*Florida International University*

*School of Computing and Information Sciences*

CIS 4911 - Senior Capstone Project

Software Engineering Focus

Final Deliverable

Collaborative Platform

Team # 6

**Spring 2014 Fall2015** **Instructor:**

Henry M. Romero Ricardo Dominguez Masoud Sadjadi

Steven S. Sanabria

Lorenzo Sanchez **Mentor:**

Juan Caraballo

**Summer 2014**

Ramon Gomez

Jonathan Santiago

**Fall 2014**

Nicholas Madariaga

Jonathan Sanchez

**Spring 2015**

Adrian Alfonso

Jorge Travieso

**Summer 2015**

Mandiel Lastra

Michael Machin

Copyright © Florida International University 2015. All Rights Reserved. No part of the Collaborative Platform Project or documentation may be reproduced or modified without the express consent of Florida International University. Redistribution or commercial use is strictly prohibited.

***Abstract***

*The Final Deliverable of the Collaborative Platform is meant to provide the reader with all the necessary development information regarding the project. The Collaborative Platform is a website that connects students, employees, professors, and anyone else with a desire to ask questions and learn. The website provides the availability to quickly pose a question, with the system sending automated alerts, allowing for an immediate response. The task of the Collaborative Platform is to add a mentoring module in which there will be three types of mentors that can actively interact with the community of mentees. This will bridge the gap of slow and unreliable information often found online or through waiting at professor’s office hours. Within this document, detailed system design, object design and overall implementation is analyzed and described.*

**Table of Contents**

Introduction .................................................................................................................................... 5

Current System ............................................................................................................................... 5

Purpose of New System ................................................................................................................. 7

User Stories .................................................................................................................................... 9

Implemented User Stories ............................................................................................................ 10

Pending User Stories .................................................................................................................... 27

Project Plan .................................................................................................................................. 28

Hardware and Software Resources .............................................................................................. 28

Sprints Plan .................................................................................................................................. 29

Sprint 1 ......................................................................................................................................... 29

Sprint 2 ......................................................................................................................................... 33

Sprint 3 ......................................................................................................................................... 37

Sprint 4 ......................................................................................................................................... 42

Sprint 5 ......................................................................................................................................... 47

System Design ............................................................................................................................. 50

Architectural Patterns ................................................................................................................... 50

System and Subsystem Decomposition ....................................................................................... 51

Deployment Diagram ................................................................................................................... 59

Design Patterns ............................................................................................................................ 59

System Validation ........................................................................................................................ 60

Glossary …................................................................................................................................... 70

Appendix ...................................................................................................................................... 71

Appendix A - UML Diagrams ..................................................................................................... 71

Static UML Diagrams .................................................................................................................. 71

Dynamic UML Diagrams ............................................................................................................ 88

Appendix B - User Interface Design .......................................................................................... 146

Appendix C - Sprint Review Reports ........................................................................................ 153

Appendix D - Sprint Retrospective Reports .............................................................................. 157

Appendix E - User Manuals for Implemented user stories ........................................................ 159

References .................................................................................................................................. 222

# 

# 

# **Introduction**

The introductory chapter provides a brief overview of what the Collaborative Platform entails. First, the current system is explained, and the issues and limitation of the system is stated. Secondly the new system will be introduced, detailing what improvements will be made to the current system.

## Current System

The Collaborative Platform: Mentoring Module currently allows students, known as mentees, to interact with mentors who can be Personal Mentors, Project Mentors, or Domain Mentors. This interaction is through tickets which are entered into the system by students or by a live Video Conference which any user can schedule as well as invite other users. Tickets created by the different users are initially sent to tier 1 domain mentors. If the ticket is not handled within the time limit the system will automatically reassign the ticket to another tier 1 domain mentor. A tier 1 domain mentor with a pending ticket has the option to defer the ticket to a specific tier 1 or tier 2 mentor or defer the ticket to the system for automatic reassignment. The system allows Mentors to see what tickets belong to them depending on the role that they have in the system (personal, project, domain). New mentors can self-register and apply for the desired mentorship, process that may reduce the amount of data entry performed by the administrator of the system. Video Conference can be scheduled by any user, and it is only that user who can start the conference. On this page users can share their screen as well as a whiteboard where they can draw while other users see it.

The system allows for an integration of three modules into one single collaborative platform. The system allows users to register and log into the site and pick their main functionality. In some cases users can have various roles and thus will have multiple views available within the site. Each user has a profile page and depending on their role, they have a more detailed interaction for their account(s). For example, within the mentoring module there are three mentors: domain, project and personal. All types of mentors are allowed to sync their profile with LinkedIn in order to upload a picture and provide a short bio, as well as any other related information.

The project mentor can see all of the projects they are mentoring and will have the ability to screen share, use a whiteboard, and other functionality that will be taken from the other two modules and be integrated within the mentoring module. They also have the ability to grab the project information directly from the senior project website. The domain mentors are mentors of specific skills. Currently domain mentors are not able to suggest a new domain. They have the functionality to see all tickets assigned, whether answered, dropped, or pending. From here, they can answer the questions, append comments, and in some cases they can delegate it to all other tier 1 mentors of that domain. They can also communicate within the internal mail service to other mentors.

The system allows self-registration for users and provides self-serve mentorship applications. These applications provide options for manually selecting preferred choices and the option to defer selection to the system. Submitted applications will be vetted by an admin. The admin is able to approve/deny what he chooses and also send counter-offers back to the mentor.

The system allows the reporting of mentees, mentors and tickets. Those reports are available to the administrator in order to gather as much insight of the data as possible. Additionally the system provides the administrator with a way of making a quantitative analysis of the utilization of the system via a dashboard. The utilization dashboard includes the following reports: Amount of tickets created, Amount of tickets closed, Average duration of a ticket, Average time a mentor takes to answer a ticket, Amount of tickets currently open, Amount of tickets that are currently closed. Also, as part of the analysis, the system is able to analyze and pull the most frequent type of questions that mentees may ask.

The system matches the tickets created by mentees to the most appropriate domain mentors – those with the most expertise in that specific skill, and which mentor of the appropriate skill is next in the order of round robin. The system will send an email notification to the assigned mentor that will include: the subject, the description, the time limit the mentor has before reassignment, a link that will take the mentor to view the ticket, and a link that will automatically assign the ticket to a new mentor. If the system notices the inactivity on one of the ticket and notices that the last comment added to the ticket was not made by the assigned mentor, it proceeds to rank it to another mentor in the same tier. If this situation happens a set number of consecutives times, set by the system administrator, for the same ticket, the system will have automatically notified the administrator for immediate resolution. The system administrator will also be able to set the time frames that will be used to detect the lack of activity. Besides being able to append comments to tickets, the mentees will be able to attach files. Once these tickets have been answered, the mentees will be allowed to close the ticket or reject the answer provided.

The system is able to receive emails, currently for the purpose of detecting out of office emails. An email server will be constructed on the server hosting the platform, and the system will be able to read these emails and detect out of office replies. This will ensure that tickets are not assigned to mentors who are unavailable, as well as reassign a set number of tickets to new mentors that were assigned to the mentor who is out of office. The selection criteria for the reassigned tickets will be set by the system administrator.

In order to facilitate the interaction mentor-mentee, the system supports video conferences where the mentor and mentee can meet face to face and collaborate with tools such as screen sharing, whiteboard and chat without synchronization issues. Those video conferences can be scheduled from tickets or from the video conference page. Video conferences for the current day will be displayed on user’s homepage so that they can be easily and quickly accessed.

The system provides users with a second screen-share option when in a videoconference. This will allow attendees to share their screen in one section and on the other share for example the notes being taken during the conference or a Mingle board, etc. The system will allow users to edit a video conference in case the user needs to reschedule the conference, edit the notes, or invite more users. The system will also provide users with a clean and user friendly environment and at the same time as bug-free as possible.

The main drawbacks of the current System is that there is no way for the users to provide feedback to the administrators to create a more complete user experience. This is desired because it will make it easier to head in the right direction during future iterations as the administrator will know, from a user perspective, what the system is lacking. At the same time, there needs to be a way to handle this feedback and set up a reply system where administrators can ask for more details or give their comments on what the users say.

The current system also lacks a suite of unit and functional tests that lets future developers know where the system is lacking and in which areas to improvement and refactoring is needed. Unit tests would provide a good way of making sure that the models are coded correctly and that they have the correct functions. They would also make sure that the system is coded with the correct dependencies as this would make future coding a lot easier and would prove to be guidelines in the future. The functionality testing will work in the same light and will provide insight more towards the frontend rather than the backend.

## Purpose of New System

The system will record user feedback and store it for later retrieval. Users who make feedback will be able to see their own feedback and will be able to view other feedback but will not be able to do anything beyond that to the feedback of other users. Administrators will be able to manage the feedback. Another feature implemented was the feedback replies. The user will be able to reply to his/her own feedback but will not be able to reply to the feedback of others. The system will allow the user to also update their original feedback and edit something they may have left out. The administrator may reply to any feedback provided.

The system now has a set of basic unit tests. They cover all of the website major features and will be held in the test directories. The system also has several suites of functional tests that cover the frontend views and automates user interaction with the application. The reasoning for these tests is to aid future developers when refactoring code or making new systems. The tests are easy to edit and segmented enough to avoid clutter and will serve as a future resource when the code needs to be refactored.

**User Stories**

In this section we will talk about the user stories requested by our product owner as well as the ones defined by us throughout this semester. We will also note the user stories that were implemented during the semester and the ones that are still pending (if any).

User stories:

1. Prepare production Database
2. Clean up Production Database and Test Features
3. Set up Production Website
4. Set up email server for production site
5. Prepare and Test Video Conferencing
6. Set up PHPUnit using Yii Framework
7. Begin Feedback Feature
8. Yii / PHPUnit Testing
9. Finish Feedback Functionality
10. Finish Feedback Replies Functionality
11. Feedback Documentation
12. Feedback/Feedback Replies Interface
13. User Unit Testing
14. Ticket Unit Testing
15. UI Changes, Fixes
16. Finish Documentation on Feedback
17. Mentor Unit Testing
18. Feedback/Replies Unit Tests
19. Feedback Search Function
20. Portal Login
21. Write Mentor Functionality Tests
22. Report/Message Functionality Tests
23. Write assorted unit tests
24. Write Video Conference unit Tests
25. Write Message unit tests
26. Write Administrator tests
27. Write application functionality Tests
28. Write email functionality Tests

## Implemented User Stories

**User Story #1 - Prepare production Database**

### Description:

* The database for the production site needs to be prepared. It needs to be cleared of all extraneous tickets/mentors/users.

### Acceptance Criteria:

* Unnecessary tables are dropped
* Unnecessary rows for necessary tables are deleted
* The site is fully functional with the wiped database schema

**User Story #2 - Clean up Production Database and Test Features**

### Description:

* Clean up Database for CPv7 and make sure everything is reset to production. Make sure that all features are usable as they are on cp-dev.

**Acceptance Criteria**:

* Clean up Production Database but keep important dependencies with accounts that need to be kept.
* Test out features as they are on cp-dev
* Find cause of any bugs and fix

**User Story #3 – Set up Production Site**

### Description:

* The production site was not up to date.. it seemed to be running version 3. Here we will update the production site and make sure it fits the criteria.

**Acceptance Criteria**:

* Move files from CP-DEV onto CP and re-configure Github
* Make sure CPv6 is displayed when typing in cp.cis.fiu.edu

**User Story #4 – Set up email server for production site**

### Description:

* CP-DEV has an email server that notifies the user and parses through emails in order to perform tasks. These tasks will have to function in cp.cis.fiu.edu

**Acceptance Criteria**:

* Email server set up at cp.cis.fiu.edu
* Emails are sent to registered / registering users
* fiucoplat@cp.cs.fiu.edu can parse through emails for away mentors.

**User Story #5 – Prepare and Test Video Conferencing**

### Description:

* In order to have a video conference the accounts need to be made. This will help test the register feature as well as the meeting feature of the production site.

**Acceptance Criteria**:

* Get accounts registered.
* Set up video conference with 3+ people and verify it works.
* Download chrome plugin that allow screen sharing.

**User Story #6 – Begin Feedback Feature**

### Description:

* There was discussion between Juan and myself of a feedback feature that should be added to the website that will allow users to comment on the website and allow moderators and administrators to see the comment.

**Acceptance Criteria**:

* Feedback can be posted/seen
* Administrators are the only ones who can see it (for now)
* Feedback can be replied to by the administrator

**User Story #7– Set up PHPUnit and Yii Framework**

### Description:

* Understand how to do tests for Yii framework. Which kind of tests are most important to run and how to go about it.

**Acceptance Criteria**:

* Read through Yii Unit testing documentation
* Do mock unit testing on Feedback Feature
* Branch out to other classes and begin writing mock unit tests for those

**User Story #8 – Finish Feedback Functionality**

### Description:

* Finishing the feedback feature and make sure it works as intended.

**Acceptance Criteria**:

* Can post feedback
* Admin pages and user pages set up to display different things
* None of the links are broken

**User Story #9 – Finish Feedback Replies Functionality**

### Description:

* Make a reply system to feedback that shows replies to certain feedback.

**Acceptance Criteria**:

* Only admin can reply and user can add to his own comment
* Replies can be viewed by anyone but not edited

**User Story #10 – Feedback Documentation**

### Description:

* Complete documentation for feedback

**Acceptance Criteria**:

* Complete use cases
* Complete UML diagrams
* Complete testing

**User Story #11 – Feedback/Feedback Replies Interface**

### Description:

* Make minimalist interface that is easy to go through for Feedback/Feedback Replies

**Acceptance Criteria**:

* Use same table styles when displaying results
* Make sure all links are in the correct place and lead to the correct page
* Have admin and user access the correct pages

**User Story #12 – User Unit Testing**

### Description:

* Complete User Function Unit Testing Suite

**Acceptance Criteria**:

* Use cases researched
* Unit Tests written
* All tests run successfully

**User Story #13 – Ticket Unit Testing**

### Description:

* Write a test suite for the Ticketing function

**Acceptance Criteria**:

* Research Use Cases that involve ticketing
* Write Unit tests for use cases
* Make sure all unit tests pass for use cases

**User Story #14 – UI changes and fixes**

### Description:

* Fixing various things on the UI home page and some bugs through the unit tests.

**Acceptance Criteria**:

* UI Changes that were made to the front page
* Certain bug fixes through the unit tests

**User Story #15 – Finish Documentation on Feedback**

### Description:

* Finish feedback/replies documentations

**Acceptance Criteria**:

* Finish commenting
* Finish diagrams

**User Story #16 – Mentor Unit Testing**

### Description:

* Write the test suite that concentrates on unit testing Mentor functionalities.

**Acceptance Criteria**:

* Test Domain, Project, and Personal Mentor Applications
* Test Administrative perspective on accepting and rejecting mentor applications
* Test other mentor-related functionality

**User Story #17 – Feedback/Replies Unit Tests**

### Description:

* Write the Feedback and Feedback Replies test suites

**Acceptance Criteria**:

* Write unit tests that deal with the feedback feature
* Write unit tests that deal with the feedback replies feature
* Write unit tests that deal between the interaction of feedback and feedback replies.

**User Story #18 – Feedback Search Function**

### Description:

* Create a search function for Feedback that allows a user to search for listed feedback.

**Acceptance Criteria**:

* All fields within Feedback must be searchable
* All search fields have to be configured to the different tables used by Feedback
* Must return values that the user understands.

**User Story #19 – Portal Login**

### Description:

* Add portal login functions and actions that allows SPW to login to CPv7 securely.

**Acceptance Criteria**:

* Action must be implemented
* SPW must be able to communicate with it in some way
* SPW must be able to log in users with the same username and password into CPv7

**User Story #20 – Write Mentor Functionality Tests**

### Description:

* Write Selenium tests that automate user interaction with mentor functions

**Acceptance Criteria**:

* Test mentor application for the three types of mentor that the system accepts
* Test administrative functionalities in regards to mentoring
* Test user interaction with mentors as mentees

**User Story #21 – Write User Functionality Tests**

### Description:

* Write a series of User functionality tests that automate user interaction with the system.

**Acceptance Criteria**:

* Test that users can successfully login and log out, as well as change anything in their profile.
* Test that a user can register to the system using the register function the system provides.
* Test other miscellaneous functionalities that a user can perform.

**User Story #22 – Report/Message Functionality Tests**

### Description:

* Write functional tests for report and message functionalities

**Acceptance Criteria**:

* Make sure all reports can be accessed by admin
* Make sure messages can be sent between users
* Make sure messages can be managed by users

**User Story #23 – Write assorted unit tests**

### Description:

* Write a series of tests that aren’t covered by other functionalities

**Acceptance Criteria**:

* Write tests for user invitation by the administrator
* Write tests for the meetings functionality
* Write tests for retrieving project information

**User Story #24 – Write Video Conference Unit Tests**

### Description:

* Write the unit tests that deal with the Video Conference feature and anything extended by it.

**Acceptance Criteria**:

* Write tests for setting up future and present conferences
* Write tests that allow users to edit current conferences that are scheduled
* Write tests to allow the admin to manage existing conferences

**User Story #25 – Write Message Unit Tests**

### Description:

* Write unit tests for the messaging system that was implemented to Coplat in past points in time.

**Acceptance Criteria**:

* Write tests regarding message creation
* Write tests to ensure that messages can be passed between users
* Write tests to make sure that messages can be managed by the user after sent

**User Story #26 – Write Administrator Tests**

### Description:

* Write simple test suite for the administrator model in the coplat project

**Acceptance Criteria**:

* Write tests to create new administrator
* Write tests do manage and remove current system administrators

**User Story #27 – Write application functionality tests**

### Description:

* Write a test suite that will deal with functional tests that are not covered by other functions in the system

**Acceptance Criteria**:

* Write test that shows us the FIU Sign in feature
* Write test that manually syncs Coplat with SPW

**User Story #28 – Write email functionality tests**

### Description:

* Write a series of tests that make sure that email notifications are working as intended.

**Acceptance Criteria**:

* Test comment and invite notification
* Test Meeting email notification
* Test Ticket reassign and close notification

## Pending User Stories

**User Story #29 - Only allow to disable users.**

**Description**: As an administrator I want to only allow to disable users, so that I can have the information there all the time.

**User Story #30 - Senior Project website integration (Manual).**

**Description**: As an administrator I want to be able to schedule a system synchronization with the Senior Project Website so that the projects information on the CP is automatically created and/or created

**User Story #31 - Provide user with a tool to take notes.**

**Description**: As a user I want to have a tool that will allow me to take notes while on a video conference so that I don’t have to navigate away from the page the conference is being held on or use another application and then be able to save the notes to my computer.

**User Story #32 - BUG: System Admin should be able to delete tickets.**

**Description**: As a system administrator I would like to be able to delete tickets that are old, insignificant, or unused. So that the database is not overfilled with tickets that provide no value to any user. Currently the administrator is given the option to delete but an error is found, the system needs to delete everything related to the ticket on the database before deletion.

**User Story #33 - Senior Project website synchronization (Automatic)**

**Description**: As an administrator I want to be able to schedule Cron jobs to synchronize the CP with the Senior Project website so that the projects information on the CP is automatically created and/or created.

**User Story #34 – Archiving System**

**Description**: As an administrator I should be able to archive everything on the application or later use. This includes Tickets, Mentor Applications, Messages, etc, even if they appear to have been deleted. The system should keep a permanent record of every written data that has passed through the system in some shape or another.

**User Story #35 – Finish Unit/Functionality Tests**

**Description:** While a lot of things are covered by the current unit and functionality tests. Some error cases have not been written. For example, a case for what if a person fills a ticket like they’re not supposed to has not been written as in the unit/functionality test that was written the system would be assumed to fail.

# **Project Plan**

This section describes the project plan which includes the project organization, in the software development process. The following sections include the hardware and software resources and also the identification of task, milestone and user stories implemented in each sprint.

Hardware and Software Resources

Provide a comprehensive list of the hardware and software you will use to develop the system. For each software resource (OS, programming language, framework, development tool, etc) briefly specify the reason for your selection.

**Hardware Resources**

* Personal computer

**Software Resources**

* Apache v.2.4.10
* MySQL v5.5.42
* PHP v5.6.7
* phpMyAdmin
* phpMyStorm
* Yii Framework
* Linux
* Google Chrome
* StarUML
* PHPUnit5.1
* Selenium IDE
* SourceTree
* MySQL Workbench

The main components of the system are the Apache and MySQL environment hosted on a Linux operating system. The Yii framework environment is using apache to execute, and contains our various artifacts used in development (Model, View, and Controller). The models are mapped to tables in the MySQL environment setup on the same machine. The browser on the client’s machine communicates with the server using HTTP.

We decided to use phpMyAdmin to have an interface with the database and better understand the structure of the different tables. We chose phpMyStorm as our IDE for editing the code as it provides autocompletion for code allowing us to develop our application faster. Google Chrome was used as our prefered browser as it is the most compatible with the RTCMulticonnection library used for the video conference module. We used StarUML in order to quickly create static and dynamic diagrams to represent functionalities. SourceTree was used to provide us with an interface to github and update the repository. We used MySQL Workbench to run queries on both the development and the production server because it made it more convenient as it stored the connections to the servers. PHPUnit was used to write the test suites for the unit tests that were coded. Selenium IDE was used to write the test suites for the functionality tests that were coded.

## Sprints Plan

### Sprint 1

**User Story #1 - Prepare production Database**

### Description:

* The database for the production site needs to be prepared. It needs to be cleared of all extraneous tickets/mentors/users.

### Acceptance Criteria:

* Unnecessary tables are dropped
* Unnecessary rows for necessary tables are deleted
* The site is fully functional with the wiped database schema

**User Story #2 - Clean up Production Database and Test Features**

### Description:

* Clean up Database for CPv7 and make sure everything is reset to production. Make sure that all features are usable as they are on cp-dev.

**Acceptance Criteria**:

* Clean up Production Database but keep important dependencies with accounts that need to be kept.
* Test out features as they are on cp-dev
* Find cause of any bugs and fix

### Sprint 2

**User Story #3 – Set up Production Site**

### Description:

* The production site was not up to date.. it seemed to be running version 3. Here we will update the production site and make sure it fits the criteria.

**Acceptance Criteria**:

* Move files from CP-DEV onto CP and re-configure Github
* Make sure CPv6 is displayed when typing in cp.cis.fiu.edu

**User Story #4 – Set up email server for production site**

### Description:

* CP-DEV has an email server that notifies the user and parses through emails in order to perform tasks. These tasks will have to function in cp.cis.fiu.edu

**Acceptance Criteria**:

* Email server set up at cp.cis.fiu.edu
* Emails are sent to registered / registering users
* fiucoplat@cp.cs.fiu.edu can parse through emails for away mentors.

**User Story #5 – Prepare and Test Video Conferencing**

### Description:

* In order to have a video conference the accounts need to be made. This will help test the register feature as well as the meeting feature of the production site.

**Acceptance Criteria**:

* Get accounts registered.
* Set up video conference with 3+ people and verify it works.
* Download chrome plugin that allow screen sharing.

### Sprint 3

**User Story #6 – Begin Feedback Feature**

### Description:

* There was discussion between Juan and myself of a feedback feature that should be added to the website that will allow users to comment on the website and allow moderators and administrators to see the comment.

**Acceptance Criteria**:

* Feedback can be posted/seen
* Administrators are the only ones who can see it (for now)
* Feedback can be replied to by the administrator

**User Story #7– Set up PHPUnit and Yii Framework**

### Description:

* Understand how to do tests for Yii framework. Which kind of tests are most important to run and how to go about it.

**Acceptance Criteria**:

* Read through Yii Unit testing documentation
* Do mock unit testing on Feedback Feature
* Branch out to other classes and begin writing mock unit tests for those

**User Story #8 – Finish Feedback Functionality**

### Description:

* Finishing the feedback feature and make sure it works as intended.

**Acceptance Criteria**:

* Can post feedback
* Admin pages and user pages set up to display different things
* None of the links are broken

**Sprint 4**

**User Story #9 – Finish Feedback Replies Functionality**

### Description:

* Make a reply system to feedback that shows replies to certain feedback.

**Acceptance Criteria**:

* Only admin can reply and user can add to his own comment
* Replies can be viewed by anyone but not edited

**User Story #10 – Feedback Documentation**

### Description:

* Complete documentation for feedback

**Acceptance Criteria**:

* Complete use cases
* Complete UML diagrams
* Complete testing

**User Story #11 – Feedback/Feedback Replies Interface**

### Description:

* Make minimalist interface that is easy to go through for Feedback/Feedback Replies

**Acceptance Criteria**:

* Use same table styles when displaying results
* Make sure all links are in the correct place and lead to the correct page
* Have admin and user access the correct pages

**User Story #12 – User Unit Testing**

### Description:

* Complete User Function Unit Testing Suite

**Acceptance Criteria**:

* Use cases researched
* Unit Tests written
* All tests run successfully

### 

### Sprint 5

**User Story #13 – Ticket Unit Testing**

### Description:

* Write a test suite for the Ticketing function

**Acceptance Criteria**:

* Research Use Cases that involve ticketing
* Write Unit tests for use cases
* Make sure all unit tests pass for use cases

**User Story #14 – UI changes and fixes**

### Description:

* Fixing various things on the UI home page and some bugs through the unit tests.

**Acceptance Criteria**:

* UI Changes that were made to the front page
* Certain bug fixes through the unit tests

**User Story #15 – Finish Documentation on Feedback**

### Description:

* Finish feedback/replies documentations

**Acceptance Criteria**:

* Finish commenting
* Finish diagrams

**User Story #16 – Mentor Unit Testing**

### Description:

* Write the test suite that concentrates on unit testing Mentor functionalities.

**Acceptance Criteria**:

* Test Domain, Project, and Personal Mentor Applications
* Test Administrative perspective on accepting and rejecting mentor applications
* Test other mentor-related functionality

**User Story #17 – Feedback/Replies Unit Tests**

### Description:

* Write the Feedback and Feedback Replies test suites

**Acceptance Criteria**:

* Write unit tests that deal with the feedback feature
* Write unit tests that deal with the feedback replies feature
* Write unit tests that deal between the interaction of feedback and feedback replies.

**User Story #18 – Feedback Search Function**

### Description:

* Create a search function for Feedback that allows a user to search for listed feedback.

**Acceptance Criteria**:

* All fields within Feedback must be searchable
* All search fields have to be configured to the different tables used by Feedback
* Must return values that the user understands.

### Sprint 6

**User Story #19 – Portal Login**

### Description:

* Add portal login functions and actions that allows SPW to login to CPv7 securely.

**Acceptance Criteria**:

* Action must be implemented
* SPW must be able to communicate with it in some way
* SPW must be able to log in users with the same username and password into CPv7

**User Story #20 – Write Mentor Functionality Tests**

### Description:

* Write Selenium tests that automate user interaction with mentor functions

**Acceptance Criteria**:

* Test mentor application for the three types of mentor that the system accepts
* Test administrative functionalities in regards to mentoring
* Test user interaction with mentors as mentees

**User Story #21 – Write User Functionality Tests**

### Description:

* Write a series of User functionality tests that automate user interaction with the system.

**Acceptance Criteria**:

* Test that users can successfully login and log out, as well as change anything in their profile.
* Test that a user can register to the system using the register function the system provides.
* Test other miscellaneous functionalities that a user can perform.

**User Story #22 – Report/Message Functionality Tests**

### Description:

* Write functional tests for report and message functionalities

**Acceptance Criteria**:

* Make sure all reports can be accessed by admin
* Make sure messages can be sent between users
* Make sure messages can be managed by users

**User Story #23 – Write assorted unit tests**

### Description:

* Write a series of tests that aren’t covered by other functionalities

**Acceptance Criteria**:

* Write tests for user invitation by the administrator
* Write tests for the meetings functionality
* Write tests for retrieving project information

### Sprint 7

**User Story #24 – Write Video Conference Unit Tests**

### Description:

* Write the unit tests that deal with the Video Conference feature and anything extended by it.

**Acceptance Criteria**:

* Write tests for setting up future and present conferences
* Write tests that allow users to edit current conferences that are scheduled
* Write tests to allow the admin to manage existing conferences

**User Story #25 – Write Message Unit Tests**

### Description:

* Write unit tests for the messaging system that was implemented to Coplat in past points in time.

**Acceptance Criteria**:

* Write tests regarding message creation
* Write tests to ensure that messages can be passed between users
* Write tests to make sure that messages can be managed by the user after sent

**User Story #26 – Write Administrator Tests**

### Description:

* Write simple test suite for the administrator model in the coplat project

**Acceptance Criteria**:

* Write tests to create new administrator
* Write tests do manage and remove current system administrators

**User Story #27 – Write application functionality tests**

### Description:

* Write a test suite that will deal with functional tests that are not covered by other functions in the system

**Acceptance Criteria**:

* Write test that shows us the FIU Sign in feature
* Write test that manually syncs Coplat with SPW

**User Story #28 – Write email functionality tests**

### Description:

* Write a series of tests that make sure that email notifications are working as intended.

**Acceptance Criteria**:

* Test comment and invite notification
* Test Meeting email notification
* Test Ticket reassign and close notification

# **System Design**

This section will describe in detail our system architectures for the Collaborative Platform giving an overview of the primary and the secondary architecture patterns. In addition, it will provide a detailed subsystem decomposition description for each of the major subsystems. Also, the hardware and software mapping are defined, with data structures that the system will need to perform the operations required by the client. Finally, some aspects of security and privacy are covered as they are needed for keeping the integrity of the data stored in the database.

## Architectural Patterns

As a result of our understanding of the system, we determined to use the Model View Controller (MVC) architecture, because this architecture will benefit the team at the time of implementation, because we can effectively divide the work, so any change in one of the classes does not need a large alteration, if any, to other classes.

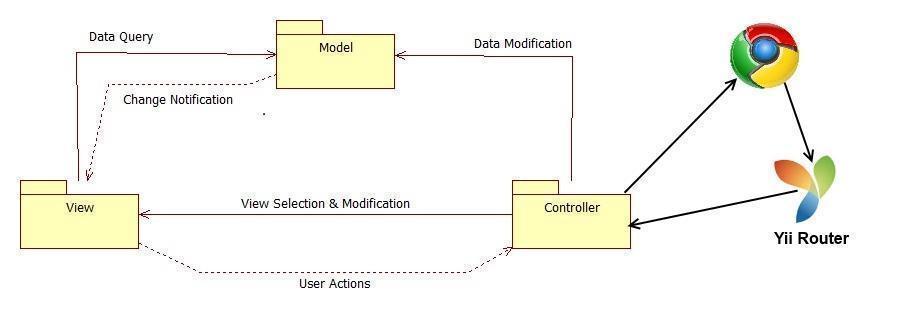


Figure 1.2 Model View Controller

## System Design Overview

As a result of our understanding of the system, we determined that the system would be divided in three major components where the objects and classes have to be independent. By this, the team chose the three tier architecture model; this is because the system architecture allows the system to be divided into three independent layers, as shown in figure 2.1.a

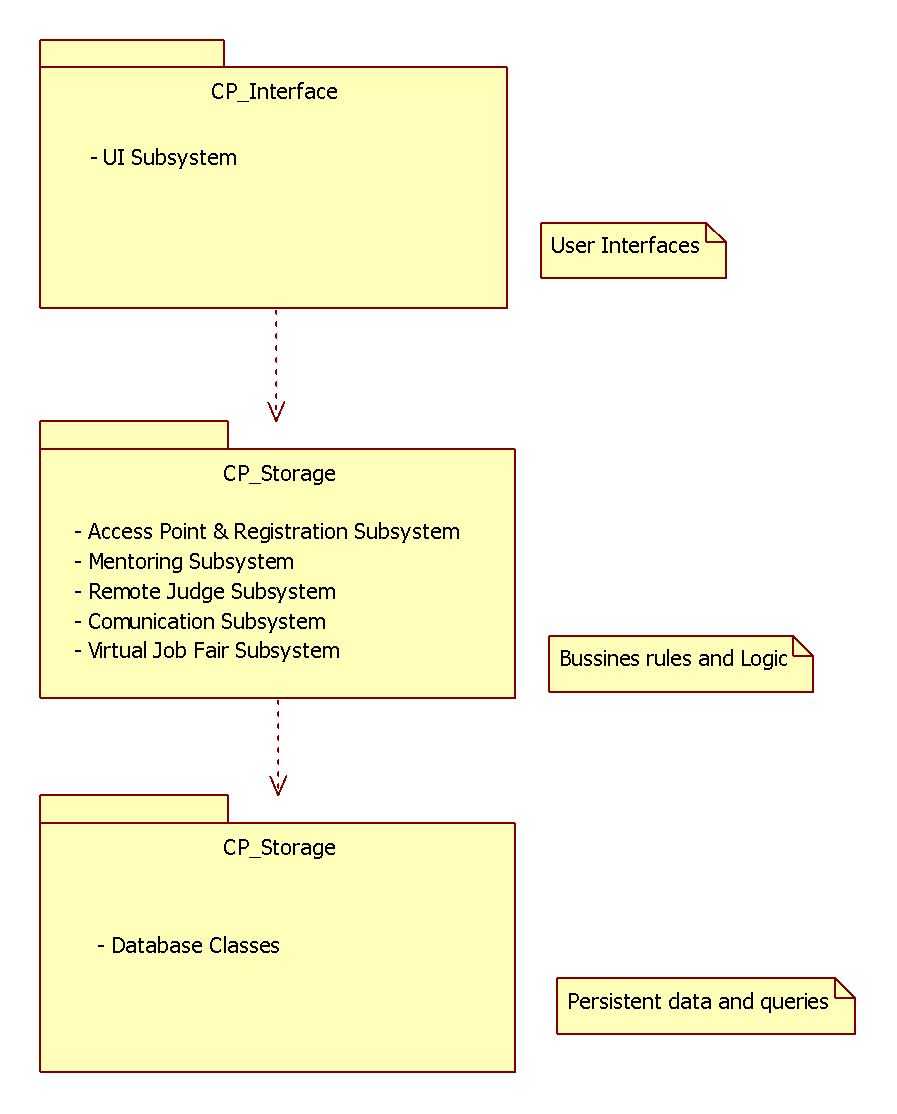


Figure 1.0 Three Tier

In the figure the interface includes all user interfaces that serves as boundaries with the users. The application Logic will serve as an intermediary between the Interface and the Storage. Moreover, this layer ensures that all processing is performed correctly. The Storage contains all the classes needed to perform the queries of the system objects.

System and Subsystem Decomposition

Below we present a description of each of the major subsystems and their respective functional requirements.

**Registration and Access Point Subsystem**

The registration and access point subsystem will be the main entry point for our platform. This will facilitate the process across the user database. Since there will be one login for each individual module, the access point is simplified into one subsystem. This subsystem will also cover the creation of new users and the retrieval of passwords for current users.

* Allow the user to log in.
* Allow the user to log out.
* Allow the user to retrieve forgotten password.
* Allow the admin to register mentors (project/domain/personal)
* Allow admin to edit mentors roles
* Allow mentors to login with credentials provided by admin

**Mentoring Subsystem**

The mentoring subsystem will cover the essentials of the mentoring module. Specifics to the mentoring module are separated into this subsystem. These functionalities include the essential workflow of the ticketing system for issues created by mentees for the different types of mentors.

* Auto assigns the most appropriate domain mentor to a ticket.
* Allow administrator to reassign the ticket to other domain mentor.
* Allow project mentor to retrieve the description of the projects assigned.
* Allow project mentor to retrieve all the mentees for each project.
* Allow project mentor to set up meetings with his/her mentees.
* Allow project mentor to retrieve all the upcoming meetings.
* Allow project mentor to assign ticket to other project mentor.
* Allow project mentor to see the ticket created by their mentees.
* Allow mentees to assign tickets to his/her project mentor.
* Allow user create a ticket.
* Allow user to select a specific domain for a new ticket.
* Allow user to select a specific sub domain.
* Allow user to upload file to a ticket.
* Allow user to download file from a ticket.
* Allow user to append comments to a ticket.
* Allow user to retrieve all the details of the ticket of the ticket that he/she created or was assigned.
* Allow user to close a ticket.
* Allow user to reject a ticket.
* Allow System Administrators (SA) to create new administrators.
* Allow SA to change user profile information and scope.
* Allow SA to manage Domains and Sub-Domains.
* Allow SA to disable and enable users.
* Allow SA to send invitations
* Allow SA to manage projects
* Allow users to edit their availability
* Allow users to edit their photo
* Allow project mentor to select desired projects
* Allow personal mentor to select desired personal mentees
* Allow domain mentors to add domain and expertise ratings
* Allow administrators to view and edit user profiles
* Allow mentees to assign tickets to his/her personal mentor.
* Allow personal mentor to see the ticket created by their mentees.
* Allow personal mentor to comment the ticket created by their mentees.
* Allow Domain mentor in tier 1 to escalate the ticket to a Domain Mentor in tier 2
* Allow user to select a specific priority for a new ticket.
* Allow administrator to change the priority waiting time.
* Allow the system to automatically reassign the tickets.

**Communication and Notification Subsystems**

The communication subsystem is a key subsystem that provides functionality derived from the mentoring module. This communication will be vital to the ticket system as users may frequently send time-sensitive information regarding projects or questions made by mentees.

* Allow users to send a message to another user.
* Allow users to read a message.
* Allow users to delete a message.
* Send ticket due email notification.
* Send new message email notification.
* Send password change email notification.
* Send profile change email notification.
* Send ticket assigned email notification.
* Send comment added to a ticket email notification.
* Send ticket closed email notification.
* Send ticket not available mentor email notification in the Automatic Reassign.
* Send ticket reassigned email notification to admin when the ticket was automatically reassigned three times.
* Send ticket escalated email notification.

**Integration with SPW**

* Allow admin to import data from SPW manually
* Allow the system to import data(students/projects) from SPW automatically
* Allow senior project students to login with their FIU Google login credentials
* Allow system to update reassign projects based on changes made in SPW
* Allow system to update reassign mentors based on changes made in SPW

**Mentor Module: Version 3**

The system shall…

**Self-serve Mentor Registration**

* Allow a potential mentor to register for a mentor account from a new landing page.
* Allow mentors to apply for new mentorships from and all-in-one Mentor Application portal
* Allow mentors to begin their mentorships whenever approved by an admin
* Allow a Project Mentor to apply by selecting projects they like or deferring the decision to the system.
* Allow a Personal Mentor to apply by selecting students they like or deferring the decision to the system.
* Allow a Domain Mentor to apply by selecting topics they’re proficient in.
* Allow a Domain Mentor to suggest new domains.
* Allow Project Mentor to approve/reject projects suggested by admin.
* Allow Domain Mentor to approve/reject topics suggested by admin.
* Allow Personal Mentor to approve/reject mentee suggested by admin.
* Provide user verification for all applications
* Provide client side validation for all forms

**Admin Dashboard**

* Allow admin to view pending applications
* Allow admin to approve/reject Projects in a Project Mentor Application
* Allow admin to propose Projects in a Project Mentor Application
* Allow admin to approve/reject Mentees in a Personal Mentor Application
* Allow admin to propose Mentees in a Personal Mentor Application
* Allow admin to approve/reject Domains in a Domain Mentor Application
* Allow admin to propose Domains in a Domain Mentor Application
* Allow admin to view open invitations/re-invites
* Allow admin to customize the body of the message while sending an invite
* Allow admin to gain insight on site usage through admin dashboard
* Allow admin to gain insight on a particular project
* Allow admin to gain insight on a particular user
* Allow admin to gain insight on a particular domain
* Allow admin to gain insight on a particular subdomain
* Allow admin to view tickets
* Allow admin to manage tickets
* Allow admin to use an advanced search for tickets.

**Mentor Module: Version 5**

* Allow admin to pull reports on mentors.
* Allow admin to pull reports on mentees.
* Allow admin to pull reports on Tickets.
* Allow admin to research on the utilization of tickets created.
* Allow admin to research on the utilization of tickets closed.
* Allow admin to research on the utilization of tickets duration (time opened to closed)
* Allow admin to research on the time that takes mentors to answer.
* Allow admin to research on the tickets that are still open (may or may not be answered).
* Allow admin to get system suggestion on what kind of mentors to find (based on frequent mentee subdomains).
* Allow users to schedule meetings on-demand.
* Allows users to schedule meetings ahead of time.
* Allow mentors to schedule meetings from a ticket.
* Allow users to see/hear each other during videoconferences.
* Allow users to use the chat-room during videoconferences.
* Allow users to have a whiteboard to draw and share it during videoconferences.
* Allow users to share their screens with the other meeting participants.
* Allow users to stop sharing their screens with the other meeting participants.
* Allow users to invitee others during a videoconference.
* Allow invitee to accept a video conference invitation
* Allow invitee to reject a video conference invitation
* Allow moderators to delete their videoconferences.
* Allow moderators to cancel their videoconferences.

**Mentor Module: Version 6**

**The system Shall.**

* Allow users to turn their webcam on/off during a video conference.
* Allow users to reset their password using the forgot password option.
* Allow users to register on their own from the homepage.
* Allow users registered by an administrator to change their password.
* Allow users to have a second screen for screen-sharing.
* Allow users to join a video conference without synchronization problems.
* Allow moderators to edit a video conference’s information such as title, date and time, notes, and invitees.
* Allow users to have a robust video conference room.
* Provide users with a new chat feature, one that they can hide and unhide for convenience.
* Allow users to join a conference after it has started and still receive all the video streams and screens being shared.
* Allow users to invite others by name when creating or editing a video conference.
* Allow users to access past and canceled video conferences.
* Provide users with a calendar widget when creating or editing a conference.
* Alert users if they don’t have the “Screen Capturing” plugin when joining a conference.
* Allow users to be set as away and no longer receive tickets
* Have the system reassign tickets that need to be reassigned automatically
* Allow users to send the system email.
* Allow the mentors to be selected by round robin
* Allow mentors to see all the information of the ticket in the email we send them
* Allow mentor to pass on a ticket through a link in the email
* Allow the admin to manage the mentors who are away, removing them, or adding them by using their name with autocomplete
* Allow mentors to manually remove themselves from being away
* Provide the ability for users to sort their tickets the way they like on their homepage
* Allow prospective mentors to suggest a new domain that fits their area of expertise
* Allow the system administrator to adjust the time limits given to tickets
* Allow users to be able to see the last recorded for each ticket on their home page.
* Allow users to see their upcoming meetings on their home page
* Allow users to look at tickets that are not assigned to them or created by them
* Allow the user to search these tickets by certain criteria.
* Allow the mentor to link directly to the ticket through the email sending them the ticket.
* Notify mentors when their tickets are reassigned

**Module Version 7**

**The system shall:**

* Allow users to post feedback
* Allow users to view other user feedback
* Allow users to update feedback
* Allow admin to manage feedback
* Allow user to reply to own feedback
* Allow admin to reply to user feedback

Deployment Diagram

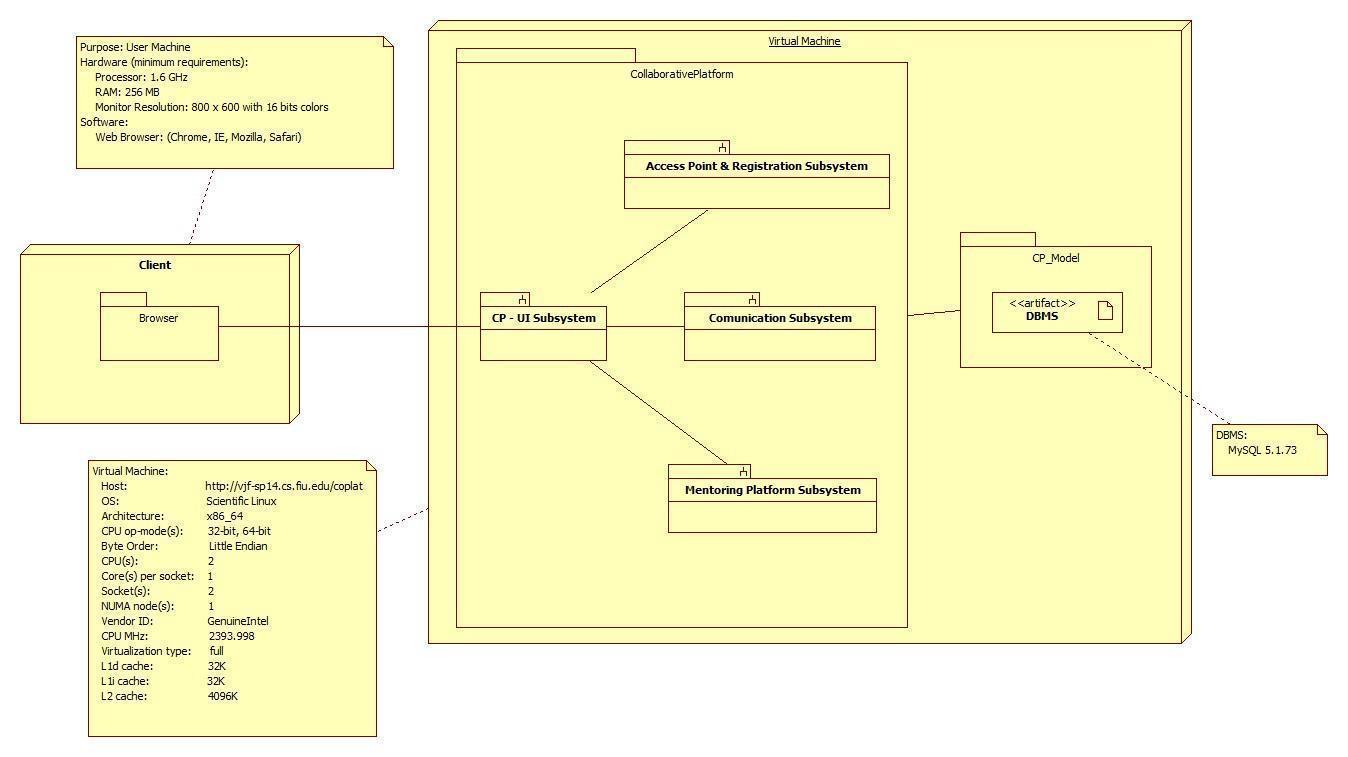


Figure 1.3 Deployment Diagram

## Design Patterns

We have used the unified software process model as the basis for our planning and organization during the design of our software. The model is largely based on the construction of the specific use cases to help formulate and drive the design process. After formulating a set of uses cases, the team can then take those specific use cases and apply them to different UML diagrams to flesh out the design process. For example, class diagrams can show the relationships between different classes within subsystems and how specific subsystems can relate to one another. Sequence diagrams help plan out specific paths of method calls before the team actually begins to code. Finally having carefully specified requirements document allows the team to stay focused and on task as to the purpose of the design.

**System Validation**

The system validation chapter introduces testing to make sure that the project works as was intended. This sections contains use cases that were used to test the different functionalities implemented during this term.

**User Story #8 (also #17) – Finish Feedback Functionality**

|  |  |  |  |
| --- | --- | --- | --- |
| Create | Unit | FeedbackTest.php | testCreate() |
| AfterCreate | Unit | FeedbackTest.php | testAfterCreate() |
| Delete | Unit | FeedbackTest.php | testDelete() |
| AfterDelete | Unit | FeedbackTest.php | testAfterDelete() |

**User Story #9 (also #17) – Finish Feedback Replies Functionality**

|  |  |  |  |
| --- | --- | --- | --- |
| AfterCreate | Unit | Feedback\_RepliesTest.php | testAfterCreate() |
| AfterDelete | Unit | Feedback\_RepliesTest.php | testAfterDelete() |

**User Story #12 – User Unit Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| Existing\_User\_Login | Unit | UserActionTest.php | testExisting\_User\_Login() |
| Logout | Unit | UserActionsTest.php | testLogout() |
| Register | Unit | UserActionsTest.php | testUser\_Has\_Been\_  Registered\_Successfully() |
| Forgot Password | Unit | UserActionsTest.php | testPassword\_Change\_  Using\_Forgot\_Password() |
| Edit Bio | Unit | UserActionsTest.php | testEdit\_User\_Biography() |
| Edit Availability | Unit | UserActionsTest.php | testSet\_Available\_User\_  Unavailable() |
| Change Password | Unit | UserFunctionalityTest.php | testChange\_Password\_  When\_Logged\_In() |

**User Story #13 – Ticket Unit/Functionality Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| AddDomain | Functionality | TicketFunctionlaity | SuggestDomain.html |
| Send Ticket Reassigned Notification | Functionality | EmailNotifications | TicketAssignReassigned.html |
| Reassign Ticket | Functionality | TicketFunctionality | ReassignTicket.html |
| Reassign Project Mentor | Functionality | TicketFunctionality | ReassignProjectMentor.html |
| Create Ticket | Functionality | TicketFunctionality | CreateTicket.html |
| Upload a file to a ticket | Functionality | TicketFunctionality | UploadFile.html |
| Download file from a ticket | Functionality | TicketFunctionality | DownloadFile.html |
| Append a Comment | Functionality | TicketFunctionality | AppendAComment.html |
| Find Ticket | Functionality | TicketFunctionality | FindTicket.html |
| Close A Ticket | Functionality | TicketFunctionality | CloseTicket.html |
| Reject A Ticket | Functionality | TicketFunctionality | RejectTicket.html |
| Escalate A Ticket | Functionality | TicketFunctionality | EscalateTicket.html |
| Allow admin to view tickets | Functionality | TicketFunctionality | AdminTicketView.html |
| Allow admin to manage tickets | Functionality | TicketFunctionality | AdminManageTickets.html |
| Allow admin to use advanced search for tickets | Functionality | TicketFunctionality | AdminManageTickets.html |
| Automatically assign ticket | Unit | TicketFunctionalityTest.php | testAuto\_Assign\_Ticket() |
| Reassign Ticket | Unit | TicketFunctionalityTest | testReassign\_Ticket($id) |
| Automatically assign ticket | Unit | TicketFunctionalityTest.php | testAuto\_Assign\_Ticket() |
| Reassign Ticket | Unit | TicketFunctionalityTest | testReassign\_Ticket($id) |
| Assign ticket to other project mentor | Unit | TicketFunctionalityTest.php | testReassign\_Ticket\_To\_  Other\_Project\_Mentor($id) |
| Retrieve the tickets created by their mentees | Unit | TicketFunctionalityTest.php | testRetrieve\_The\_Tickets\_  Created\_by\_Their\_  Mentees() |
| Assign tickets to his/her project mentor | Unit | TicketFunctionalityTest.php | testCreate\_A\_Ticket() |
| Create a ticket | Unit | TicketFunctionalityTest.php | testCreate\_A\_Ticket() |
| Select a specific domain for a new ticket | Unit | TicketFunctionalityTest.php | testCreate\_A\_Ticket() |
| Select a specific sub-domain | Unit | TicketFunctionalityTest.php | testCreate\_A\_Ticket() |
| Append comments to a ticket | Unit | TicketFunctionalityTest.php | testAppend\_comments\_to\_  a\_ticket() |
| Retrieve the ticket details | Unit | TicketFunctionalityTest.php | testRetrieve\_the\_ticket\_  details($id) |
| Close a Ticket | Unit | TicketFunctionalityTest.php | testClose\_A\_Ticket($id) |
| Reject a Ticket | Unit | TicketFunctionalityTest.php | testReject\_A\_Ticket($id) |
| System Automatically Reassigns Ticket | Unit | TicketFunctionalityTest.php | testReassign\_Ticket($id) |
| Retrieve To Personal Mentors the Tickets Created by their Mentees | Unit | TicketFunctionalityTest.php | testRetrieve\_The\_Tickets\_  Created\_By\_Their\_  Mentees() |
| Edit Priority | Unit | TicketFunctionalityTest.php | testChange\_Ticket\_Priority() |
| Escalate Ticket | Unit | TicketFunctionalityTest.php | testEscalate\_A\_Ticket($id) |
| Assign tickets to his/her personal Mentor | Unit | TicketFunctionalityTest.php | testReassign\_Ticket\_To\_  Other\_Project\_Mentor($id) |

**User Story #16 – Mentor Unit Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| Add Domain | Unit | MentorApplicationTest.php | testAdd\_a\_Domain() |
| Add Subdomain | Unit | MentorApplicationTest.php | testAdd\_a\_SubDomain() |
| Apply for Personal Mentorship | Unit | MentorApplication.php | testApply\_for\_Personal\_  Mentorship() |
| Search for User(s) | Unit | MentorApplicationTest.php | testAllow\_admin\_to\_view\_  specified\_user() |
| Apply for Project Mentorship | Unit | MentorApplication.php | testApply\_for\_Project\_  Mentorship() |
| Apply for Domain Mentorship | Unit | MentorApplication.php | testApply\_for\_Domain\_  Mentorship() |
| Respond to Mentorship Offer | Unit | MentorApplicationTest.php | testAllow\_Admin\_to\_  approve\_items\_in\_  application($id) |
| Allow admin to gain insight on particular user | Unit | MentorApplicationTest.php | testAllow\_admin\_to\_view\_  specified\_user() |

**User Story #20 – Mentor Functionality Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| SuggestDomain | Functionality | Mentor Functionality Suite | SuggestDomain.html |
| PersonalMentorApply | Functionality | Mentor Functionality Suite | Personal Mentor Apply.html |
| ProjectMentorApply | Functionality | Mentor Functionality Suite | Project Mentor Apply.html |
| Allow admin to view pending application | Functionality | Mentor Functionality Suite | ViewPendingApps.html |
| DomainMentorApply | Functionality | Mentor Functionality Suite | Domain Mentor Apply.html |
| Allow admin to gain insight on a particular domain | Functionality | Mentor Functionality Suite | FindDomain.html |
| Allow admin to approve/reject items in an application | Functionality | Mentor Functionality Suite | AppItemAcceptReject.html |
| Allow admin to propose items in an application | Functionality | Mentor Functionality Suite | ProposeItemsOnApp.html |
| AcceptApplication | Functionality | Mentor Functionality Suite | AcceptApplication.html |
| RejectApplication | Functionality | Mentor Functionality Suite | RejectApplication.html |

**User Story #21 – User Functionality Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| Login | Functionality | UserFunctionality | Login.html |
| Logout | Functionality | UserFunctionality | Logout.html |
| RegisterUser | Functionality | UserFunctionality | RegisterUser.html |
| ForgotPassword | Functionality | UserFunctionality | ForgotPassword.html |
| EditBiography | Functionality | UserFunctionality | EditBiography.html |
| SetMentorAway | Functionality | UserFunctionality | SetMentorAway.html |
| Allow the admin to Register Mentors | Functionality | UserFunctionality | AdminRegisterUser.html |
| Find User | Functionality | UserFunctionality | FindUser.html |
| Allow admin to see particular user | Functionality | UserFunctionality | FindUser.html |
| Allow admin to open invites | Functionality | UserFunctionality | ViewInvites.html |
| Allow admin to send custom invite | Functionality | UserFunctionality | SendCustomInvite.html |
| ChangePassword | Functionality | UserFunctionality | ChangePassword.html |

**User Story #22 – Report/Message Functionality Tests**

|  |  |  |  |
| --- | --- | --- | --- |
| Pull amount of tickets created | Functionality | ReportFunctionality | TicketReports.html |
| Pull tickets created details | Functionality | ReportFunctionality | TicketReports.html |
| Pull amount of tickets closed | Functionality | ReportFunctionality | ClosedTickets.html |
| Pull tickets closed details | Functionality | ReportFunctionality | ClosedTickets.html |
| Pull amount of tickets currently open | Functionality | ReportFunctionality | OpenTickets.html |
| Pull tickets currently open details | Functionality | ReportFunctionality | OpenTickets.html |
| Pull Mentee Reports | Functionality | ReportFunctionality | MenteeReports.html |
| Pull Mentor Reports | Functionality | ReportFunctionality | MentorReports.html |
| Pull Ticket Reports | Functionality | ReportFunctionality | TicketReports.html |
| Get Inbox Messages | Functionality | MessageFunctionality | OpenInbox.html |
| Get Sent Messages | Functionality | MessageFunctionality | SendMessage.html |
| Get Trashed Messages | Functionality | MessageFunctionality | DeleteMessage.html |
| DeleteMessage | Functionality | MessageFunctionality | DeleteMessage.html |
| ReplyToMessages | Functionality | MessageFunctionality | ReplyToMessages.html |

**User Story #23 – Write assorted unit tests**

|  |  |  |  |
| --- | --- | --- | --- |
| Retrieve the Project Information | Unit | AssortedFunctionalitiesTest | testRetreieve\_Project\_  Information() |
| Set up Meetings | Unit | AssortedFunctionalitiesTest | testSet\_Up\_Meetings() |
| Retrieve all the Upcoming Meetings | Unit | AssortedFunctionalitiesTest | testView\_Upcoming\_  Meetings() |
| Allow admin to view open invitations/re-invites | Unit | AssortedFunctionalitiesTest | testView\_Open\_Invitations() |
| Allow admin to send a custom invite | Unit | AssortedFunctionalitiesTest | testSend\_Custom\_Invite() |

**User Story #24 – Write Video Conference Unit Tests**

|  |  |  |  |
| --- | --- | --- | --- |
| Set up Meetings | Unit | MeetingsTest.php | testSet\_Up\_Meetings() |
| Retrieve All Upcoming Meetings | Unit | MeetingsTest.php | testRetrieve\_All\_  Upcoming\_Meetings() |
| Invite More People | Unit | MeetingsTest.php | testInvite\_More\_  People() |
| Accept Conference Invitation | Unit | MeetingsTest.php | testAccept\_Conference\_  Invitation() |
| Reject Conference Invitation | Unit | MeetingsTest.php | testReject\_Conference\_  Invitation() |
| Cancel Conference Invitation | Unit | MeetingsTest.php | testCancel\_Conference\_  Invitation() |
| Edit Video Conference | Unit | MeetingsTest.php | testEdit\_Video\_  Conference\_Information() |

**User Story #25 – Write Video Conference Unit/Functionality Tests**

|  |  |  |  |
| --- | --- | --- | --- |
| Reply to Messages | Unit | MessageTest.php | testReply\_To\_Messages() |
| Delete Messages | Unit | MessageTest.php | testDelete\_Message($id) |
| Invite More People | Functionality | VideoConference  Functionality | InviteMorePeople.html |
| Accept Video Conference Invitation | Functionality | VideoConference  Functionality | AcceptConference.html |
| Reject Video Conference Invitation | Functionality | VideoConference  Functionality | RejectConference.html |
| Cancel Video Conference Meeting | Functionality | VideoConference  Functionality | CancelConference.html |
| Schedule a Video Conference Ahead of Time | Functionality | VideoConference  Functionality | FutureConference.html |
| Schedule On-Demand Video Conference | Functionality | VideoConference  Functionality | onDemandConference.html |
| Join Video Conference | Functionality | VideoConference  Functionality | JoinRoom.html |
| Share Screen | Functionality | VideoConference  Functionality | ShareScreen.html |

**User Story #26 – Write Administrator Tests**

|  |  |  |  |
| --- | --- | --- | --- |
| Create Admin | Unit | AdministratorTest.php | testAfterCreate() |
| Delete Admin | Unit | Administratortest.php | testAfterDelete() |

**User Story #27 – Write application functionality tests**

|  |  |  |  |
| --- | --- | --- | --- |
| Allow Mentees to login using their FIU Google credentials | Functionality | ApplicationFunctionality | FUISignIn.html |
| Import Data from SPW Manually | Functionality | ApplicationFunctionality | SyncWithSPW.html |

**User Story #28 – Write email functionality tests**

|  |  |  |  |
| --- | --- | --- | --- |
| Send Meeting Notification | Functionality | EmailNotifications | MeetingNotification.html |
| Send Ticket Assigned Notification | Functionality | EmailNotifications | TicketAssignReassigned.html |
| Send Comment Added Notification | Functionality | EmailNotifications | CommentNotification.html |
| Send Invitation Notification | Functionality | EmailNotifications | InviteNotification.html |

# **Glossary**

Definitions:

* **User** – A generalization given to all user groups.
* **Student** – Individual who is currently enrolled in the senior project class.
* **Mentorship** - is a personal developmental relationship in which a more experienced or more knowledgeable person helps to guide a less experienced or less knowledgeable person.
* **Mentor** - An Individual with a Collaborative Platform account seeking Mentorship.
* **Project** **Mentor** – An individual who is assigned to overlook one or many senior projects.
* **Domain** **Mentor**– An individual with particular knowledge in a certain skill or language(2 Tiers)
* **Personal Mentor** –An individual that mentors only a single person – not limited to career advice.
* **Product owner**–The product owner represents the client and has enough domain knowledge to answer the questions the development team may have about the software product.
* **Mentee** - An individual that may request help or ask a question.
* **Administrator** –An individual assigned with administering the web portal.
* **Application** - A form containing information provided by a Mentor seeking a Mentorship.
* **System Pick** - An automated selection done by the system.

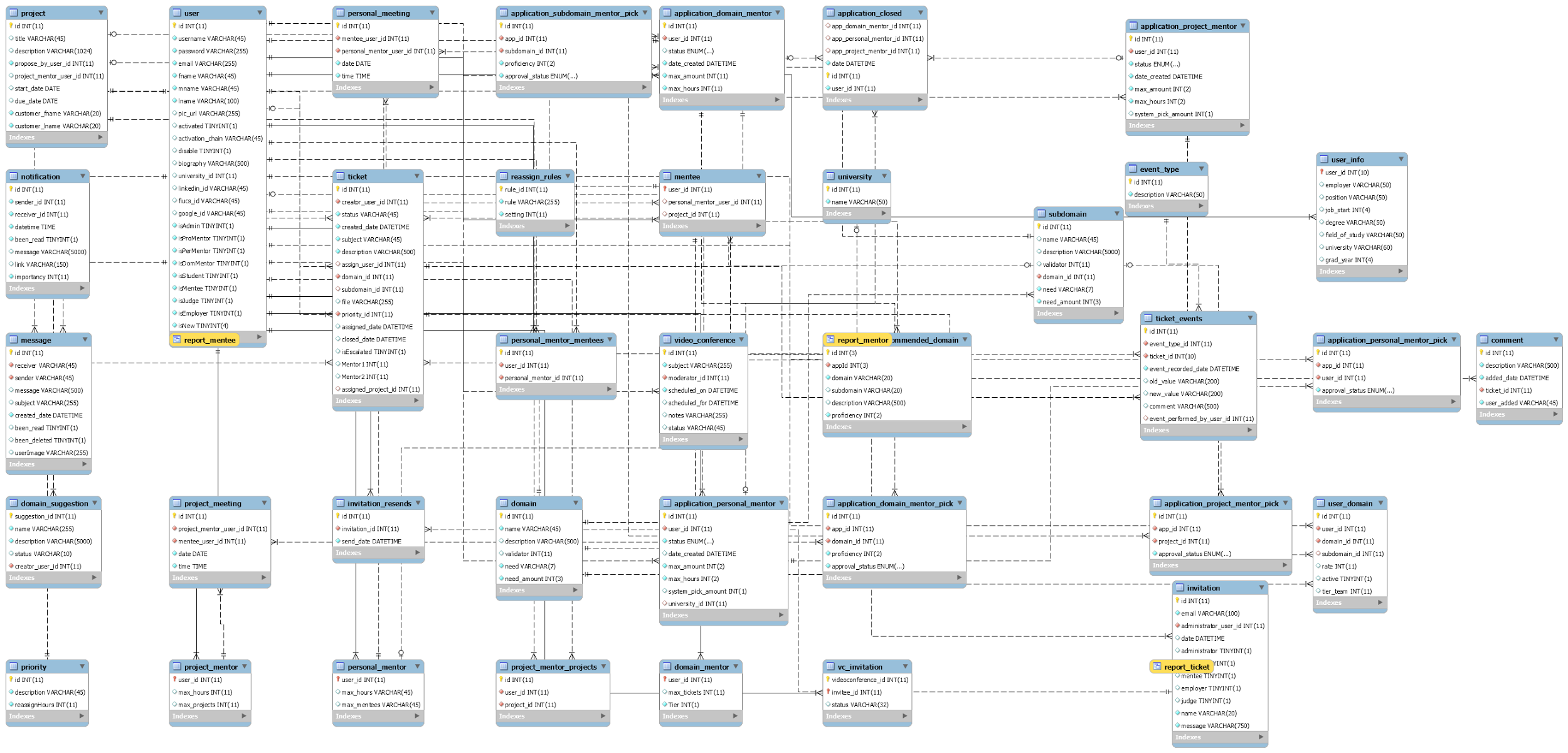
Abbreviations:

* **N/A**

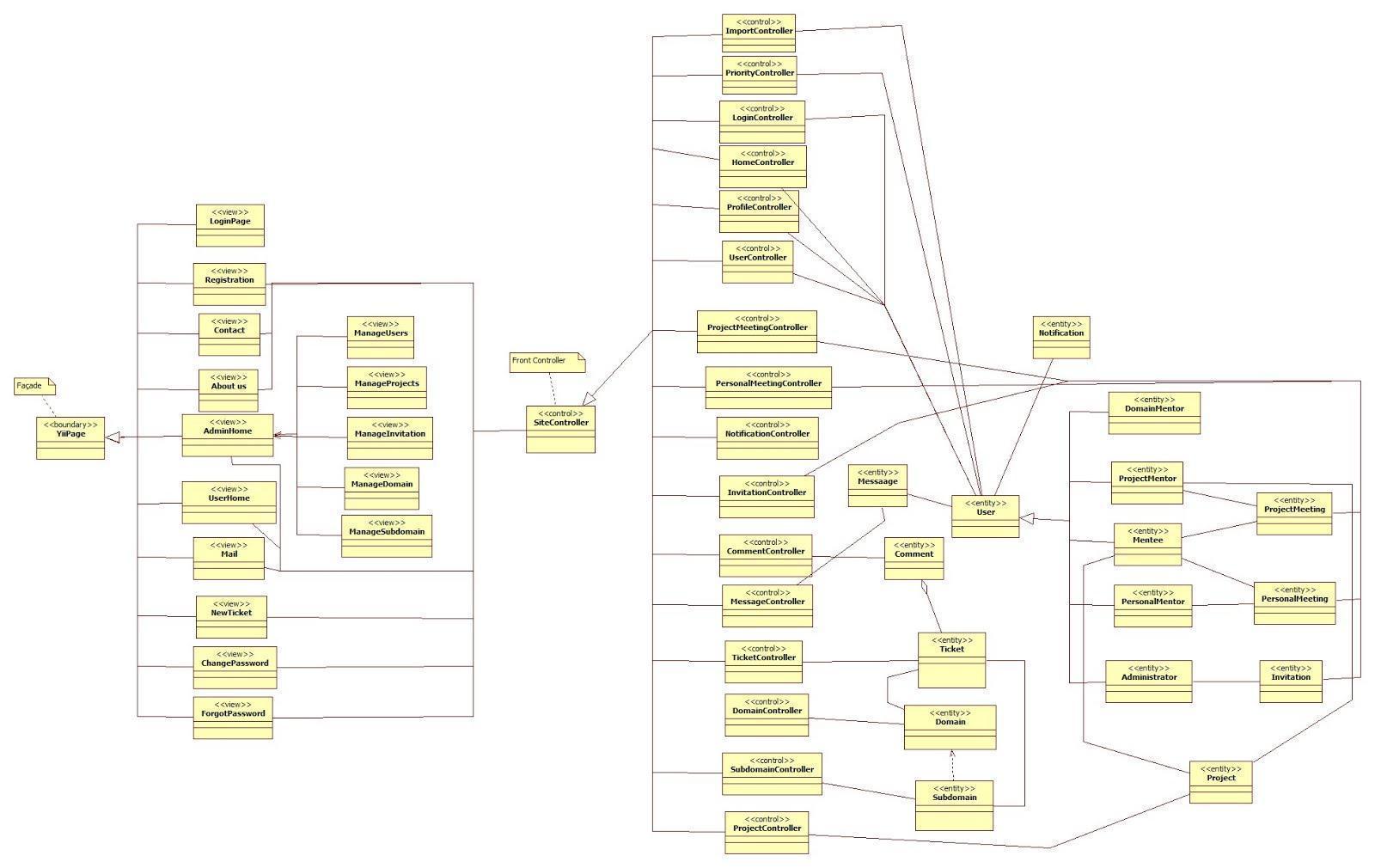
# **Appendix**

## Appendix A - UML Diagrams

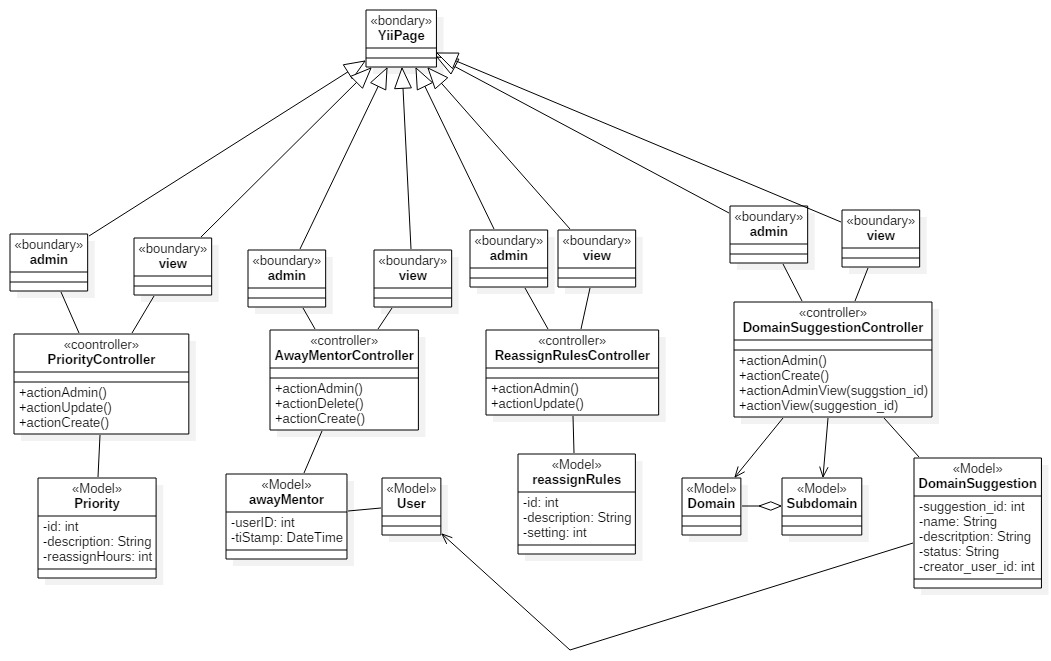
### Static UML Diagrams



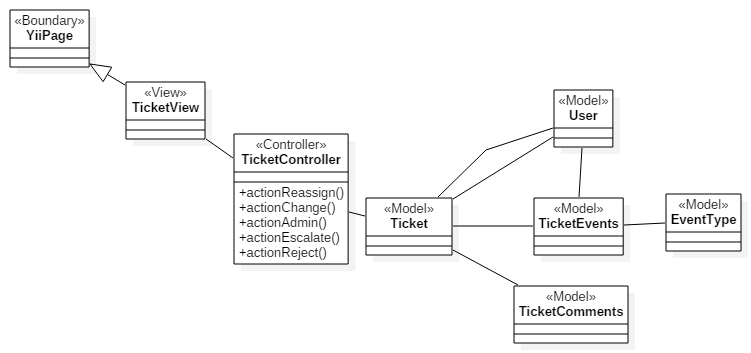
AS.1 Persistent Data Model



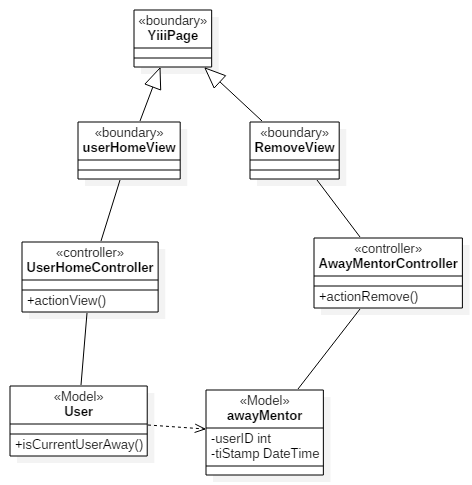
AS.2 Minimal Class Diagram



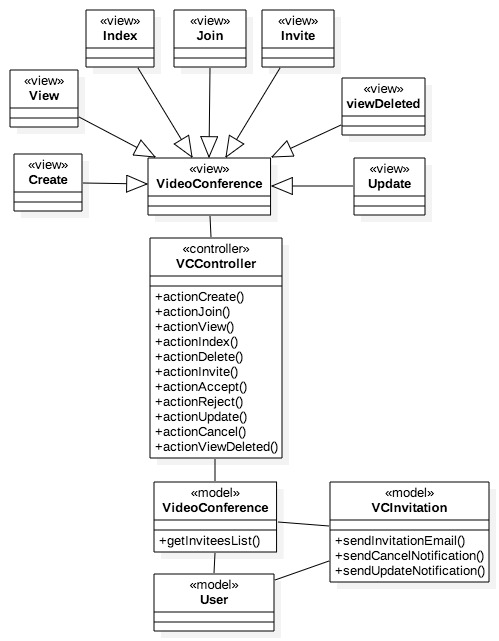
AS.3 New Controllers, Models, And Views For new systems



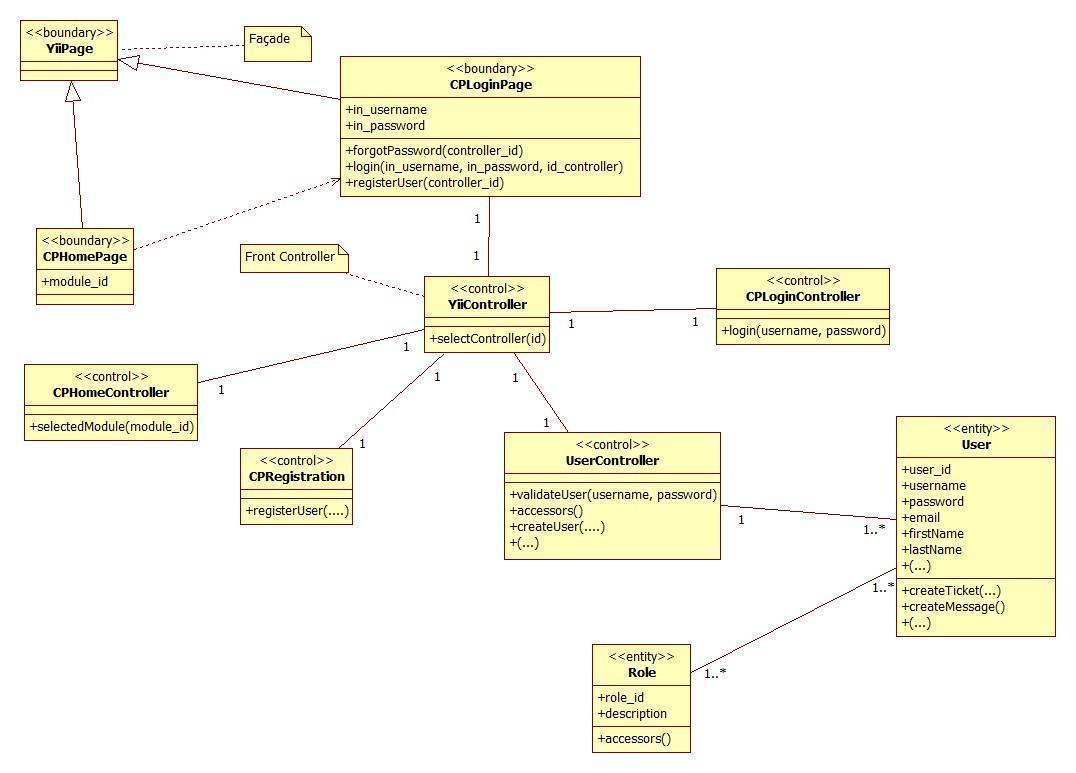
AS.4 New ticket controller and view class interactions



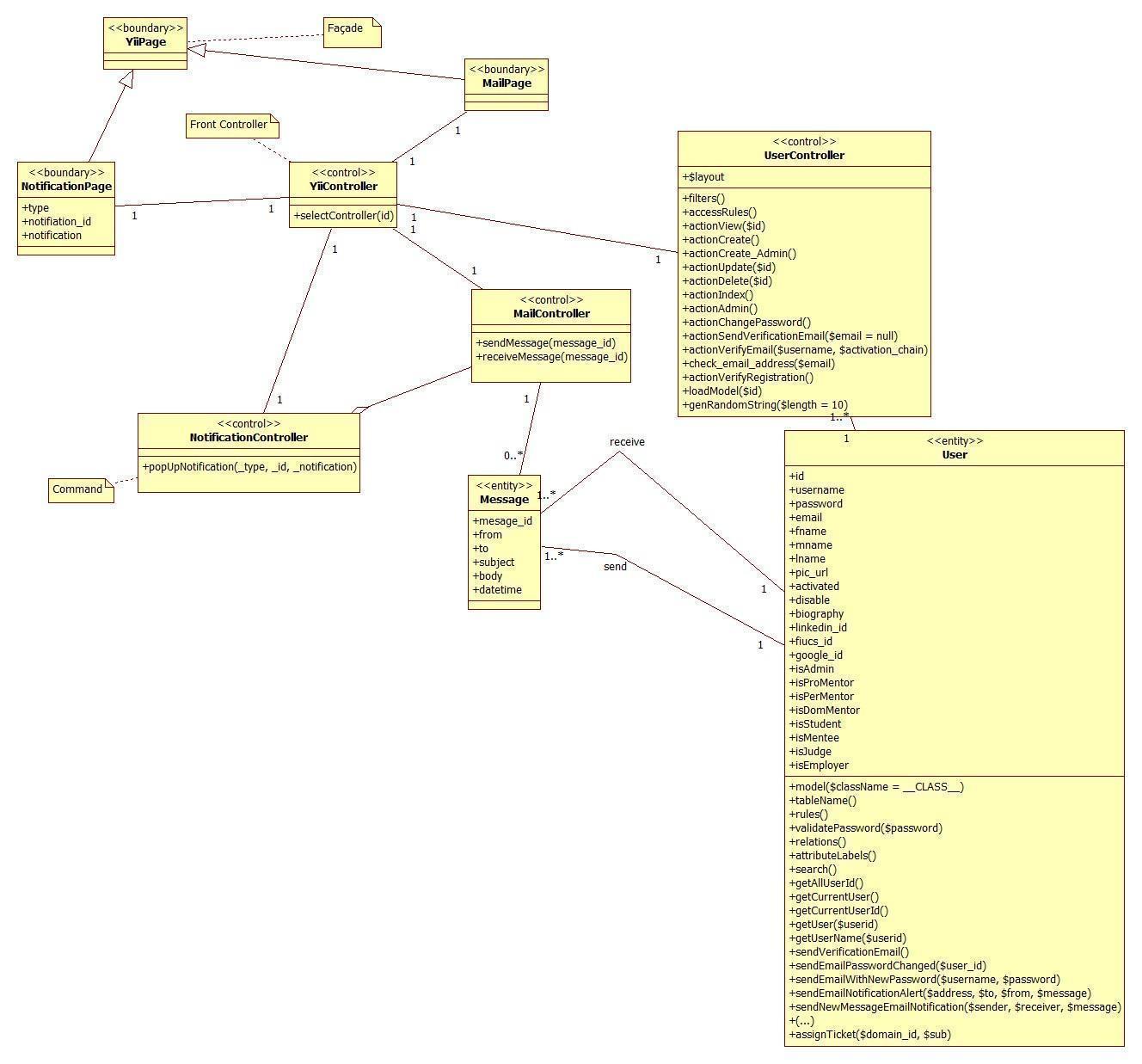
AS.5 New User, UserHome, AwayMentor interactions for user home



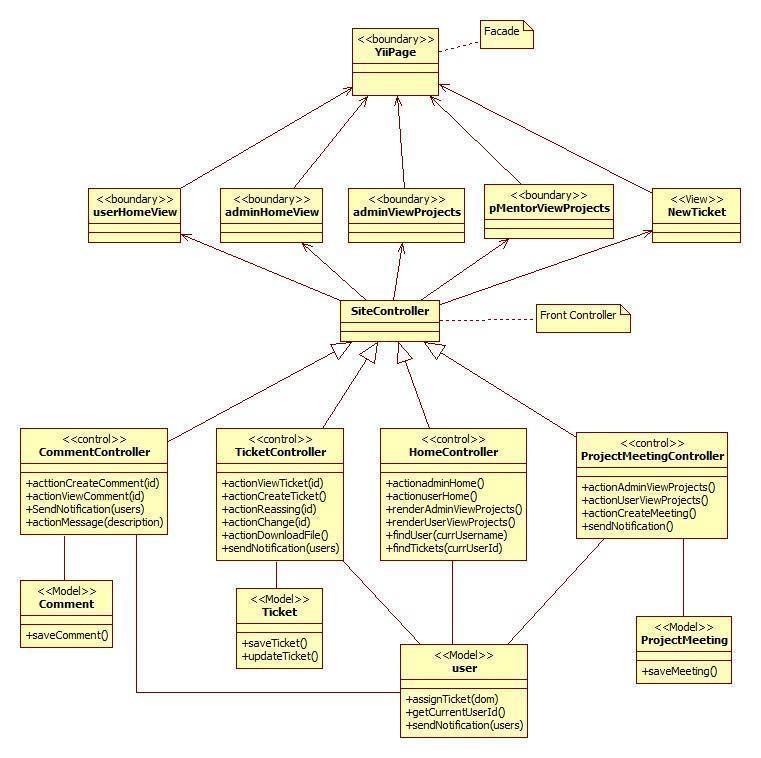
AS.6 Class Diagram for the Video Conference Module



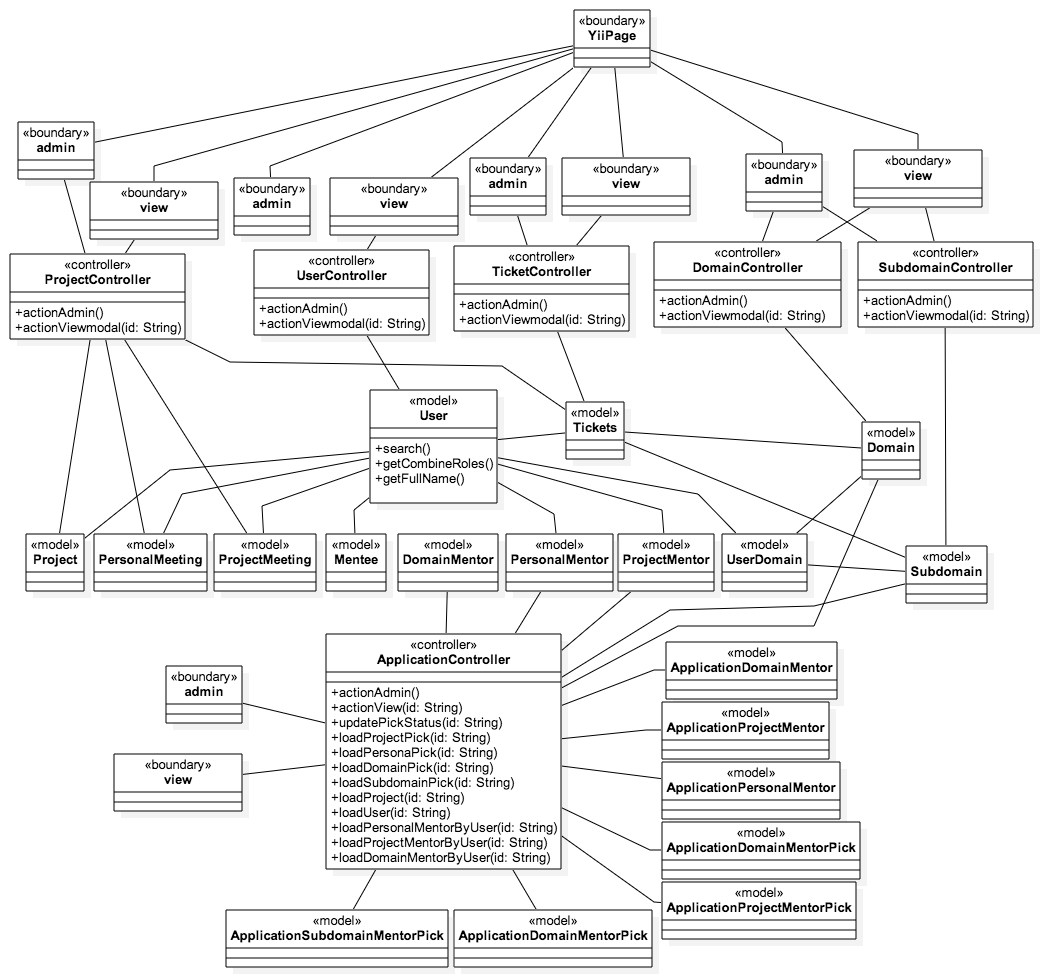
**AS.7 Access Point & Registration Detailed Class Diagram**



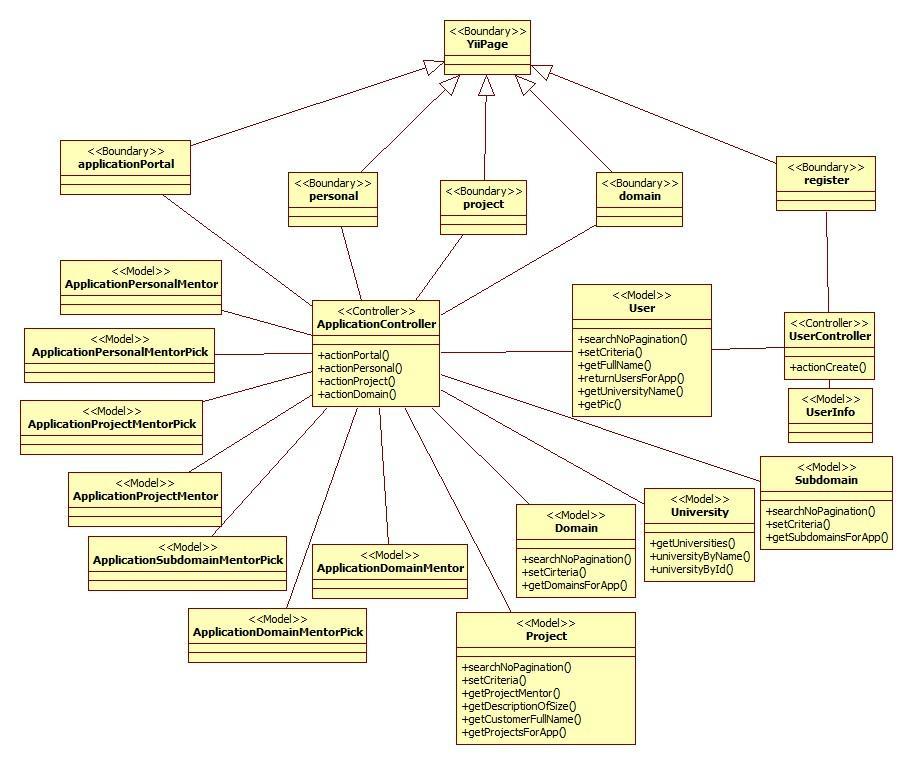
**AS.8 Communication and Notification Subsystem Detailed Class Diagram**



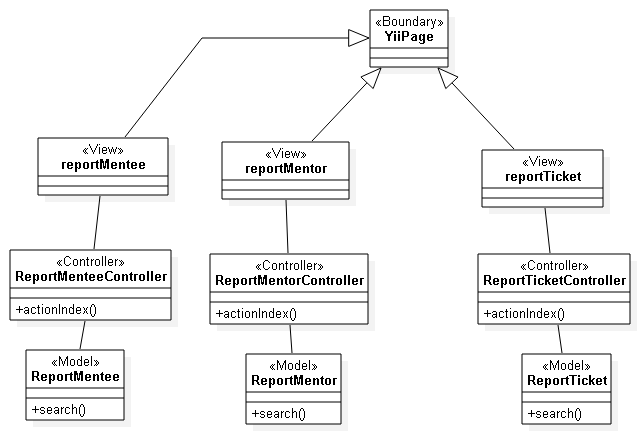
**AS.9 Mentoring Subsystem Detailed Class Diagram**



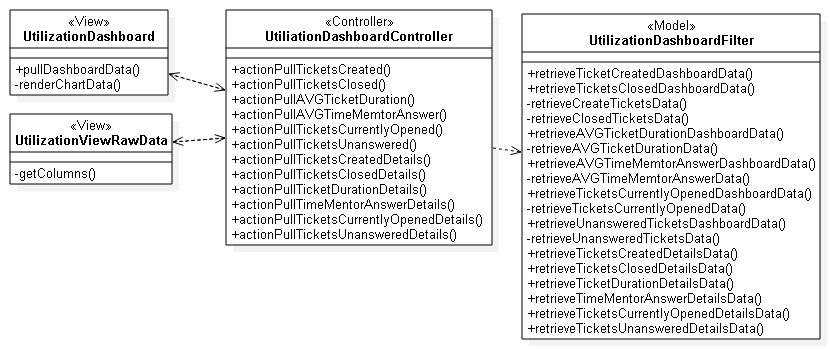
**AS.10: Admin Management Detailed Class Diagram**



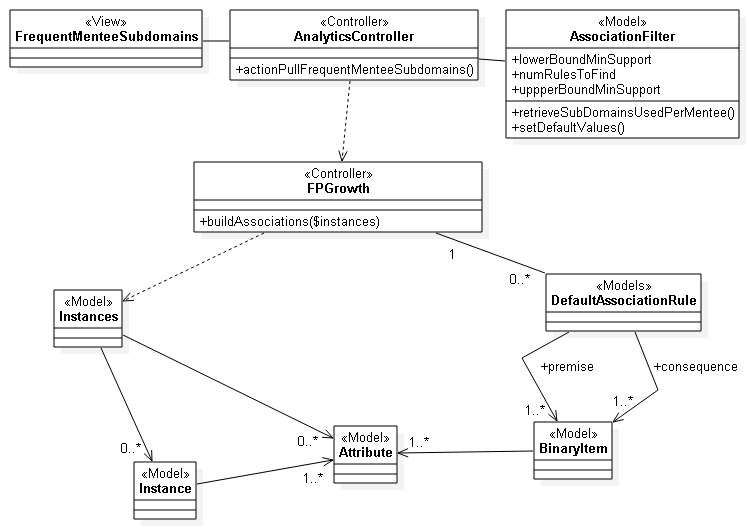
**AS.11: Self-Registration Detailed Class Diagram**



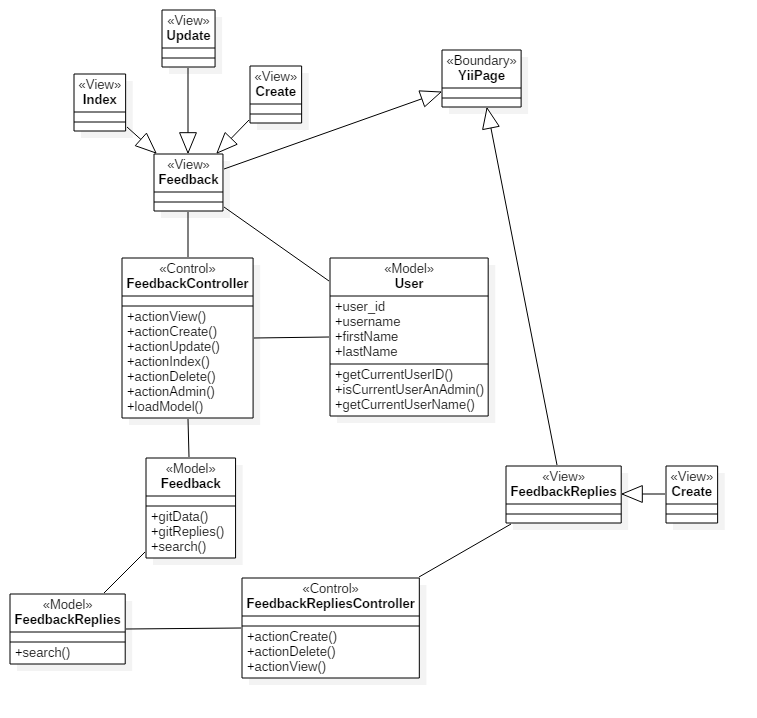
**AS.12: Admin reports Detailed Class Diagram**



**AS.13: Utilization dashboard Detailed Class Diagram**

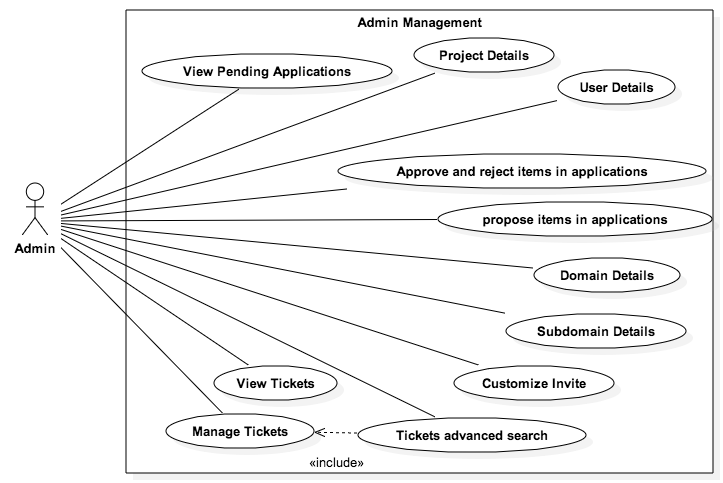


**AS.14: Analytical reporting (Frequent sub-domain questions by mentee) detailed class diagram.**

****

**AS.15, Feedback and Feedback Replies detailed class diagram.**

CPv3-UDC-02: Admin Dashboard

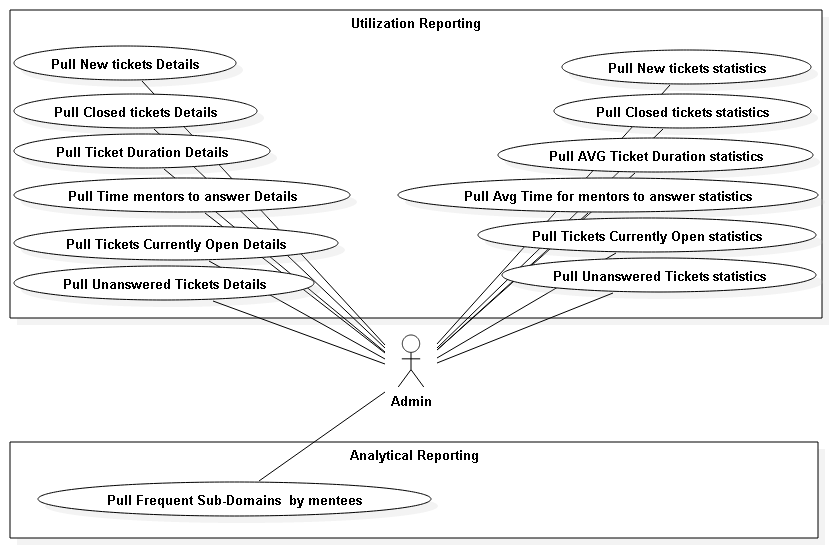


**Mentoring Module: Version 5**

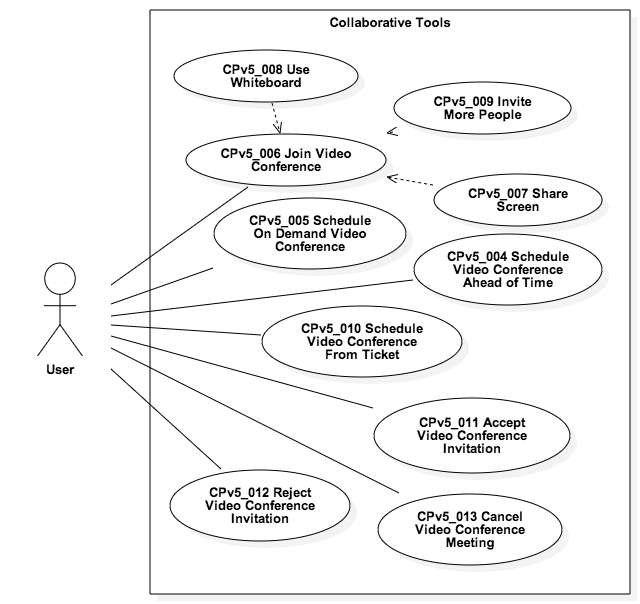
CPv5-UDC-01: Admin Report



CPv5-UDC-02: Utilization reporting and analytics



CPv5-UDC-03: Collaborative Tools



**Mentoring Module: Version 6 UC1**

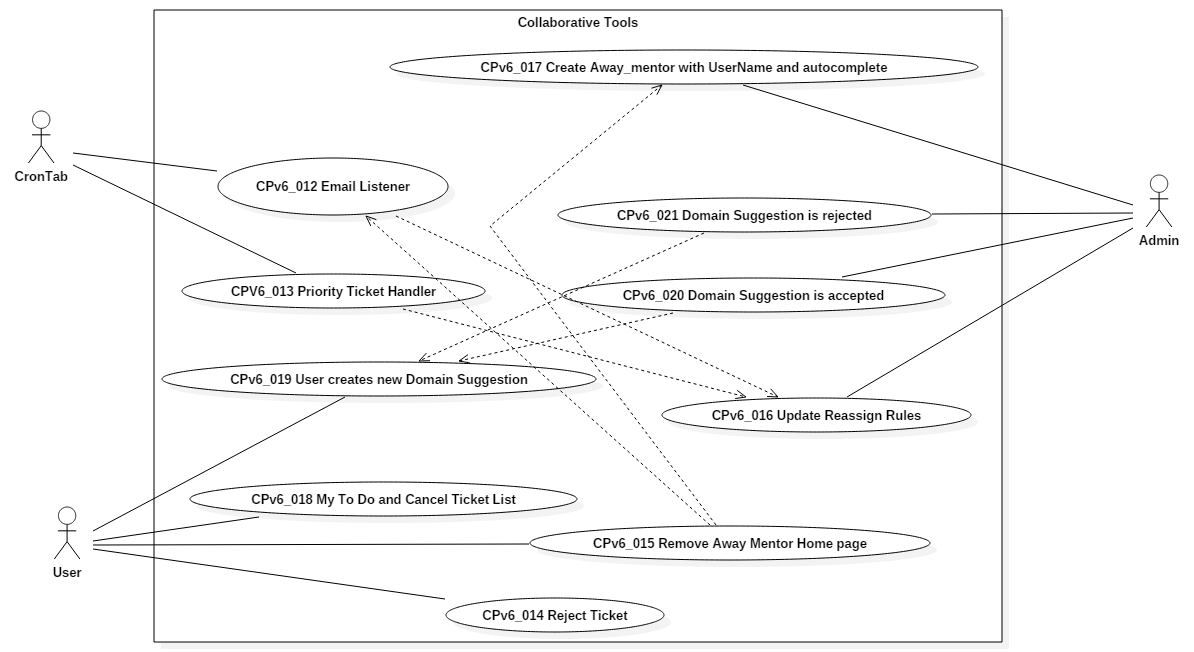
### 

### 

### 

### 

**Mentoring Module: Version 6 UC2**



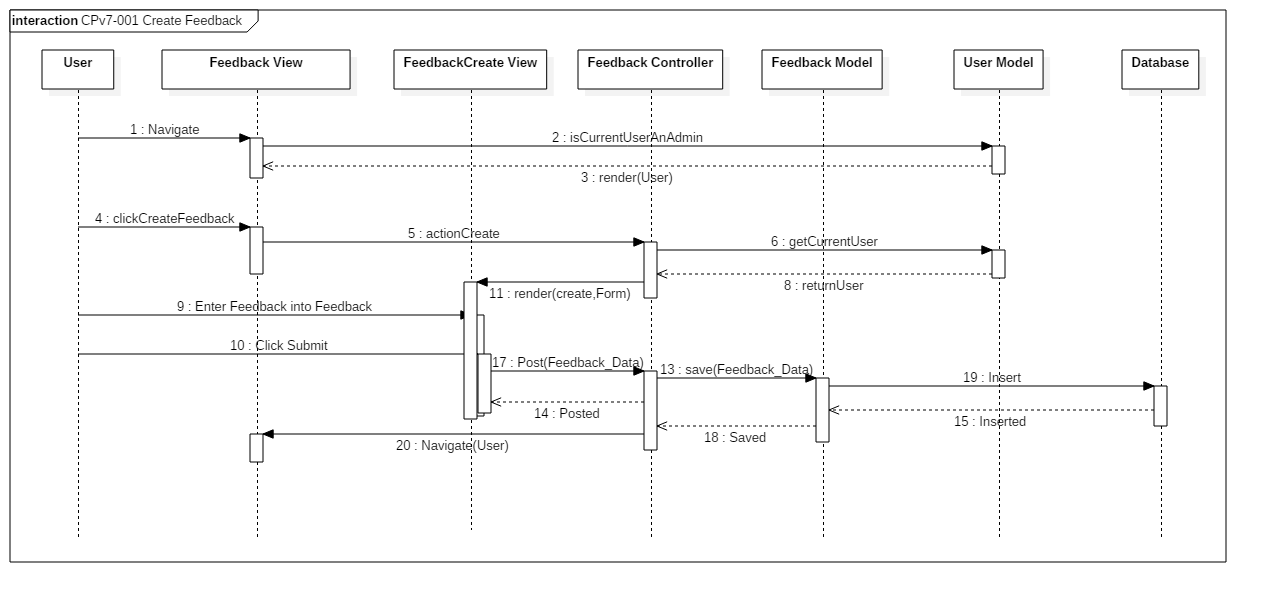
### 

### 

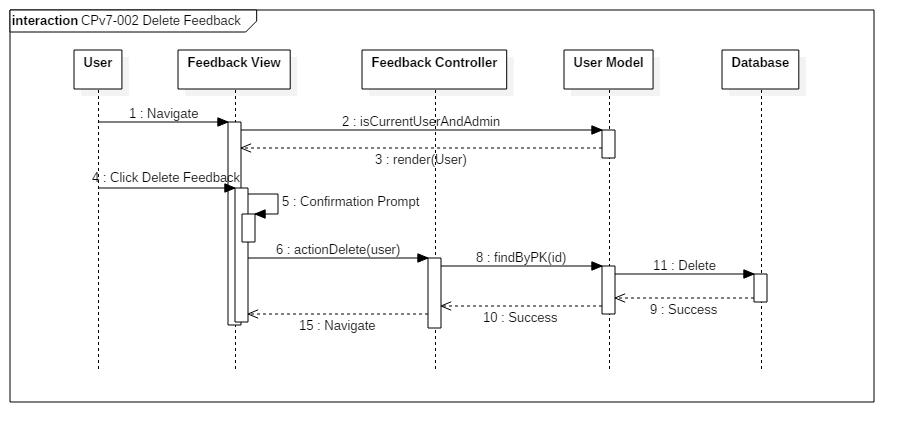
### 

### Dynamic UML Diagrams

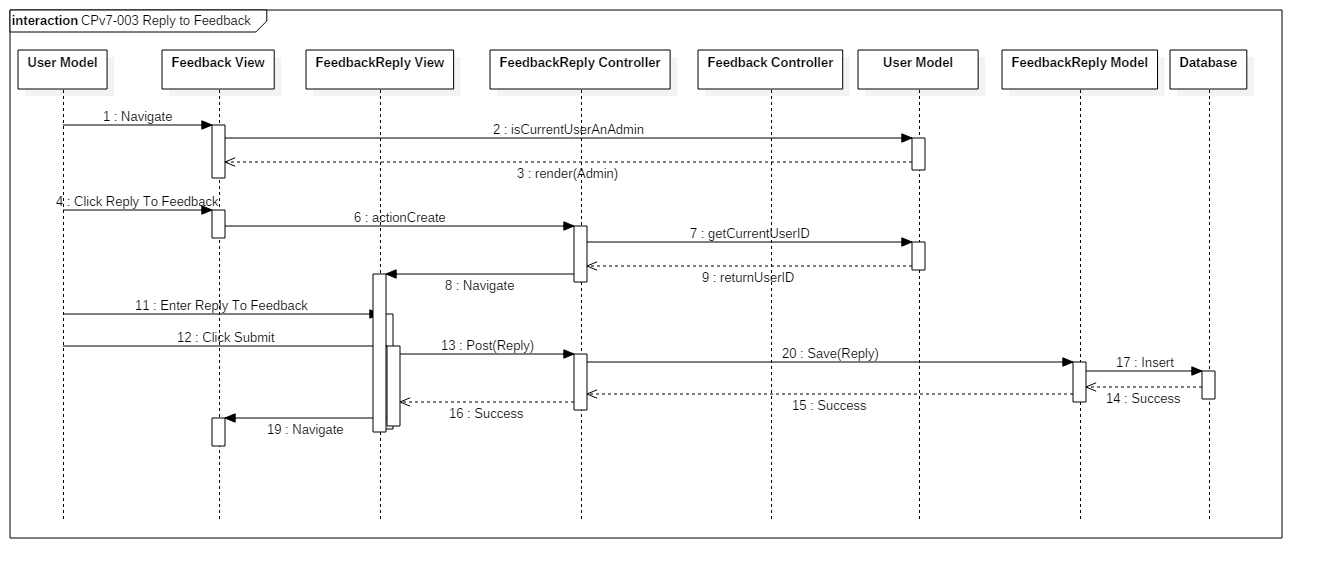
**Collaborative tools**



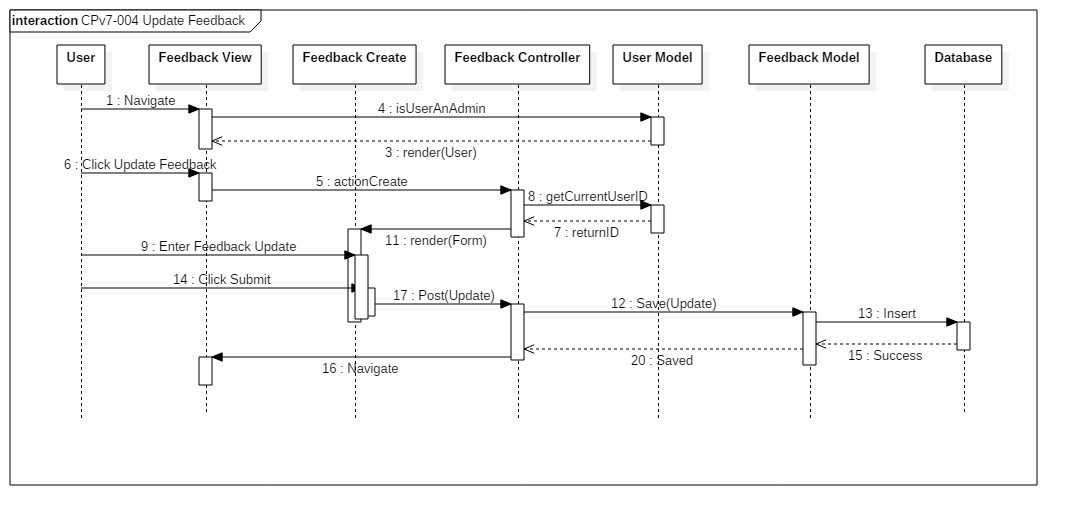
CPv7-001 Create Feedback Sequence Diagram



CPv7-002 Delete Feedback Sequence Diagram

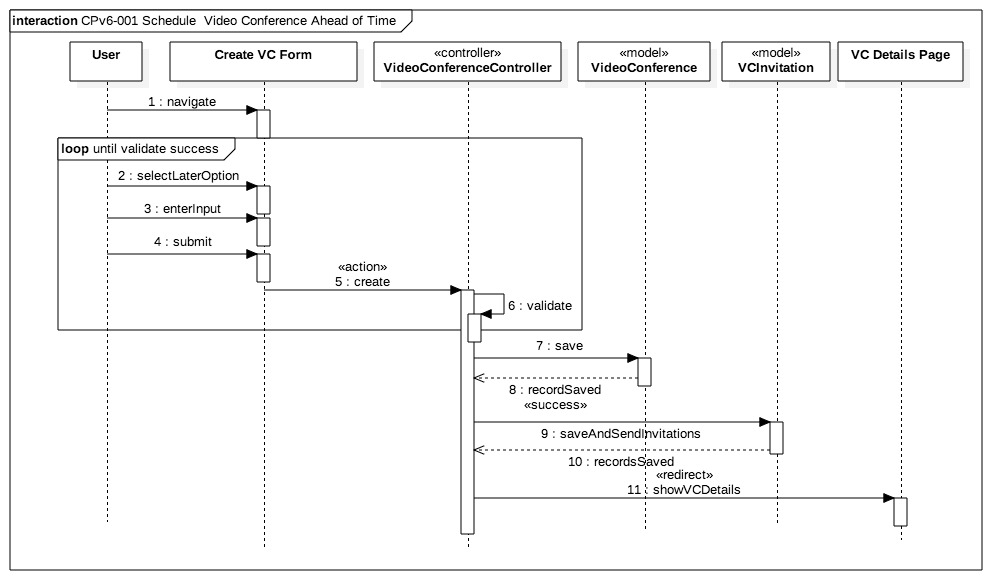


CPv7-003 Reply to Feedback Sequence Diagram

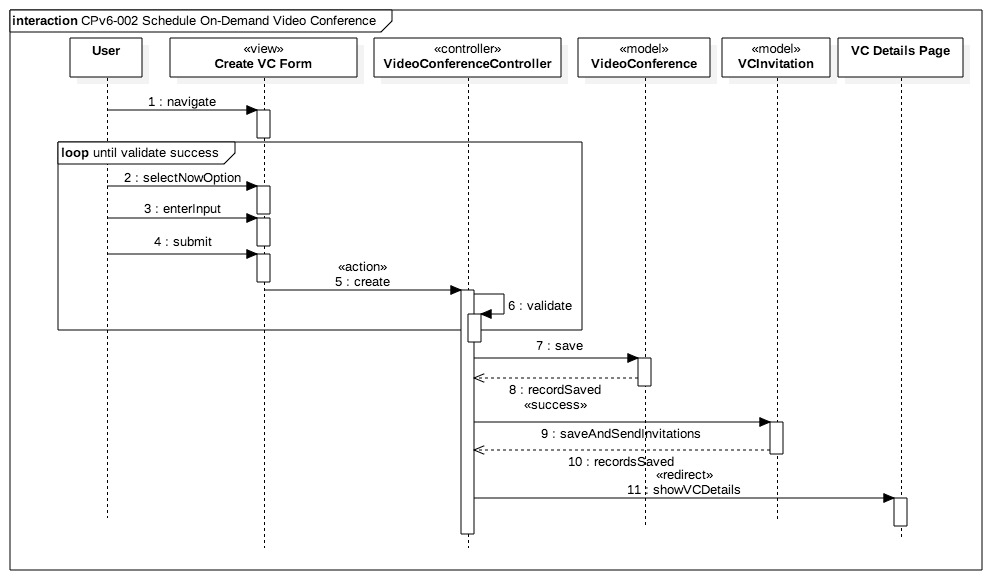


CPv7-004 Update Feedback Sequence Diagram

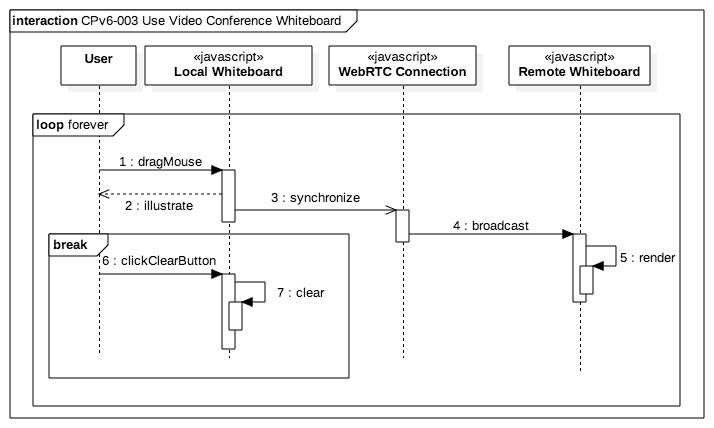
Previous Versions Dynamic Models



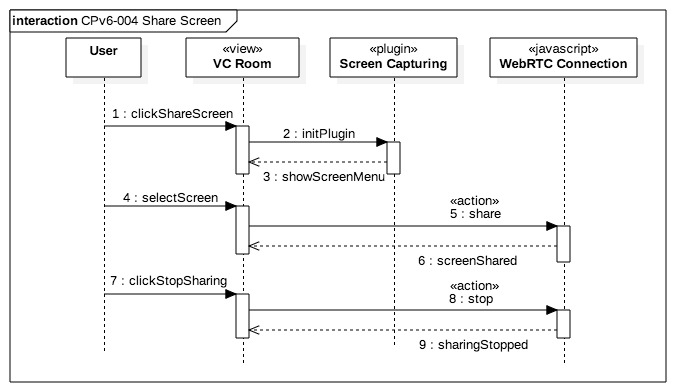
AD.01 Schedule Video Conference Ahead of Time Sequence Diagram



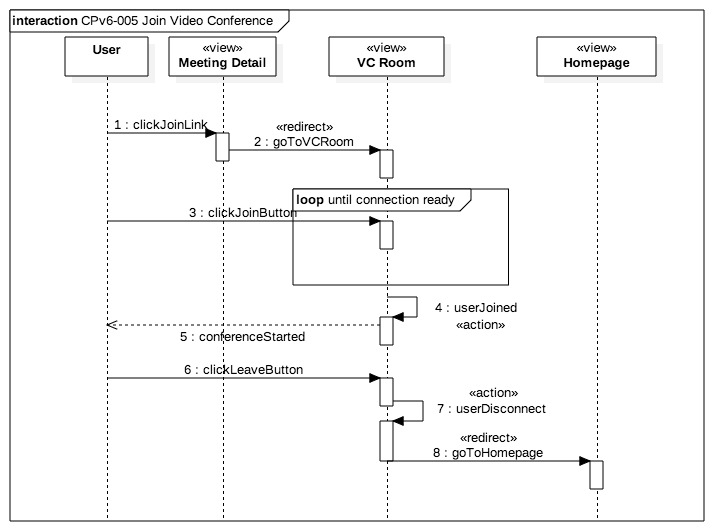
AD.02 Schedule On-Demand Video Conference Sequence Diagram

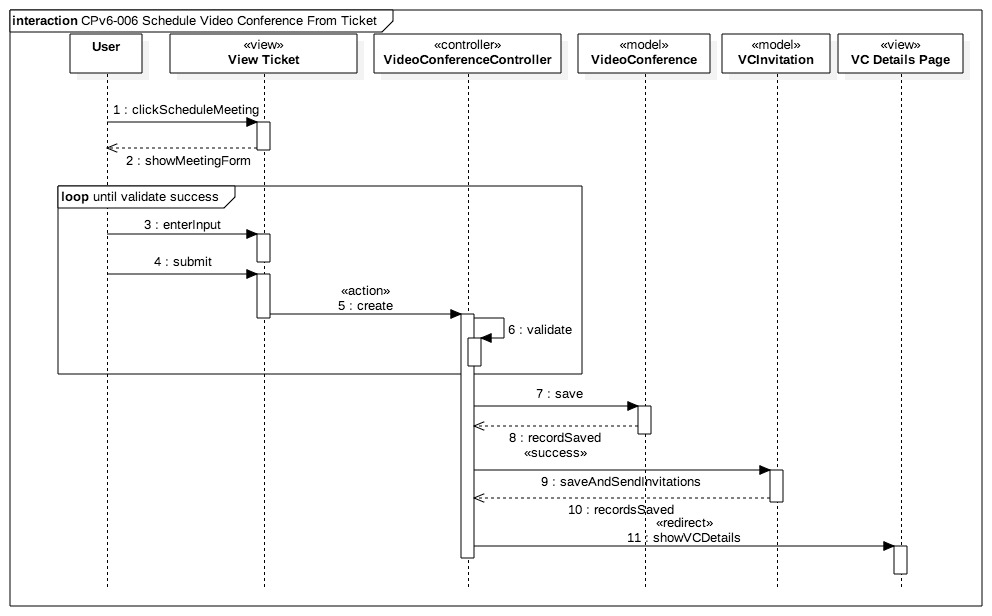


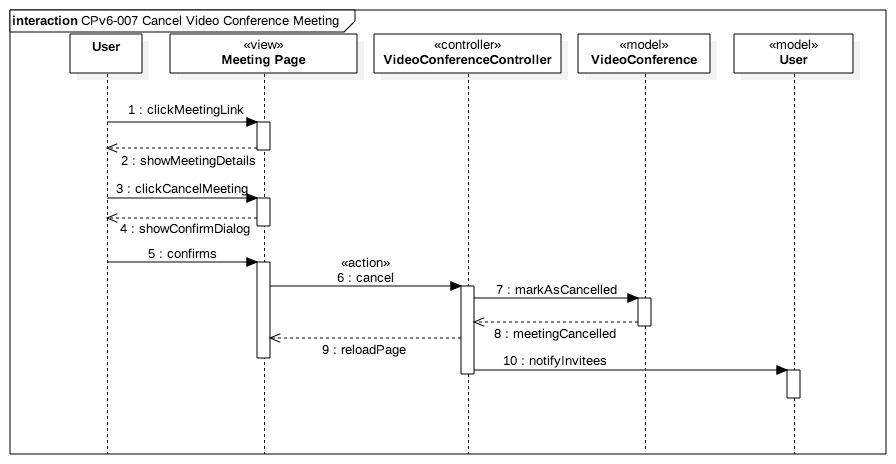
AD.-03 Use Video Conference Whiteboard Sequence Diagram



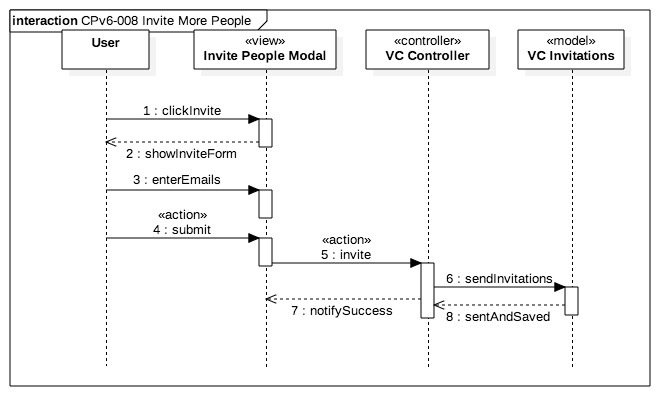
AD.04 Share Screen Sequence Diagram

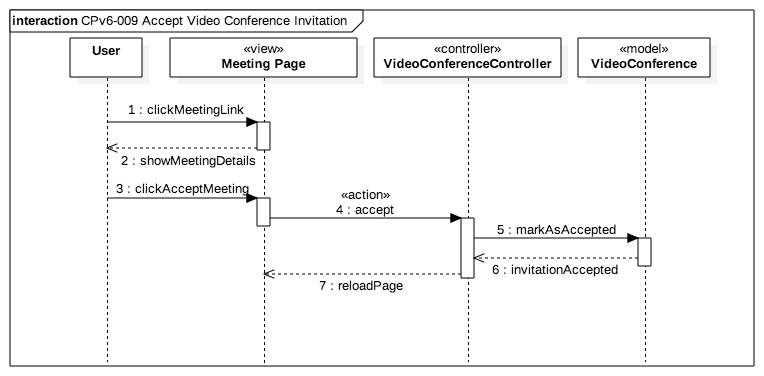
AD.05 Join Video Conference Sequence Diagram

AD.06 Schedule Video Conference from Ticket Sequence Diagram

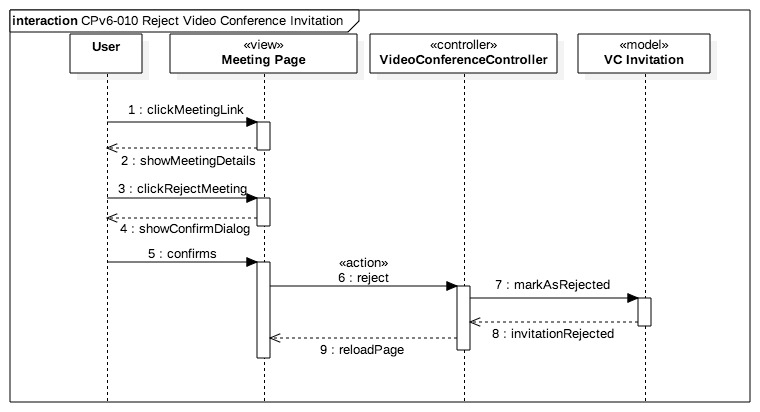


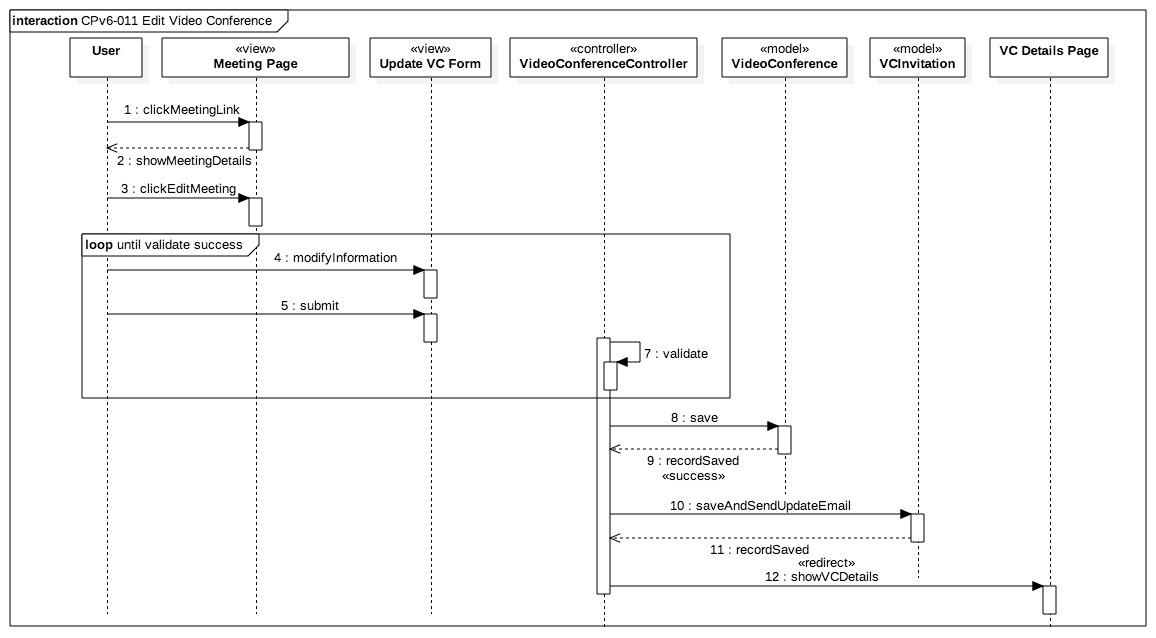
AD.07 Cancel Video Conference Meeting Sequence Diagram

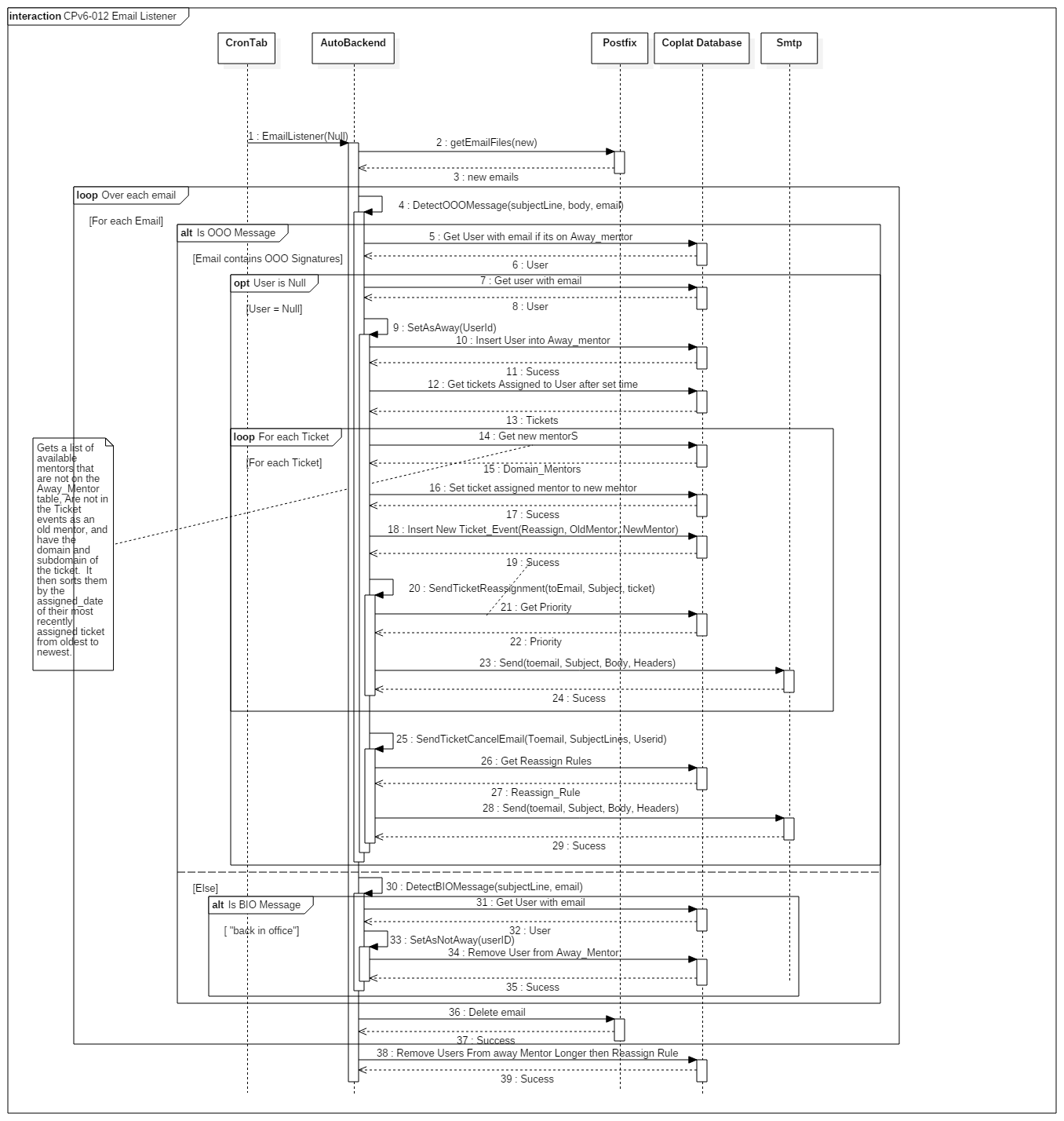
AD.08 Invite More People Sequence Diagram

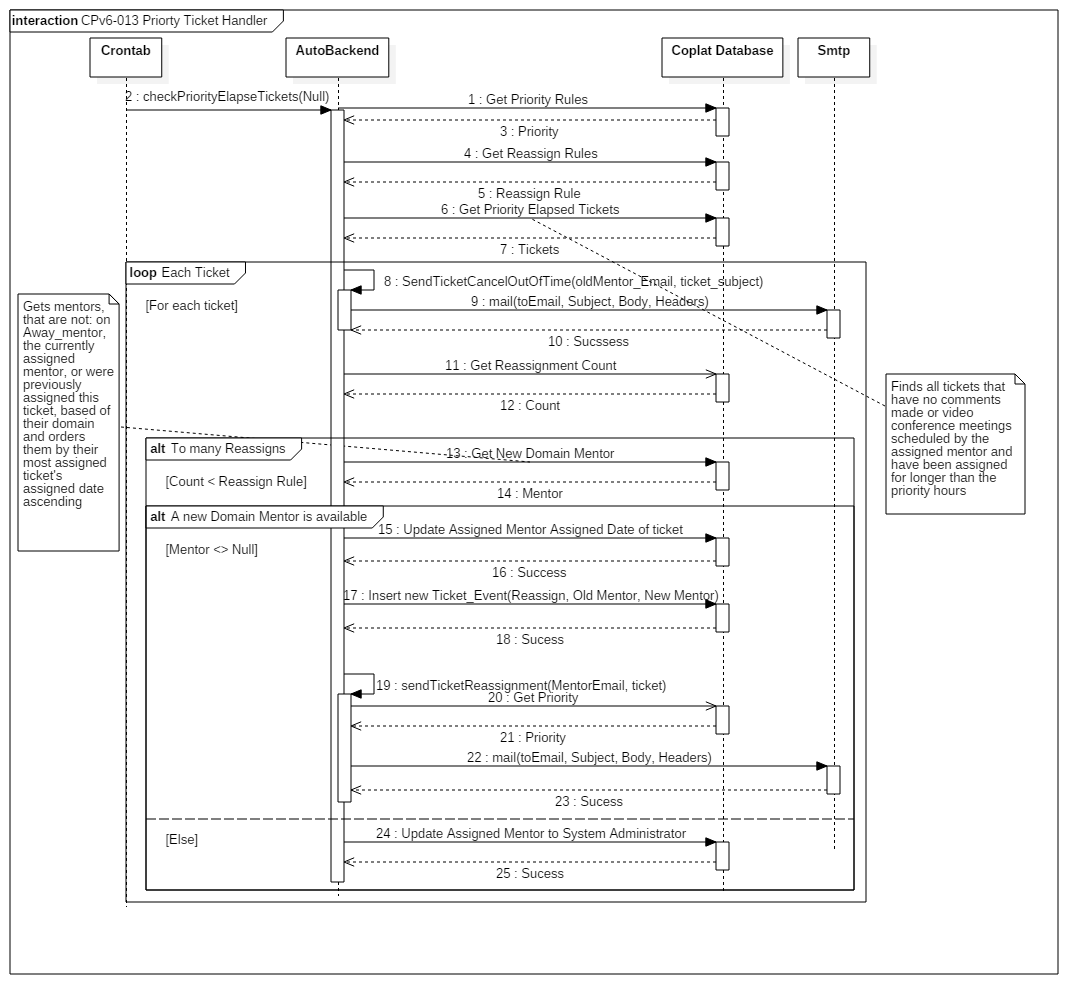


AD.09 Accept Video Conference Invitation Sequence Diagram

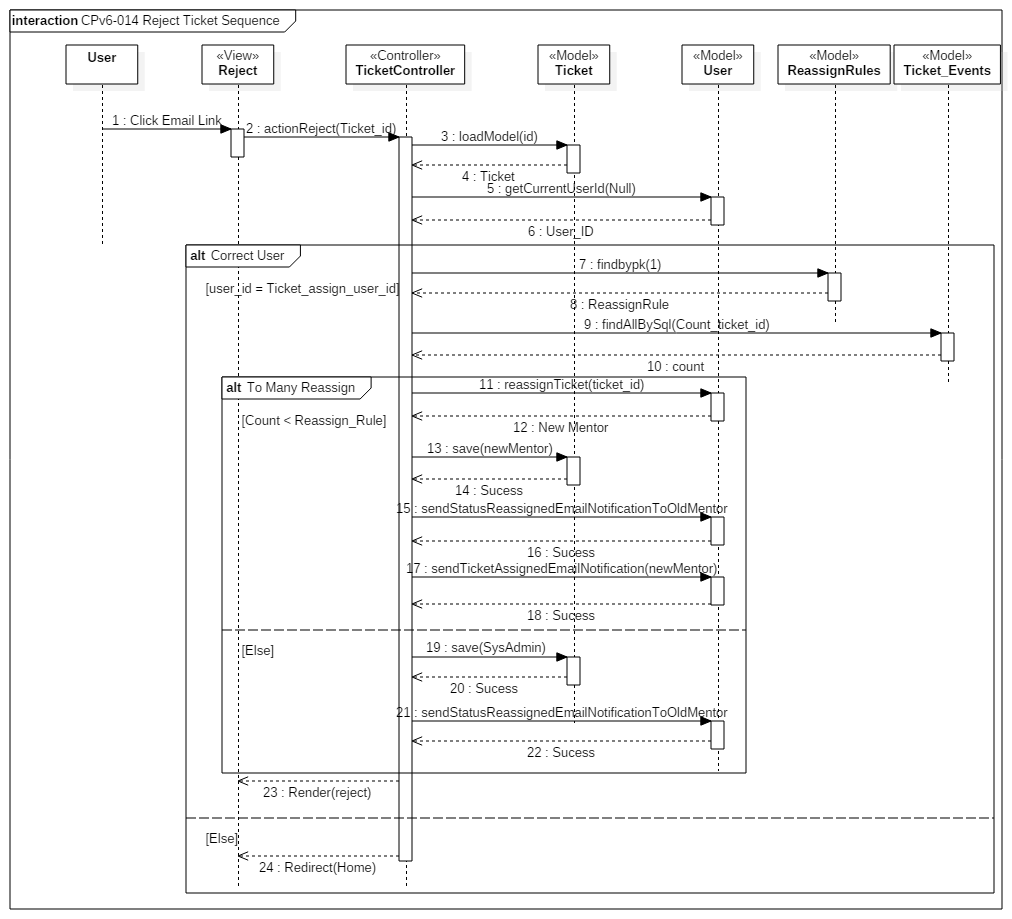
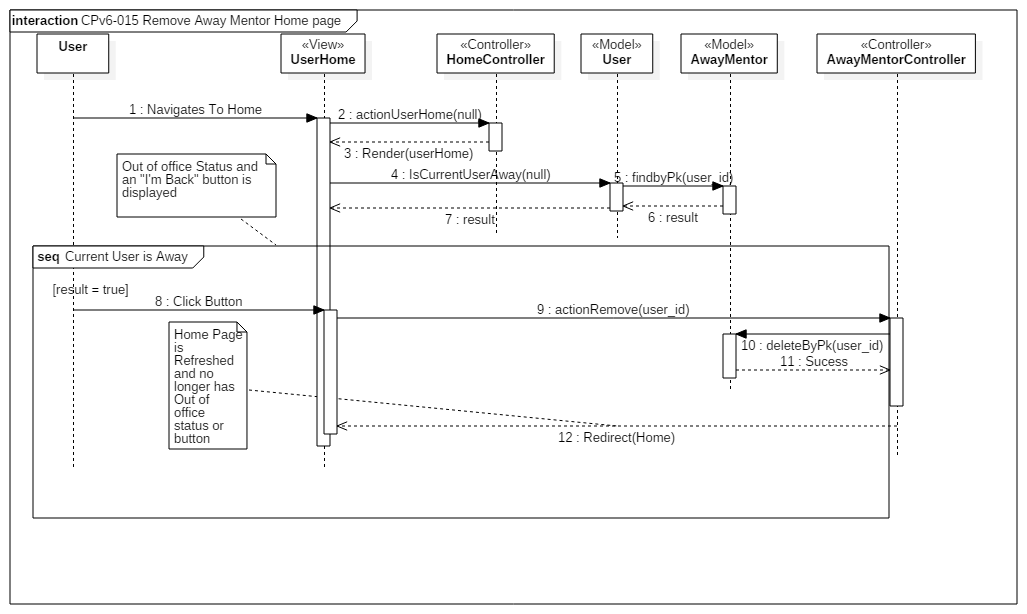
AD.10 Reject Video Conference Invitation Sequence Diagram

AD.11 Edit Video Conference Sequence Diagram

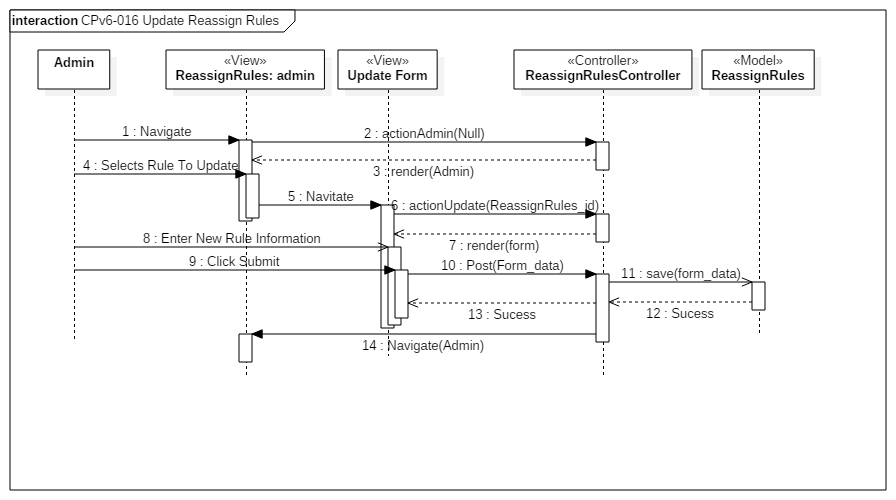
AD.12 Email Listener Sequence Diagram

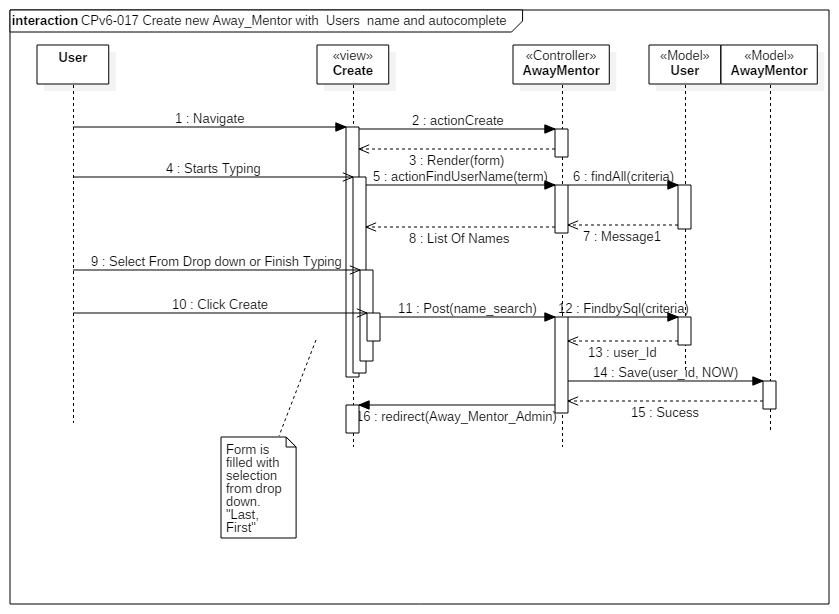


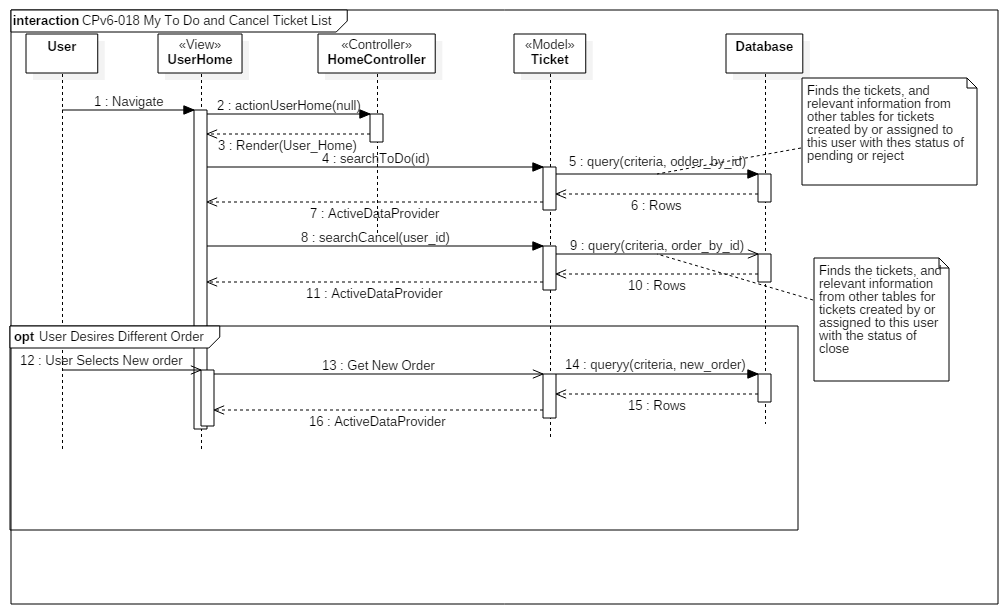
AD.13 Priority Ticket Handler Sequence Diagram

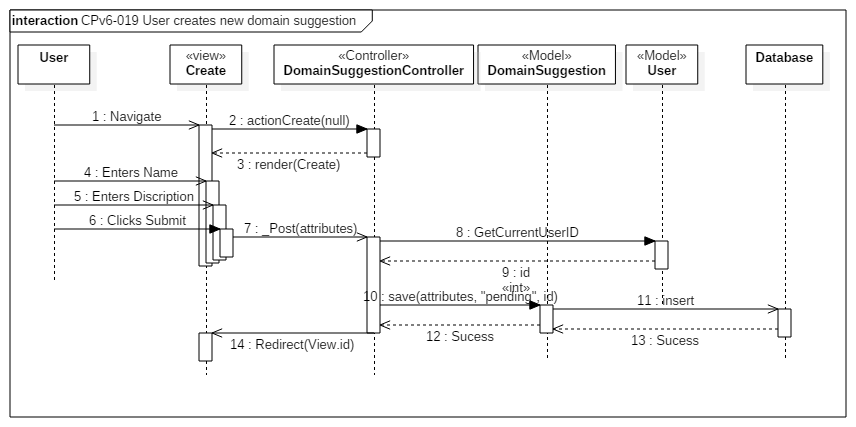
AD.14 Reject Ticket Sequence Diagram

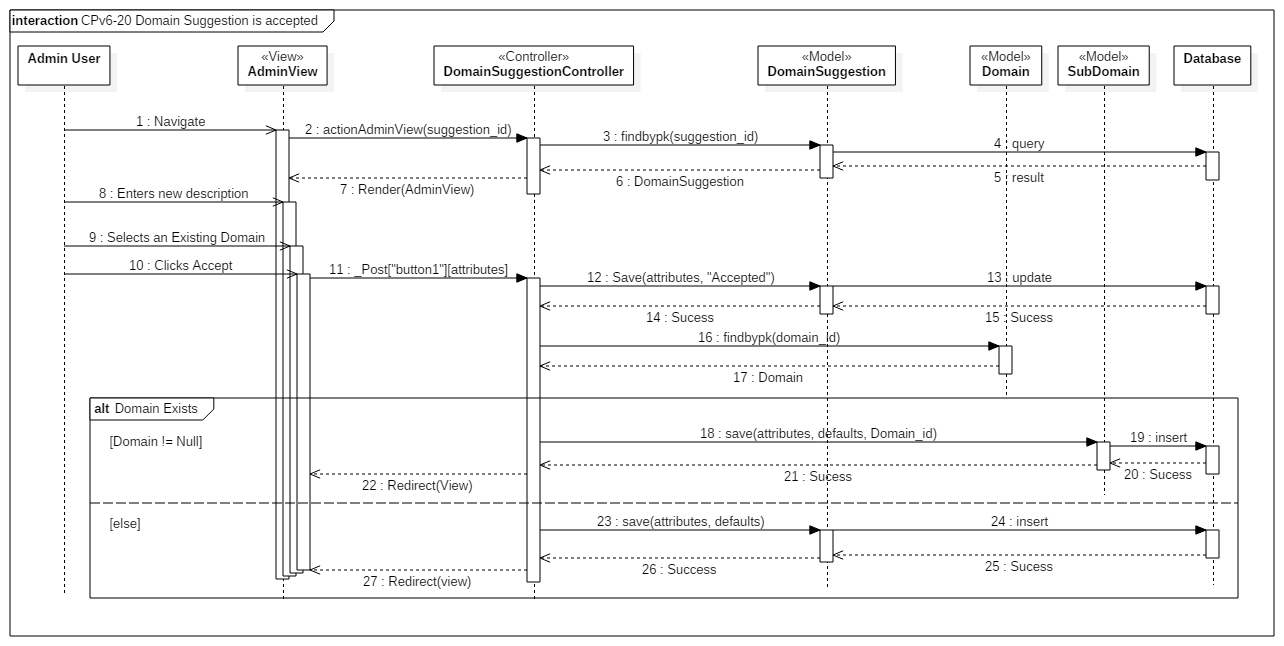
AD.15 Remove Away Mentor Home page Sequence Diagram

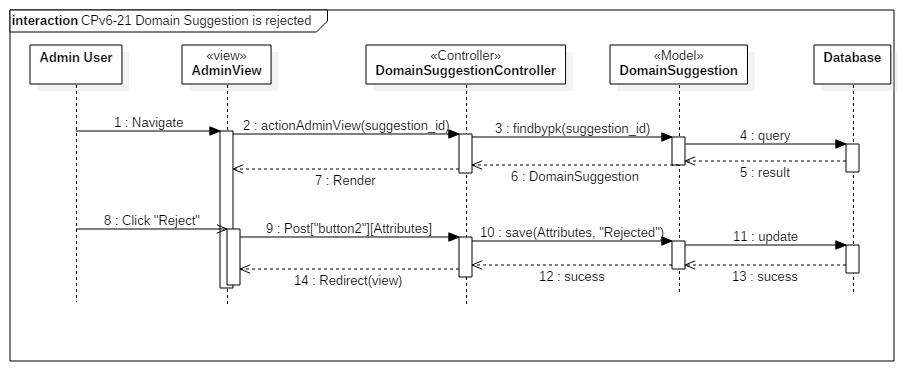
AD.16 Update Reassign Rules Sequence Diagram

AD.17 Create new Away\_Mentor with Users name and autocomplete Sequence Diagram

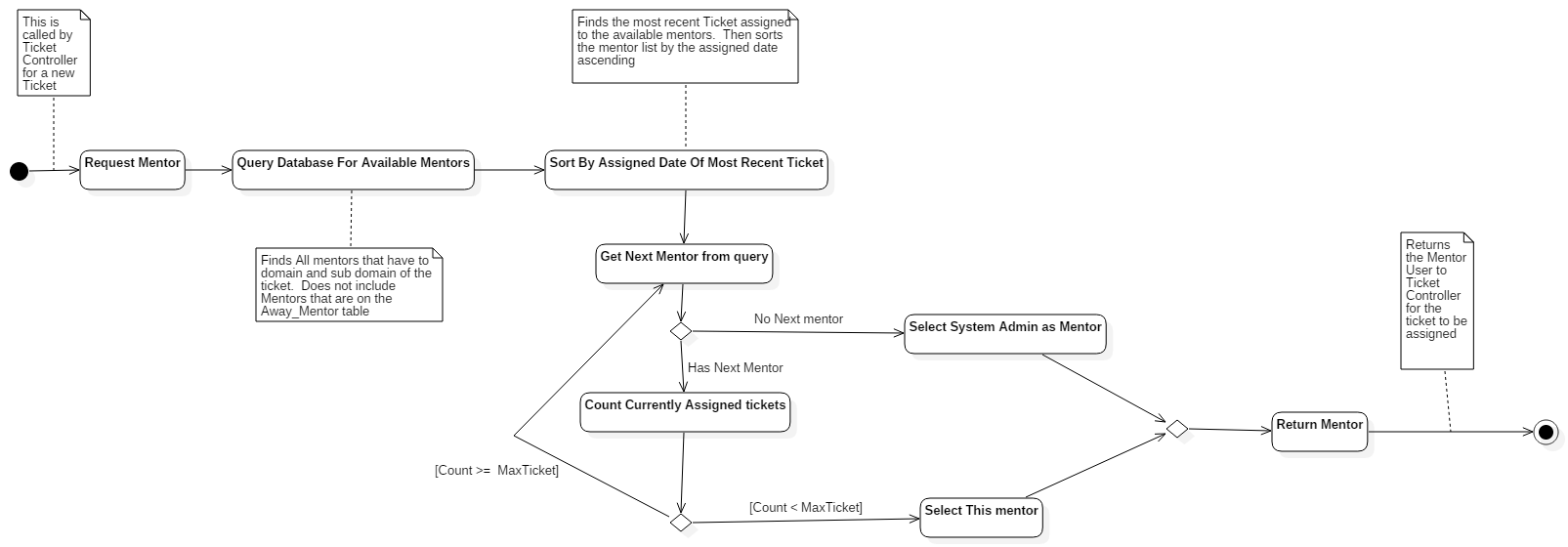
AD.18 My To Do and Cancel Ticket List Sequence Diagram

AD.19 User creates new domain suggestion Sequence Diagram

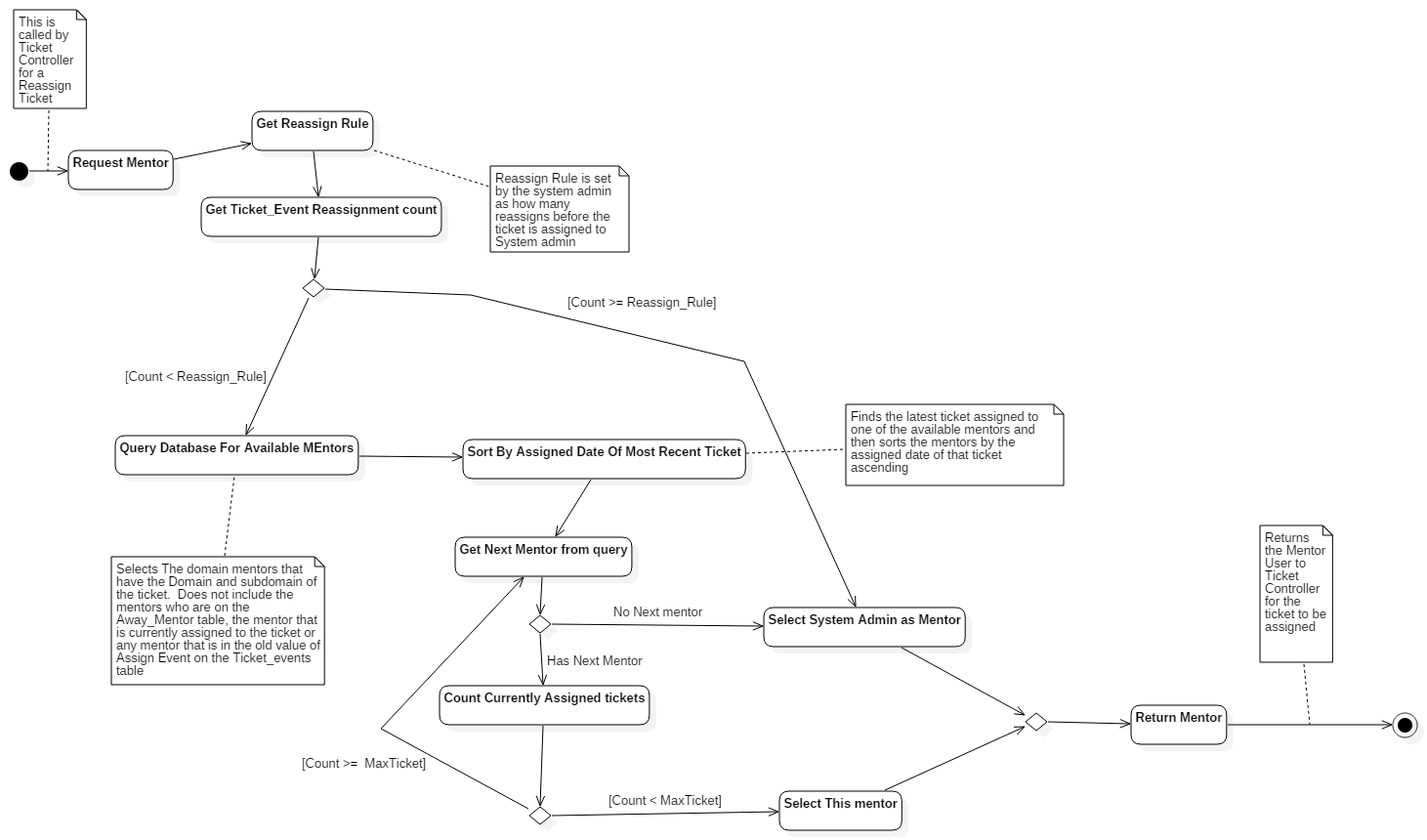
AD.20 Domain Suggestion is accepted Sequence Diagram

AD.21 Domain Suggestion is rejected Sequence Diagram

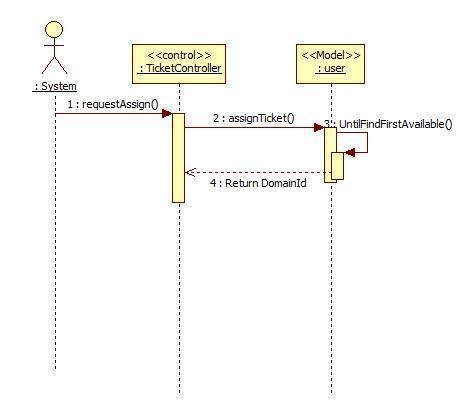
New Process when selecting a mentor for a new ticket:

AD.22 New Process when selecting a mentor for a new ticket

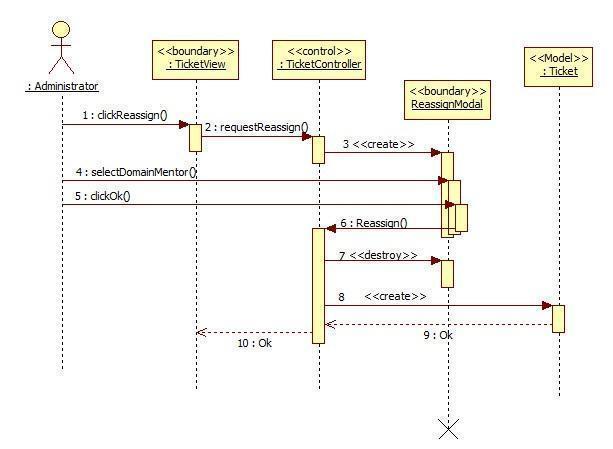
New Process when selecting a mentor for a ticket that’s being reassigned



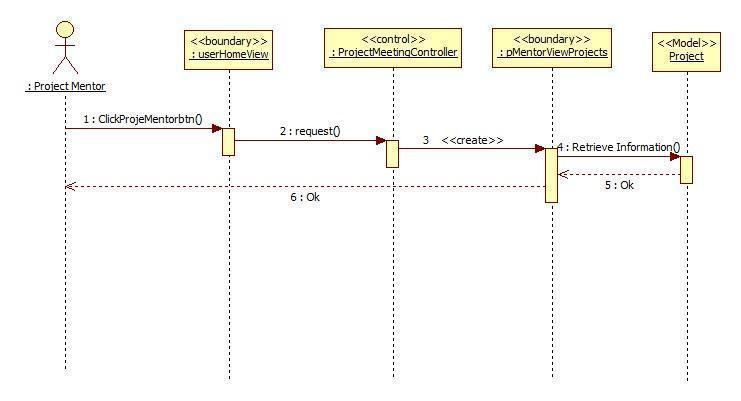
AD.23 New Process when selecting a mentor for a ticket that’s being reassigned



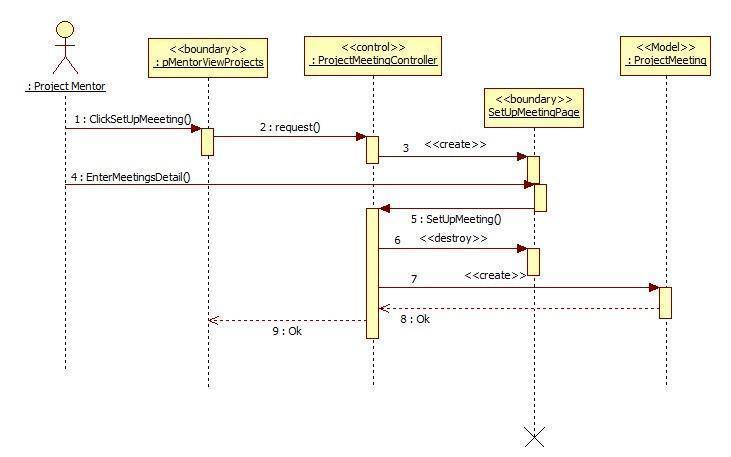
AD.24 Auto Assign Ticket



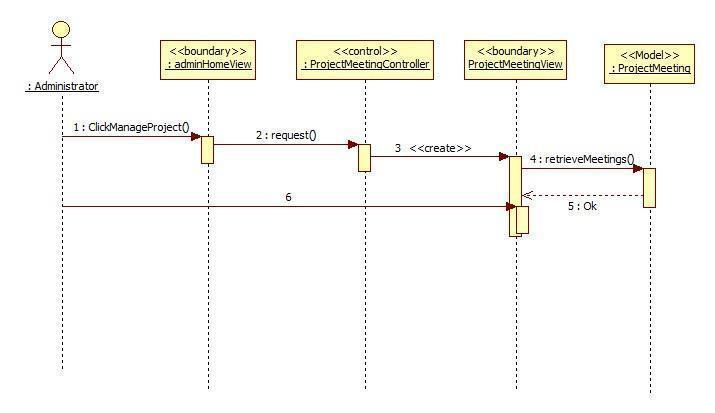
AD.25 MP-Reassign Ticket



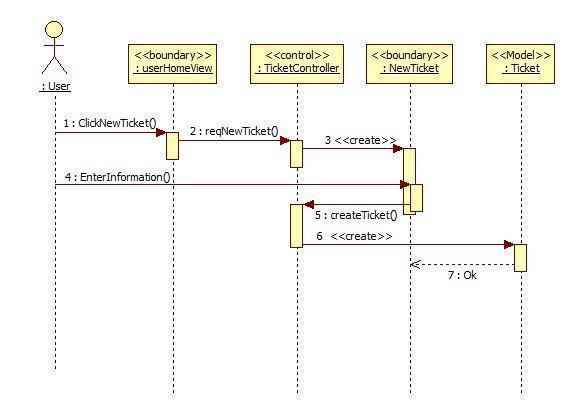
AD.26 MP-Retrieve the project information



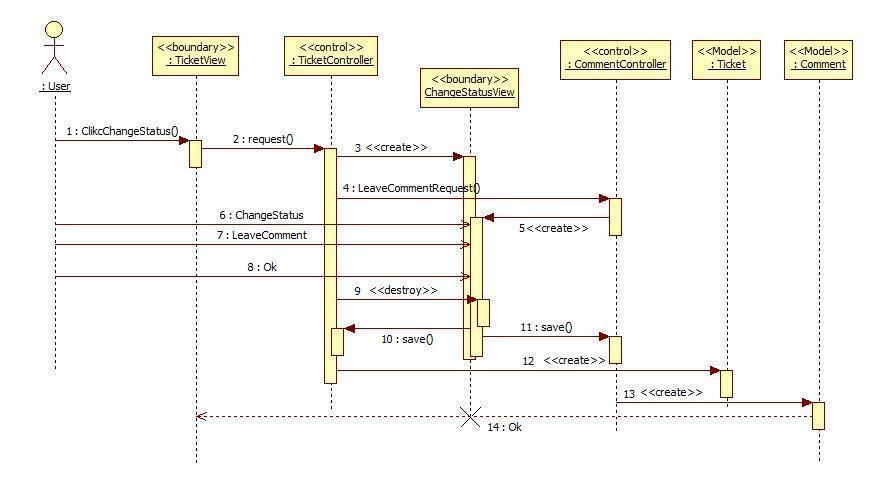
AD.27 MP-Setup Meetings



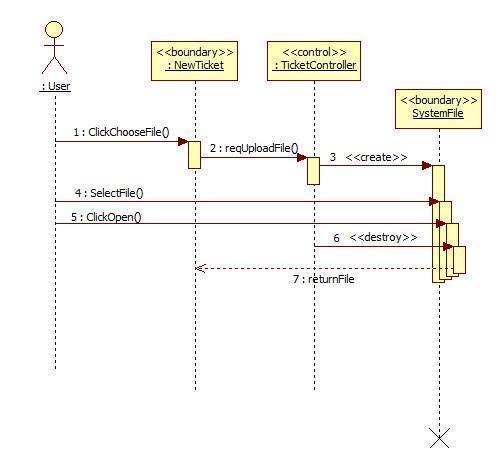
AD.28 MP-Retrieve all the upcoming meetings



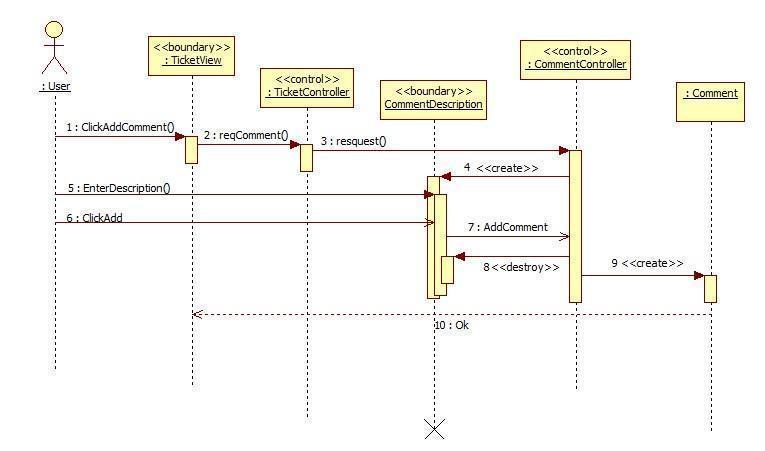
AD.29 MP-Reassign Ticket



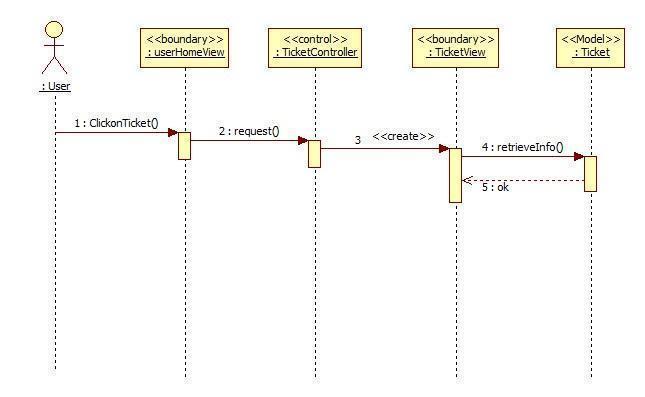
AD.30 MP-Change Status



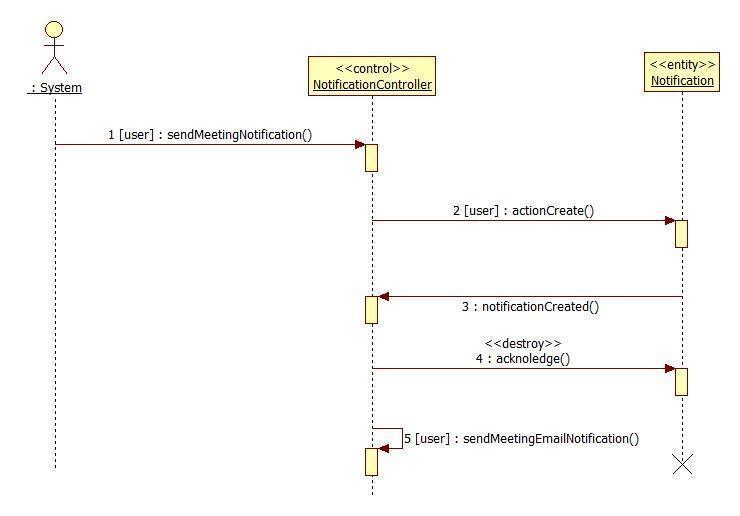
AD.31 MP-Upload File



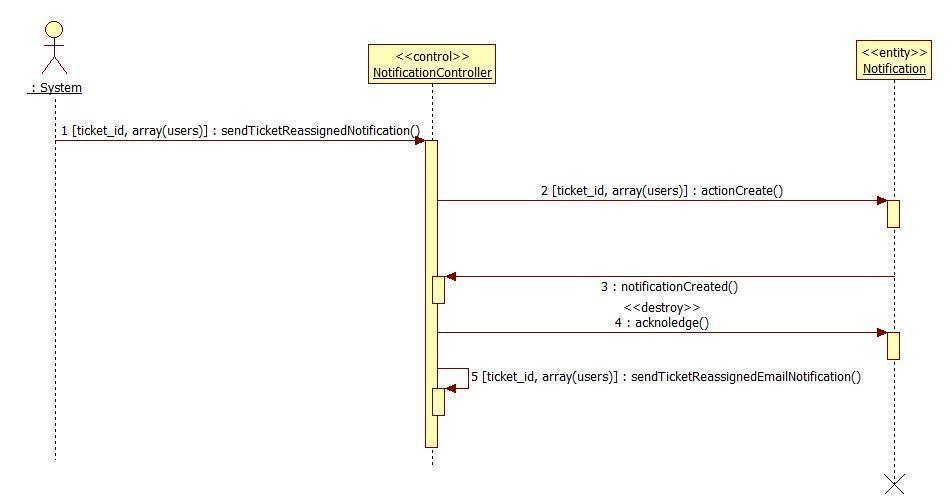
AD.32 MP-Append Comment



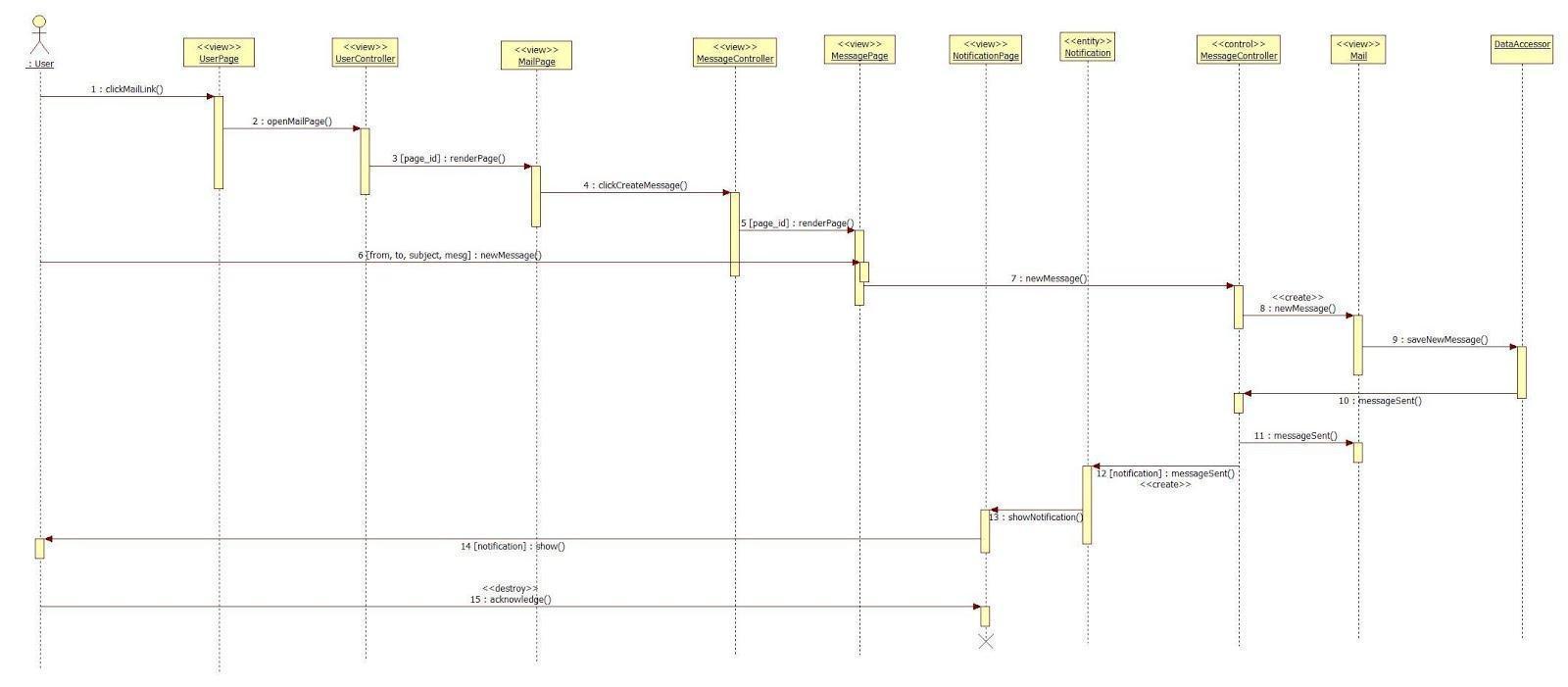
AD.33 MP-Retrieve the ticket detail



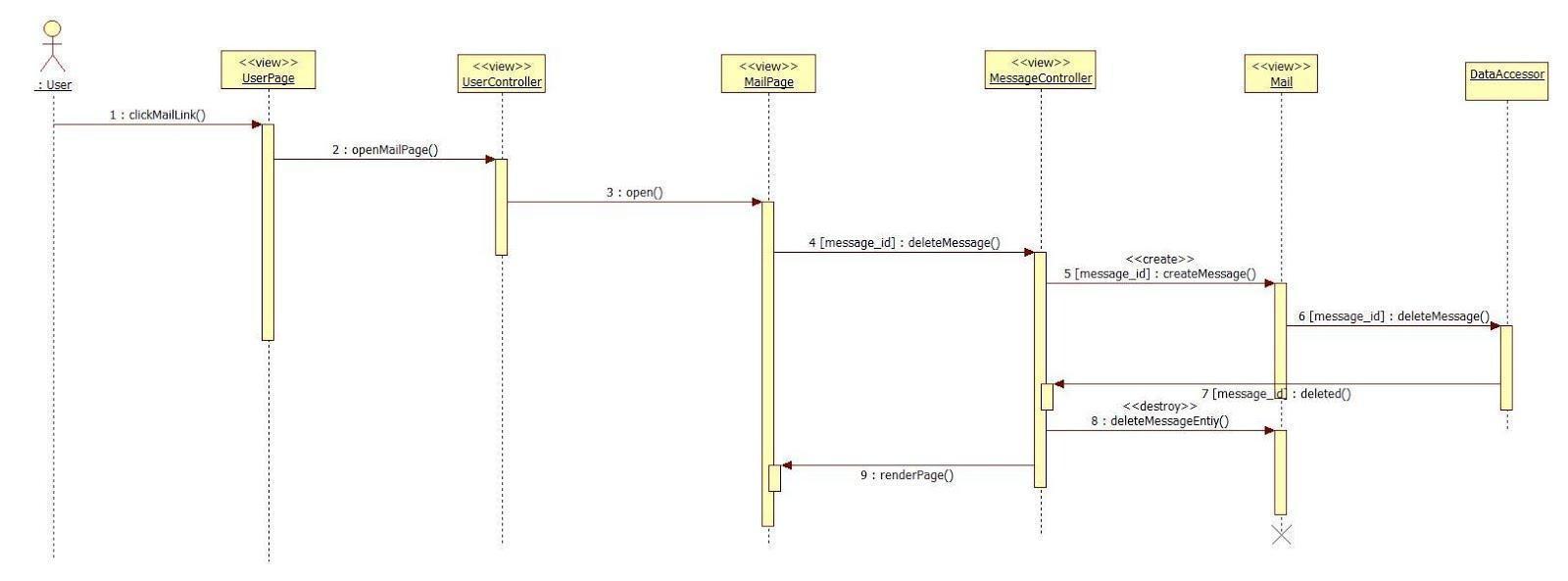
AD.34 New Meeting Notification



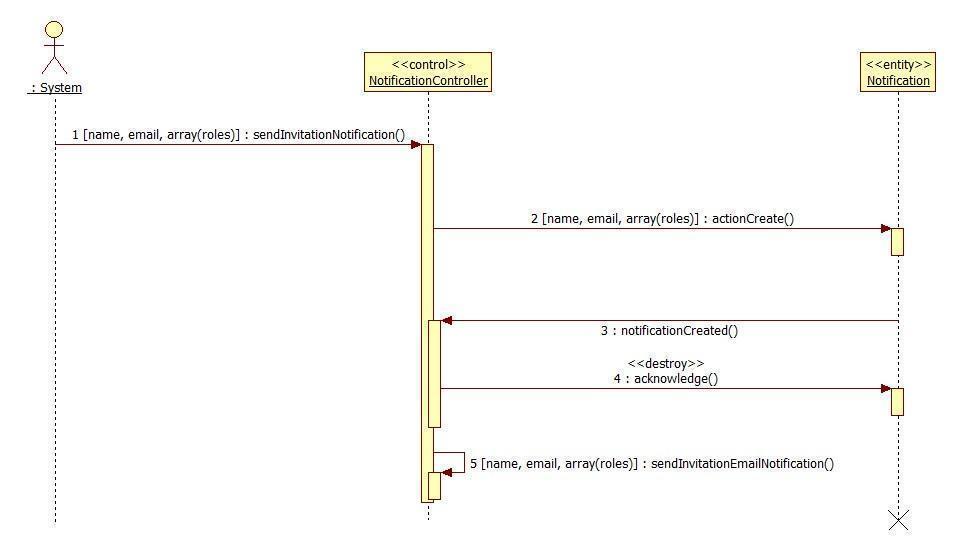
AD.35 Ticket Reassigned Notification



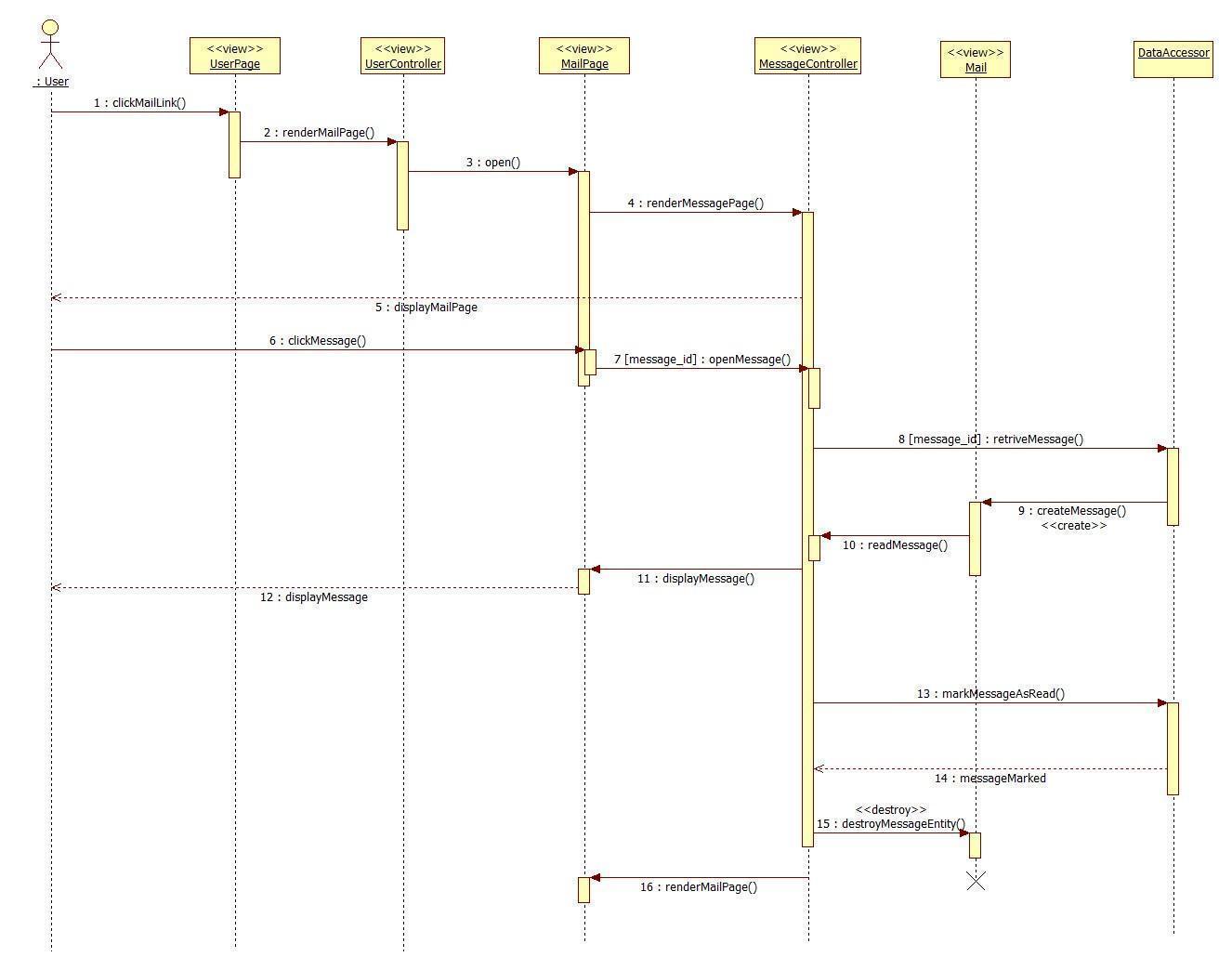
AD.36 Send New Message



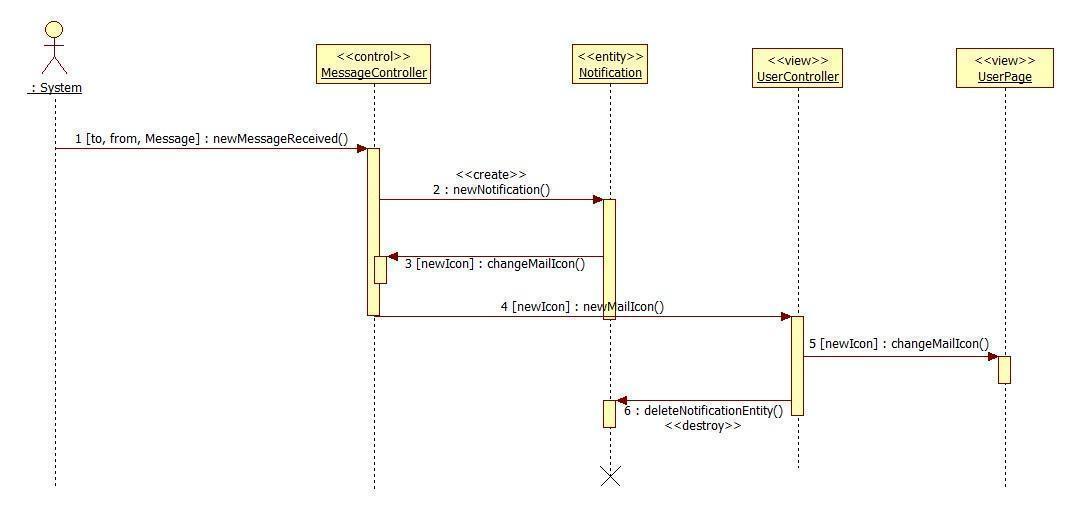
AD.37 Delete Message



AD.38 Invitation Notification



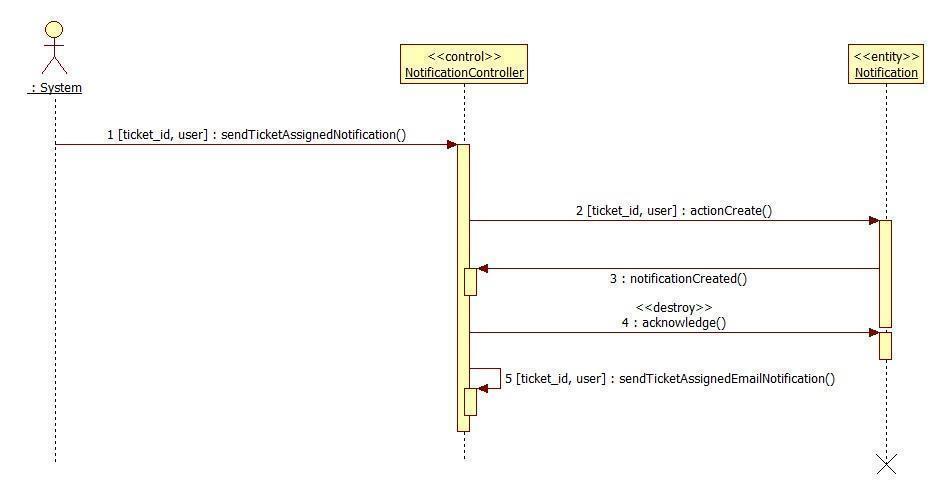
AD.39 Read Message



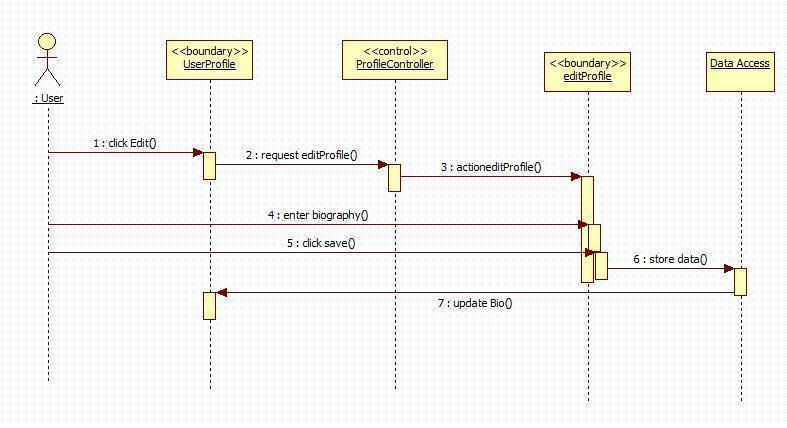
AD.40 New Message Notification



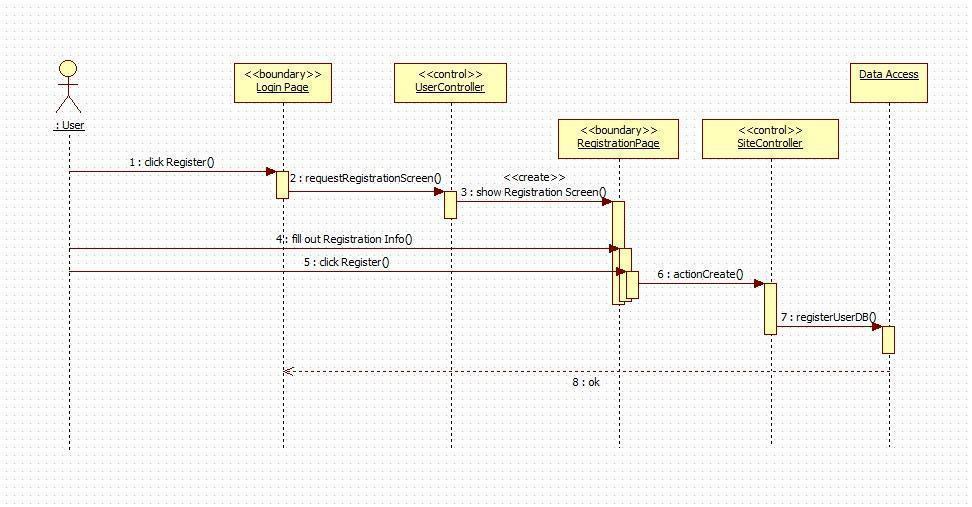
AD.41 Comment Added Notification



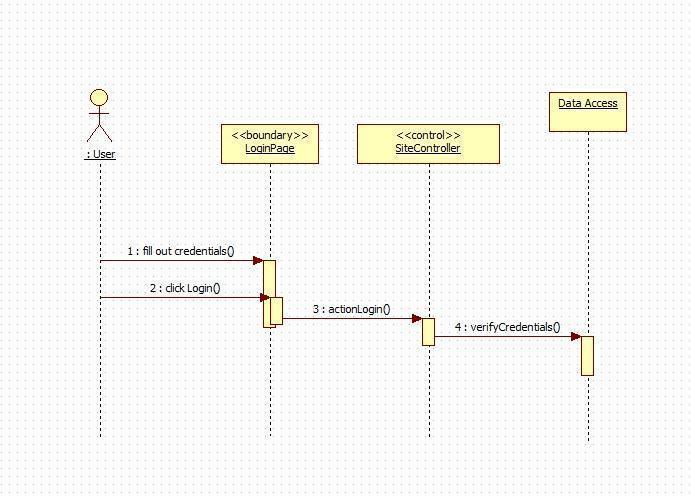
AD.42 Ticket Assigned Notification

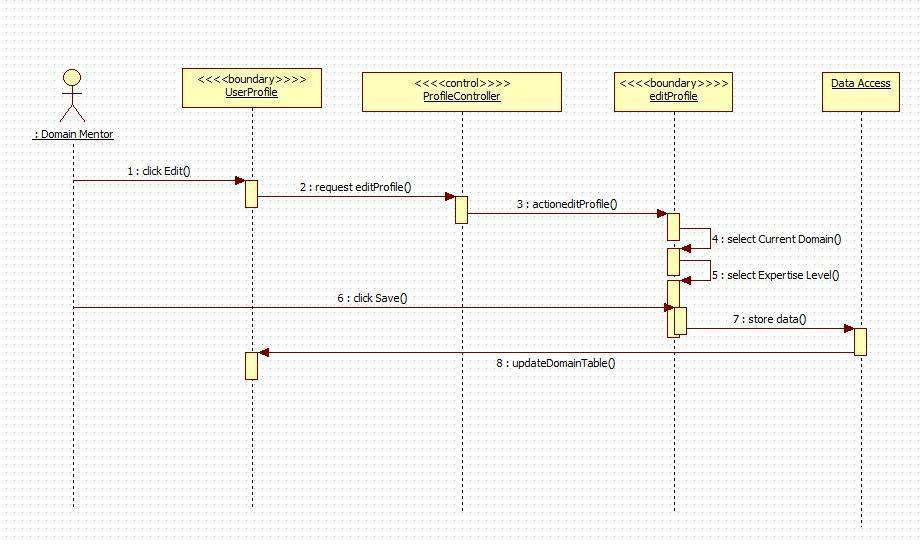


AD.43 Edit Biography

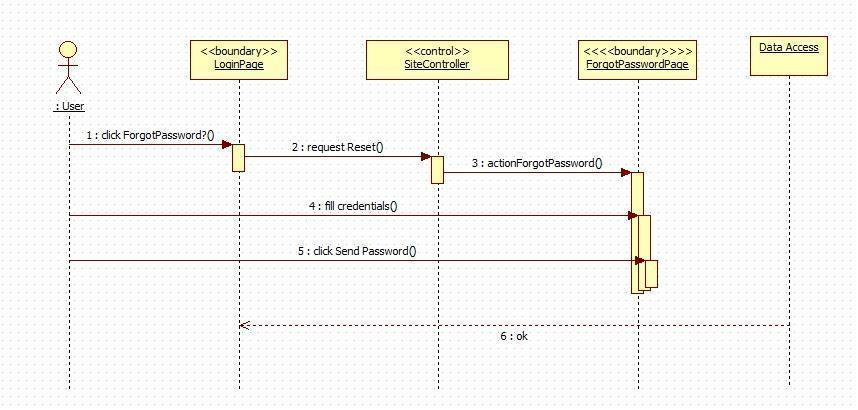


AD.44 Register

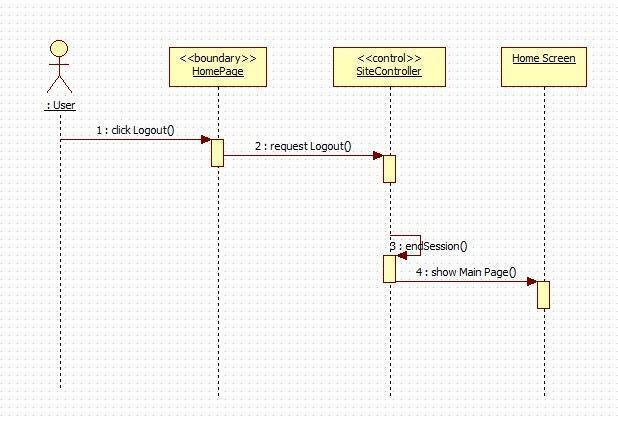


AD.45 Login

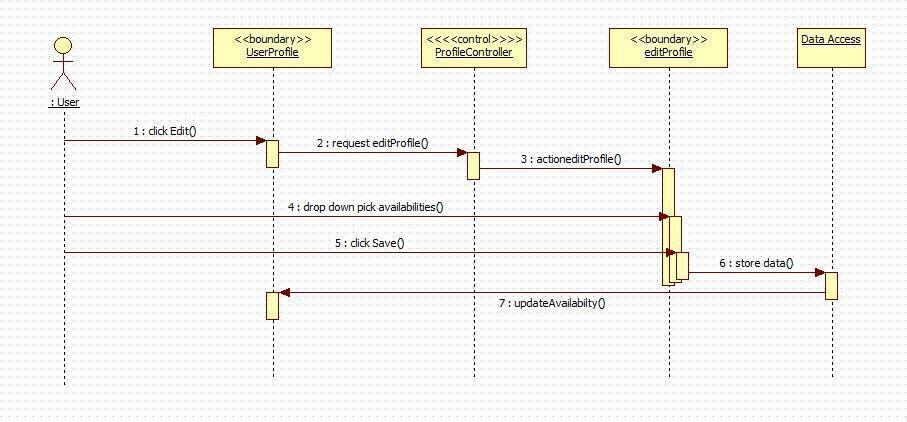
AD.46 Add Domain



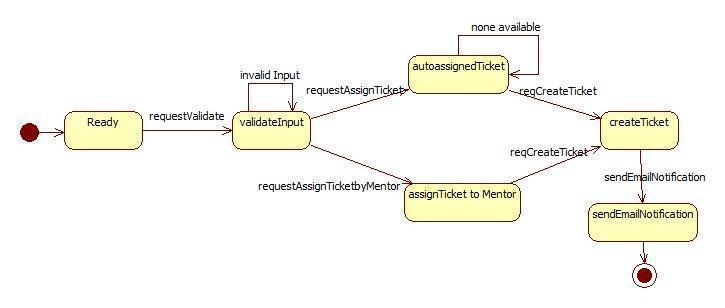
AD.47 Forgot Password



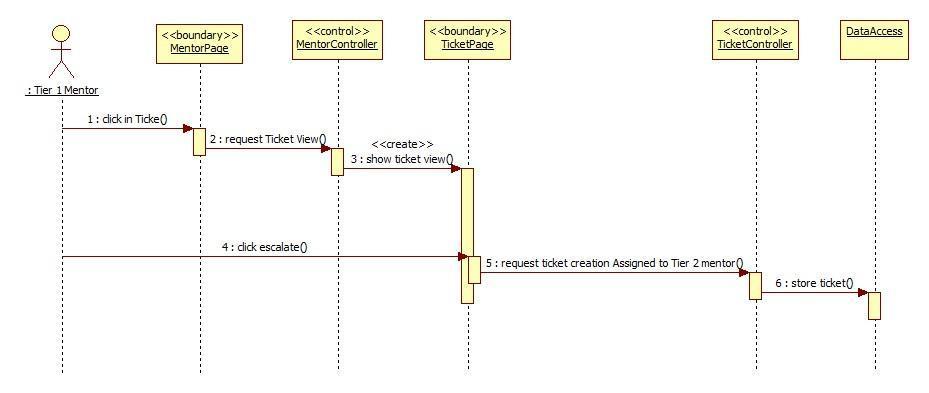
AD.48 Logout



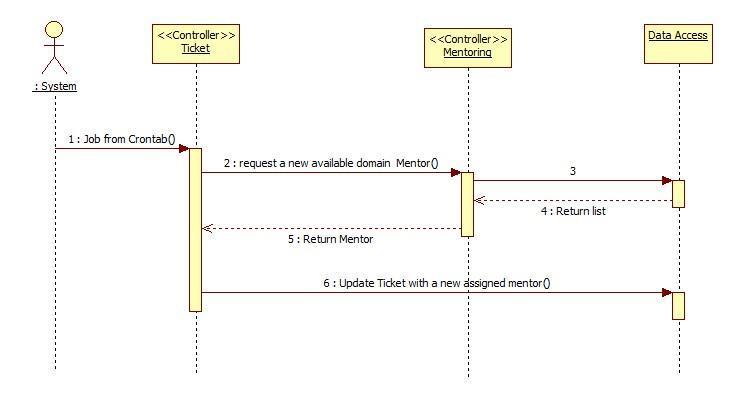
AD.49 Edit Availability



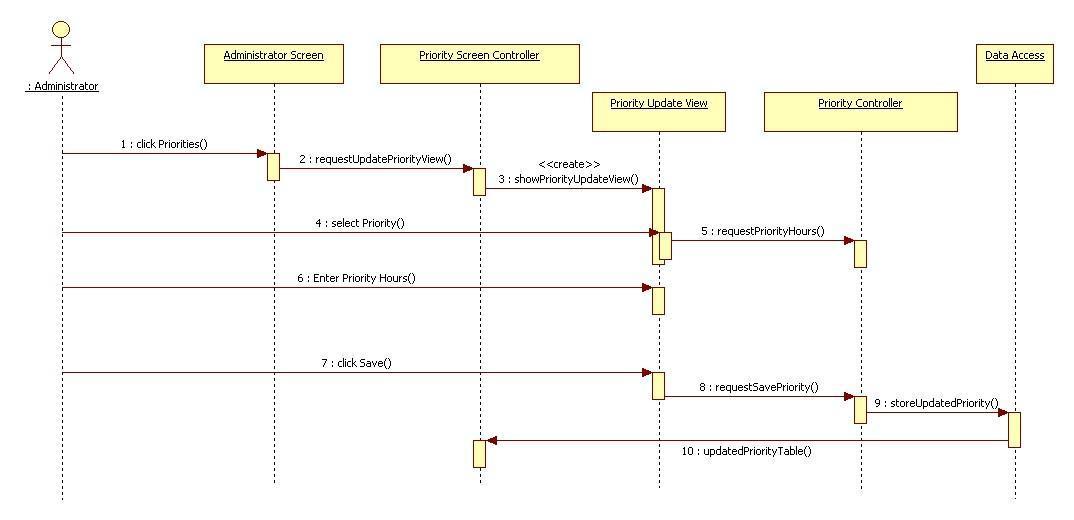
AD.50 Create Ticket



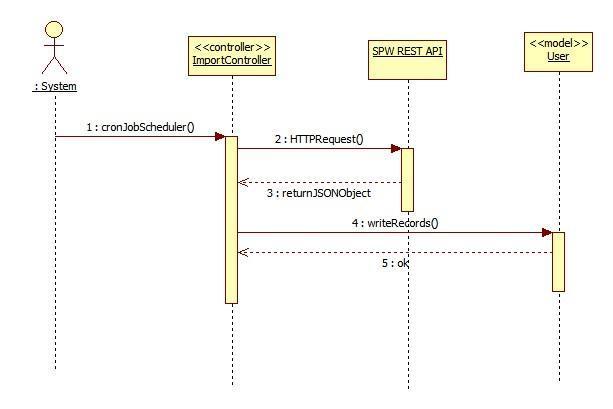
AD.51 MP- Escalate Ticket



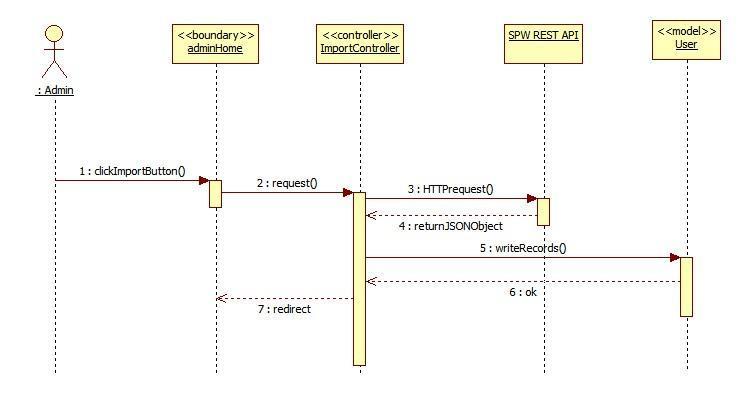
AD.52 MP-Automatically reassign ticket

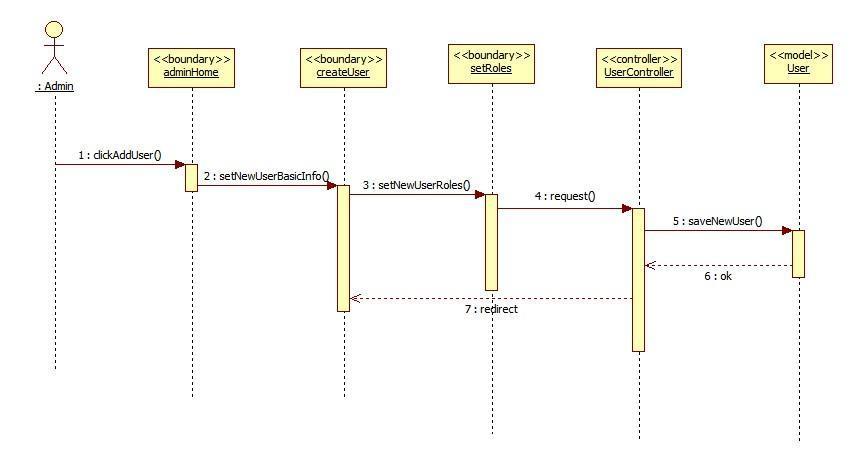


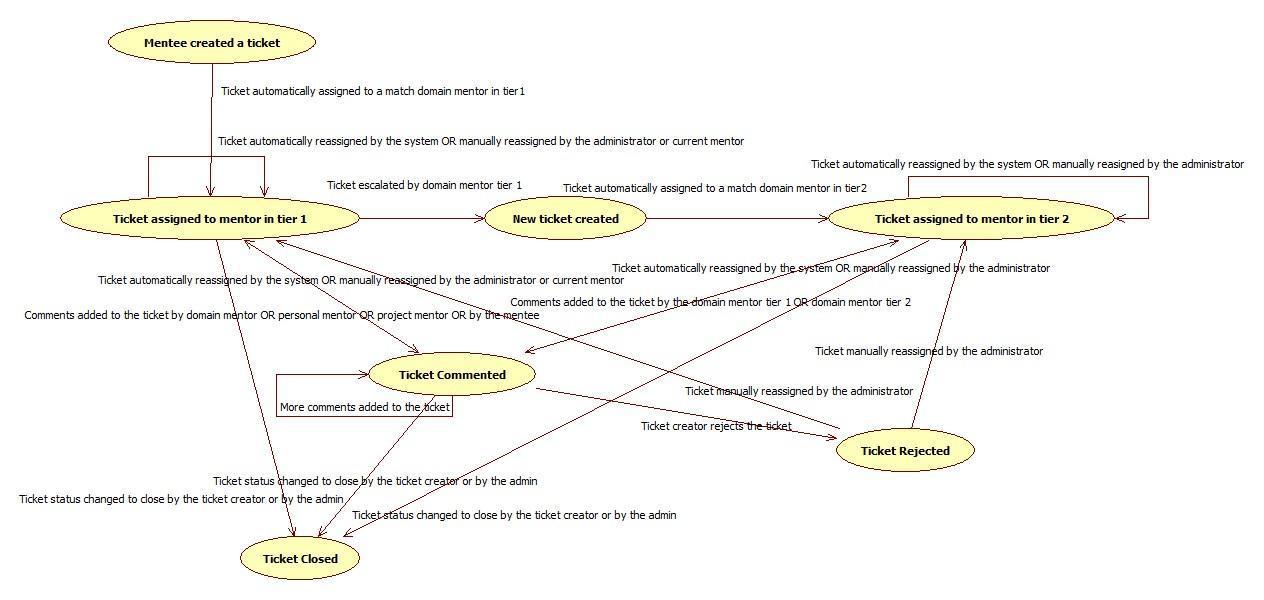
AD.53 MP – Edit Priority



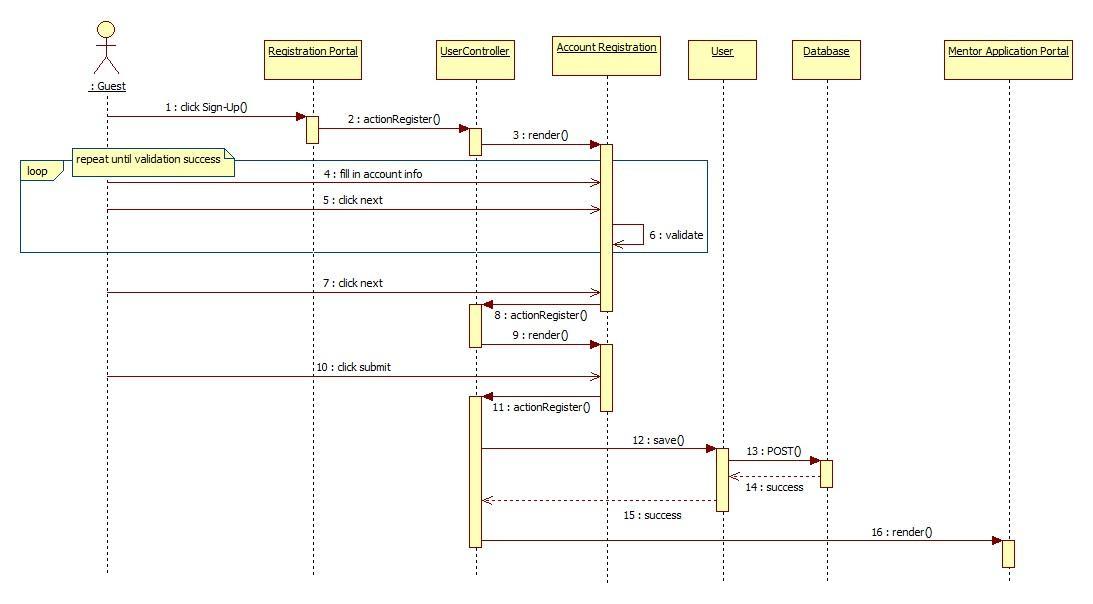
AD.54 MP-Import data from SPW automatically

AD.55 MP-Import data from SPW Manually

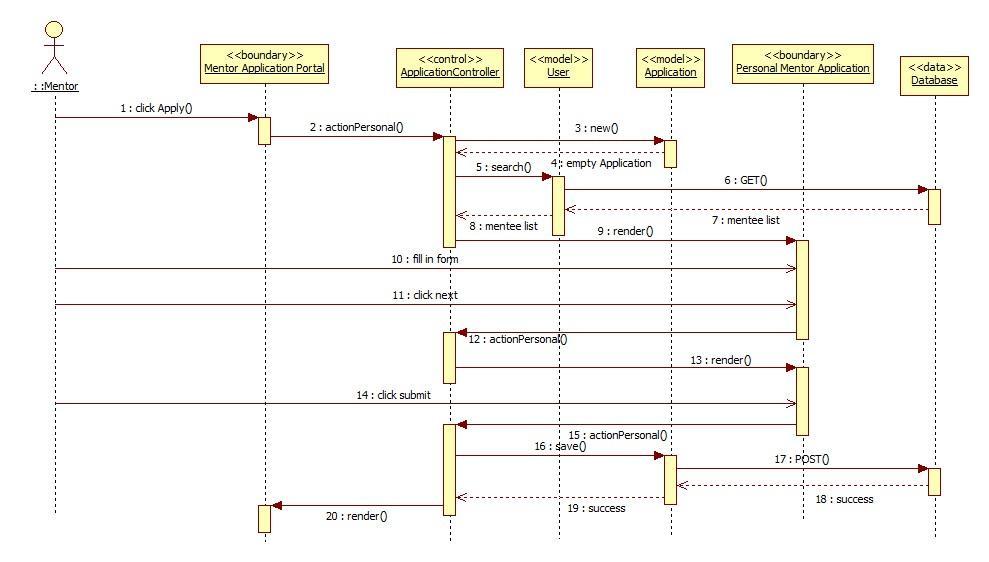
AD.56 MP-Allow the admin to register mentors

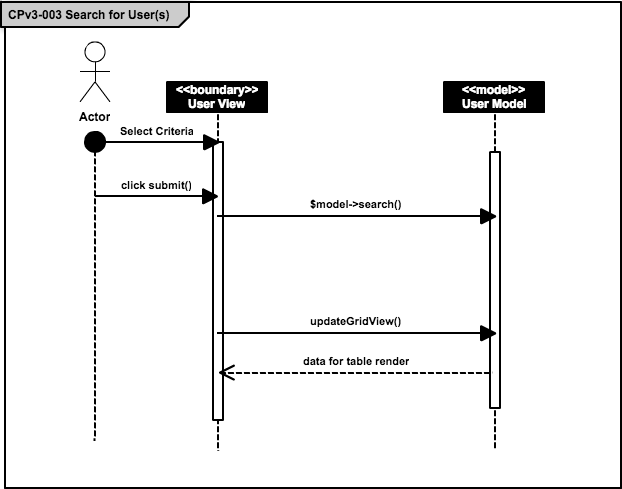
AD.57 Workflow (Ticket)

**Mentoring Module: Version 3 - Sequence Diagrams**

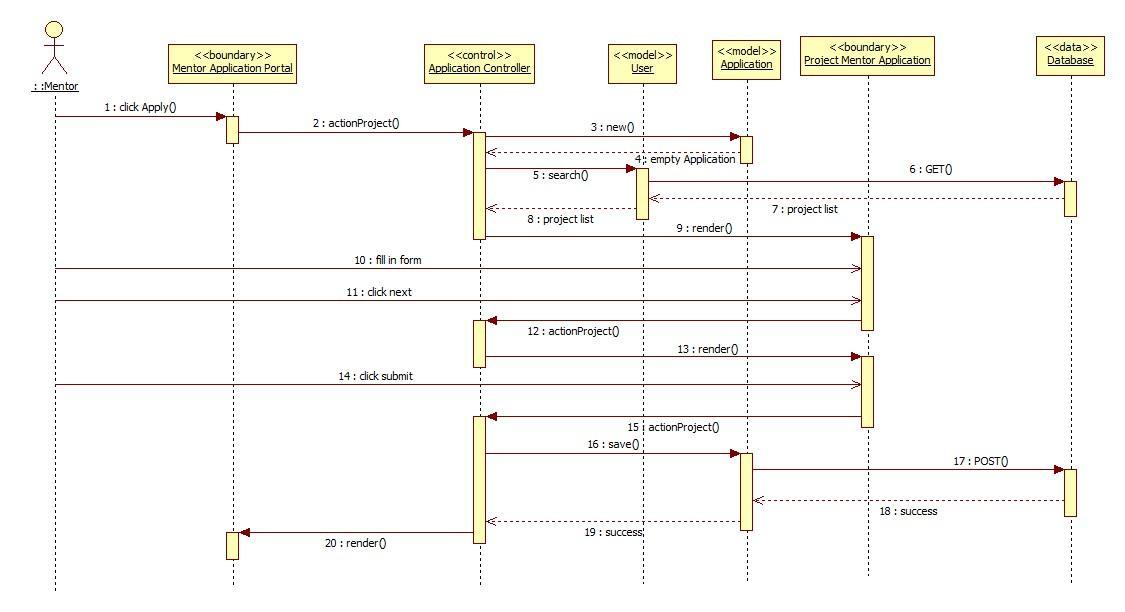


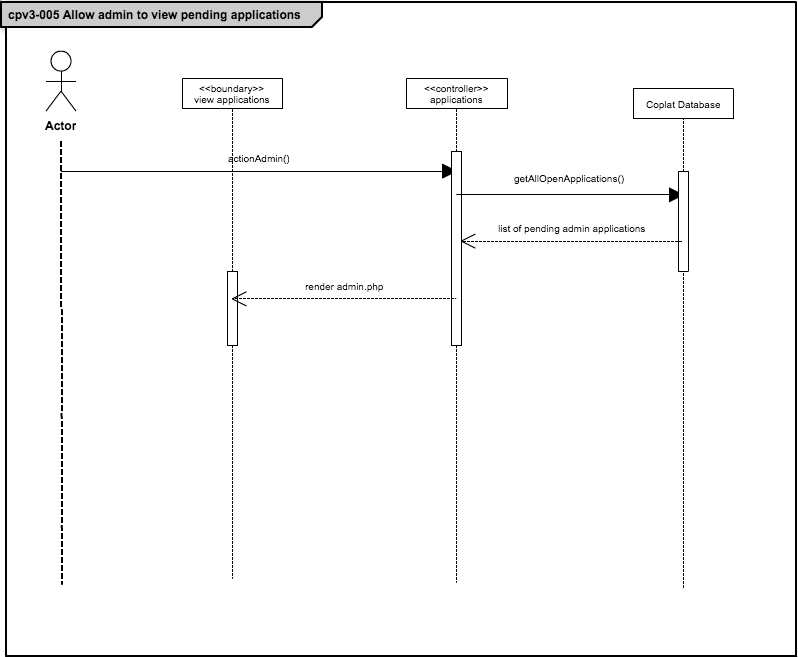
AD. 58 Self Registration for Collaborative Platform

AD. 59 Self Registration for Collaborative Platform

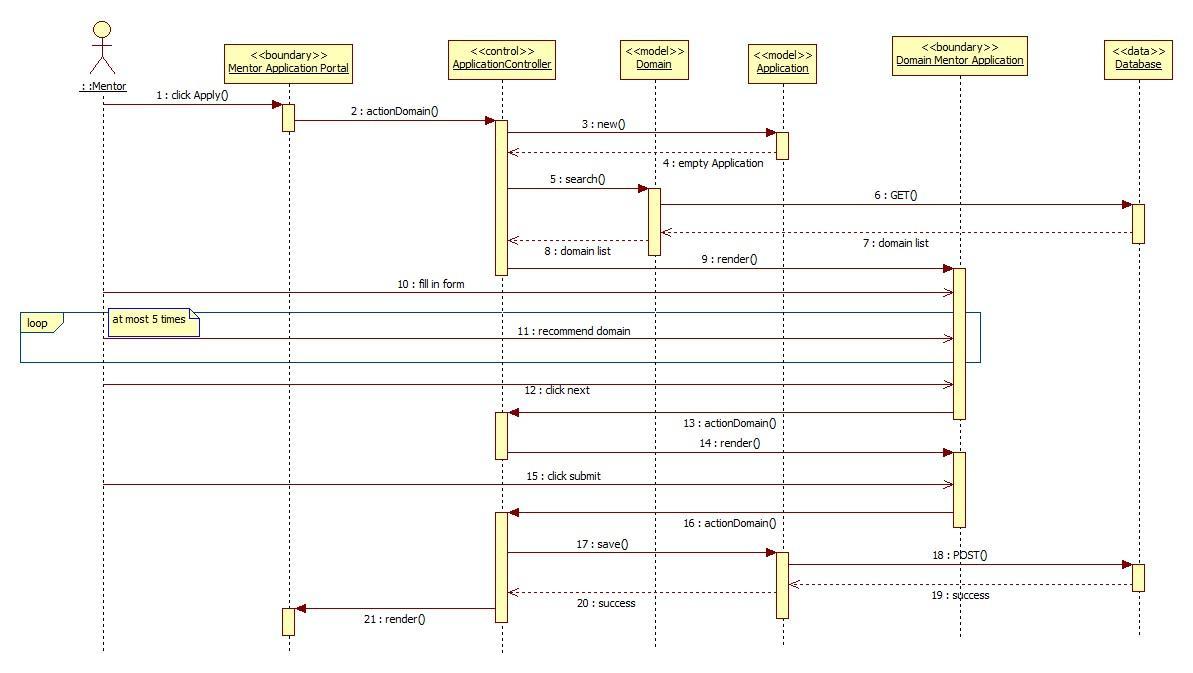


AD. 60 Search for User(s)

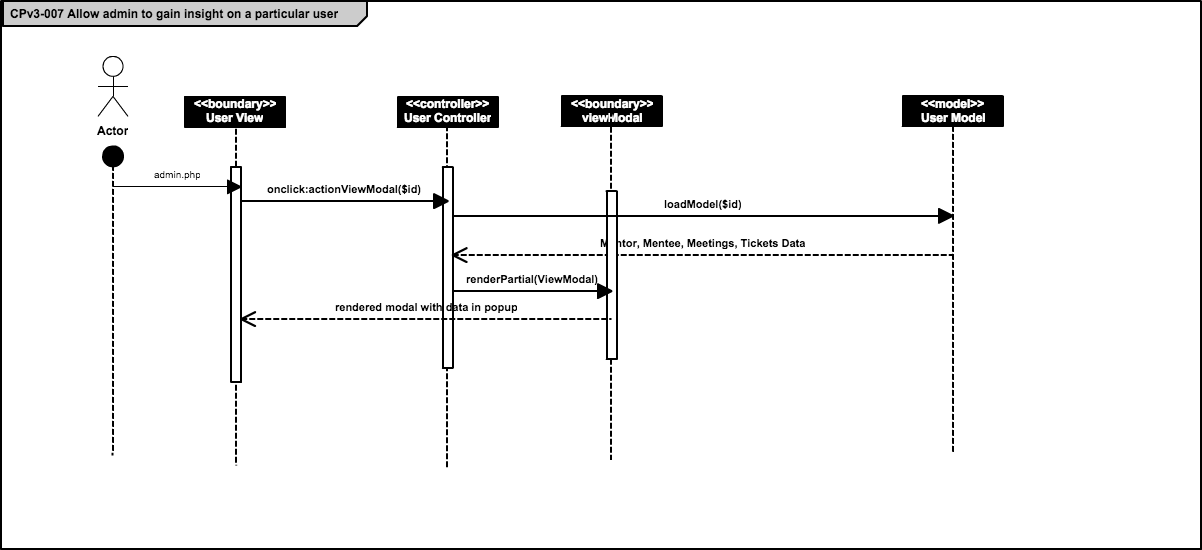
AD. 61 Apply for Project Mentorship

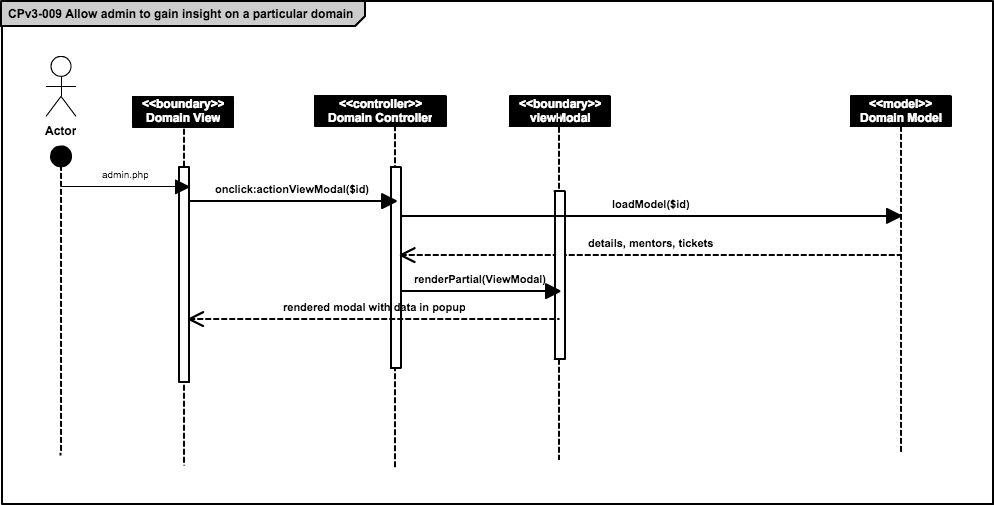


AD. 62 Allow admin to view pending applications

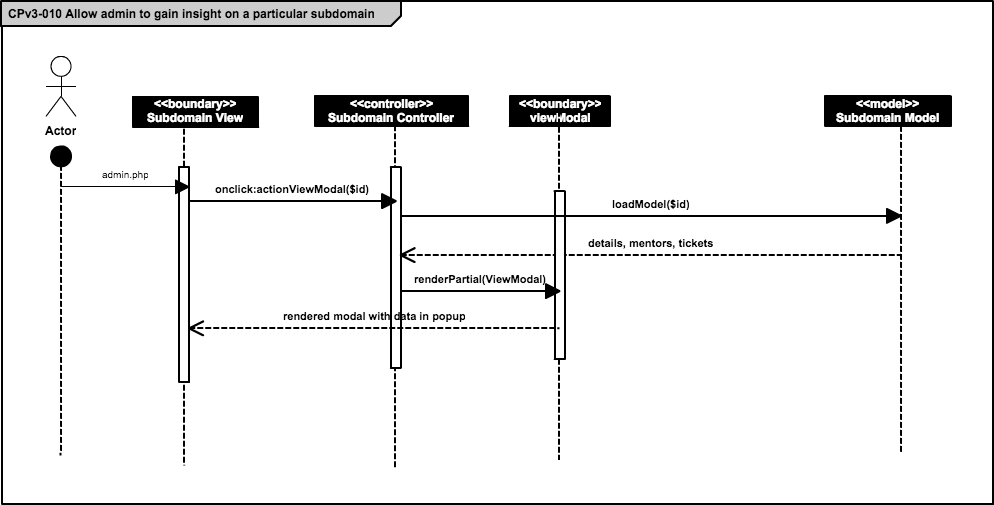


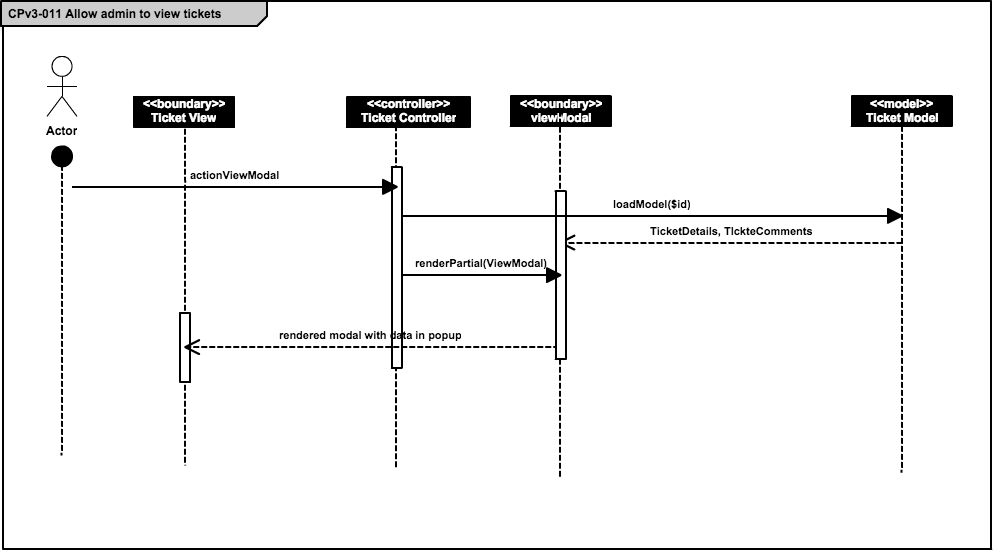
AD. 63 Apply for Domain Mentorship

AD. 64 Allow admin to gain insight on a particular user

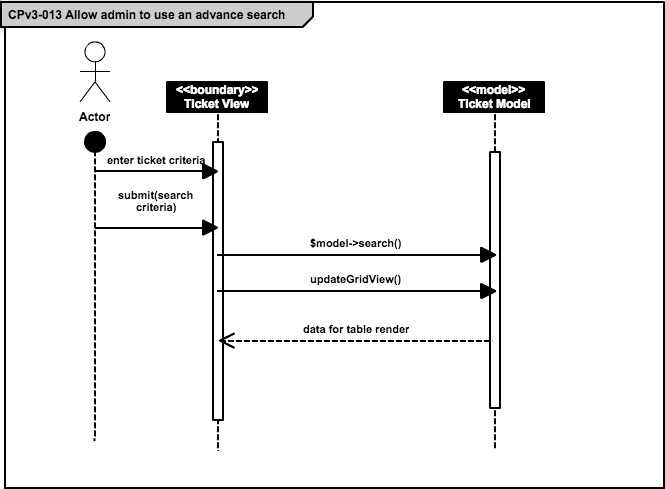


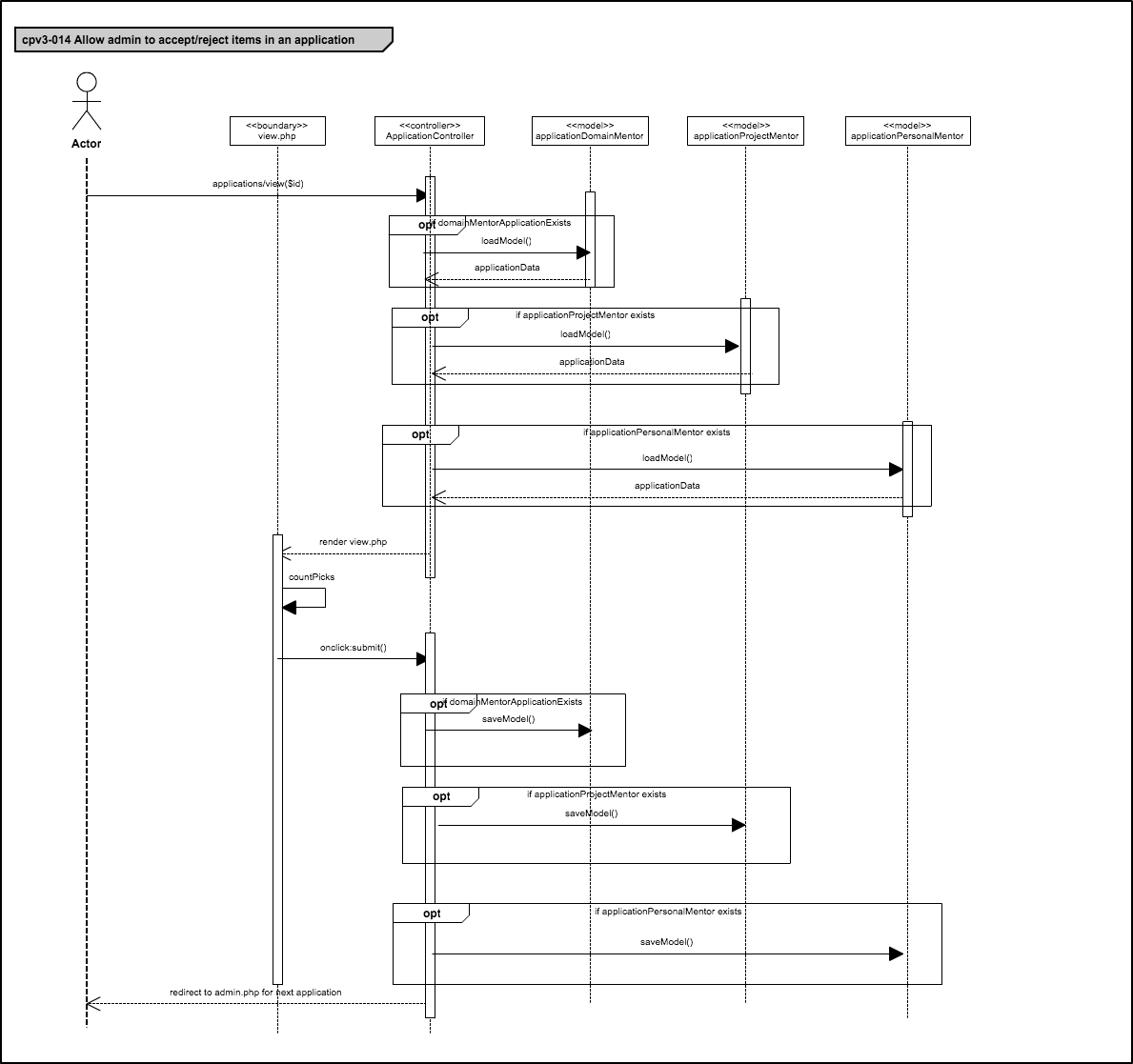
AD. 65 Allow admin to gain insight on a particular domain

AD. 66 Allow admin to gain insight on a particular subdomain

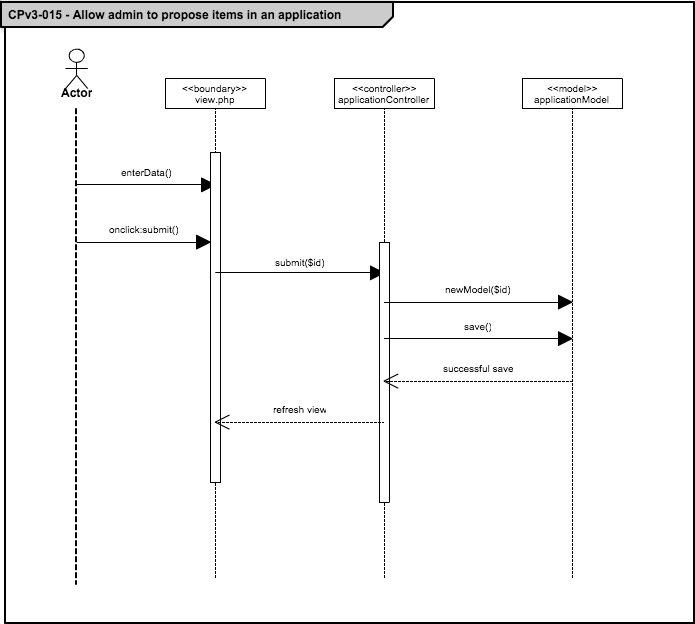


AD. 67 Allow admin to view tickets

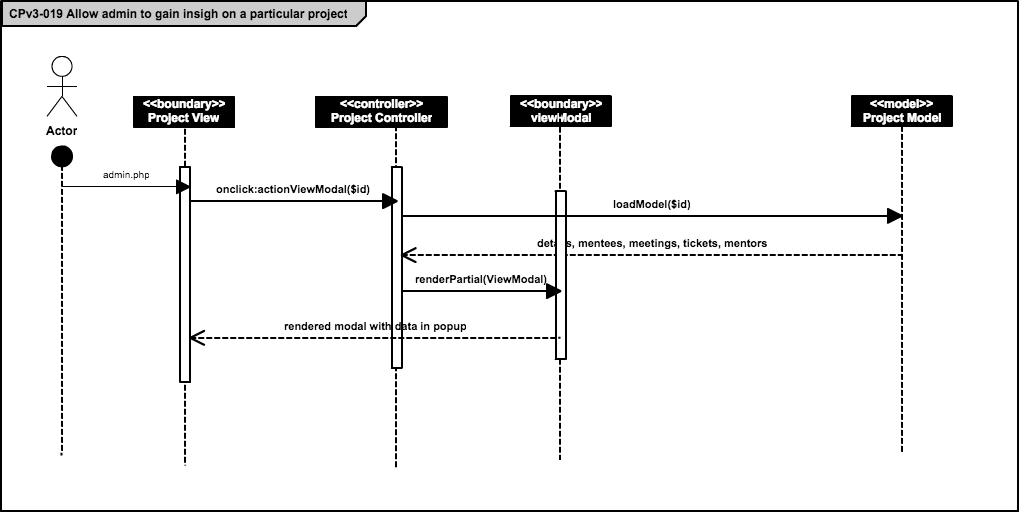
AD. 68 Allow admin to use an advance search



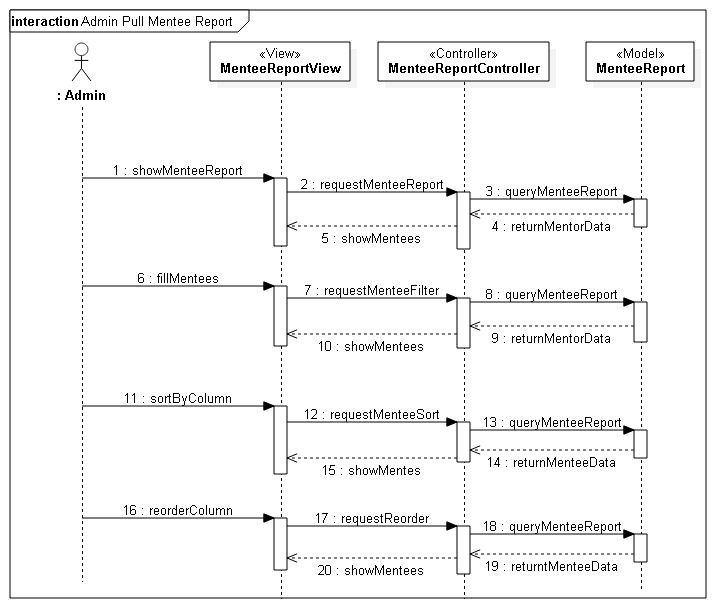
AD. 69 Allow admin to accept/reject items in an application

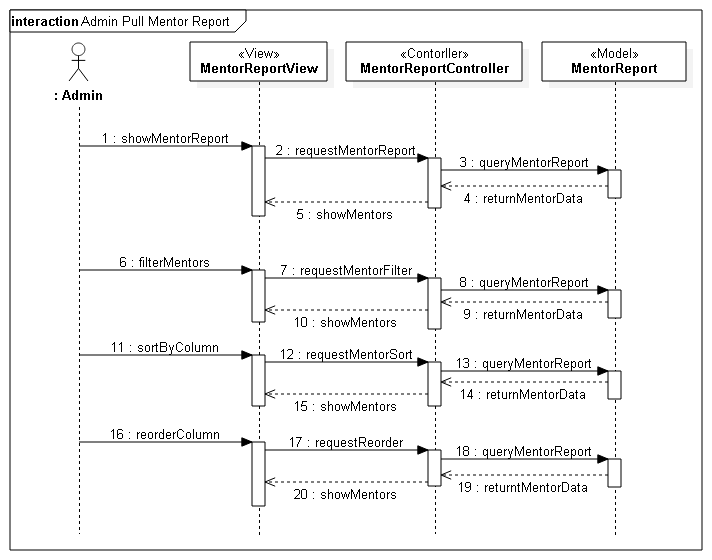


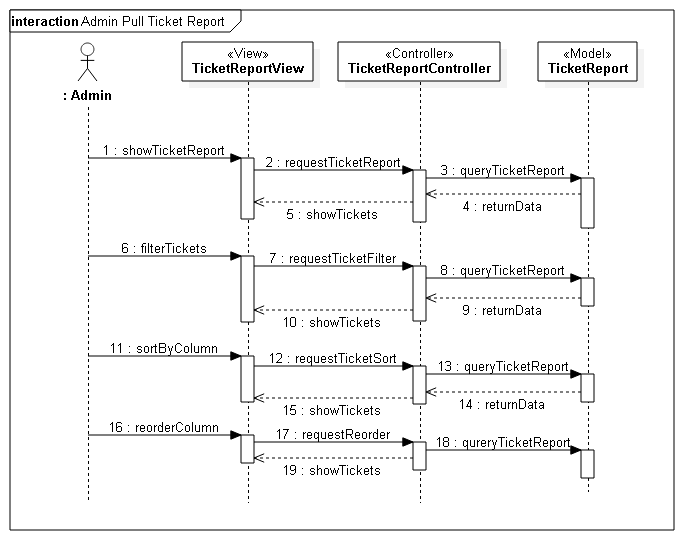
AD. 70 Allow admin to propose items in an application

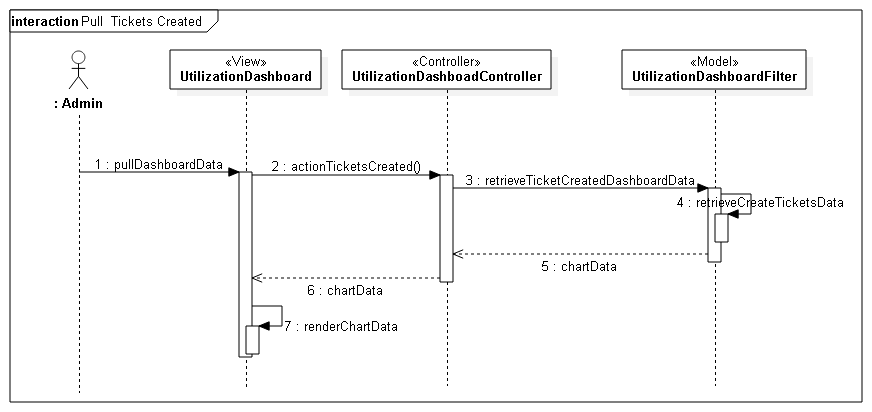
AD. 71 Allow admin to gain insight on a particular project

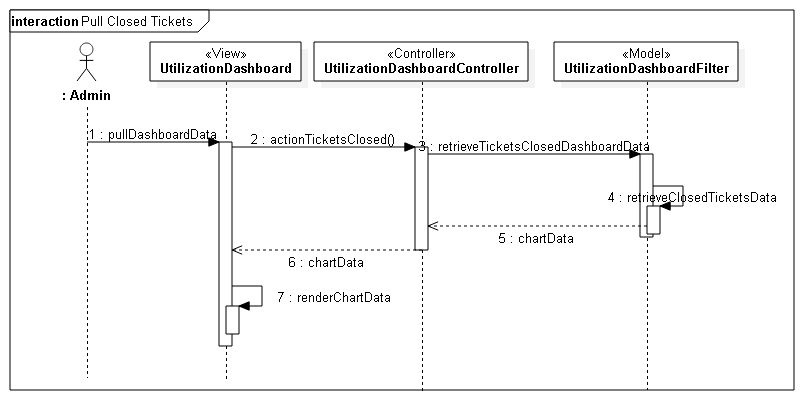
**Mentoring Module: Version 5 - Sequence Diagrams**

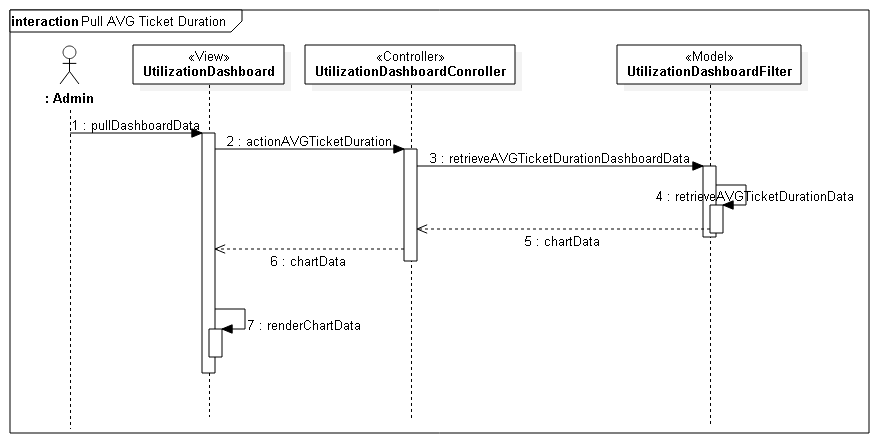
AD. 72 Admin pull mentee report

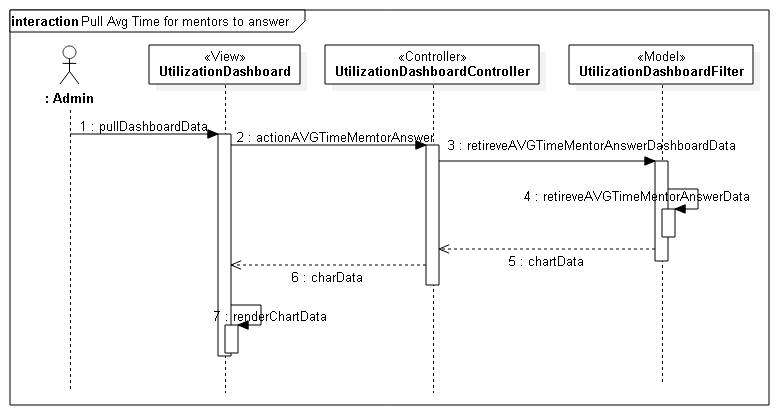
AD. 73 Admin pull mentor report

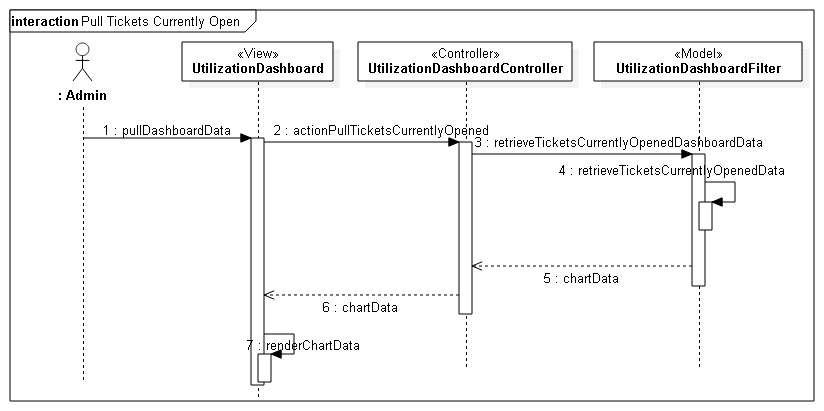
AD. 74 Admin pull ticket report

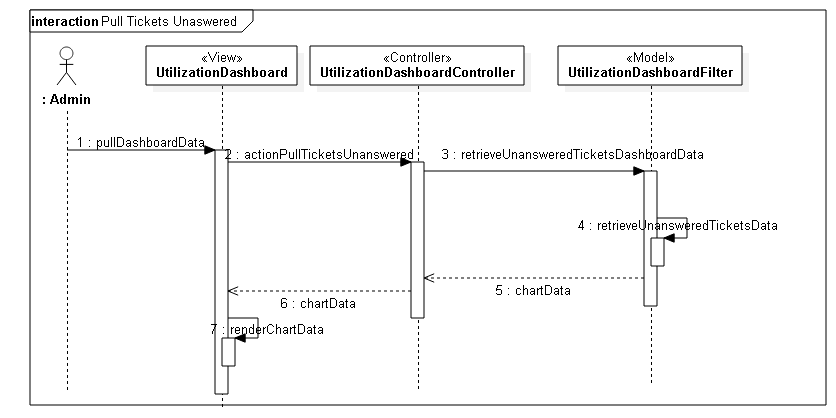
AD. 75 Pull tickets created

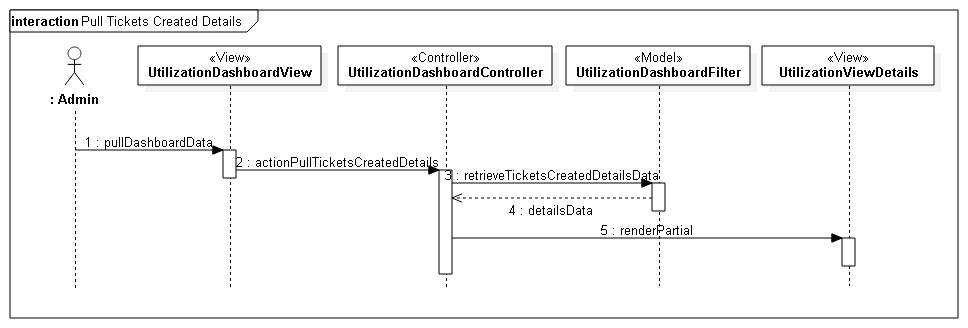
AD. 76 Pull closed tickets

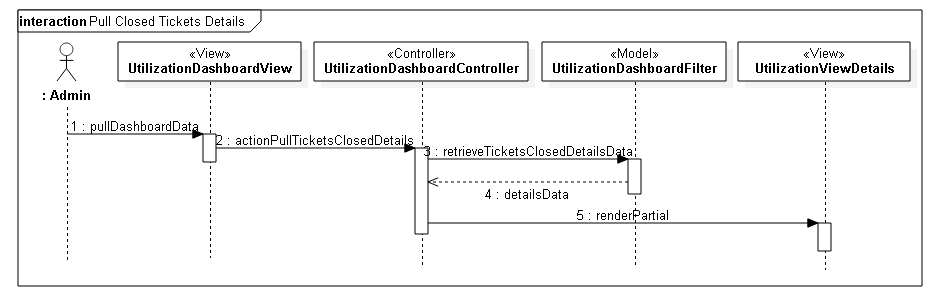
AD. 77 Pull Avg ticket duration

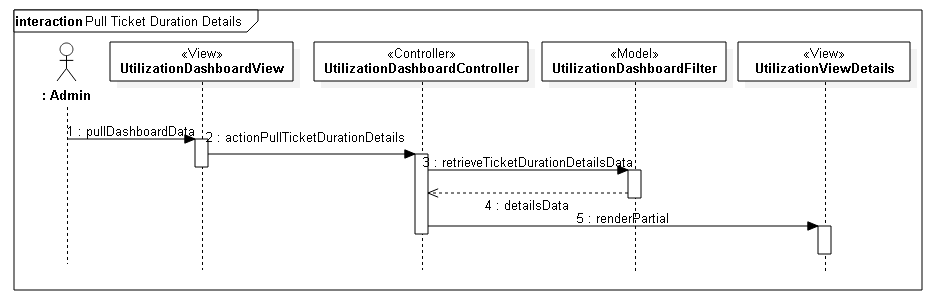
AD. 78 Pull Avg time for mentors to answer

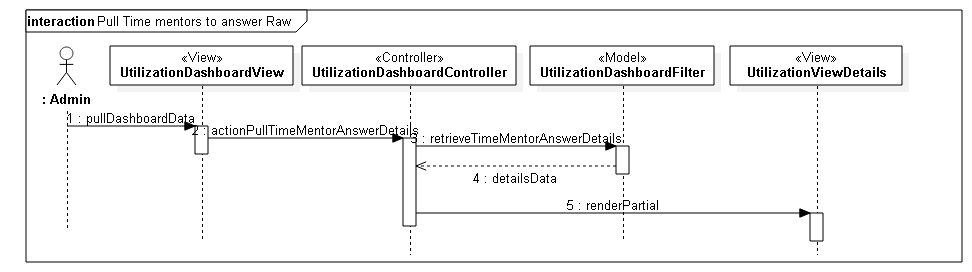
AD. 79 Pull tickets currently open

AD. 80 Pull tickets unanswered

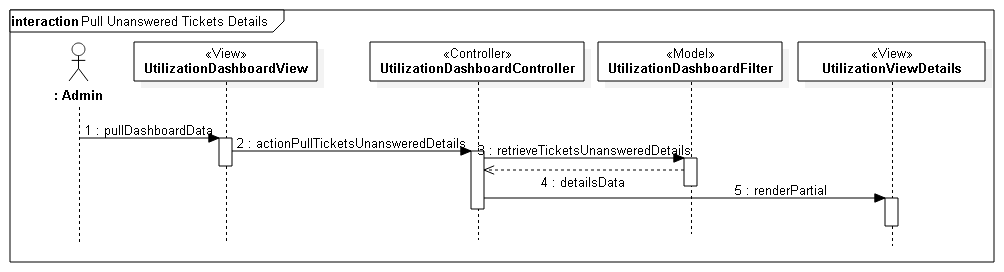
AD. 81 Pull tickets created details

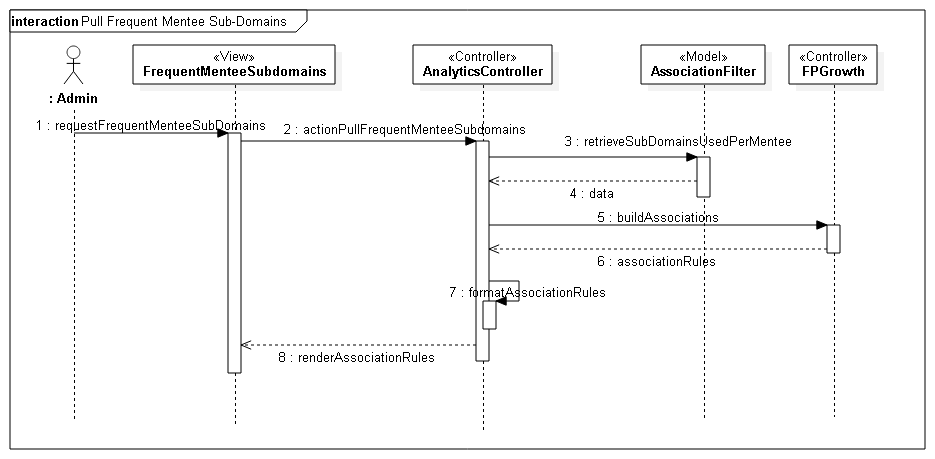
AD. 82 Pull closed ticket details

AD. 83 Pull ticket duration details

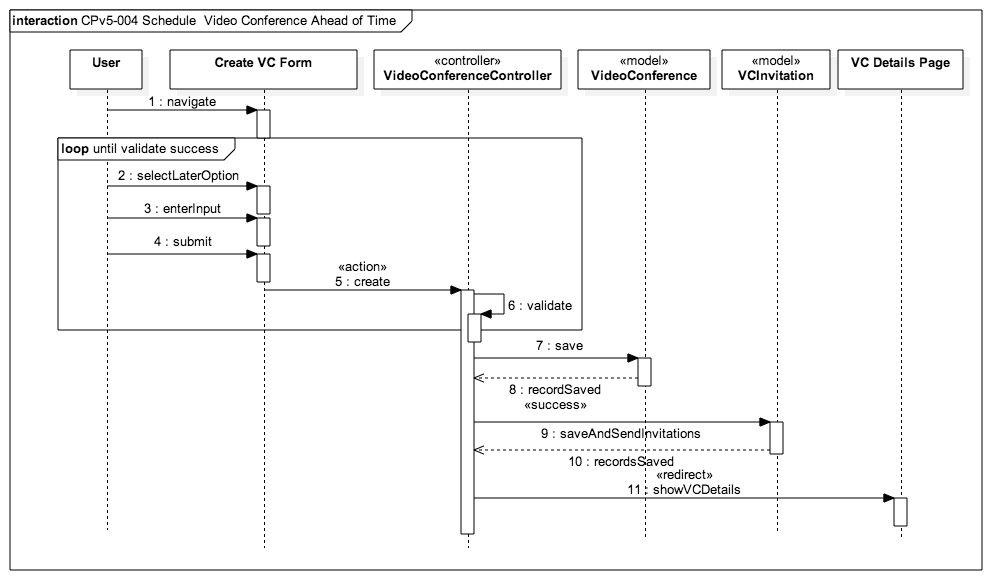
AD. 84 Pull time mentors to answer raw

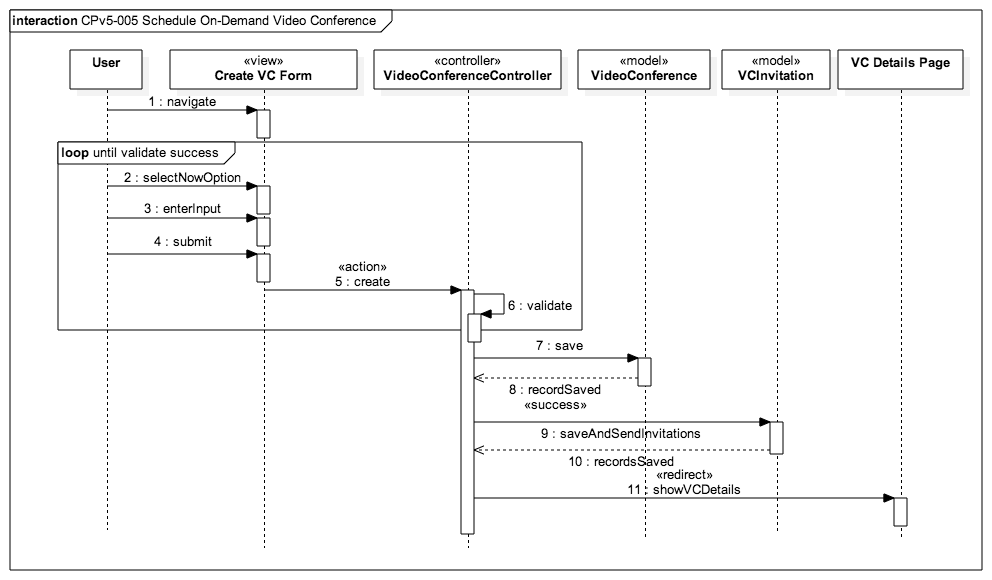
AD. 85 Pull tickets currently open details

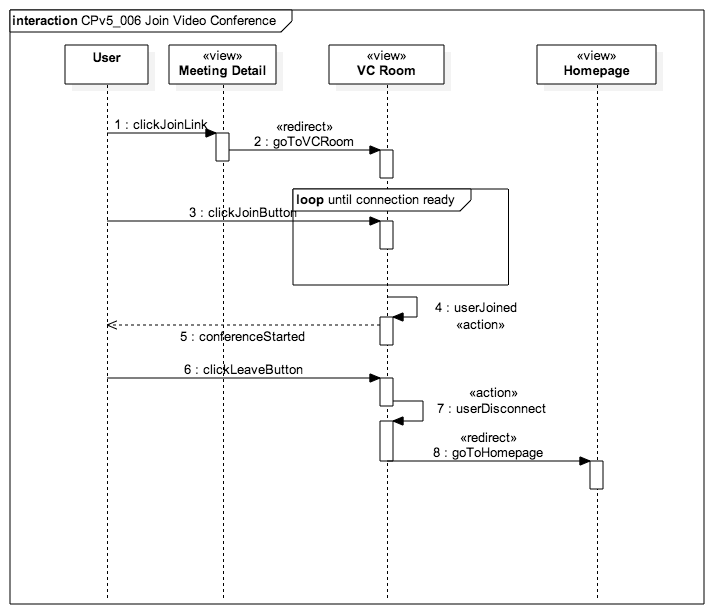
AD. 86 Pull unanswered ticket details



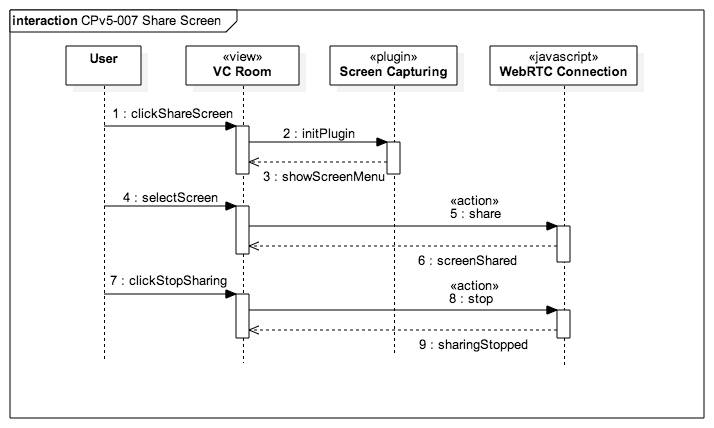
AD. 87 Pull frequent mentee sub-domains

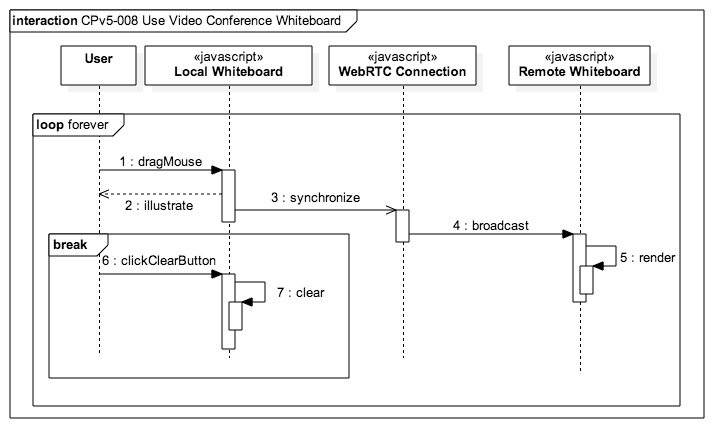
AD. 88 Schedule a video conference ahead of time

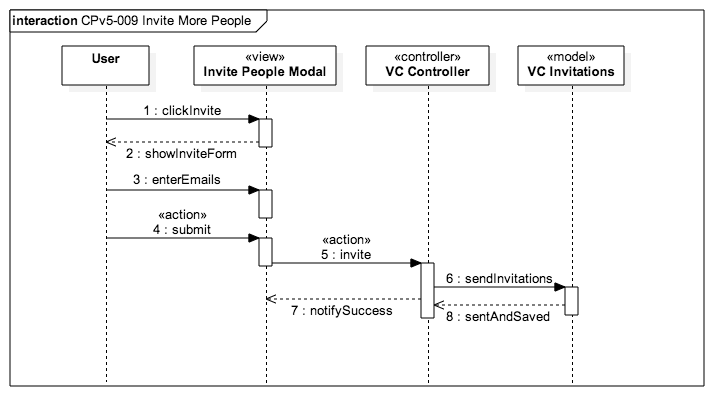
AD. 89 Schedule on-demand video conference

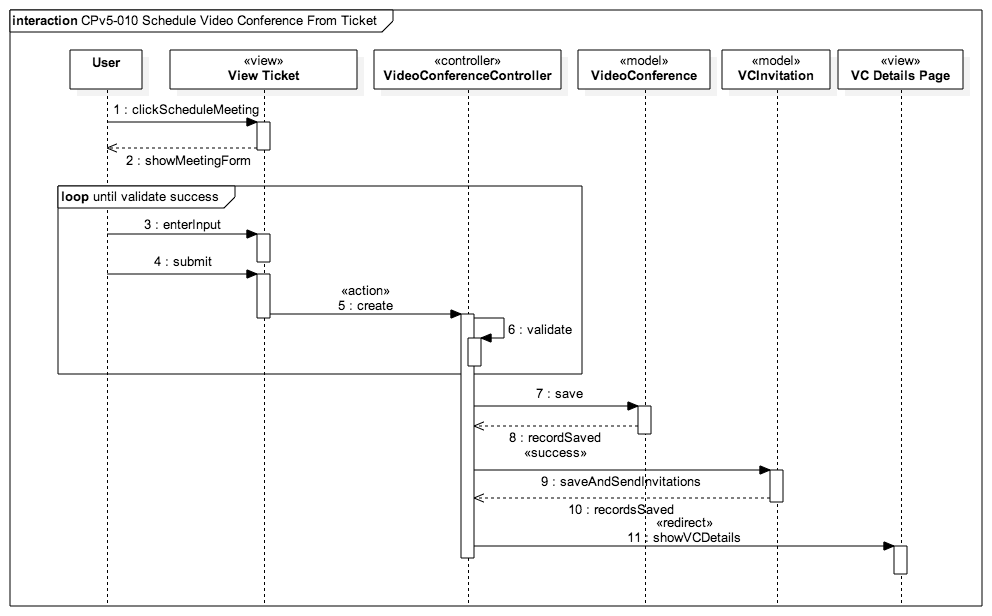


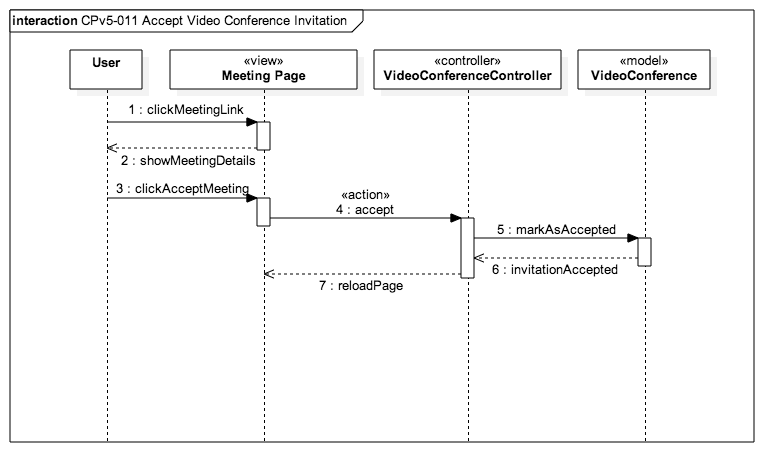
AD. 90 Join video conference

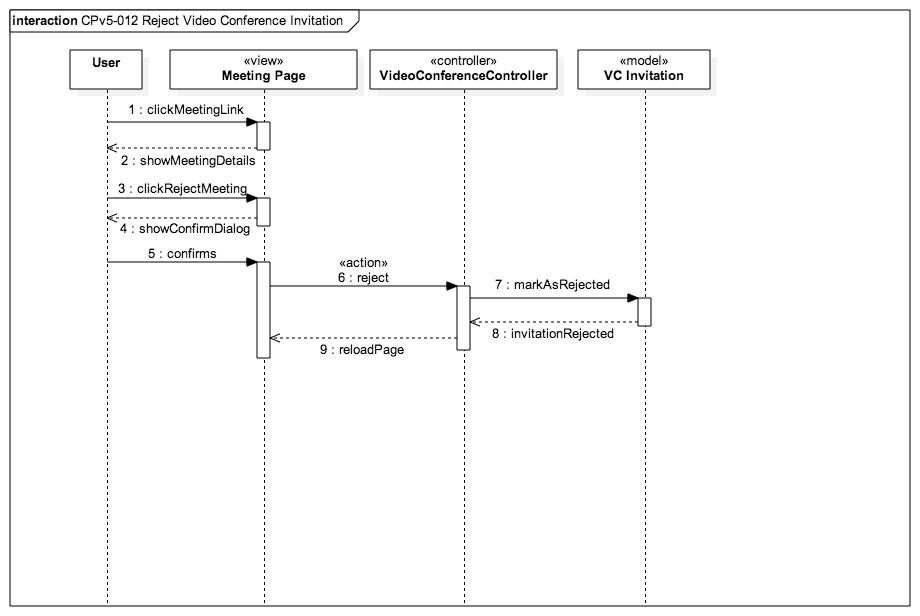
AD. 91 Share screen

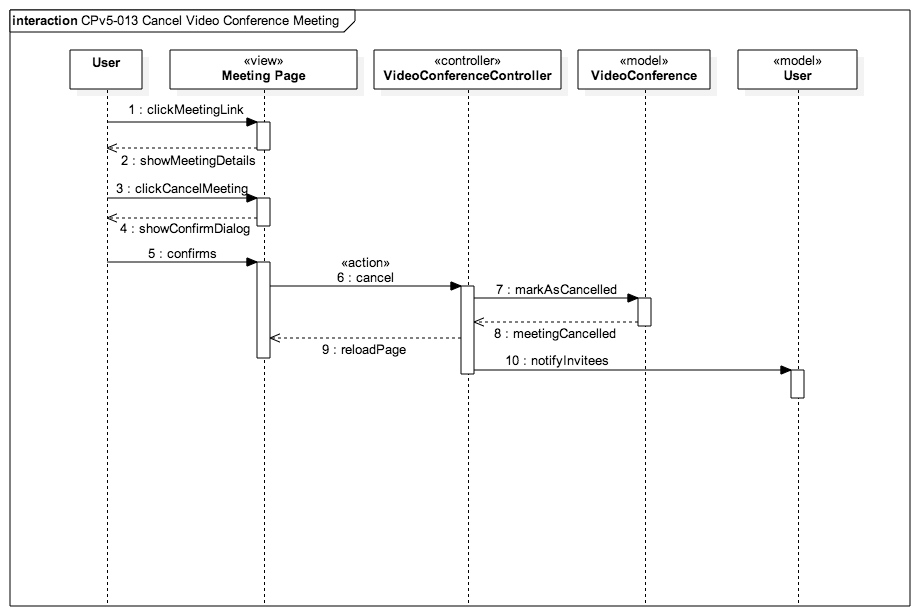
AD. 92 use video conference whiteboard

AD. 93 Invite more people

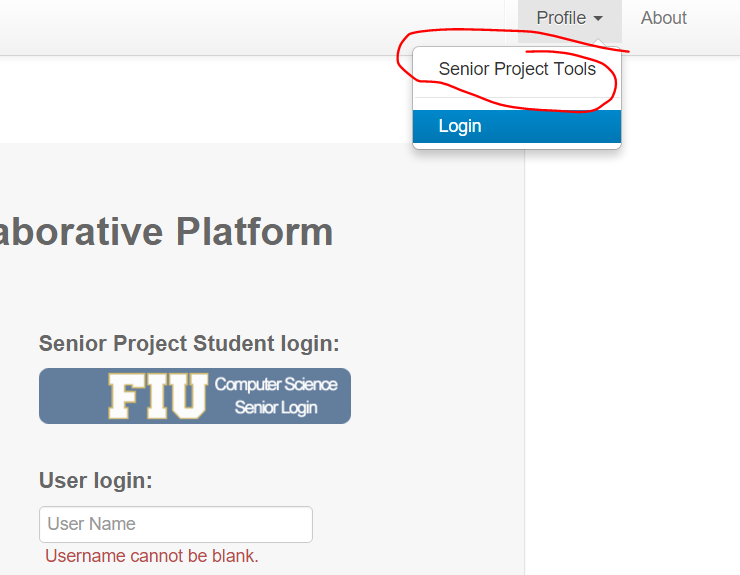
AD. 94 Schedule video conference from ticket

AD. 95 Accept video conference invitation

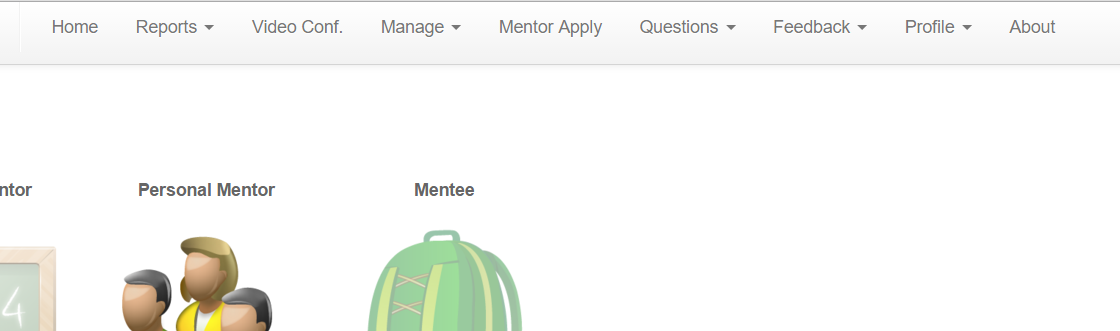
AD. 96 Reject video conference invitation

AD. 97 Cancel video conference meeting

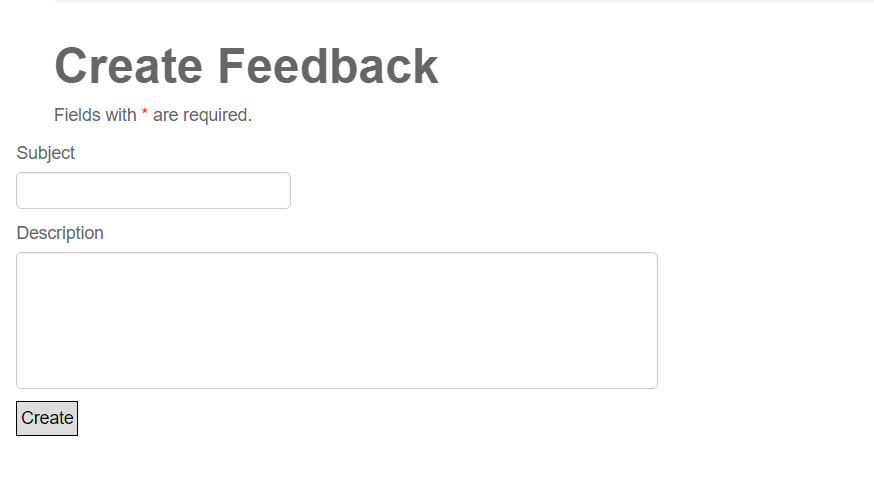
Appendix B - User Interface Design



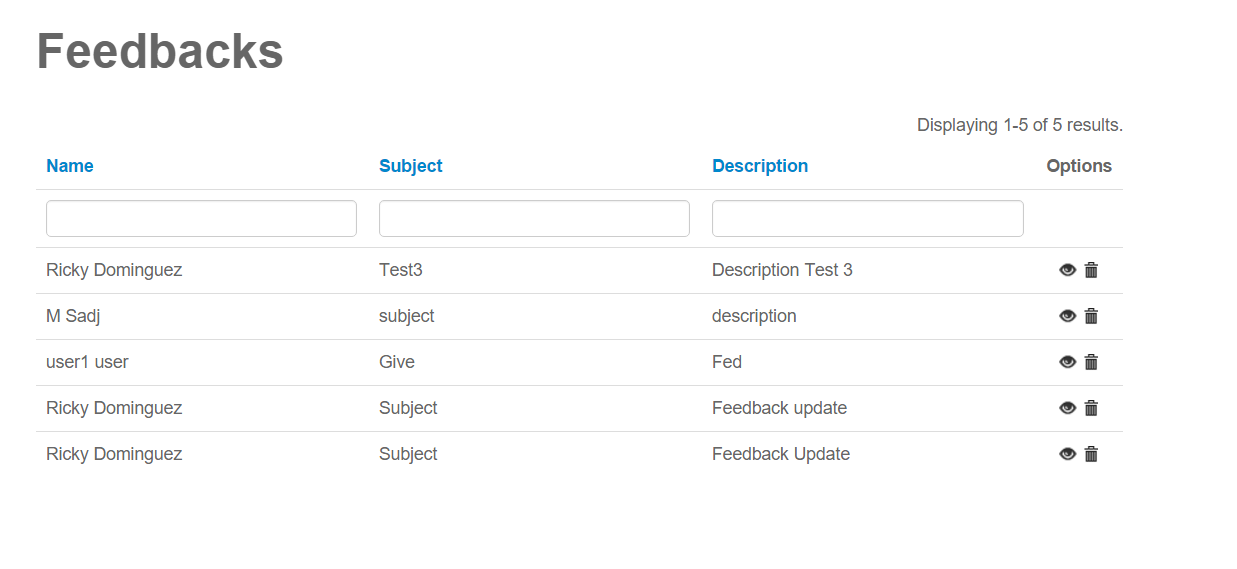
New Top Menu (no login)



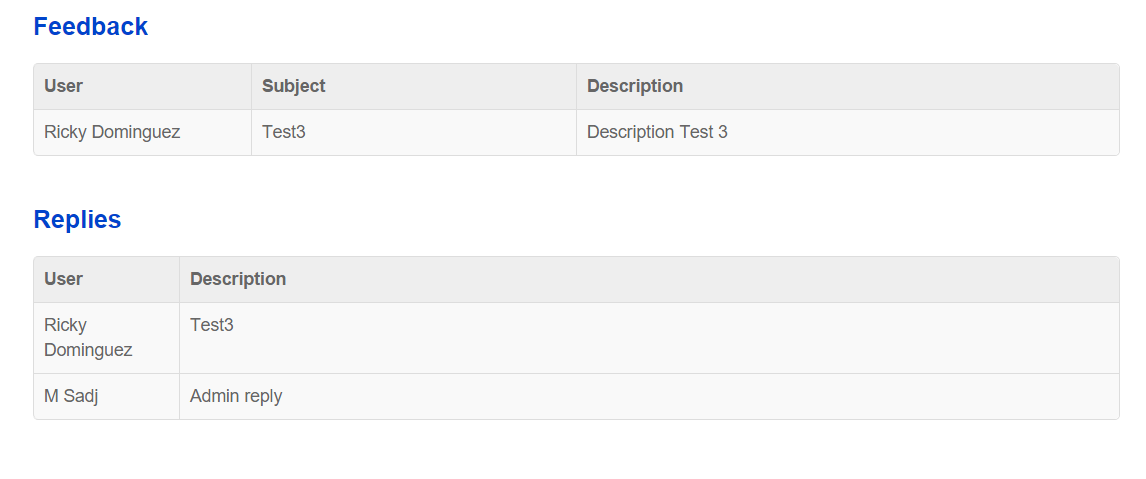
New top menu for logged in users



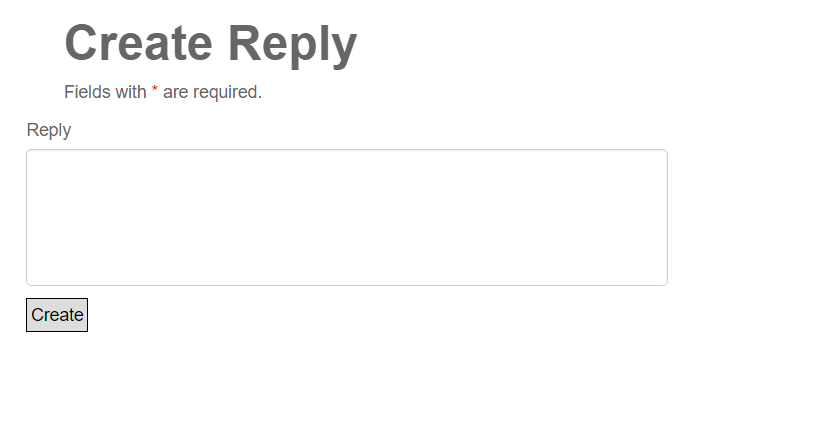
Create Feedback User Page



Feedback View Page



Specific Feedback View



Create Reply Page

## Appendix C - Sprint Review Reports

**Sprint 1 Report**

**Date:** 09/11/2015

**Attendees:** Christopher Jones, Ricardo Dominguez, Frank Hernandez, Anthony Amador, Masoud Sadjadi

**Discussed Topics:**

I, Ricardo Dominguez, was able to successfully complete the goal of installing the previous system onto my machine. I made sure that the localhost ran and that I was able to interact with it like I was able to interact with the development server. I made users and tickets and made sure that everything was already set to go for the following sprint. I started working on the production database as well and began wiping unnecessary information from it as Juan wanted it completely fresh.

**Sprint 2 Report**

**Date:** 09/25/2015

**Attendees:** Christopher Jones, Ricardo Dominguez, Frank Hernandez, Anthony Amador, Masoud Sadjadi

**Discussed Topics:**

I, Ricardo Dominguez, was able to successfully complete the goal of fully installing the production version of Collaborative Platform. I had to begin by wiping the database and making sure that everything worked once I migrated the application over. I had several difficulties and the site stopped working for a while but in the end I managed to get it up and running. I also verified with the SCIS admin that the email server had been set up under [fiucoplat@cp.cis.fiu.edu](mailto:fiucoplat@cp.cis.fiu.edu) and that should work like the email that we have set up for the development server. I made sure that the video conferencing also worked on the production site.

**Sprint 3 Report**

**Date:** 10/09/2015

**Attendees:** Christopher Jones, Ricardo Dominguez, Frank Hernandez, Anthony Amador, Masoud Sadjadi

**Discussed Topics:**

I, Ricardo Dominguez, was able to successfully complete the goal of installing and familiarizing myself with PHPUnit using the Yii Framework. I was able to write a very few simple set of tests for a sample class I had made and verified that the tests ran. I also began working on the Feedback and feedback replies system. I managed to flesh out the initial system in a very rough manner but plan to better it in the upcoming sprint. I also began to code the logic behind the feedback replies system.

**Sprint 4 Report**

**Date:** 10/23/2015

**Attendees:** Christopher Jones, Ricardo Dominguez, Frank Hernandez, Anthony Amador, Masoud Sadjadi

**Discussed Topics:**

I, Ricardo Dominguez, was able to accomplish my goal of finishing the Feedback and Feedback replies feature. I managed to change the overall look of it to give it a more distinguished “Manage” look using GridView. I managed to make them both appear on the same page despite having 2 different models and controllers and made sure that the replies were linked to the original feedback. I changed some of the view information around to show more relevant information to the user and finished the documentation for it (class diagram and sequence diagrams) as well as fixing a few minor issues with the interface that now makes the feedback feature easier to parse for the user.

**Sprint 5 Report**

**Date:** 11/06/2015

**Attendees:** Christopher Jones, Ricardo Dominguez, Frank Hernandez, Anthony Amador, Masoud Sadjadi

**Discussed Topics:**

I, Ricardo Dominguez, was able to accomplish my goal of writing out the first set of unit tests. I had been working with Selenium IDE and began writing the unit tests based on, what I perceived to be, the most important functions of Collaborative Platform. This was User functionality, Ticketing, Mentor Application, and Video Conferencing. The tests made sure that the backend behind these features worked as it should. I also managed to do the Feedback and Feedback Replies unit tests and managed to change various things about the UI that cluttered it given certain circumstances.

**Sprint 6 Report**

**Date:** 11/20/2015

**Attendees:** Christopher Jones, Ricardo Dominguez, Frank Hernandez, Masoud Sadjadi

**Discussed Topics:**

I, Ricardo Dominguez, was able to accomplish my goal of writing the first set of functionality tests. I had been looking at Selenium IDE and began writing the corresponding functionality tests for the unit tests I had already written. This covered the User functionality tests, ticketing functionality tests, Mentor application, and Video Conferencing. I was able to document and further improve some of the unit tests I had written during the previous sprints as well as put suites together for the functionality tests mentioned prior. I also helped Christopher by coding a page that would accept a set of user/pass and log the user into my system automatically.

**Sprint 7 Report**

**Date:** 12/04/2015

**Attendees:** Christopher Jones, Ricardo Dominguez, Frank Hernandez, Masoud Sadjadi

**Discussed Topics:**

I, Ricardo Dominguez, was able to accomplish the goal of the sprint was to complete the second pass of unit and functionality testing. What this means is that, while I had covered, the main areas of the application, there were other areas that had not been tested. I managed to write tests for those areas (such as the messaging function) and write functionality tests that covered these areas so that they could be used in the future. I documented the previous code I had written and compiled a list of all the use cases to date, as well as the corresponding unit and functionality tests for them.

## 

## Appendix D - Sprint Retrospective Reports

**Sprint 1 Retrospective**

**Date:** 9/11/2015

**Attendees:** Christopher Jones, Ricardo Dominguez, Frank Hernandez, Anthony Amador, Masoud Sadjadi

**Discussed Topics:** What did we do wrong and what could we do better for the next sprint.

All things considered, everything seemed to have gone well. At the time I didn’t really have an understanding of Yii or what my main goal was or even how many of the things I was installing worked but as I worked with them in the following sprints, I picked up on how everything came together. I could’ve definitely spent more time learning the system and learning the way that the code was structured in order to progress faster through future sprints.

**Sprint 2 Retrospective**

**Date: 09/25/2015**

**Attendees:** Christopher Jones, Ricardo Dominguez, Frank Hernandez, Anthony Amador, Masoud Sadjadi

**Discussed Topics:** What did we do wrong and what could we do better for the next sprint.

I feel like for this sprint I had more trouble than I should have had. I wasn’t familiar with git or how to clone things and pull things into a Red Hat server but as time went on, I managed to get the hang of it. I spent probably 5x the time I would now setting up the production server and wiping the database. I did work very hard that sprint so there’s really not much else I could’ve done.

**Sprint 3 Retrospective**

**Date: 10/09/2015**

**Attendees:** Christopher Jones, Ricardo Dominguez, Frank Hernandez, Anthony Amador, Masoud Sadjadi

**Discussed Topics:** What did we do wrong and what could we do better for the next sprint.

During this sprint was when I had first found about PHPUnit and wanted to learn it. I could’ve spent more time writing tests and familiarizing myself with the system. My knowledge of the coplat system was shaky at best, I could have figured things out but nowhere near as quickly as if I would’ve studied the system well. I should have sat down more and really started writing some unit tests and started looking at the project so that I could have written a list of the kinds of tests I would be needing to write once I had finally learnt how PHPUnit worked within Yii.

**Sprint 4 Retrospective**

**Date: 10/23/2015**

**Attendees:** Christopher Jones, Ricardo Dominguez, Frank Hernandez, Anthony Amador, Masoud Sadjadi

**Discussed Topics:** What did we do wrong and what could we do better for the next sprint.

During this sprint, I started late so I didn’t spend nearly as much time as I wanted writing the unit tests because I had been finishing up the Feedback and Feedback Replies. What I should have been doing was writing the unit tests before I started implementing the feedback and feedback reply models so that I could get an understanding of how the project and the features worked together.

**Sprint 5 Retrospective**

**Date: 11/06/2015**

**Attendees:** Christopher Jones, Ricardo Dominguez, Frank Hernandez, Anthony Amador, Masoud Sadjadi

**Discussed Topics:** What did we do wrong and what could we do better for the next sprint.

This is where I wrote the first big part of all the unit tests. What I could have done better was spend more time writing these tests and refactor the code as I went along. Because I didn’t do this, I didn’t have time in the end to refactor code that was needed for these tests making some of them redundant until the code is refactored into the proper functions for the model. Instead what I did was recreated the logic used in the controllers with the tests but it didn’t exactly click that this wouldn’t help anyone in the future until a future sprint.

**Sprint 6 Retrospective**

**Date: 11/20/2015**

**Attendees:** Christopher Jones, Ricardo Dominguez, Frank Hernandez, Masoud Sadjadi

**Discussed Topics:** What did we do wrong and what could we do better for the next sprint.

This is where I did the first pass of the functionality tests. Something I could have done better was learn different selenium commands that would have allowed for stored procedures to make hard-coding values into the tests less of a thing. I didn’t exactly have a lot of experience with it and at the same time I was still fixing the front end of the site, writing some unit tests and making sure that all of these were included in suites.

**Sprint 7 Retrospective**

**Date: 12/04/2015**

**Attendees:** Christopher Jones, Ricardo Dominguez, Frank Hernandez, Masoud Sadjadi

**Discussed Topics:** What did we do wrong and what could we do better for the next sprint.

I could have done many things differently these last 2 weeks. I could have probably started on the final week’s work so that it wouldn’t all pile up. I could have managed my time much better between my classes so that I could get the work done sooner and document my code more. Overall, I was pretty pressed for time as finals came around so I did the best I could with what I had. I feel like time management is really the only issue these last 2 weeks.

# **Installation Manual**

Collaborative Platform v7

**Abstract**

This document includes a detailed guide about how to setup and install the Collaborative Platform system as of version 7.0. It will guide and prepare you for any problems you may encounter. It should be read before starting to work on the project.

**Installation**

This chapter will cover how to install and setup the project before you begin working.

**Software Needed**

You will need the following software regardless of OS.

* 1. An application server –
     1. Apache
     2. Tomcat
  2. An IDE supporting PHP
     1. Netbeans
     2. PHPStorm (Recommended)
  3. MySQL server and Database Management Software
     1. PHPMyadmin
     2. MySQL Workbench
  4. PHPUnit 5.1
  5. A subversion software
     1. GitHub
     2. SourceTree
  6. Selenium IDE
     1. Plugin for Mozilla

A much easier way to do this is to download XAMP. This will install the application server, the database server and the PHP configuration files.

**Github**

First clone the project from GitHub. This operation can be performed in Github.com whenever you create your repository for the semester.

Version 7.0 can be found at the following link:

<https://github.com/FIU-SCIS-Senior-Projects/Collaborative-Platform>

You need to set up your local environment. This means that you will have a copy of your Git repository locally and changes would have to be synchronized. Start your Sub-version system (I strongly recommend SourceTree) and clone your project from the Git repository you clone it in the first place. That is, where you want to keep the files.

**IDE**

To install your IDE: if you chose WAMP/XAMP then you need to remember where you install it because you will need to create a symbolic link (very important) from the www folder inside XAMP. The symbolic link needs to be named coplat and point to the Website folder inside the Code/Website folder in the local Git repo folder.

Another way to do it is to put a copy of the coplat folder in XAMP and whenever you decide that you want to update the repository just overwrite the files within the GitHub repository on your machine.

**Database and Website Setup**

For the database: if you installed WAMP/XAMP, this was already made for you. Create a database with name coplat.

Within the Code/Database folder there should be a file with the latest database information. All you have to do is open up MySQL workbench, click the Open SQL file button, select the latest version (in this case 7.0) and just execute it. Your database will populate but it won’t show until you hit “Refresh All”.

For the website: Make sure you have your coplat folder from Code/Website inside the htdocs folder in XAMP. Once you enable it, you will be able to go to your localhost/coplat/index.php and begin to run the application. Something you should do is go to your main.php and under connectionString, make sure the password is ‘’ rather than the development database password otherwise it might give you an error.

**PHPUnit**

Your XAMPP comes with PHPUnit already so all you would really have to do is go to the settings of your IDE, go to languages -> PHP -> PHP Unit, make sure the IDE file knows where to find the autoload.php (It’s in your XAMPP/PHP/pear/phpunit) and you should be good to go. From that point on you can write tests should you extend the CTestCase.

**Selenium IDE**

In this section we will cover how to install Selenium IDE plugin for Mozilla Firefox browser.

1. Open a browser and navigate to <http://www.seleniumhq.org/download/>

2. Download the Selenium IDE which is the \*.xpi and install it as a Firefox plugin.

3. The IDE should appear as a button on your browser once you refresh and when pressed it should bring up a window where you can open test files and suites created in previous semesters.

Then you can run any of the tests left to you from previous semesters. If you run into any problems, there’s a visual representation of this in the installation video.

# **References**

**Mentor Module V1**

<http://users.cis.fiu.edu/~sadjadi/Teaching/SeniorProject/Deliverables/SP14/02-CollaborativePlatform.rar>

**Mentor Module V2**

<http://users.cis.fiu.edu/~sadjadi/Teaching/SeniorProject/Deliverables/SU14/02-CollaborativePlatformV2.zip>

**Mentor Module V3**

[**http://users.cis.fiu.edu/~sadjadi/Teaching/SeniorProject/Deliverables/FA14/2014-Fall-FinalDeliverable-CollaborativePlatform-v3.zip**](http://users.cis.fiu.edu/~sadjadi/Teaching/SeniorProject/Deliverables/FA14/2014-Fall-FinalDeliverable-CollaborativePlatform-v3.zip)

Mentor Module V5

[**http://users.cis.fiu.edu/~sadjadi/Teaching/SeniorProject/Deliverables/SP15/02-CollaborativePlatform-Ver5.zip**](http://users.cis.fiu.edu/~sadjadi/Teaching/SeniorProject/Deliverables/SP15/02-CollaborativePlatform-Ver5.zip)