**System requirement specification**

**1.Non-functional requirements**

* 1. **Usability requirements**

**1.1.1 There is a GUI**

Priority: high

**1.1.2 GUI shows relevant experimental output and let the user select which ones to view**

Priority: Medium

**1.1.3 GUI is clear and follows usability heuristics**

Priority: High

**1.1.4 What graphs to display can be selected by the user**

Priority: Low

**1.1.5 There is a specific interface for the set-up calibration system**

Priority: Low

**1.1.6 Help and documentation is available for the system**

Priority: Medium

**1.1.7 Code is well documented**

Priority: Medium

* 1. **Performance requirements**

**1.2.1 Delay between mouse input and when the system records data input is low or can be calibrated for, such that it doesn't affect experimental data**

Priority: High

**1.2.2 System performs well over longer experiments**

Priority: Low

**1.2.3 System displays output in real time**

Priority: Medium

* 1. **Other**

**1.3.1 Software is easy to maintain**

Priority: Medium

**1.3.2 Data is saved in standard format**

Priority: High

**1.3.3 Timing of trigger and sensor inputs is correlated**

Priority: Low (?)

**1.3.4 All data files are timestamped at the start of recording**

Priority: High

**2. Functional requirements**

* 1. **Set-up calibration**

Precondition: Hardware setup including sensor placements relative to center must be done and known.

Trigger: Upon start up, program will alert the user in a clear fashion that calibration is a must for running the software successfully. Options between loading an existing calibration file or creating a new one is presented to the user. If no calibration file exist, the user must calibrate so its not possible to access the main functionality until this has been done.

Flow: After selecting, “create new calibration file” the calibration screen appear, here input fields for ball radius (or diameter) exists. The user can also input the sensor placement relative to each other and to the center of the ball. Finally, the user can run experiments where the ball is moved and the distance moved will be showed in form of sideways, forward and rotation.

Alternate flow: If the old calibration file is loaded, before this is executed a warning pop up appears with information emphasizing the need of correct calibration.

Result: After calibrating the system is fully functional and correctly calibrated.

Priority: High

* 1. **Mouse identification**

Precondition: Data calibration is done and both sensors are connected to the computer.

Trigger: The user selects “calibrate mouse function” in the GUI. If the user haven't identified the mice and tries to start the system the user will be prompted to do the calibration before being able to run the experiment.

Flow: A new window opens where the user can select which two mice to use to read data from for the experiment. When selecting two there will be a output showing which one is selected by activating the sensor corresponding to the given mouse selection on screen.

Alternate flow: Load old calibration from file and info pop-up appears emphasizing the importance of correct settings

Result: Sensor data will now be read from the correct mice and execution of experiment can commence.

Priority: High

* 1. **Data is saved to file**

Precondition: Directory path for where to save data is set.

Trigger: A recording is started.

Flow: Data is saved iteratively to file.

Alternate flow: Any I/O-error or errors related to corrupt data leads to an error-message for the user providing details of what went wrong.

Result: Data is saved into file and user is informed of that fact or an error message describing the problem will be provided for the user

Priority: high

* 1. **Data can be recorded using trigger**

Precondition: Set-up calibration and mouse identification is done

Trigger: “Use trigger” is selected and “start recording” is pressed.

Flow: System will wait for trigger before starting recording.

Alternate flow: -

Result: A recording is saved to file

Priority: high

* 1. **Data can be recorded without trigger**

Precondition: Data and mouse calibration is done.

Trigger: In settings, record without trigger is activated, then requirement is triggered by user pressing “start recording”.

Flow: User press start, recording is ongoing until stopped or timer runs out.

Alternate flow: -

Result: Recording is shown in plots selected by the user.

Priority: Medium

* 1. **User can set trigger threshold**

Sampsamp

* 1. **User can view object movement on a 2D map**

Precondition: A successful recording must have been made

Trigger: User has selected to view data in this particular format

Flow: As the recording starts the 2D-trajectories are drawn as long as the system keeps on recording. Stops but is still visible when the recording is terminated (for whatever reason). At any point the user can clear the map and it will start redrawing from scratch.

Alternate flow: Data is corrupted and makes no sense to present in this format. In this case an error-message will pop up informing the user what is going on.

Result: A 2D map is displayed.

Priority: Low

* 1. **User can view object movement speed, forward and sideways**

Precondition: A successful recording must have been made.

Trigger: The user has selected to view data in this particular format

Flow: As the recording starts the map is drawn as long as it keeps on recording. Stops but is still visible when the recording is terminated (for whatever reason). At any point the user can clear the map and it will start redrawing it from scratch

Alternate flow: Data is corrupted and makes no sense to present in this format. In this case an error-message will pop up informing the user whats going on

Result: Graphs for each speed dimension is shown

Priority: Low

* 1. **User can stop recording using keyboard as input device**

Precondition: A recording must be going on.

Trigger: The user presses key for stopping recording (standard space, can be set in system though)

Flow: When the user sends the stop command the recording stops and the mouse cursor is once again under the control of the user.

Alternate flow: -

Result: Recording stops

Priority: Medium

* 1. **A timer can be used to dictate recording interval**

Precondition: Set-up calibration and mouse identification must be done.

Trigger: User has selected to view data in this particular format.

Flow: As the recording starts the map is drawn as long as it keeps on recording. Stops but is still visible when the recording is terminated (for whatever reason). At any point the user can clear the map and it will start redrawing it from scratch.

Alternate flow: Data is corrupted and makes no sense to present in this format. In this case an error-message will pop up informing the user whats going on.

Result: Graphs for each speed dimension is shown.

Priority: Low

* 1. **There is a python client for network triggering**

Precondition: Network triggering is selected as trigger option

Trigger: -

Flow: -

Alternate flow: -

Result: -

Priority: High

* 1. **U****ser can set port and hostname for trigger server**

Precondition: -

Trigger: -

Flow: -

Alternate flow: -

Result: -

Priority: Low

* 1. **User can start recording**

Precondition: -

Trigger: -

Flow: -

Alternate flow: -

Result: -

Priority: High

1. **Requirements sorted by priority**

**High:**

* There is a GUI which will have necessary user input
* GUI is clear and follows usability heuristics
* Delay between mouse input....
* Data is saved in standard format
* All data files are timestamped
* Set-up calibration
* Mouse identification
* Data can be saved to file by the user
* User can start recording

**Medium**

* GUI shows relevant experimental output....
* Help and documentation is available for the system
* System displays output in real time
* Code is well documented
* Software is easy to maintain
* User can stop recording using keyboard as input device

**Low:**

* There is a data buffer for mouse inputs
* What graphs to display can be selected by the user
* System performs well over longer experiments
* User can set timer on recording via user interface
* User can view object movement speed, forward and sideways
* User can view object movement on a 2D map

1. **User stories**
2. **Sprint planning**

**Sprint 1:**

* Calibration GUI
* Main GUI
* Mouse calibration
* User can start recording
* There is a python client for network triggering
* Implement trigger solution

**Sprint 2:**

* All data is timestamped at certain interval
* Data can be saved
* User can view object movement on 2D-map
* User can view object movement speed (thrust, sideways, yaw)
* User can choose between data blocks to show

**Sprint 3:**

* Setup calibration
* (User can set timer on recording)

**Sprint 4:**

* (User can stop recording using keyboard as input device)
* What plots to display can be selected by user in a drop down menu
* Data buffer
* User can set port and hostname for trigger server via the GUI
* User manual
* (User can set save-path)
* GUI settings is saved upon clean shutdown

**Reserve week**