Panther Buddy - Student Information Dashboard

CEN 5011 Advance Software Engineering

Prof. Peter Clarke

Team 6:

Abhinav Dutt

Allan Shaji Manamel

Sharat Kedari

Yulong Qiu

Abdur Rahman Bin Shahid

Yue Wu

James Angelo

October 5, 2015

• Abstract

This document presents the results of the requirements and analysis phase for the development of a Student information system called "Panther Buddy". Panther buddy is a web application through which an user can get all the relevant information about their study, living, etc. at FIU or enquire for information regarding it.

For this project, we have created use case models for the major functionalities of the panther buddy system. Scenarios, object diagrams, sequence diagrams, and activity diagrams have been created for ten of the most important use cases.

During requirement elicitation and analysis, the team got together and defined the actors, scenarios and use cases and formalized them using the use case, object, class and sequence diagram.

Table of Contents

CHAPTER 1

1. Introduction

This chapter covers an overview of the system, which contains the purpose and the scope of the system. Furthermore, it will provide the different terminologies that are required to become familiar with the system, along with a brief explanation of what to expect in the next chapters of this document.

1.1 Purpose of system

This project consists of the creation of web application called Panther Buddy, which would help any person who wants to access/ post relevant information regarding FIU to the message board. Any user can register with panther buddy application, and can have access to the posts made in the system and also can post relevant information.

1.2 Scope of system

When delivered the system will allow any user to get registered with panther buddy application.

The system will allow users to view and post messages in the panther buddy application, users can also delete his/her own messages, users can also view others profile.

1.3 Definitions, acronyms, and abbreviations

FP	Function point
SVN	Subversion – version control system
UML	Unified Modeling Language
XSS	Cross-site scripting
ILF	Internal Logical Files
EIF	External Interface Files
EI	External Inputs
EO	External Outputs
EQ	External Inquiries
PM	Person month
JSF	Java server Faces
HTML	Hyper Text Markup Language

1.4 **Overview of document**

In chapter 2, we will analyze the current system to understand its limitations and problems. Section 3, will provide a description of the project plan, where you will find the assignment of roles, the identified milestones, the cost estimation, hardware and software requirements, and deliverables. Section 4, contains a high level description of the proposed functionality that will include non-functional requirements relating to the areas of usability, reliability, performance, supportability and implementation. Additionally, there will be a use case diagram depicting how the different users of the system interact with the functionality provided by the system. Moreover, in section 5, you will find a variety of models depicting the static and dynamic views of the system. Lastly, you will find a glossary of terms used in this document and an appendix with more details on the project schedule, uses cases, user interface designs and diary of meetings and tasks.

CHAPTER 2

2. Current System

There are systems currently that do a similar job, but they are distributed in functionality i.e. they deal only with a particular line of queries for e.g. furniture, books, etc. We wish to collate these different lines of queries in a single place for the convenience of the user.

- **Possible Downtime**:

There will be maintenance downtime for one hour on every first of the month.

- **Technical Issues:**

You should be aware of the fact that this technology is always prone to outages and other technical issues. Besides, you will need an Internet connection to be logged onto the system at all times. You will invariably be stuck in case of network and connectivity problems.

CHAPTER 3

3. Project Plan

3.1 Introduction

The project plan is explained in this part of the document, this includes the roles assigned to every team member, the hardware and software needed to complete the project and the work breakdown that will be presented in a Gantt chart. The tasks presented in this chart may change and new tasks might be added and another might be deleted or modified.

3.2 Project organization – assignment of roles for the entire project.

The organizational structure of the roles involved in this project for each of the three phases of the project is described below:

	Phase	1					
	Role	**Lead**	**Document Writer**	**Modeller**	**Presentation**	**Time Keeper**	**Minute taker**
Members	Abhinav Dutt	X		X			
	Allan Manamel			X			
	Abdur Rahman Bin Shahid			X			
	Sharat Kedari					X	X
	April Wu		X				
	Yulong Qui		X				
	James Angelo				X		

	Phase	2						
	Role	**Lead**	**Architect**	**Developer**	**Technical writer**	**Presentation**	**Time Keeper**	**Minute taker**
Members	Abhinav Dutt			X				
	Allan Manamel	X	X	X				
	Abdur Rahman Bin Shahid			X				
	Sharat Kedari			X				
	April Wu				X	X	X	X
	Yulong Qui			X				
	James Angelo			X	X			

	Phase	3						
	Role	**Lead**	**Developer**	**Technical writer**	**Tester**	**Presentation**	**Time Keeper**	**Minute taker**
Members	Abhinav Dutt		X					
	Allan Manamel		X	X				
	Abdur Rahman Bin Shahid	X			X			
	Sharat Kedari		X	X				
	April Wu				X			
	Yulong Qui				X	X		
	James Angelo				X		X	X

- **Lead**: Oversees all project tasks and ensures all milestones are reached. He/she will be primary facilitator for the group meetings, questions regarding the project, and assigns work to each member of the group. And also work with some scenarios, use cases and diagrams.

- **Document writer**: Will oversee and do the creation of the required documents for the phase of software engineering life cycle.
- **Modeller**: Handles the modelling of the system. Each of the members will do some use cases and scenarios and their related diagrams. Finally, the lead combines all the diagrams in one model and validates the model.
- **Presentation**: The person will do the creation of the presentation for the deliverable.
- **Time keeper**: Is responsible for keeping track of time and notifying the facilitator if a discussion consumes more time than is allocated. A vote might be required to continue discussion or move on to the next point.
- **Minute keeper**: Is responsible for recording the meeting i.e. information for the diary.
- **Architect**: Will be responsible for architecture design and implementation for the software.
- **Developer**: Will help the architect in developing the software.
- **Document writer**: Will oversee and do the creation of the required documents for the phase of software.
- **Tester**: Will test the system for validation and verification.

3.3 Cost Estimation

This section shows the cost estimation done for the project. We have used COCOMO 2 Early Design model to estimate the cost of the project.

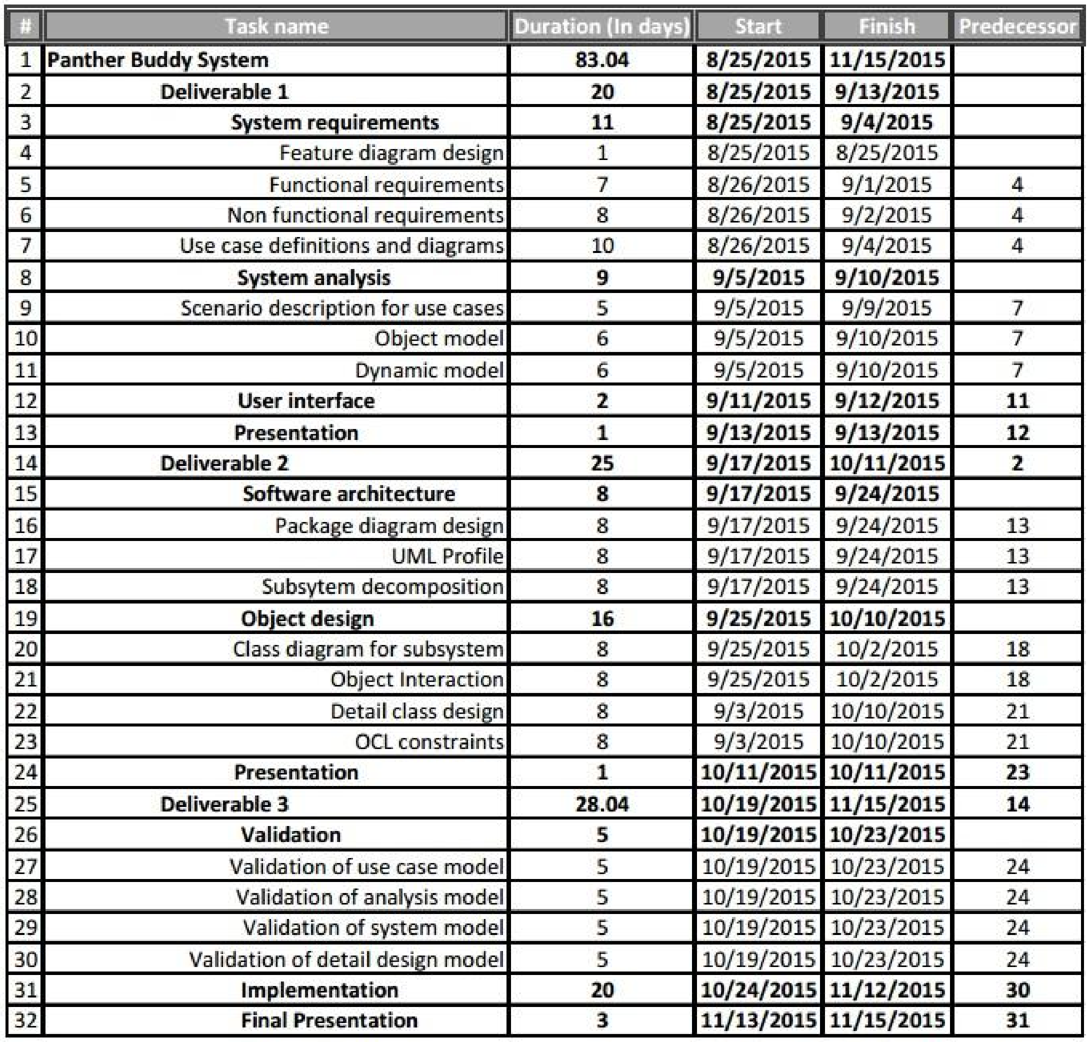
The function point cost estimation approach is based on the amount of functionality in a software project and a set of individual project factors. Function points are useful estimators since they are based on information that is available early in the project life cycle.

Function points measure a software project by quantifying the information processing functionality associated with major external data or control input, output, or file types. Five user function types should be identified as defined in table below.

External Input (Inputs)	Count each unique user data or user control input type that (i) enters the external boundary of the software system being measured and (ii) adds or changes data in a logical internal file.
External Output (Outputs)	Count each unique user data or control output type that leaves the external boundary of the software system being measured.
Internal Logical File (Files)	Count each major logical group of user data or control information in the software system as a logical internal file type. Include each logical file (e.g., each logical group of data) that is generated, used, or maintained by the software system.
External Interface Files (Interfaces)	Files passed or shared between software systems should be counted as external interface file types within each system
External Inquiry (Queries)	Count each unique input-output combination, where an input causes and generates an immediate output, as an external inquiry type.

We will use the use cases defined for the system to focus on the core functionality that should be provided by the system to decide upon the functional points to be used to estimate the cost

Below is the project task timelines:



#	Task name	Duration (In days)	Start	Finish	Predecessor
1	**Panther Buddy System**	**83.04**	**8/25/2015**	**11/15/2015**	
2	**Deliverable 1**	**20**	**8/25/2015**	**9/13/2015**	
3	**System requirements**	**11**	**8/25/2015**	**9/4/2015**	
4	Feature diagram design	1	8/25/2015	8/25/2015	
5	Functional requirements	7	8/26/2015	9/1/2015	4
6	Non functional requirements	8	8/26/2015	9/2/2015	4
7	Use case definitions and diagrams	10	8/26/2015	9/4/2015	4
8	**System analysis**	**9**	**9/5/2015**	**9/10/2015**	
9	Scenario description for use cases	5	9/5/2015	9/9/2015	7
10	Object model	6	9/5/2015	9/10/2015	7
11	Dynamic model	6	9/5/2015	9/10/2015	7
12	**User interface**	**2**	**9/11/2015**	**9/12/2015**	**11**
13	**Presentation**	**1**	**9/13/2015**	**9/13/2015**	**12**
14	**Deliverable 2**	**25**	**9/17/2015**	**10/11/2015**	**2**
15	**Software architecture**	**8**	**9/17/2015**	**9/24/2015**	
16	Package diagram design	8	9/17/2015	9/24/2015	13
17	UML Profile	8	9/17/2015	9/24/2015	13
18	Subsytem decomposition	8	9/17/2015	9/24/2015	13
19	**Object design**	**16**	**9/25/2015**	**10/10/2015**	
20	Class diagram for subsystem	8	9/25/2015	10/2/2015	18
21	Object Interaction	8	9/25/2015	10/2/2015	18
22	Detail class design	8	9/3/2015	10/10/2015	21
23	OCL constraints	8	9/3/2015	10/10/2015	21
24	**Presentation**	**1**	**10/11/2015**	**10/11/2015**	**23**
25	**Deliverable 3**	**28.04**	**10/19/2015**	**11/15/2015**	**14**
26	**Validation**	**5**	**10/19/2015**	**10/23/2015**	
27	Validation of use case model	5	10/19/2015	10/23/2015	24
28	Validation of analysis model	5	10/19/2015	10/23/2015	24
29	Validation of system model	5	10/19/2015	10/23/2015	24
30	Validation of detail design model	5	10/19/2015	10/23/2015	24
31	**Implementation**	**20**	**10/24/2015**	**11/12/2015**	**30**
32	**Final Presentation**	**3**	**11/13/2015**	**11/15/2015**	**31**

Below are the worksheet for the project estimation calculation.

1. Functional points and their weightage

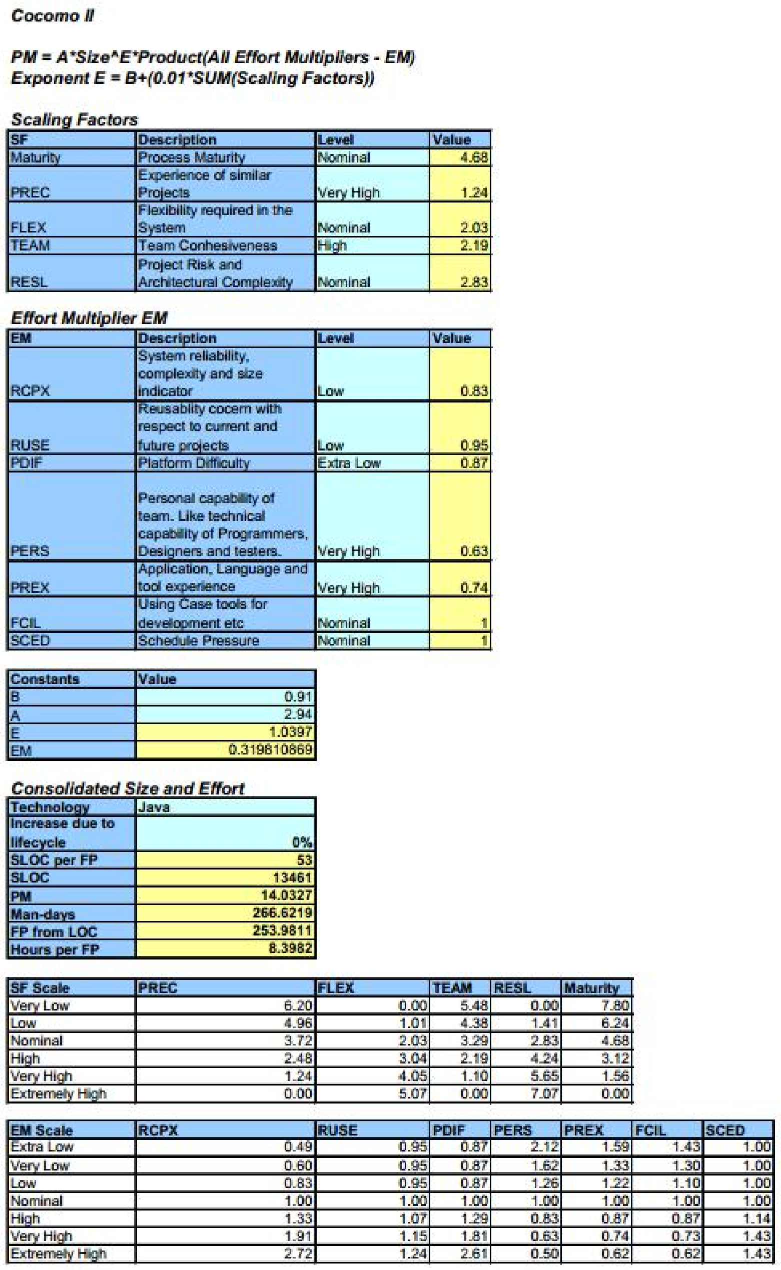
			ILF			EIF			EI			EO			EQ			
Features	**Comments**	**Reference**	Low	Avg	High	Low	Avg	High	Low	Avg	High	Low	Avg	High	Low	Avg	High	***Total FPs***
Login	The function of login by user		0	2	0	0	0	0	0	0	0	1	0	0	1	1	0	31
Logout	The function for logging out a user		1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	11
User registration	The function of registering a user		0	1	1	0	0	0	1	0	0	1	0	0	1	1	0	39
Forgot password	The function for forgot password		1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	29
View posts	Function to view already posted messages		0	2	0	0	0	0	0	0	0	0	1	0	0	1	0	29
Post message	Function to post new messages		1	1	0	0	0	0	0	1	0	0	1	0	1	0	0	29
Delete message	Function to delete an already posted message by the user		1	1	0	0	0	0	1	0	0	1	0	0	1	0	0	27
View other profile	Function to view the profile of another user of the system		2	0	0	0	0	0	0	0	0	1	0	0	1	0	0	21
View own profile	Function to view one's own profile		2	0	0	0	0	0	0	0	0	1	0	0	1	0	0	21
Total Artifacts			8	8	1	0	0	0	2	1	0	7	3	0	7	3	0	

Total FP	**237**

2. Source line of code

Features	**Comments**	**Reference**	Language	Estimated SLOC
User Activation	The function of activating a user on first login		Java	300
Login data filtering	Function to check login data to prevent SQL injection based attack		Java	300
Message filteration	Function to check message being posted for XSS attacks		Java	300
			Total:	**900**

3. Effort estimation



Cocomo II

PM = A*Size^E*Product(All Effort Multipliers - EM)
Exponent E = B+(0.01*SUM(Scaling Factors))

Scaling Factors

SF	Description	Level	Value
Maturity	Process Maturity	Nominal	4.68
PREC	Experience of similar Projects	Very High	1.24
FLEX	Flexibility required in the System	Nominal	2.03
TEAM	Team Conhesiveness	High	2.19
RESL	Project Risk and Architectural Complexity	Nominal	2.83

Effort Multiplier EM

EM	Description	Level	Value
RCPX	System reliability, complexity and size indicator	Low	0.83
RUSE	Reusablity cocern with respect to current and future projects	Low	0.95
PDIF	Platform Difficulty	Extra Low	0.87
PERS	Personal capability of team. Like technical capability of Programmers, Designers and testers.	Very High	0.63
PREX	Application, Language and tool experience	Very High	0.74
FCIL	Using Case tools for development etc	Nominal	1
SCED	Schedule Pressure	Nominal	1

Constants	Value
B	0.91
A	2.94
E	1.0397
EM	0.319810869

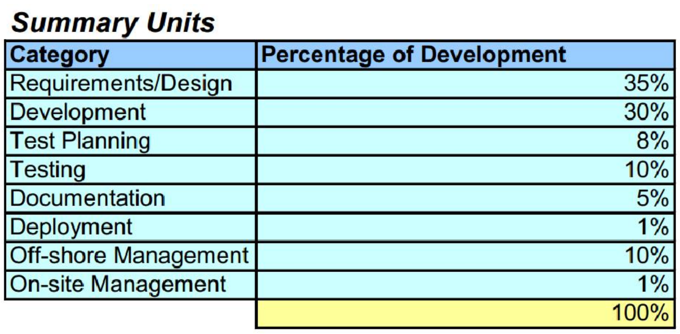
Consolidated Size and Effort

Technology	Java
Increase due to lifecycle	0%
SLOC per FP	53
SLOC	13461
PM	14.0327
Man-days	266.6219
FP from LOC	253.9811
Hours per FP	8.3982

SF Scale	PREC	FLEX	TEAM	RESL	Maturity
Very Low	6.20	0.00	5.48	0.00	7.80
Low	4.96	1.01	4.38	1.41	6.24
Nominal	3.72	2.03	3.29	2.83	4.68
High	2.48	3.04	2.19	4.24	3.12
Very High	1.24	4.05	1.10	5.65	1.56
Extremely High	0.00	5.07	0.00	7.07	0.00

EM Scale	RCPX	RUSE	PDIF	PERS	PREX	FCIL	SCED
Extra Low	0.49	0.95	0.87	2.12	1.59	1.43	1.00
Very Low	0.60	0.95	0.87	1.62	1.33	1.30	1.00
Low	0.83	0.95	0.87	1.26	1.22	1.10	1.00
Nominal	1.00	1.00	1.00	1.00	1.00	1.00	1.00
High	1.33	1.07	1.29	0.83	0.87	0.87	1.14
Very High	1.91	1.15	1.81	0.63	0.74	0.73	1.43
Extremely High	2.72	1.24	2.61	0.50	0.62	0.62	1.43

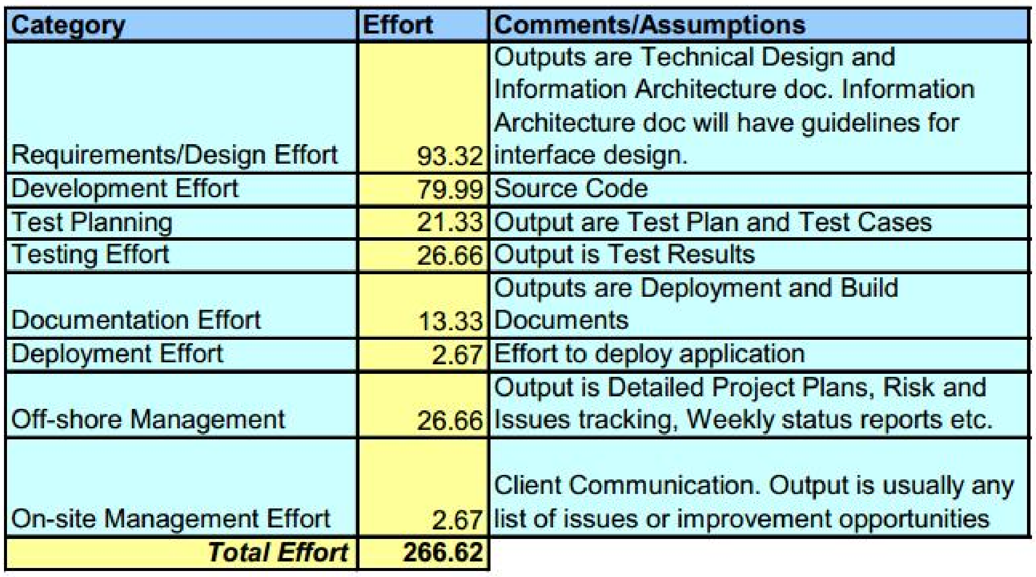
4. Effort distribution



Summary Units

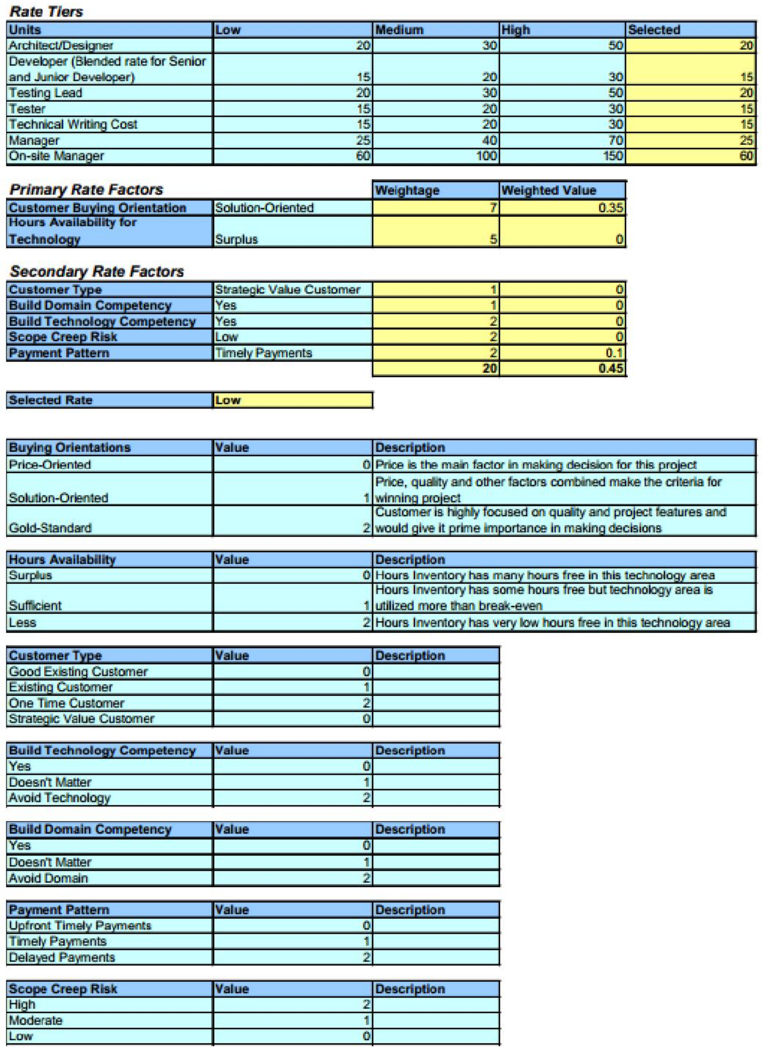
Category	Percentage of Development
Requirements/Design	35%
Development	30%
Test Planning	8%
Testing	10%
Documentation	5%
Deployment	1%
Off-shore Management	10%
On-site Management	1%
	100%

5. Effort summary



Category	Effort	Comments/Assumptions
Requirements/Design Effort	93.32	Outputs are Technical Design and Information Architecture doc. Information Architecture doc will have guidelines for interface design.
Development Effort	79.99	Source Code
Test Planning	21.33	Output are Test Plan and Test Cases
Testing Effort	26.66	Output is Test Results
Documentation Effort	13.33	Outputs are Deployment and Build Documents
Deployment Effort	2.67	Effort to deploy application
Off-shore Management	26.66	Output is Detailed Project Plans, Risk and Issues tracking, Weekly status reports etc.
On-site Management Effort	2.67	Client Communication. Output is usually any list of issues or improvement opportunities
Total Effort	**266.62**	

6. Rates



Rate Tiers

Units	Low	Medium	High	Selected
Architect/Designer	20	30	50	20
Developer (Blended rate for Senior and Junior Developer)	15	20	30	15
Testing Lead	20	30	50	20
Tester	15	20	30	15
Technical Writing Cost	15	20	30	15
Manager	25	40	70	25
On-site Manager	60	100	150	60

Primary Rate Factors		Weightage	Weighted Value
Customer Buying Orientation	Solution-Oriented	7	0.35
Hours Availability for Technology	Surplus	5	0

Secondary Rate Factors

Customer Type	Strategic Value Customer	1	0
Build Domain Competency	Yes	1	0
Build Technology Competency	Yes	2	0
Scope Creep Risk	Low	2	0
Payment Pattern	Timely Payments	2	0.1
		20	0.45

Selected Rate	Low

Buying Orientations	Value	Description
Price-Oriented	0	Price is the main factor in making decision for this project
Solution-Oriented	1	Price, quality and other factors combined make the criteria for winning project
Gold-Standard	2	Customer is highly focused on quality and project features and would give it prime importance in making decisions

Hours Availability	Value	Description
Surplus	0	Hours Inventory has many hours free in this technology area
Sufficient	1	Hours Inventory has some hours free but technology area is utilized more than break-even
Less	2	Hours Inventory has very low hours free in this technology area

Customer Type	Value	Description
Good Existing Customer	0	
Existing Customer	1	
One Time Customer	2	
Strategic Value Customer	0	

Build Technology Competency	Value	Description
Yes	0	
Doesn't Matter	1	
Avoid Technology	2	

Build Domain Competency	Value	Description
Yes	0	
Doesn't Matter	1	
Avoid Domain	2	

Payment Pattern	Value	Description
Upfront Timely Payments	0	
Timely Payments	1	
Delayed Payments	2	

Scope Creep Risk	Value	Description
High	2	
Moderate	1	
Low	0	

7. Cost

Cost Units

Units	per hour	per days
Architect/Designer	20	160
Developer (Blended rate for Senior and Junior Developer)	15	120
Testing Lead	20	160
Tester	15	120
Technical Writing Cost	15	120
Manager	25	200
On-site Manager	60	480

Total Project Cost

Category	Effort (Days) Estimated	Effort (Days) to Quote	Cost
Requirements/Design Effort	93.3	39.0	$6,240
Development Effort	80.0	19.0	$2,280
Test Planning	21.3	2.0	$320
Testing Effort	26.7	3.0	$360
Documentation Effort	13.3	5.0	$600
Deployment Effort	2.7	1.0	$160
Off-shore Management	26.7	3.0	$600
On-site Management Effort	2.7	1.0	$480
Total	**266.6**	**73.0**	**$11,040**
		Blended Rate	**$18.90**

Total Support Cost

Man Months	1		
Man Days	5		
Category	**Hours per day**	**Total Hours**	**Cost**
Designers	1	5	$100
Developers	8	40	$600
Testers	1	5	$75
Managers	0.5	2.5	$63
		Total	**$838**

8. Plan

Assumptions

Test Planning, Documentation and Management activities are carried out in parallel to other main activities of design, development and testing

Resources

Designers	3
Developers	5
Testers	3

Others

Possible Critical Chain Delay in Development	5%
Possible Fixation Delay in Testing	5%
Working days in a month	30
Ratio of calendar days	1.033333333

Slacks

Initial Ramp-up	0.5
Deployment for testing etc	0.5
Code/Peer Reviews	1
Shipment Packaging & Review	1
Total Slack	3

Project Plan - Waterfall

Phase	Duration (days)	Duration (months)
Requirements/Design Phase	13.0	0 Month 13 Day
Development Phase	4.0	0 Month 4 Day
Testing Phase	1.1	0 Month 1 Day
Total	**18.0**	**0 Month 19 Day**
Slack	**3.0**	
Grand Total	**21.0**	**0 Month 22 Day**

Iterative Plan

Iterations	1	Iteration Duration (without Slacks)	Requirements/Design	Development	Testing
1	100%	18.04	13.0	4.0	1.1
2	0%	0.00	0.0	0.0	0.0
3	0%	0.00	0.0	0.0	0.0
4	0%	0.00	0.0	0.0	0.0
5	0%	0.00	0.0	0.0	0.0
Total	100%				

3.4 Hardware and software requirements to complete the project.

Hardware: This project requires minimum 512 MB hard disk space. It requires minimum 512 MB of RAM but we recommended 1GB of memory for best performance.

Software:

Microsoft Windows 7, 8 or 10 (32 bit or 64 bit).

Microsoft Word 2010/2013

Microsoft Excel 2010/2013

Microsoft PowerPoint 2010/2013

Adobe Acrobat Reader 10

Eclipse Luna 4.4.2

Papyrus SDK plugin (Version 1.0.2)

Graphical Modeling Framework (GMF Version 3.0.1)

Eclipse Modeling Framework (EMF Version 1.5.1)

Subclipse 1.10 SDK plugin

Tortoise SVN

Assembla SVN repository

Asana project management tool

GanttPro Gantt chart tool

LeanTesting test management tool

MySql Database server 5.6

Wildfly 8 webserver

Selenium browser automation 2.47

3.5 Work breakdown (See Appendix A)

Tasks and Milestones: The tasks in the development of Panther Buddy System were divided into three milestones

Milestone 1 consisted of the completion of the Use Case Phase and the Analysis Phase. This resulted in a software requirements document handed in to the client. It also covered Object and Dynamic models.

Milestone 2 will consist of the completion of Design Phase and start on the model driven software development approach. After generating models and code, the necessary transformations will be made. This will result in a design document handed to the client. Finally, **Milestone 3** will consist of the completion of the Testing Phase as well as the completion of the entire project.

Please refer to Appendix A for project schedule and Appendix D for diary.

CHAPTER 4

4. Requirement Elicitation

In this section we will describe the functional and nonfunctional requirements of the system so we can justify our solution. In addition we will talk about specific functions that the system must perform together with client-defined constraints.

4.1 Overview

In the following two subsections we will discuss the overall functionality of the system, what it should do and how it should do it.

4.2 Functional Requirements – describes high-level functionality. Refer to the use cases in the Appendix.

√√ The system shall allow users to register to the system, activate himself in the system, login and logout from the system.

√√ The system shall allow users to view profiles and contact information of users, change user password and recover his password from the system.

√√ The system shall allow users to post, view, and delete message.

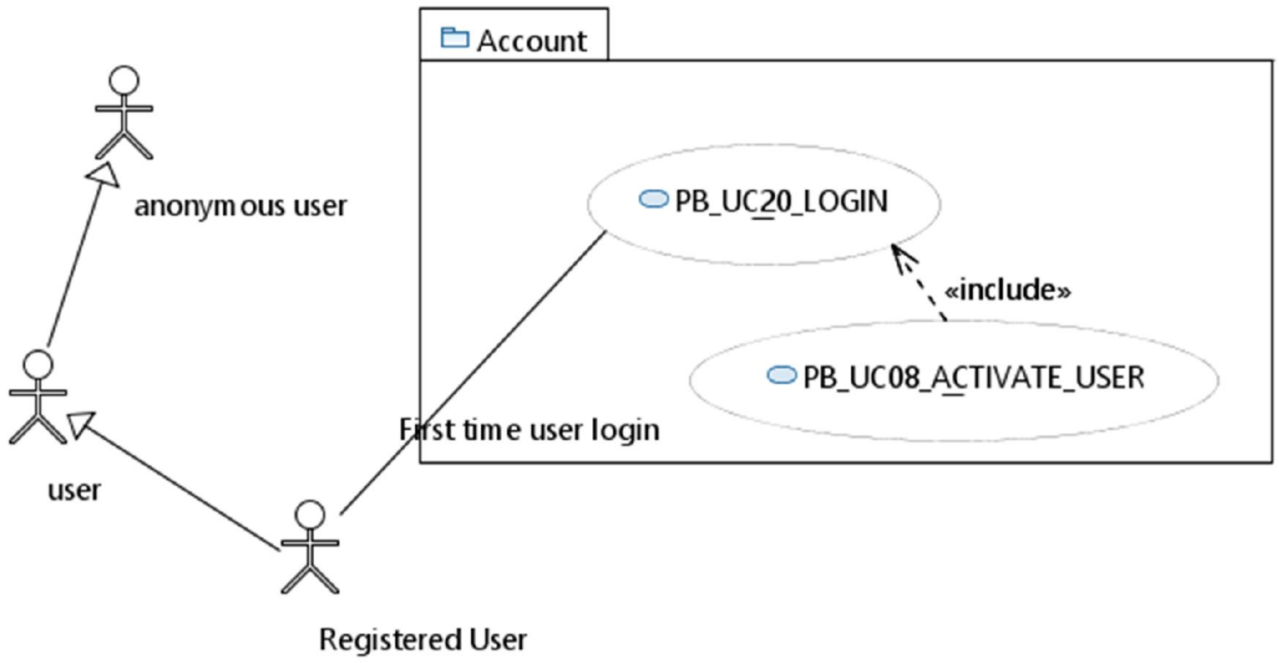
4.3 Non-functional requirements

The user level requirements not directly related to functionality are:

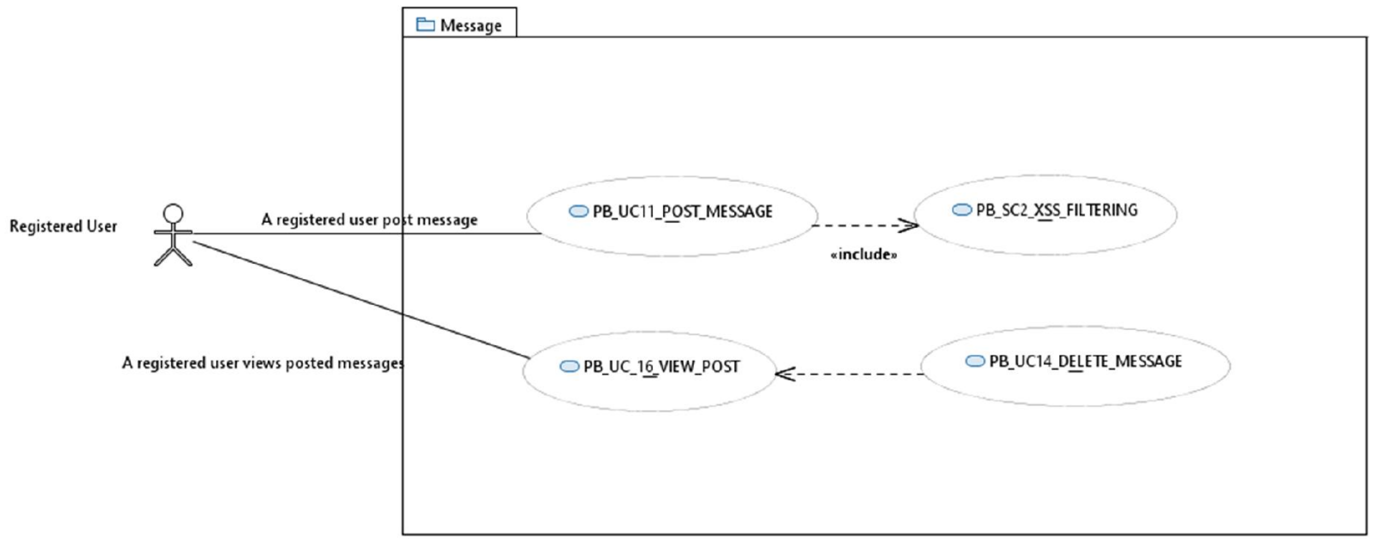
1. Usability
   a) A user will not require any prior training.
2. Reliability
   a) Mean time to failure – one failure for every 20 uses is acceptable.
   b) Availability – the panther buddy environment is always available to the user barring some technical issues for which the software is not responsible.
3. Performance
   a) Navigation between pages should happen within 2 secs on a 45Mbps internet connection.
4. Supportability
   a) The use cases should be able to be reproduced in any session of panther buddy environment.
5. Implementation
   a) The software application must be developed using the Java.
   b) The database should be MySql Dataserver 5.6
   c) The application should be deployed in JBoss WildFly 8 webserver.

4.4 Use case model. Refer to the use cases in the appendix.

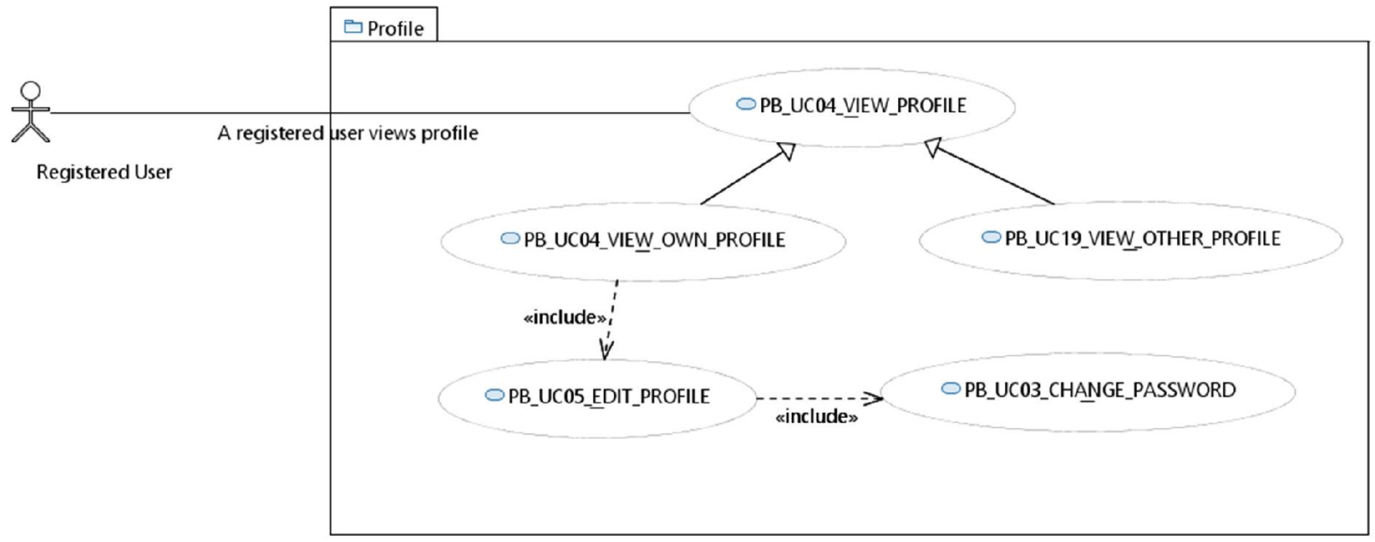
Activate User



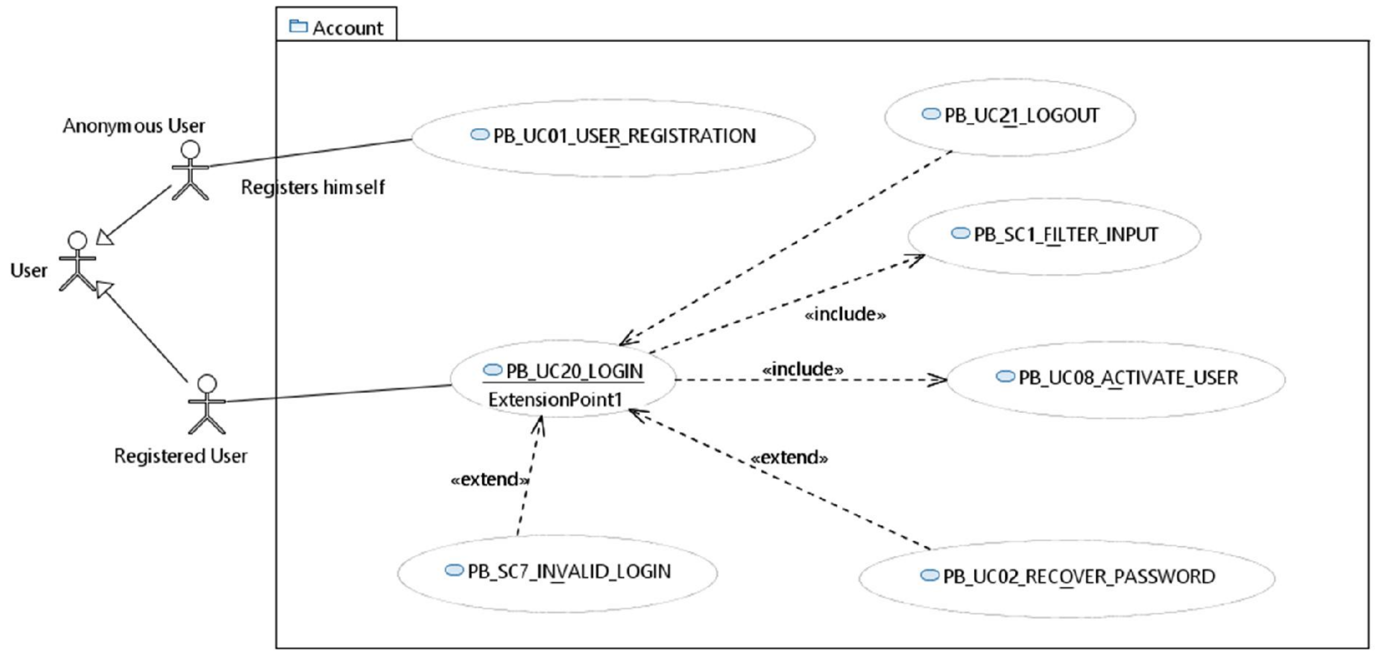
View, Post, Delete and XSS Filtering



View Profile and Change Password



Registration, Login, Logout, Recover Password and SQL Injection Security



CHAPTER 5

5. Requirement Analysis

In this chapter we present models that show both static and dynamic views of the system.

5.1 Scenarios

5.1.1 Description of ten (10) scenarios.

1. USER LOGIN

Use case ID: PB_UC20_LOGIN

Use case level: System level (End-to-End)

Details:

Actors: Any user.

Pre-conditions: The user has the login page open.

Description: The user logins into the system using his email and password.

1. User enters his email into the email field
2. The user enters his password in the password field
3. The user clicks the login button to submit his data.

Post condition: The user is redirected to the Panther Buddy home page.

Extensions: PB_SC7_INVALID_LOGIN

Exceptions: None

Related use cases: PB_SC1_FILTER_INPUT

Decision support:

Frequency: Moderate - Performed every time a user logins to the Panther buddy system

Criticality: Very Critical - If the user wants to use the 'Panther Buddy' site he needs to be able to login into the system.

Risk: High

Usability: No previous training needed.

Reliability:

Mean time to failure: 1 failure for every 2 months of operation is acceptable.

Performance:

1. The user will be redirected to the home page in 2 seconds

Supportability:

The software will support all browsers that support HTML 5.

Implementation:

Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

--

Modification history

Owner: Allan Shaji Manamel

Initiation date: 09/18/2015

Date last modified: 09/18/2015

Example scenario:

Actor: Allan Shaji Manamel

Pre-conditions: Allan Shaji Manamel has the login page open.

1. Allan enters his email amana100@fiu.edu into the email field.

2. Allan enters his password DX345tv in the password field.
3. Allan clicks the login button to submit the data.

Post condition: Allan Shaji Manamel is redirected to the home page of Panther Buddy system.

2. LOGOUT

Use case ID: PB_UC21_LOGOUT

Use case level: System level (End-to-End)

Details:

Actors: Registered user.

Pre-conditions: The user has logged into the system and is on the home page.

Description: The user logs out from the system.

1. User clicks the logout button on the screen
2. The system destroys the users session and redirects him to the login page

Post condition: The user is redirected to the Panther Buddy login page.

Extensions:

Exceptions:

Related use cases: PB_SC3_SESSION_TIMEOUT

Decision support:

Frequency: High - Performed every time a user logs out from the Panther buddy system

Criticality: low

Risk: low

Usability: No previous training needed.

Reliability:

Mean time to failure: 1 failure for every 2 months of operation is acceptable.

Performance:

1. The user will be redirected to the login page in 2 seconds

Supportability:

The software will support all browsers that support HTML 5.

Implementation:

Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

--

Modification history

Owner: Allan Shaji Manamel

Initiation date: 09/19/2015

Date last modified: 09/19/2015

Example scenario:

Actor: Allan

Pre-conditions: Allan has logged into the system and is on the home page.

1. Allan clicks on the logout button on the page.
2. Allan's session is destroyed and he is redirected to the Panther Buddy login page

Post condition: Allan Shaji Manamel is redirected to the Panther Buddy login page.

3. USER REGISTRATION

Use case ID: PB_UC1_USER_REGISTRATION

Use case level: System level (End-to-End)

Details

Actors: Any anonymous user or an already registered user.

Pre-conditions: The user has the user registration page for panther buddy open in a web browser.

Description: The user registers himself on "Panther Buddy".

1. The user will be asked to enter his details.
   - First name: The first name of the user.
   - Last name: The last name or surname of the user
   - Email id: The email id the user wants to register on 'Panther Buddy' with.
   - Phone number: The user primary phone number.
2. The user clicks the submit button to submit the data.
3. The user data is saved in the database along with a generated password. His status is marked as inactive and a default date value (Sun, 2 Dec 292269055 BC 16:47:04 +0000) is put for his activation date.
4. The user is sent his password by mail.
5. The user is then redirected to the login page.

Post condition: The user is marked as an inactive registered user in database.

Alternative courses of action:

At any time during step 1 the user can click the cancel button to return to the 'Panther Buddy' login page.

Extensions:

Exceptions:

The user at step 1.3 enters an email id that is already used to register by a user in the 'Panther Buddy' system. The user is asked to re-enter a new email id.

Related use cases: PB_UC2_LOGIN, PB_UC3_ACTIVATE_USER, PB_UC4_INVALID_LOGIN, PB_UC5_RECOVER_PASSWORD.

Decision support:

Frequency: Moderate - Performed every time a new user wants to register on 'Panther Buddy' (30 times per day).

Criticality: Very Critical - If the user wants to use the 'Panther Buddy' site he needs to register. We also need to verify the user's data input by him during registration for completeness and malicious activity.

Risk: High

Usability: No previous training needed.

Mean time to failure: 1 failure for every 2 months of operation is acceptable.

Performance: The user data will be saved in 2 seconds.

Supportability: The software will support all browsers that support HTML 5.

Implementation: Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history:

Owner: Allan Shaji Manamel

Initiation date: 09/07/2015

Date last modified: 09/14/2015

Example Scenario:

User registration

Actors: Allan Shaji Manamel

Pre-conditions: Allan Shaji Manamel has the user registration page for panther buddy open in a web browser.

Description: Allan Shaji Manamel need to registers himself on "Panther Buddy".

Allan Shaji Manamel will be asked to enter his details.

- First name: Allan
- Last name: Manamel
- Email id: amana010@fiu.edu
- Phone number: 987 654 2134

Trigger: Allan Shaji Manamel clicks the submit button to submit the data.

Allan Shaji Manamel data is saved in the database along with a generated password (D43sx21). His status is marked as inactive and a default date value (Sun, 2 Dec 292269055 BC 16:47:04 +0000) is put for his activation date.

Allan Manamel sent his password (D43sx21) by email.

The user is then redirected to the login page.

Post-conditions: Allan Shaji Manamel is marked as an inactive registered user in database.

4. USER ACTIVATION

Use Case ID: PB_UC08_ACTIVATE_USER

Use Case Level: System Level

Details:

Actor: Registered user

Pre-conditions:

1. The user should be registered on panther buddy

Description: The user will hit "Login" button to activate his/her account on panther buddy after entering his login credentials.

Trigger: The user clicks the "Login" button on the main page after entering his login credentials.

The system responds by:

1. The system authenticates the username and password
2. The system marks the user as activated
3. The user gains access to the systems functionality.

Post-conditions:

1. The user status is changed to active in the Panther Buddy system.

Alternative Courses of Action:

Extensions: None.

Exceptions: Incorrect login credentials will not activate the user.

Related Use Cases: PB_UC_20_LOGIN, PB_UC01_USER_REGISTRATION

--

Decision Support

Frequency: Moderate

Criticality: High.

Risk: High- This is the core feature.

Constraints:

Usability: No previous training needed.

Reliability:

Mean time to failure - 1 failure for every 24 hours of operation is acceptable

Performance:

User will be activate in 2 seconds.

Supportability

The application will rely on HTML 5.

Implementation

Implementation will be done in HTML 5, java and Eclipse framework.

Modification History:

Owner: Abhinav Dutt

Initiation date: 09/06/2015

Date last modified: 09/06/2015

Example Scenario:

User Activation

Actors: John

Pre-conditions: John has login page for panther buddy open in a web browser.

Description: John needs to login himself on "Panther Buddy".

The John will be asked to enter his details.

- User Id: John@fiu.edu
- Password: Manamel0012

Trigger: The user clicks the Login button to Login.

Post-conditions: John's profile is activated and John is directed to home page.

5. POST MESSAGE

Use Case ID: PB _Posting message

Use Case Level: System Level (end to end)

Details:

Actor: Registered user

Pre-conditions: The user must be logged-in and the post message page should be opened in a web browser.

Description: The user will add post by filling the information that he wants and post it.

1. The user writes his message to be posted in the space provided.

2. The user clicks the add post button to submit his message

Post-conditions: The post will be saved in the system.

Alternative Courses of Action:

At steps 1 the user can hit “Cancel” button to cancel post.

Extensions:

Exceptions: The message will be rejected if it contains illegal characters or keywords or their combination.

Related Use Cases: PB_SC2_XSS_FILTERING

Decision Support

Frequency: Medium - User

Criticality: High

Risk: Medium - attacker may try to attack using XSS

Constraints:

Usability: No training is needed.

Reliability:

Mean time to failure - 1 failure for every 24 hours of operation is acceptable.

Performance:

The file or text should be saved along with the post.

Supportability:

The software will support all browsers that support HTML 5.

Implementation:

Implementation will be done in HTML 5, java and Eclipse framework.

Modification History:

Owner: Abhinav Dutt

Initiation date: 09/06/2015

Date last modified: 09/06/2015

Example Scenario:

Post message

Actors: Kedari Sharat

Pre-conditions: Kedari Sharat must be logged-in and the post message page should be opened in a web browser.

Description:

1. Kedari Sharat writes his message to be posted in the space provided.
2. Kedari Sharat clicks the add post button to submit his message

Post conditions: The post will be saved in the system.

6. CHANGE PASSWORD

Use case ID: PB_UC03_CHANGE_PASSWORD

Use case level: System level (End-to-End)

Details

Actors: Logged in registered user.

Pre-conditions: The user has his own profile page open on 'Panther Buddy'.

Description: The user tries to change his password.

1. User clicks the change password button on his profile page
2. 3 fields appear on the page. They are:
   a) Current Password.
   b) New Password
   c) Confirm Password
3. The user enters the data asked

3.1 User enters his current password in the current password field

3.2 User enters his new desired password in the new password field.

3.3 User re-enters his desired new password in the confirm password field.

4. The user clicks the submit button to submit his data.
5. The 3 Fields for password disappear.

Post condition: The user's password is changed as per his desire.

Alternative courses of action:

1. At any time during step 1 to 4, the user can click the cancel button to return to the 'Panther Buddy' login page.

Extensions:

Exceptions:

1. The user at step 3.1 enters a wrong password. He is asked to correct his entered password.
2. The user at step 3.2 enters a password that does not fulfil the password criteria for 'Panther Buddy'. He is asked to correct his entered password.
3. The user at step 3.3 enters a password that does not match the password entered in step 3.2. He is asked to correct his entered password.

Related use cases:

Decision support:

Frequency: Moderate - Performed every time a user wants to change his password. (Used twice a day)

Criticality: Moderate

Risk: High

Usability: No previous training needed.

Reliability:

Mean time to failure: 1 failure for every 6 months of operation is acceptable.

Performance:

The user password will be changed in 2 seconds

Supportability:

The software will support all browsers that support HTML 5.

Implementation:

Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Allan Shaji Manamel

Initiation date: 09/18/2015

Date last modified: 09/18/2015

Example scenario:

Actor: Allan Shaji Manamel

Pre-conditions: Allan Shaji Manamel is at his own profile page open on 'Panther Buddy'.

Allan wants to change his password.

1. Allan logs in to the 'Panther Buddy' system and opens his profile page on laptop browser that supports HTML 5.
2. Allan clicks the change password button.
3. Fields appear on the page. They are:
   a) Current Password.

b) New Password

c) Confirm Password

4. Allan enters his current password = DX56ssUT in current password field.
5. Allan enters his new desired password = FS566ssCV in the new password field.
6. Allan re-enters his new desired password = FS566ssCV in the confirm password field.
7. Allan clicks the submit button to submit his data.
8. Allan's password is changed by the system to his desired password.

Post condition: Allan Shaji Manamel’s password is changed as per his desire.

7. PASSWORD RECOVERY

Use case ID: PB_UC02_RECOVER_PASSWORD

Use case level: System level (End-to-End)

Details

Actors: Any user.

Pre-conditions: The user has the login page open.

Description: The user tries to recover his password, which is sent to him at his registered email.

1. User clicks the recover password button on the login page
2. The user is redirected to the recover password page.
3. The user enters his email id used to register in the system.
4. The user clicks the submit button to submit his email id.
5. The user with the email-id is recovered and his password is sent to him by email.

Post condition: The user gets his password on his registered email id.

Alternative courses of action:

At any time during step 1 to 4, the user can click the cancel button to return to the 'Panther Buddy' login page.

Extensions:

Exceptions:

The user at step 3 enters an email id that is not registered in the system. The user is asked to re-enter a valid email id.

Related use cases:

Decision support:

Frequency: Moderate - Performed every time a use forgets his password. (Used twice a day)

Criticality: Very Critical - If the user wants to use the 'Panther Buddy' site he needs to be able to login into the system, for which he needs his password.

Risk: High

Usability: No previous training needed.

Mean time to failure: 1 failure for every 2 months of operation is acceptable.

Performance:

The user will be sent a mail in 2 seconds.

Supportability:

The software will support all browsers that support HTML 5.

Implementation:

Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Allan Shaji Manamel

Initiation date: 09/18/2015

Date last modified: 09/18/2015

Example scenario:

Actor: Allan Shaji Manamel

Pre-conditions: The user has the login page open.

Allan forgets his password.

3. Allan opens the 'Panther Buddy' login page on his laptop browser that supports HTML 5.
4. Allan clicks the recover password button.
5. Allan is redirected to the recover password page.
6. Allan enters his email id: amana100@fiu.edu used to register into the system.
7. Allan clicks the submit button to submit the data.
8. Allan's password: DX56ssUT is recovered by the system and sent to his email account.

Post condition: Allan Shaji Manamel gets his password on his registered email id: amana100@fiu.edu.

8. DELETE MESSAGE

Use Case ID: PB_UC14_DELETE_MESSAGE

Use Case Level: System level (end to end)

Details:

Actor: Registered user

Pre-conditions: The user must be logged-in. The user must be owner of that post that he wants to delete

Description: The user will delete the post that has privileges to do so

. Trigger: The user clicks delete post button near the message.

The system responds by

3. A window will appear.
4. The user needed to confirm wither he want to delete the post .
5. By clicking “ok” button, the post will be deleted.

Post-conditions: The page will be loaded after deleting the post

Alternative Courses of Action:

At steps 2 and 3, the user can hit “Cancel” to cancel deletion.

Extensions: none.

Exceptions: The post need to be owned by user.

Related Use Cases: log-in, Post messages.

Decision Support

Frequency: Medium - User

Criticality: Moderate - to provide better visualization of the post.

Risk: Medium.

Constraints:

Usability: No previous training needed.

Reliability

Mean time to failure - 1 failure for every 24 hours of operation is acceptable.

Performance

The file or text should be saved along with the post.

Supportability

The application will be web-based. It will have supported on all sorts of modern devices.

Implementation

Implementation will be done in HTML 5, java and Eclipse framework.

Modification History:

Owner: Kedari Sharat

Initiation date: 09/05/2015

Date last modified: 09/06/2015

Example scenario:

Actor: Kedari Sharat

Pre-conditions: Kedari Sharat be logged-in. Kedari Sharat must be owner of that post that he wants to delete

Description: Kedari Sharat will delete the post that has privileges to do so.

Trigger: Kedari Sharat clicks delete post button from the menu.

The system responds by:

1. A window will appear.
2. Sharat needed to confirm wither he want to delete the post .
3. By clicking "ok" button, the post will be deleted.

Post-conditions: The page will be loaded after deleting the post

9. XSS FILTERING

Use case ID: PB_SC2_XSS_FILTERING

Use case level: System level

Details

Actors: Registered user.

Pre-conditions: The user has the post message page open.

Description: The user message is filtered to check for XSS attack.

1. The user writes his message in the space provided for writing his message.
2. The user clicks the post message button to submit the message.
3. The system checks the message string for malicious code.
4. System saves the message for moderation by moderator.

Post condition: The users post is saved in the system for moderation by moderator.

Alternative courses of action:

At any time during step 1 to 4, the user can click the cancel button to return to the home page.

Extensions:

Exceptions:

The user at step 1 puts malicious code in his message. The system shows an error for in proper content at asks user to rectify it.

Related use cases:

Decision support

Frequency: High - Performed every time a user posts his message. (Used 20 times a day)

Criticality: Very

Risk: High

Usability: No previous training needed.

Reliability:

Mean time to failure - 1 failure for every 2 months of operation is acceptable.

Performance:

The user will be able to submit his message in 2 sec

Supportability:

The software will support all browsers that support HTML 5.

Implementation:

Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Allan Shaji Manamel

Initiation date: 09/18/2015

Date last modified: 09/18/2015

Example scenario:

Actor: Allan Shaji Manamel

Pre-conditions: Allan Shaji Manamel has the post message page open.

Allan submits his message.

1. Allan logs into the 'Panther Buddy' system and opens the post message page on his laptop browser that supports HTML 5.
2. Allan writes his message "Used book - Introduction to Algorithms by Thomas Cormen up for sale at 19$".
3. Allan clicks the submit button to submit his message.
4. The system checks if the message contains any malicious code or not.
5. The system verifies the message to be ok.
6. The system saves the message to be moderated by a moderator.

Post condition: Allan Shaji Manamel's post is saved in the system for moderation by moderator.

Misuse case
Use case ID: PB_MC2_XSS_FILTERING

Use case level: System level

Details

Actors: Misuser.

Pre-conditions: The misuser has the post message page open.

Description: The misuser embeds malicious scripts.

1. The misuser writes his message in the space provided for writing his message.
2. The misuser clicks the post message button to submit the message.

Post condition: The user's message when rendered on page runs the script to hack into the system to fetch/manipulate unauthorised data from the system.

Criticality: Very

Risk: High

10. SQL INJECTION PREVENTION ON LOGIN

Use case ID: PB_SC1_FILTER_INPUT

Use case level: System level (End-to-End)

Details

Actors: Any user.

Pre-conditions: The user has the login page open.

Description: The user logins into the system using his email and password.

1. User enters his email into the email field
2. The user enters his password in the password field
3. The user clicks the login button to submit his data.
4. The system checks the email and password for malicious content before querying for the user with the entered login credentials

Post condition: The user is redirected to the Panther Buddy home page.

Extensions:

Exceptions:

Related use cases: PB_UC_20_LOGIN, PB_SC7_INVALID_LOGIN

Decision support:

Frequency: Moderate - Performed every time a user logins to the Panther buddy system

Criticality: Very Critical

Risk: High

Usability: No previous training needed.

Reliability:

Mean time to failure - 1 failure for every 2 months of operation is acceptable.

Performance:

The user will be redirected to the home page in 2 seconds

Supportability:

The software will support all browsers that support HTML 5.

Implementation:

Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Allan Shaji Manamel

Initiation date: 09/18/2015

Date last modified: 09/18/2015

Example scenario:

Actor: Allan Shaji Manamel

Pre-conditions: The user has the login page open.

1. Allan forgets his password.
2. Allan opens the 'Panther Buddy' login page on his laptop browser that supports HTML 5.
3. Allan enters his email amana100@fiu.edu into the email field.
4. Allan enters his password DX345tv in the password field.
5. Allan clicks the login button to submit the data.
6. The system checks the input email amana100@fiu.edu and password DX345tv for malicious content

Post condition: The user is redirected to the Panther Buddy home page.

Misuse case

Use case ID: PB_MC1_FILTER_INPUT

Use case level: System level (End-to-End)

Details

Actors: Misuser.

Pre-conditions: The misuser has the login page open.

Description: The misuser gains entry into the system.

1. The misuser enters his email into the email field
2. The misuser enters a password : dummy' OR 1=1 OR '' = '
3. The misuser clicks the login button to submit his data.

4. The system will put the string in the where clause of a SQL statement used to compare existing data for input data like WHERE password = ‘dummy’ OR 1=1 OR ‘’ = ‘’, thereby gaining entry into the system.

Post condition: The user is redirected to the Panther Buddy home page.

Related use cases: PB_UC_20_LOGIN, PB_SC7_INVALID_LOGIN

Criticality: Very Critical

Risk: High

11. VIEW POST

Use Case ID: PB_UC_16_VIEW_POST

Use Case Level: System leve (end to end)

Details:

Actor: Registered User

Pre-conditions: The registered user has to login in order to be able to view a post.

Description

1. The user clicks the home page button on the screen he is at.
2. User is redirected to the homepage where he can view a list of posts ordered by most recent at top.

Post-conditions: The user can view the already posted messages.

Alternative Courses of Action:

Extensions:

Exceptions: The system might not be able to display a longer version of the message.

Related Use Cases: PB_UC15_SEARCH_MESSAGE

Decision Support

Frequency: More than 30 times in an hour

Criticality: Moderate

Risk: Moderate

Usability: The registered user should be able to view any post at any moment

Reliability: Mean time to failure is about 3 failures for every 24 hours of operation

Performance: The post should be displayed in less than two seconds

Supportability: The application will support any text format

Implementation: Bootstrap frontend framework and Java JSF

Modification History:

Owner: James W. Angelo

Initiation date: 09/22/2015

Date last modified: 09/18/2015

Example scenario:

Actor: James Angelo

Pre-conditions: The registered user has to login in order to be able to view a post.

Description:

1. The user clicks the home page button on the screen he is at.

2. User is redirected to the homepage where he can view a list of posts ordered by most recent at top.

Post-conditions: The user can view the already posted messages.

12. LOGOUT

Use case ID: PB_UC21_LOGOUT

Use case level: System level (End-to-End)

Details

Actors: Registered user.

Pre-conditions: The user has logged into the system.

Description:

1. The user clicks the name of the user he wants the profile of.
2. The user is redirected to the profile page for the user

Post condition: The user is redirected to the profile page of the user he wants to view details of.

Extensions:

Exceptions:

Related use cases: PB_UC03_CHANGE_PASSWORD

Decision support:

Frequency: Moderate - Performed every time a user logs out from the Panther buddy system

Criticality: low

Risk: low

Usability: No previous training needed.

Reliability:

Mean time to failure - 1 failure for every 2 months of operation is acceptable.

Performance:

The user will be redirected to the profile page in 2 seconds

Supportability:

The software will support all browsers that support HTML 5.

Implementation:

Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Allan Shaji Manamel

Initiation date: 09/18/2015

Date last modified: 09/18/2015

Example scenario:

Actor: Allan Shaji Manamel

Pre-conditions: Allan Shaji Manamel has logged into the system and is on the home page

Description:

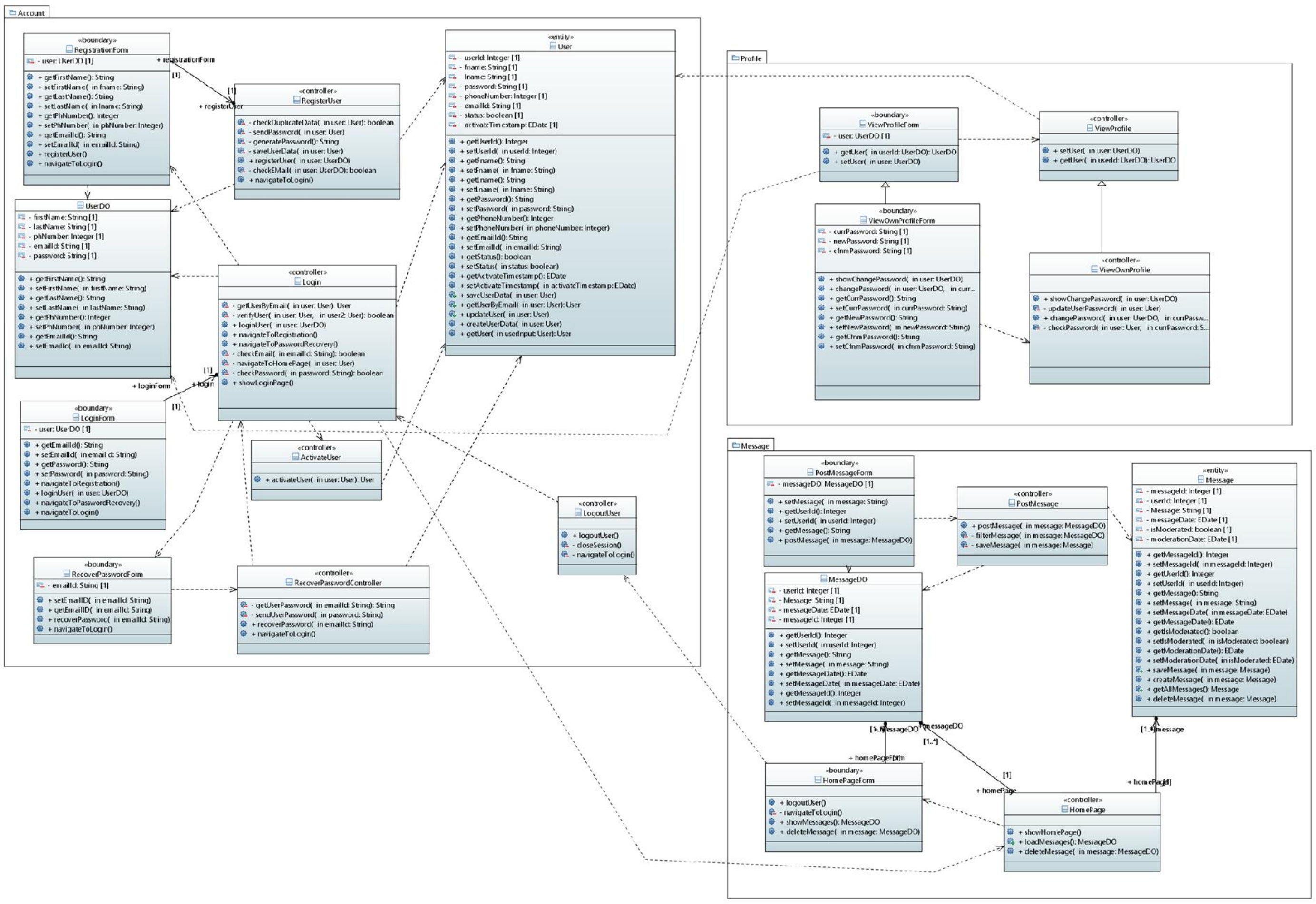
1. Allan clicks on the name of Abhinav, which appears above a post created by Abhinav.
2. Allan is redirected to Abhinav's profile page.

Post condition: Allan Shaji Manamel can view Abhinav's profile.

5.2 Static model – object diagrams (one per scenario), class diagram

Class Diagram

Below is the high level class diagram.

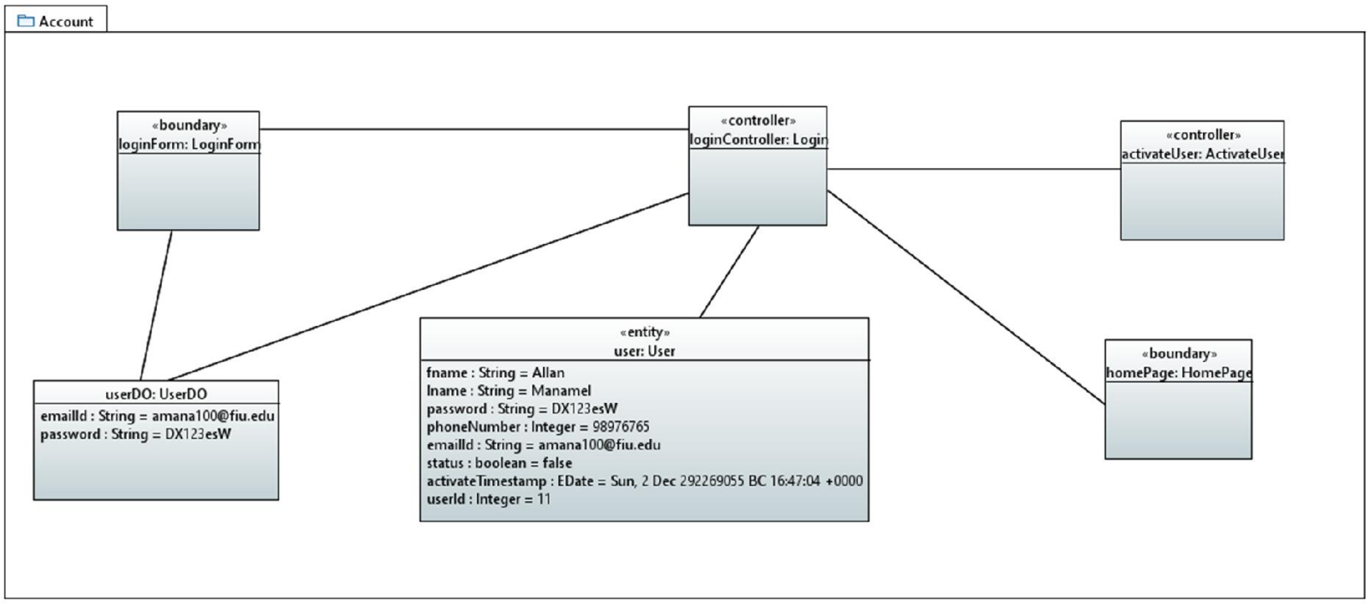


Account
Profile
Message
«boundary» RegistrationForm
«controller» RegisterUser
«entity» User
UserDO
«controller» Login
«boundary» LoginForm
«controller» ActivateUser
«controller» LogoutUser
«boundary» RecoverPasswordForm
«controller» RecoverPasswordController
«boundary» ViewProfileForm
«controller» ViewProfile
«boundary» ViewOwnProfileForm
«controller» ViewOwnProfile
«boundary» PostMessageForm
«controller» PostMessage
«entity» Message
MessageDO
«boundary» HomePageForm
«controller» HomePage

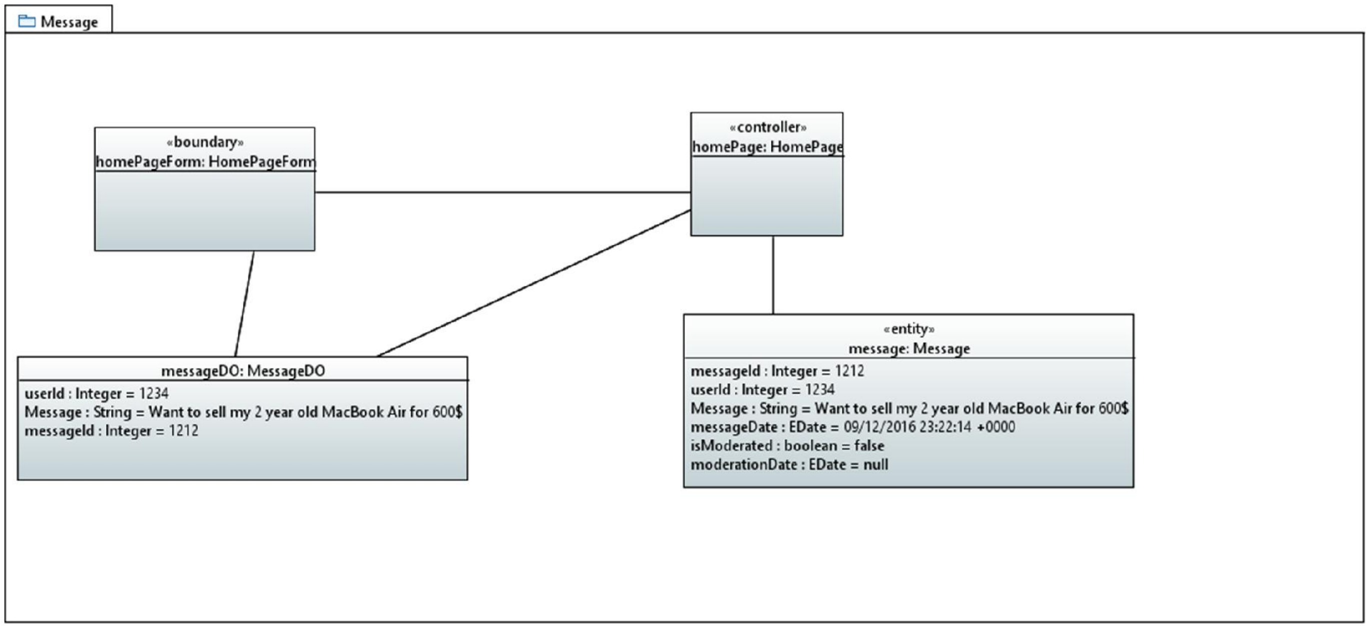
Object Diagrams

Below are the object diagrams for the use cases being implemented.

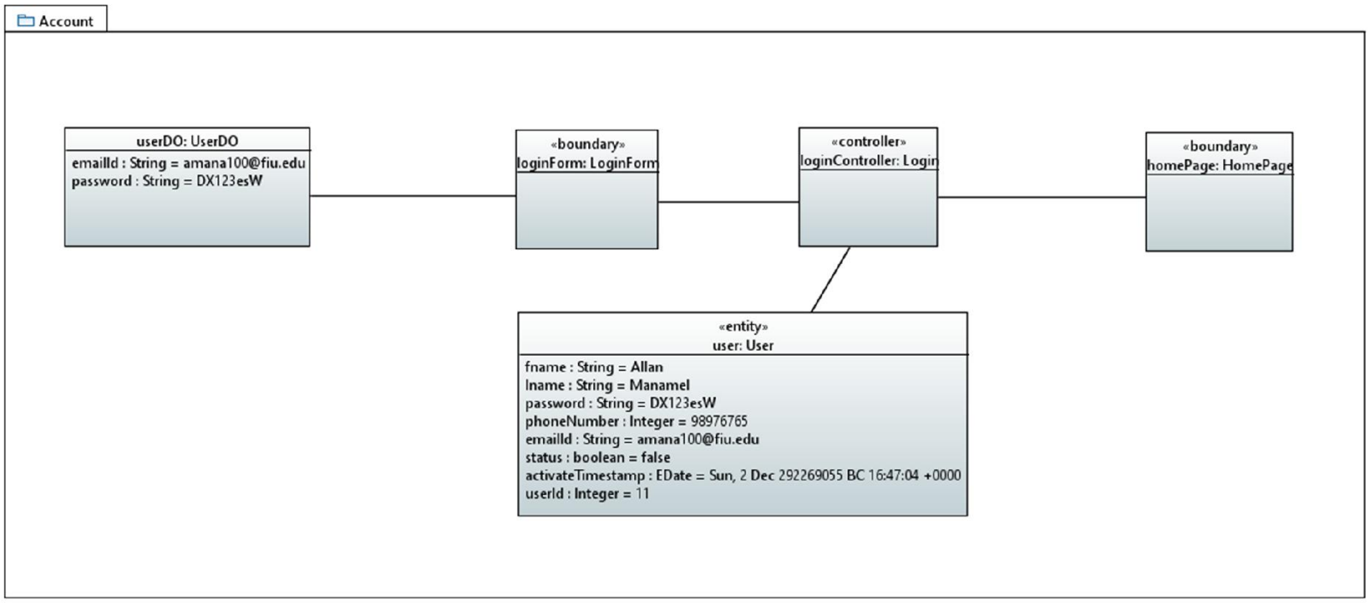
Activate User


Account
«boundary»
loginForm: LoginForm
«controller»
loginController: Login
«controller»
activateUser: ActivateUser
«entity»
user: User
fname : String = Allan
lname : String = Manamel
password : String = DX123esW
phoneNumber : Integer = 98976765
emailId : String = amana100@fiu.edu
status : boolean = false
activateTimestamp : EDate = Sun, 2 Dec 292269055 BC 16:47:04 +0000
userId : Integer = 11
«boundary»
homePage: HomePage
userDO: UserDO
emailId : String = amana100@fiu.edu
password : String = DX123esW

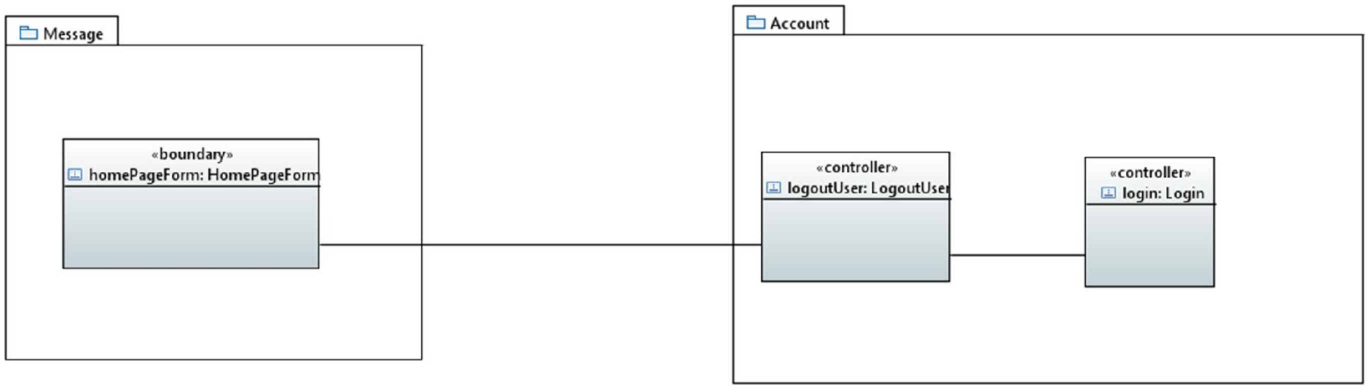
Delete Message


Message
«boundary»
homePageForm: HomePageForm
«controller»
homePage: HomePage
messageDO: MessageDO
userId : Integer = 1234
Message : String = Want to sell my 2 year old MacBook Air for 600$
messageId : Integer = 1212
«entity»
message: Message
messageId : Integer = 1212
userId : Integer = 1234
Message : String = Want to sell my 2 year old MacBook Air for 600$
messageDate : EDate = 09/12/2016 23:22:14 +0000
isModerated : boolean = false
moderationDate : EDate = null

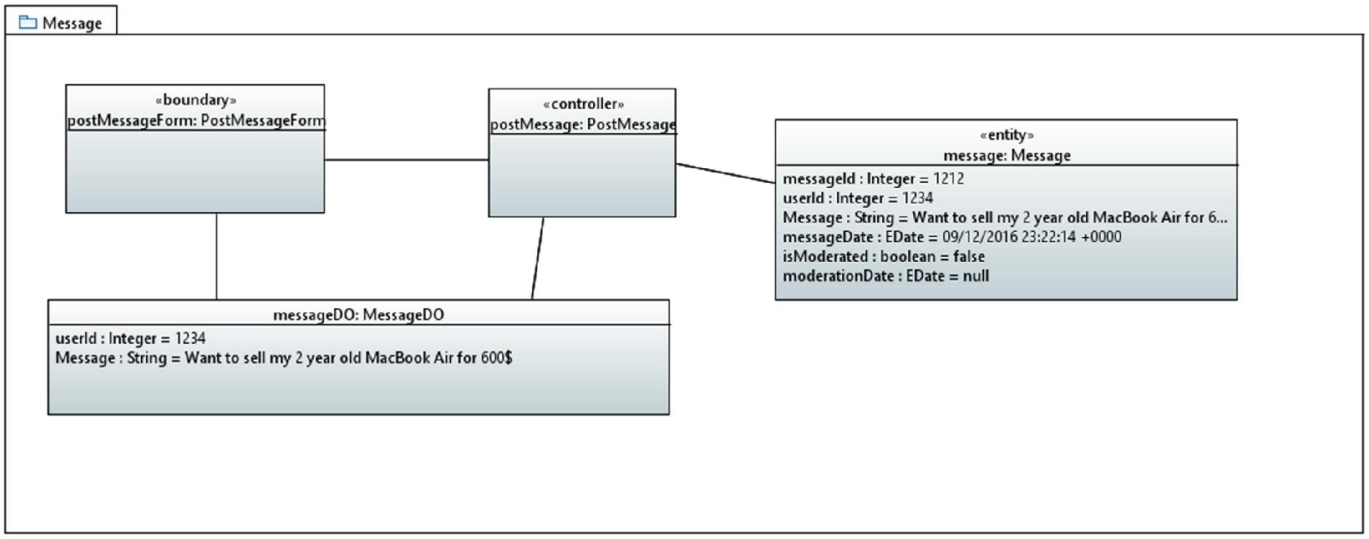
User Login and SQL Injection


Account
userDO: UserDO
emailId : String = amana100@fiu.edu
password : String = DX123esW
«boundary»
loginForm: LoginForm
«controller»
loginController: Login
«boundary»
homePage: HomePage
«entity»
user: User
fname : String = Allan
lname : String = Manamel
password : String = DX123esW
phoneNumber : Integer = 98976765
emailId : String = amana100@fiu.edu
status : boolean = false
activateTimestamp : EDate = Sun, 2 Dec 292269055 BC 16:47:04 +0000
userId : Integer = 11

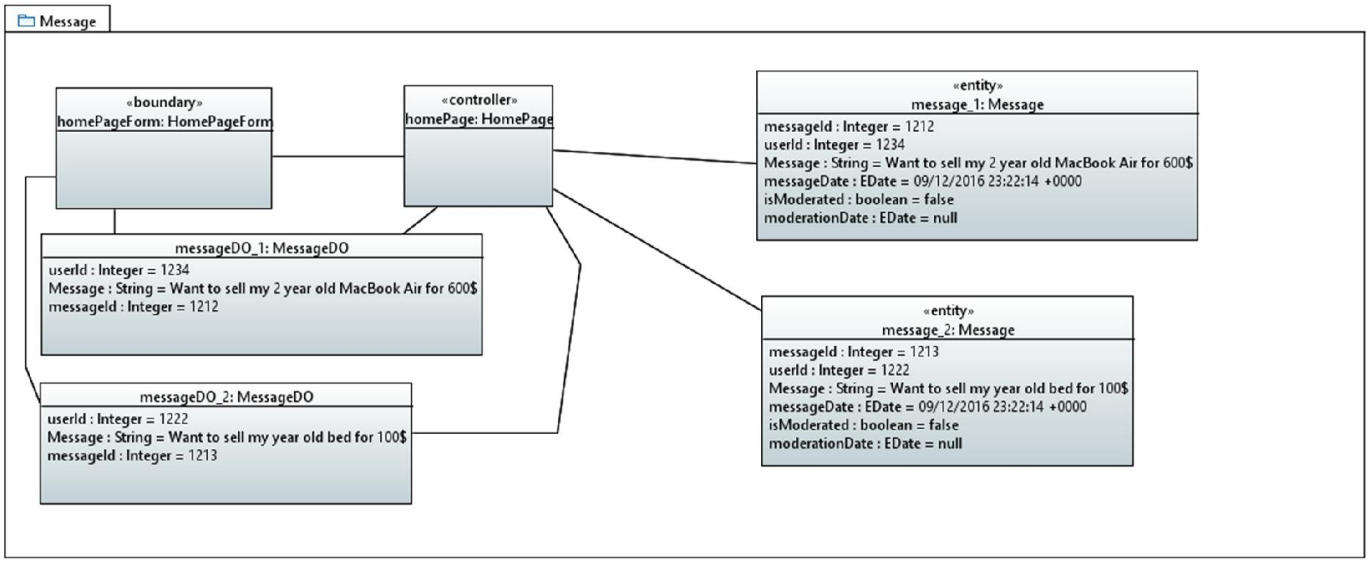
User Logout



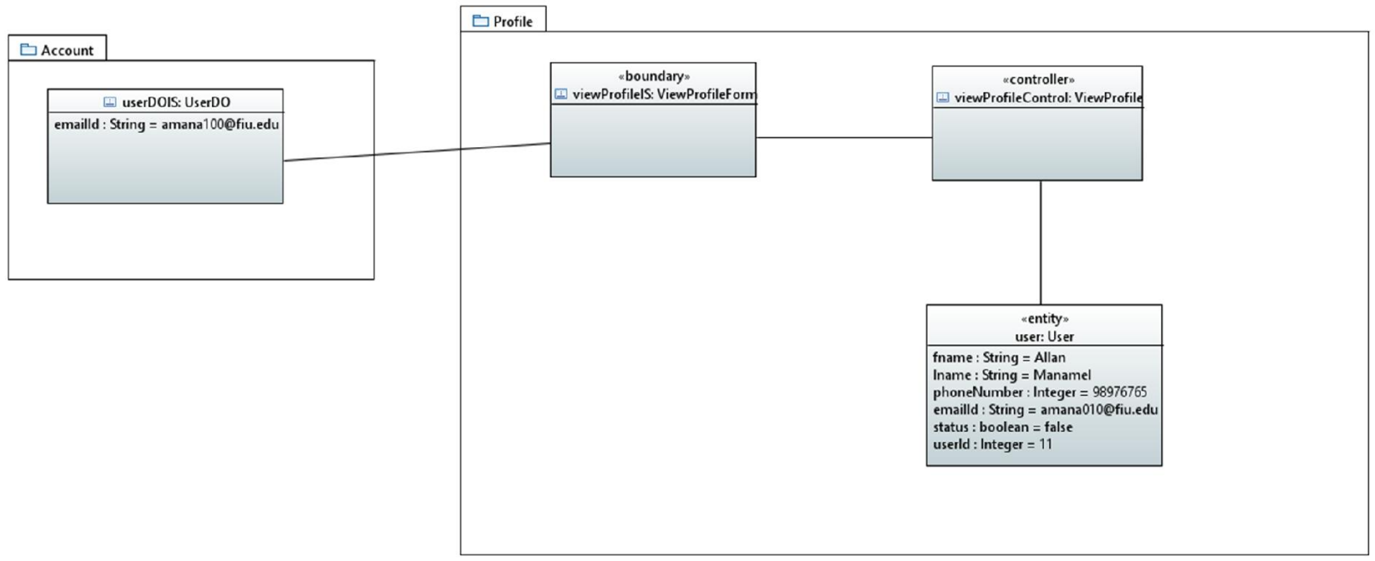
Post Message and XSS Filtering


Message
«boundary»
postMessageForm: PostMessageForm
«controller»
postMessage: PostMessage
«entity»
message: Message
messageId : Integer = 1212
userId : Integer = 1234
Message : String = Want to sell my 2 year old MacBook Air for 6...
messageDate : EDate = 09/12/2016 23:22:14 +0000
isModerated : boolean = false
moderationDate : EDate = null
messageDO: MessageDO
userId : Integer = 1234
Message : String = Want to sell my 2 year old MacBook Air for 600$

View Post

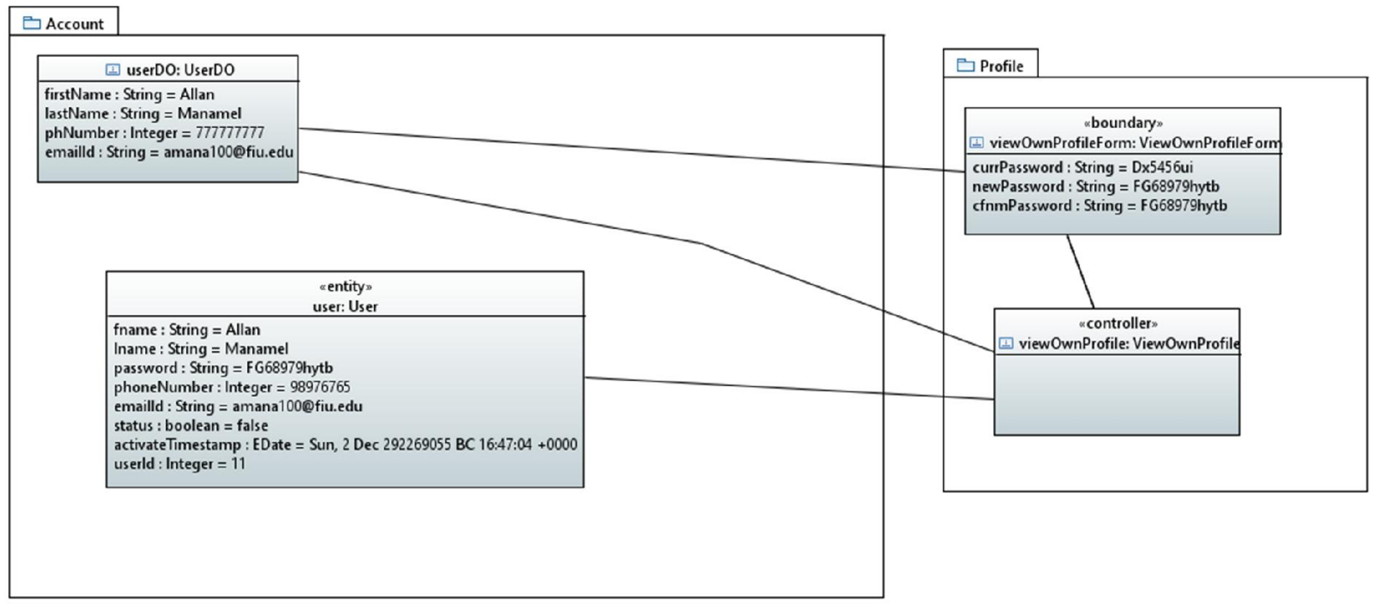

Message
«boundary»
homePageForm: HomePageForm
«controller»
homePage: HomePage
«entity»
message_1: Message
messageId : Integer = 1212
userId : Integer = 1234
Message : String = Want to sell my 2 year old MacBook Air for 600$
messageDate : EDate = 09/12/2016 23:22:14 +0000
isModerated : boolean = false
moderationDate : EDate = null
messageDO_1: MessageDO
userId : Integer = 1234
Message : String = Want to sell my 2 year old MacBook Air for 600$
messageId : Integer = 1212
«entity»
message_2: Message
messageId : Integer = 1213
userId : Integer = 1222
Message : String = Want to sell my year old bed for 100$
messageDate : EDate = 09/12/2016 23:22:14 +0000
isModerated : boolean = false
moderationDate : EDate = null
messageDO_2: MessageDO
userId : Integer = 1222
Message : String = Want to sell my year old bed for 100$
messageId : Integer = 1213

View Profile



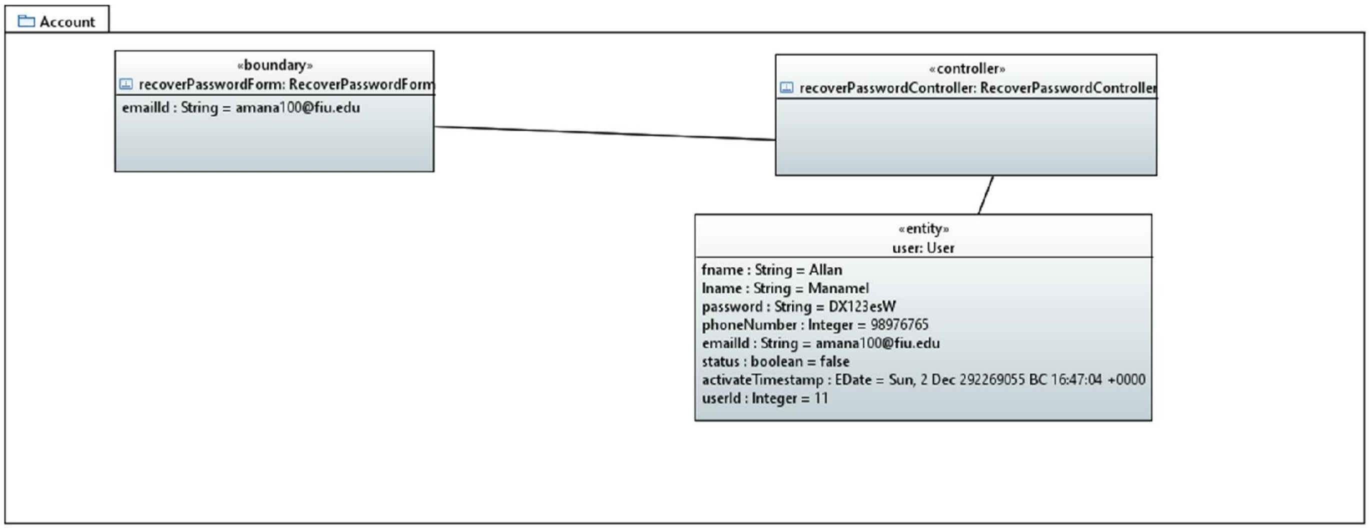
Account
userDOIS: UserDO
emailId : String = amana100@fiu.edu
Profile
«boundary»
viewProfileIS: ViewProfileForm
«controller»
viewProfileControl: ViewProfile
«entity»
user: User
fname : String = Allan
lname : String = Manamel
phoneNumber : Integer = 98976765
emailId : String = amana010@fiu.edu
status : boolean = false
userId : Integer = 11

Change Password

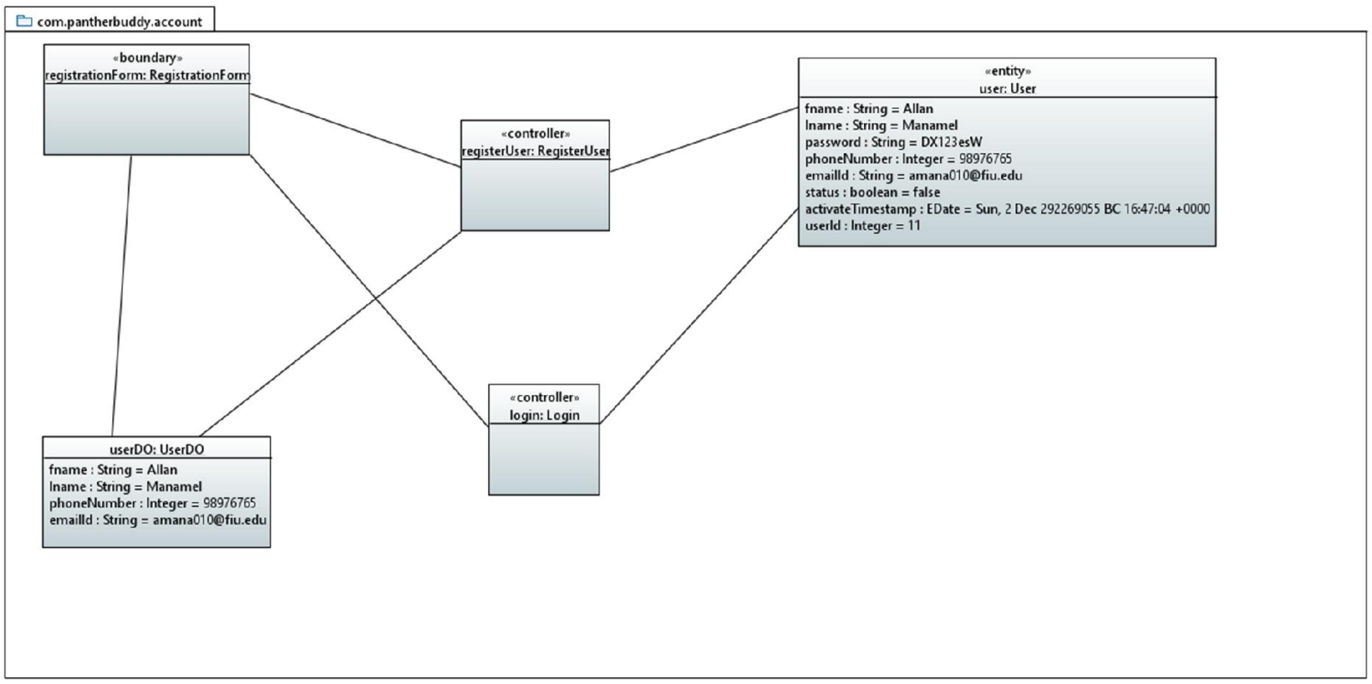


Account
userDO: UserDO
firstName : String = Allan
lastName : String = Manamel
phNumber : Integer = 777777777
emailId : String = amana100@fiu.edu
«entity»
user: User
fname : String = Allan
lname : String = Manamel
password : String = FG68979hytb
phoneNumber : Integer = 98976765
emailId : String = amana100@fiu.edu
status : boolean = false
activateTimestamp : EDate = Sun, 2 Dec 292269055 BC 16:47:04 +0000
userId : Integer = 11
Profile
«boundary»
viewOwnProfileForm: ViewOwnProfileForm
currPassword : String = Dx5456ui
newPassword : String = FG68979hytb
cfnmPassword : String = FG68979hytb
«controller»
viewOwnProfile: ViewOwnProfile

Recover Password


Account
«boundary»
recoverPasswordForm: RecoverPasswordForm
emailId : String = amana100@fiu.edu
«controller»
recoverPasswordController: RecoverPasswordController
«entity»
user: User
fname : String = Allan
lname : String = Manamel
password : String = DX123esW
phoneNumber : Integer = 98976765
emailId : String = amana100@fiu.edu
status : boolean = false
activateTimestamp : EDate = Sun, 2 Dec 292269055 BC 16:47:04 +0000
userId : Integer = 11

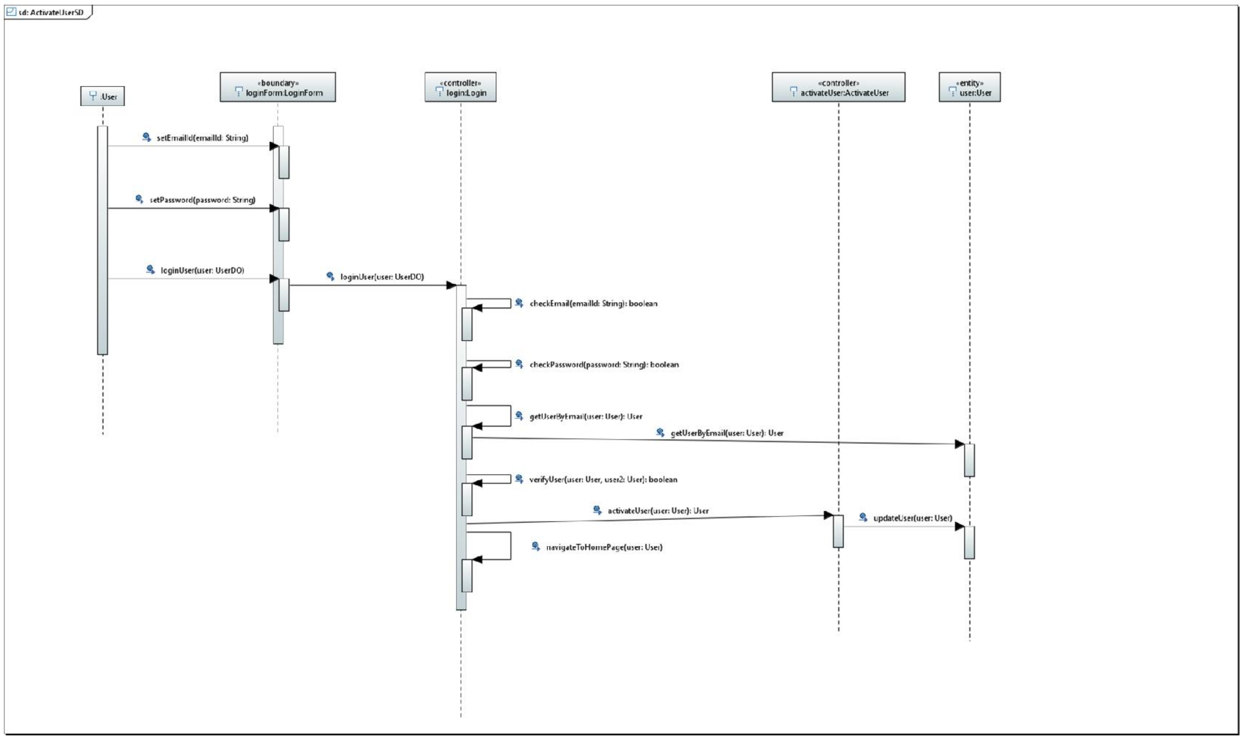
User Registration


com.pantherbuddy.account
«boundary»
registrationForm: RegistrationForm
«controller»
registerUser: RegisterUser
«entity»
user: User
fname : String = Allan
lname : String = Manamel
password : String = DX123esW
phoneNumber : Integer = 98976765
emailId : String = amana010@fiu.edu
status : boolean = false
activateTimestamp : EDate = Sun, 2 Dec 292269055 BC 16:47:04 +0000
userId : Integer = 11
«controller»
login: Login
userDO: UserDO
fname : String = Allan
lname : String = Manamel
phoneNumber : Integer = 98976765
emailId : String = amana010@fiu.edu

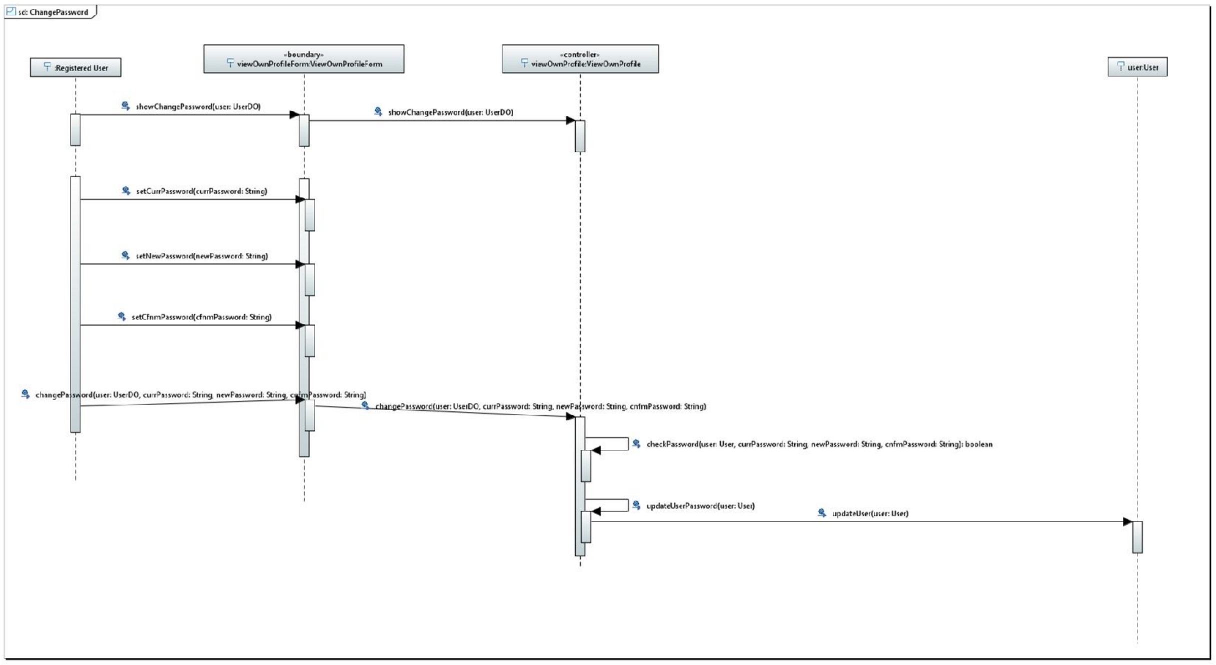
5.3 Dynamic model – Sequence diagrams (one per scenario)

Sequence Diagrams.

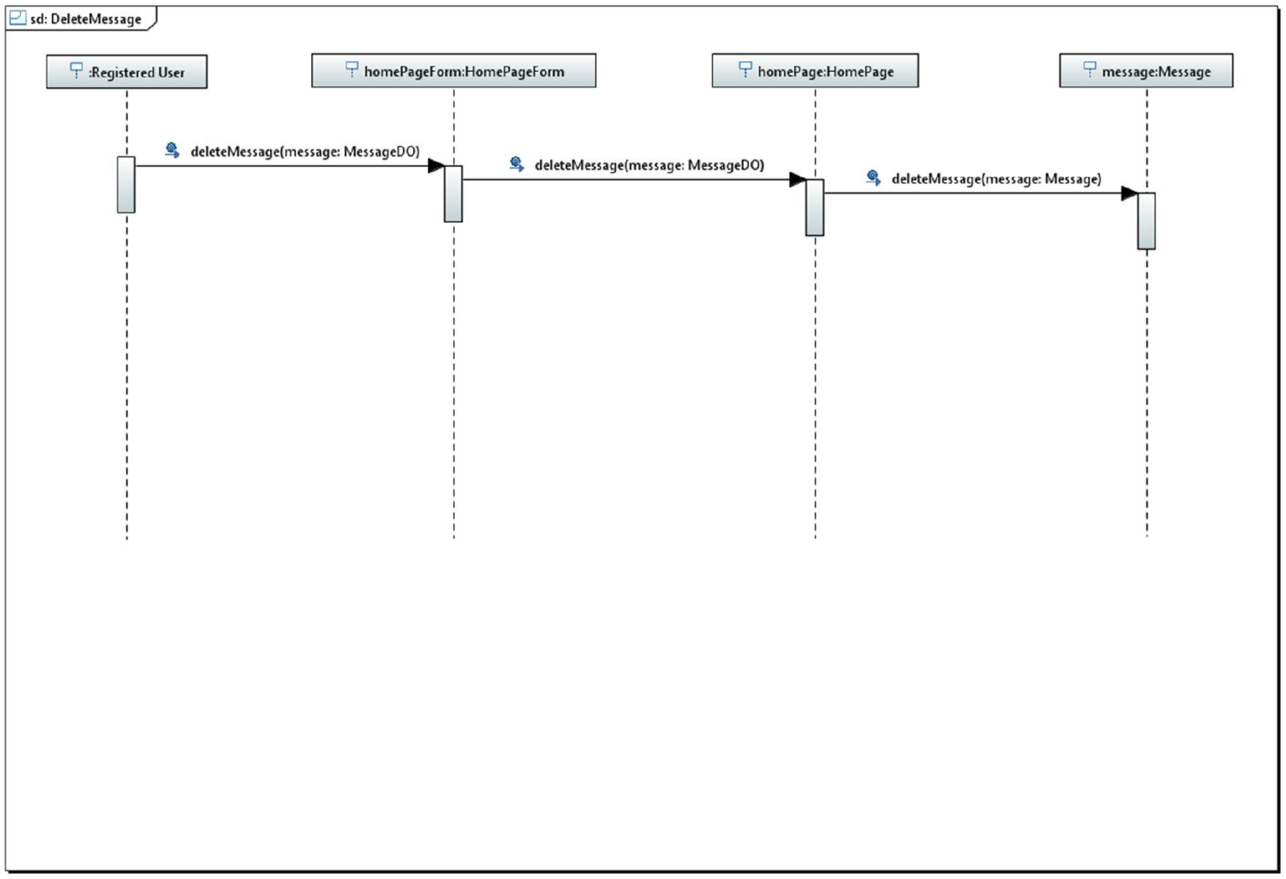
Activate User


sd: ActivateUserSD
:User
«boundary» loginForm:LoginForm
«controller» login:Login
«controller» activateUser:ActivateUser
«entity» user:User
setEmailId(emailId: String)
setPassword(password: String)
loginUser(user: UserDO)
loginUser(user: UserDO)
checkEmail(emailId: String): boolean
checkPassword(password: String): boolean
getUserByEmail(user: User): User
getUserByEmail(user: User): User
verifyUser(user: User, user2: User): boolean
activateUser(user: User): User
updateUser(user: User)
navigateToHomePage(user: User)

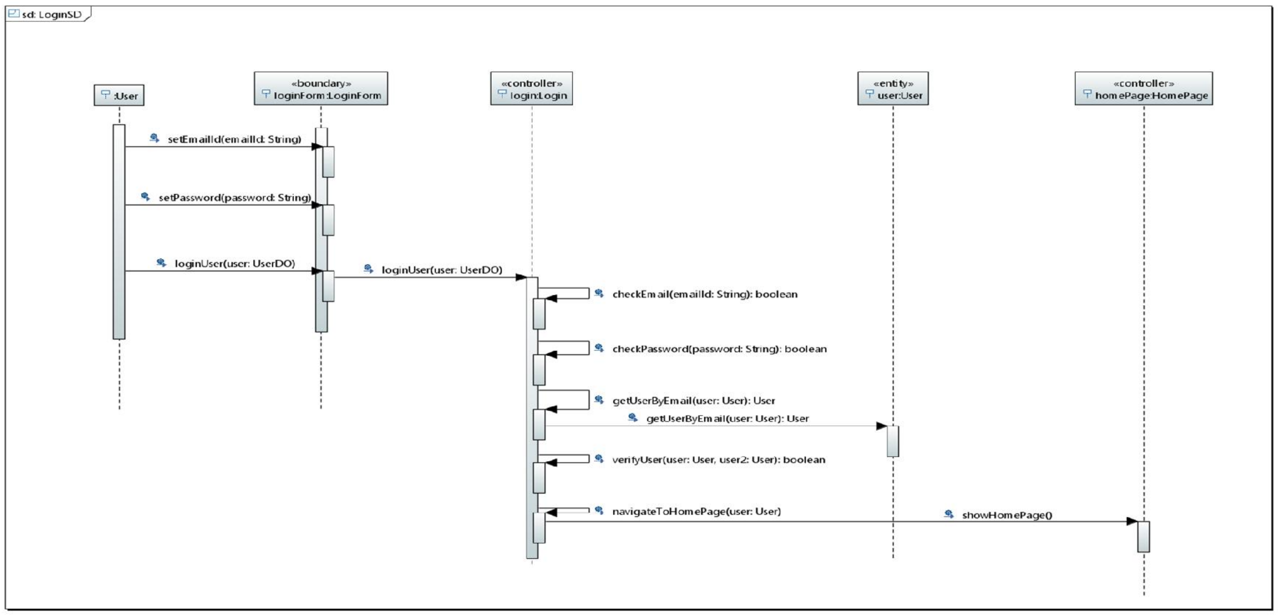
Change Password


sd: ChangePassword
:Registered User
«boundary» viewOwnProfileForm:ViewOwnProfileForm
«controller» viewOwnProfile:ViewOwnProfile
user:User
showChangePassword(user: UserDO)
showChangePassword(user: UserDO)
setCurrPassword(currPassword: String)
setNewPassword(newPassword: String)
setCfnmPassword(cfnmPassword: String)
changePassword(user: UserDO, currPassword: String, newPassword: String, cnfmPassword: String)
changePassword(user: UserDO, currPassword: String, newPassword: String, cnfmPassword: String)
checkPassword(user: User, currPassword: String, newPassword: String, cnfmPassword: String): boolean
updateUserPassword(user: User)
updateUser(user: User)

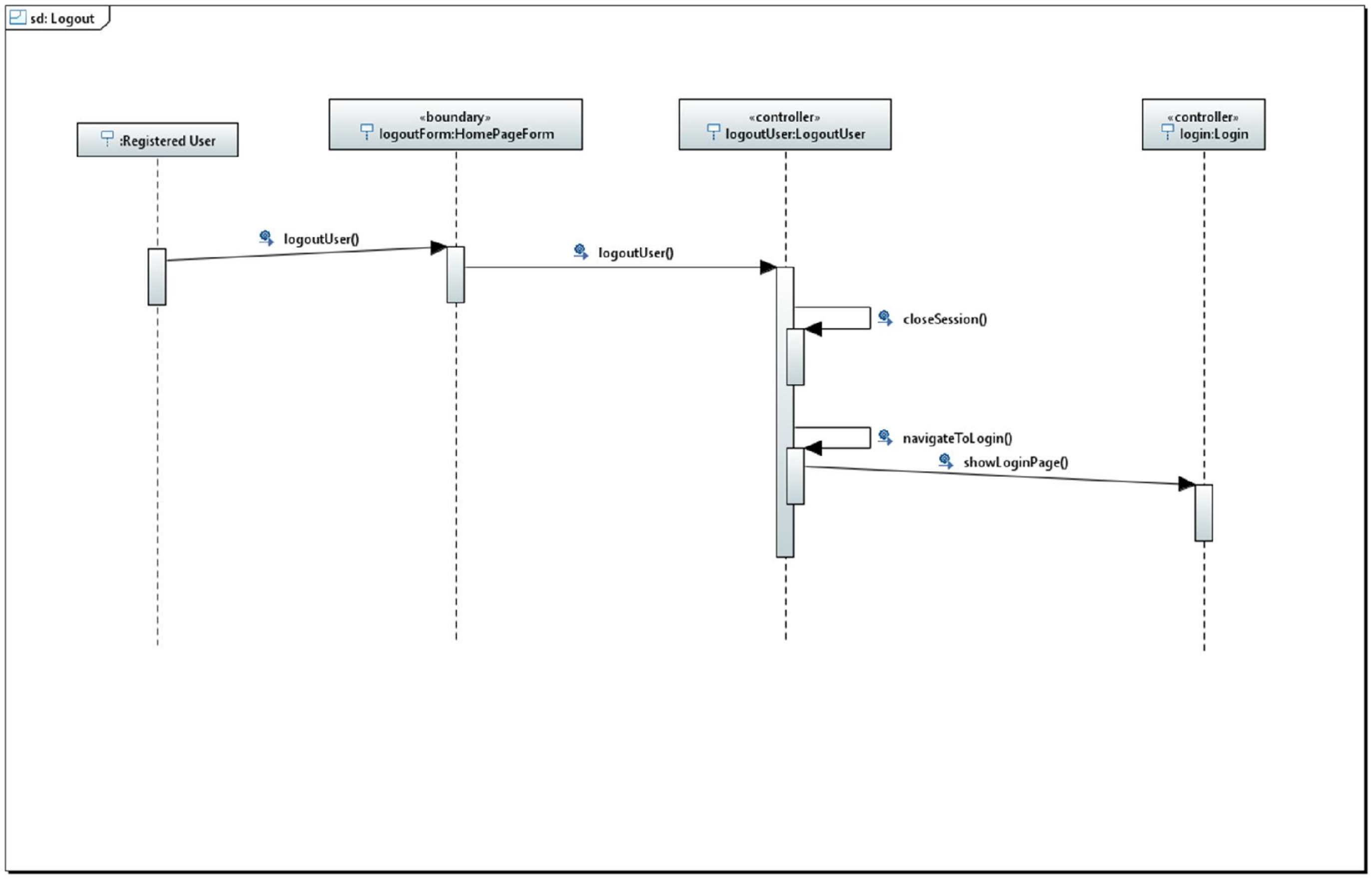
Delete Message


sd: DeleteMessage
:Registered User
homePageForm:HomePageForm
homePage:HomePage
message:Message
deleteMessage(message: MessageDO)
deleteMessage(message: MessageDO)
deleteMessage(message: Message)

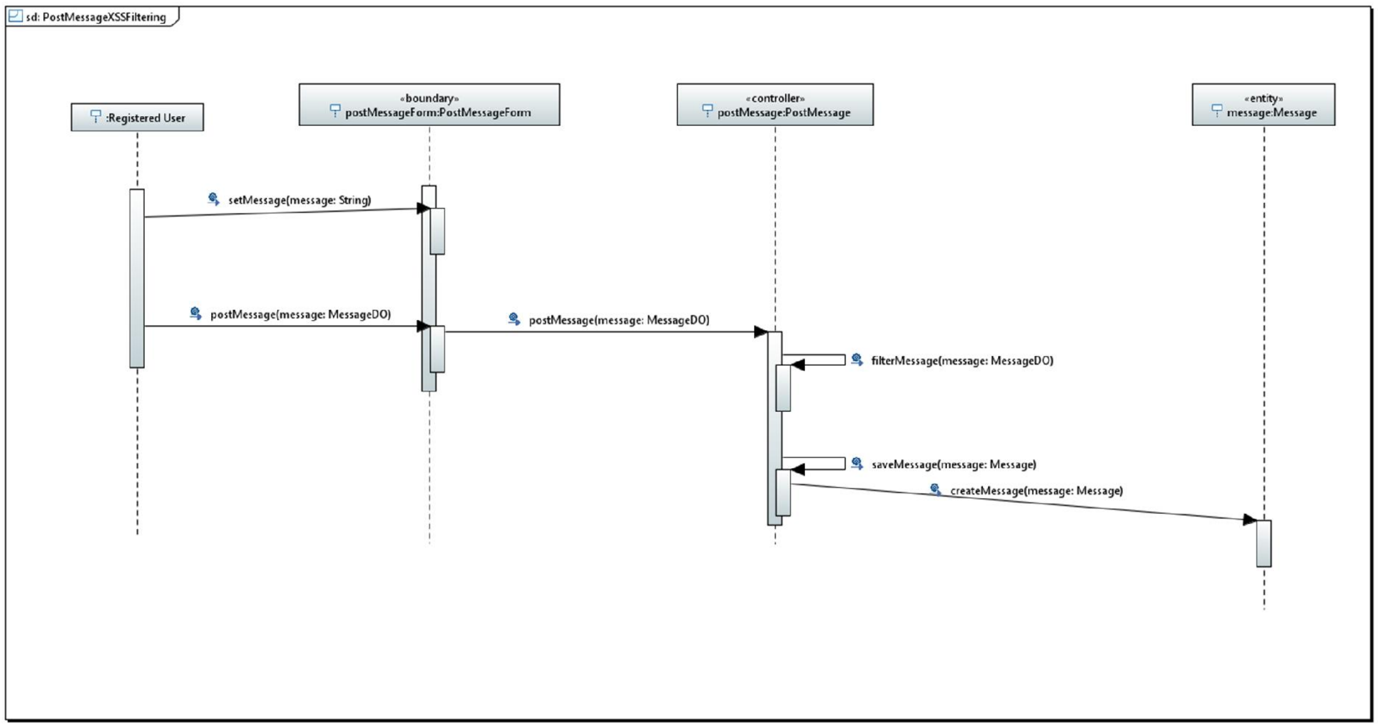
User Login and SQL injection prevention


sd: LoginSD
:User
«boundary» loginForm:LoginForm
«controller» login:Login
«entity» user:User
«controller» homePage:HomePage
setEmailId(emailId: String)
setPassword(password: String)
loginUser(user: UserDO)
loginUser(user: UserDO)
checkEmail(emailId: String): boolean
checkPassword(password: String): boolean
getUserByEmail(user: User): User
getUserByEmail(user: User): User
verifyUser(user: User, user2: User): boolean
navigateToHomePage(user: User)
showHomePage()

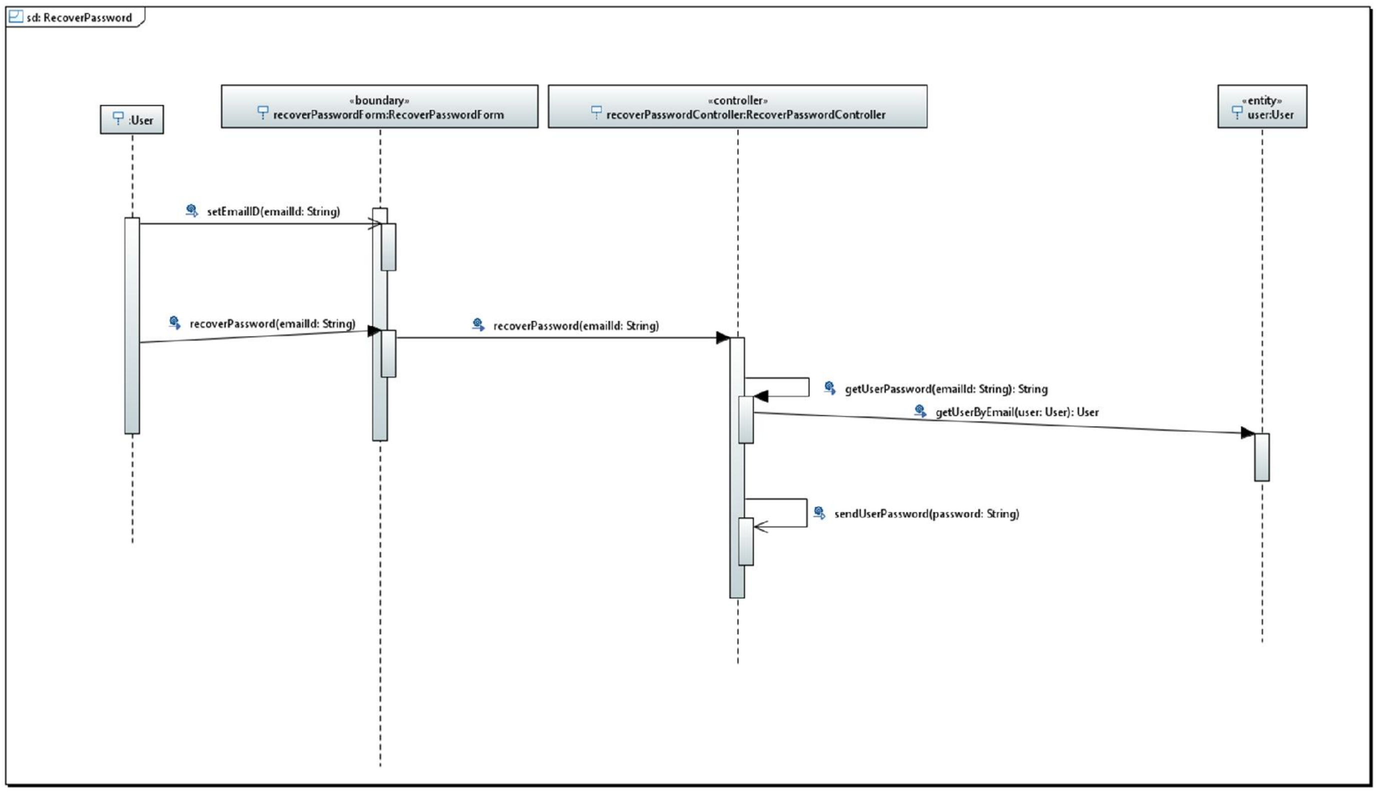
User Logout



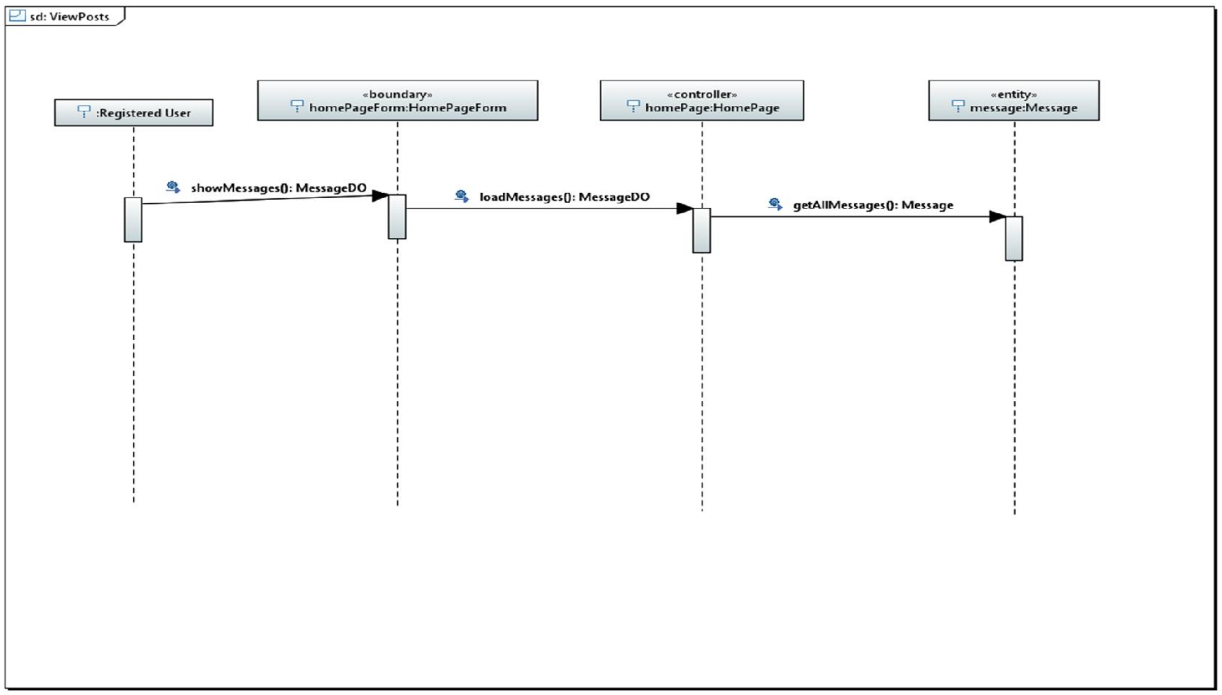
Post Message&XSS Filtering


sd: PostMessageXSSFiltering
:Registered User
«boundary» postMessageForm:PostMessageForm
«controller» postMessage:PostMessage
«entity» message:Message
setMessage(message: String)
postMessage(message: MessageDO)
postMessage(message: MessageDO)
filterMessage(message: MessageDO)
saveMessage(message: Message)
createMessage(message: Message)

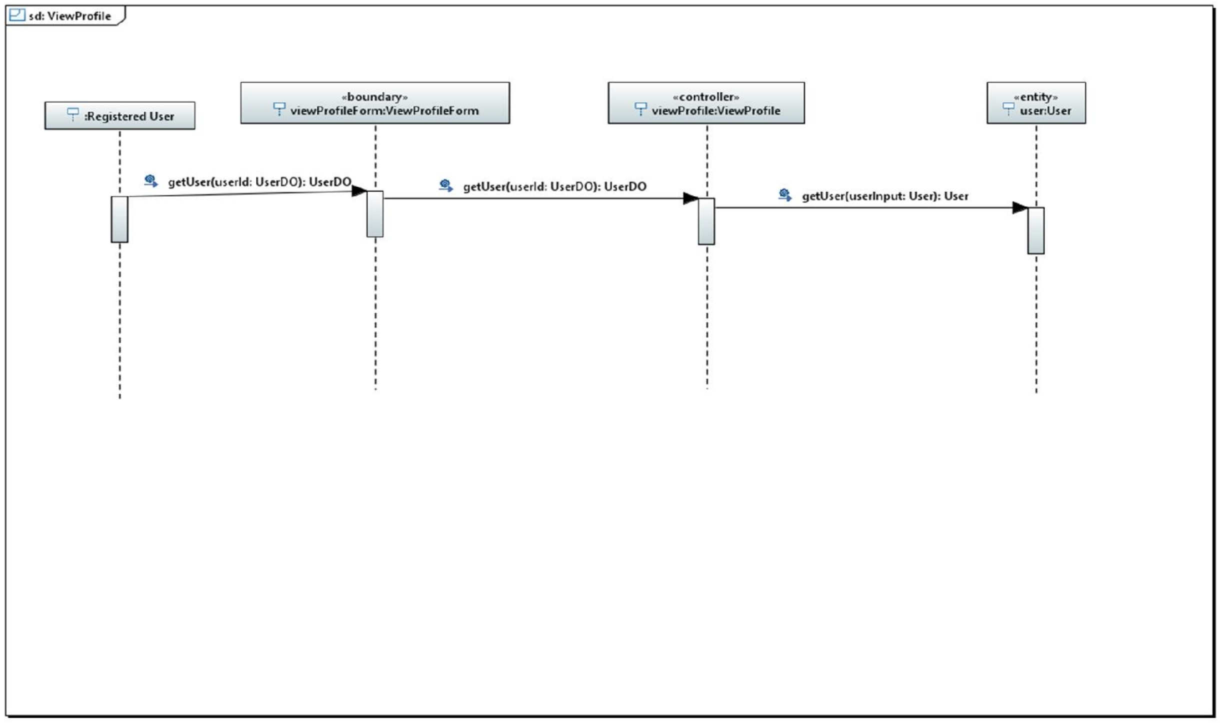
Recover Password


sd: RecoverPassword
:User
«boundary» recoverPasswordForm:RecoverPasswordForm
«controller» recoverPasswordController:RecoverPasswordController
«entity» user:User
setEmailID(emailId: String)
recoverPassword(emailId: String)
recoverPassword(emailId: String)
getUserPassword(emailId: String): String
getUserByEmail(user: User): User
sendUserPassword(password: String)

View post


sd: ViewPosts
:Registered User
«boundary» homePageForm:HomePageForm
«controller» homePage:HomePage
«entity» message:Message
showMessages(): MessageDO
loadMessages(): MessageDO
getAllMessages(): Message

View Profile


sd: ViewProfile
:Registered User
«boundary»
viewProfileForm:ViewProfileForm
«controller»
viewProfile:ViewProfile
«entity»
user:User
getUser(userId: UserDO): UserDO
getUser(userId: UserDO): UserDO
getUser(userInput: User): User

CHAPTER 6

6. Glossary - define terms used in document, especially domain specific terms.

TERMS	DEFINITIONS
Application Programming Interface (API)	Specifies a set of functions or routines that accomplish a specific task or are allowed to interact with a specific software component.
Unified modeling language (UML)	UML is a language used to visualize, specify, construct, and document the artifacts of a software-intensive system.
Cross-site Scripting (XSS)	Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it
JavaServer Faces (JSF)	It is a Java-based Web application framework that implements the Model-View-Controller pattern and simplifies the development of web interfaces for Java EE applications.
Subversion (SQL)	Is a software versioning and revision control system

7. Approval page with signatures of the team.

TEAM NAME: Panther Buddy

Team members of Panther Buddy participated in the preparation of this plan/charter, understanc contents, approve the plan/charter as the team's charter and operating plan, and agree to be hel mutually accountable for adherence to the plan/charter. Evidence of agreement is reflected by e team member's signature affixed below.

Abhinav Dutt ([illegible]).
Team Lead Print name

10. 5. [illegible]
Date

ALLAN MANAMEL ([illegible])
Team Member Name

10/5/2015
Date

ABDUR RAHMAN BIN SHAHID ([illegible])
Team Member Name

10/5/2015
Date

Yue Wu (April)
Team Member Name

10/5/2015
Date

Yulong Qiu ([illegible])
Team Member Name

10/5/2015
Date

Kedar Sharat
Team Member Name

10/5/2015
Date

James Angelo
Team Member Name

10/5/2015
Date

Project Planning Document Approval History

Approving Party	Version Approved	Signature	Date
Project Manager			

Project Planning Document Review History

Reviewer	Version Reviewed	Signature	Date
Group Member	1.0	[illegible]	10/05/2015
Group Member	1.0	[illegible]	10/05/2015
Group Member	1.0	[illegible]	10/5/2015
Group Member	1.0	[illegible]	10.5/2015
Group Member	1.0	[illegible]	10.5/2015
Group Member	1.0	[illegible]	10/5/2015
Group Member	1.0	[illegible]	10/5/2015

CHAPTER 8

8. References

1. Acamedia.stackexchange. (2015). *Post list*. Retrieved from Stackexchange: http://academia.stackexchange.com/

2. Ambler, S. W. (2009). *UML 2 Class Diagrams.* Weldon .

3. Burnett, M. (2007). *Blocking Brute Force Attacks.* Retrieved from Virginia Tech CS: http://www.cs.virginia.edu/~csadmin/gen_support/brute_force.php

4. C, J. D., & Darwen, H. (1997). *A Guide to the SQL standard : a user's guide to the standard database language SQL* (Vol. 4). USA: Addison Wesley.

5. Craiglist. (n.d.). *Show Contact*. Retrieved from Craiglist: http://www.craigslist.org/about/sites

6. Facebook. (2015). *Report*. Retrieved from Facebook help center: https://www.facebook.com/help/181495968648557

7. FIU. (2015). *student housing*. Retrieved from FIU student affairs: http://housing.fiu.edu/

8. Google . (2014). *Google reCaptcha*. Retrieved from Google reCAPTCHA: https://www.google.com/recaptcha/intro/index.html

9. group, F. s. (2014). *FIU Students Selling Textbooks*. Retrieved from Facebook: https://www.facebook.com/groups/287168468030148/

10. Group, O. M. (2005). *UML Superstructure Specification.*

11. Holus Associates. (2007). *UML Reference Card* (Vol. 2). Retrieved March 12, 2011

12. Microsoft. (2011). *SQL injection.* Retrieved from technet Microsoft: https://technet.microsoft.com/en-us/library/ms161953(v=sql.105).aspx

13. Owasp. (2014, 04 22). *Cross-site Scripting (XSS).* Retrieved from Owasp: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

14. Stackexchange. (2015). *Questions (Post)*. Retrieved from Stackexchange: http://codegolf.stackexchange.com/questions/ask

15. Stackexchange. (2015). *Tagging post*. Retrieved from stackexchange: http://codegolf.stackexchange.com/tags

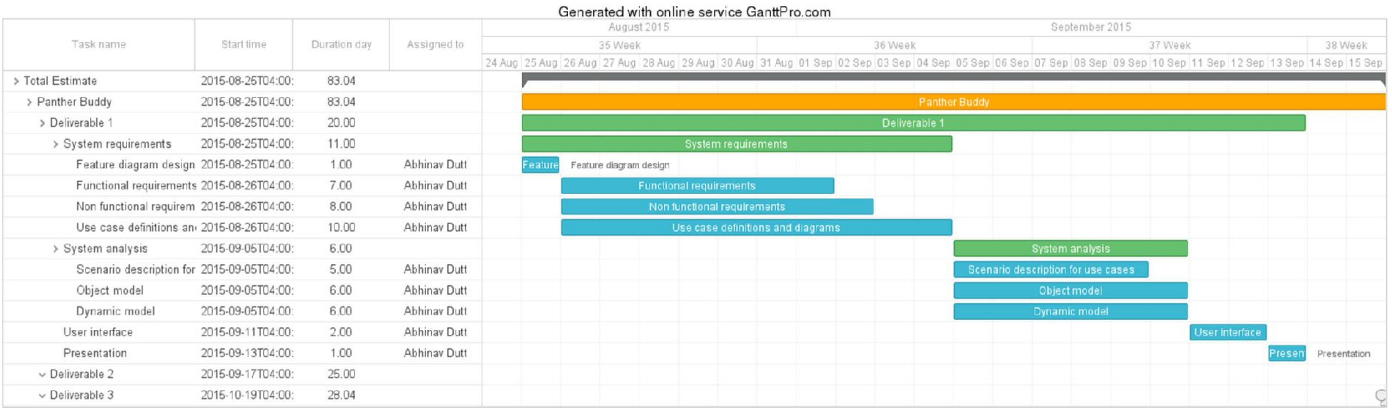
16. Tanji, M. (2013, 11). *CryptoLocker Decryption Engine.* Retrieved from KYRUS: http://www.kyrus-tech.com/2013/11/12/cryptolocker-decryption-engine/

17. Twitter. (n.d.). *Twitter Follow User*. Retrieved from Twitter: https://dev.twitter.com/web/follow-button

CHAPTER 9

9. Appendix
In this chapter, we present the project schedule, 21 use cases, 7 security use cases and 7 misuse cases of the 7 team members. The screenshots of the 10 use cases that we are implementing are also presented.

9.1 Appendix A - Project schedule


Generated with online service GanttPro.com

Task name	Start time	Duration day	Assigned to
› Total Estimate	2015-08-25T04:00:	83.04	
› Panther Buddy	2015-08-25T04:00:	83.04	
› Deliverable 1	2015-08-25T04:00:	20.00	
› System requirements	2015-08-25T04:00:	11.00	
Feature diagram design	2015-08-25T04:00:	1.00	Abhinav Dutt
Functional requirements	2015-08-26T04:00:	7.00	Abhinav Dutt
Non functional requirem	2015-08-26T04:00:	8.00	Abhinav Dutt
Use case definitions an	2015-08-26T04:00:	10.00	Abhinav Dutt
› System analysis	2015-09-05T04:00:	6.00	
Scenario description for	2015-09-05T04:00:	5.00	Abhinav Dutt
Object model	2015-09-05T04:00:	6.00	Abhinav Dutt
Dynamic model	2015-09-05T04:00:	6.00	Abhinav Dutt
User interface	2015-09-11T04:00:	2.00	Abhinav Dutt
Presentation	2015-09-13T04:00:	1.00	Abhinav Dutt
⌄ Deliverable 2	2015-09-17T04:00:	25.00	
⌄ Deliverable 3	2015-10-19T04:00:	28.04	

9.2 Appendix B – Use Cases and Misuse cases
In this section we present all the 28 use cases and misuse cases.

9.2.1 Use cases

1. USER REGISTRATION

Use case ID: PB_UC01_USER_REGISTRATION

Use case level: System level (End-to-End)

Details:

- **Actors**: Any anonymous user or an already registered user.
- **Pre-conditions:** The user has the user registration page for panther buddy open in a web browser.
- **Description:** The user registers himself on "Panther Buddy".
  The user will be asked to enter his details.
  1.1. First name: The first name of the user.
  1.2. Last name: The last name or surname of the user
  1.3. Email id: The email id the user wants to register on 'Panther Buddy' with.
  1.4. Phone number: The user primary phone number.
  1.5. The user clicks the submit button to submit the data.

1.6. The user data is saved in the database along with a generated password. His status is marked as inactive and a default date value (Sun, 2 Dec 292269055 BC 16:47:04 +0000) is put for his activation date.
1.7. The user is sent his password by mail.
1.8. The user is then redirected to the login page.

- **Post condition**: The user is marked as an inactive registered user in database.

Alternative courses of action:

1. At any time during step 1 the user can click the cancel button to return to the 'Panther Buddy' login page.

Extensions:

Exceptions:

1. The user at step 1.3 enters an email id that is already used to register by a user in the 'Panther Buddy' system. The user is asked to re-enter a new email id.

Related use cases:

PB_UC2_LOGIN, PB_UC3_ACTIVATE_USER, PB_UC4_INVALID_LOGIN, PB_UC5_RECOVER_PASSWORD.

--

Decision support:

Frequency: Moderate - Performed every time a new user wants to register on 'Panther Buddy' (30 times per day).

Criticality: Very Critical - If the user wants to use the 'Panther Buddy' site he needs to register. We also need to verify the user’s data input by him during registration for completeness and malicious activity.

Risk: High

Constraints:

- **Usability:** No previous training needed.
- **Mean time to failure:** 1 failure for every 2 months of operation is acceptable.
- **Performance**: The user data will be saved in 2 seconds.
- **Supportability:** The software will support all browsers that support HTML 5.
- **Implementation**: Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Allan Shaji Manamel

Initiation date: 09/07/2015

Date last modified: 09/14/2015

Example Scenario:

User registration

Actors: Allan Shaji Manamel

Pre-conditions Allan Shaji Manamel has the user registration page for panther buddy open in a web browser.

Description: Allan Shaji Manamel need to registers himself on "Panther Buddy".

The Allan Shaji Manamel will be asked to enter his details.

1. First name: Allan
2. Last name: Manamel
3. Email id: amana010@fiu.edu
4. Phone number: 987 654 2134

Trigger: The user clicks the submit button to submit the data.

1. Allan Shaji Manamel data is saved in the database along with a generated password (D43sx21). His status is marked as inactive and a default date value (Sun, 2 Dec 292269055 BC 16:47:04 +0000) is put for his activation date.
2. Allan Manamel sent his password (D43sx21) by email.
3. The user is then redirected to the login page.

Post-conditions: Allan Shaji Manamel is marked as an inactive registered user in database

2. RECOVER PASSWORD

Use case ID: PB_UC02_RECOVER_PASSWORD

Use case level: System level (End-to-End)

Details:

- **Actors**: Any user.
- **Pre-conditions**: The user has the login page open.
- **Description:** The user tries to recover his password, which is sent to him at his registered email.
  4. User clicks the recover password button on the login page
  5. The user is redirected to the recover password page.
  6. The user enters his email id used to register in the system.
  7. The user clicks the submit button to submit his email id.
  8. The user with the email-id is recovered and his password is sent to him by email.

- **Post condition:** The user gets his password on his registered email id.

Alternative courses of action:

6. At any time during step 1 to 4, the user can click the cancel button to return to the 'Panther Buddy' login page.

Extensions:

Exceptions:

4. The user at step 3 enters an email id that is not registered in the system. The user is asked to re-enter a valid email id.

Related use cases:

--

Decision support:

Frequency: Moderate - Performed every time a use forgets his password. (Used twice a day)

Criticality: Very Critical - If the user wants to use the 'Panther Buddy' site he needs to be able to login into the system, for which he needs his password.

Risk: High

--

Usability: No previous training needed.

Mean time to failure: 1 failure for every 2 months of operation is acceptable.

Performance:

2. The user will be sent a mail in 2 seconds.

Supportability:

The software will support all browsers that support HTML 5.

Implementation:

Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Allan Shaji Manamel

Initiation date:

Date last modified:

Example scenario:

Allan forgets his password.

4. Allan opens the 'Panther Buddy' login page on his laptop browser that supports HTML 5.
5. Allan clicks the recover password button.
6. Allan is redirected to the recover password page.
7. Allan enters his email id: amana576@fiu.edu used to register into the system.
8. Allan clicks the submit button to submit the data.
9. Allan's password: DX56ssUT is recovered by the system and sent to his email account.

3. CHANGE PASSWORD

Use case ID: PB_UC03_CHANGE_PASSWORD

Use case level: System level (End-to-End)

Details

- **Actors**: Logged in registered user.
- **Pre-conditions**: The user has his own profile page open on 'Panther Buddy'.
- **Description**: The user tries to change his password.
  1. User clicks the change password button on his profile page
  2. 3 fields appear on the page. They are:
     a) Current Password.

b) New Password
c) Confirm Password
3. The user enters the data asked
4. User enters his current password in the current password field
5. User enters his new desired password in the new password field.
6. User re-enters his desired new password in the confirm password field
7. The user clicks the submit button to submit his data.
8. The 3 Fields for password disappear.

- **Post condition**: The user's password is changed as per his desire.

Alternative courses of action:

1. At any time during step 1 to 4, the user can click the cancel button to return to the 'Panther Buddy' login page.

Extensions:

Exceptions:

1. The user at step 3.1 enters a wrong password. He is asked to correct his entered password.
2. The user at step 3.2 enters a password that does not fulfil the password criteria for 'Panther Buddy'. He is asked to correct his entered password.
3. The user at step 3.3 enters a password that does not match the password entered in step 3.2. He is asked to correct his entered password.

Related use cases:

--

Decision support:

Frequency: Moderate - Performed every time a user wants to change his password. (Used twice a day)

Criticality: Moderate

Risk: High

--

Usability: No previous training needed.

Mean time to failure: 1 failure for every 6 months of operation is acceptable.

Performance:

1. The user password will be changed in 2 seconds

Supportability:

The software will support all browsers that support HTML 5.

Implementation:

Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

--

Modification history

Owner: Allan Shaji Manamel

Initiation date:

Date last modified:

Example scenario:

Allan wants to change his password.

1. Allan logs in to the 'Panther Buddy' system and opens his profile page on laptop browser that supports HTML 5.
2. Allan clicks the change password button.
3. Fields appear on the page. They are:
   a) Current Password.
   b) New Password
   c) Confirm Password
4. Allan enters his current password = DX56ssUT in current password field.
5. Allan enters his new desired password = FS566ssCV in the new password field.
6. Allan re-enters his new desired password = FS566ssCV in the confirm password field.
7. Allan's password is changed by the system to his desired password

4. VIEW OWN PROFILE

Use case ID: PB_UC04_VIEW_OWN_PROFILE

Use case level: System level (End-to-End)

Details

- **Actors**: An already registered and logged in user.
- **Pre-conditions:** The user has panther buddy open in a web browser and logged in.
- **Description:** The user view his profile or other user's profile on "Panther Buddy".

- **Trigger:** The user clicks the view user profile button to submit the application.
  The user will have access to see the details below:
  1. View contact: View the contact information: name, phone, school and major of the user.
  2. View user posted message: View the history of message posted by user.
  3. View last log in time: View last log in time of the user.

- **Post condition**: The user can choose edit profile of himself or follow the user he viewed profile. The administrator can choose delete profile of the profile he viewed if needed.

Alternative courses of action:

At any time during editing profile the user can click the homepage button to return to the homepage of 'Panther Buddy' page.

Extensions:

Exceptions:

1. The user could not click edit profile bottom when view other's profile
2. The user could not click follow user bottom when view profile of himself.
3. The user could not click delete profile bottom if he is not an administrator.

Related use cases: user log in, edit profile, follow user and view posted message.

Decision support:

Frequency: High - Performed every time a user wants to view profile on 'Panther Buddy'

Criticality: High - If the user wants to view profile on the 'Panther Buddy' site, follow users, view user posted messages and edit his profile he need to access view profile first, or if an administrator wants to delete profile he need to view profile first. We also need to verify the user's log in statues.

Risk: Moderate

--

Usability: Logged in needed.

Mean time to failure: None.

Performance:

The user data will be showed less than 1 second from backend.

Supportability:

The software will support all browsers that support HTML 5.

Implementation:

Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Yue Wu

Initiation date: 09/06/2015

Date last modified: 09/18/2015

Example Scenario:

Actors: Allan

Pre-conditions Allan has panther buddy open in a web browser and logged in.

Description: Allan need view her profile on "Panther Buddy".

- Allan clicks the view user profile button to submit the application.
- Allan will have access to see the details below:
  1. View contact: View the contact information: name, phone, school and major of the user.
  2. View user posted message: View the history of message posted by user.
  3. View last log in time: View last log in time of the user.

Trigger: Allan clicks the view user profile button to submit the data.

Post-conditions: Allan can choose edit profile of herself or return to the home page.

5. EDIT PROFILE

Use case ID: PB_UC05_EDIT_PROFILE

Use case level: System level (End-to-End)

Details

- **Actors**: An already registered and logged in user.
- **Pre-conditions:** The user logged in and has the view user profile page for panther buddy open in a web browser.
- **Description:** The user edit their profile on "Panther Buddy".
- **Trigger:** The user click edit user profile button.
  The user will be asked to enter his details.
  1. Name: The name of the user.
  2. School and Major: The school and major of the user.
  3. Phone number: The users phone number. (optional)
  4. The user clicks the save button to submit the data.
  5. The user data is saved in the database.
  6. The user is then redirected to the view user profile page.
- **Post condition**: The data will be saved in the backend and its name will include the id of the user.

Alternative courses of action:

2. At any time during editing profile the user can click the cancel button to return to the view profile of 'Panther Buddy' page.

Extensions:

2. The user at step 1 enters a name ID that is already used to register by another user in the 'Panther Buddy' system. The user is asked to re-enter a new user name ID.
3. The user at step 1 enter a username included an invalid character. The user is asked to re-enter a new user name ID.
4. The user couldn’t edit other’s profile.

Related use cases: user log in, view user profile.

Decision support:

Frequency: Moderate - Performed every time a user wants to edit profile on 'Panther Buddy'

Criticality: Moderate - If the user wants to edit his profile on the 'Panther Buddy' site. We also need to verify the user’s data input by him during editing for completeness and malicious activity.

Risk: High - attacker may try to enter invalid user name character to hack database system.

Usability: Logged in needed.

Mean time to failure: 3 failures for every 24 hours of operation is acceptable.

Performance:

The user data will be saved in 2 seconds along with user ID.

Supportability:

The software will support all browsers that support HTML 5.

Implementation:

Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Yue Wu

Initiation date: 09/05/2015

Date last modified: 09/19/2015

Example Scenario:

Actors: Allan

Pre-conditions Allan logged in and has the view her user profile page for panther buddy open in a web browser.

Description: Allan need to edit his profile on "Panther Buddy".

- Allan click edit user profile button.
- Allan will be asked to enter her details.
  1. Name: The name of the user.
  2. School and Major: The school and major of the user.
  3. Phone number: The users phone number. (optional)

  Trigger: Allan clicks the save profile button to submit the data.
- Allan's data will be saved in the backend and its name will include the id of the user.
- Allan is then redirected to the view user profile page.

Post-conditions: Allan's data will be saved in the backend and its name will include the id of the user.

6. DELETE PROFILE

Use case ID: PB_UC06_DELETE_PROFILE

Use case level: System level (End-to-End)

Details

- **Actors**: An administrator.
- **Pre-conditions:** The administrator has logged in and view user profile page for panther buddy open in a web browser.
- **Description:** The administrator delate user's profile on "Panther Buddy".
- **Trigger:** The administrator clicks delete profile button to submit the application.
  1. The administrator click yes or no button to make sure his decision.
  2. The user data he choose to delete is deleted in the database.
  3. The administrator is then redirected to delete successful page.

Post condition: The administrator can choose back to the homepage of "Panther Buddy".

Alternative courses of action:

After click delete profile button, he have one more time to choose yes or no. An administrator could choose no and return to view profile page if he change his mind.

Extensions:

Exceptions:

1. The user cannot click delete profile button if he is not an administrator.

Related use cases: user log in, view user profile.

--

Decision support:

Frequency: Moderate - Performed only when an administrator wants to delete profile on 'Panther Buddy'

Criticality: Moderate - If the user trying to hack 'Panther Buddy' or ask us to delete his profile when he consider his private situation.

Risk: Moderate

--

Usability: Logged in needed. Administrator needed.

Mean time to failure: None.

Performance:

The user data will be delete less than 1 second.

Supportability:

It will be supported by IE-9, Firefox, Chrome, Safari and Mobile Based applications. The application will rely on Java Server Faces (JSF) and Bootstrap for the frontend. It will rely on Java and PostgreSQL database for backend.

Implementation:

Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Yue Wu

Initiation date: 09/06/2015

Date last modified: 09/19/2015

Example Scenario:

Actors: Administrator Allan

Pre-conditions: Administrator Allan has panther buddy open in a web browser and logged in.

Description: Administrator Allan need to delete a profile on "Panther Buddy".

Trigger: Allan clicks delete profile button to submit the application.

- Allan click yes button to make sure her decision.
- Data is deleted in the database.
- Allan is then redirected to delete successful page.

Trigger: Allan clicks the delete profile button to submit the application.

Post-conditions: Allan can choose return to the home page.

7. FOLLOW USER

Use Case ID: PB_UC07_FOLLOW_USER

Use Case Level:

Detail:

- **Actor:** registered user
- **Pre-conditions:** The user must logged-in. The user must be on the view profile page.
- **Description**: The user view the profile of others and follow the user if he is interested.
- **Trigger**: The user clicks "follow" button in view profile page.
  The system responds by adding the target user to the user's following list, and return one confirm message to user.
- **Post-conditions**: the target user is added to user's following list.

Alternative Courses of Action: None

Extensions: None

Exceptions: None

Related Use Cases: PB_UC_19_VIEW_OTHER_PROFILE

Decision Support

Frequency: medium-User

Criticality: moderate - to provide better UI for displaying message

Risk: none

Constraints:

Usability: none

Reliability: mean time to failure - 1 failure for every 24 hours of operation is acceptable.

Performance: the system should respond in 1 second.

Supportability: The application will be web-based. It will have supported on all sorts of modern web browser.

Implementation: Implementation will be done in HTML 5, java and Eclipse framework.

Modification History:

Owner: Yulong Qiu

Initiation date: 09/05/2015

Date last modified: 09/06/2015

Example Scenario:

Actor: Jim

Pre-condition: Jim is viewing the profile of others

Description: Jim follow the user who he is interested in.

Trigger: Jim click the button “follow” when he view the profile of other users’.

Post-condition: the user is added the following list of Jim’s by the system.

8. ACTIVATE USER

Use Case ID: PB_UC_08_ACTIVATE_USER

Use Case Level: System level (End-to-End)

Details:

- **Actor:** User
- **Pre-conditions:** The user should be registered on panther buddy
- **Description:** The user will hit “Login” button to activate his/her account on panther buddy.
- **Trigger:** The user clicks the “Login” button on the main page.

  The system responds by…

  1. The system prompts the user for their Panther account credentials.
  2. The user enters their username and password.
  3. The system authenticates the username and password
  4. The user gains access to the systems functionality.
  5. The user gets activated.

- **Post-conditions:**

1. The user is activated.

Alternative Courses of Action:

Extensions: None.

Exceptions: Incorrect login credentials will not activate the user.

Related Use Cases: Login Use case and Registration use case.

Decision Support

Frequency: Moderate

Criticality: High.

Risk: High- This is the core feature.

Constraints:

Usability

- None.

Reliability

- Mean time to failure - 1 failure for every 24 hours of operation is acceptable

Performance

- Mean time to failure.

Supportability

- The application will rely on HTML 5.

Implementation

- Implementation will be done in HTML 5, java and Eclipse framework.

Modification History:

Owner: Abhinav Dutt

Initiation date: 09/06/2015

Date last modified: 09/06/2015

Example Scenario:

User Activation

Actors: John

Pre-conditions: John has login page for panther buddy open in a web browser.

Description: John needs to login himself on "Panther Buddy".

The John will be asked to enter his details.

- User Id: John@fiu.edu
- Password: Manamel0012

Trigger: The user clicks the Login button to Login.

- John's profile is Activated and John is directed to home page

Post-conditions: John is marked as an active user in database.

10. FLAG MESSAGE

Use Case ID: PB_UC09_FLAG_MESSAGE

Use Case Level:

Detail:

- **Actor:** registered user
- **Pre-conditions:** The user must logged-in. The user must be on the message showing page.
- **Description**: The user flag the message which he is interested when view the messages.
- **Trigger**: The user clicks "Flag" message button from the menu.
- **Post-conditions**: the message id is added to the "flagged message" set of the user.

Alternative Courses of Action: None

Extensions: None

Exceptions: None

Related Use Cases: PB_UC_20_LOGIN, PB_UC10_VIEW_FLAGGED_MESSAGE

Decision Support

Frequency: Medium - User

Criticality: it's convenient for the user to use

Risk: None

Constraints:

Usability: None

Reliability: Mean time to failure - 1 failure for 24 hours of operation is acceptable.

Performance: the system respond in 1 second.

Supportability: The application will be web-based. It will have supported on all sorts of modern web browser.

Implementation: Implementation will be done in HTML 5, java and Eclipse framework.

--

Modification History:

Owner: Yulong Qiu

Initiation date: 09/05/2015

Date last modified: 09/06/2015

Example Scenario:

Actor: Jim

Pre-condition: Jim is on the message view page.

Description: Jim flag the message which he is interested.

Trigger: Jim click the "flag" button besides the message which he is viewing.

Post-conditions: the message is added the Jim's "flagged message set" in the system.

10. VIEW FLAGGED MESSAGE

Use Case ID: PB_UC10_VIEW_FLAGGED_MESSAGE

Use Case Level:

Detail:

- **Actor:** registered user
- **Pre-conditions:** The user must logged-in. The user must be on the Flagged message checking list.

- **Description**: The user view the flagged message which he is interested when view the messages.
- **Trigger**: The user clicks flagged message in the flagged message list.

The system responds by show the message chosen.

Post-conditions: the system redirect the user to the message show page, which show the message chosen.

Alternative Courses of Action: None

Extensions: None

Exceptions: None

Related Use Cases: PB_UC_20_LOGIN, PB_UC10_VIEW_FLAGGED_MESSAGE

Decision Support

Frequency: medium-User

Criticality: moderate - to provide better UI for displaying message

Risk: none

Constraints:

Usability: none

Reliability: mean time to failure - 1 failure for every 24 hours of operation is acceptable.

Performance: the system should respond in 1 second.

Supportability: The application will be web-based. It will have supported on all sorts of modern web browser.

Implementation: Implementation will be done in HTML 5, java and Eclipse framework.

Modification History:

Owner: Yulong Qiu

Initiation date: 09/05/2015

Date last modified: 09/06/2015

Example Scenario

Actor: Jim

Pre-conditions: Jim is on the page of view flag message list page.

Description: Jim check the message which he flagged before.

Trigger: Jim click the message in the list which he flagged before.

Post-conditions: Jim is redirected to the message show page by the system where message content is displayed.

11. POST MESSAGE

Use Case ID: PB _UC11_POST_MESSAGE

Use Case Level:

Details:

- **Actor:** User
- **Pre-conditions:** The user must be logged-in. The user must fill the information that he wants and post it.
- **Description:** The user will add post by filling the information that he wants and post it or attach file.
- **Trigger:** The user clicks add post button from the menu.

  The system responds by

  6. A post window will appear.
  7. The user can specify the post URL or browse to the specific file or write in the winnow to post.
  8. By clicking "Open" button, the image will be added to the post.
- **Post-conditions:** The post will be saved in the backend and its name will include the id of the post.

Alternative Courses of Action:

At steps 2 and 3, the user can hit "Cancel" to cancel post.

Extensions: Drag-and-drop file.

Exceptions: the post need to be accepted by moderator.

Related Use Cases: Post messages.

--

Decision Support

Frequency: Medium - User

Criticality: Moderate - to provide better visualization of the post.

Risk: Medium - attacker may try to post virus.

Constraints:

Usability

- None.

Reliability

- Mean time to failure - 1 failure for every 24 hours of operation is acceptable.

Performance

- The file or text should be saved along with the post.

Supportability

- The application will be web-based. It will have supported on all sorts of modern devices.

Implementation

- Implementation will be done in HTML 5, java and Eclipse framework.

Modification History:

Owner: Kedari Sharat

Initiation date: 09/05/2015

Date last modified: 09/06/2015

Example Scenario:

Post message

Actors: Kedari Sharat

Pre-conditions: Kedari Sharat must be logged-in. Kedari Sharat must fill the information that he wants and post it.

Description: Kedari Sharat will add post by filling the information that he wants and post it or attach file.

Trigger: Kedari Sharat clicks add post button from the menu.

The system responds by…

i. A post window will appear.

3. Sharat specify the post URL or browse to the specific file or write in the winnow to post.
4. By clicking “post” button, the image will be added to the post.

Post-conditions: The post will be saved in the backend and its name will include the id of the post.

12. EDIT MESSAGE

Use Case ID: PB_ UC12_EDIT_MESSAGE

Use Case Level:

Details:

- **Actor:** User
- **Pre-conditions:** The user must be logged-in.
- **Description:** The user will hit "edit" within a post to edit his own post.
- **Trigger:** The user clicks the "edit" button from the menu.

  The system responds by…

  1. Showing the original message to edit.
  2. The user edits the message.
  3. Upon hitting "edit" button, the message sends for moderation.

- **Post-conditions:** only owner has authority to edit the message.

Alternative Courses of Action:

At steps 1 and 2, the user can hit "Cancel" to cancel the modifying the post.

Extensions: None.

Exceptions: None.

Related Use Cases: Log in, post message.

Decision Support

Frequency: Medium - User

Criticality: Moderate - to provide better visualization of the post.

Risk: Medium - attacker may try to post virus.

--

Constraints:

Usability

- None.

Reliability

- Mean time to failure - 1 failure for every 24 hours of operation is acceptable.

Performance

- The file or text should be saved along with the post.

Supportability

- The application will be web-based. It will have supported on all sorts of modern devices.

Implementation

- Implementation will be done in HTML 5, java and Eclipse framework.

Modification History:

Owner: Kedari Sharat

Initiation date: 09/05/2015

Date last modified: 09/06/2015

Example scenario

Actor: Sharat

Pre-conditions: Sharat must be logged-in.

Description: Sharat will hit "edit" within a post to edit his own post.

Trigger: Sharat clicks the "edit" button from the menu.

The system responds by…

1. Showing the original message to edit.

2 Sharat edits the message.

3 upon hitting "edit" button, the message sends for moderation.

Post-conditions: only owner has authority to edit the message.

13. SUBSCRIBE NOTIFICATION

Use Case ID: PB_UC13_SUBSCRIBE

Use Case Level: System level (End-to-End)

Details:

- **Actor:** User
- **Pre-conditions:** User must be logged-in, must know in which news he/she is interested.
- **Description:** The user will be subscribed to Email Alter for some specific topics so that whenever a new message on that topics will be posted, he/she will be notified via email.
- **Trigger:** The user will click Email alert option.

The system responds by…

1. Showing a form.

2. The user will enter the interested keywords and press button "create".

3. The system will create an alert for that user.
4. Whenever a message with those keywords will be posted, system will send email to that user.

Post-conditions: The alert id would be associated with user's id.

Alternative Courses of Action:

At steps 1 and 2, the user can hit "Cancel" to cancel the creating email alert.

Extensions: None.

Exceptions: A user may create alert with same keywords, the system needs to prevent that.

Related Use Cases: PB_UC_04_IVEW_OWN_PROFILE.

Decision Support

Frequency: Low.

Criticality: Low.

Risk: Low.

Constraints:

Usability

- The users should be aware of little delay in case of getting email alerts because of large number of such alert requests.

Reliability

- Mean time to failure - 1 failure in sending email for every 24 hours of operation is acceptable.

Performance

- Email should be send by 1 hour of posting message.

Supportability

- The application will be web-based. It will be supported on all sorts of modern devices.

Implementation

- Implementation will be done in HTML 5, java and Eclipse framework.

Modification History:

Owner: Abdur Rahman Bin Shahid

Initiation date: 09/02/2015

Date last modified: 09/05/2015

Example Scenario:

Actor: Adam

Pre-condition: Adam is logged in and he is in his profile page.

Description: Adam wants to be subscribed to the PantherBuddy's notification system for some specific interest to get automatic update.

Trigger:

1) Adam will go to subscribe notification page.

2) He clicks "create notification" button.

3) A form will appear.

4) He enters the keywords.

5) He press "Create" button.

14. DELETE MESSAGE

Use Case ID: PB _UC14_DELETE_MESSAGE

Use Case Level:

Details:

- **Actor:** User
- **Pre-conditions:** The user must be logged-in. The user must be owner of that post that he wants to delete
- **Description:** The user will delete the post that has privileges to do so
- **Trigger:** The user clicks delete post button from the menu.

  The system responds by…

  1. A window will appear.
  2. The user needed to confirm wither he want to delete the post .
  3. By clicking "ok" button, the post will be deleted.

Post-conditions: The page will be loaded by deleting the post

Alternative Courses of Action:

At steps 2 and 3, the user can hit "Cancel" to cancel deletion.

Extensions: none.

Exceptions: The post need to be owned by user.

Related Use Cases: log-in, Post messages.

Decision Support

Frequency: Medium - User

Criticality: Moderate - to provide better visualization of the post.

Risk: Medium.

Constraints:

Usability

- None.

Reliability

- Mean time to failure - 1 failure for every 24 hours of operation is acceptable.

Performance

- The file or text should be saved along with the post.

Supportability

- The application will be web-based. It will have supported on all sorts of modern devices.

Implementation

- Implementation will be done in HTML 5, java and Eclipse framework.

Modification History:

Owner: Kedari Sharat

Initiation date: 09/05/2015

Date last modified: 09/06/2015

Example scenario:

Actor: Sharat

Pre-conditions: Sharat must be logged-in. Sharat must be owner of that post that he wants to delete

Description: Sharat will delete the post that has privileges to do so

.

Trigger: Sharat clicks delete post button from the menu.

The system responds by…

4. A window will appear.
5. Sharat needed to confirm wither he want to delete the post .
6. By clicking "ok" button, the post will be deleted.

Post-conditions: The page will be loaded by deleting the post

15. SEARCH POST

Use Case ID: PB_UC15_SEARCH_POST

Use Case Level: High-level

Details:

- **Actor:** Registered User
- **Pre-conditions:** The registered user has to login in order to be able to search the system.
- **Description:** The registered user places the cursor in the search bar on the homepage. The registered user types in the search word and clicks on the search icon.
- **Trigger:** The registered user initiates the action by entering the search word in the search bar. The system responds by crawling a database searching for the relevant information.
- **Relevant requirements:** None
- **Post-conditions:** The system returns the relevant information. If no information was found based on the search word entered, the system returns no results. After the search, there is an increment in the number of times the system's database is queried.

Alternative Courses of Action: If the relevant is found, the registered will be able to view it.

Extensions: None

Exceptions: The registered user might mispel the search word, which will result in the system not being able to find it.

Concurrent Uses: None

Related Use Cases: Viewpost

Decision Support

Frequency: More than 30 times in an hour

Criticality: High

Risk: Moderate

--

Constraints:

- **Usability:** The registered user should be able to search the system at any moment
- **Reliability:** Mean time to failure is about 5 failures for every 24 hours of operation
- **Performance:** The relevant information should be returned in less than two seconds
- **Supportability:** The application will support text in any searchable format
- **Implementation:** Bootstrap frontend framework and Java JSF

--

Modification History:

Owner: James W. Angelo

Initiation date: 09/22/2015

Date last modified: 09/22/2015

16. VIEW POST

Use Case ID:PB_UC16_VIEW_POST

Use Case Level: High-level

Details:

- **Actor:** Registered User
- **Pre-conditions:** The registered user has to login in order to be able to view a post.
- **Description:**If the user searches the system and the system returns some relevant information, the user then clicks on the title of the post to view it.
- **Trigger:** The registered user initiates the action by clicking on the title of the post. The system responds by displaying a longer version of the post.
- **Relevant requirements:** None
- **Post-conditions:**The system displays a longer version of the post once the user clicks on the title of the post.
- **Alternative Courses of Action:**The user then proceeds to read the information.
- **Extensions:** None

Exceptions: The system might not be able to display a longer version of the message.

Concurrent Uses: None

Related Use Cases: Searching

Decision Support

Frequency: More than 30 times in an hour

Criticality: Moderate

Risk: Moderate

Constraints:

- **Usability:** The registered user should be able to view any post at any moment
- **Reliability:** Mean time to failure is about 3 failures for every 24 hours of operation
- **Performance:** The post should be displayed in less than two seconds
- **Supportability:** The application will support any text format
- **Implementation:** Bootstrap frontend framework and Java JSF

Modification History:

Owner: James W. Angelo

Initiation date: 09/22/2015

Date last modified: 09/22/2015

17. SORT POST

Use Case ID:PB_UC17_SORT_POST

Use Case Level: System-level end-to-end

Details:

- **Actor:** The system
- **Pre-conditions:** A registered user has to log in and searches the system.
- **Description:**Once the registered user logs in and searches the system, the system will sort the relevant information in order to present the most accurate information to the user.
- **Trigger:** The registered user initiates the action by entering the search word in the search bar. The system responds in the background by sorting the relevant information.
- **Relevant requirements:** None
- **Post-conditions:** The system sorts the relevant information.
- **Alternative Courses of Action:**Once the information is sorted, the sytem will present the most accurate information to the user based on order of relevance.
- **Extensions:** None

Exceptions: The registered user might mispel the search word, which will result in the system not being able to find it. Since the system cannot find the searchable word, it will not be able to sort it.

Concurrent Uses: None

Related Use Cases: Searching

--

Decision Support

Frequency: More than 30 times in an hour

Criticality: Moderate

Risk: Moderate

Constraints:

- **Usability:** The system should be able to sort the relevant information
- **Reliability:** Mean time to failure is about 6 failures for every 24 hours of operation
- **Performance:** The relevant information should be sorted in less than 1 second
- **Supportability:** The application should be able to sort text represented by ASCII or unicode

- **Implementation:** Bootstrap frontend framework and Java JSF

Modification History:

Owner: James W. Angelo

Initiation date: 09/22/2015

Date last modified: 09/22/2015

18. UPLOAD FILE

Use Case ID: PB_UC18_UPLOAD_FILE

Use Case Level: System level (End-to-End)

Details:

- **Actor:** User
- **Pre-conditions:** The user must be logged-in. The user knows the full path to the file.
- **Description:** The user will add a file to post and will add additional file metadata to describe the file.
- **Trigger:** The user clicks the file upload button from the menu.

  The system responds by…

  9. A file uploader window will appear.
  10. The user can specify the file URL or browse to the specific file.
  11. By clicking "Open" button, the file will be added to the post.
  12. The user will provide description of the file.
- **Post-conditions:** The file will be saved in the backend and its name will include the id of the post.

Alternative Courses of Action:

At steps 2 and 3, the user can hit "Cancel" to cancel uploading file.

Extensions: Drag-and-drop of the file.

Exceptions: The user cannot add file with same name multiple times for the same post. Malicious user can try to upload virus. File verification is required.

Related Use Cases: PB_UC_11_POST_MESSAGE.

Decision Support

Frequency: Medium - User

Criticality: Moderate - to provide better understanding of the post, file may require.

Risk: Medium - attacker may try to post virus.

Constraints:

Usability

- None.

Reliability

- Mean time to failure - 1 failure for every 24 hours of operation is acceptable.

Performance

- The file should be saved along with the post.

Supportability

- The application will be web-based. It will supported on all sorts of modern devices.

Implementation

- Implementation will be done in HTML 5, java and Eclipse framework.

Modification History:

Owner: Abdur Rahman Bin Shahid

Initiation date: 09/02/2015

Date last modified: 09/05/2015

Example Scenario:

Actor: Adam

Pre-condition: Adam is writing a post.

Description: Adam is writing a post and wants to add a file to the post.

Trigger:

1. He click “Upload File” button from the menu.
2. It will open file explorer window.
3. Adam knows the path of the file, he select the file from there.
4. By clicking “open” button in the file explorer, the image will be added to the post.
5. He will provide description of the file.

19. VIEW OTHER PROFILE

Use Case ID: PB_UC19_VIEW_OTHER_PROFILE

Use Case Level: System level (End-to-End)

Details:

- **Actor:** User
- **Pre-conditions:** User is viewing a post.
- **Description:** The user will hit “Show Profile” within a post to get the related information of the owner of that post.
- **Trigger:** The user clicks the “show profile” button.

  The system responds by…

  1. The system will show the profile of the user.

- **Post-conditions:** None.

Alternative Courses of Action:

At steps 1, the user can hit “Cancel” to cancel the showing contact.

Extensions: None.

Exceptions: None.

Related Use Cases: PB_UC_16_VIEW_POST, PB_UC_24_ SHOW_CONTACT.

Decision Support

Frequency: High

Criticality: High.

Risk: High – profile information of the owner of a post must be provided. This is a core feature.

Constraints:

Usability

- No previous training need.

Reliability

- Mean time to failure – 0.

Performance

- Profile of other user must be shown within 2 seconds.

Supportability

- The application will be web-based. It will be supported on all sorts of modern devices.

Implementation

- Implementation will be done in HTML 5, java and Eclipse framework.

Modification History:

Owner: Abdur Rahman Bin Shahid

Initiation date: 09/02/2015

Date last modified: 09/05/2015

Example Scenario:

Actor: Adam

Pre-condition: Adam is viewing a post.

Description: Adam wants to view the profile of the owner of the post.

Trigger:

1. He click the "View profile" option.
2. The page containing the profile of the post owner will be shown to him.

20. LOGIN

Use case ID: PB_UC20_LOGIN

Use case level: System level (End-to-End)

Details

- **Actors**: Any user.
- **Pre-conditions**: The user has the login page open.
- **Description**: The user logins into the system using his email and password.
- **Trigger**:
    1. User enters his email into the email field
    2. The user enters his password in the password field
    3. The user clicks the login button to submit his data.
- **Post condition**: The user is redirected to the Panther Buddy home page.

Extensions: PB_SC7_INVALID_LOGIN

Exceptions:

Related use cases: PB_SC1_FILTER_INPUT

Decision support:

Frequency: Moderate - Performed every time a user logins to the Panther buddy system

Criticality: Very Critical - If the user wants to use the 'Panther Buddy' site he needs to be able to login into the system.

Risk: High

Usability: No previous training needed.

Mean time to failure: 1 failure for every 2 months of operation is acceptable.

Performance:

1. The user will be redirected to the home page in 2 seconds

Supportability:

The software will support all browsers that support HTML 5.

Implementation:

Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Allan Shaji Manamel

Initiation date:

Date last modified:

Example scenario:

Allan forgets his password.

1. Allan opens the 'Panther Buddy' login page on his laptop browser that supports HTML 5.
2. Allan enters his email amana100@fiu.edu into the email field.

3. Allan enters his password DX345tv in the password field.
4. Allan clicks the login button to submit the data.

21. LOGOUT

Use case ID: PB_UC21_LOGOUT

Use case level: System level (End-to-End)

Details

- **Actors**: Registered user.
- **Pre-conditions**: The user has logged into the system and is on any page which only a logged in user can view.
- **Description**: The user logs out from the system.
- **Trigger**:
    1. User clicks the logout button on the screen
    2. The system destroys the users session and redirects him to the login page

- **Post condition**: The user is redirected to the Panther Buddy login page.

Extensions:

Exceptions:

Related use cases: PB_SC3_SESSION_TIMEOUT

--

Decision support:

Frequency: Moderate - Performed every time a user logs out from the Panther buddy system

Criticality: low

Risk: low

--

Usability: No previous training needed.

Mean time to failure: 1 failure for every 2 months of operation is acceptable.

Performance:

1. The user will be redirected to the login page in 2 seconds

Supportability:

The software will support all browsers that support HTML 5.

Implementation:

Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Allan Shaji Manamel

Initiation date:

Date last modified:

Example scenario:

Allan forgets his password.

1. Allan opens the Panther Buddy home page after login.
2. Allan clicks on the logout button on the page.
3. Allan's session is destroyed and he is redirected to the Panther Buddy login page

22. DEACTIVATE USER

Use Case ID: PB_UC22_DEACTIVATE_USER

Use Case Level: System level (End-to End)

Details:

- **Actor:** Admin
- **Pre-conditions:** The Admin should be logged in and has user profile page open.
- **Description:** The Admin can deactivate his/her profile.
- **Trigger:** The Admin clicks the deactivate button.

  The system responds by…

  1. The admin can click ok button to confirm the deactivation of his/her account.
  2. The admin can click cancel button also to deny the deactivation.
- **Post-conditions:** The user will be deactivated and activation date would be set to some future date.

Alternative Courses of Action:

At step 3, the Admin can hit “Cancel” to cancel the deactivation of user profile.

Extensions:

Exceptions: NA

Related Use Cases:

Decision Support

Frequency: Medium- .

Criticality: Moderate

Risk: Low

Constraints:

Usability

- None.

Reliability

- Mean time to failure - 1 failure for every 24 hours of operation is acceptable

Performance

- None

Supportability

- The application will rely on HTML 5.

Implementation

- Implementation will be done in HTML 5, java and Eclipse framework.

Modification History:

Owner: Abhinav Dutt

Initiation date: 09/05/2015

Date last modified: 09/05/2015

Example Scenario:

Deactivate User Profile

Actors: John –Administrator, Danny -user

Pre-conditions John as administrator has logged in to deactivate the Danny user profile from panther buddy.

Description: Danny should be registered user on "Panther Buddy".

Trigger: The John clicks deactivate user profile button and Danny profile is deactivated temporarily.

Post-conditions: Danny is status would be marked as deactivated and activation date would be set to some future date.

----

23. HELP & CONTACT

Use Case ID: PB_UC23_HELP_CONTACT.

Use Case Level: System level (End-to End)

Details:

- **Actor:** User
- **Pre-conditions:**
  1. The user should be registered and active on panther buddy
  2. The user is logged in
- **Description:** The user clicks on help and contact button to view the help & contact details mentioned on the website.
- **Trigger:** User clicks on help and contact button

  The system responds by

  1. Help and Contact web page is displayed , where user can find the contact details of the admin of website ,in case if he/she needs any kind of support.
- **Post-conditions:**

1 The user is able to see the help and contact information mentioned on the web page.

Alternative Courses of Action: N/A

Extensions: None.

Exceptions: None

Related Use Cases: Login into System

--

Decision Support

Frequency: Whenever the user wants to log out

Criticality: Medium.

Risk: None

--

Constraints:

Usability

- None.

Reliability

- Mean time to failure - 1 failure for every 24 hours of operation is acceptable

Performance

- None.

Supportability

- The application will rely on HTML 5.

Implementation

- Implementation will be done in HTML 5, java and Eclipse framework.

Modification History:

Owner: Abhinav Dutt

Initiation date: 09/06/2015

Date last modified: 09/06/2015

Example Scenario:

Help& Contact

Actors: John

Pre-conditions: John is active user on panther buddy and has logged in.

Description: John clicks on the Help & Contact button to view the contact details.

Trigger: The john clicks the help & contact button

1. Help & Contact web page is displayed where user can see all the relevant information related to help & contact

Post-conditions: John is able to see the help the contact information.

24. SHOW CONTACT

Use Case ID: PB_UC24_ SHOW_CONTACT

Use Case Level: System level (End-to-End)

Details:

- **Actor:** User
- **Pre-conditions:** User is viewing a profile of other user.

- **Description:** The user will hit "Show contact" within a profile to get the contact information of the owner of that profile, after security verification the contact will be showed.
- **Trigger:** The user clicks the "show contact" button from the menu.
  The system responds by…
    Security check.
    Upon successful check, the system will show the contact information.
- **Post-conditions:** Every time the page is refreshed the contact information of the user must be hidden.

Alternative Courses of Action:

At steps 1, the user can hit "Cancel" to cancel the showing contact.

Extensions: None.

Exceptions: None.

Related Use Cases: PB_UC_19_VIEW_OTHER_PROFILE.

Decision Support

Frequency: High – 1/post.

Criticality: High.

Risk: High – contact information must be provided. This is a core feature.

Constraints:

Usability

- No previous training need.

Reliability

- Mean time to failure - 1 failure in case of showing "I'm not a robot" dialog for every 24 hours of operation is acceptable.

Performance

- Contact information must be shown within 2 seconds.

Supportability

- The application will be web-based. It will be supported on all sorts of modern devices.

Implementation

- Implementation will be done in HTML 5, java and Eclipse framework.

Modification History:

Owner: Abdur Rahman Bin Shahid

Initiation date: 09/02/2015

Date last modified: 09/05/2015

Example Scenario:

Actor: Adam

Pre-condition: Adam is viewing a post related to his interest. He wants to know more detail about the post owner.

Description: Adam wants to contact the owner of the post.

Trigger:

1) He click the view contact option.
2) After security verification, the contact information will be shown

9.2.2 Misuse cases

1. FILTER INPUT

Use case ID: PB_SC1_FILTER_INPUT

Use case level: System level (End-to-End)

Details

- **Actors**: Any user.
- **Pre-conditions**: The user has the login page open.
- **Description**: The user logins into the system using his email and password.
- **Trigger**:
  9. User enters his email into the email field
  10. The user enters his password in the password field
  11. The user clicks the login button to submit his data.
  12. The system checks the email and password for malicious content before querying for the user with the entered login credentials

- **Post condition**: The user is redirected to the Panther Buddy home page.

Extensions:

Exceptions:

Related use cases: PB_UC_20_LOGIN, PB_SC7_INVALID_LOGIN

Decision support:

Frequency: Moderate - Performed every time a user logins to the Panther buddy system

Criticality: Very Critical

Risk: High

Usability: No previous training needed.

Mean time to failure: 1 failure for every 2 months of operation is acceptable.

Performance:

3. The user will be redirected to the home page in 2 seconds

Supportability:

The software will support all browsers that support HTML 5.

Implementation:

Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Allan Shaji Manamel

Initiation date:

Date last modified:

Example scenario:

Allan forgets his password.

10. Allan opens the 'Panther Buddy' login page on his laptop browser that supports HTML 5.
11. Allan enters his email amana100@fiu.edu into the email field.
12. Allan enters his password DX345tv in the password field.
13. Allan clicks the login button to submit the data.
14. The system checks the input email amana100@fiu.edu and password DX345tv for malicious content

Use case ID: PB_MC1_FILTER_INPUT

Use case level: System level (End-to-End)

Details

Actors: Misuser.

Pre-conditions: The misuser has the login page open.

Description: The user gains entry into the system.

5. misuser enters his email into the email field
6. The misuser enters a password : dummy' OR 1=1 OR '' = '
7. The user clicks the login button to submit his data.
8. The system if unprotected will put the string in the where clause of a SQL statement used to compare existing data for input data like WHERE password = 'dummy' OR 1=1 OR '' = '', thereby gaining entry into the system.

Post condition: The user is redirected to the Panther Buddy home page.

Extensions:

Exceptions:

Related use cases: PB_UC_20_LOGIN, PB_SC7_INVALID_LOGIN

Criticality: Very Critical

Risk: High

2. XSS FILTERING

Use case ID: PB_SC2_XSS_FILTERING

Use case level: System level

Details

Actors: Registered user.

Pre-conditions: The user has the post message page open.

Description: The user message is filtered to check for XSS attack.

1. The user writes his message in the space provided for writing his message.
2. The user clicks the post message button to submit the message.
3. The system checks the message string for malicious code.
4. System saves the message for moderation by moderator.

Post condition: The user's message is checked for malicious code. Once verified they are saved in the system for moderation by moderator.

Alternative courses of action:

7. At any time during step 1 to 4, the user can click the cancel button to return to the home page.

Extensions:

Exceptions:

5. The user at step 1 puts malicious code in his message. The system shows an error for in proper content at asks user to rectify it.

Related use cases:

Decision support:

Frequency: High - Performed every time a user posts his message. (Used 20 times a day)

Criticality: Very

Risk: High

Usability: No previous training needed.

Mean time to failure: 1 failure for every 2 months of operation is acceptable.

Performance:

4. The user will be able to submit his message in 2 sec

Supportability:

The software will support all browsers that support HTML 5.

Implementation:

Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Allan Shaji Manamel

Initiation date:

Date last modified:

Example scenario:

Allan submits his message.

1. Allan logs into the 'Panther Buddy' system and opens the post message page on his laptop browser that supports HTML 5.
2. Allan writes his message "Used book - Introduction to Algorithms by Thomas Cormen up for sale at 19$".
3. Allan clicks the submit button to submit his message.
4. The system checks if the message contains any malicious code or not.
5. The system verifies the message to be ok.
6. The system saves the message to be moderated by a moderator.

Misuse case
Use case ID: PB_MC2_XSS_FILTERING

Use case level: System level

Details

Actors: Misuser.

Pre-conditions: The user has the post message page open.

Description: The user embeds malicious scripts.

3. The user writes his message in the space provided for writing his message.
4. The user clicks the post message button to submit the message.

Post condition: The user's message when rendered on page runs the script to hack into the system to fetch/manipulate unauthorised data from the system.

Criticality: Very

Risk: High

3. SESSION TIMEOUT HANDLING

Use case ID: PB_SC3_ SESSION_TIMEOUT

Use case level: System level (End-to-End)

Details

Actors: Any user.

Pre-conditions: A user could be on any of the pages: Login.

Description: The validation checks will be performed for below operations.

Trigger: When user forgot to logout of the session, and currently not working on the account then its logout itself so attacker cannot steal session or any information

Post condition:

Alternative courses of action: NA

Extensions:

Exceptions: None**.**

Related use cases:

Frequency:

1) Log in: Very high – every time a user wants to log in.

2) Registration: Moderate

3) Search: Very high

Criticality: Very Critical.

Risk: Very High.

Constraints:

- **Usability:** No previous training needed.
- **Mean time to failure:** 2 failures in 4 months is acceptable.
- **Performance**: Validation Check can be done within 1 or 2 seconds.
- **Supportability:** It will be supported by IE-9 , Firefox , Chrome, Safari and Mobile Based applications on Android and iPhone.
- **Implementation**: Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Kedari Sharat

Initiation date: 09/07/2015

Date last modified: 09/14/2015

Example Scenario:

User registration

Actors: Sharat

Pre-conditions: Sharat   must Login

Description: The validation checks will be performed for below operations when Sharat forgot to logout of the session, and currently not working on the account then its logout itself so attacker cannot steal session or any information

Trigger When user forgot to logout of the session, and currently not working on the account then its logout itself so attacker cannot steal session or any information. To access that account they need to login, which can be done by Sharat only

Misuse case:

Use case ID: PB_MC3_SESSION_MISUSE

Use case level: System level (End-to-End)

Details

Actors: Any user.

Pre-conditions:

Login

Description:

1. When user forget to logout, to not allow others to steal or use user session

Post condition: Attacker can control the data

Alternative courses of action: NA

Extensions:

Exceptions: None**.**

Related use cases:

Frequency: medium-

Criticality: Very Critical –

Risk: Very High

Constraints:

- **Usability:** training needed.
- **Mean time to failure:** Expects a failure
- **Performance**: NO Validation Check is done.
- **Supportability:** It will be supported by IE-9 , Firefox , Chrome, Safari and Mobile Based applications
- **Implementation**: Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Kedari Sharat

Initiation date: 09/07/2015

Date last modified: 09/14/2015

Example Scenario:

Actors: Kedari Sharat
Pre-conditions: Login, Registration and search message

Description: The chances of violation occurs when Sharat forgets singing out from his account or some

Trigger: Kedari use his session for stealing information or privacy. Session timeout logs out of the account after a period of time if he is not working on the account

Post condition: Attacker (Kedari) can control the data base or server

4. FILE EXTENSION VERIFICATION

Use case ID: PB_SC4_FILE_EXTENSION_VERIFICATION

Use case level: System level (End-to-End)

Details

Actors: Any user.

Pre-conditions:

A logged in user on the post message pages and click upload file button:

Description: The validation checks will be performed for below operations

1. When user post message and upload a file, validation check will performed to check whether user upload a file with a valid extension.
2. If the file extension is jpg .jpeg .gif, then the upload picture will be attach on the message user will post.
3. If the file extension is not any of a jpg .jpeg .gif file, then the upload file will not uploaded and user will see upload file failed message on screen.

Post condition: The user can either choose add another upload file or post this message.

Alternative courses of action: NA

Extensions:

Exceptions: The user couldn't upload a file with an extension except from jpg .jpeg .gif

Related use cases: Post message

Frequency:

Moderate – every time a user wants to post a message and upload file.

Criticality: Very Critical.

Risk: Very High.

Constraints:

- **Usability:** No previous training needed.

- **Mean time to failure:** One user 3 failures for every 24 hours of operation is acceptable.
- **Performance**: Extension validation check can be done within 1 or 2 seconds.
- **Supportability:** It will be supported by IE-9, Firefox, Chrome, Safari and Mobile Based applications. The application will rely on Java Server Faces (JSF) and Bootstrap for the frontend. It will rely on Java and PostgreSQL database for backend.
- **Implementation**: Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Yue Wu

Initiation date: 09/07/2015

Date last modified: 09/20/2015

Example Scenario:

Upload File

Actors: Allan Shaji Manamel

Pre-conditions:

Allan must Login, post message and click upload file button.

Description: The extension validation checks will be performed for below operations

2. When Allan post message and upload a file, validation check will performed to check whether Allan upload a file with a valid extension.
3. If Allan uploaded file and its extension is not jpg .jpeg .gif, then the upload picture will not be uploaded. And Allan will receive a failure message tell him the file couldn't be uploaded.

Trigger: when Allan upload a file such as zip and exe (e.g. to dump the database contents to the attacker)". The file extension validation check doesn't allow the attacker Allan to dump invalid contents into database

Post condition: Allan will have to upload another file, post this message with no upload file or give up post message.

Use case ID: PB_MC4_MISUSE_FILE_EXTENTION

Use case level: System level (End-to-End)

Details

Actors: Any user.

Pre-conditions:

A logged in user on post message page and click upload file.

Description:

1. When user upload file, and with invalid extension.
2. The file with an invalid extension is a virus from attacker.

Post condition: Attacker can control the data base or server

Alternative courses of action: NA

Extensions: None.

Exceptions: None**.**

Related use cases: Post message

Frequency: medium- check each time when upload file

Criticality: Very critical

Risk: Very High- data base or server will be attacked

Constraints:

- **Usability:** training needed.
- **Mean time to failure:** Expects a failure
- **Performance**: NO File Extension Validation Check is done.
- **Supportability:** It will be supported by IE-9, Firefox, Chrome, Safari and Mobile Based applications. The application will rely on Java Server Faces (JSF) and Bootstrap for the frontend. It will rely on Java and PostgreSQL database for backend.
- **Implementation**: Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification history

Owner: Yue Wu

Initiation date: 09/07/2015

Date last modified: 09/20/2015

Example Scenario:

Attacker user upload a virus file

Actors: Allan Shaji Manamel

Pre-conditions:

Login, post message and click upload file.

Description: The chances of violation occurs

- When Allan upload a file, and contains virus code with an invalid file extension.
- The file with an invalid extension is a virus from attacker Allan.

Trigger: when Allan tries to upload file such as a virus file with an extension of exe (e.g. to dump the database contents to the attacker)"

Post condition: Attacker (Allan) can control the data base or server

5. DUAL LOGIN

Use Case ID: PB_SC5_DUAL_LOGIN

Use Case Level: High-level

Details:

- **Actor:** Registered User
- **Pre-conditions:** The registered user is registered on another machine at another location.
- **Description:** The will prevent the registered from being logged in at too many locations at the same time.
- **Trigger:** It will be triggered once the registered user tries to log in on another machine while he or she is already logged in on another machine.

- **Relevant requirements:** None
- **Post-conditions:** None
- **Alternative Courses of Action:** The system will email the user letting him or her know that he or she tries to log in on another machine.
- **Extensions:** None

Exceptions: None

Concurrent Uses: None

Related Use Cases: None

Decision Support

Frequency: More than 3 times in a day

Criticality: High

Risk: High

Constraints:

- **Usability:** The system should email the the owner of the account
- **Reliability:** Any failure is unacceptable
- **Performance:** The should system should email the registered in less than two seconds
- **Supportability:** The system should support email messaging
- **Implementation:** Bootstrap frontend framework and Java JSF

Modification History:

Owner: James W. Angelo

Initiation date: 10/3/2015

Date last modified: 10/3/2015

Use Case ID: PB_MC5_DUAL_LOGIN

Use Case Level: High-level

Details:

- **Actors:** Registered user
- **Pre-conditions:** The registered user is logged in on another machine at a different location
- **Description:** The registered user or someone pretending to be him tries to log on a different machine at a different location. The system denies the attempt and emails the owner of the account that he or she is trying to log in on a different machine at a different location.
- **Trigger:** The registered user or someone pretending to be him or her tries to log in on a different machine.

Post-conditions: The system denies the attempt and emails the user

Extensions:

Exceptions:

Related use cases: PB_UC_20_LOGIN, PB_SC7_INVALID_LOGIN

Criticality: Very Critical

Risk: High

Example Scenario:

Dual login

Actor: James, Rock

Pre-Condition: James knows ID and password of Rock's profile.

Description: James, who is not real owner of a profile, trying to log in from different machine, at the same time, the profile is logged in from any other location by Rock.

Trigger: The system sees log in attempt from different location, different machine. It prevent the dual log in and send email to the user.

6. ROBOT CHECK

Use Case ID: PB_SC6_ROBOT_CHECK

Use Case Level: System level (End-to-End)

Details:

Actor: User

Pre-conditions: User is viewing a post and pressed "show contact information" of post.

Description: To stop divulging user contacts by malicious users through website scrapping, robot check will be done.

Trigger: The user will click "show contact information" button.

The system responds by…

1. Checking whether user is logged in.
2. If user is logged in, a dialog with checkbox will be shown.
3. The user will check the box.
4. Upon successful check, the contact information will be shown.

Post-conditions: None.

Alternative Courses of Action:

At step 2, the user can hit “Cancel” to cancel the security check.

Extensions: None.

Exceptions: None.

Related Use Cases: view post, log in.

Decision Support

Frequency: Low.

Criticality: Low.

Risk: Low.

Constraints:

Usability

- The user needs to go through 1-2 posts to be familiar with this security feature.

Reliability

- Mean time to failure - 1 failure in verifying robot for every 24 hours of operation is acceptable.

Performance

- If the security check failed, contact information must not be shown.

Supportability

- The application will be web-based. It will be supported on all sorts of modern devices.

Implementation

- Implementation will be done in HTML 5, java and Eclipse framework.

Modification History:

Owner: Abdur Rahman Bin Shahid

Initiation date: 09/05/2015

Date last modified: 09/05/2015

Use Case ID: PB_MC6_ CONTACT_SCRAPPING

Use Case Level: System level (End-to-End)

Details:

Actor: User

Pre-conditions: User is logged-in.

Description: Misuser will scrap the posts automatically by scripting and collect contact address for misuse.

Post-conditions: None.

Alternative Courses of Action:

NA.

Extensions: None.

Exceptions: None.

Related Use Cases: Log in.

Decision Support

Frequency: Low.

Criticality: Low.

Risk: Low.

Constraints:

Usability: Training on web scrapping is needed.

Reliability:

Performance: No validation check is done.

Supportability: It will be supported by IE 9, Firefox, Chrome, Safari, iPhone and Samsung android devices.

Implementation: Client requests the backend implementation to be done in Java and the Eclipse framework. He requests us to use PostgreSQL as database provider and Java Server Faces and Bootstrap as frontend.

Modification History:

Owner: Abdur Rahman Bin Shahid

Initiation date: 09/05/2015

Date last modified: 09/05/2015

Example Scenario:

Actor: Adam

Post condition: Adam is logged in.

Description: He wants to collect contact information of the users of PantherBuddy for misuse.

Trigger:

1) He write a website scrapper for PantherBuddy.

2) He log in to the system.

3) He run the scrapper script.

4) The script will automatically collect all the links of the post.

5) Each post contain "View profile" link to the owner of post. The script will collect all the links.

6) From the each link, the script will navigate to profile and scrap the contact information.

7. INVALID LOGIN

Use Case ID: PB_SC7_INVALID_LOGIN

Use Case Level:

Details:

Actor: User

Pre-conditions: None

Description: This will prevent any invalid number of login attempts.

Trigger: If user enters his/her login credentials wrong for three times, then his/her account is blocked temporarily.

Post-conditions: User Account is blocked temporarily, for reactivation needs to change the password.

Alternative Courses of Action: None.

Extensions:

Exceptions: None.

Related Use Cases:

Decision Support

Frequency: High.

Criticality: High.

Risk: None.

Constraints:

Usability

- None.

Reliability

- Mean time to failure - 1 failure for every 24 hours of operation is acceptable

Performance

- None

Supportability

- The application will rely on HTML 5.

Implementation

- Implementation will be done in HTML 5, java and Eclipse framework.

Modification History:

Owner: Abhinav Dutt

Initiation date: 09/05/2015

Date last modified: 09/05/2015

Example Scenario:

Brute Force Login Attack Prevention

Actors: John

Pre-conditions John may or may not be registered user on the panther buddy.

Description: John has more than three invalid login attempts, john account is blocked temporarily.

Trigger: The john tries to login to panther buddy application more than three times unsuccessful attempts.

1. On the second unsuccessful attempt john is required to enter the letters mentioned in the captcha.
2. On the third unsuccessful attempt john account is blocked temporarily.

Post-conditions: John is temporarily blocked, for activation john needs to change his password.

Use Case Id: PB_MC7_BRUTE_FORCE_LOGIN

Actors: Misuser

Pre-conditions User may or may not be registered on the panther buddy.

Description: Misuser tries to log in to the system will brute force attack.

Trigger: A misuser writes a script to automate the brute force log in.

1) The script will automatically try with different combination of user id and password to log in.

2) If a match is found, misuser can log in to the system.

Post-conditions:

Alternative Courses of Action: None.
Extensions:
Exceptions: None.
Related Use Cases:

Decision Support
Frequency: High.
Criticality: High.
Risk: None.
Constraints:
Usability

- Misuser needs to know how to write script.

Reliability
Performance
Supportability

- The application will rely on HTML 5.

Implementation

- Implementation will be done in HTML 5, java and Eclipse framework.

Example Scenario:

John is a misuser. He wants to enter into the PantherBuddy system and misuse the information of the users of the system.

1) He writes a script to automate the log in attempt.

2) The script will try with different combination of user id and password.

3) If one is successful, john will be logged into the system.

9.3 Appendix C – User Interface designs.

In this section we present the UIs of the use cases that we will implement in this project.

1) Log in

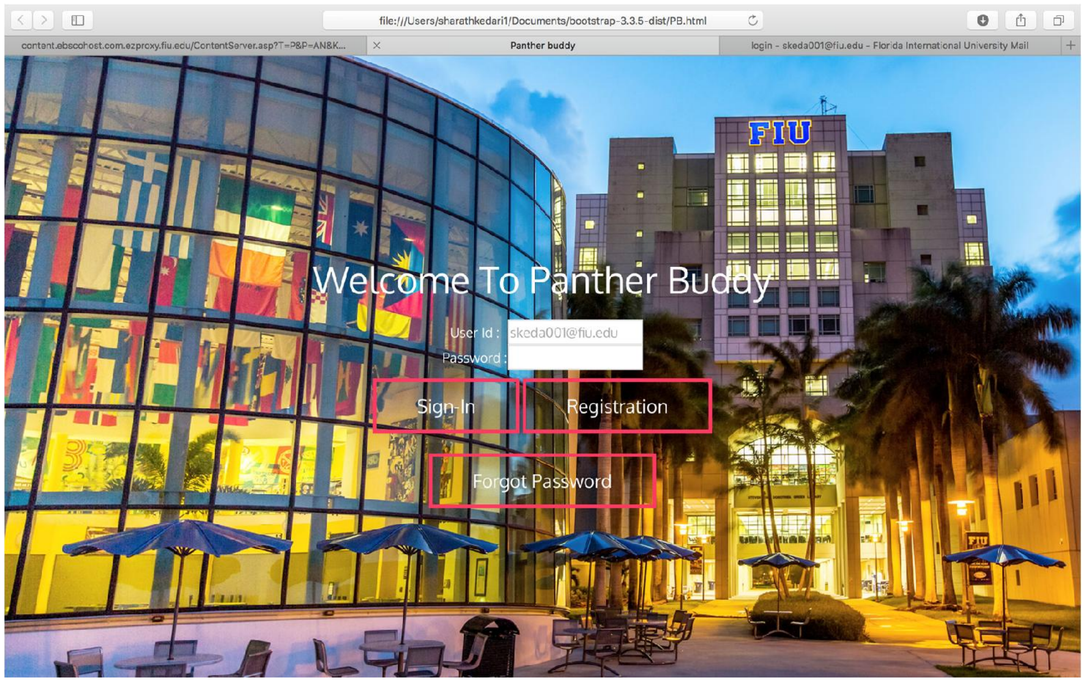


Figure 1 Log in UI

2) Password Recovery

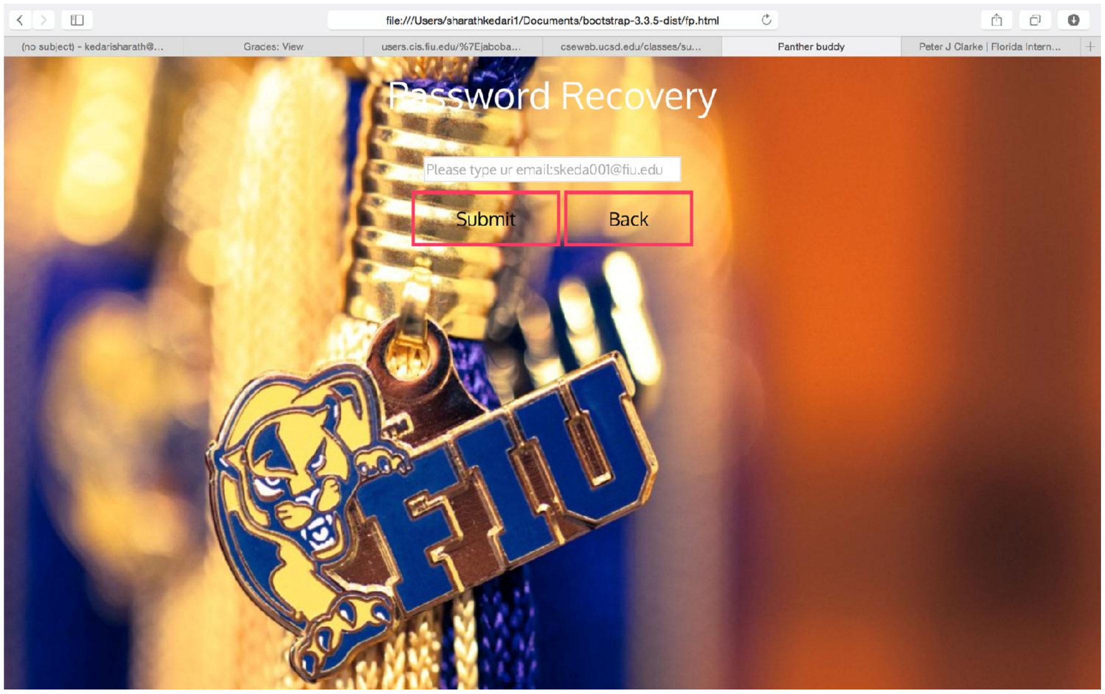
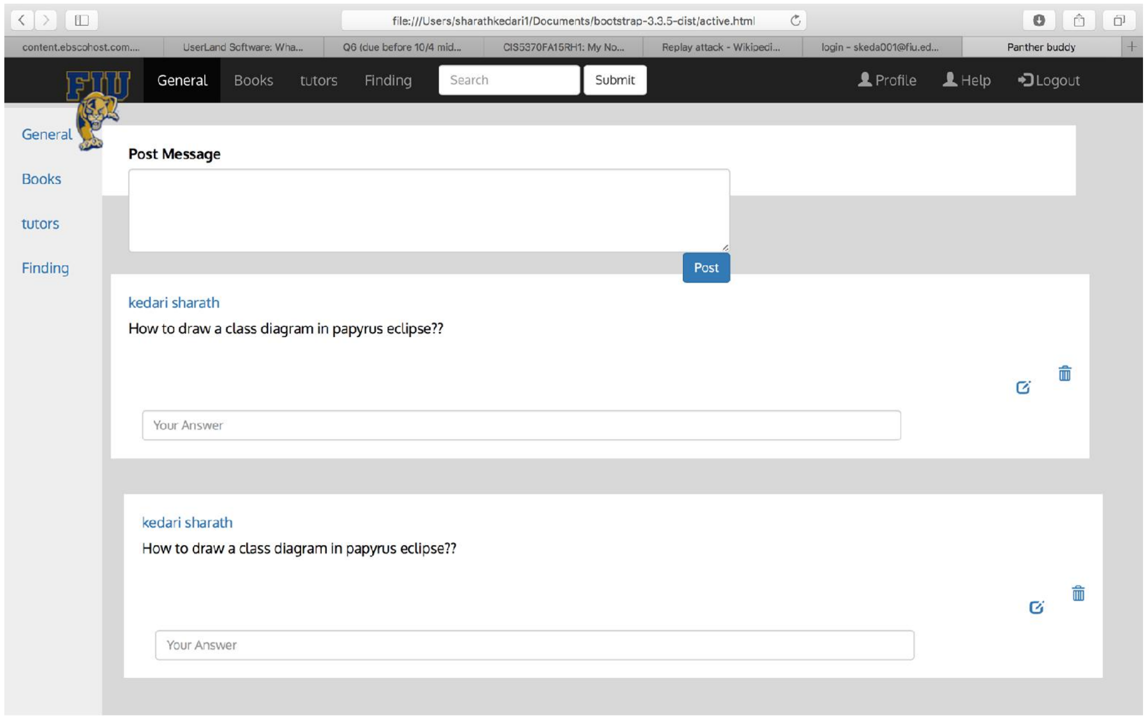

file:///Users/sharathkedari1/Documents/bootstrap-3.3.5-dist/fp.html
Password Recovery
Please type ur email:skeda001@fiu.edu
Submit
Back

Figure 2 Password Recovery UI

3) Post and view messages



kedari sharath
How to draw a class diagram in papyrus eclipse??

kedari sharath
How to draw a class diagram in papyrus eclipse??

Figure 3 Post and view message UI

4) User registration

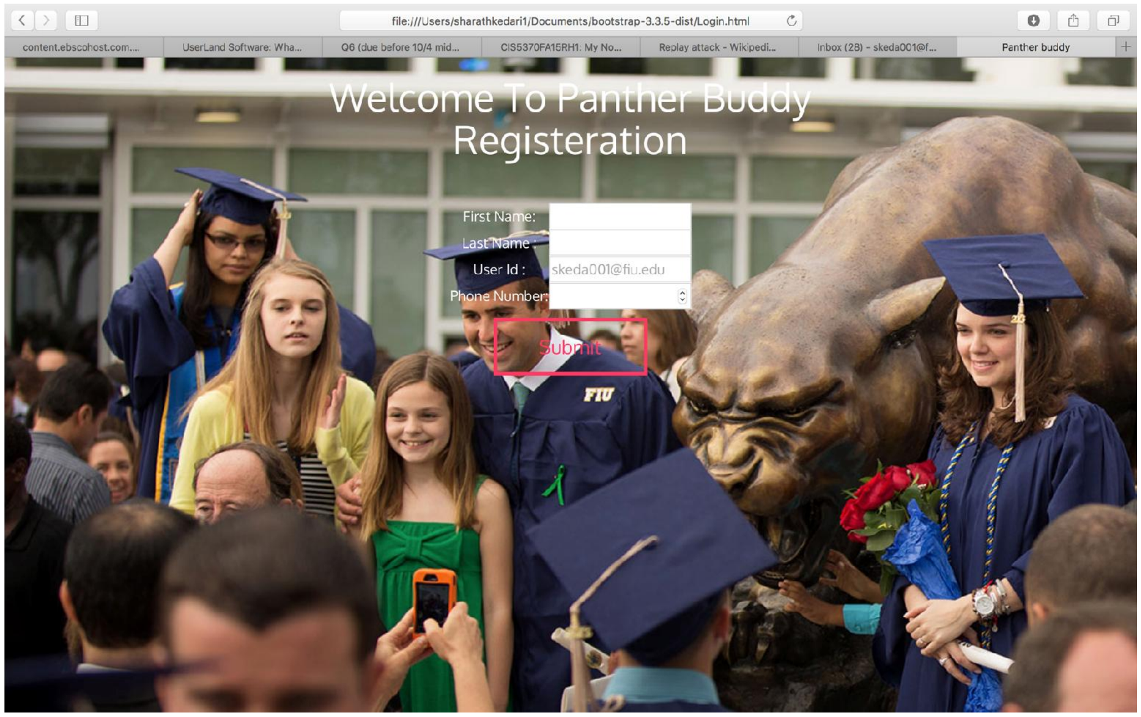
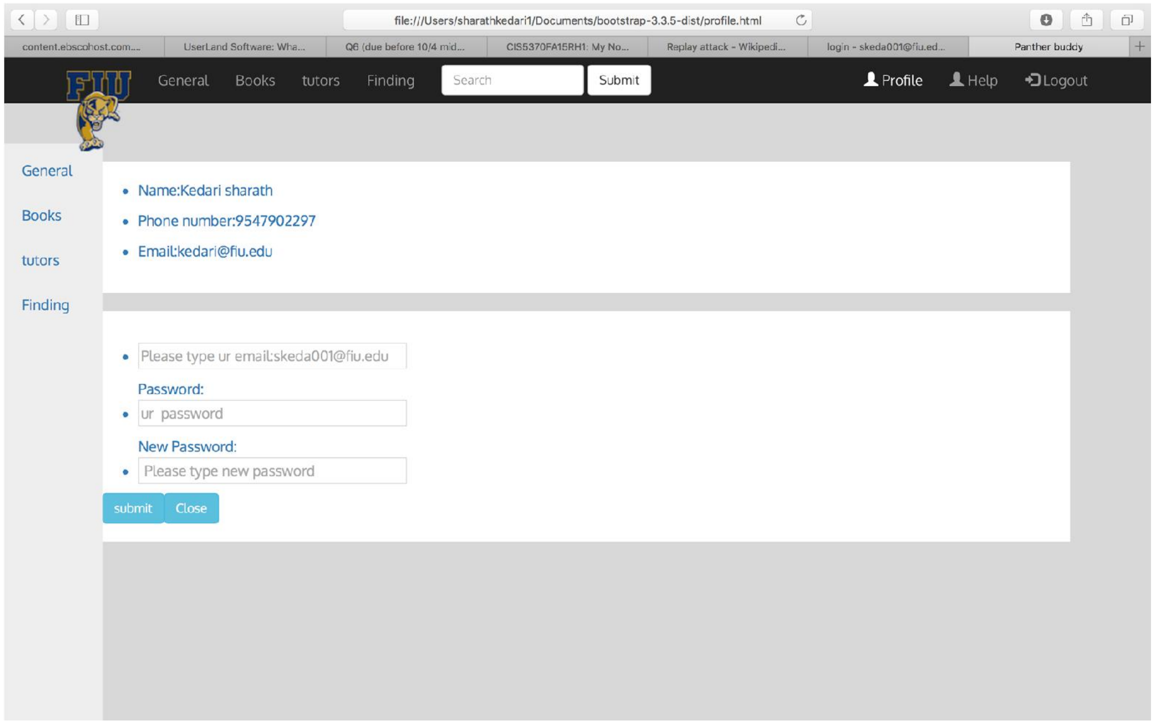


Figure 4 User Registration

5) View own profile and password change


file:///Users/sharathkedari1/Documents/bootstrap-3.3.5-dist/profile.html
General
Books
tutors
Finding
Search
Submit
Profile
Help
Logout

- Name:Kedari sharath
- Phone number:9547902297
- Email:kedari@fiu.edu

- Please type ur email:skeda001@fiu.edu

Password:
- ur password

New Password:
- Please type new password

submit Close

Figure 5 View own profile and change password

9.4 Appendix D – Diary of meeting and tasks

The meeting log is given in the below table:

Number	Description
1	Date: 08/25/2015 Time: 1 hour 30 minutes Location: Grad Lab ECS 252 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: Project Idea Summary of Discussion: Discussion on what type project we can implement.
2	Date: 08/26/2015 Time: 1 hour Location: Grad Lab ECS 252 Person attended: ALL Late: None Minutes Taker: Sharat Kedari Assigned Task: Project Idea generation Summary of Discussion: Finalizing PantherBuddy project and its functionalities.
3	Date: 08/27/2015 Time: 3 hour Location: Grad Lab ECS 266 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: Functionalities for project. Assigned 4 functionalities each Summary of Discussion: Designing about what project need to contain 21 functionalities and 7 security.
4	Date: 09/01/2015 Time: 3 hour Location: Grad Lab ECS 266 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: Functionalities for project Summary of Discussion: Assigned 3 functionalities and 1 security to each team member.
5	Date: 09/06/2015 Time: 3 hour Location: Grad Lab ECS 266 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: Functionalities for project Summary of Discussion: Assigned 3 functionalities and 1 security each.

6	Date: 09/10/2015 Time: 3 hour Location: Grad Lab ECS 258 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: UML diagrams for the user registration functionalities Summary of Discussion: UML diagrams for the user registration functionalities.
7	Date: 09/11/2015 Time: 3 hour Location: Grad Lab ECS 266 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: Functionalities for project Summary of Discussion: Assigned 3 functionalities and 1 security each
8	Date: 09/12/2015 Time: 1 hour 30 minutes Location: Grad Lab ECS 252 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: Functionalities for project Summary of Discussion: Designing about what project need to contain
9	Date: 09/13/2015 Time: 1 hour 30 minutes Location: Grad Lab ECS 252 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: Functionalities for project Summary of Discussion: Designing about what project need to contain
10	Date: 09/15/2015 Time: 1 hour 30 minutes Location: Grad Lab ECS 252 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: Functionalities for project Summary of Discussion: Continuation of the designing about what project need to contain
11	Date: 09/17/2015 Time: 1 hour 30 minutes Location: Grad Lab ECS 252 Person attended: ALL Late: none Minutes Taker: Sharat Kedari

	Assigned Task: Functionalities for project Summary of Discussion: Continuation of the functionalities design for project in Papyrus
12	Date: 09/18/2015 Time: 1 hour 30 minutes Location: Grad Lab ECS 252 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: Functionalities for project Summary of Discussion: Designing Functionalities for project in Papyrus
13	Date: 09/21/2015 Time: 1 hour 30 minutes Location: Grad Lab ECS 252 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: Functionalities for project Summary of Discussion: Designing Functionalities for project in Papyrus
14	Date: 09/23/2015 Time: 1 hour 30 minutes Location: Grad Lab ECS 252 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: Functionalities for project Summary of Discussion: Designing Functionalities for project in Papyrus
15	Date: 09/24/2015 Time: 1 hour 30 minutes Location: Grad Lab ECS 252 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: Functionalities for project Summary of Discussion: Designing Functionalities for project in Papyrus
16	Date: 09/25/2015 Time: 1 hour 30 minutes Location: Grad Lab ECS 252 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: Functionalities for project Summary of Discussion: Designing Functionalities for project in Papyrus and implementation
17	Date: 10/01/2015 Time: 2 hour 30 minutes

	Location: Grad Lab ECS 252 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: SRD writing Summary of Discussion: SRD writing.
18	Date: 10/02/2015 Time: 4 hour 30 minutes Location: Grad Lab ECS 252 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: SRD writing. Summary of Discussion: SRD writing.
19	Date: 10/03/2015 Time: 4 hour 30 minutes Location: Grad Lab ECS 252 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: SRD writing. Summary of Discussion: SRD writing.
20	Date: 10/04/2015 Time: 4 hour 30 minutes Location: Grad Lab ECS 252 Person attended: ALL Late: none Minutes Taker: Sharat Kedari Assigned Task: SRD writing. Summary of Discussion: SRD writing.