing.

* 1. **Parse the data:** 12 hours

Description: After a dataset has been selected by the User, the System will parse the data, organizing and formatting it to facilitate accurate and time-efficient visualizations. This formatting will chronologically reorder the entries in ascending order as well as remove any data entries that have missing fields. As the data is parsed, it will be stored in several data structures representative of key objects in the scientific experiment and System, such as Mouse, Grid, Row, Visualization, and Timeline classes.

* + 1. **Threading considerations**: 4 hours

Description: The baseline data file provided to us by the Client is a significantly large CSV file and there is no upper bound on file size of data sets accepted by the System. Considering these factors and interface responsivity, the implementation team will investigate the use of multithreading techniques to alleviate potential performance issues during the parsing of data sets by the System.

* + 1. **Read data into classes**: 7 hours

Description: The System will convert the data contained in the selected data set to data structures that will facilitate the System’s operations on the data.

* + - 1. **Clean the data**: 1 hour

Description: During the conversion of the data set into data structures, the System will disregard any unnecessary column headers and any other information not essential to the creation of visualizations.

* + - 1. **Order the rows by timestamps**: 1 hour

Description: The data within the data structures will be organized by timestamp value, from earliest time to latest time.

* + - 1. **Row class**: 1 hour

Description: The System will have a general purpose data structure containing row data of each record from the data set.

* + - 1. **Mouse class**: 1 hour

Description: The System will have a mouse class for the purpose of storing location, duration, and identification information specific to each mouse.

* + - 1. **Grid class**: 1 hour

Description: The System will have a grid class to represent the enclosure of the mouse experiment.

* + - 1. **Visualization settings class**: 1 hour

Description: The System will have an visualization settings class to represent the state of the current visualization parameters.

* + - 1. **Timeline class for animation**: 1 hour

Description: The timeline class will be used to create keyframe based animations of the visualizations.

* + 1. **Loading indicator:** 1 hour

Description: A loading indicator will be used to provide feedback to the User about the progress of loading operations and visualization generations.

* 1. **Prompt User to enter grid size associated with data set**: 1 hour

Description: The System will ask the User to provide the grid dimensions, represented as the (length x width) number of grid sectors. The System will use this information primarily as a validation check when processing the selected data set.

* 1. **Prompt User to save current Session**: 1 hour

Description: The System will store information about the current data set being used and the current visualization options as a Session in JSON format. The User will be able to save the current Session through the use of specific GUI controls. The JSON object will be saved in a text file to a designated directory on the operating system.

* 1. **Enable the visualization options**: 1 hour

Description: Once a Session or new data set has been loaded, the System will enable the visualization options for use by the User.

* 1. **Create Session storage scheme, folders, files**: 2 hours

Description: The System will utilize a scheme of storing and retrieving Session data contained in text files for the purpose of allowing Users to quickly load previous Sessions and their associated data. This will involve the designation of a particular directory for saving / reading Session files and file format validation.

* 1. **Enable Users to rename / delete Sessions**: 1 hour

Description: Users will be able to rename and delete previous Sessions that appear in the Session Manager.

1. **Visualizations**: 32 hours

Description: The System will be capable of displaying two different graphical representations of the data set selected by the User. The visualization options available allow the User to specify visualization type, mouse selection, static / animated generation, and animation parameters. Once the User specifies visualization parameters, the User will be able to activate a GUI control to begin the rendering of the visualization, which will then appear in the Viewer area of the GUI. The large amount of time required to implement this deliverable is primarily a consequence of the mathematical complexity of the mapping algorithms, which the members of the development team are unfamiliar implementing in Java FX.

* 1. **Implement visualization options**: 32 hours

Description: The implementation team will investigate different avenues of approach when creating the code necessary to generate the heat, vector, and overlay maps. If pre-existing Java solutions to the creation of these types of visualizations are found, the implementation team may use those solutions through their APIs or libraries, if permissible by Dr. Anewalt.

* + 1. **Heat / Vector / Overlay map**: 19 hours

Description: The User will have the choice of generating three types of visualizations: heat map, vector map, or an overlay of both the heat and vector map.

* + - 1. **Heat map implementation**: 8 hours

Description: The User will be able to select heat map as the visualization type. The heat map will show aggregate mouse activity over time within the experiment enclosure. For a definition of heat map, see section 4.1.

* + - 1. **Vector map default implementation**: 9 hours

Description: The User will be able to select vector map as the visualization type. The vector map will show a history of the path taken by a mouse within the experiment enclosure over time. For a definition of vector map, see section 4.1.

1. **Disappearing vectors**: 1 hour

The User will be able to specify the duration of how long the vector trails of mice should remain on screen during the generation of vector maps.

* + - 1. **Overlay map default implementation**: 2 hours

Description: The User will be able to select overlay map as the visualization type. The overlay map will contain the superimposed visualizations of both the heat and vector maps.

* + 1. **Legend**: 1 hour

Description: The System will provide the user with a legend detailing information about the visualization output.

* + 1. **Select mice to visualize**: 3 hours

Description: The User will be able to instruct the System on which of the eight mice should be visualized. A multi-select GUI control will be implemented.

* + - 1. **Implement multi-select**: 2 hours

Description: The user will be able to select 1 - 8 mice to visualize.

* + - 1. **Color picker**: 1 hour

Description: The user will be able to associate each mouse with a particular color.

* + 1. **Static / animated visualization generation**: 1 hour

Description: The User will be able to select static generation, corresponding to an immediate rendering of the visualization, or animated generation, which will incrementally produce the visualization depending on the animation parameters set by the User.

* + 1. **Start - stop time indices**: 1 hour

Description: The User will be able to specify the beginning and ending timestamp indices that will contain the range of data for the System to visualize. The System will automatically populate these fields with the first and last data indices of the data set.

* + 1. **Animation options**: 7 hours

Description: The User will be able to customize the animation of the selected map type with the abilities to adjust speed and use transport controls to pause, play, and restart the animation.

* + - 1. **Animation speed**: 3 hours

Description: The user will be able to adjust the speed that the System incrementally displays updates to the visualization; the speed will be controlled by the rate of the passage of animation keyframes.

* + - 1. **Transport controls**: 4 hours

Description: The user will be able to pause, start, and restart the animation.

1. **Export / Save options**: 4 hours

Description: The User will be able to save their progress with the System by exporting generated visualizations and by saving Sessions.

* 1. **Visualizations**:2 hours

Description: The User will be able to save the generated visualizations to a specific location on the OS file system.

* + 1. **PNG, JPEG format**: 1 hour

Description: The User will be able to specify the format that the visualization should be exported in.

* + 1. **Image export dimensions**: 1 hour

Description: The User will be able to specify the dimensions of the image export of the visualization.

* 1. **Sessions**: 2 hours

Description: The System will provide both an explicit save operation exposed to the User through a dedicated GUI control and a continuously saving feature for preserving Sessions.

* + 1. **Save data command**: 1 hour

Description: The User will be able to explicitly save the current Session by activating a particular GUI control.

* + 1. **Data continuously saved**: 1 hour

Description: Whenever the User makes a change impacting the state of the System, the System will automatically write those changes to the Session’s file, if one exists.

1. **Test Windows, Mac support**: 2 hours

Description: The implementation team will test the System for cross-platform support on Windows and Mac machines.

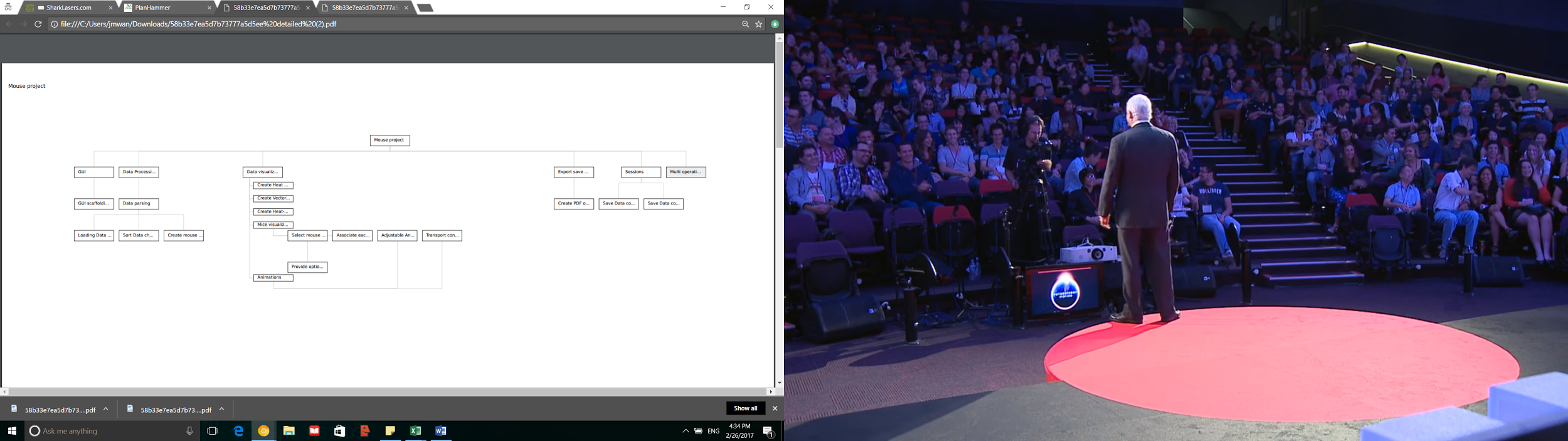
1. **Style improvements**: 2 hours

Description: The System should appear professionally designed. CSS will be used to create the styles for individual GUI elements.

**TOTAL ESTIMATED TIME TO COMPLETION: 62 hours.**

**3.3 Work Breakdown Structure**

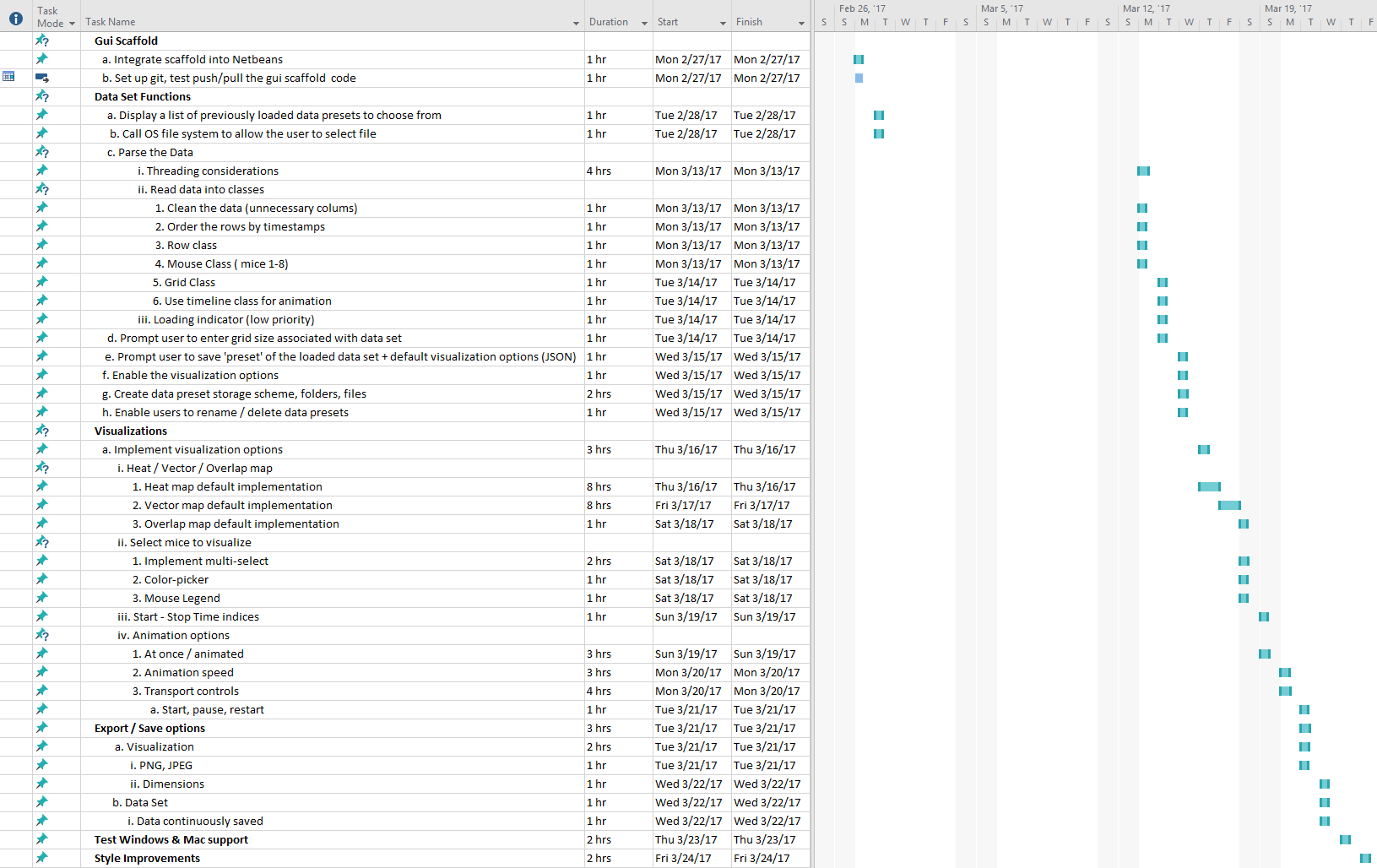
This section contains a visual breakdown of project corresponding to the milestones and deliverables.



**Figure 1**: Work breakdown structure.

**3.4 Gantt Chart**

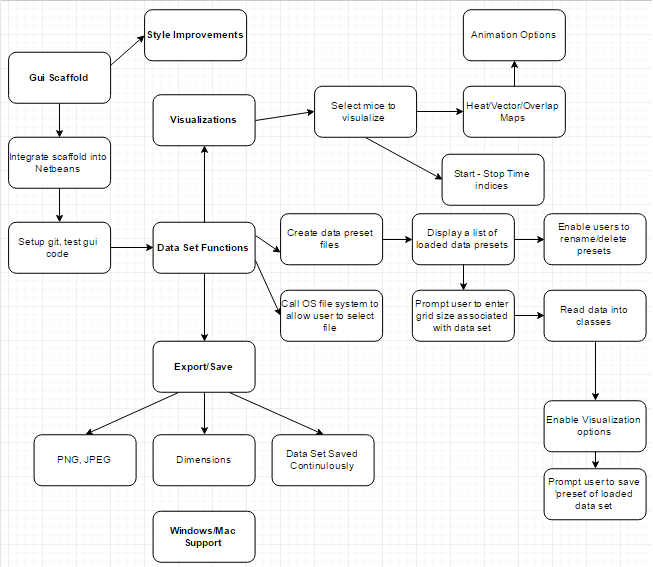
This section contains a visual depiction of the durations of individual project milestones and their projected start and stop times.



**Figure 2**: Gantt chart depicting the beginnings, durations, and ends of project milestones and deliverables.

**3.5 Task Dependency Diagram**

This section contains a visualization of dependencies in the System, followed by justifications for why such dependencies exist.



**Figure 3**:Task Dependency Chart depicting what needs to done first before

moving onto new milestones. Main dependencies are bolded

**Dependency justifications**

**Gui Scaffold**: Since this project uses a GUI interface, everything is dependent on the GUI being fully developed to test properly throughout the project.

**Style Improvements**: Once the core functionality of the System has been implemented, style changes will be added to the System for the purpose of enhancing the User’s experience. Changes will impact the shapes, sizes, colors, and graphic responsivity of most GUI elements.

**Data Set Functions**: This is the core part of the project. Without any data set functionality, the System will do little except display an empty viewer area in the GUI . This section involves everything dealing with the data sets given to the development team for testing purposes. The visualizations generated by the System will be based on these data sets.

**Export/Save**: A data set is required to be parsed and processed into data structures before the User is able to export the visualization maps for use outside the System. The System will also save data sets as part of the current Session’s data.

**Windows/Mac Support**: Cross-platform support will require the implementation of platform-specific features, such as the Windows and Mac versions of function calls involving the file system. Although only a small fraction of the System will contain platform-specific code, the entire System depends on operating system detection and correct usage of platform-specific functions.

**4. APPENDICES**

**4.1 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.

**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.

**4.2 Team Member Contributions**

All team members met throughout various times within the deadline

to discuss the layout and the intended information to comprise this

project plan document. Team members may have also contributed to this document in other minor details that are not listed in the following:

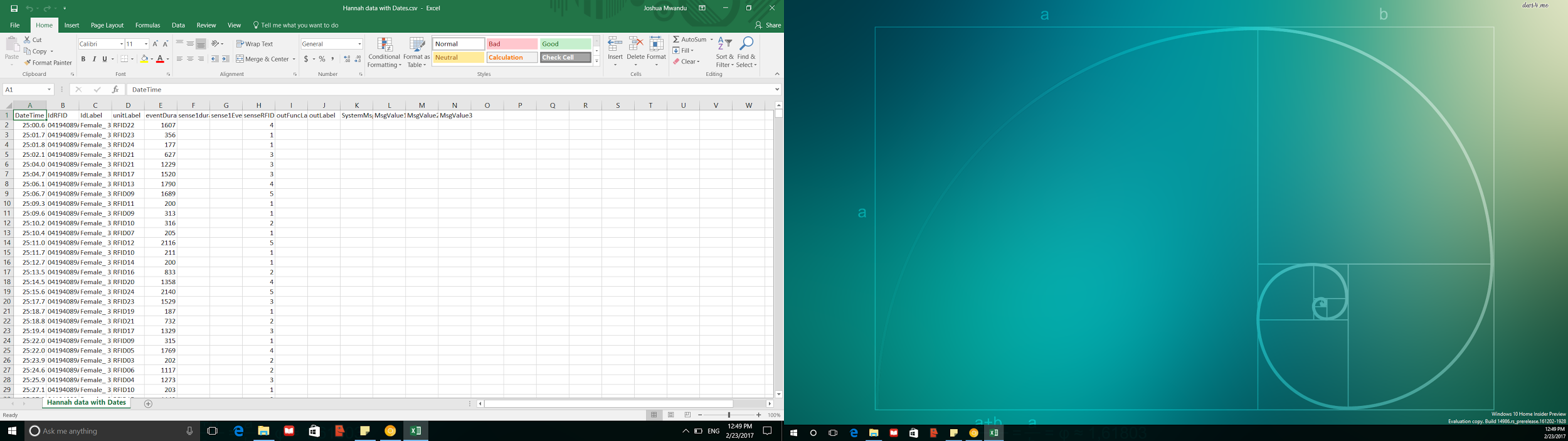
**Parker Rowland:** Merging specification parts into project plan, 3.1 approach, 3.2 Milestones and Deliverables, use of terminology consistency checking, additional definitions added to the Glossary.

**Josh Mwandu**: Visualizing mouse behavior, 3.2 Milestones and Deliverables, work breakdown structure, formatting document consistency

**Alex Brown**: Table of Contents, merging specification parts into project plan, Gantt Chart, Task Dependency chart.

**4.3 References**

The Client has given the development team a data set to use for the purpose of creating the data handling functions of the System, including data set loading, data parsing, and visualization generation. Below is a screenshot of a portion of the data, which is in CSV format:



**Figure 4**: Screenshot of the data set to be parsed by the System.