Human Participant Pool Management System

A Manuscript
Submitted to
the Department of Computer Science
and the Faculty of the
University of Wisconsin-La Crosse
La Crosse, Wisconsin

by

Praveen Mutyam

in Partial Fulfillment of the

Requirements for the Degree of

Master of Software Engineering

December, 2010

Human Participant Pool Management System

By Praveen Mutyam

We recommend acceptance of this manuscript in partial fulfillment of this candidate's requirements for the degree of Master of Software Engineering in Computer Science. The candidate has completed the oral examination requirement of the capstone project for the degree.

Dr. Kasi Periyasamy Examination Committee Chairperson	Date
Dr. Tom Gendreau Examination Committee Member	Date
Dr. Mark Headington Examination Committee Member	Date

ABSTRACT

Praveen Mutyam., “Human Participant Pool Management System”, Master of Software Engineering, December 2010, (Dr. Kasi Periyasamy).

Human Participant Pool Management System (HPPMS) is a web-based application to be used primarily by the Psychology Department at the University of Wisconsin, La Crosse. Psychology students conduct research projects, called experiments, during their course work. There is a good deal of administrative work done by a researcher to conduct an experiment. Every researcher prepares the experiment information, seeks faculty approval and posts the experiments for other students to participate. During the period an experiment is conducted, the researcher who created and is responsible for the experiment updates participation details. Currently, the existing process is paper based and the approval process is manually done. The main objective of the HPPMS system is to eliminate the existing paper based system by automating the administrative activities related to the life cycle of an experiment from creating a new experiment until closing the experiment including the activities such as seeking approvals for the experiment, posting it, signing up for the experiment, and entering experiment related data. The HPPMS system also effectively communicates with its users based on various events occurring within the system. All users are notified in timely manner, and thus the system makes the experiment administration process more efficient. This application enables all users of this system work remotely, and so they need not be present in the University building physically. UWL psychology department faculty is the customer of this application.

This manuscript describes development of the HPPMS including the challenges that arose during its development and what counter measures were taken to address these challenges.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my project advisor Dr. Kasi Periyasamy and project sponsors Dr. Betsy Morgan and Dr. Alex O'Brien. I thank the advisor for valuable inputs during development life cycle. I also would like to thank the sponsors for giving me this opportunity to work on this project and spending time for providing the requirements and testing. I would also like to thank UWL Information Technology Services (ITS) for providing the computing resources for development and deployment of the project. Heath Ahnen needs a special mention who was always available to help with system issues. Finally I would like to thank my wife Sravanthi and 5 month old son Gautham for their understanding, patience and support during the project and also during my MSE program.

It would have not been possible without the help of all the people mentioned above.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY

ASP.NET

ASP.NET is a web application framework developed and marketed by Microsoft to allow programmers to build dynamic web sites, web applications and web services [7].

Active Directory

Active Directory is a technology created by Microsoft that provides a variety of network services, including: LDAP-like directory services, Kerberos-based authentication, DNS-based naming and other network information, Central location for network administration and delegation of authority [7].

Administrator

Person with the highest privileges in the HPPMS system.

Approver

Person who is authorized to approve or reject an experiment.

CSS

Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation semantics (the look and formatting) of a document written in a markup language [7].

Experiment

A research project conducted by the Psychology department students or faculty.

Experiment Administration

All activities related to conducting an experiment right from the creation of the experiment till it is closed.

HPPMS

Human Participant Pool Management System, a web-based application to automate experiment administration activities.

IEEE

An acronym for the Institute of Electrical and Electronics Engineers, Inc., which is an international organization whose constitution describes their purpose as "scientific and educational, directed toward the advancement of the theory and practice of several engineering fields including computer science."

IIS

Internet Information Services (IIS) is a web server application and set of feature extension modules created by Microsoft for use with Microsoft Windows [7].

ODP.Net

Oracle Data Provider for .NET (ODP.NET) features optimized ADO.NET data access to the Oracle database. ODP.NET allows developers to take advantage of advanced Oracle database functionality, including Real Application Clusters, XML DB, and advanced security [4].

Oracle Database

The Oracle Database (commonly referred to as Oracle RDBMS or simply as Oracle) is a relational database management system (RDBMS) produced and marketed by Oracle Corporation [7].

Participant

A person who attends an experiment, other than the researcher of the experiment.

Posting

Process of making an experiment available for signup.

Researcher

Person who is conducting an experiment.

SQL

Structured Query Language is a database computer language designed for managing data in a relational database [7].

Session

A time window during which an experiment is conducted. An experiment can be conducted in multiple sessions.

Signup

The registration process to participate in an experiment.

Stored Procedure

A stored procedure is a subroutine available to applications accessing a relational database system. Stored procedures (sometimes called a proc, sproc, StoPro, or SP) are actually stored in the database data dictionary [7].

1. Background Information

The students in the Psychology department at the University of Wisconsin-La Crosse conduct various research projects. The student conducting an experiment is called a 'researcher'. The purpose of an experiment is to collect several data from various participants of the experiment through interviews and other manual interaction processes performed by the researcher. The experiments must be approved by faculty advisers before the researcher contacts any of the participants.

Various administrative activities are involved in conducting experiments. To start with, a researcher picks up an experiment, and the date and time at which the experiment is started. He/she then needs to provide project information to his/her faculty adviser in order to seek their approval. Once approved, the researcher will post the experiment details on a printed paper on a bulletin board at the university. Interested students, called 'participants' are expected to see the bulletin board for any posted experiments and sign-up by writing their names on the paper. Bulletin board is an open area in the university and is accessible to everyone who can enter the building, which is not secure. There is no authorization required to sign-up for an experiment, since this is accessible to anyone who can enter the building. Thus, participants can write their names on the experiment signup sheet. Participants need to remember the date and time of the experiments they are participating in, and so there is no reminder sent by the researcher. If a researcher decides to send a reminder about experiment, he/she needs to collect participant email-ids from the corresponding signup sheet and send emails manually. After conducting the experiment, the researcher collects the participation attendance information, participation time and submits the same to the faculty advisor who approved this experiment initially. The faculty advisor uses this information to provide credit to those students for their participation. A faculty advisor also conducts experiments and the process used is the same except for the approval.

Currently, all activities related to an experiment including its creation, approval, posting, signup and reporting are all done manually. Researchers, faculty and participants are involved in every step of an experiment administration and the process is inefficient and time consuming. Researchers should be at the university to post the experiments, and participants also need to be physically present on campus to signup. Communication is a manual process, and so faculty advisors and researchers need to communicate personally or through emails about experiment approval or rejection. In addition, a researcher sends attendance participation information through email. Researchers and participants are not notified about participation signup or cancellations.

The aim of this project is to develop 'Human Participant Pool Management System' (HPPMS), a web based application for maximum automation of activities related to experiment administration. This system should allow a researcher to create experiments electronically and route them to faculty advisors for approval. The system should also let faculty advisor email the decision back to the researcher. Once an experiment is approved, the system should automatically post the experiment. If, on the other hand, an experiment is rejected, the researcher should receive an email with reasons for rejection and corresponding actions to be taken. All posted experiments should be available for participants to browse and sign-up. The system should allow a participant to cancel the registration if he/she decides not to attend the experiment. Once the experiment is conducted, the researcher should be able to update attendance information and able to export this information in a Microsoft Excel document.

The HPPMS system should allow only authenticated users to use the system and authorize access based on the user roles. Thus, for example, a participant should not have access to create experiments. An administrator user should have privileges to change the roles of other users and their privileges.

This report describes the processes used to develop the HPPMS system. The sponsor of this project compiled the initial set of requirements which were very primitive at the

beginning. Consequently, this report also explains how these requirements are used as a starting point and how the final application was developed.

2. Brief Introduction to Software Life Cycle Models

A software life cycle model depicts the significant phases or activities of a software project from conception until the product is retired. It specifies the relationships between project phases, including transition criteria, feedback mechanisms, milestones, baselines, reviews, and deliverables. Typically, a life cycle model addresses the various phases of a software project: requirements phase, design phase, implementation, integration, testing, operations and maintenance [6].

This document briefly explains the two popular software life cycle models

1. The Waterfall Model
2. The Prototyping Model

The Waterfall Development Model

The waterfall process model encourages the development teams to specify what the software is supposed to do (gather and define system requirements) before developing the system. This model breaks the complex mission of development into phases (design, code, test and so forth) with intermediate deliverables that lead to a final product. To ensure proper execution with expected deliverables, each phase has a validation, entry, and exit criteria. This Entry-Task-Validate-Exit (ETVX) paradigm is the key characteristic of the waterfall process [2]. Figure 1 shows the phases of the Waterfall Process Model. This model follows strict sequencing of phases with no or little feedback to the previous stages. In waterfall model

- Work flow through a series of stages (phases)
- Each phase is completed before moving to the next phase
- Previous phase is not visited again unless there is an error found in the current phase

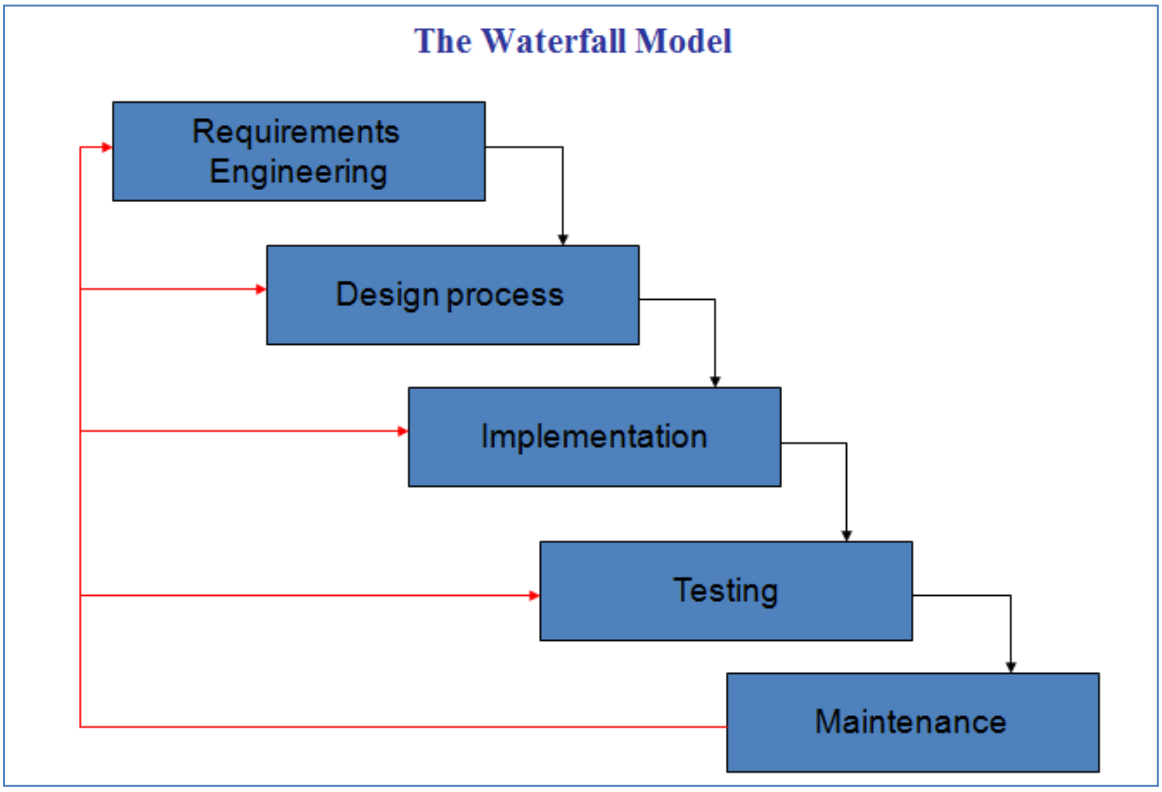


Figure 1. The phases of the Waterfall Model

The Prototyping Approach

A prototype is a partial implementation of the product expressed either logically or physically with all its external interfaces presented. The potential customers use the prototype and provide feedback to the development team before full-scale development begins. *Seeing is believing*, and that is really what prototyping intends to achieve. By using this approach, the customer and the development team can clarify requirements and their interpretation [1].

The incremental prototyping approach uses iterative approach and following steps are used in each iteration:

- Add a subset (possibly small subset) of requirements
    - Correspond to most important functionalities that were not yet implemented
- Analyze the requirements, design, implement, test (by developer), and check with the customer
- Requires customer's approval before moving to the next iteration

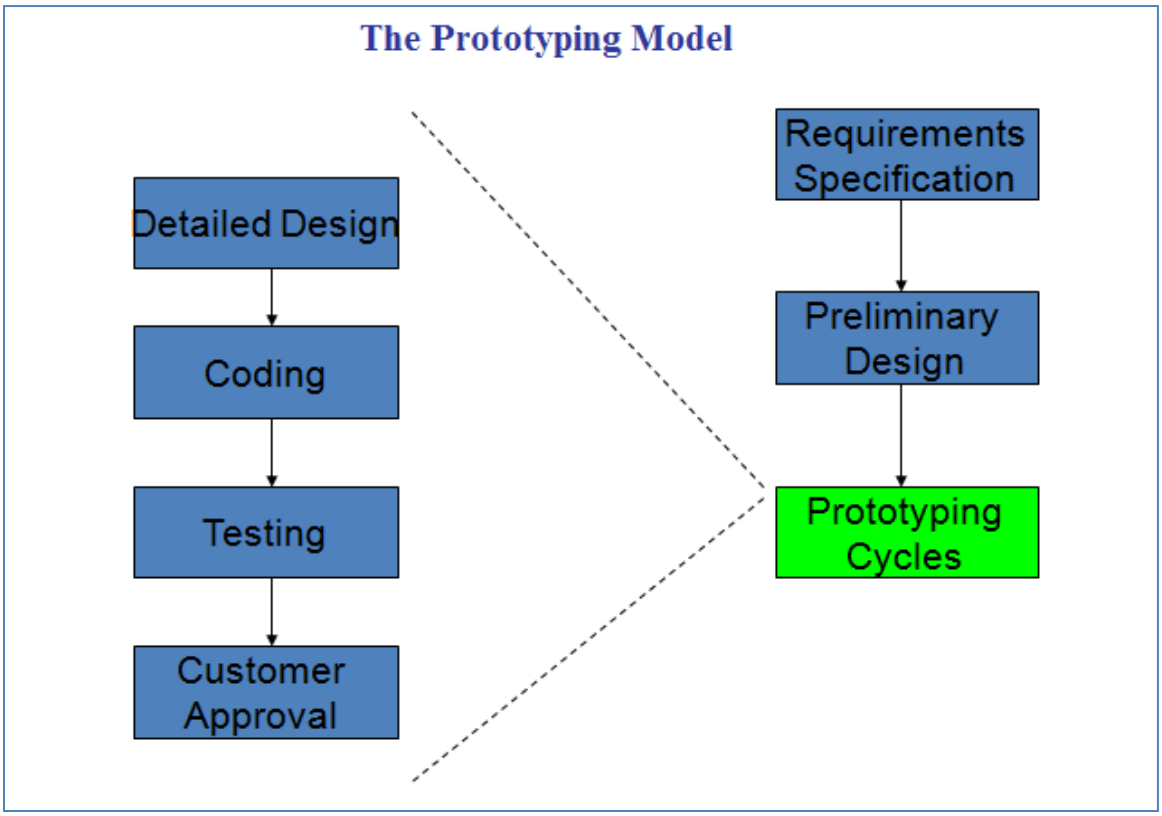


Figure 2. Steps in Prototyping Model

The critical factor for success of the prototyping approach is its quick turnaround in designing and building the prototypes. Several methodologies can be used to achieve such an objective. Reusable software parts could make the design and implementation of a prototype easier. Formal specification languages could facilitate the generation of executable code (e.g. the Z notation and Input/output Requirements Language (IORL)) [1]. Two popular prototype approaches are explained here.

Rapid Throwaway Prototyping

The rapid throwaway prototyping approach of software development was made popular by Gomaa and Scott in 1981, but is not used widely in the industry, especially in application development. It is generally used with high-risk items or with parts of the system that the development team does not understand thoroughly. In this approach, a "quick and dirty" prototype is built, verified with customers, and is thrown away until a satisfactory prototype is reached. The full-scale development begins once the goal and objectives of the end product are understood by the developer and the customers [1].

Rapid throwaway prototype process uses following steps:

- Initial:
    - A 'quick and dirty' prototype is developed that implements the most important subset of requirements
    - quick because it is developed faster
    - dirty because development process may not follow any standard or disciplined approach
    - Get the approval of the customer
- Follow-up:
    - After the customer's approval, discard the prototype
    - Start the project again from scratch; may use any model afterwards
    - Use the knowledge and expertise gained in developing the prototype

Evolutionary Prototyping

In evolutionary prototyping approach, a prototype is built based on some known requirements and understanding. The prototype is then refined and evolved instead of thrown away. Whereas throwaway prototypes are usually used with the aspect of the

system that are poorly understood, evolutionary prototypes are likely to be used with aspects of the system that are well understood and thus built on the development team's strengths. The prototypes are also based on prioritized requirements, sometimes referred to as "chunking" in application development. For complex applications, it is not reasonable or economical to expect the prototypes to be developed and thrown away rapidly [1].

2.1.Life Cycle Model Used

The evolutionary prototyping method was used to develop the HPPMS application. The customer had an initial set of requirements written as a two page document. Several interviews were held with the customer to get detailed requirements initially. However, the customer could only give high level requirements such as "Experiment data entry page should be user friendly" and "Students should be able to sort the data". Since the customer did not have prior software development experience, it became difficult to extract detailed requirements. Existing manual process was reviewed with the help of the customer. The bulletin boards at the university was reviewed by the developer to get sample experiment data and to find out the method used to register for experiments. Document formats used by the current students to post experiments were also reviewed during interviews.

Two iterations were used for HPPMS development. The first iteration included key requirements agreed by the customer. Functions to create experiment, signup for experiment and cancel from experiment and excel export feature of experiment participants were key requirements agreed by the customer. User authentication was done against a local database in the first iteration. This iteration helped customer feel the product first hand and come up with several changes to the previous set of requirements. The customer also added new set of requirements which were implemented in the second iteration. One of the major changes included in the second iteration is that the customer wanted an experiment to have multiple sessions. This triggered changes in many areas of

the previous iteration. Additional data was included to store an experiment. This included faculty, length of experiment, special requirements, session, and location. Users were authenticated against UW-L network in the second iteration. More information on the requirement changes in iterations is included in chapter 3 - Development of HPPMS.

3. Development of HPPMS

This section describes the development process of HPPMS.

3.1.Requirements

Multiple meetings were held with the customer to collect and understand the initial requirements for the project. The existing paper based process was reviewed to understand the processes and functions. The customer for the project wanted to automate the experiment administration process using a web-based application, where all experiment administration activities are performed electronically. The initial requirements as given by the customer were at high level of abstraction, too short, and were typically given as a two-page document. For example, some of these high level requirements were "create and post experiment", "HPPMS should be user friendly", "user should be able to sort the data", and "the system should notify different events in a timely manner". Each requirement was discussed in detail with the customer to derive additional details. All requirements were documented in IEEE format [5]. Meetings were also held with Information Technology Services (ITS) department at the university to decide the technology and development resources to be used for this project.

Once initial requirements were gathered from the customer and ITS, key requirements were identified to develop the first prototype of application. This prototype was demonstrated to the customer to give them a feel of application and to gather the next set of requirements and changes to be made to the previously implemented requirements. Two iterations were used for HPPMS development. The prototypes were useful in refining the requirements because the customer could see the development of project at various phases and could provide a detailed set of requirements.

The initial prototype included key requirements only. This version allowed creation and posting of experiment, signup and cancellation. Experiment information included only basic information such as unique experiment identification, one start time and end time. User account information was stored in HPPMS database. Users should register for the first time and their user id, password information were stored in HPPMS database. This version allowed all the users of system to create experiments.

The second prototype included major changes to existing requirements and new requirements were added. Changes were done to experiment information to allow multiple start times and end times to allow researcher to conduct experiment in different sessions. This prototype also allowed a researcher to record attendance information and upload the same to an Excel document. Option to send email notification to all participants was also included. Workflow is implemented to route experiment to faculty advisor for approval. Faculty advisor is able to either approve or deny experiment. Participants have additional option to look at all experiments they participate.

Security changes were done to authenticate users against UW-L active directory to access HPPMS. Role based access is implemented in this version. A user by default is provided with a participant role. An administrator of system can provide additional roles to a user. Generally at the beginning of a semester, a faculty advisor provides the researcher role to selected students. This role gives access to create experiment function and related administrative functions. Researcher role may be revoked at the end of the semester.

Table 1 lists the roles of HPPMS users and their functions identified during requirements gathering. Administrator role is given to all faculty advisors so that they can change the security of other users.

Role	Allowed Functions
Administrator	• Create an experiment • Browse and signup for an experiment • Review an experiment • Approve and post an experiment • Deny an experiment • Assign roles to users • Revoke roles from users
Participant	• Browse and register for an experiment • Cancel registration for an experiment • View signed up experiments
Researcher	• Create an experiment • Update an experiment • Request for approval of a newly created experiment • Updating participation information such as attendance • Browse and register for an experiment other than the one created by himself/herself • Cancel registration for an experiment

Table 1. Roles of HPPMS users and their functions.

3.2. Use Cases

Generally, use case diagrams are used in capture key requirements of a software product. Use case diagrams help customer see requirements in a pictorial format. New use case diagrams are developed when there were changes in existing requirements or when new requirements were added. Figure 3 shows requirements while developing the initial prototype and Figure 4 shows the requirements implemented in the final application. Together, Figures 2 and 3 indicate how the requirements evolved during the development of the HPPMS system. To give an idea of the details of each use case, the narrative of the "Add Experiment" use case is included in Table 2. This narrative explains the details in adding a new experiment.

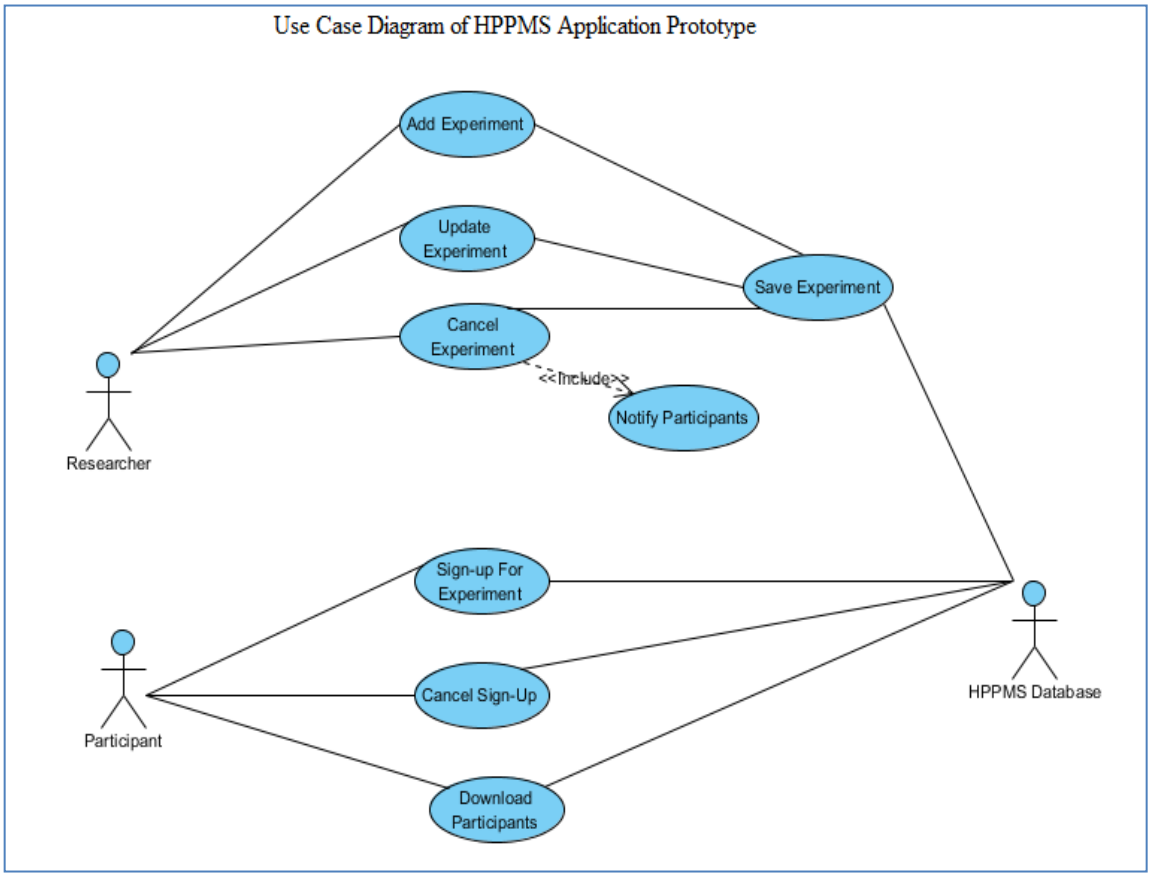


Figure 3. Use case diagram used in the initial prototype

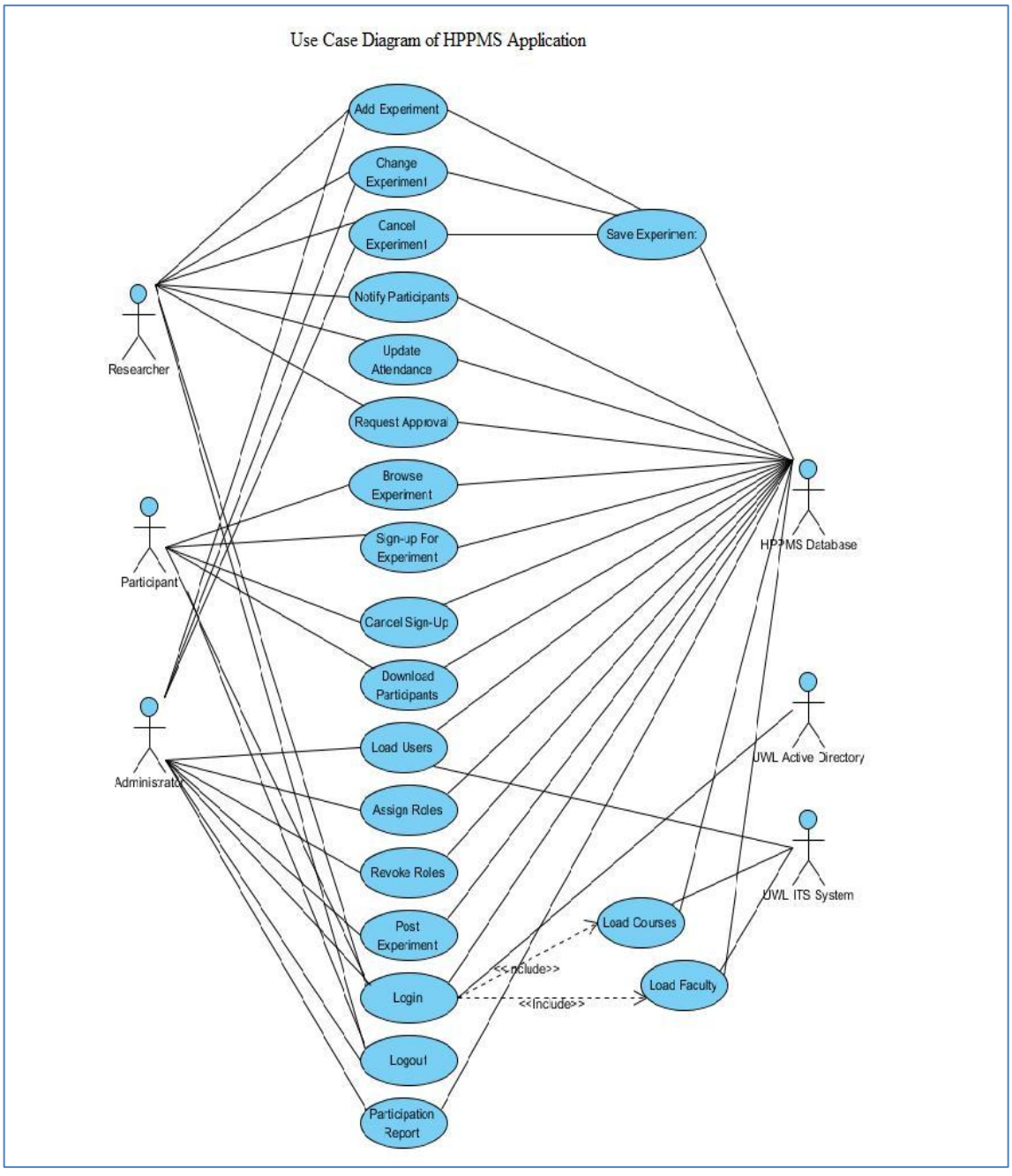


Figure 4. Use case diagram implemented in the final application

Use Case Narratives

The narrative of the "Add Experiment" use case is listed in Table 2.

Index	RF1
Name	Add Experiment
Priority	1
Purpose	A researcher can create new experiments in the system and request participation from other students. The purpose of this function is to add a new experiment into the system.
Input parameters	• Experiment ID • Student ID • Title • Description • Pre-requisites • Study Length • Study Type • Faculty Name • Session ID • Start Date and Time • End Date and Time • Location (Building and Room number) • Capacity • Open Slots • Status There can be multiple entries for fields from 9 to 14 (multiple sessions for an experiment)
Actions	1. User enters values for all required fields and then clicks on the save button. 2. Ensure that title is not blank. 3. Ensure that description is not blank 4. Ensure that study type is entered. 5. Ensure that faculty name is entered if study type selected is 331 or Honors. 6. Ensure that study length is entered as either 30 or 60 minutes.

	7. Auto generates unique session ID for the experiment. 8. Ensure that start date and time are valid and not in the past. 9. Ensure that end date and time are valid and not in the past, and make sure this are greater than start date and time respectively. 10. Ensure that the difference between end time and start time is equal to study length. 11. Ask user to enter building and room number for location and ensure it is not blank. 12. Ensure that capacity is not a negative number. 13. Initialize open slots to maximum capacity 14. Ensure that status is "Open", which is the initial state of experiment 15. Generate unique experiment ID based on study type, using the following prefix based on study type. a. 331 – E (for Experimental Psychology) b. Honors - H c. Faculty – F d. Others – O 16. Add the new experiment to the database.
Exceptions	1. Title is blank 2. Description is blank 3. Start date and time are in the past 4. End date and time are in the past 5. End date and time is sooner than start date and time. 6. Difference between end and start time is not equal to study length. 7. Location is blank 8. Length of study is negative 9. Type of study is blank 10. Capacity is negative number. 11. Status is not valid (see remarks section for allowed values). 12. Researcher may have a conflict with another

	experiment session time. Researcher may have signed-up for another experiment which conflicts with.
Output parameters	New experiment is created
Remarks	• Status of an experiment can be one of the following values – • Open (experiment is not posted and is not available for participation) • Pending (waiting for an administrator to post experiment) • Posted (students can sign-up for the experiment) • Cancelled (cancelled for some reason, only allowed before end date/time) • Closed (completed the experiment successfully) • Start and end date/time also state the validity of experiment, past end date/time students cannot sign-up for the experiment.
Cross references	None

Table 2. Add Experiment Use Case Narrative

4. Object-Oriented Design of HPPMS

After the requirements were thoroughly analyzed and understood both by the customer and the developer, the architecture of the HPPMS system was developed and represented using UML class diagram. The interactions between the objects were described using a set of sequence diagrams. This section illustrates the class diagram and some of the sequence diagrams developed for the HPPMS system.

4.1. High Level Architecture

HPPMS was developed using the classic three-tier architecture, where presentation, application, and the data layers are logically separated. This architecture provided a model for the developer to create a flexible and reusable application. By breaking up an application into tiers, the developer only needed to modify or add specific layer rather than have to rewrite the entire application.

Figure **5** shows the high level architecture of HPPMS.

The presentation layer was developed using ASP.Net. The user interface is presented in HTML format to end users. Style sheets are used to make the user interface consistent in the application so that user experience is same in all pages. To ease date entry process, popup calendar controls are used.

The application layer is the core part of HPPMS, which handles the application logic and also interacts with the data layer. An end user will never directly interact with the data layer. The application layer is designed to interact with multiple users concurrently. It was developed using C# and ODP.Net, C# is used for the application logic and ODP.Net for the data layer interaction. The Data layer was developed using Oracle database. An external database called ‘Wings database’ available at the University of Wisconsin-La Crosse was also used, which is outside of HPPMS; this database is accessed using a database link.

Since the users of the HPPMS system are all UW-L faculty and students, it was decided by the customer that all users should be able to login with their existing UW-L network account and password. In order to support this requirement, the UW-L Active Directory is used for authenticating users. Figure 4 shows a high level architecture of the HPPMS system indicating the different layers and the databases used.

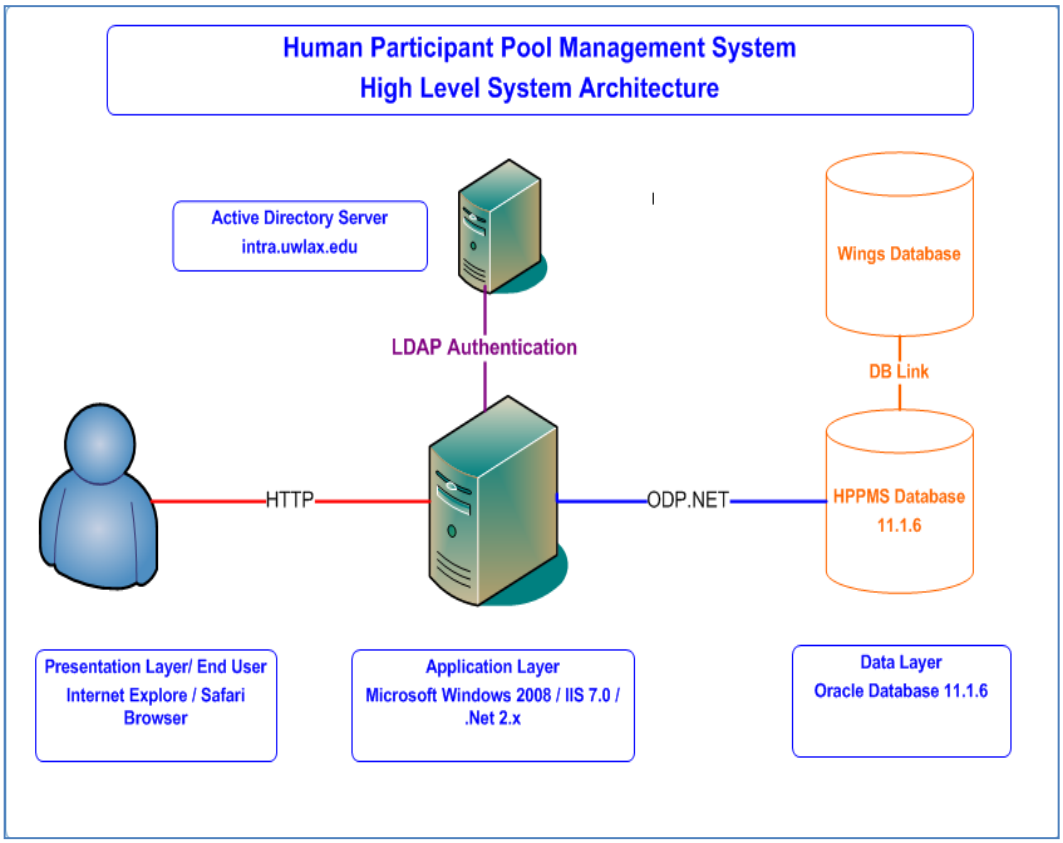


Figure 5. High Level Architecture of HPPMS

4.2. Class Diagram

The class diagram for the HPPMS system is shown in Figure 6. Due to space limitations, only some of the classes are explained in detail in Tables 3-6.

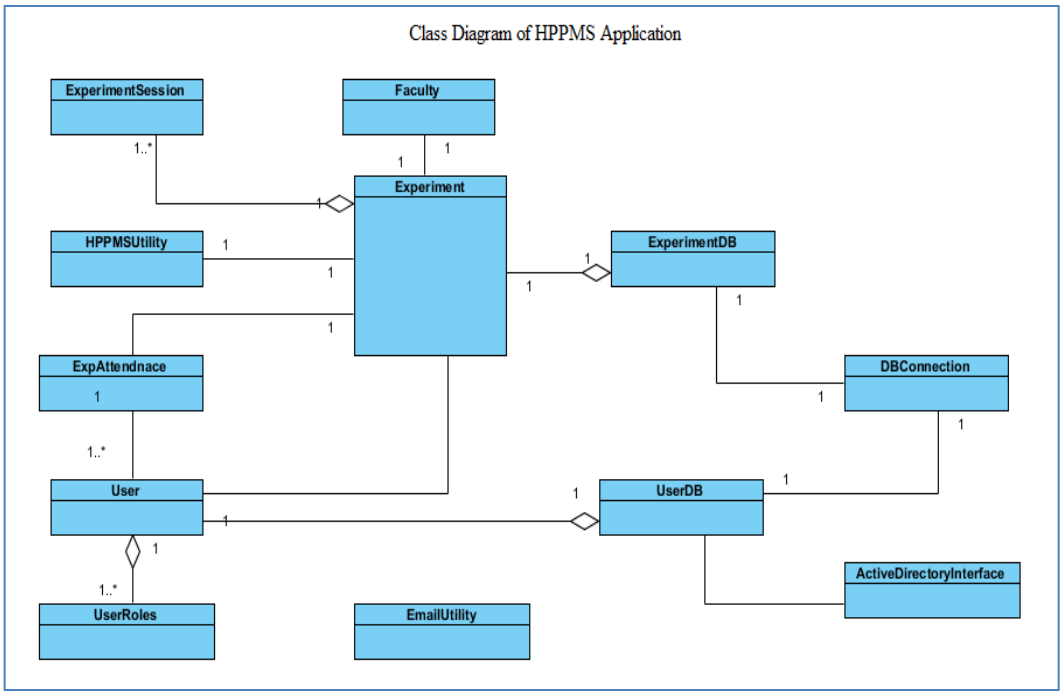


Figure 6. Class Diagram of HPPMS Application

Class Name	Experiment
Purpose	This class is designed to store single Experiment information.
Attributes	• private m_nStudyLength : int • private m_strDescr : String • private m_StrExpID : String • private m_strFacultyName : String • private m_strSpecialRequs : String • private m_strStatus : String • private m_StrStudID : String • private m_strStudyType : String • private m_strTitle : String • private m_nStudyLength : decimal
Methods	• public Initialize() • piblic ExpID() • piblic StudID() • piblic Title() • piblic Descr()

	• piblic StudyType() • piblic FacultyName() • piblic SpecialReq() • piblic StudyLength() • piblic Status()

Table 3. Class Experiment

Class Name	ExperimentDB
Purpose	This class is designed to communicate to database for Experiment related transactions.
Attributes	• protected m_dbConn: DBConnection • private m_ m_strConnectionString : String
Methods	• public AddExperiment(Experiment objparamExp, DataTable objParamDTExpSessions) • public AddSessions(OracleCommand cmd, DataTable objParamDTExpSessions) • public CancelSignUpOracle(String strParamExpID, int nSessionID, String strParamStudID) • public CanSignUp(String strParamExpID, String strParamStudentID) • public ChangeExpStatus(String strParamexpID, String strParamStatus) • public DoesSessionExist(String strParamExpID, Decimal dParamSessID) • public Experiment DB() • public GetEmailIDExpOwner(String strParamExpID) • public GetExperiment(String strParamExpID) • public GetFacEmailExp(String strParamExpID) • public GetNextExpID(Exp Type enParamExpType) • public GetPartEmailID(String strParamExpID) • public IsAConflict(DateTime dtParamStartTime, DateTime dtParamEndTime, String strParamStudID) • public IsAConflictOracle(DateTime dtParamStartTime, DateTime dtParamEndTime, String strParamStudID) • public RequestApproval(String strParamExpID) • public SendEmails(String strParamExpID, int

	nParamSessionId, String strParamAction,User objParamCurrUser) • public SignoutExperiment(String strParamExpID, String strParamStudID) • public SignupExperiment(String strParamExpID, String strParamStudID) • public SignupExperimentOracle(String strParamExpID, int nParamSession, String strParamStudID) • public UpdateAttendence(List<ExpAtt> objListAtt) • public UpdateExperiment(Experiment objParamExp,DataTable objParamDTExpSessions) • public UpdateExperiment_OLD(Experiment objExperiment) • public UpdateOpenSlots(String strParamExpID, int nParamSessionID,int nParamUpdateBy)

Table 4. Class ExperimentDB

Class Name	ActiveDirectoryInterface
Purpose	This class is designed to communicate with UWL Active Directory.
Attributes	• private m_objDirEntry : DirectoryEntry
	• private m_strADPathOther : String • private m_strADPathStud : String
Methods	• public ActiveDirectoryIntfc() • public AuthOther(String strUserP, String strPwdP) • public AuthStudents(String strUserP, String strPwdP) • public GetADSILogin() • public GetADUserInfo(String str84ID)

Table 5. Class ActiveDirectoryInterface

Class Name	UserDB
Purpose	This class is designed to communicate to database for

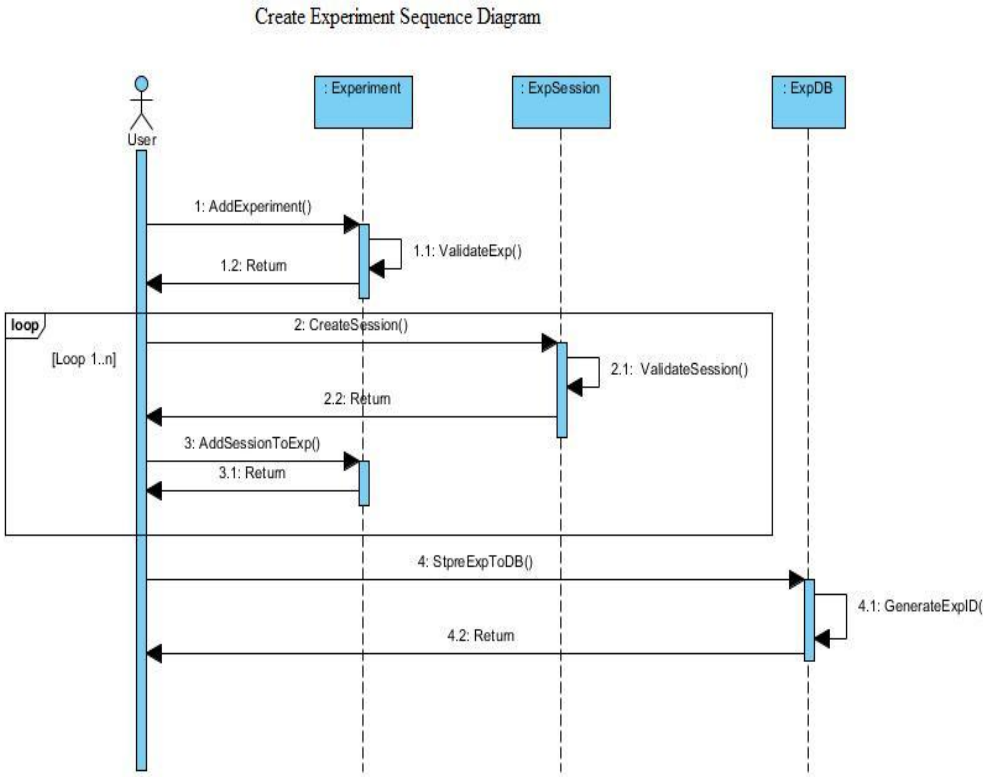
	User related transactions.
Attributes	• private m_strConnectionString : String • private m_dbConnection : DBConnection • private m_strADPathOther : String • private m_strADPathStud : String
Methods	None

Table 6. Class UserDB

4.3.Sequence Diagrams

Sequence diagrams are developed to document key interaction of objects in HPPMS for important scenarios. Only selected sequence diagrams are listed in this document. These selected sequence diagrams illustrate the successful scenarios ‘Add Experiment’, ‘Approve or Reject Experiment’ and ‘Login’.

Scenario Name	Add Experiment
Scenario	Adding a new Experiment successfully to database.
Sequence Diagram	

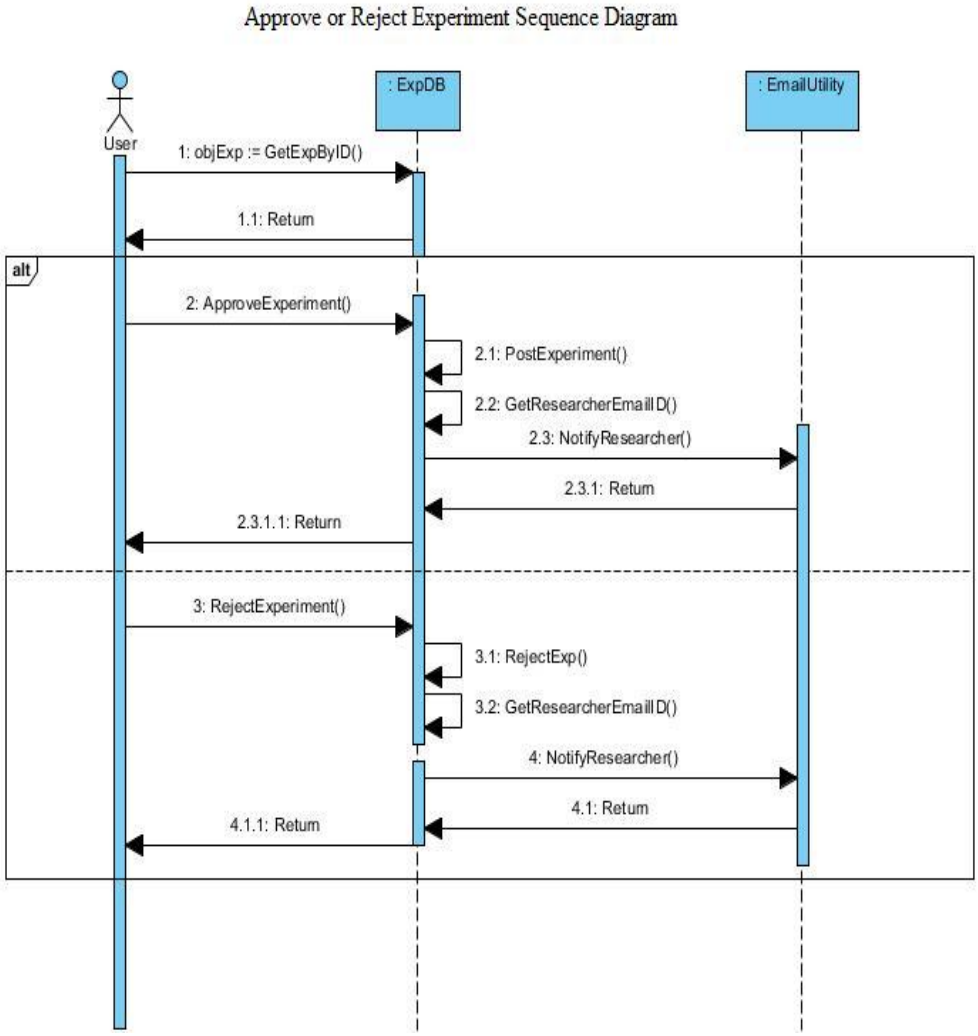


Label in Sequence Diagram	Notes
1	New Experiment object is created using following information • Title • Description • User ID • Status • Study Type • Study Length • Faculty Name • Special Requirements
2	New Session is created with the following information • Start Time • End Time • Location • Capacity
3	Session is added to Experiment.
4	Experiment is saved to database using StoreExpToDB

	method.

Table 7. Add Experiment Sequence Diagram

Scenario Name	Approve or Reject Experiment
Scenario	Approving or rejecting an experiment successfully. An experiment will be automatically posted on approval.
Sequence Diagram	

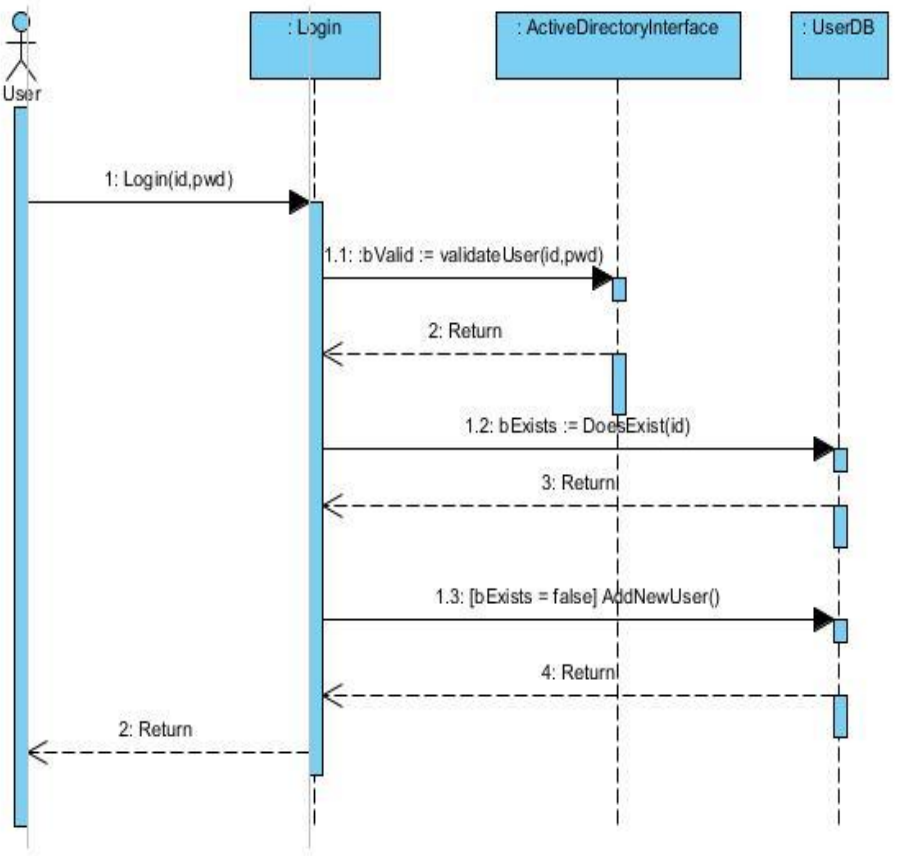


Label in Sequence Diagram	**Notes**
1	Existing experiment is retrieved using Experiment ID

2	Review experiment information and approve.
2.1	Post the experiment by changing the status to “Posted”
2.2	Get Researcher’s EmailID
2.3	Notify Researcher that experiment is approved.
3	Review and Reject experiment
3.1	Change Experiment status to “Open”
3.2	Get Researcher’s Email ID
4	Notify participant about rejection.

Table 8. Approve or Reject Experiment

Scenario Name	**Login**
Scenario	Login successfully into HPPMS. If this is the first time usage for the user, then add user details to HPPMS database.
Sequence Diagram	



Label in Sequence Diagram	**Note**
1	User logs in with id, password.
1.1	Validate user credentials using Active Directory.
1.2	Check if user already exists in HPPMS database.
1.3	If user does not exist in HPPMS then add user to database.
2	Return control to user.

Table 9. Login

4.4.Database Design

Generally a database structure is defined during the design where all tables, primary, foreign keys and relations have been identified. The customer for HPPMS was looking to utilize data from an external database called Wings for student registration information. This decision was made in order to use the database links for accessing Wings data. Database views are created to access ITS data from HPPMS. Database design was changed a few times during development iteration process. Figure 9 shows the Entity Relation diagram from initial prototype and Figure 10 shows the Entity Relation diagram from final prototype.

The initial prototype had only four tables and user account information was stored in HPPMS database. Experiment data was stored in a single table and there was no interface to ITS database.

The final prototype included additional tables to store experiment session data, faculty data and registration information. Database views are created to access ITS database. These views are accessed through database link and are read-only. HPPMS can never update ITS database using this database link. Changes in ITS databases are made available instantly to HPPMS database. Foreign constraints are created to maintain data integrity and field level constraints are crate to make sure correct values are stored in the database.

Since databases are very powerful in computing operations than front-end application, stored procedures are used extensively in the application to reduce the load on front-end application and improve performance. Figure 7 shows stored procedure to read email IDs of participants. This stored procedure is used to read email IDs and send reminder notifications a day before the experiment starts. This stored procedure is used in SQL statement as shown in Figure 8. Using this approach helps bulk load of processing done in the database than front-end application. Similar approach is used in other parts of the application.

```
create or replace
function emailid_transpose(pinExpID in varchar2,pinSessID number)
return varchar2
is
    poutEmailIDs  varchar2(10000) default null;
    strSep  varchar2(1) default null;
begin
    for X in (select USR.EMAILID from USERDEFN USR, EXP_SIGNUP ES, EXP_SESSIONS SESS where
    ES.STUDENTID = USR.STUDENTID and
    ES.EXPID = SESS.EXPID and SESS.EXPID = PINEXPID and SESS.SESSIONID = PINSESSID) LOOP
    poutEmailIDs := poutEmailIDs || strSep || x.EMAILID;
        strSep := ';';
    end loop;
    return poutEmailIDs;
end;
/
```

Figure 7. Stored procedure to read email IDs

```
select SESS.EXPID,SESS.SESSIONID,EMAILID_TRANSPOSE(SESS.EXPID,SESS.SESSIONID)
from EXPERIMENT EXP, EXP_SESSIONS SESS
where EXP.EXPID = SESS.EXPID
and to_date(sess.starttime,'DD-Mon-YYYY') = to_date(sysdate+1,'DD-Mon-YYYY')
```

Figure 8. SQL statement to use stored procedure

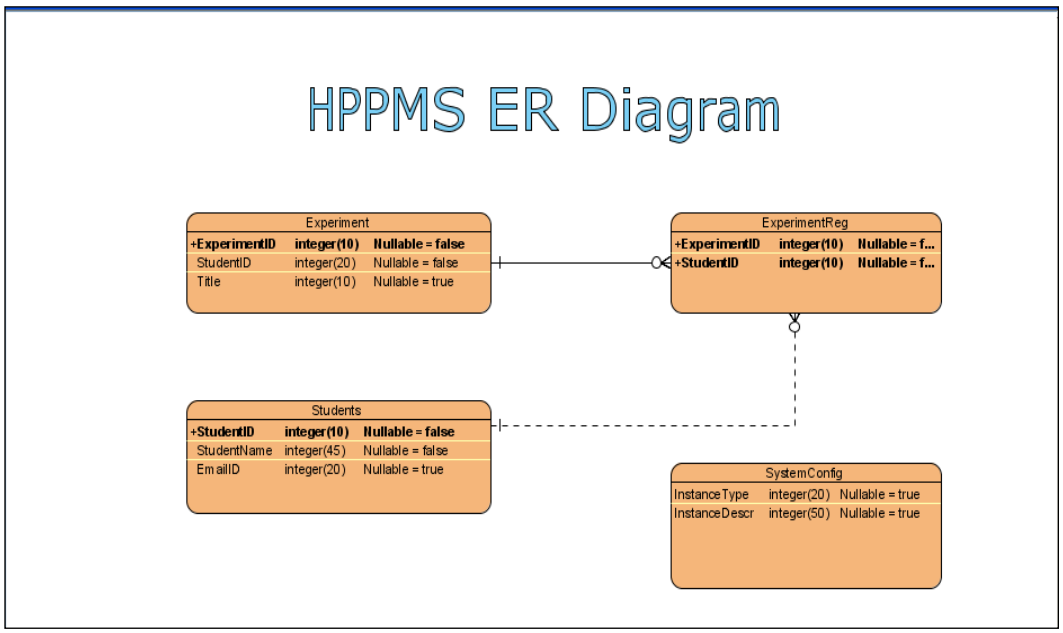


Figure 9. ER Diagram from Initial Prototype

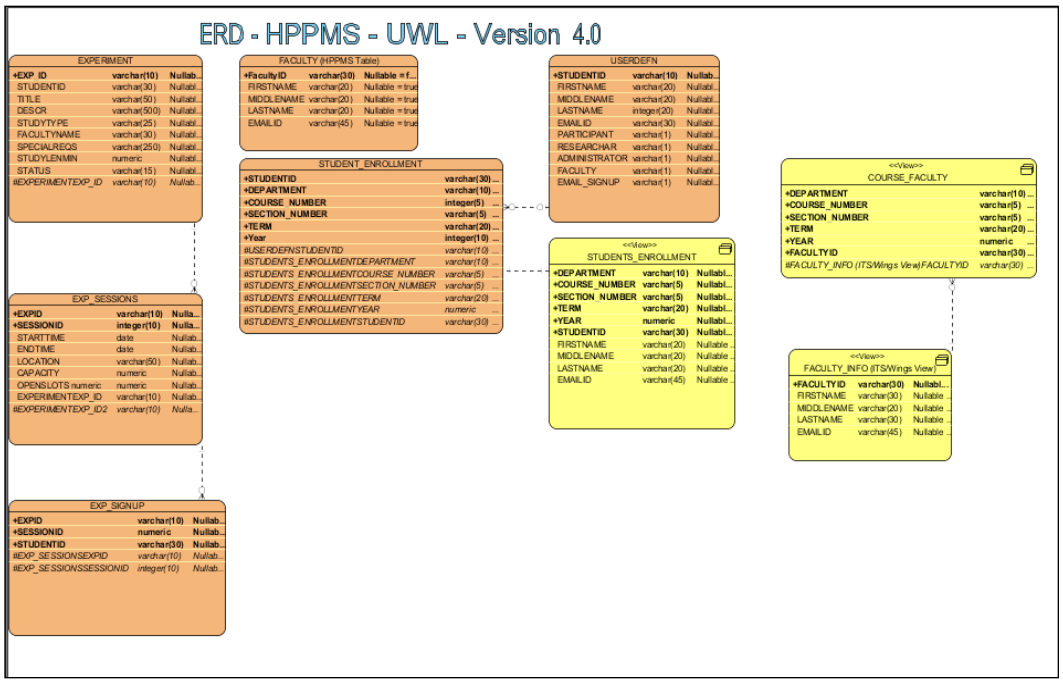


Figure 10. ER Diagram from the final application

5. Implementation

The customer of the HPPMS application, faculty from the Psychology Department at the University of Wisconsin-La Crosse, does not have a server to host the HPPMS application. However, the customer has experience working with Microsoft Windows servers. Initially, they wanted the HPPMS application to be deployed to their departmental machine, where they have full control because they wanted to administer the application on their own. This decision drove the technology to be ASP.Net as front-end and Microsoft SQL server as back end, since these two can run on a Windows server. The initial prototype and further development was done using ASP.Net and Microsoft SQL server. The customer then changed the decision of hosting the application in the middle of the development process. They wanted the application to be hosted by UW-L ITS data center, which can only support Oracle database. Due to this decision, most of the application layer was re-written to use Oracle database. Table 10 shows the final platform details of HPPMS.

Application Layer	Technology	Hosted Platform
Presentation Layer	ASP.Net 2.0	Microsoft Windows 2008/IIS 7
Application Layer	C#	Microsoft Windows 2008/IIS 7
Data Layer	Oracle 11.1.6	Unix

Table 10. Technology Used

5.1. User Interface

ASP.Net master pages are used to create a consistent layout for all pages in HPPMS. A single master page named ‘masterlayou.Master’ is created for the look and feel of the application and the same is used across all pages. A menu is attached to masterlayout.Master, so that no additional work is done in other pages for menu operations. The master layout is used as a template for all other pages. All other ASP pages are created as ASP.Net ‘Content Pages’, where content related to a specific function is created and they all include the master page which displays a consistent interface. For example, the page to create an experiment and the one to show the experiments created by a particular researcher are created as content pages by referring to master page, so that both display the same interface menu. Figure 11. Shows manage_exp.aspx (Manage Experiment Page) header which refers to the masterlayout.Master.

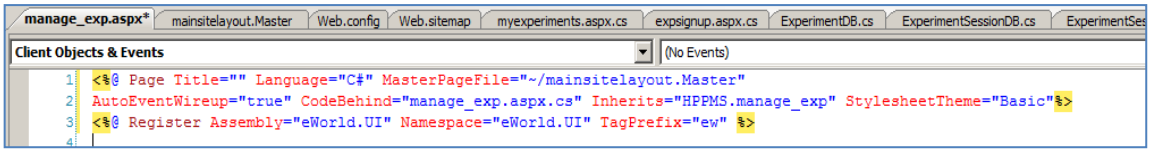


Figure 11. Sample ASP Page Header

Menu Control Adapters are used to create menus dynamically which use CSS extensively. These adapters take the menu definition and dynamically create required HTML using CSS to display the menus. Figure 12 shows a menu in HPPMS application.

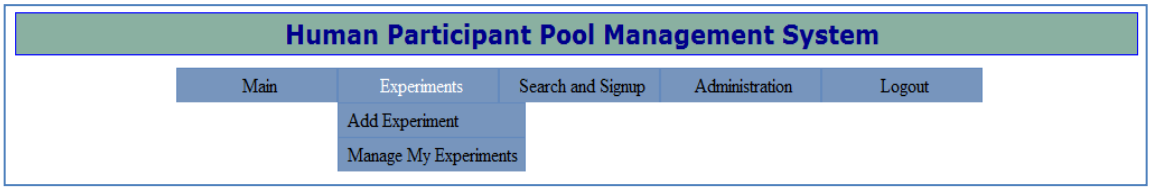


Figure 12. Drop Down Menu

The customer wanted searching pages to be done in a user-friendly manner and also wanted participants be able to search using multiple fields. Based on this requirement, a search page is developed to include search based on title, location, description, special requirements and start/end dates. This page will display experiments based on different combinations of search parameters. Optionally, a user can browse through all experiments by not entering any search criteria. Figure 13 shows the 'Search Experiment' screen where participants search for experiments to signup.

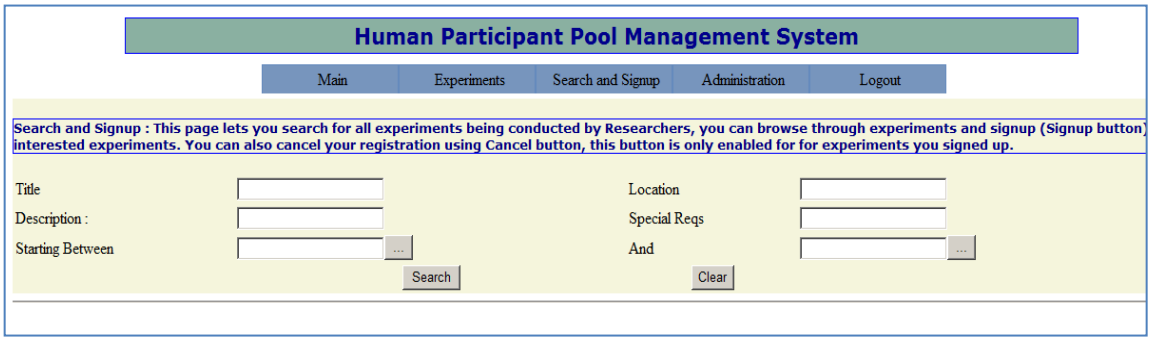


Figure 13. Search Experiment Screen

There are many pages in HPPMS where data is displayed in a grid format. Customer wanted the ability to sort the grid data and also the ability to export grid data to Microsoft Excel format. Sort options and 'Export to Excel' options are available on multiple pages in the application; for example, a researcher can export attendance data to an Excel document.

5.2.Database Interface

Oracle ODP.Net is used for database interaction. Oracle Data Provider for .NET (ODP.NET) features optimized ADO.NET data access to the Oracle database. ODP.NET allows developers to take advantage of advanced Oracle database functionality. Database interface is developed to be SQL injection free. In applications where strings are used to process SQL statements using data entered by an end user, there is a risk of user injecting

additional SQL into user input and thus corrupting the data in the database. Figure 14 shows an example where a string variable is used to form SQL statement and strParamID is used as parameter. If we are taking strParamID from user input and if user enters *'doe.jane;truncate table experiment'*, then there is a chance of losing data in the experiment table.

```
public String GetEmailID(String strParamID)
{
    String strEmailID = "";
    OracleConnection objConnection = new OracleConnection();
    try
    {
    OracleCommand cmd = new OracleCommand();
    objConnection = m_dbConnection.GetOracleConnection();
    cmd.Connection = objConnection;
    cmd.CommandText = "select emailid from userdefn where lower(studentid)=lower('" + strParamID + "')";
    cmd.CommandType = CommandType.Text;

        objConnection.Open();
        OracleDataReader reader = cmd.ExecuteReader();
        if (reader.Read())
        {
```

Figure 14. SQL execution using string

Wherever SQL statements are executed based on user input, code is written to use bind variables rather than string based SQL statements. A sample code is shown in Figure 15.

```
521     */
522     protected bool DoesSessionExist(String strParamExpID, Decimal dParamSessID) {
523         bool bExists = false; /* Does not Session - Assume*/
524         DBConnection db = new DBConnection();
525         OracleDataReader objReader;
526         OracleConnection connection = db.GetOracleConnection();
527         try
528         {
529             connection.Open();
530             String strSQLSelect = "SELECT 'EXISTS' FROM EXP_SESSIONS where EXPID=:pexpid and SESSIONID=:psessid";
531
532             OracleCommand command = new OracleCommand(strSQLSelect, connection);
533             OracleParameter paramExpID = command.CreateParameter();
534             paramExpID.ParameterName = "pExpID";
535             paramExpID.OracleDbType = OracleDbType.Varchar2;
536             paramExpID.Value = strParamExpID;
537             command.Parameters.Add(paramExpID);
538
539             OracleParameter paramSessionID = command.CreateParameter();
540             paramSessionID.ParameterName = "psessid";
```

Figure 15. Sample Database Interface Code

HPPMS is a multi-user system, where multiple users can interact with the system at the same time. For example, multiple participants can search and signup for experiments at the same time. In order to support this, a pessimistic update approach is used for update operations to handle concurrency. For example, if two users try to sign-up for the same experiment where only one slot is available, a check is made to make sure that there is an empty slot before signing up participant.

5.3. Security

Authentication

The customer wanted all users to use their UW-L network id and password to login into application. HPPMS uses UW-L Active Directory for authenticating users. Passwords are not stored in HPPMS.

Authorization

Every user is assigned set of roles in HPPMS. Once users are successfully logged into the application, they are directed to appropriate pages based on their roles. When a user requests a page in the application, a security check is made to make sure that the user has proper access to the corresponding page. A user with administrator access can change the roles of others. The system is initialized with a pre-defined set of administrator roles. But other parts of the system are accessible to everyone in the university who has a valid UW-L network id and password.

5.4. Testing

The application was tested using Internet Explorer and Apple Safari browsers.

Unit Testing

The application was unit tested by the developer. Several dummy users were created with Participant, Researcher and Administrator roles and used for unit testing. HPPMS

uses email for communicating with the users at various events in the system. Since this application was developed off line on the developer's machine, Google email server was used as a test email server. Once the application was deployed to a UW-L server, the university email server will be used. The actual Active Directory server was also not available in development environment. Alternate methods are written to skip authentication for the test users.

Customer Testing

The developer scheduled multiple testing sessions with customer for system testing. Once the application is deployed onto a UW-L server, the customer was given access to test the basic functions. The initial testing was done by a faculty member from the Psychology department and further testing was done to include actual users. UW-L Active Directory server and UW-L email server were used for system testing. The system testing environment used the Capstone oracle database at UW-L.

6. Challenges

This section describes some of the challenges encountered by the developer during the project life cycle.

- The customer was not familiar with software development process and did not have adequate software development experience. This posed a challenge for the developer while explaining some features during requirements gathering phase. The development of the initial prototype really helped at this stage. The customer could then see a working system through the first prototype and they could articulate the next set of features easily.
- There were significant delays in getting required development resources in time which caused delays in multiple phases of project.
- The customer initially wanted this application be developed using ASP.Net and SQL Server. After two development iterations, the customer wanted this application to be deployed onto Oracle database. This caused some major rework in application development. In particular, the data layer of the application was redeveloped to work with Oracle database. As a result of this change, most of the functions in HPPMS application can work with both MS SQL Server and Oracle Database.
- The developer was not familiar with web programming at the beginning of the project. Learning web programming and developing the application was therefore a challenge for the developer at the beginning.
- Due to security constraints by ITS, the Active Directory can only be interfaced with ITS machines, and so unit testing of the Active Directory using the developer's machine was not possible. All Active Directory functions had to be tested only using ITS machine. Alternate methods had

to be used for unit testing the application. Also, there were no test accounts in Active Directory to verify user's authentication, and hence only the developer's credentials were used during unit testing.

7. Continuing Work

The following are expected to be added in the near future:

- Though HPPMS implements most of the functions requested by the customer, there are few more requirements that are not implemented in the current version of the application. These are listed below as continuing work:
- HPPMS can be further integrated with ITS system to update credits towards student courses when an experiment is closed. This automates the credit assignment process. Using this system, a researcher only needs to export student participation data and provide this to faculty advisor, who uses this information to provide credits in a different system manually. This can be automated so that credit information can be assigned in HPPMS and transferred to other systems.
- Email communication can be changed to make use of XML / HTML in order to make it more user-friendly.
- The application can be made available to other departments in the university. This may require a major revision to some of the requirements which are department-specific.
- Additional security as per UW-L ITS guideline needs to be implemented to enable the application to be used over internet.
- This application can be extended for multiple customers or universities. This kind of work is a common requirement for many universities. This application can be changed to store data from multiple universities and can be hosted in a cloud. Such a networked application can easily become a Software As Service (SAS) product. The customers can pay based on number of users and faculty.

8. Conclusion

HPPMS application was created to automate the research projects in the Psychology Department at the University of Wisconsin-La Crosse. A research project termed as ‘experiment’ includes several experiment administration activities. These tasks include creating, posting and administering experiments as well as manipulating participation of students in the experiments. Email communication from HPPMS will help notify users instantly about different events. Once an experiment is completed, a researcher can update the participation information. Faculty advisers can use student participation information to provide credits towards the courses registered by the participants. This is not supported in the existing paper based approach. Since experiment data and research participation data are stored in the database, this information can be retrieved at later time, if required.

BIBLIOGRAPHY

1. UML 2 Toolkit, Hans-Erik Eriksson (Author), Magnus Penker (Author), Brian Lyons (Author), David Fado (Author), 2nd Edition, 2004.
2. Pro ASP.NET 3.5 in C# 2008, Matthew MacDonald and Mario Szpuszta, 2nd Edition, 2008Active Directory / LDAP.Net.
3. Metrics and Models In Software Quality Engineering, Stephen H. Kan, 2nd Edition, 2002.
4. Inside C#, Tom Archer, Sep 2009.
5. http://www.oracle.com/technetwork/topics/dotnet/index-085163.html, Oracle, Oracle Data Provider for .NET, Nov 2010.
6. IEEE Recommended Practice for Software Requirements Specifications - 830-1998, IEEE Computer Society, 1998.
7. http://www.softpanorama.org/SE/software_life_cycle_models.shtml, Software Life Cycle Models, August 12, 2009.

APPENDIX A: Selected HPPMS Screen Shots

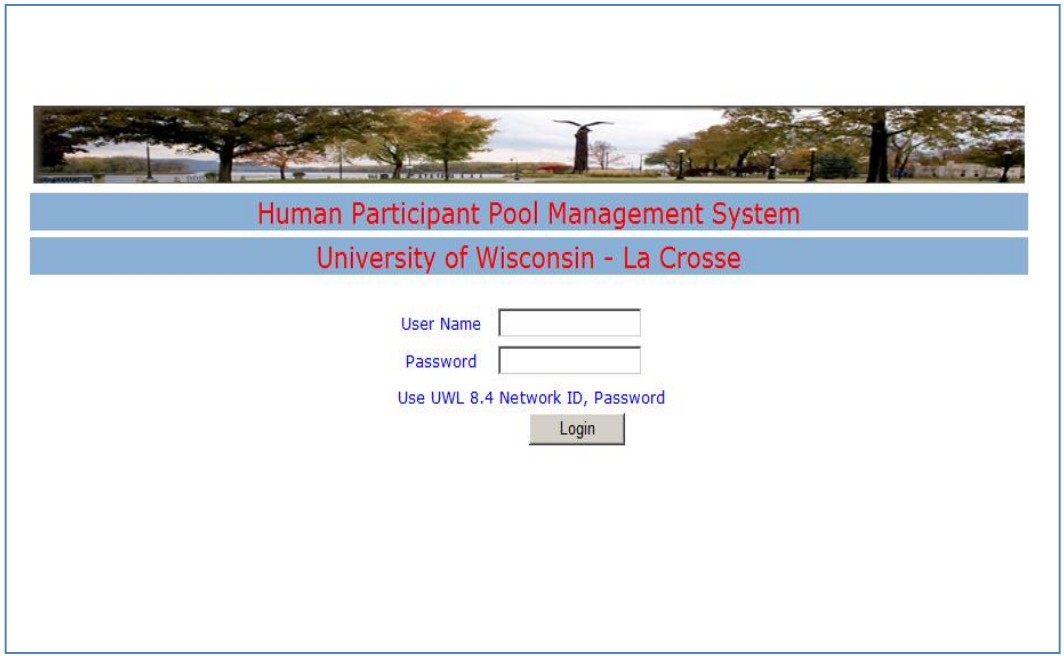
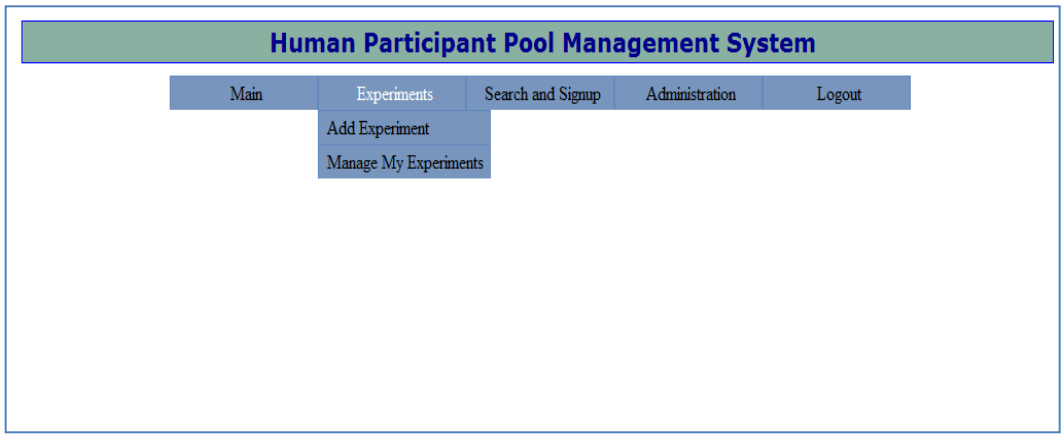


Figure 16. Login Screen



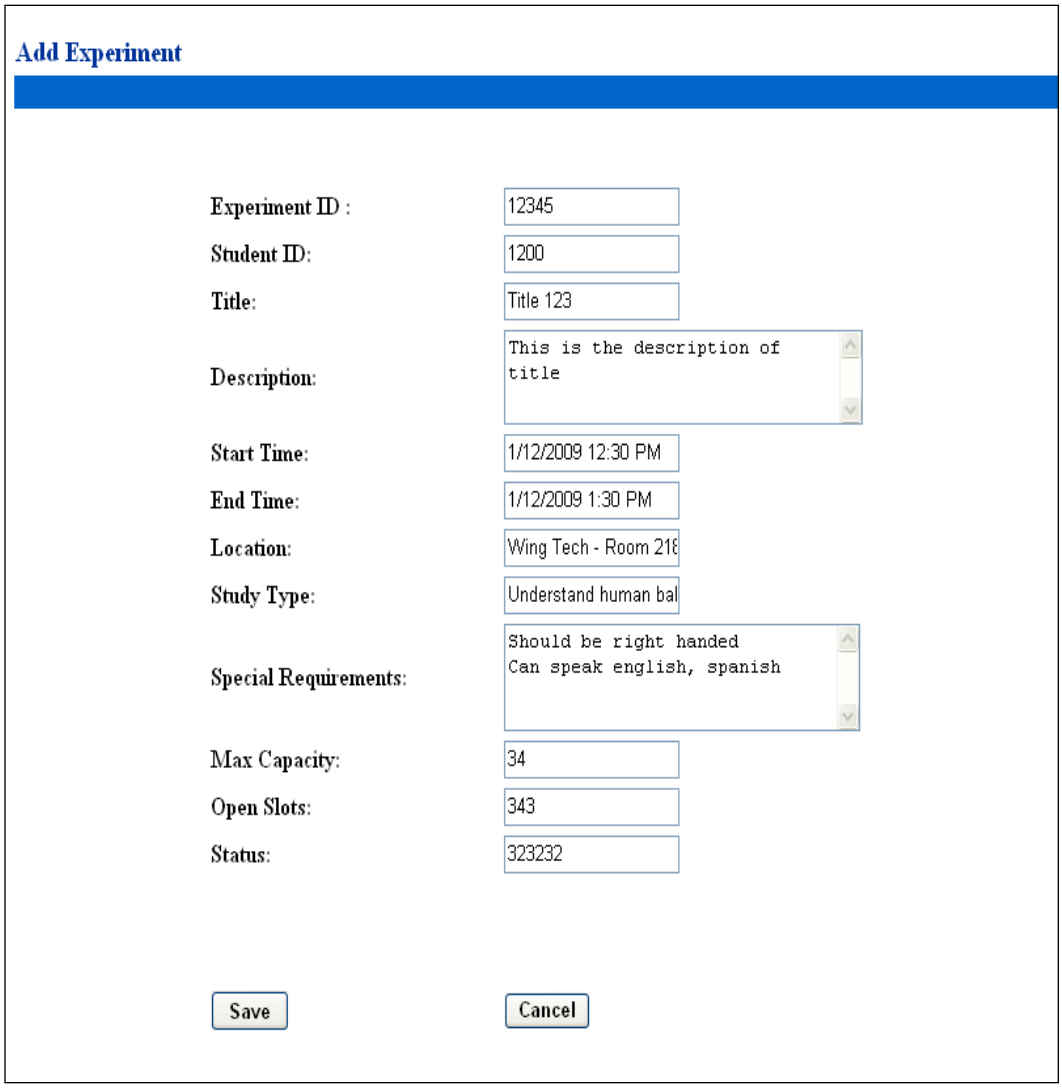


Figure 17. Add Experiment Screen from Initial Prototype

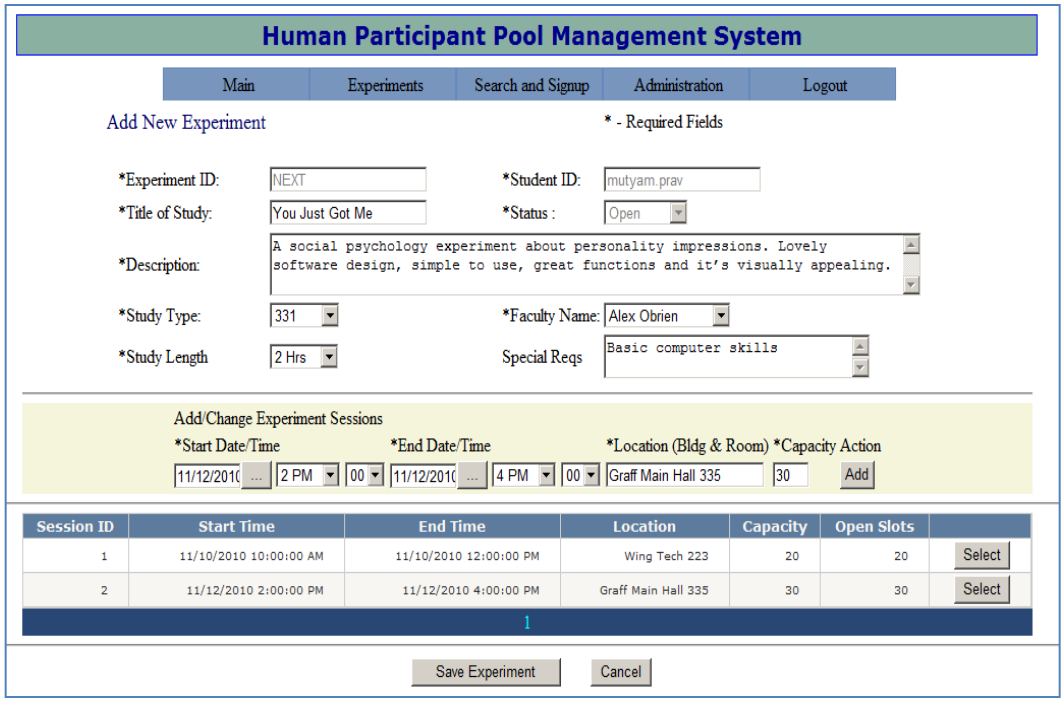


Figure 18. Final Screen to Add Experiment.

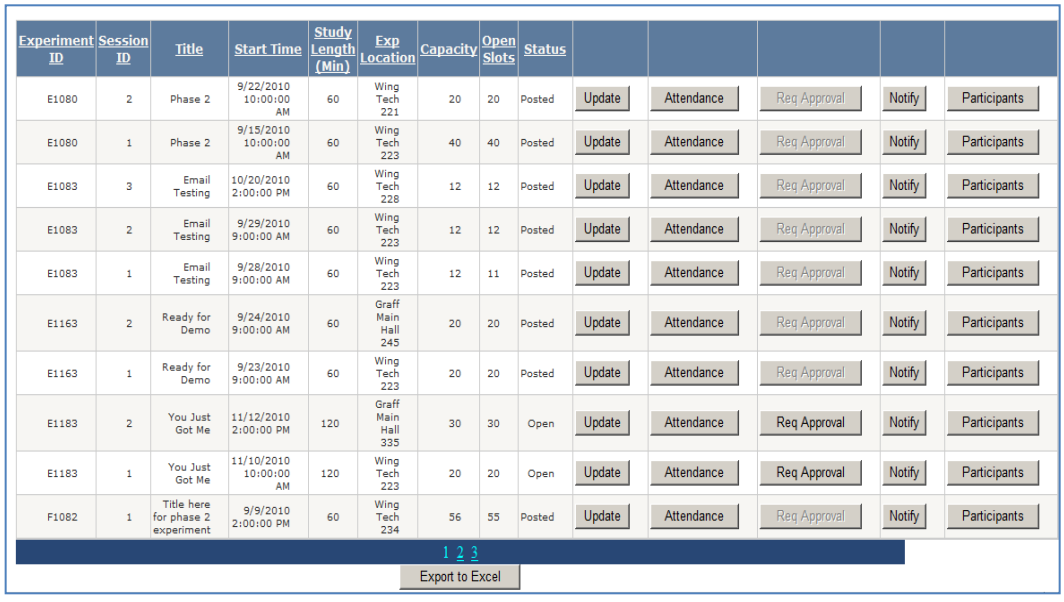


Figure 19. My Experiments Screen

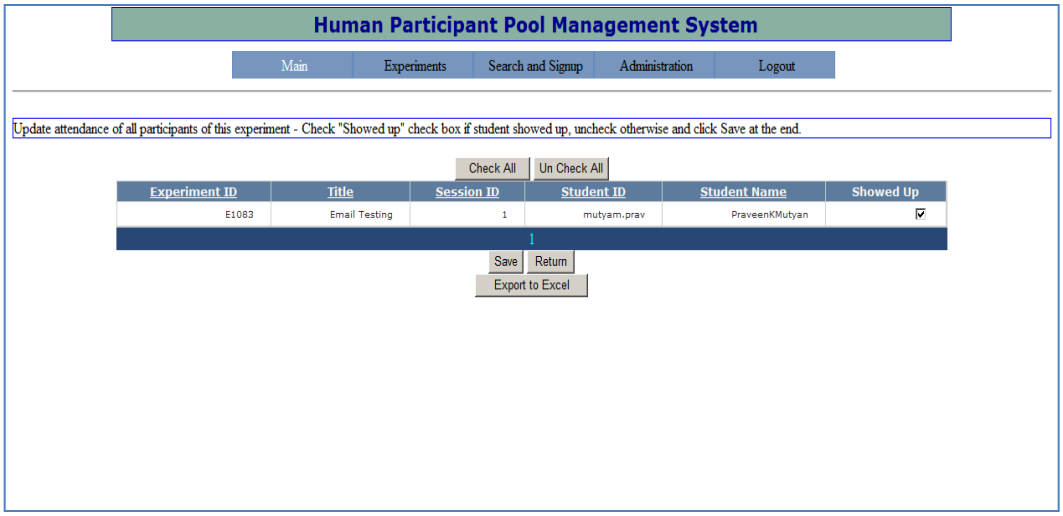


Figure 20. Update Attendance Screen