Rose-Hulman Institute of Technology

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

Milestone 5

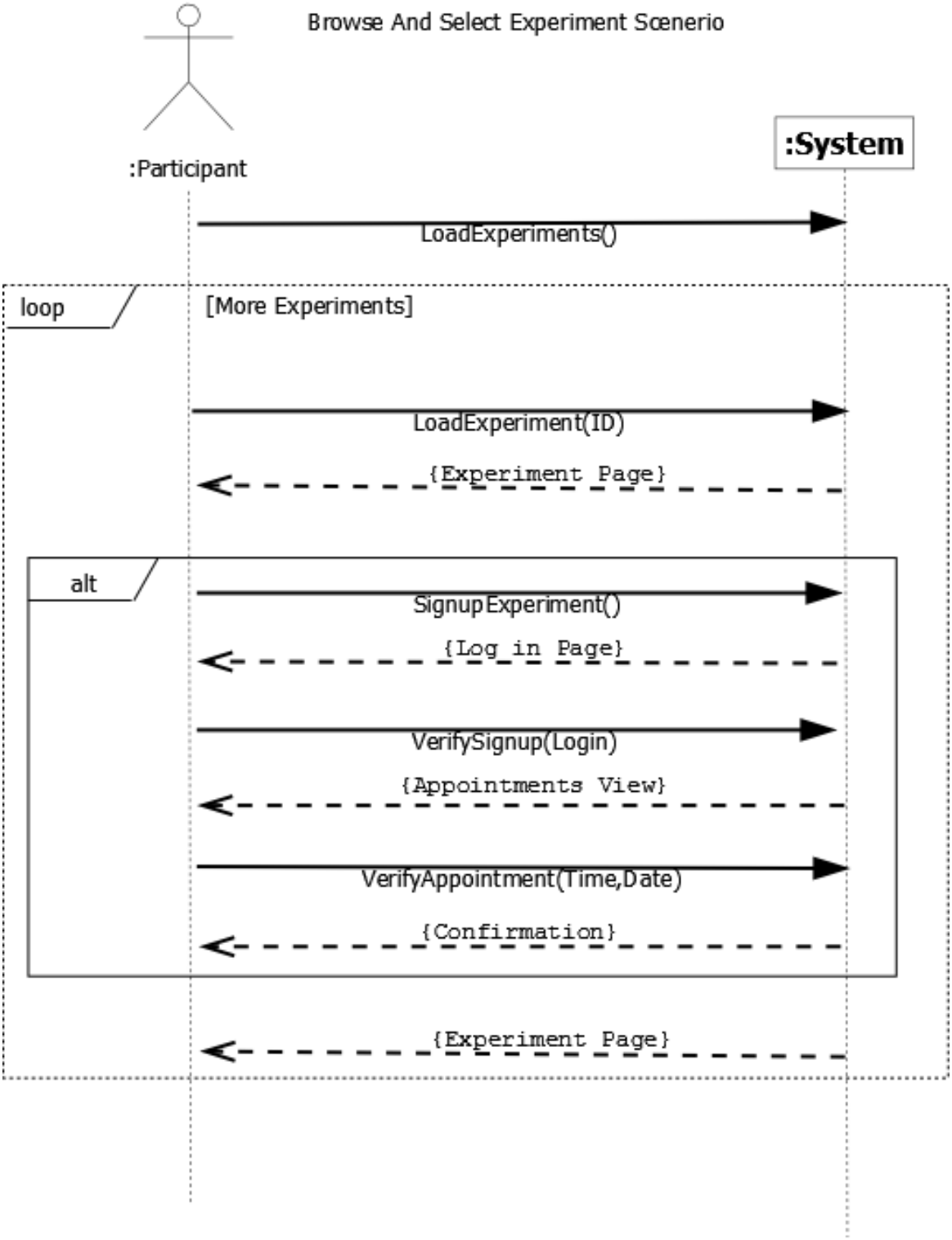
Trey Cahill Katie Greenwald Samad Jawaid Kevin Risden

February 2, 2012

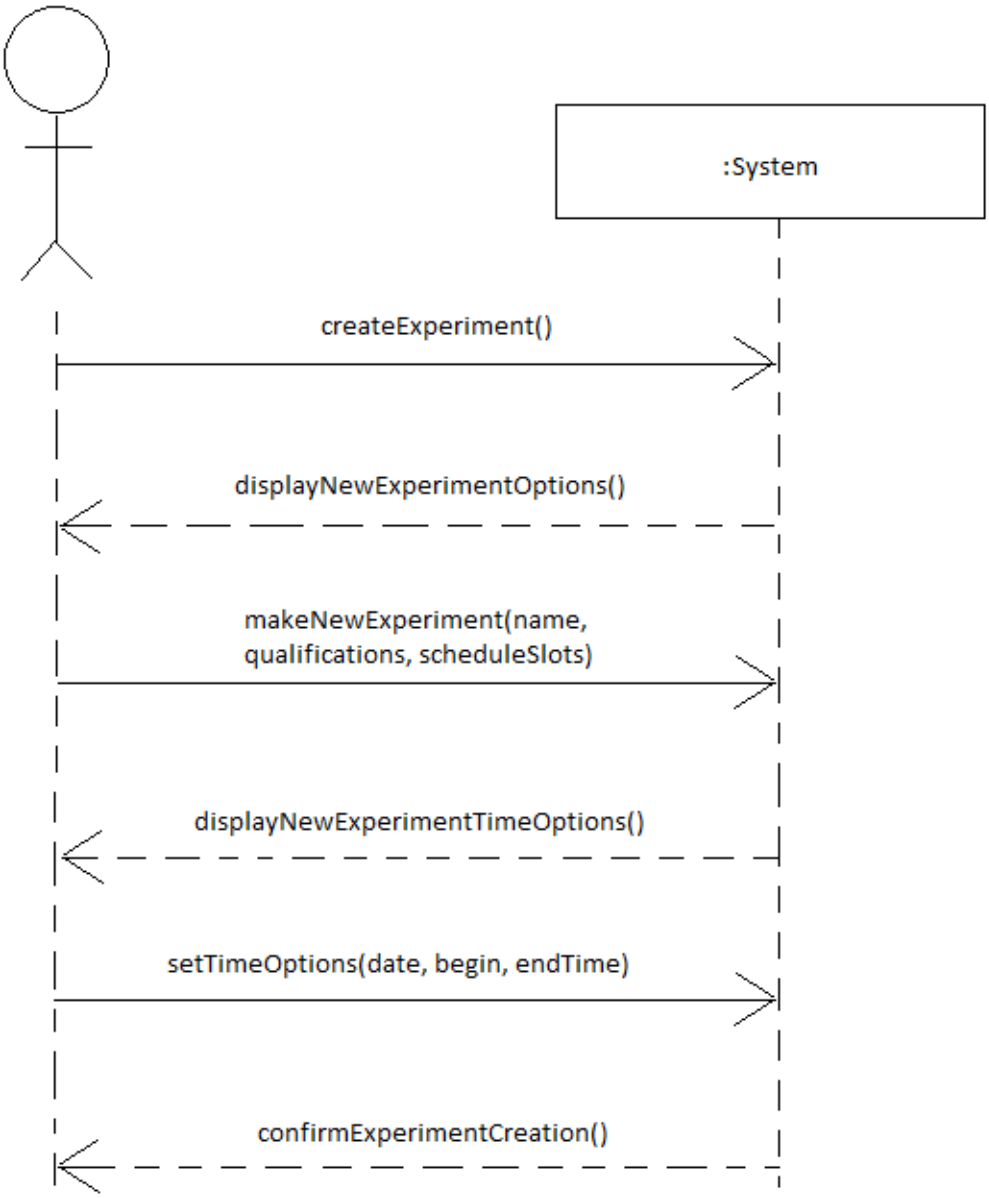
Contents

1 System Sequence Diagrams

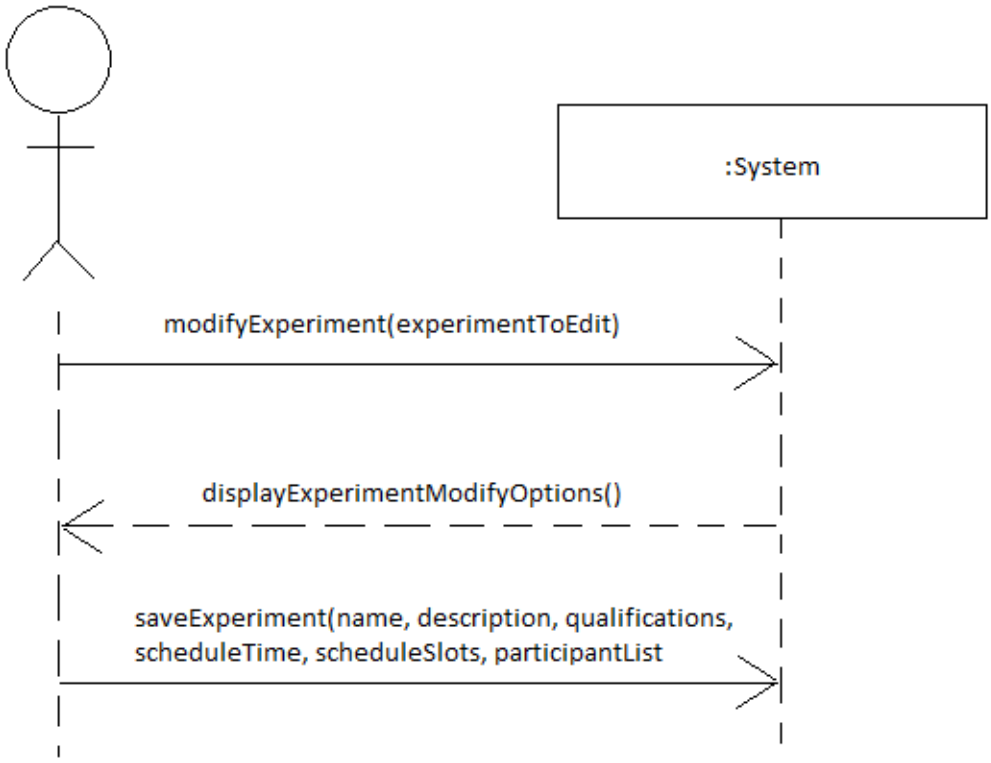
1.1 Browse and Select Experiment



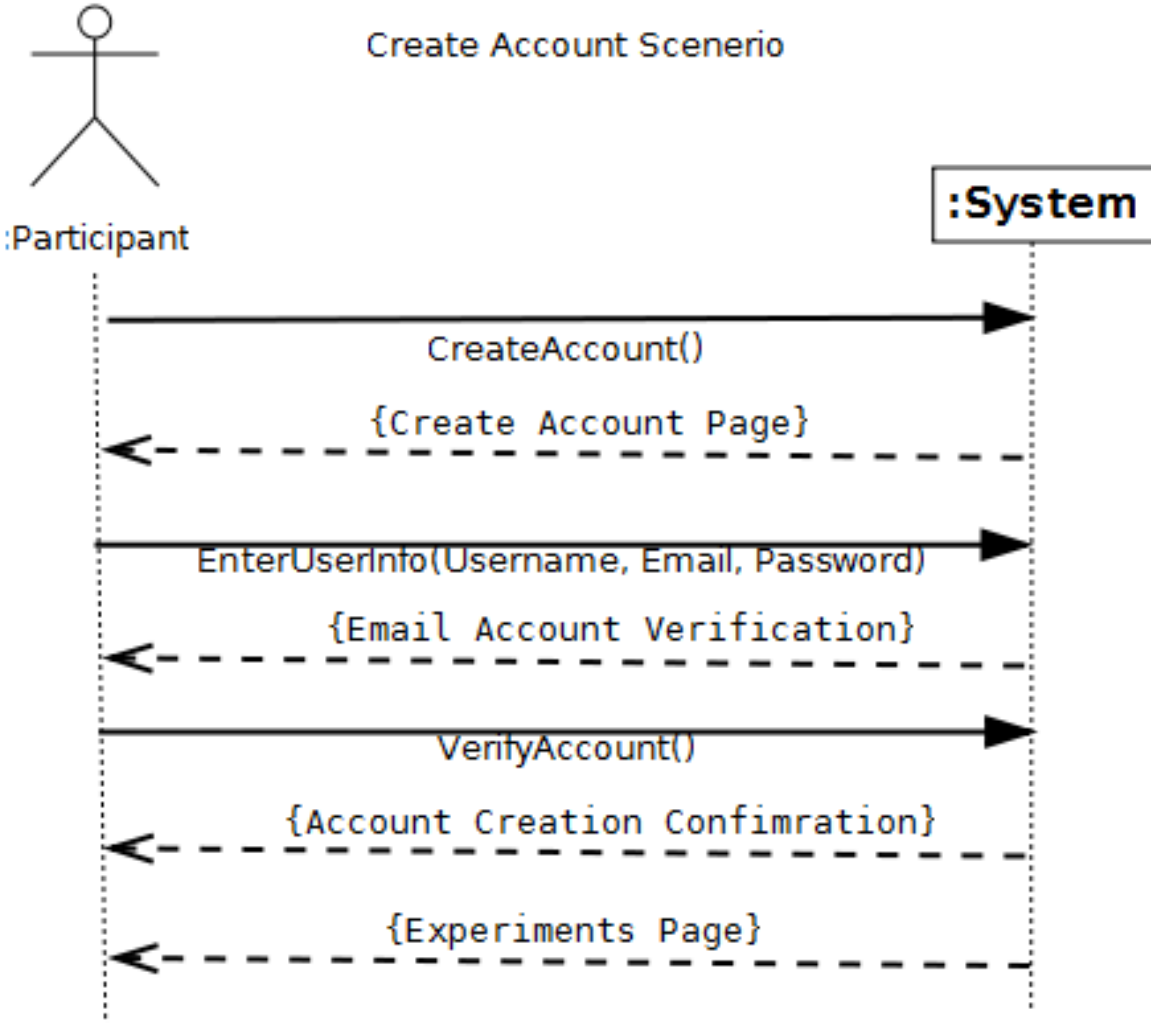
1.2 Create Experiment



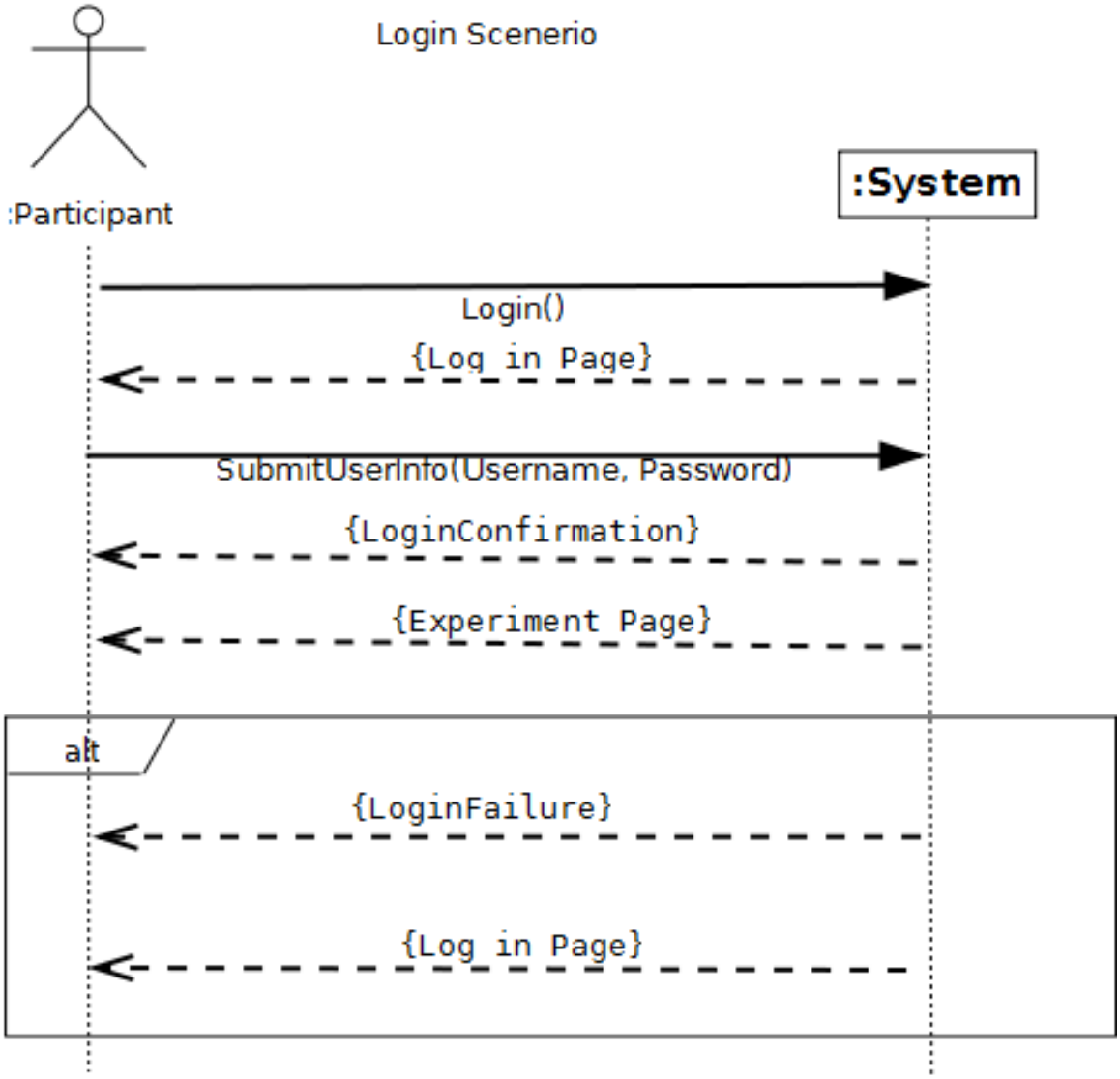
1.3 Modify Experiment



1.4 Create Account



1.5 Login



2 Operations Contracts

This section provides operation contracts for vital operations.

2.1 CreateExperiment

Operation:	CreateExperiment()
Cross References:	Uses Cases: Add Experiment
Preconditions	User is an Administrator and/or a Researcher and has authenticated
Postconditions:	Experiment object will have been created, or an error message will have been displayed

2.2 ModifyExperiment

Operation:	ModifyExperiment(ID)
Cross References:	Uses Cases: Modify Experiment
Preconditions	User is an Administrator and/or a Researcher and has authenticated
Postconditions:	The experiment object will have its fields modified, will have been deleted, or an error message will have been displayed

2.3 LoadExperiments

Operation:	LoadExperiments()
Cross References:	Uses Cases: Select Experiment
Preconditions	The participant has loaded the web page.
Postconditions:	Experiments collection was created (instance creation)

2.4 LoadExperiment

Operation:	LoadExperiment(ID)
Cross References:	Uses Cases: Select Experiment
Preconditions	The participant as clicked on an experiment.
Postconditions:	Experiment was created (instance creation)
	Experiment attributes was loaded into the web page

2.5 Sign Up Experiment

Operation:	SignupExperiment()
Cross References:	Uses Cases: Sign up for Experiment
Preconditions	The participant has clicked on an experiment to sign up for
Postconditions:	LogIn was created (instance creation)
	LogIn.logged became loggedIn (attribute modification)

2.6 Verify Sign up

Operation:	VerifySignup(Login)
Cross References:	Uses Cases: Sign up for Experiment
Preconditions	The participant has logged into or created an account
Postconditions:	this.hasConflict is set to false (attribute modification)

2.7 Verify Appointment

Operation:	VerifyAppointment(Time,Date)
Cross References:	Uses Cases: Sign up for Experiment
Preconditions	The participant has selected a time and date slot.
Postconditions:	experiment.slots has been modified (attribute modification)
	Database is updated

2.8 Create Account

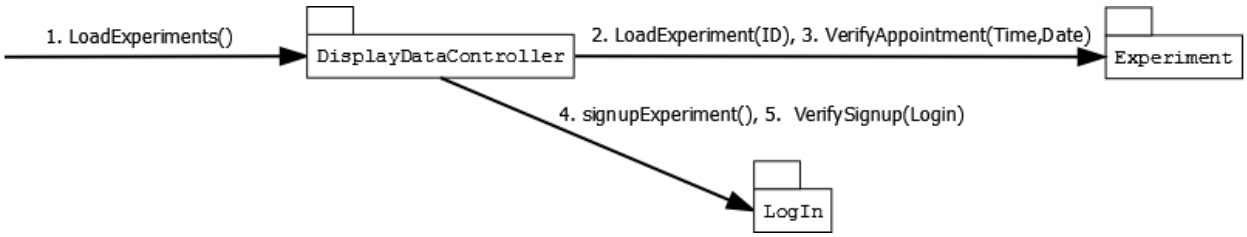
Operation:	CreateAccount()
Cross References:	Uses Cases: Create Account
Preconditions	None.
Postconditions:	The user has an account.
	A verification email has been sent to the specified email address.
	The user is logged in.

2.9 Login

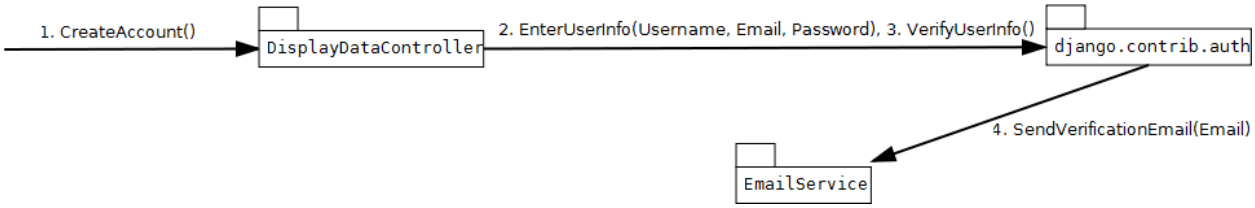
Operation:	Login()
Cross References:	Uses Cases: Login
Preconditions	The user has an account.
Postconditions:	The user is logged in.
	Appropriate rights have been given to the account logged in.

3 Interaction Diagrams

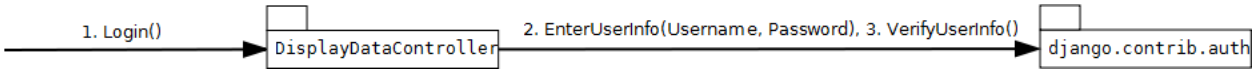
3.1 Sign Up For Experiment



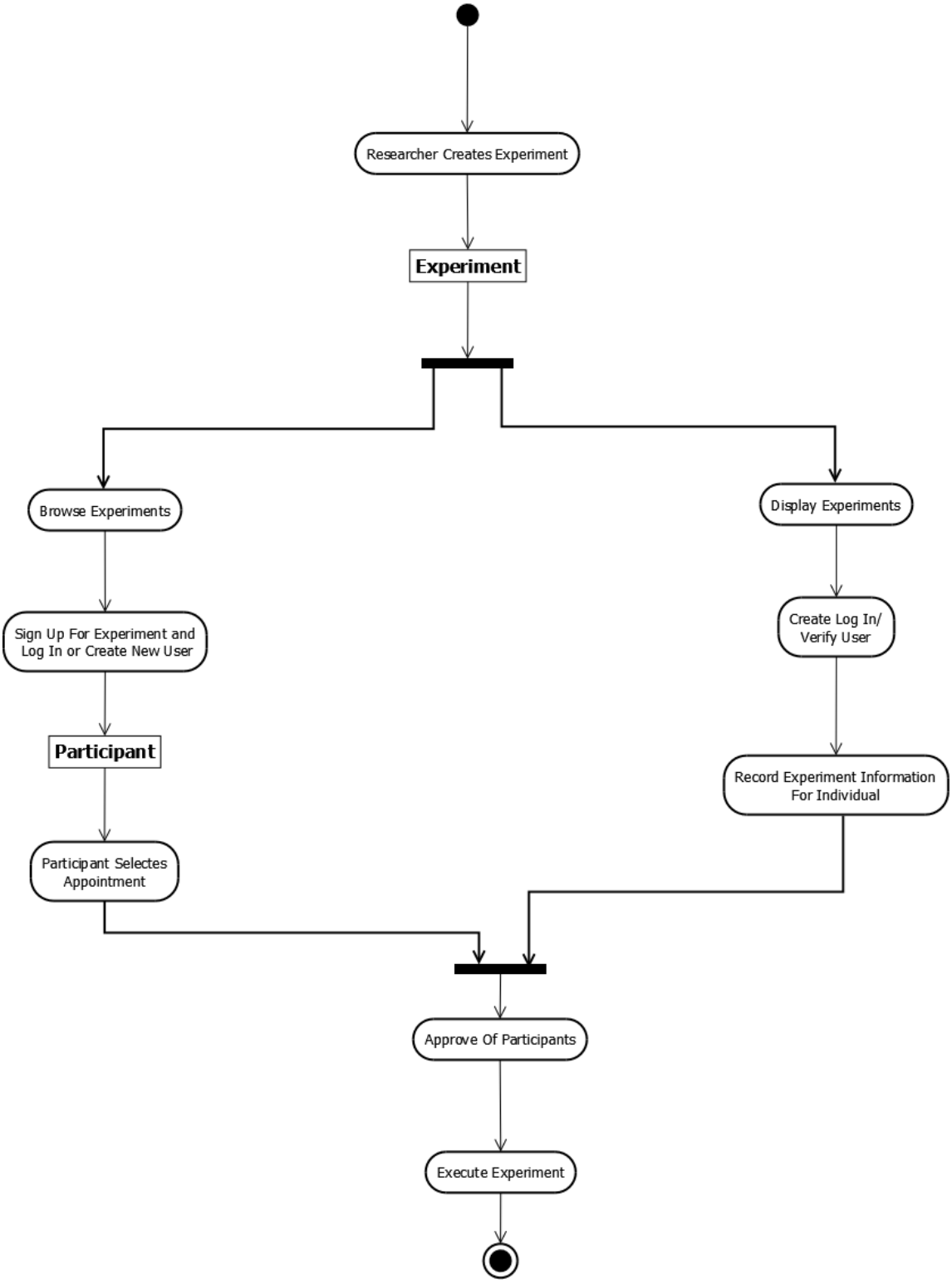
3.2 Create Account



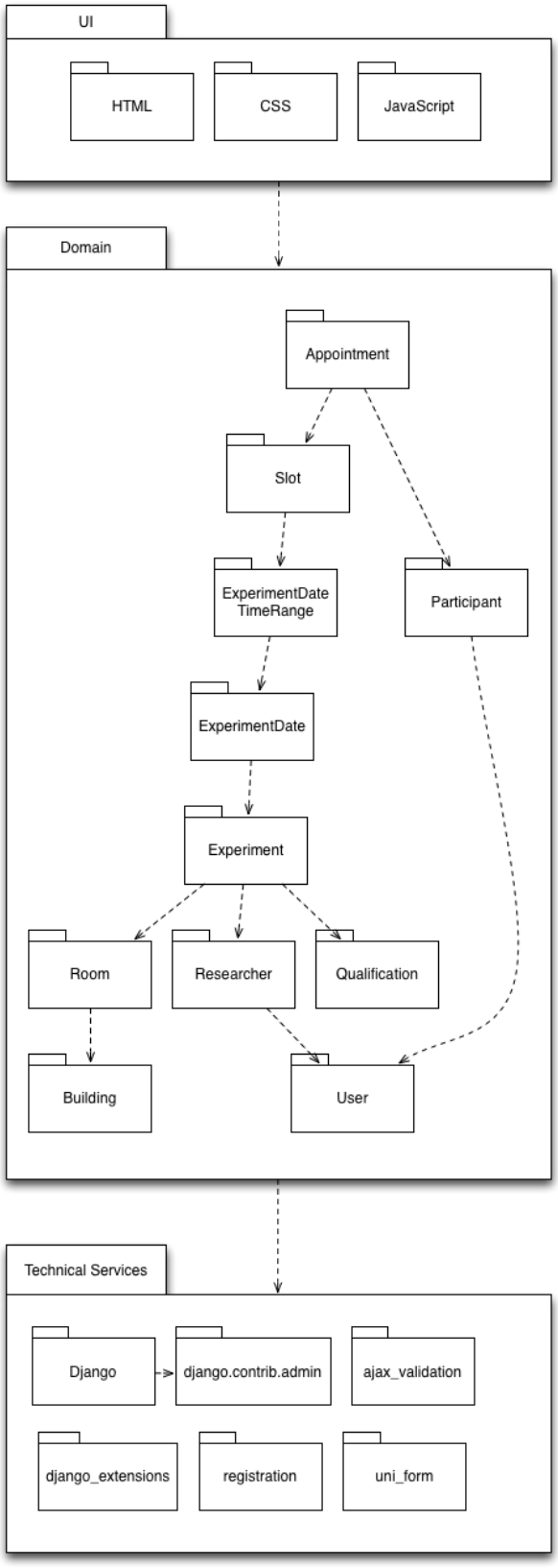
3.3 Login



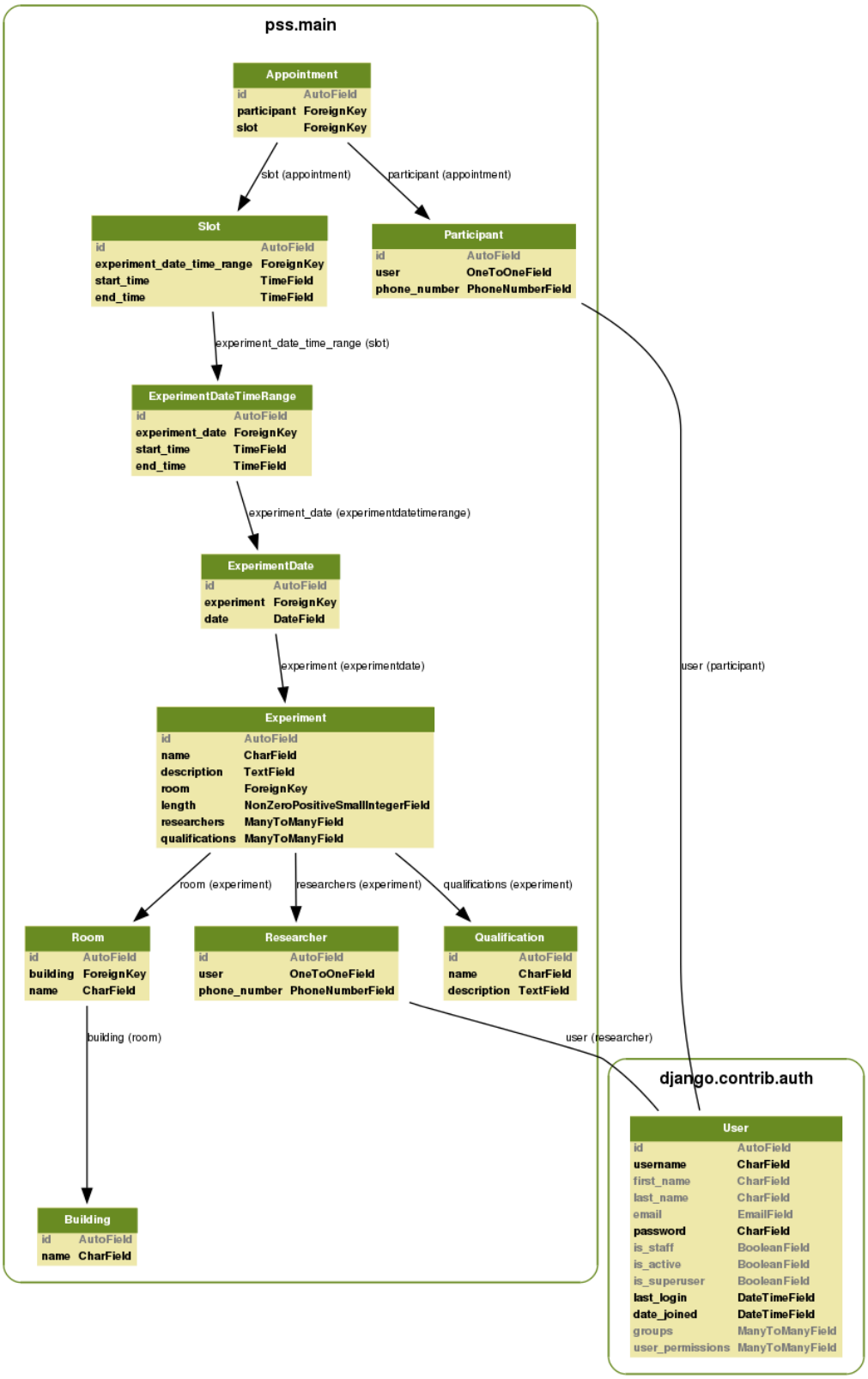
4 Activity Diagram



5 Package Diagram



6 Class Diagram



7 GRASP Principles

7.1 Creator

In our system, there are many objects which must be created. One notable example of this is the creation of participation slots. A date-time range, the time period over which the experiment is being run, can create for itself several slots, or chunks within the time period, which participants can sign up for. Since these slots only pertain to one date-time range, the range is the one to create and manage them.

7.2 Info Expert

For information expert, an example from our system would be the Experiment class. An Experiment has knowledge of ExperimentDate, ExperimentDateTimeRange, Slots, Appointments, Participants, Room, Researchers, and everything else pertaining to an instance of an experiment. When something needs to be done by or to an experiment, that experiment will have all of the necessary information, so it can be relied on to complete those tasks.

7.3 Controller

One nice example of a controller in our system is the URLs. Each URL can be broken up so the system will know what should handle the user's input. For example, in the case of experiments/dates/edit/[an experiment id], the system would turn to the experiment editing code. Each URL our system uses works this way, meaning that all the power of what is displayed on each page is delegated exclusively by the URL handling.

7.4 High Cohesion

High cohesion is achieved in many parts of our code, but an example of where cohesion is extremely strong is in the Administration code. All of our Administration procedures, fields, responsibilities, and all similar objects are located in the administration package, specifically in the django.contrib.admin package as seen in the package diagram in section 4 of this document. By having administration as its own package, it leaves other administration tasks out of other unrelated classes in our system.

7.5 Low Coupling

By separating out objects and responsibilities into our their own packages, we achieve a fairly low coupling. Each separate package contains a responsibility and is connected only to the classes to which must be connected. Our class diagram in section 5, shows how our classes interact with each other and also shows that each class is used only by a class that needs its information. For example, Qualification is only connected to Experiment; if could possibly be connected to participant, but because you will not be able to completely verify the participants qualifications until they show up for the experiment, there is no need to make the connection between Participant and Qualifications.

7.6 Pure Fabrication

Pure Fabrication becomes uses in our system by creating ExperimentDateTimeRange and ExperimentDate. Neither of these two classes are in the domain model, but since they make the code easier to work with and separate out responsibility, increasing cohesion, they become very useful as classes. Should we not have

these two classes Slot or Experiment would have to contain this information, which would decrease cohesion and generally add to the complexity of the Slot or Experiment or both.

7.7 Indirection

The GRASP principle indirection directly relates to our system for how we need to represent experiment dates and the experiment datetime ranges. An experiment must keep track of the dates and time slots for each day that it is offered. We decided that having two intermediate classes, ExperimentDate and ExperimentDateTimeRange, would reduce coupling and ensure easier maintainability of the system. The ExperimentDate class keeps track of the slots that the ExperimentDateTimeRange can generate. This enables the user to enter a time range and the system will then calculate the specific time slots. The Experiment class just has to have the ExperimentDates. This makes the coupling of time slots to experiments cleaner. An alternative would be for the experiment to have a massive list of all the dates and slots that it is offered. This would make it difficult to add or remove slots later and not know which slots are current filled participants.

7.8 Polymorphism

Currently, the Participant Scheduling System requires both researchers and participants. In order to accomplish this, a User class was introduced to provide a standard base class and then the Researcher class was derived from this. This provides our solution with the polymorphism GRASP principle. The other option was to create two separate classes for researcher and user, but then there would be duplicated code. Furthermore, if there needs to be another type of user then it will be simpler to just extend the current User class.

7.9 Protected Variation

In our system, protected variation builds off our decision for polymorphism. The User class enables the protected variation GRASP principle since it protects us from changes in the type of users who need to use the system. If the client comes back with a request for another type of user besides participant and researcher, the system is setup to handle this by just extending the User class. This provides the most elegant solution to the problem since the other option would have been to create the different user classes separately and would make it difficult to extend later.

8 Gang of Four Principles

8.1 Observer

The class views.py is an observer pattern, as in the class it recieves a request, which then completes the request. The views class reacts to what request it is given and then return the correct view for the page to display, with the correct information and checking for correct authorization for each request, so that a particpant cannot create an experiment.

9 Test Cases

9.1 Experiments

9.1.1 List Experiment Participants

Conditions:

A Researcher does not own any experiments

B Selected experiment has no participants

Test Case	Scenario	Description	Cond A	Cond B	Expected Result
1	1	Basic flow	I	I	System displays list of all participants for selected experiment
2	2	Alternate flow: Researcher does not own any experiments	V	N/A	System displays an empty table of experiments
3	3	Alternate flow: Selected experiment has no participants	N/A	V	System displays an empty table of participants

9.1.2 Cancel Experiment Appointment

Conditions:

A Participant selects confirm

B Participant selects cancel

C Participant has no appointments

Test Case	Scenario	Description	Cond A	Cond B	Cond C	Expected Result
1	1	Basic flow: Participant selects confirm	V	I	I	Appointment is marked cancelled and system returns with an affirmation message
2	1	Basic flow: Participant selects cancel	I	V	I	System returns user to page they came from
3	2	Alternate flow: Participant has no appointments	N/A	N/A	V	System displays an empty table of appointments

9.2 Experiment Management

9.2.1 Add Experiment

Conditions:

A Experiment information (includes name, description, qualifications, date/time schedule, and slot length) - Must check each combination

Test Case	Scenario	Description	Cond A	Expected Result
1	1	Basic flow: Administrator enters experiment information	V	Experiment is created and user is notified that creation of experiment was successful
2	2	Basic flow: Administrator enters experiment information	I	Experiment is not created and user is notified of invalid field entry and the corresponding field
3	3	Alternate flow: Administrator tries to save experiment	N/A	System notifies user that the save failed and returns user to Add Experiment page with pre-filled values
4	4	Alternate flow: Administrator clicks cancel	N/A	System returns user to page they came from

9.2.2 Modify Experiment

Conditions:

A Experiment information (includes name, description, qualifications, date/time schedule, and slot length) - Must check each combination

Test Case	Scenario	Description	Cond A	Expected Result
1	1	Basic flow: Administrator enters experiment information	V	Experiment information is updated and user is notified that modification of the experiment was successful
2	2	Basic flow: Administrator enters experiment information	I	Experiment information is not updated and user is notified of invalid field entry and the corresponding field
3	3	Alternate flow: Administrator tries to save experiment	N/A	System notifies user that the save failed and returns user to Add Experiment page with pre-filled values
4	4	Alternate flow: Administrator clicks cancel	N/A	System returns user to page they came from
5	5	Alternate flow: Administrator deletes the experiment	N/A	System notified the user that the experiment has been deleted and returns user to experiment management page

9.3 Authentication

9.3.1 Login

Conditions:

A Email entered and in database

B Password entered and matches email in database

Test Case	Scenario	Description	Cond A	Cond B	Expected Result
1	Any page	Basic flow: User clicks “login/create account” link from any page.	N/A	N/A	System navigates user to login page.
2	Login page	Alternate flow: User clicks “submit” button.	I	N/A	System navigates user to create account page (see test cases for Account Creation).
2	Login page	Alternate flow: User clicks “submit” button.	V	I	System displays message informing user their password is incorrect.
3	Login page	Basic flow: User clicks “submit” button.	V	V	System displays message confirming successful login.
4	Success message	Basic flow: User clicks “return immediately” link.	N/A	N/A	System takes down success message and navigates back to initial page.
5	Success message	Alternate flow: 10 seconds pass after message displayed.	N/A	N/A	System takes down success message and navigates back to initial page.

9.3.2 Logout

Test Case	Scenario	Description	Expected Result
1	Any page	Basic flow: User clicks “logout” link from any page.	System displays message confirming successful logout.
2	Logout message	Basic flow: User clicks “return immediately” link.	System takes down logout message.
3	Logout message	Alternate flow: 10 seconds pass after clicking “logout”.	System takes down logout message.

9.3.3 Create Account

Conditions:

A Email entered and is not in database

i. Emails must be of the form <name>@<domain>

B Password entered and follows guidelines

i. Password guidelines: must be at least six characters, containing at least two of the following character types: letters, numbers, special characters.

C Confirm password entered and matches password

D User name entered and follows guidelines

i. User name guidelines: Names cannot include any special characters other than periods, commas, and apostrophes.

9.4 Appointments

9.4.1 Select Experiment

9.4.2 Sign up for Experiment

Conditions:

A User logged in

B User has clicked check box and selected valid timeslot

Test Case	Scenario	Description	Cond A	Cond B	Cond C	Cond D	Expected Result
1	Any page	Basic flow: User clicks "login/create account" link from any page.	N/A	N/A	N/A	N/A	System navigates user to login page.
2	Login page	Basic flow: User clicks "create account" button.	N/A	N/A	N/A	N/A	System navigates user to create account page.
3	Login page	Alternate flow: User clicks "submit" button.	V	N/A	N/A	N/A	System navigates user to create account page.
4	Create account page	Alternate flow: User clicks "submit" button.	I	N/A	N/A	N/A	System displays message informing user that email is already in use.
5	Create account page	Alternate flow: User clicks "submit" button.	V	I	N/A	N/A	System displays message informing user that password does not meet guidelines.
6	Create account page	Alternate flow: User clicks "submit" button.	V	V	I	N/A	System displays message informing user that password and confirmation do not match.
7	Create account page	Alternate flow: User clicks "submit" button.	V	V	V	I	System displays message informing user that name does not meet guidelines.
8	Create account page	Basic flow: User clicks "submit" button.	V	V	V	V	System displays message confirming successful account creation.
9	Success message	Basic flow: User clicks "return immediately" link.	N/A	N/A	N/A	N/A	System takes down success message and navigates back to initial page.
10	Success message	Alternate flow: 10 seconds pass after message displayed.	N/A	N/A	N/A	N/A	System takes down success message and navigates back to initial page.

Test Case	Scenario	Description	Expected Result
1	Home page	Basic flow: User clicks an experiment from the table.	System navigates user to that experiment's page.
2	Experiment page	Basic flow: User clicks "join experiment".	System navigates to appointment page (see Sign Up for Experiment).
3	Experiment page	Alternate flow: User selects a timeslot.	System navigates to appointment page (see Sign Up for Experiment) with timeslot information.

Test Case	Scenario	Description	Cond A	Cond B	Expected Result
1	Appointment page	Alternate flow: User is on appointment page.	I	N/A	System navigates user to login page.
2	Appointment page	Basic flow: User is on appointment page.	V	N/A	Selected timeslot from experiment page already selected.
3	Appointment page	Alternate flow: User clicks “confirm appointment” button.	V	I	System informs user they must complete the form.
4	Appointment page	Basic flow: User clicks “confirm appointment” button.	V	V	System displays message confirming successful appointment.
5	Success message	Basic flow: User clicks “return immediately” link.	N/A	N/A	System takes down success message and navigates back to home page.
6	Success message	Alternate flow: 10 seconds pass after message displayed.	N/A	N/A	System takes down success message and navigates back to home page.

9.5 Reports

9.5.1 Export Participants

Test Case	Scenario	Description	Precondition	Expected Results
1	Export Experiment Participant List	Basic Flow: The researcher clicks Export to CSV	There are no experiments checked	Error box asking the user to select an experiment
2	Export Experiment Participant List	Basic Flow: The researcher clicks Export to CSV	There are experiments checked	The system generates a CSV file
3	Export Experiment Participant List	Basic Flow: The system starts the download of the file	The CSV creation succeeded	The file downloads and a message box is displayed “Export Complete”
4	Export Experiment Participant List	Alternate Flow: An Error occurs when pulling from the database	The researcher has selected experiments and Clicked Export CSV	The system displays an error message
5	Export Experiment Participant List	Alternate Flow: The system cannot download the file to the researchers computer	The CSV creation succeeded and the download has started	The system displays an error message
6	Export Experiment Participant List	Alternate Flow: The researcher denies the download of the CSV	The CSV creation succeeded and the download has started	The system displays a message box

10 References

11 Appendix