Rose-Hulman Institute of Technology

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

Milestone 4

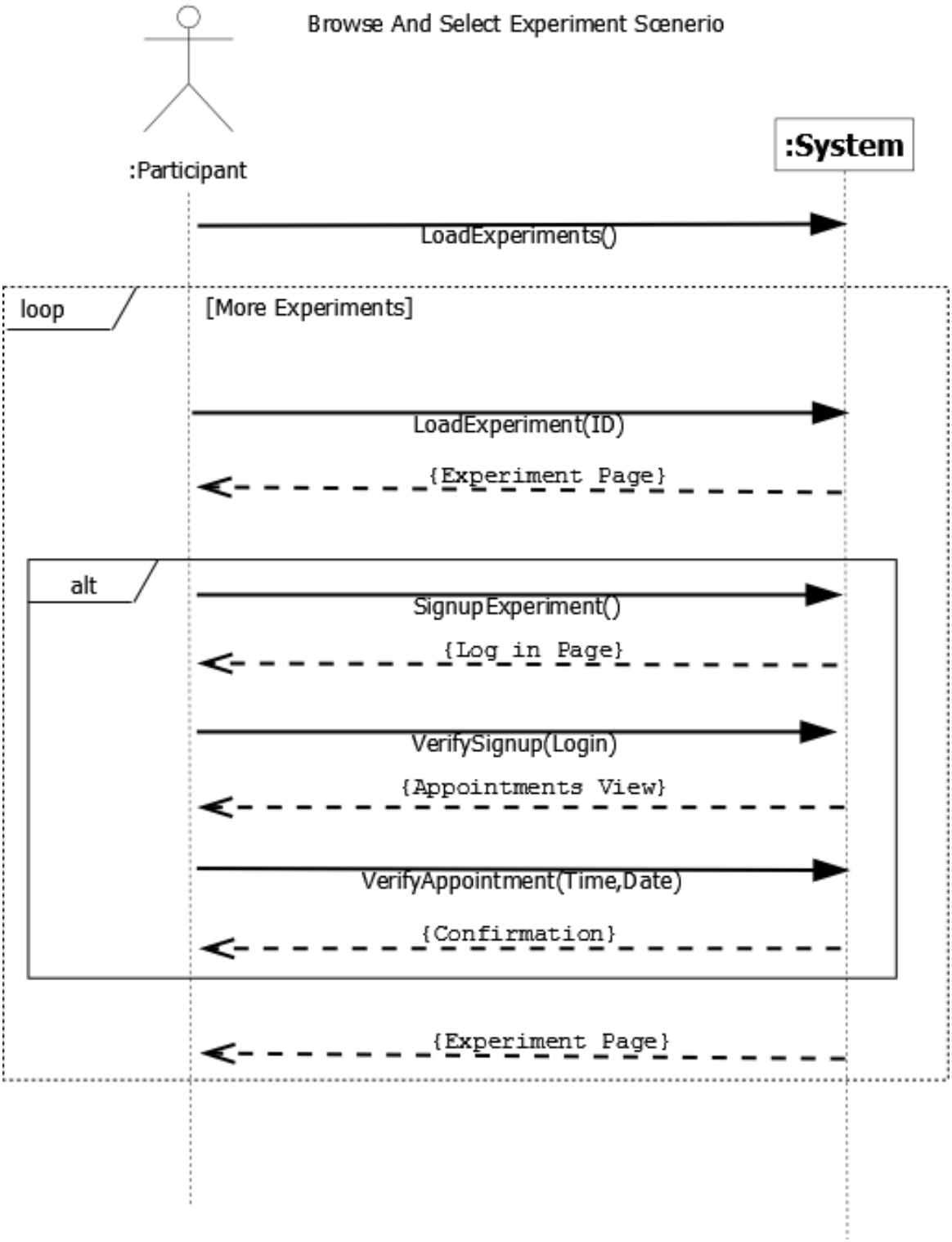
Trey Cahill Katie Greenwald Samad Jawaid Kevin Risden

January 23, 2012

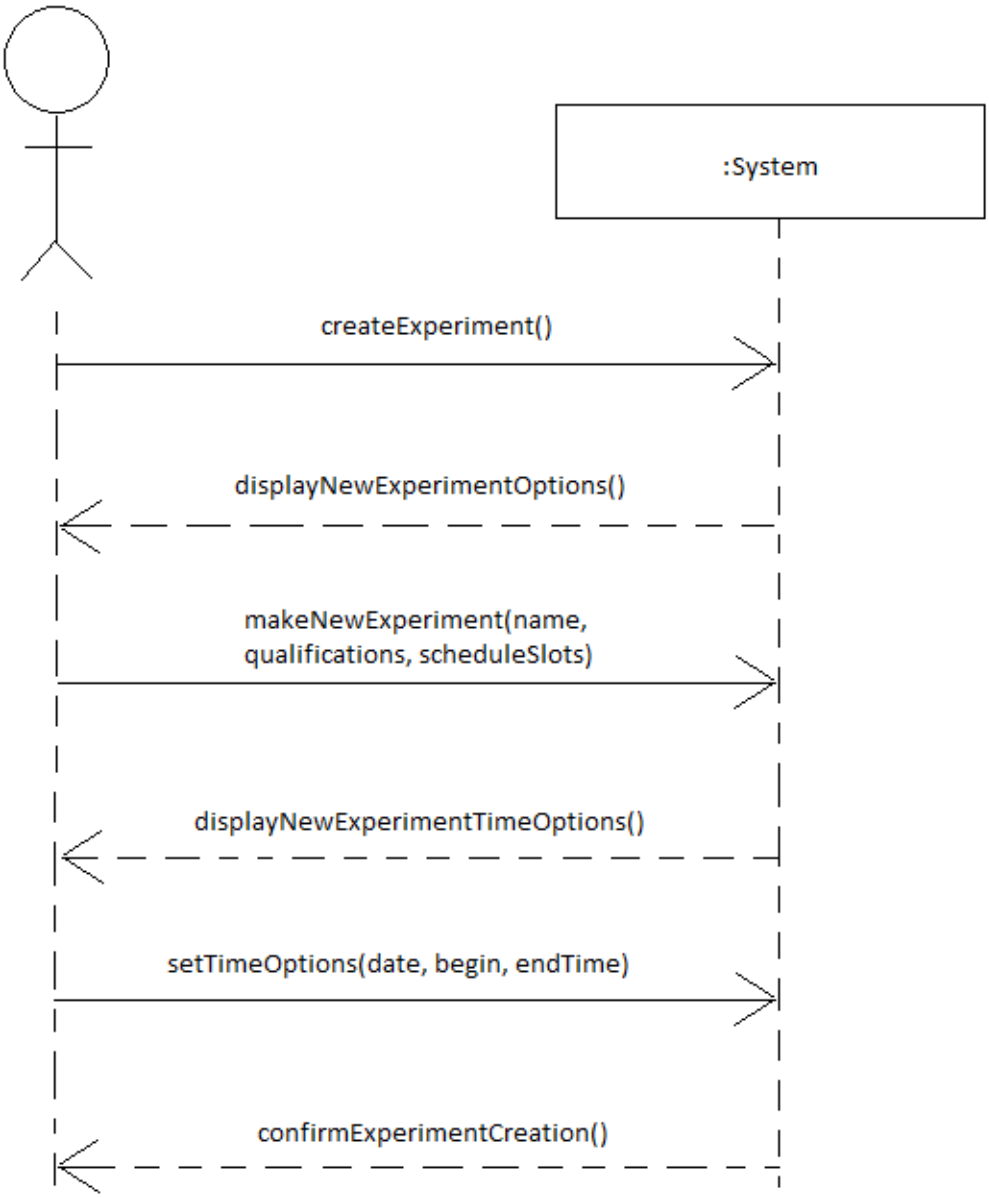
Contents

1 System Sequence Diagrams

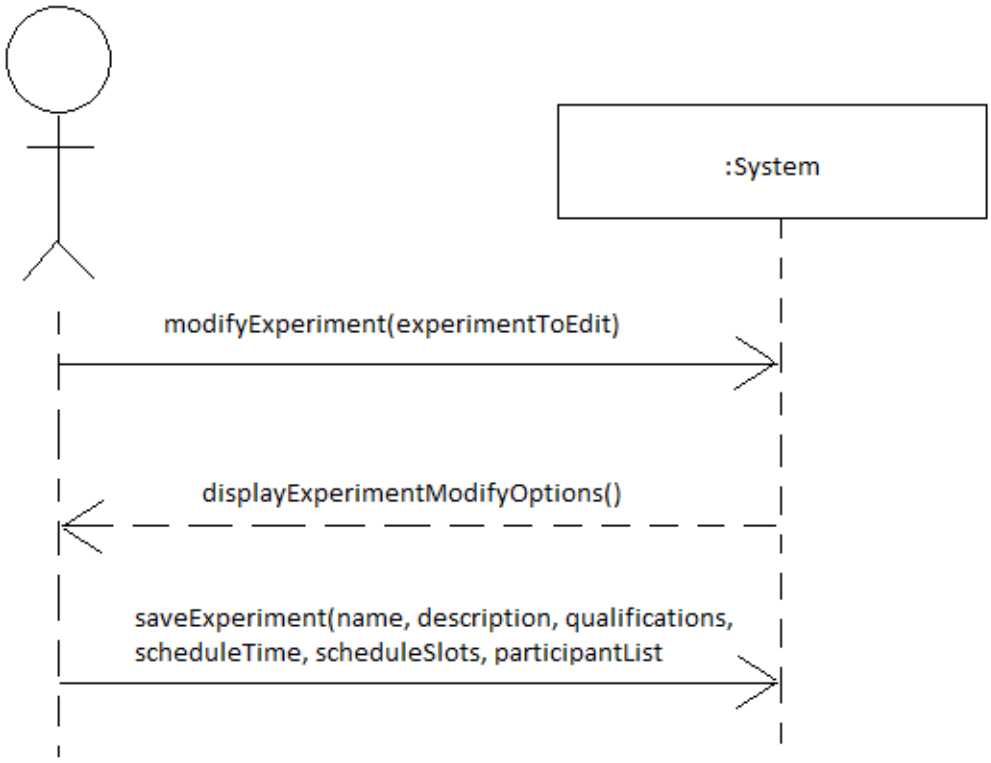
1.1 Browse and Select Experiment



1.2 Create Experiment



1.3 Modify Experiment



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 ha

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

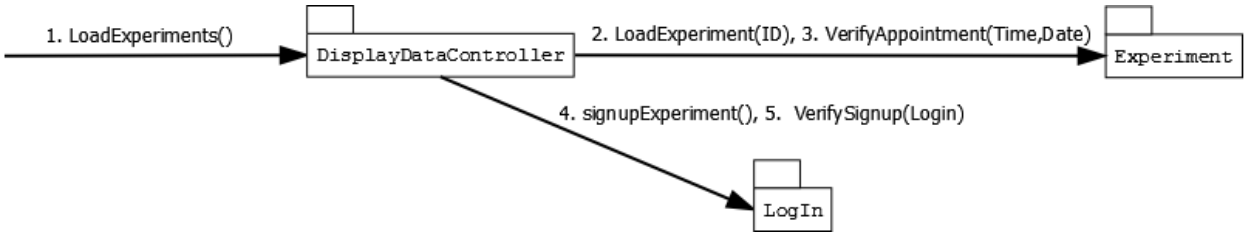
Operation:	VerifySignup(Login)
Cross References:	Uses Cases: Sign up for Experiment
Preconditions	The participant has logged into or created an account
Postconditions:	this.hasConflict is set to false (attribute modification)

2.7 Verify Appointment

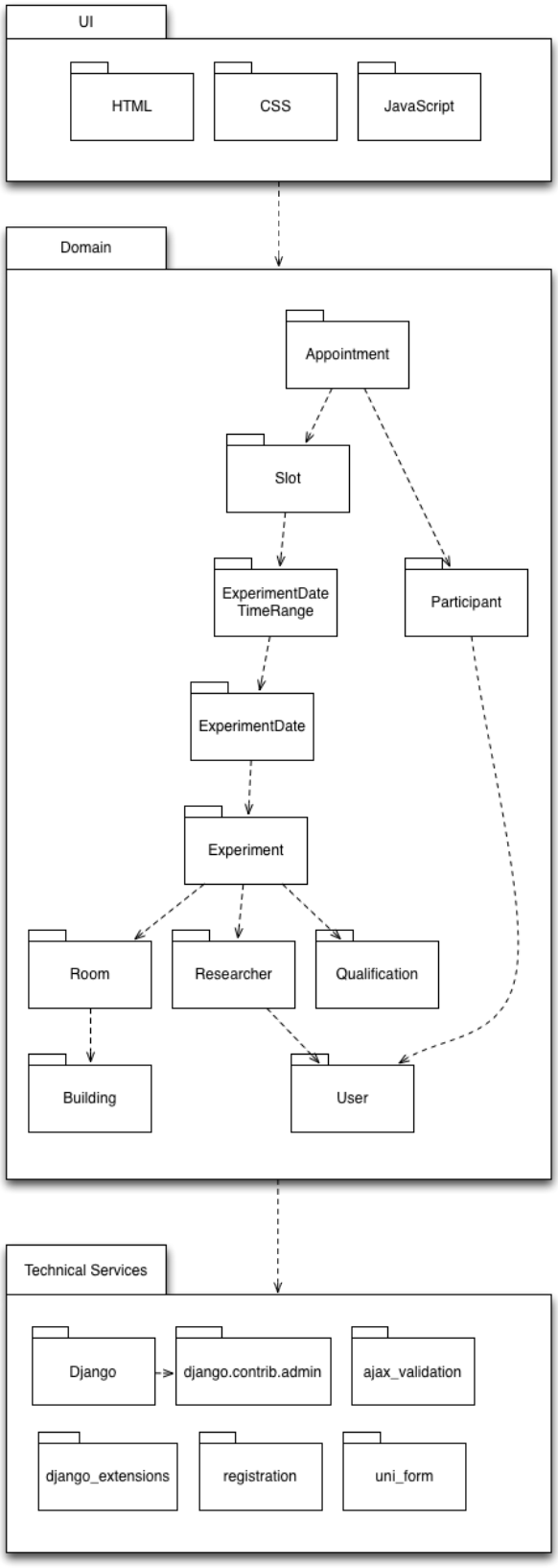
Operation:	VerifyAppointment(Time,Date)
Cross References:	Uses Cases: Sign up for Experiment
Preconditions	The participant has selected a time and date slot.
Postconditions:	experiment.slots has been modified (attribute modification)
	Database is updated

3 Interaction Diagrams

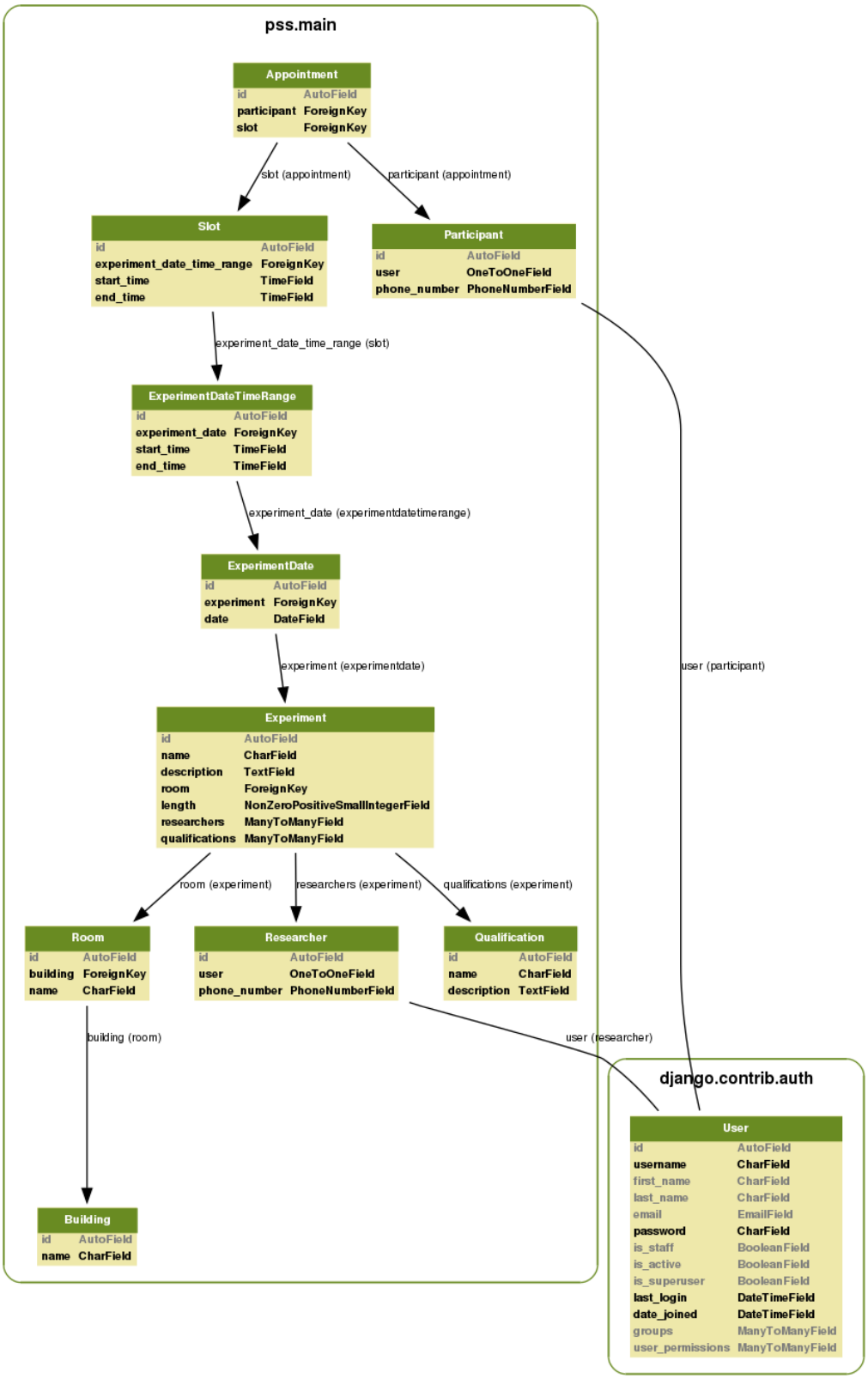
3.1 Sign Up For Experiment



4 Package Diagram



5 Class Diagram



6 GRASP Principles

6.1 Creator

6.2 Info Expert

6.3 Controller

6.4 High Cohesion

6.5 Low Coupling

6.6 Pure Fabrication

6.7 Indirection

ExperimentDate, ExperimentDateTimeRange

6.8 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.

6.9 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.

7 References

8 Appendix