Rose-Hulman Institute of Technology

University of Wisconsin–Madison | Department of Computer Sciences
Human-Computer Interaction Laboratory

Milestone1

Trey Cahill Katie Greenwald Samad Jawaid Kevin Risden

December 1, 2011

Contents

1 Introduction

The Human-Computer Interaction Lab at the University of Wisconsin-Madison wants a web-based system to better manage the scheduling of participants for their studies. These studies range from one-on-one experiments to group interactions, and many of them involve the robot used by the lab. Currently, each researcher arranges studies independently via email and is responsible for scheduling rooms, avoiding conflicts, and notifying participants of changes; unifying this information onto one system simplifies all of these tasks. To the client, the most important benefit of a unified system is the ability for participants to easily browse all available experiments, which is not possible over email. However, a variety of other functionality should be integrated into this utility to take advantage of the unity of information; most notable is recognizing room conflicts when scheduling studies, since the lab has only one robot and it cannot be moved.[1]

Project information will be documented as follows: Milestone 1 provides an overview of the project, from client background to key features and requirements. Milestone 2 covers the behaviour of the system, including use cases and data flow diagrams. Milestone 3 details constraints, back-end requirements, and elaborates upon the user interface. Testing and maintenance information can be found in Milestone 4. Milestone 5 will include usability data and interface re-design related to such data.

2 Client Background

The client is the Human-Computer Interaction Lab at the University of Wisconsin-Madison. Their research focus is the on the way humans perceive computers, and how this perception influences their actions. The main goal is to learn about this interaction through making hypotheses, experimenting, analysing the data, and then publishing papers on the results. They draw the participants for their experiments from a wide range of people, usually ranging from 18-65 years of age and from diverse technical backgrounds. As such, any system they use must be designed for all levels of technical competency.

3 Requirements

3.1 Usability

- For anyone who has used a web application before, there should be near-zero training time required before being able use this system. (Of course, all web applications are at least slightly different to use, so the training time cannot be zero.)
- For anyone who has never used a web application before, there should be minimal training time (a maximum of ten minutes) required before being able to use this system thanks to the on-screen help in the form of a walk-through. This assumes that the user possesses basic computer literacy. Otherwise, it may take the user significantly longer to learn this system.
- No more than half of the time required to actually visit, email, or call a researcher to schedule an appointment will be needed to use this system.
- The new system shall be judged by 99% of the user community to be at least as useful as the existing system and by 90% of the user community to be at least as useful as competing state of the art system.
- On-screen help will be an accessible option for struggling users.
- There will be no drastic interface changes from existing, accepted web application paradigms, so there will be no more than a slight learning curve for the user.

3.2 Performance

- The average response time for the next screen to appear after the user has selected an option that will change the screen will be thirty milliseconds and the maximum will be sixty milliseconds, not including network latency that cannot be controlled.
- A minimum of one page load per second will be supported, again not including network latency that cannot be controlled.
- Each individual session will support one concurrent user, while the overall system will support up to twenty users using the web application in parallel. The client specified the number of twenty. In the rare case that more than twenty users attempt to use the web application in parallel, everyone will be able to do so, but the ideal performance estimates will be invalidated.

3.3 Reliability

- The system will be up 99.99% of the time (less than one hour of downtime per year), independent of the maintenance crew taking it down up to twice a year for eight to ten hours at a time.
- On average, failures will not occur within three months of one another.
- Due to innovative remote diagnostics, updates, and repairs, 90% of all system failures will be able to be repaired within five minutes and 99.9% of all failures will be able to be repaired within one hour, depending on the maintenance crew.
- All appointments will be accurate to the nearest minute 100% of the time.
- Appointments will be successfully scheduled 99.99% of the time, and in the case a scheduling fails, the user will be able to try again 100% of the time.
- There will be no more than one bug per thousand lines of code.
- There will not be any critical or significant bugs. However, there will be up to one minor bug per thousand lines of code. Minor is defined as not affecting the usability of the system or scheduling data.
- It has been assumed that the system will be properly updated and patched at all times by the maintenance crew. Also, system settings, like date and time, have been assumed to always be correct.

3.4 Supportability

While the development team will not be involved with supporting the system once it is handed over to the customer, the system will still have some support. The team will create documentation throughout the development process. Also, the customer will receive all source code when the system is handed over. In order to facilitate supportability for the customer, the system will be written in Python and use PostgreSQL, both with which the current customer is familiar. With the code in a familiar language and the documentation, the customer should be able to fully support the system by themselves.

3.5 Hardware and Software Interfaces

3.5.1 Software Interfaces

The software required for the system will be an operating system of Red Hat Enterprise Linux Server 6.1. Also, the software will be written in Python and use PostgreSQL database management system.

3.5.2 Hardware Interfaces

There are no hardware specifications that we must conform to since we are writing a hardware independent web application.

3.6 Documentation

As part of the usability requirements, the system we provide should be intuitive enough that the end user should have no difficulties navigating the system. However, the client prefers that we provide tool-tip type documentation for the entry forms. This documentation should provide clear, concise direction for the user when entering their information. On the back-end, similar tool-tip documentation will provide the administrator with the guidance to complete their tasks such as modifying experiments or researchers. Additionally, documentation for the installation will be provided in the form of step-by-step instructions as detailed in the installation requirements section. Finally, developer documentation will be in the source code. There will be documentation relating to the use of each class and public method.

3.7 Installation

Since our client is technically proficient, they prefer either step-by-step installation instructions or an install script. Installation instructions are typically more robust than an install script so we will be providing installation instructions with our solution. The installation instructions must be detailed enough that our client can install the web application on the machine provided by them. We have replicated to the best of our abilities the platform on which the web application will be deployed to be able to provide proper installation instructions.

3.8 Legal/Licensing

The client has specified that there are no legal or licensing requirements for this project. The code will be released under an open source license that is yet to be determined. The code is hosted publicly on GitHub and is freely available. The users will have to agree to the terms of the client when creating an account but these have not been finalized yet.

3.9 Reliability

- The system will be up 99.99% of the time (less than one hour of downtime per year), independent of the maintenance crew taking it down up to twice a year for eight to ten hours at a time.
- On average, failures will not occur within three months of one another.
- Due to innovative remote diagnostics, updates, and repairs, 90% of all system failures will be able to be repaired within five minutes and 99.9% of all failures will be able to be repaired within one hour, depending on the maintenance crew.
- All appointments will be accurate to the nearest minute 100% of the time.
- Appointments will be successfully scheduled 99.99% of the time, and in the case a scheduling fails, the user will be able to try again 100% of the time.
- There will be no more than one bug per thousand lines of code.
- There will not be any critical or significant bugs. However, there will be up to one minor bug per thousand lines of code. Minor is defined as not affecting the usability of the system or scheduling data.

- It has been assumed that the system will be properly updated and patched at all times by the maintenance crew. Also, system settings, like date and time, have been assumed to always be correct.

3.10 Usability

- For anyone who has used a web application before, there should be near-zero training time required before being able use this system. (Of course, all web applications are at least slightly different to use, so the training time cannot be zero.)
- For anyone who has never used a web application before, there should be minimal training time (a maximum of ten minutes) required before being able to use this system thanks to the on-screen help in the form of a walk-through. This assumes that the user possesses basic computer literacy. Otherwise, it may take the user significantly longer to learn this system.
- No more than half of the time required to actually visit, email, or call a researcher to schedule an appointment will be needed to use this system.
- The new system shall be judged by 99% of the user community to be at least as useful as the existing system and by 90% of the user community to be at least as useful as competing state of the art system.
- On-screen help will be an accessible option for struggling users.
- There will be no drastic interface changes from existing, accepted web application paradigms, so there will be no more than a slight learning curve for the user.

3.11 Performance

- The average response time for the next screen to appear after the user has selected an option that will change the screen will be thirty milliseconds and the maximum will be sixty milliseconds, not including network latency that cannot be controlled.
- A minimum of one page load per second will be supported, again not including network latency that cannot be controlled.
- Each individual session will support one concurrent user, while the overall system will support up to twenty users using the web application in parallel. The client specified the number of twenty. In the rare case that more than twenty users attempt to use the web application in parallel, everyone will be able to do so, but the ideal performance estimates will be invalidated.

3.12 Supportability

While the development team will not be involved with supporting the system once it is handed over to the customer, the system will still have some support. The team will create documentation throughout the development process. Also, the customer will receive all source code when the system is handed over. In order to facilitate supportability for the customer, the system will be written in Python and use PostgreSQL, both with which the current customer is familiar. With the code in a familiar language and the documentation, the customer should be able to fully support the system by themselves.

4 Use Cases

4.1 General Behaviour

Every page on the website possesses a "login/logout/create account" button. If the user is logged in, follow use case "Logout". Otherwise, follow use case "Login". In either case, unless noted otherwise, upon completion of that use case, the system will return to the page the button was clicked from. If that page had user-entered fields, they will be in the same state they were when the user clicked the button. A common exception is when a researcher or admin account logs out of a researcher or admin page, in which case they will be returned to the homepage; unauthenticated users cannot view or use researcher-only features.

Every page also possesses a button to return to the system homepage. This will exit any use case they are currently following and discard any temporarily stored information related to that use case, such as entered fields or selected options.

Whenever a table is displayed, there is some concern related to its size; however, due to the low expected number of experiments and participants per experiment, any handling of large tables will be outsourced to the browser (scroll bars being most common). Because the whole table will be loaded at once, browser search functionality is also sufficient to handle most searching needs; tables that can be otherwise sorted, filtered, or searched will be specifically noted.

4.2 Authentication Use Cases

1. **Name: Login**
   (a) Brief Description: User logs in.
   (b) Actors: User
   (c) Basic Flow:
      i. User clicks the "login/logout/create account" button from any page.
      ii. User prompted for Email and Password via text boxes.
      iii. The system sends their login information to the database. [A2] [A3] [A4]
      iv. System displays a message confirming successful login. The user is now logged in.
      v. After 10 seconds or when the user clicks a link to do so immediately, the user is navigated out of the login use case as specified in General Behavior.
   (d) Alternate Flows:
      A1 User navigates elsewhere on the website, through their browser or the "home" button. Unless the page they attempt to visit requires authentication, this simply drops them out of the use case.
      A2 User entered an email that the database did not recognize. Run use case "Account Creation".
      A3 User entered an email recognized by the database but not the password associated with it. Return to email/password entry, displaying the message "Incorrect password, please retry."
      A4 System fails to connect to database. Display the message "Database unavailable; we are sorry for the inconvenience. Please try again later." Then return the user to the page they entered the use case from.
   (e) Pre-conditions:
      i. System is functional.

ii. User is not logged in.

iii. User has already created an account.

(f) Post-conditions:

i. User is logged in, or cancelled login process.

(g) Special Requirements:

i. N/A

(h) Feature Mapping:

i. Levels of Authentication

ii. Accounts

2. **Name: Logout**

(a) Brief Description: User logs out.

(b) Actors: User

(c) Basic Flow:

i. User clicks the "login/logout/create account" button from any page.

ii. System displays a message confirming successful logout. The user is now logged out.

iii. After 10 seconds or when the user clicks a link to do so immediately, the user is navigated out of the logout use case as specified in General Behaviour.

(d) Alternate Flows:

A1 User entered this use case from a page not available while logged out (appointment confirmation or researcher interfaces). The system will return them to the homepage unless otherwise specified.

(e) Pre-conditions:

i. System is functional.

ii. User is logged in.

(f) Post-conditions:

i. User is logged out.

(g) Special Requirements:

i. N/A

(h) Feature Mapping:

i. Levels of Authentication

ii. Accounts

3. **Name: Create Account**

(a) Brief Description: User creates an account.

(b) Actors: User

(c) Basic Flow:

i. User clicks the "login/logout/create account" button from any page.

ii. The system navigates the user to the login page.

iii. User clicks the "create account" button from the login page.

iv. The system navigates the user to the account creation page.

v. User prompted for Email, Name, Phone, Password, and Confirm Password via text boxes.

vi. User clicks "Submit" button. [A2] [A3] [A4]

vii. The system sends the entered information to the database. [A5] [A6]

viii. The system sends an email to the entered email address. [A7]

ix. System displays a message confirming successful account creation. The user is now logged in.

x. After 10 seconds or when the user clicks a link to do so immediately, the user is navigated out of the create account use case as specified in General Behaviour.

(d) Alternate Flows:

A1 User navigates elsewhere on the website, through their browser or the "home" button. Unless the page they attempt to visit requires authentication, this simply drops them out of the use case.

A2 User clicked "Submit" before filling in all fields on the account creation page. System does not leave the page, and displays the message "All fields must be completed to continue."

A3 User entered a user name, email, or password that does not meet requirements. See "special requirements". System does not leave the page, and displays the message "Please check guidelines for account creation, one or more fields were not acceptable."

A4 User entered different text in the Password and Confirm Password fields on the account creation page. System does not leave the page, and displays the message "Password confirmation failed; please re-type it."

A5 System fails to connect to database. Display the message "Database unavailable; we are sorry for the inconvenience. Please try again later." Then return the user to the page they entered the use case from.

A6 User entered an email already present in the database on the account creation page. System does not leave the page, and displays the message "Email already registered."

A7 System fails to send an email to the entered address. System does not leave the page, and displays the message "Invalid email, please re-type."

(e) Pre-conditions:

i. System is functional.

ii. User is not logged in.

(f) Post-conditions:

i. User is logged in with their new account, or cancelled account creation process.

(g) Special Requirements:

i. Emails must be of the form <name>@<domain>. They are checked for validity when the system attempts to send to them.

ii. Names cannot include special characters other than . , '

iii. Passwords must be at least six characters, and must have at least two of the following; letters, numbers, special characters.

(h) Feature Mapping:

i. Levels of Authentication

ii. Accounts

4.3 Appointments

1. **Name: Select Experiment**
   (a) Brief Description: Participant views and selects experiment to join
   (b) Actors: Participant (henceforth "user")
   (c) Basic Flow:
      i. User can sort or filter experiment table by date, time, and location.
      ii. User clicks an experiment. The system navigates them to that experiment's page.
      iii. Experiment page: Each experiment has a webpage with its name, description, and a list of timeslots, as well as a button to join the experiment.
      iv. User reads experiment description and required qualifications.
      v. User can sort or filter timeslot list.
      vi. User clicks "join experiment" or a timeslot button. This takes them to use case "sign up for experiment".
   (d) Alternate Flows:
      A1 User decides to view a different experiment by navigating with their browser or clicking a button on any page. They are returned to the homepage.
   (e) Pre-conditions:
      i. System is functional.
      ii. There is at least one experiment currently offered.
   (f) Post-conditions:
      i. User has clicked "join experiment" or a timeslot button for some experiment.
   (g) Feature mapping:
      i. Browse Experiments
      ii. Filter Experiments
2. **Name: Sign up for Experiment**
   (a) Brief Description: Participant enters data and confirms appointment.
   (b) Actors: Participant (henceforth "user")
   (c) Basic Flow:
      i. If the user is not currently logged in, run use case "login". They must be logged in to continue.
      ii. Confirmation Page: This page is available only while logged in. It displays experiment name, required qualifications, and a check box for the user to verify they meet those qualifications. There is a list of timeslots. There is a "Confirm Appointment" button. [A2]
      iii. The user selects a timeslot (or lets the system do it for them if they did so in "Select Experiment") and checks the check box. [A2]
      iv. The user clicks the "Confirm Appointment" button. [A1] [A2]

v. If there are no problems with the entered data, the system returns the user to the homepage and displays a message informing them of their successful registration. The system will also send an email containing the experiment and timeslot information to the email account used to register.

(d) Alternate Flows:

A1 User attempts to "Confirm Appointment" before selecting a timeslot or checking the check box.

A. The system will return them to the confirmation page and inform them of what still needs to be done.

A2 User logs out while on the confirmation page. The system will return them to the experiment page.

(e) Pre-conditions:

i. System is functional.

ii. User has selected an experiment via the "Select Experiment" use case.

iii. The selected experiment has at least one viable timeslot.

(f) Post-conditions:

i. Database has added appointment to user and experiment data.

ii. (Side effect) user is authenticated.

(g) Special Requirements:

i. N/A

(h) Feature Mapping:

i. Participant Schedule Experiment

ii. Notify Participant when Creating Appointment

iii. Prevent Scheduling Conflicts (Participant)

4.4 Experiment Management

1. **Name: Add Experiment**

(a) Brief Description: Experiments can be created by Administrators and Researchers

(b) Actors: Administrators and Researchers

(c) Basic Flow: (user can cancel at any time and follow A1)

i. User must click on Add New Experiment link from the Administration "home" page

ii. System will display a screen with text boxes to enter experiment name, description, and qualifications, multiple date/time choosers for the schedule times, and a drop down list to specify the length of the schedule slots

iii. User must enter the experiment information for name, description, qualifications, and schedule slots

iv. User can then begin setting up the schedule times by choosing date, begin, and end time for each slot they want to run the experiment

v. User then must save the experiment by clicking the Save button

vi. System will then save the experiment to persistent storage and provide the user with confirmation that the experiment was created successfully and redirect user to all experiment view [A2]

(d) Alternate Flows:

A1 User cancels out of creating an experiment

A2 Saving an experiment fails

(e) Pre-conditions:

i. User is an Administrator and/or a Researcher and has authenticated

(f) Post-conditions:

i. System will have recorded the experiment or the system will notify the user why the creation of the experiment failed

(g) Special Requirements:

i. End times for each slot must be after begin times.

(h) Feature mapping:

i. Add Experiment

ii. Prevent Scheduling Conflicts (Administrator)

2. **Name: Modify Experiment**

(a) Brief Description: Experiments can be modified by Administrators and Researchers to change all assets of the experiment

(b) Actors: Administrators and Researchers

(c) Basic Flow: (user can cancel at any time and follow A1)

i. System will display experiment fields (name, description, qualifications, schedule time, schedule slots, and participant list)

ii. User will click on desired field to modify [A3]

iii. System will allow field that user chooses to be editable in line

iv. User will then change field as desired and click away from the field or save when finished

v. System will update the database with the modified experiment information [A2] [A3]

(d) Alternate Flows:

A1 User cancels out of creating an experiment. System will return user to the page where user came from

A2 Saving an experiment fails

A3 User deletes an experiment. System will remove experiment from database after user confirmation and display a message to the user indicating this was successful

(e) Pre-conditions:

i. User is an Administrator and/or a Researcher and has authenticated

ii. User chose experiment through one of the experiment views

(f) Post-conditions:

i. System will have recorded the modifications to the experiment or the system will notify the user why the modification of the experiment failed

(g) Special Requirements:

i. End times for each slot must be after begin times.

(h) Feature mapping:

i. Modify Experiment

ii. Remove Experiments

iii. Prevent Scheduling Conflicts (Administrator)

4.5 Reports

1. **Name: List Experiment Participants**

(a) Brief description: Researcher logs in and views a list of all participants for a selected experiment.

(b) Actors: Researcher

(c) Basic flow:

i. (1) Researcher logs in, using the Login use case with a Researcher account

ii. (2) System displays table of researcher's experiments [A1]

iii. (3) Researcher selects experiment from table

iv. (4) System displays list of all participants for selected experiment

(d) Alternate flows:

A1 Researcher does not own any experiments

A. (2) displays an empty table

B. He cannot proceed past (2) until he creates an experiment or is added to another researcher's

i. Selected experiment has no participants

A. (4) displays an empty table

B. Nothing is displayed in (4) until a participant signs up for the selected experiment

(e) Pre-conditions

i. System is running

ii. System is in ready state

iii. Researcher has account with correct permissions/groups

(f) Post-conditions

i. Researcher knows who is signed up to participate in his selected experiment or there are no experiments/participants

(g) Special Requirements:

i. N/A

(h) Feature mapping:

i. Experiment Participants

2. **Name: Cancel Experiment Appointment**

(a) Brief description: Participant logs in and cancels an appointment.

(b) Actors: Participant (User)

(c) Basic flow

i. (1) Participant logs in

ii. (2) System displays table of participant's appointments [A1]

iii. (3) Participant selects appointment from table

iv. (4) System displays details for selected appointment

v. (5) Participant selects cancel

vi. (6) System displays confirmation prompt

vii. (7) Participant selects confirm: appointment is marked cancelled and system returns to (2) with an affirmation message

viii. (8) Participant selects keep appointment: system returns to (4)

(d) Alternate flows

A1 Participant has no appointments

A. (2) displays an empty table

B. He cannot proceed past (2) until he signs up for an experiment

(e) Pre-conditions

i. System is running

ii. System is in ready state

iii. Participant has account

(f) Post-conditions

i. Participant cancelled selected appointment or participant cancelled operation

ii. Researcher(s) owning said appointment's experiment are notified via email

(g) Special Requirements:

i. N/A

(h) Feature mapping:

i. Cancel Experiment Appointment

ii. Notify Participant Appointment Cancellation Reminder

3. **Name: Report Experiment Participant Lists**

(a) Brief Description: When the user is a researcher, the user will be able to export a CSV file, filed with the Experiment name and participant and times.

(b) Actors: Researcher

(c) Basic Flow:

i. The researcher will check what experiments to export to the CSV file from the list of experiments in the researcher side view

ii. The researcher will click "Export to CSV [A1]

iii. The system will generate a CSV file from the selected experiment displaying the name of the experiment and the names of participants with their times [A2] [A3]

iv. The system will then start the download of the file to the researcher's computer [A4] [A5]

v. When the system has completed 3 and 4, the system will display a message box Export Complete!

vi. The researcher will click "OK or the exit button on the message box

vii. The system will return to the researcher side view.

(d) Alternative Flow:

A1 The researcher did not select any experiment. An error window will appear.

A2 The system encounters an error when pulling data from the database. An error window will appear

A3 The system encounters any error when creating the CSV file. An error window will appear

A4 The system cannot download the file to the researcher's computer. An error window will appear

A5 The researcher will deny the download of the CSV. A message box will appear

A6 The user exits the browser

(e) Preconditions:

i. The researcher must be logged in as a researcher

ii. The system is in the researcher side view

iii. The researcher must already have experiments scheduled

(f) Postconditions:

i. The system is back in the researcher side view

(g) Feature mapping:

i. Export Experiment Participant List

4. **Name: Calendar/List of All Experiments**

(a) Basic Description: The list will show all ongoing experiments and will allow for a user to click and view more information on the experiment

(b) Basic Flow:

i. The system displays all experiments that have not yet occurred [A1]

ii. The user can scroll down the list

iii. The user selects an experiment, as per use case Select Experiment

(c) Alternative Flow:

A1 There are no experiments to display. In this case, there is nothing to show the user, and no experiment can be selected.

(d) Preconditions:

i. The user is on the web page

(e) Postconditions:

i. The system is showing an experiment or the browser is on a new page

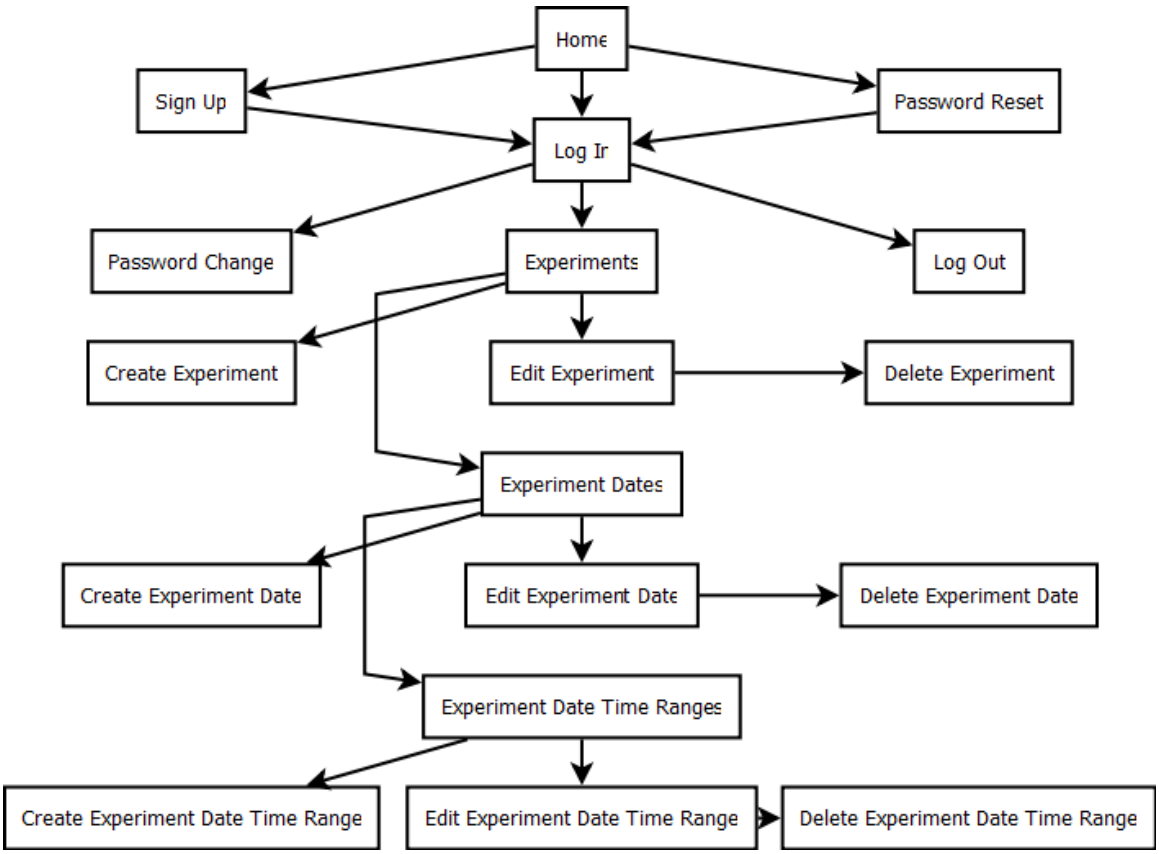
(f) Feature mapping:

i. All calendar Experiments

ii. Browse Experiments

iii. Persistent Experiment Storage

5 Interaction Architecture



6 Who Did What

Who	Section / Part Completed	Task / Comments	Effort
Trey	Document	Put together entire Docment from Parts	.5 hours

7 References

[1] University of Wisconsin-Madison. Human-Computer Interaction Laboratory, 2010.

8 Appendix

Index