Florida International University
School of Computing and Information Sciences

CEN 4010 - VIP Project
Software Engineering I

Final Deliverable

Vertically Integrated Projects
(VIP)

Product Owner: Masoud Sadjadi
Instructor: Masoud Sadjadi

Team Members

Eric Aguiar	Lukas Borges	Joseph Casal
Emmanuel Gonzalez	Eduardo Guerra Fernandez	Sean Iamartino
Lucas Iglesias	Brenda Izquierdo	Pierre Jimenez
Tom Lam	Jacob Leschen	Lizette Nicole
Andrew Mitchell	Tiago Moore	Christopher Naranjo
Alan Nieto	Luis Puche Rondon	Steven Rowe
Alex Sanchez	Emmanuel Vinas	Garrett Lemieux

Abstract

This document contains information about how the team of Software Engineering I students began implementing the VIP (Vertically Integrated Projects) project at FIU. It contains information about the developing environments we used, as well as tasks we performed to get the project started. Additionally it contains some of our development process such as UML diagrams and some information about our agile sprints. Design decisions and goals are also listed for some user stories.

Table of Contents

Introduction

This section will introduce you to the new VIP project. It begins by discussing the current system in place, or lack of, for handling faculty research and development projects. Due to the current state of our system, we elaborate on the need to for new system and how we manage to obtain it.

Current System

Currently, there exists no centralized system to display and manage current faculty research and development projects. While some faculty members have created small systems for personal use, it remains that these tasks are mainly being handled by facilities and tools that are also available to Computer Science Senior Project students. Additionally, there is no easy way for a student to browse and apply to projects he or she is interested in working on as well as no simple, automated system to announce new projects available to students.

Purpose of New System

The purpose of this new system; henceforth called *VIP,* is to design a centralized system that allows faculty members to advertise and manage their current research and development projects more easily. It will also enable faculty members to directly propose new projects to administrators, and the owner of the VIP project for approval and project funding.

Due to the new, centralized management system, we will also create a searchable database of projects that undergraduate students may apply and potentially work on for academic credit. Since VIP projects are more accessible and potentially allow many students to work on a single project, there will also be a simple undergraduate student management interface.

Our intended audience for this project is as follows: *Principal Investigator (PI),* who owns and manages the VIP project and funding. *Faculty and Staff*, who can submit and manage their own projects. And finally, *Students,* who are able to apply to and work on projects for academic credit.

USER STORIES

User Stories are the backbone of our development process. These are what we use to designate what type of work we will be doing for the next sprint. User stories follow a simple template:

> As a <Type of User> I want <a specific goal or function> for <some reason>.

This section is a compilation of user stories that were developed by team members in the project. In addition, we included a further description of each user story and the acceptance criteria required for them to be completed.

Implemented User Stories

USER STORY # 667 - SUBSCRIPTION

Description:

- As a student, I would like to subscribe to fiu vip to receive emails notifications

Acceptance Criteria:

1. A User can subscribe to fiu vip by checking the checkbox on the profile page
2. A User can unsubscribe to fiu vip by unchecking the checkbox on the profile page

Owner: Eric Aguiar

USER STORY # 693 - ENABLE PROJECT PROPOSAL SUBMISSIONS

Description:

- As a staff or faculty member I would like to be able to submit and edit a proposal, check the status of it, and receive notifications about it.

Acceptance Criteria:

1. Staff/Faculty can submit new proposal(s) for approval on the website
2. Staff/Faculty can check the status of their pending proposal(s) on the website
3. Staff/Faculty can edit their pending proposal(s) on the website.
4. Staff/Faculty can receive notifications about changes to approval status

Owner: Chris Naranjo

USER STORY # 698 - MANAGE PROJECT GROUPS

Description:

- As a PI, I would be able to view, and modify VIP project groups and its members

Acceptance Criteria:

1. Must be able to add group members
2. Must be able to remove group members
3. Must be able to reassign group members

Owner: Emmanuel Gonzalez

USER STORY # 701 - 6 MONTH REPORTING

Description:

- As a PI, I would like to be able to answer the questions posed to the PI by the sponsor, so that I can generate the 6-month report as requested by the sponsor.

Acceptance Criteria:

1. The PI can answer every questions that the form supplies

a. Implementation, Initial Experiences, and Outcomes

i. Please describe steps taken toward institutionalization of VIP. Institutionalization applies to new processes, policies and structures that will likely continue beyond the funding period.

ii. Please describe any technology challenges you have encountered.

iii. Please describe your strategies for departmental/college approval on how VIP credits can be used to fulfill degree requirements

iv. Please describe faculty receptiveness to the VIP model and program.

v. Please describe any changes in your approach to VIP program implementation, especially approaches and/or models that bring a new dimension to the VIP consortium.

vi. Please describe successes that you have had in your VIP implementation and program activities.

vii. Please list any accepted conference abstracts, conference proceedings, academic or other publications that have resulted from your VIP Program.

b. Evaluation

. Please describe the name and the status (in development, used, etc.) of data systems used to track VIP.

i. Please describe the ways in which you monitor activities and performance of VIP on your campus.

ii. Please describe how you are monitoring your VIP program for progress, success, and areas for improvement. Be sure to note whether your site has implemented evaluation tools for VIP (surveys, questionnaires, focus groups, etc.). Please also indicate the contact person for these activities.
iii. Please describe whether/how you have interacted with the evaluation team, and the status of data sharing with the evaluation team. Be sure to indicate any issues or challenges in this process.

Owner: Steven Rowe

USER STORY # 705 - UPCOMING EVENTS

Description:

- As a student, I would like to see a list of the upcoming events so that I can plan my deadlines accordingly

Acceptance Criteria:

1. A user can view the upcoming events by clicking the link
2. A user can delete an event by clicking a delete button

Owner: Pierre Jimenez

USER STORY # 706 - CREATE EVENTS

Description:

- As a PI, I would like to be able to update the calendar with events

Acceptance Criteria:

1. A PI can visit the page to create events by clicking the button on the menu bar
2. A PI can create an event for a future or current date (cannot create an event for a past date)
3. The information for a created event will be stored in the database
4. After an event is created, an email is sent to everyone who has subscribed

Owner: Joseph Casal

USER STORY # 708 - REPORT PROJECT PROPOSAL SUBMISSIONS

Description:

- As a PI, I would like to be able to view a report on all project proposal submissions within a defined period of time, so that I can include the report in the 6-month report to the sponsor.

Acceptance Criteria:

1. Report contains a list of all proposals submitted in the requested period.
2. Report details each proposal with their acceptance/rejection status, as well as department and project leader.
3. Report will be included in the 6-month report to the sponsor.

Owner: Jacob Leschen

USER STORY # 709 - SEND NOTIFICATIONS TO STUDENTS IN A PROJECT.

Description:

- As a Staff leading a group, I would like to send notifications to students working on my project.

Acceptance Criteria:

1. A Staff can visit the page to create notifications button clicking the button “Notifications” on the menu bar when logged in as a Staff or Faculty.
2. The notifications will be stored in the database.
3. After notifications is sent, an email is sent to everyone who is working on the project specified by user.

Owner: Alan Nieto

USER STORY # 715 - REVIEW STUDENT PARTICIPATION IN A PROJECT

Description:

- As a faculty member, I would like to review each student’s resume that has been chosen by PI to participate in faculty member’s project so that the faculty member can accept or reject the student’s participation.

Acceptance Criteria:

1. A faculty member can view a list of chosen students for his or her project and when faculty member chooses a student to review then resume can be viewed.
2. A faculty member makes a decision on a student’s participation in project when he or she proceeds to express choice then the faculty member can choose either accept or decline statements.
3. A faculty member can express his or her reasons for accepting or rejecting student when he or she clicks additional comments and then can submit comments for PI to review.

Owner: Garrett Lemieux

USER STORY # 717 - REGISTRATION

Description:

- As a Guest User I would like to Register for the VIP Project so I can apply for projects and join the program.

Acceptance Criteria:

1. User must be able to input all the information given by the Product Owner, first name, optional middle name, last name, department, optional cellphone, @fiu.edu email address, and user type.
2. The information must be checked for authenticity (10 integer phone number, @fiu.edu email account etc).
3. The information must be stored in a database. Including a Pending User Type.
4. Verification Process via Email must be implemented.
5. PI Must be able to verify a user if they are faculty, staff, or PI/CoPI (Via Email)
6. PI Can delete a user if they are not of the proper type.

Owner: Andrew Mitchell

USER STORY # 720 - RESUME UPLOADING

- As a student I want to be able to list skills and upload my resume so that my application can be reviewed thoroughly

Description:

- The design for this system will work as follows:
    - A student will Click on a button under the “Resume” Web page
    - A student will then enter pertinent resume information which will added to his user profile fields

Acceptance Criteria:

1. Relevant courses
2. Prior work experience
3. Professional clubs
4. Major
5. GPA
6. Skills

7. References
8. Demographics

Owner Tiago Moore

USER STORY # 737 - REPORT TEAM MEMBERSHIP

Description:

- As a PI, I would like to know the membership of teams so I can generate the 6-month report.

Acceptance Criteria:

1. Report members of the teams.
2. Report all details necessary to the teams such as department, faculty/staff, and team names.

Owner: Tom Lam

USER STORY # 743 - PEER EVALUATIONS

Description:

- As a student I would like to able to enter my evaluation for every team member. All in order to create and maintain proper evaluation of team members.

Acceptance Criteria:

1. Students should have available a comment section.
2. There must be simple pre-made ratings. For ease of use.
3. View students in the project

Owner: Luis Puche

USER STORY # 744 - FEEDBACK TO STUDENTS

Description:

- As a faculty member in charge of a project, I would like to provide feedback to my students based on their performance throughout the semester so they are aware of what is expected of them as the semester continues.

Acceptance Criteria:

1. A faculty member in charge of a project can provide feedback to the students in his project.

2. Students participating in a project can receive and review the feedback sent to them by the faculty member.

Owner: Lizette Mendoza

USER STORY # 746 - MANAGING PROFILE

Description:

- As a user, I would like to have a profile page which I can access/edit information of my account, so that other students/faculty can view my information.

Acceptance Criteria:

1. Be able to edit account information and store when done.
2. Student is able to view their own profile and other students/faculty can view students profile as well.
3. Profile will retrieve information from registration form and registration form.
4. Using student's email and password, student will have access to view/edit their profiles.

Owner: Emmanuel Vinas

USER STORY # 745 - LOGIN ACCESS

Description:

- As any user (student, faculty member, pi/copi, or staff), after registered, I want to be able to sign in to the website.

Acceptance Criteria:

1. Google OAuth2 API should perform (token) authentication.
2. Different users should get different views of the webpage.
3. Other developers should have access to logged-in user information from anywhere in our app using an API call.

Owner: Lukas Borges

USER STORY # 747 - FACULTY MEMBER ACCESS PEER EVALUATION FORM

Description:

- As a faculty member, I would like to be able to access any Peer Evaluation Form so I can assess my students' progress.

Acceptance Criteria:

1. Evaluations viewed by Faculty must be anonymous
2. Faculty can access evaluations for any student at any time

Owner: Brenda Izquierdo

USER STORY # 748 - GENERAL FEEDBACK

Description:

- As a student or faculty member I would like to provide feedback on different areas of the overall VIP program so that the PI can evaluate the program.

Acceptance Criteria:

1. Questions will be populated from database.
2. Students and faculty may submit feedback.
3. Feedback will be stored anonymously.

Owner: Alex Sanchez

USER STORY # 752 - EVALUATIONS FACULTY INTERFACE

Description:

- As a faculty member in charge of a project, I want an interface to view all project and peer evaluations submitted by my students.

Acceptance Criteria:

1. A faculty member in charge of a project can review submitted evaluations to assess students progress.

Owner: Garrett Lemieux

USER STORY # 779 - PI REVIEW USER PARTICIPATION IN PROJECT

Description:

- As a PI, I would like to review each user's resume in order to a accept or reject a user into a project he or she has applied to.

Acceptance Criteria:

1. PI can view all projects and users that applied for it , when PI chooses a user to review then resume can be viewed.
2. PI makes a decision on users participation in project when he or she proceeds to express choice then the PI can choose either accept or decline statements.
3. PI can express his or her reasons for accepting or rejecting user when he or she clicks additional comments and then can submit comments.

Owner: Sean Iamartino

USER STORY # 788 - PROPOSAL APPROVALS

Description:

- As a PI, I would like to view all proposals so I can accept/reject/comment.

Acceptance Criteria:

1. I can view list of proposals
2. I can accept/reject each proposal
3. I can make a comment on a proposal

Owner: Lucas Iglesias

USER STORY # 789 - AUTOMATED EMAILS

Description:

- As a PI, I would like to send automated notifications when an event is created.

Acceptance Criteria:

1. The selected recipient receives an automated email with the details of the event.

Related Tasks:

As per Sadjadi, create a mailing service that can be used by team members to send automated emails.

Owner: Eduardo Guerra

USER STORY # 790 - EVALUATION QUESTIONS

Description:

- As a PI, I would like to be able to create evaluation questions. So that peer and project evaluations can have a question bank.

Acceptance Criteria:

1. Selection on type of question. Range, True and False, and comment
2. Selection on evaluation is for Peers or Projects
3. Confirmation message upon submission.

Owner: Luis Puche

Project Plan

The hardware and software components that enabled us to build the VIP website are listed below. Following that are our Sprint plans throughout the course of the semester. The Sprint Plans occurred biweekly.

Hardware and Software Resources

Our system is running the Bitnami MEAN Stack. It runs on any operating system, including Linux servers, Mac and Windows. We currently are using a Linux server provided by FIU. The MEAN Stack provides a complete development environment, including:

MongoDB - Repository
Express JavaScript
Angular JavaScript
Node JavaScript

In addition to the Bitnami MEAN Stack, the package.json consists of the latest modules running on the server and their versions. Core modules utilized in the server include:

Express - Back-end routing
Morgan - debugging
Mongoose - communicate with the database
JWT-simple - JSON web token
nodemailer - email system
UI-Router - front-end routing
Bootstrap - CSS skeleton for front-end (http://getbootstrap.com/)
jQuery - JavaScript library
Satellizer - Google Authentication/logins

In order to properly develop this project, we used the following development tools:

Jetbrains Webstorm - JavaScript IDE
Mozilla Firefox - Internet browser with an excellent developer console
StarUML - Constructing UML diagrams
Putty (Windows users) - SSH terminal to gain access to the Linux server

Some students opted to using a Virtual Machine to run the Bitnami MEAN Stack rather than running natively. Lukas Borges wrote a tutorial on what was needed to set that up.

Sprints Plan

The user stories were all approved in Sprint 3, and development began in Sprint 4. To see the user stories in detail, please see the “Implemented User Stories” under the heading “User Stories”.

Sprint 0

(08/26/2015 - 09/09/2015)

- Determine time limits on how much work each team member will put towards the project each week.
- Become familiar with Scrum and Agile.
- Come up with a system design
- Agree upon a common development environment for team members
    - *IDE, UML software, etc.*

Sprint 1

(09/09/2015 - 09/23/2015)

- Split up into groups, and have each Scrum Master announce which user story they are claiming. (Preferably by the end of the day, email professor with choice)
    - *All of the user stories are up on Mingle.*
- New projects will be created in Mingle for each of the groups.
- This is going to be an incremental process

Keep in mind:

At the end of the day (semester), we all want to be able to showcase every aspect of Software Engineering that we applied here.

In addition, our documentation and work needs to be understandable for whomever works on this project in the future.

Once we're in groups, we don't need to have a major accomplishment every day.

Sprint 2

(09/23/2015 - 10/07/2015)

- Submit user stories for final approval
    - Post stories on Mingle once approved
    - Stories must relate to whatever epic your team is working on
- Develop proper documentation for approved user stories
    - Required UML Diagrams
        - Use case
        - Sequence
        - Class
- Determine what architecture and stack to use for project

Sprint 3

(10/07/2015 - 10/21/2015)

- Finish a basic skeleton for the website, and push to *master* and *group* branches.
    - Once this is done, we can just pull from our group's branch and begin implementation
- Begin development on approved user stories
    - Discuss how we'll create our database schemas for any stories that will share collections in our database
- Continue working on diagrams if they are not done yet. Submit to Sadjadi to look over them if you need help.

Sprint 4

(10/21/2015 - 11/04/2015)

- Work on individual user story implementation on local branches.

Sprint 5

(11/04/2015 - 11/18/2015)

- Continue and complete what you can for you individual user stories.
- Get ready to integrate with other members for next sprint (Think about what you need from others and what others may need from you).

Sprint 6

(11/18/2015 - 11/30/2015)

- Work on integration for this sprint.
    - The next show and tell will *only* accept code that is running on our live server, *vip-dev.cis.fiu.com*
    - Integrate within your groups first, then each group Scrum Master will come together during the scrum of scrums to integrate the 4 groups together onto the Master repository
    - Talk to your scrum master for integration deadlines

Sprint 7

(11/30/2015 - 12/11/2015)

- No more class lectures/meetings after 11/30, since we still have the room you may show up to continue working with others.
- Development (any code) must be finished by 12/4.
    - Push your changes to your group's branch for integration.
    - If you miss this date, you are responsible for integration on your own (Master branch)
        - Channel on slack will be created to notify whomever else is in there that you are making changes to the Master branch.
        - You are responsible for any issues that may occur during integration onto Master
    - Steven is going to do a final pull to our server on 12/6 at midnight.
        - Any code submitted after this date will not be reflected on the webpage.
- Documentation (UML diagrams) must be sent to Lizette by 12/3 for compilation.

- Videos (requirements on Moodle) must be completed and sent to Tiago by 12/10.
    - There is a common YouTube account we will all post our videos to.

SYSTEM DESIGN

System design discusses which architecture we are using as well as how we apply that architecture to our web development stack. The system design also contains information about the actual system and subsystems which were implemented via user stories.

Architectural / Design Patterns

Overview of Modules / Components

Model-View-Controller

The Model-View-Controller is the software architectural pattern used in the project. The architecture can be broken down as follows:

The model consists of routes (all of our API's) which communicates with the Mongo Database. The controller consists of both controllers and services. By means of the Angularjs Framework UI-Router, views can be routed to specific URLs. A view includes all the HTML pages and a combination with our controllers (MEAN Stack provides two way data-binding for the controller and view).

Core Components

Controllers

The controller in the MEAN Stack is a core component. It communicates with the front-end views as well as calling related services.

Services

Services are the connection of the controllers to the API's.

Back-End Routes (Express/NodeJS)

Back-end routes include all API's which connect to the Mongo Database, called by services.

Front-End Routes (appRoutes.js)

appRoutes provides a location for implementing HTML pages. Since MEAN Stack is predominantly used for single page applications, it was developed as such. appRoutes simply injects HTML pages into the home page (index.html).

System and Subsystem Decomposition

Registration

Upon mutual agreement, it was decided not to use any form of the FIU PantherSoft API (which is virtually non-existent). Unfortunately, that meant no database to confirm the status of the registering users for the VIP project. In this sense, another method was needed in order to verify the user as a current FIU member without checking directly with FIU APIs. This was done by means of email and PI verification covered below.

Please refer to Appendix A - UML Diagrams under the subheading “Registration” for more information about the in depth functionalities performed by this this subsystem.

Student Verification

Verification for a student is quite a simple task. If a guest user registers as a student, he will be emailed at his @fiu.edu account. If a guest user attempts to register using another domain and not @fiu.edu, the user will be forced to register using the FIU email. Once registered, the user can click a link emailed to him to verify the account as a student.

Faculty/Staff and PI/CoPI Verification

Similar to the student process, faculty/staff and PI/CoPI user types must also verify by email. However, a student is the lowest form of user type in the system, so any FIU member can sign up as a student. In the case of a faculty/staff member or PI/CoPI registration, the verification link will be sent to the PI of the project. This verification link allows the PI to delete the user if he is registering under the wrong user type, or where the PI can approve the user of the user type. After this approval button is selected by the PI, the user will finally be registered as a faculty/staff member or PI/CoPI.

Proposals, Submissions, and 6 Month Reporting

For proposals: a system was created to enable faculty and staff user types to submit proposals to the PI for approval. Added features include allowing faculty members to edit submitted proposals and receive notifications on any approval status changes. A system was also created to allow PI user types to view a collection of all proposals, view each proposal individually in detail, and approve/deny submitted proposals as well as add notes to them for the faculty member to view.

With the 6 month reporting, we provided a questionnaire for the PI to fill out every 6 months. The report includes the progress of the project or if any complications arose during the project.

A functional page was designed and implemented so that the PI would have the rights and ability to assign group memberships to users. The PI can move students/faculty around from group to group and has the ability to unassign students/faculty from a group.

Please refer to Appendix A - UML Diagrams under the subheading “Proposals, Submissions, and 6 Month Reporting” for more information about the in depth functionalities performed by this this subsystem.

Evaluations

This subsystem deals with any function related to evaluations of peers or projects alike. We have implemented methods for PIs to create questions that faculty members can use in their evaluation forms. Project evaluations completed by faculty members in charge of a project are sent directly to the corresponding student’s email to receive immediate feedback. Evaluations, both peer and project, are stored in the database to be retrieved for the students and others who require access to it.

Please refer to Appendix A - UML Diagrams under the subheading “Evaluations” for more information about the in depth functionalities performed by this this subsystem.

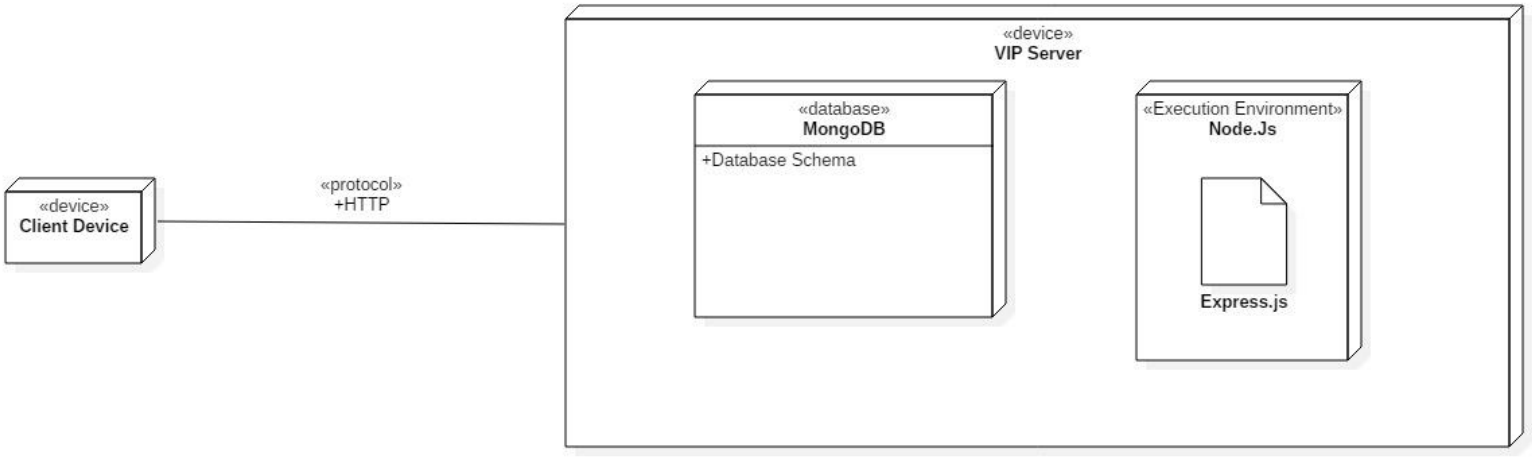
Announcements

Announcements subsystem handles all the mailing and notifications functionalities of the system.

A mailService was created to route all mail requests to the nodemailer mail function. A saveNotifications function was created to store the information of a notification in the database. A subscription function was created, which provides a checkbox under the profile page that allows the student to subscribe to FIU VIP in order to receive emails. A saveEvent function was created in the controller to store the event information in the database and email the subscribers stored in the database. An event function was created to retrieve the event information to display to the user. A deleteEvent function was also created so that the user can delete an old or incorrect event.

Please refer to Appendix A - UML Diagrams under the subheading “Announcements” for more information about the in depth functionalities performed by this this subsystem.

Deployment Diagram



GLOSSARY

User Types

- Student - Registered student or prospective student
- Faculty - Head of a specific project
- Staff - Added at the product owners request
- PI/CoPI - Principal Investigator and Co-Principal Investigator

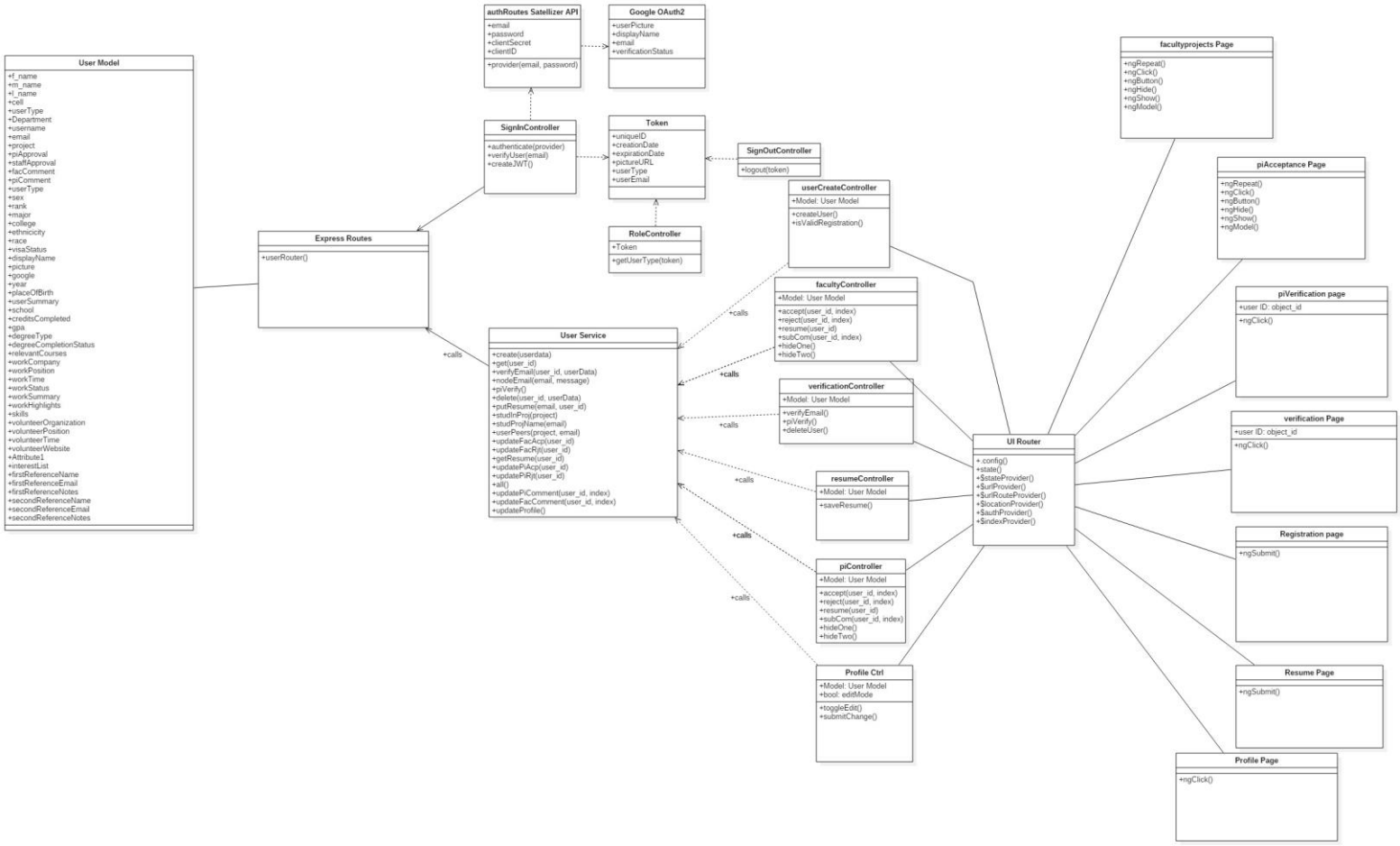
APPENDIX

Appendix A - UML Diagrams

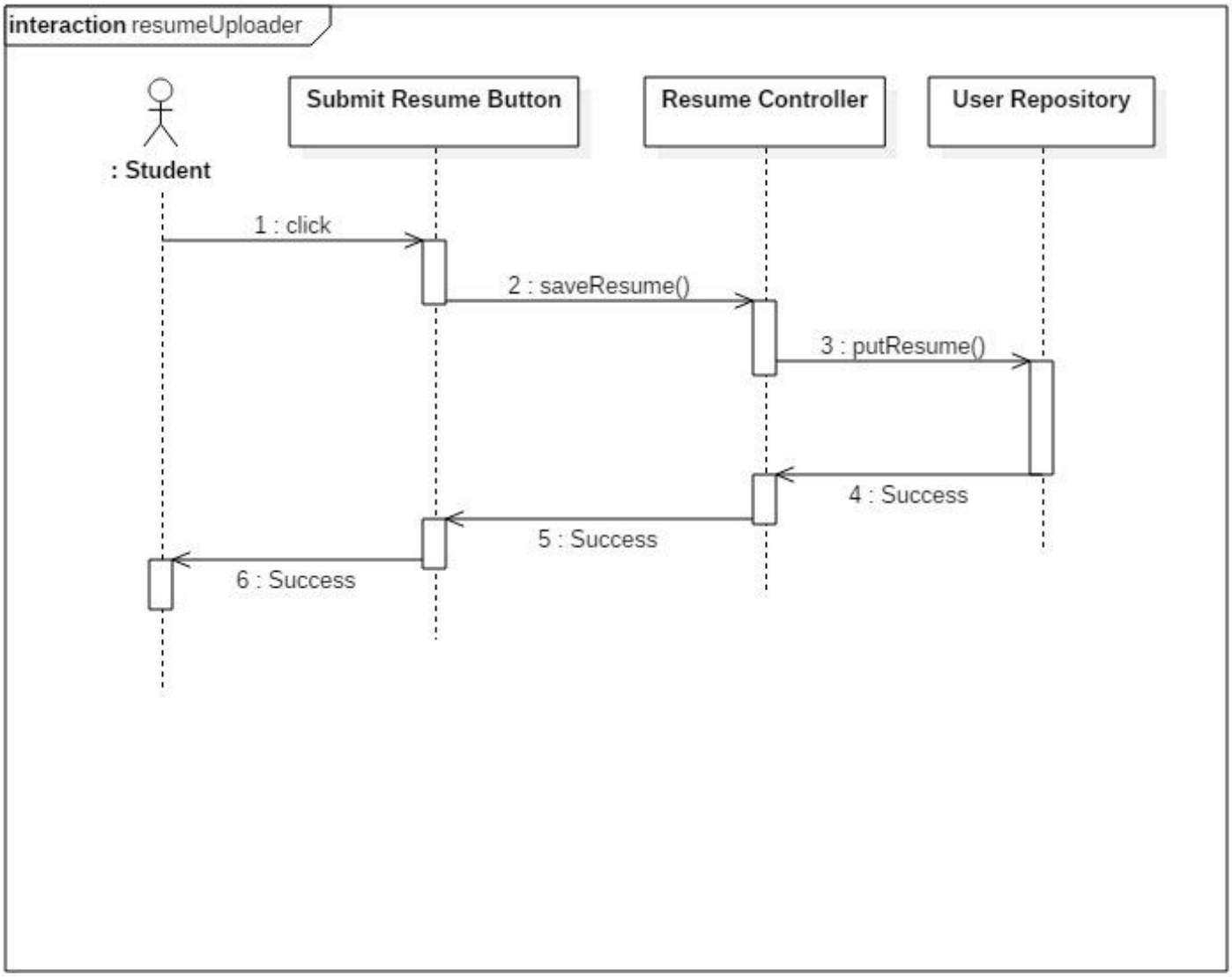
Registration

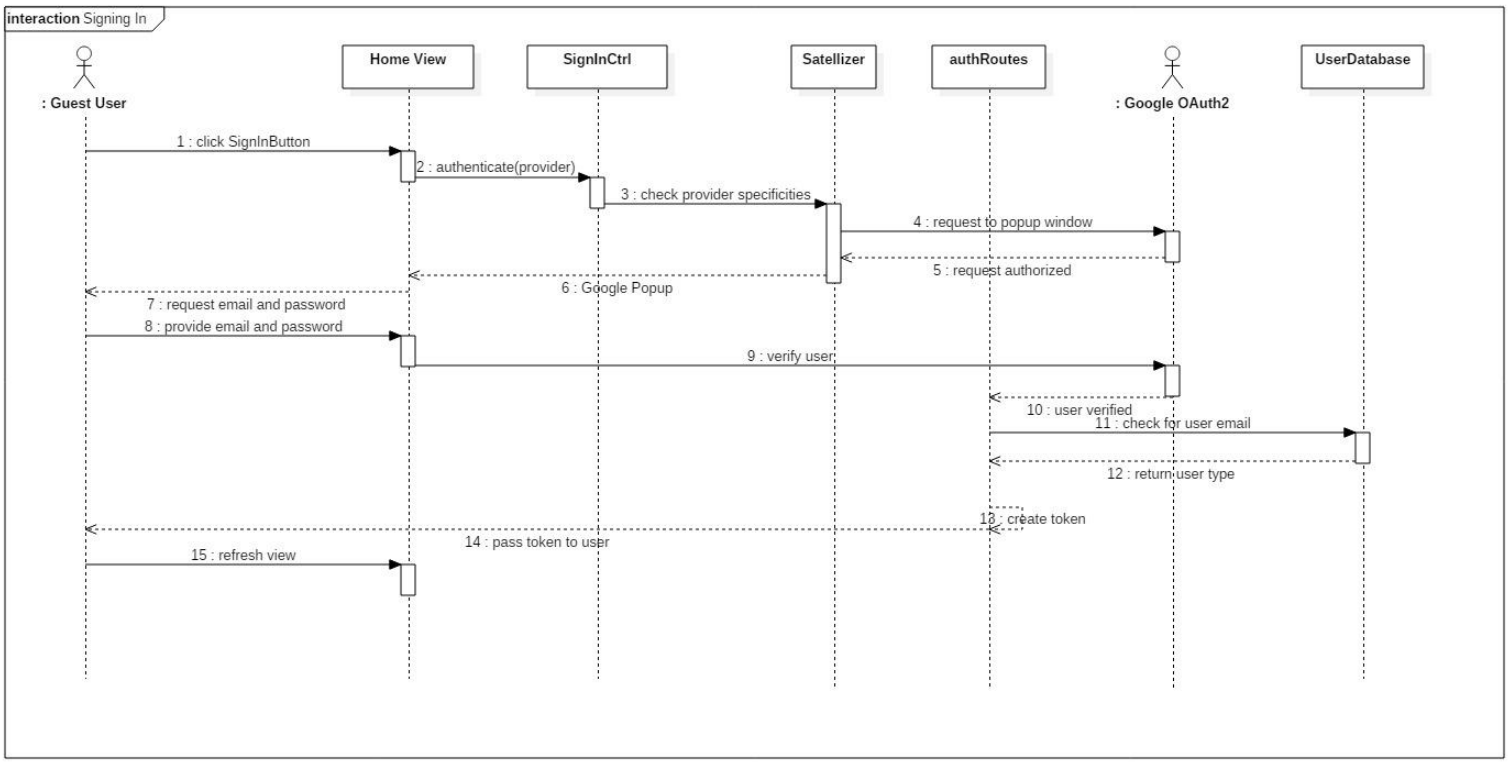
Use Case Diagram

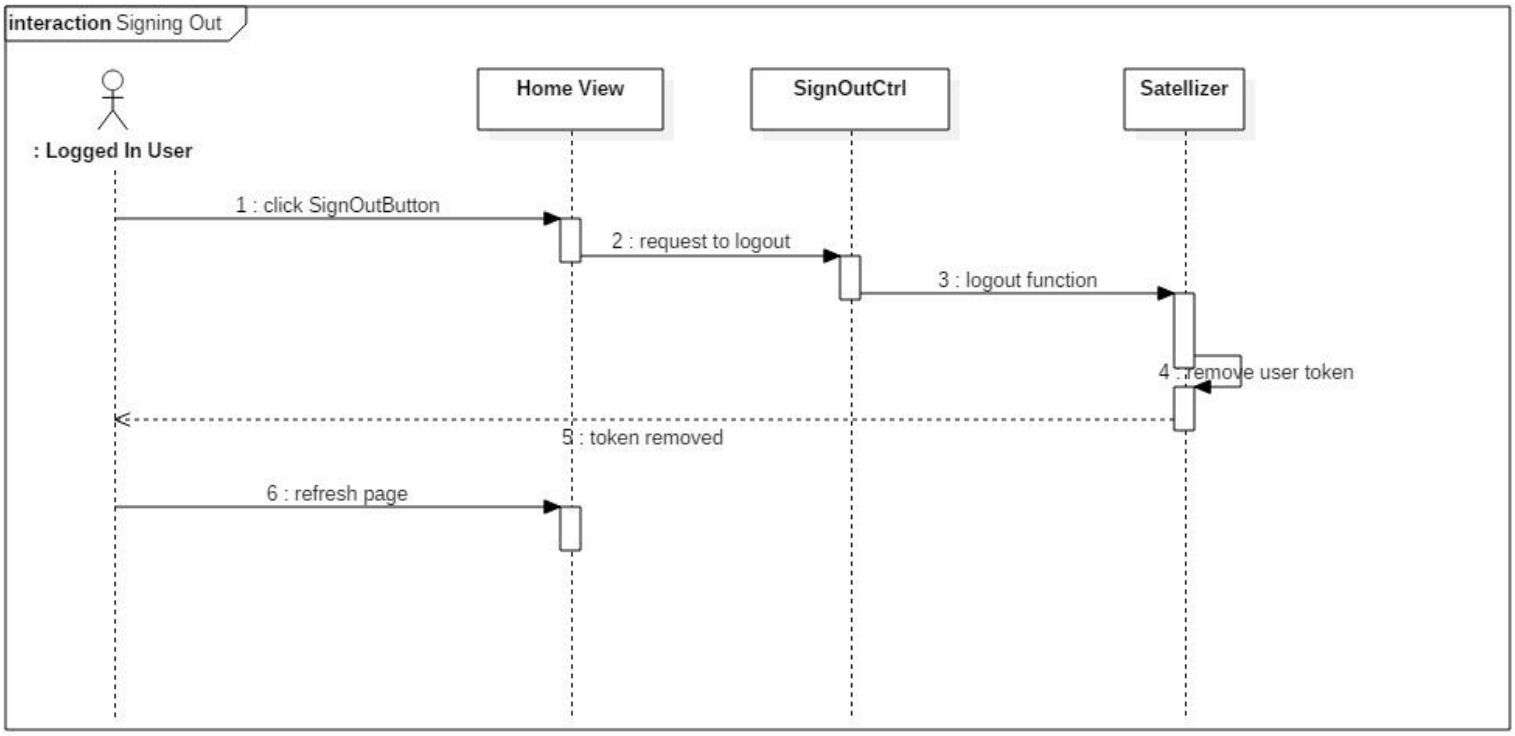
Class Diagram

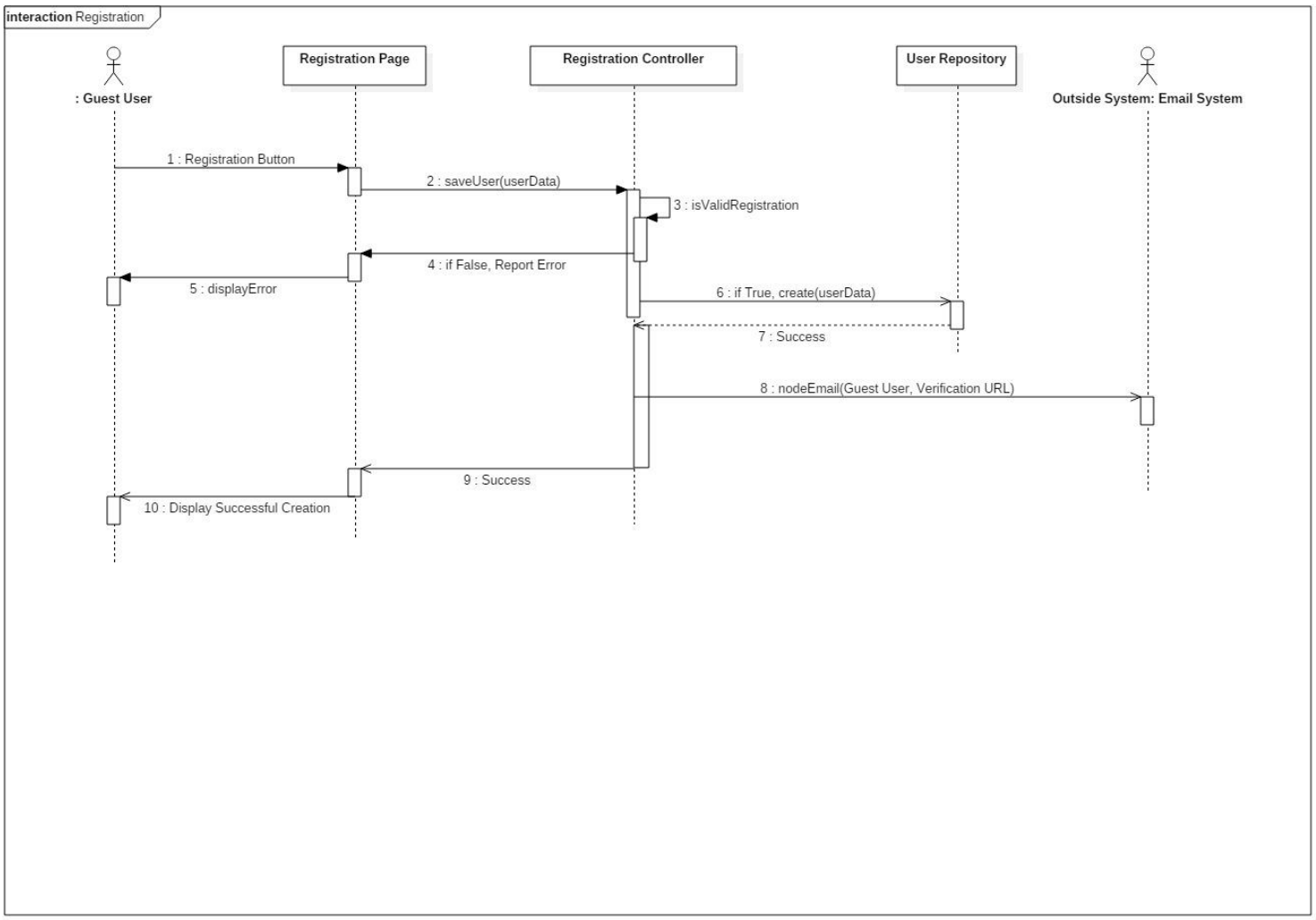


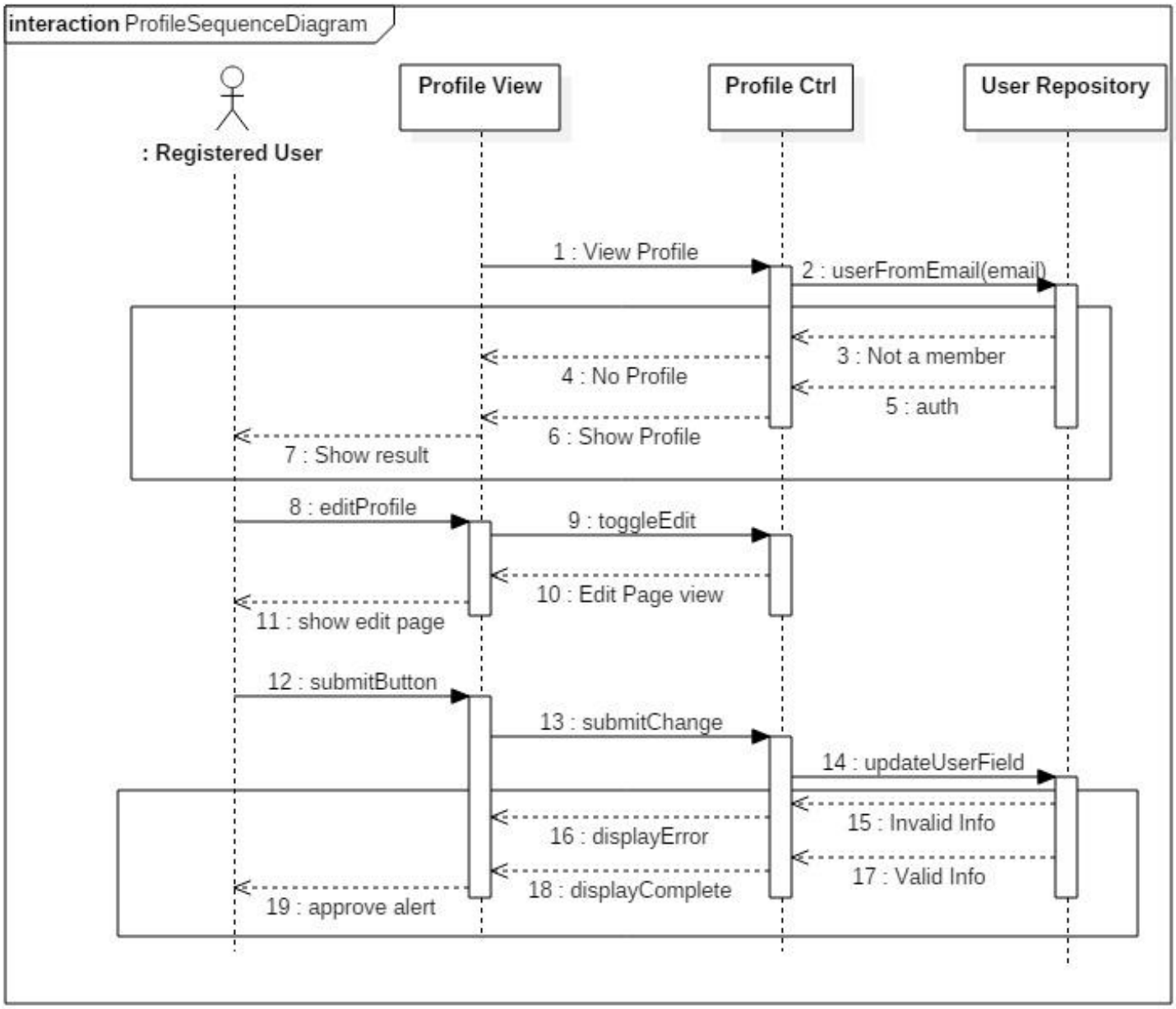
Sequence Diagram

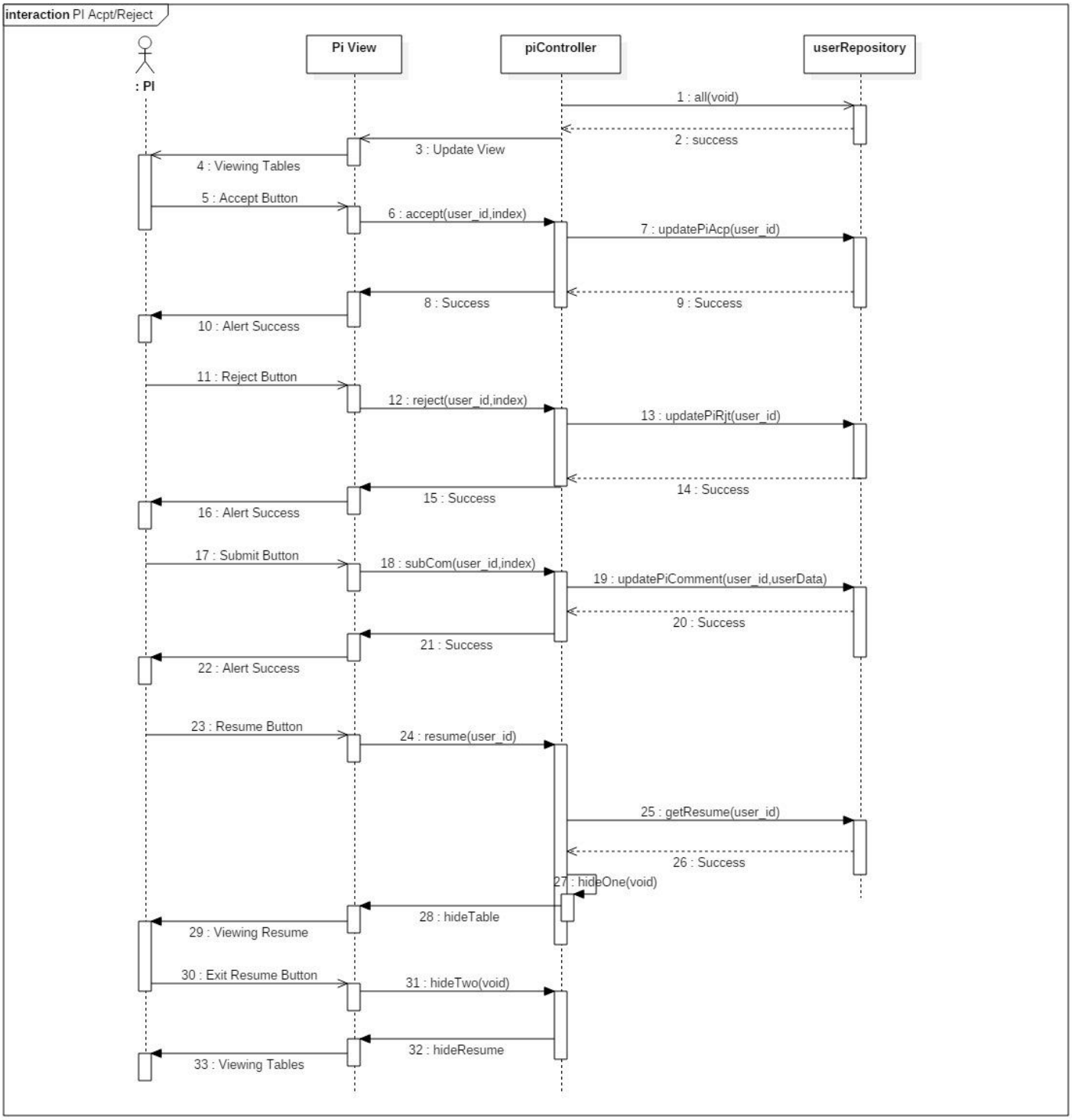


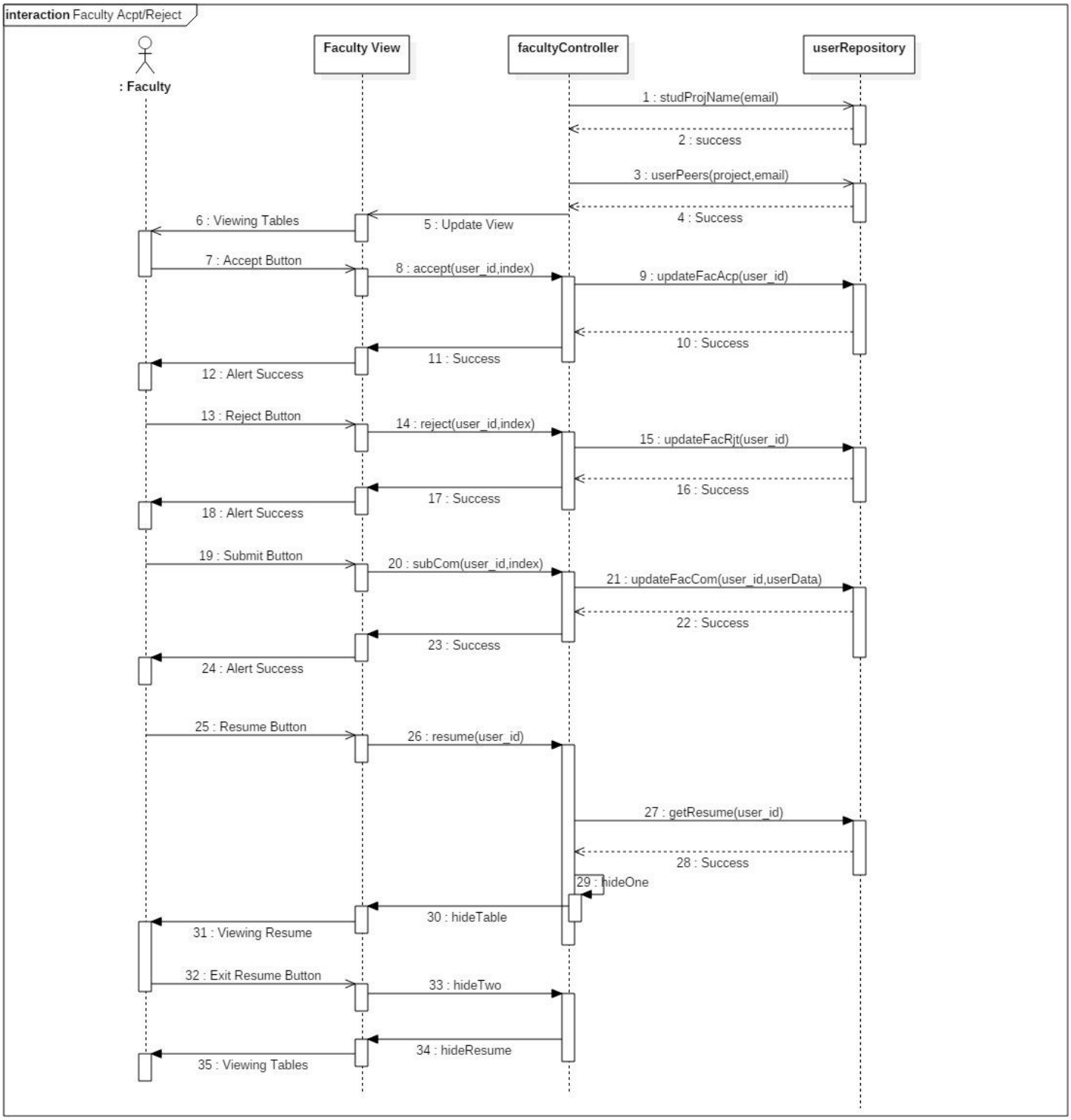

interaction Signing In
: Guest User
Home View
SignInCtrl
Satellizer
authRoutes
: Google OAuth2
UserDatabase
1 : click SignInButton
2 : authenticate(provider)
3 : check provider specificities
4 : request to popup window
5 : request authorized
6 : Google Popup
7 : request email and password
8 : provide email and password
9 : verify user
10 : user verified
11 : check for user email
12 : return user type
13 : create token
14 : pass token to user
15 : refresh view

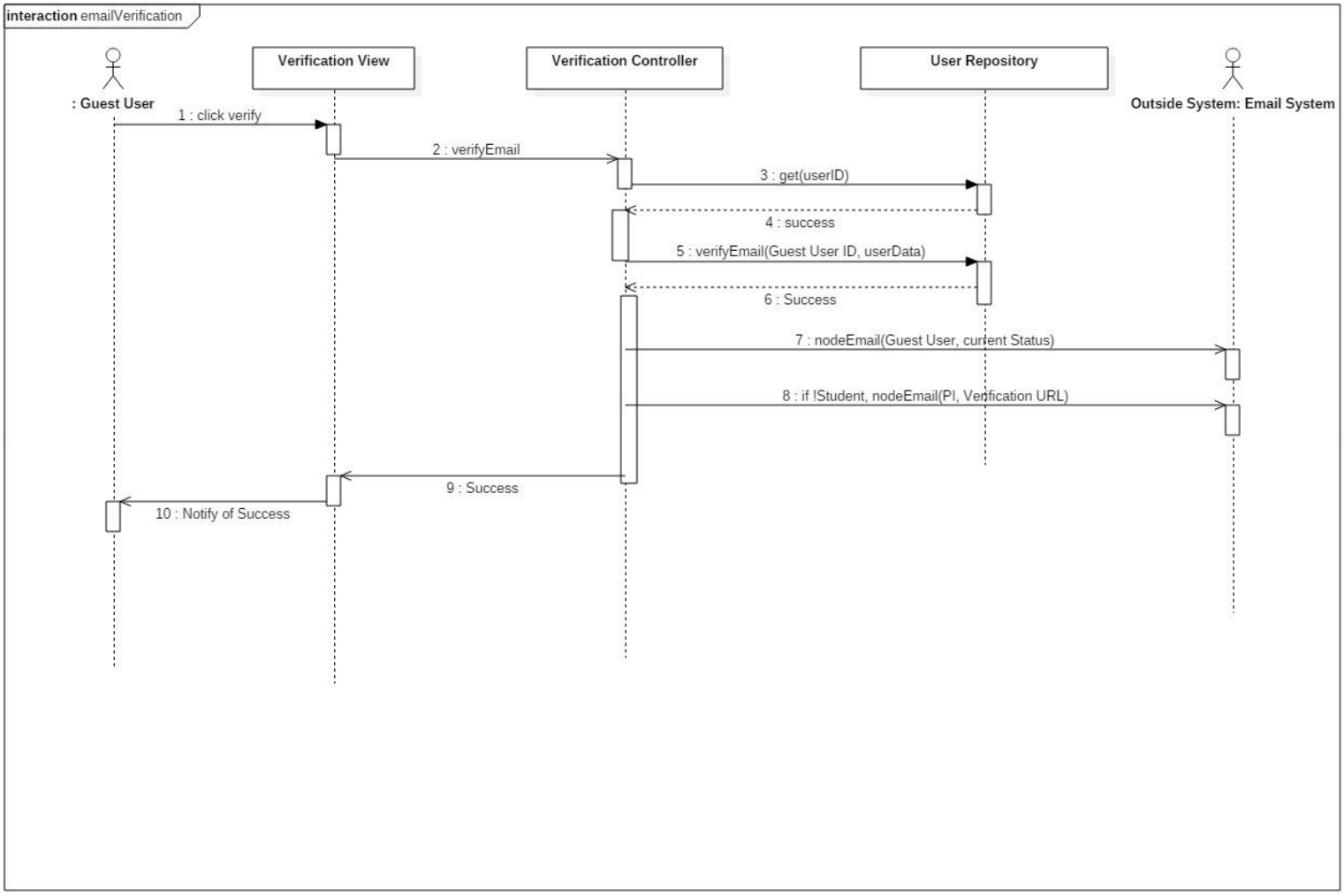

interaction Signing Out
: Logged In User
Home View
SignOutCtrl
Satellizer
1 : click SignOutButton
2 : request to logout
3 : logout function
4 : remove user token
5 : token removed
6 : refresh page

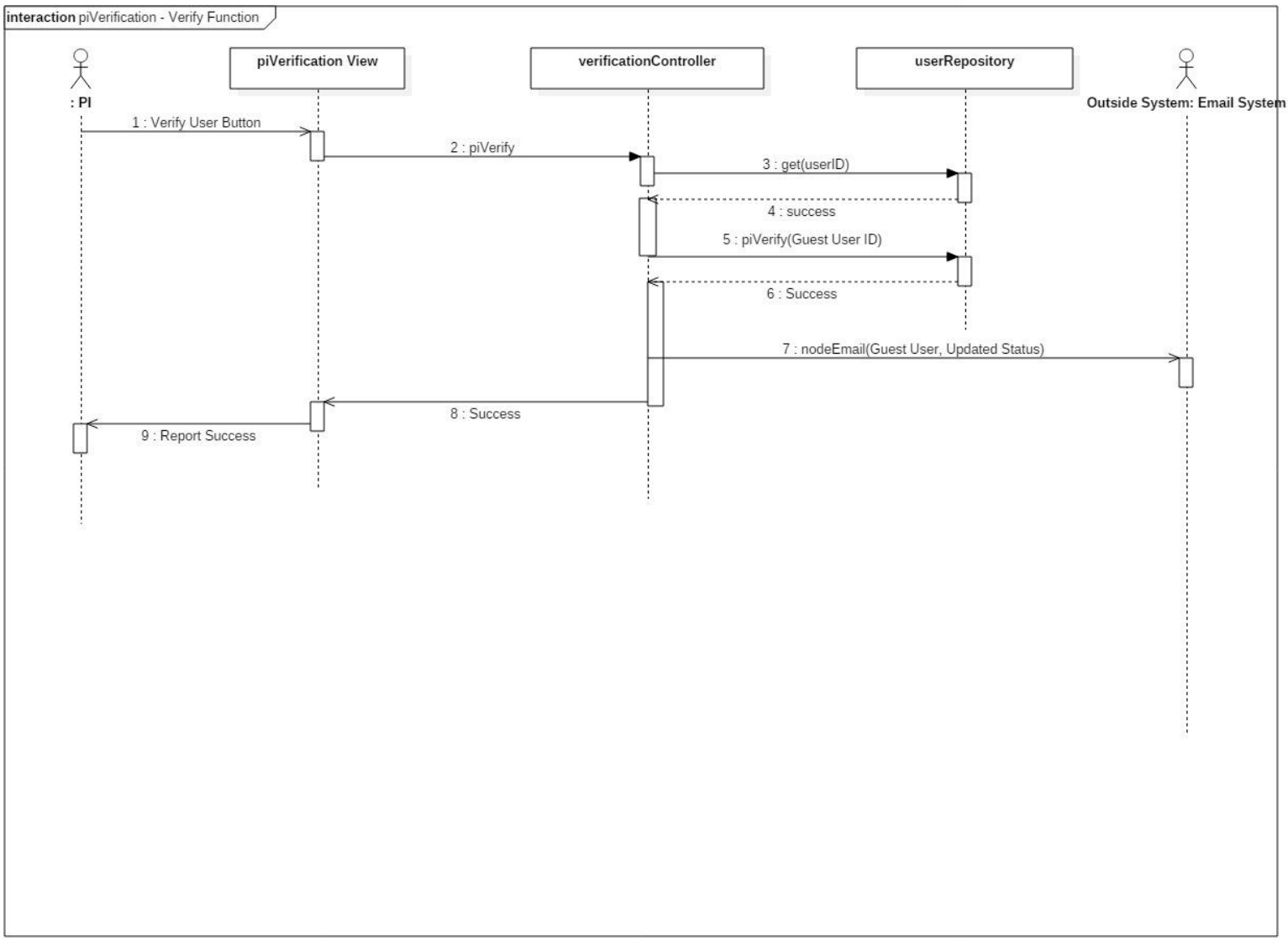

interaction Registration
: Guest User
Registration Page
Registration Controller
User Repository
Outside System: Email System
1 : Registration Button
2 : saveUser(userData)
3 : isValidRegistration
4 : if False, Report Error
5 : displayError
6 : if True, create(userData)
7 : Success
8 : nodeEmail(Guest User, Verification URL)
9 : Success
10 : Display Successful Creation

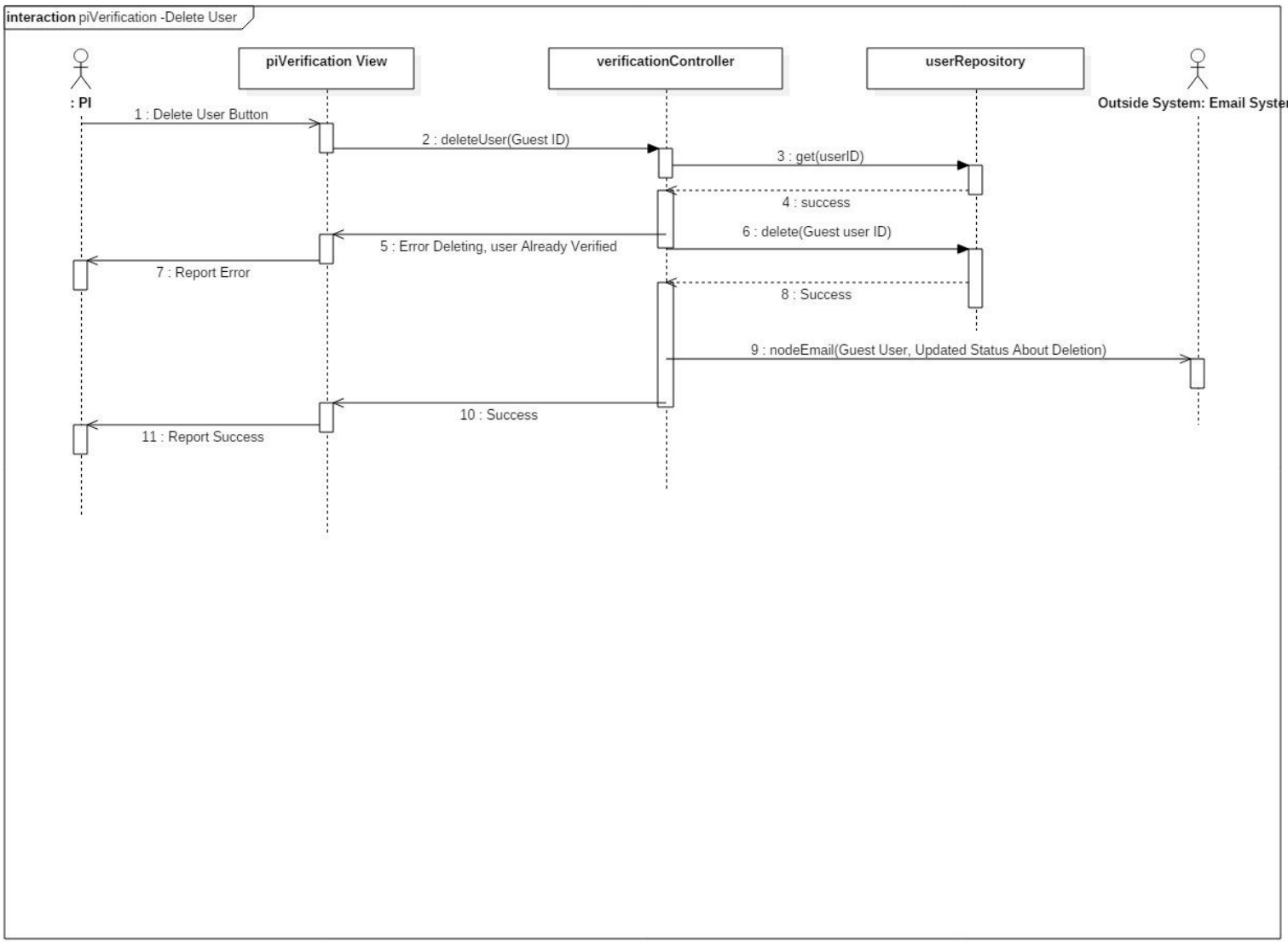

interaction ProfileSequenceDiagram
: Registered User
Profile View
Profile Ctrl
User Repository
1 : View Profile
2 : userFromEmail(email)
3 : Not a member
4 : No Profile
5 : auth
6 : Show Profile
7 : Show result
8 : editProfile
9 : toggleEdit
10 : Edit Page view
11 : show edit page
12 : submitButton
13 : submitChange
14 : updateUserField
15 : Invalid Info
16 : displayError
17 : Valid Info
18 : displayComplete
19 : approve alert


interaction PI Acpt/Reject
: PI
Pi View
piController
userRepository
1 : all(void)
2 : success
3 : Update View
4 : Viewing Tables
5 : Accept Button
6 : accept(user_id,index)
7 : updatePiAcp(user_id)
8 : Success
9 : Success
10 : Alert Success
11 : Reject Button
12 : reject(user_id,index)
13 : updatePiRjt(user_id)
14 : Success
15 : Success
16 : Alert Success
17 : Submit Button
18 : subCom(user_id,index)
19 : updatePiComment(user_id,userData)
20 : Success
21 : Success
22 : Alert Success
23 : Resume Button
24 : resume(user_id)
25 : getResume(user_id)
26 : Success
27 : hideOne(void)
28 : hideTable
29 : Viewing Resume
30 : Exit Resume Button
31 : hideTwo(void)
32 : hideResume
33 : Viewing Tables


interaction Faculty Acpt/Reject
: Faculty
Faculty View
facultyController
userRepository
1 : studProjName(email)
2 : success
3 : userPeers(project,email)
4 : Success
5 : Update View
6 : Viewing Tables
7 : Accept Button
8 : accept(user_id,index)
9 : updateFacAcp(user_id)
10 : Success
11 : Success
12 : Alert Success
13 : Reject Button
14 : reject(user_id,index)
15 : updateFacRjt(user_id)
16 : Success
17 : Success
18 : Alert Success
19 : Submit Button
20 : subCom(user_id,index)
21 : updateFacCom(user_id,userData)
22 : Success
23 : Success
24 : Alert Success
25 : Resume Button
26 : resume(user_id)
27 : getResume(user_id)
28 : Success
29 : hideOne
30 : hideTable
31 : Viewing Resume
32 : Exit Resume Button
33 : hideTwo
34 : hideResume
35 : Viewing Tables

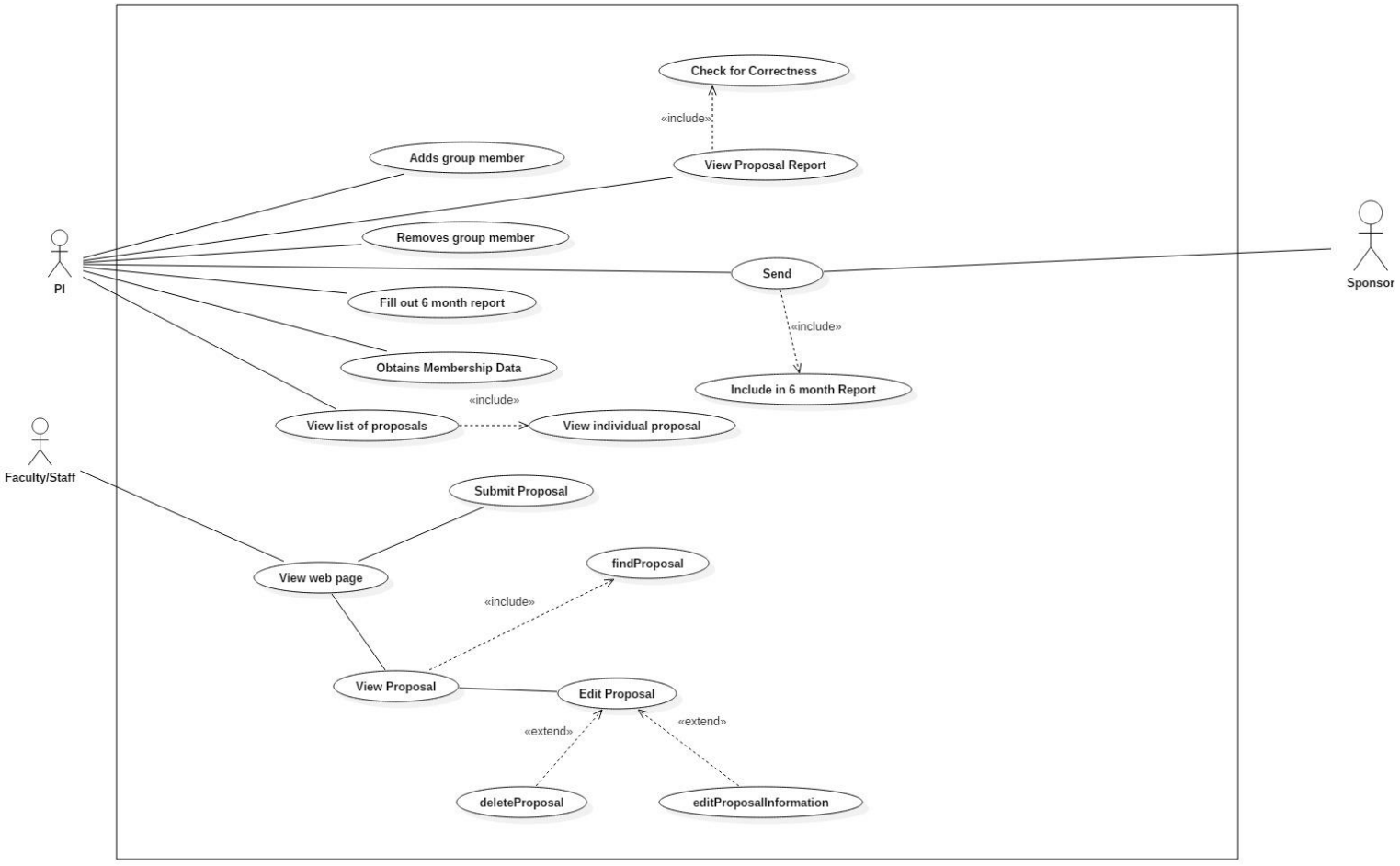

interaction emailVerification
: Guest User
Verification View
Verification Controller
User Repository
Outside System: Email System
1 : click verify
2 : verifyEmail
3 : get(userID)
4 : success
5 : verifyEmail(Guest User ID, userData)
6 : Success
7 : nodeEmail(Guest User, current Status)
8 : if !Student, nodeEmail(PI, Verification URL)
9 : Success
10 : Notify of Success


interaction piVerification - Verify Function
: PI
piVerification View
verificationController
userRepository
Outside System: Email System
1 : Verify User Button
2 : piVerify
3 : get(userID)
4 : success
5 : piVerify(Guest User ID)
6 : Success
7 : nodeEmail(Guest User, Updated Status)
8 : Success
9 : Report Success

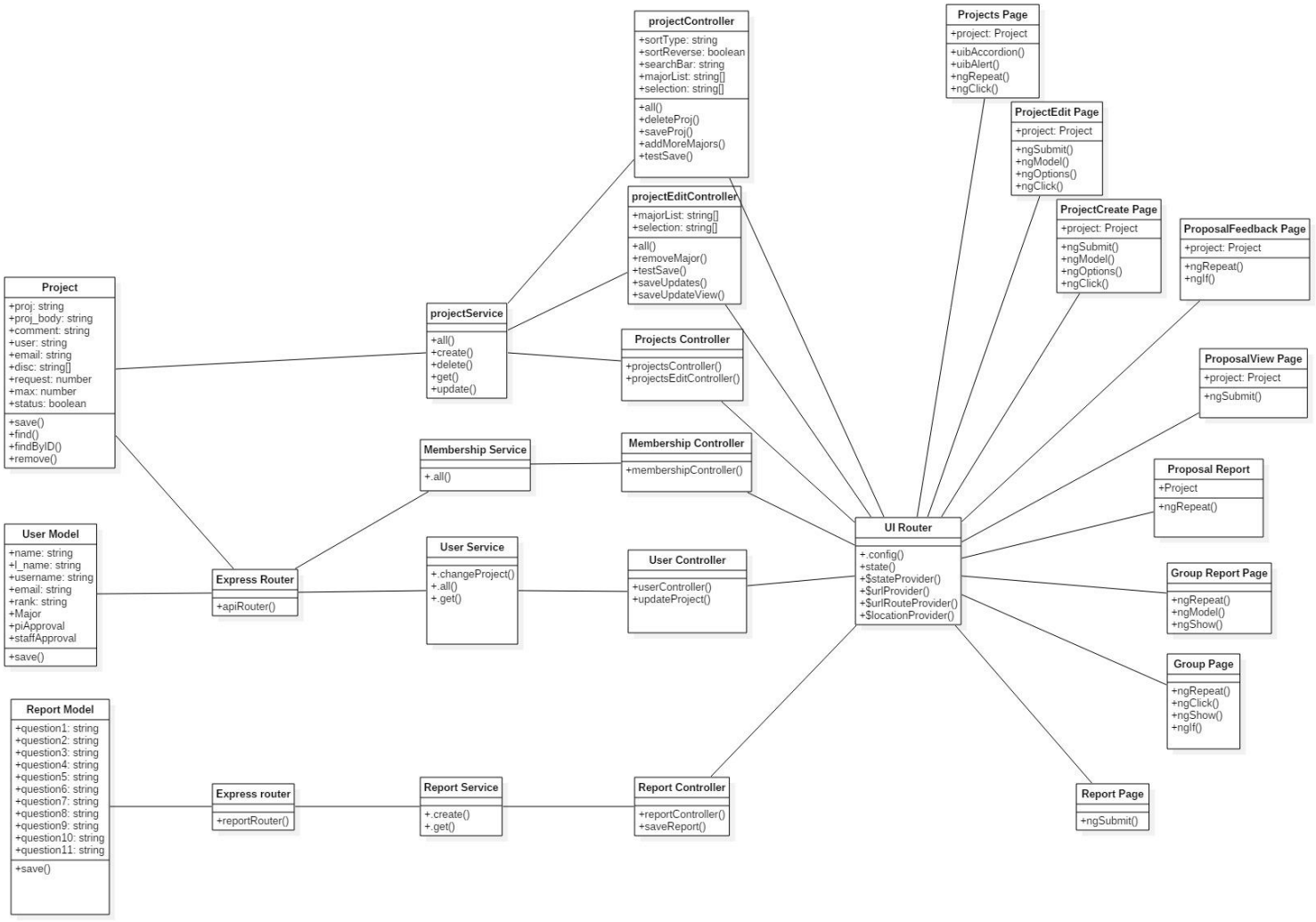

interaction piVerification -Delete User
: PI
piVerification View
verificationController
userRepository
Outside System: Email System
1 : Delete User Button
2 : deleteUser(Guest ID)
3 : get(userID)
4 : success
5 : Error Deleting, user Already Verified
6 : delete(Guest user ID)
7 : Report Error
8 : Success
9 : nodeEmail(Guest User, Updated Status About Deletion)
10 : Success
11 : Report Success

Proposals, Submissions, and 6 Month Reporting

Use Case Diagram

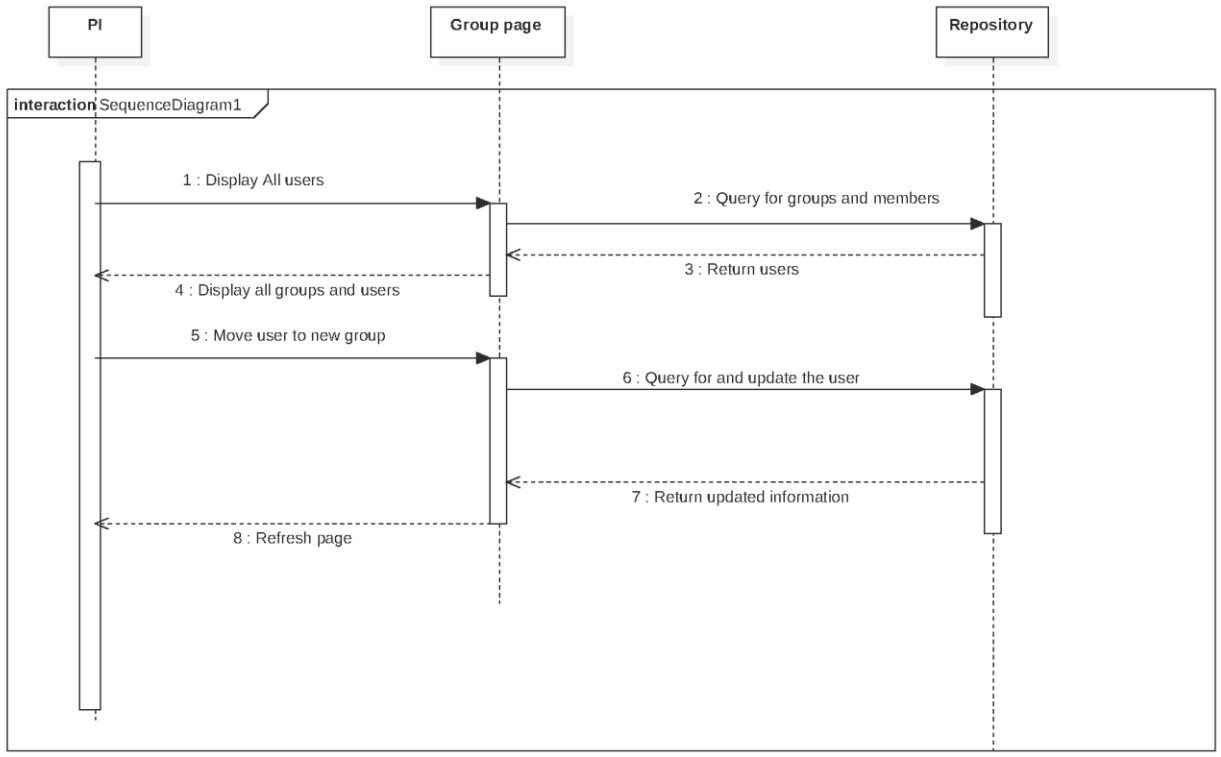


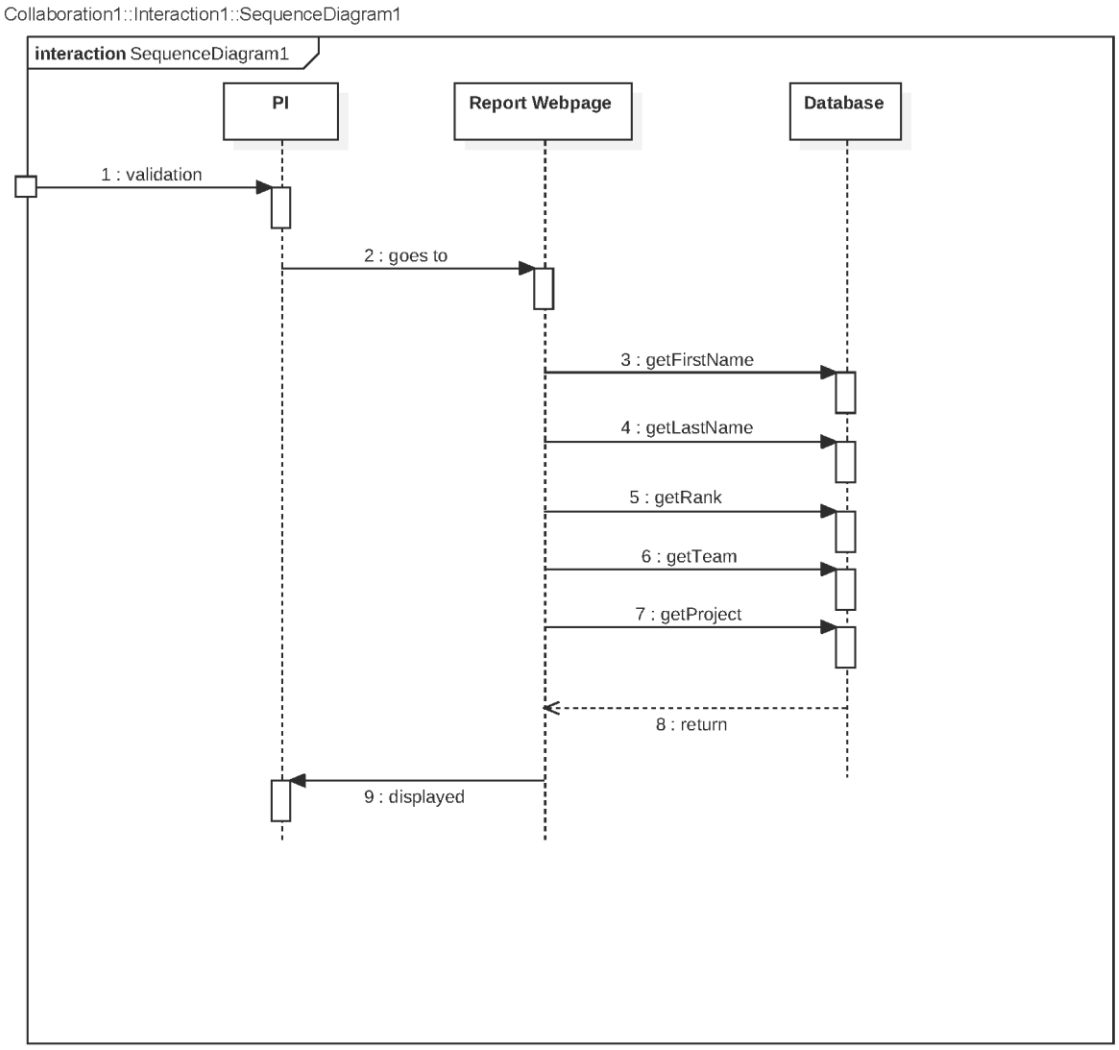
Class Diagram

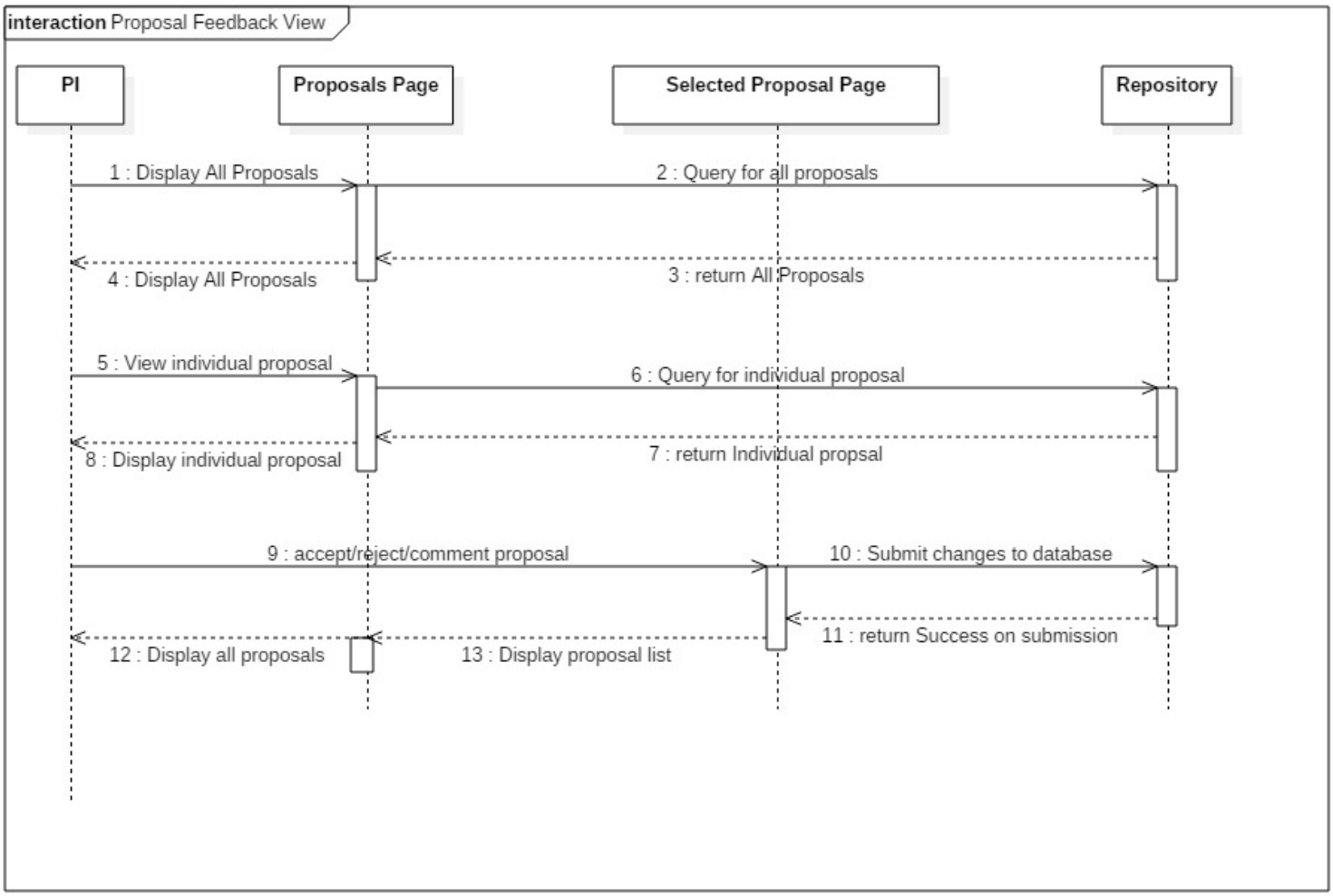


Sequence Diagram

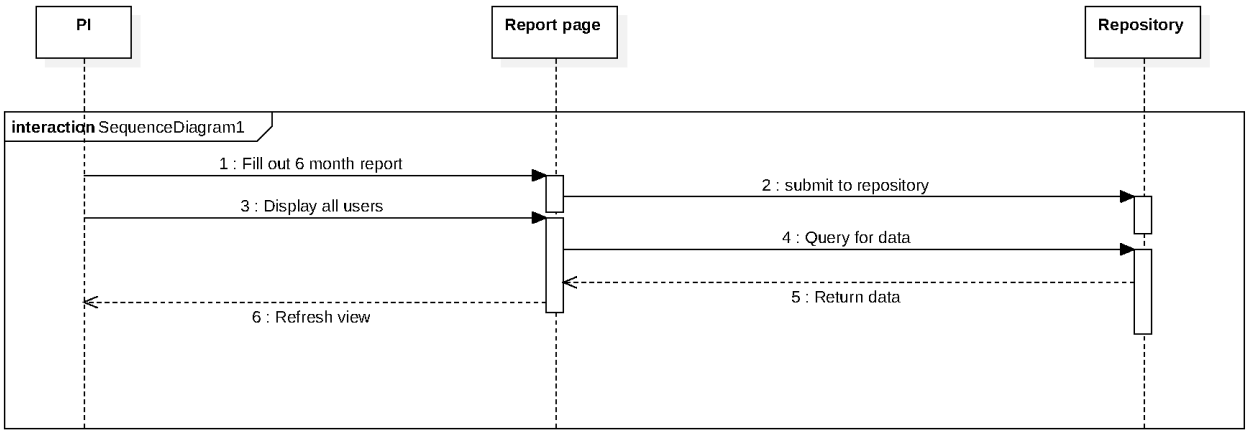
Collaboration1::Interaction1::SequenceDiagram1

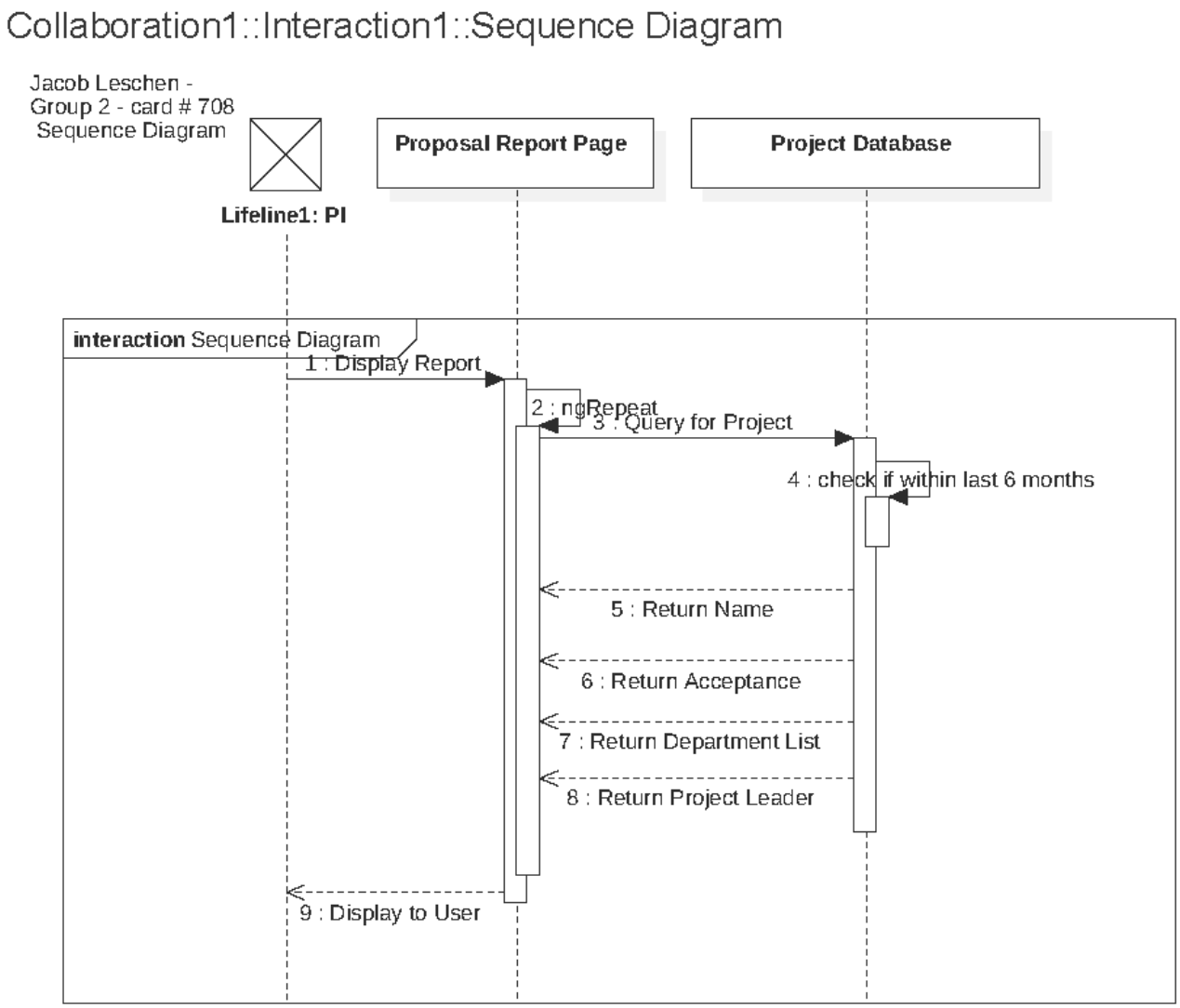



Collaboration1::Interaction1::SequenceDiagram1
interaction SequenceDiagram1
PI
Report Webpage
Database
1 : validation
2 : goes to
3 : getFirstName
4 : getLastName
5 : getRank
6 : getTeam
7 : getProject
8 : return
9 : displayed

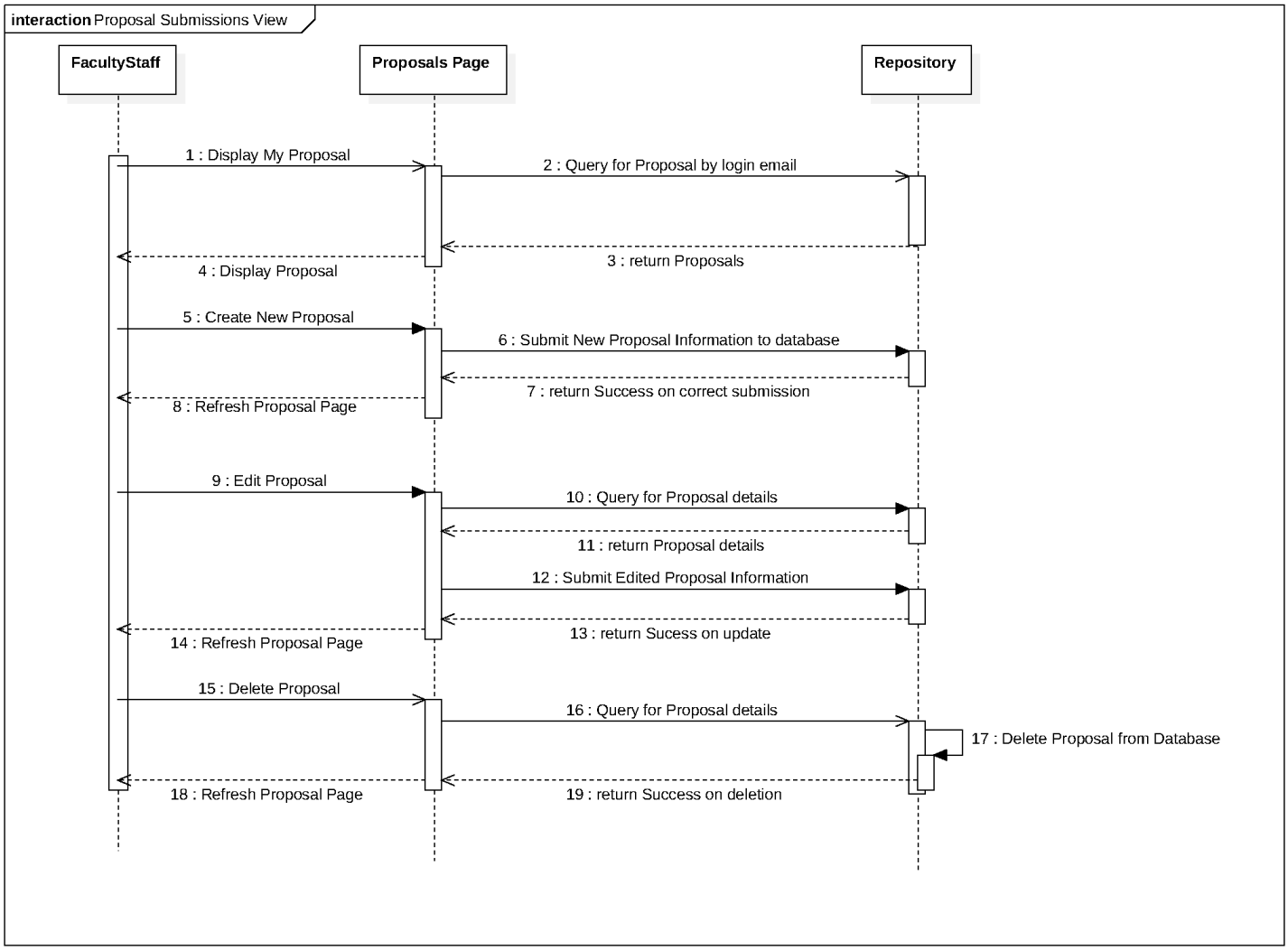

interaction Proposal Feedback View
PI
Proposals Page
Selected Proposal Page
Repository
1 : Display All Proposals
2 : Query for all proposals
4 : Display All Proposals
3 : return All Proposals
5 : View individual proposal
6 : Query for individual proposal
8 : Display individual proposal
7 : return Individual propsal
9 : accept/reject/comment proposal
10 : Submit changes to database
11 : return Success on submission
12 : Display all proposals
13 : Display proposal list

Collaboration1::Interaction1::SequenceDiagram1




Collaboration1::Interaction1::Sequence Diagram
Jacob Leschen -
Group 2 - card # 708
Sequence Diagram
Lifeline1: PI
Proposal Report Page
Project Database
interaction Sequence Diagram
1 : Display Report
2 : ngRepeat
3 : Query for Project
4 : check if within last 6 months
5 : Return Name
6 : Return Acceptance
7 : Return Department List
8 : Return Project Leader
9 : Display to User

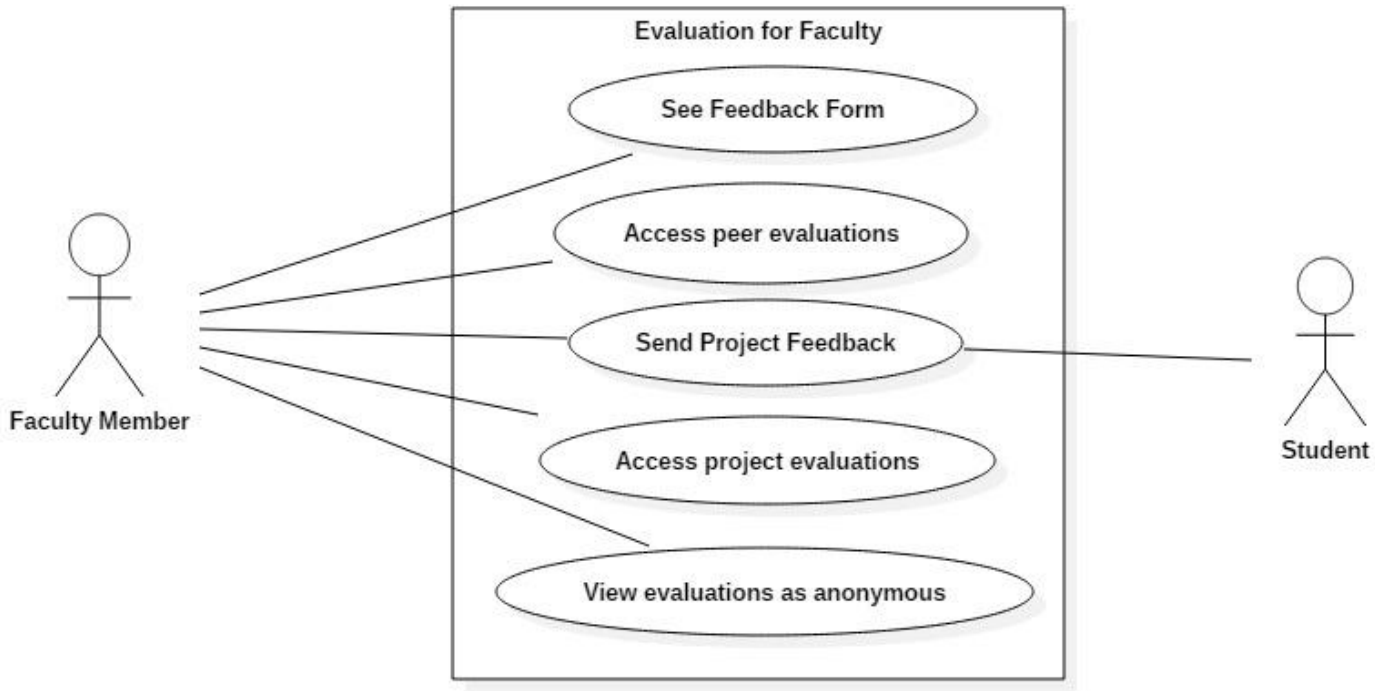
Sequence Diagram::Proposal Submissions::Proposal Submissions View



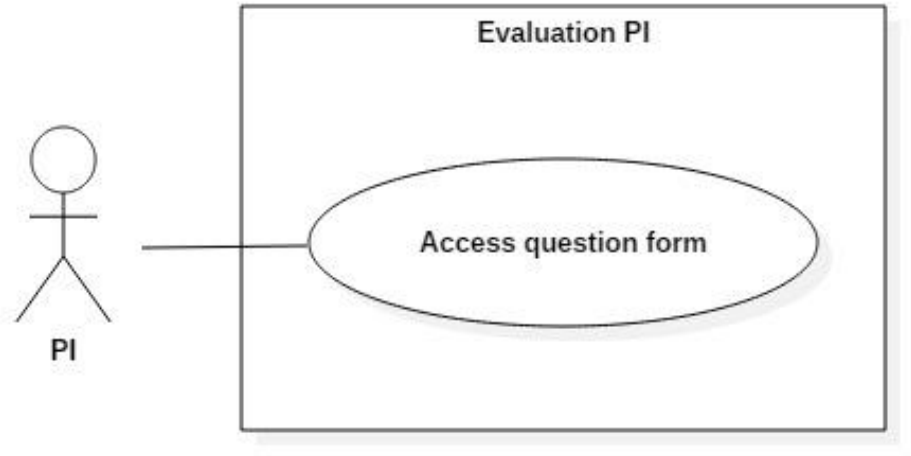
Evaluations

Use Case Diagrams

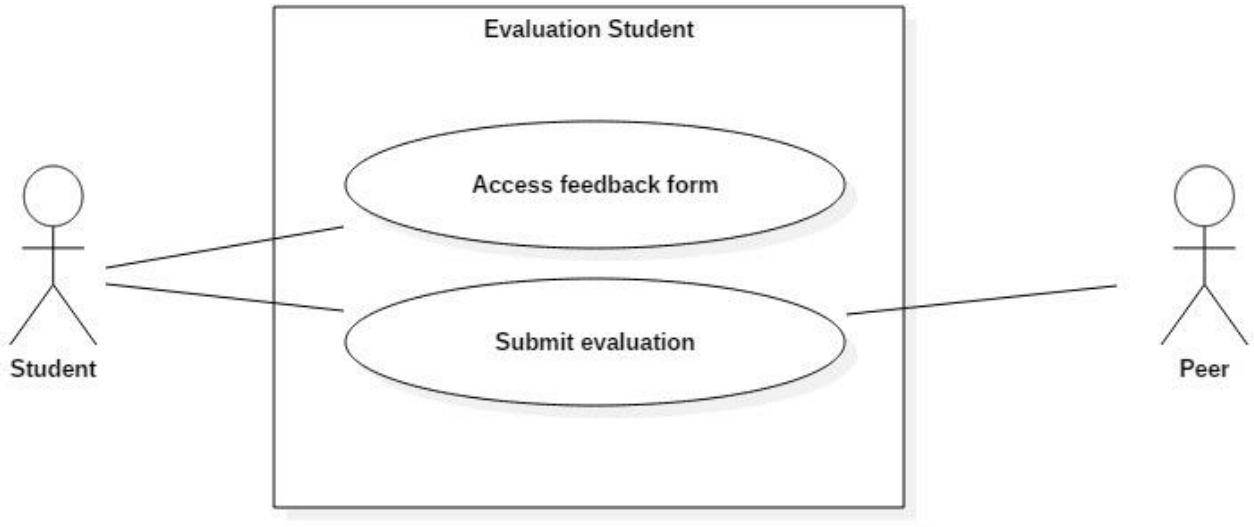
Faculty



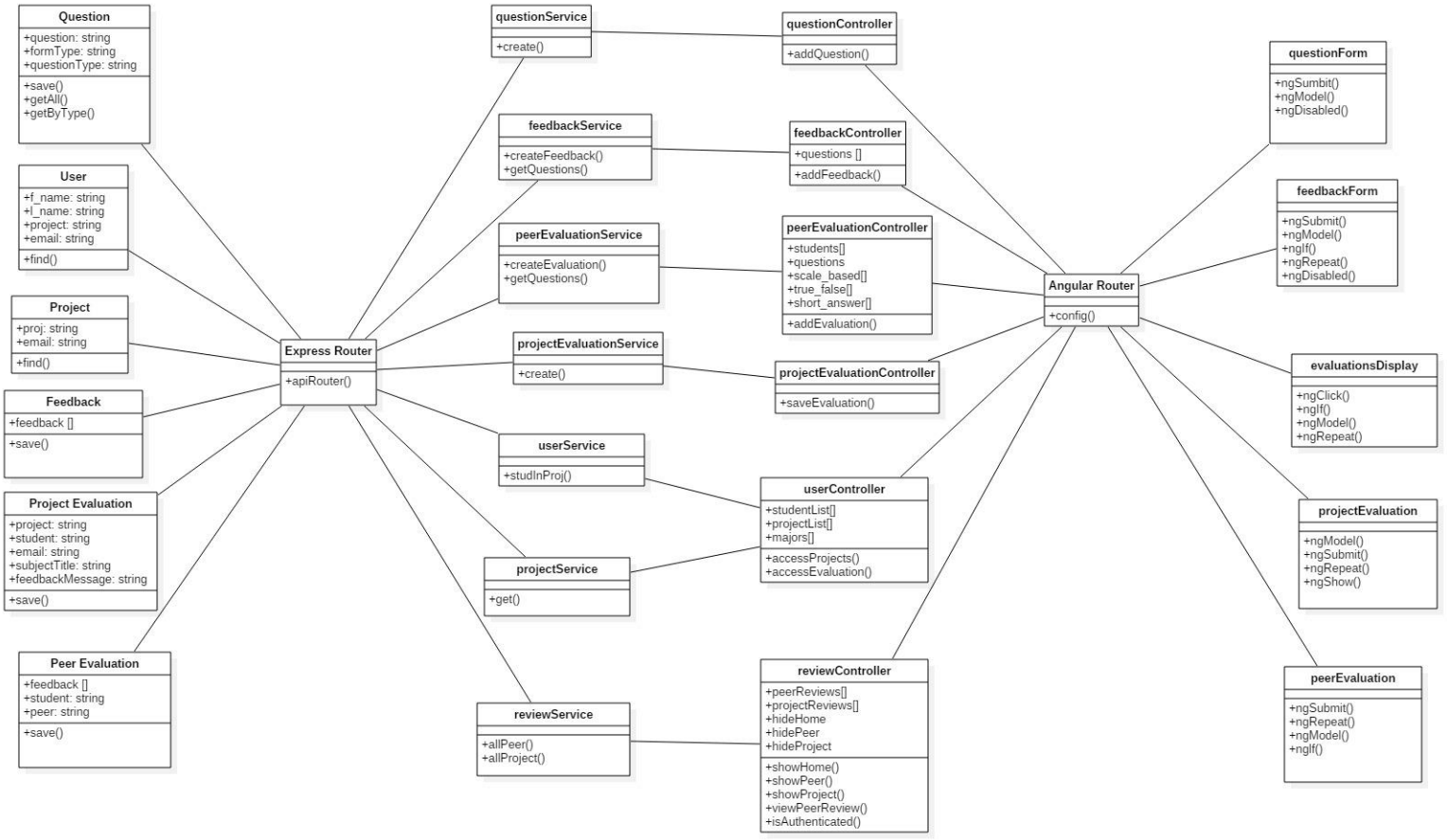
PI


Evaluation PI
Access question form
PI

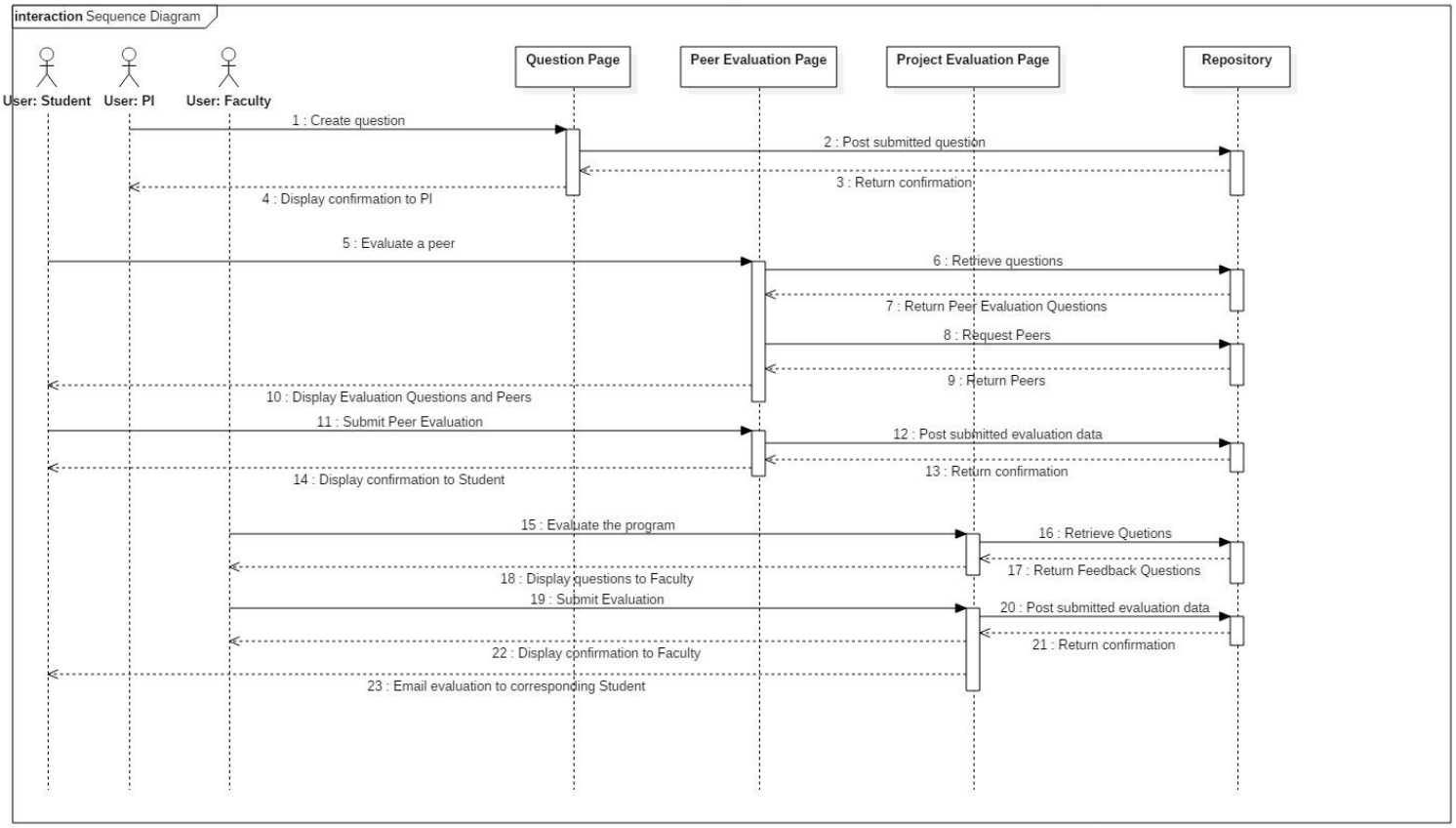
Student

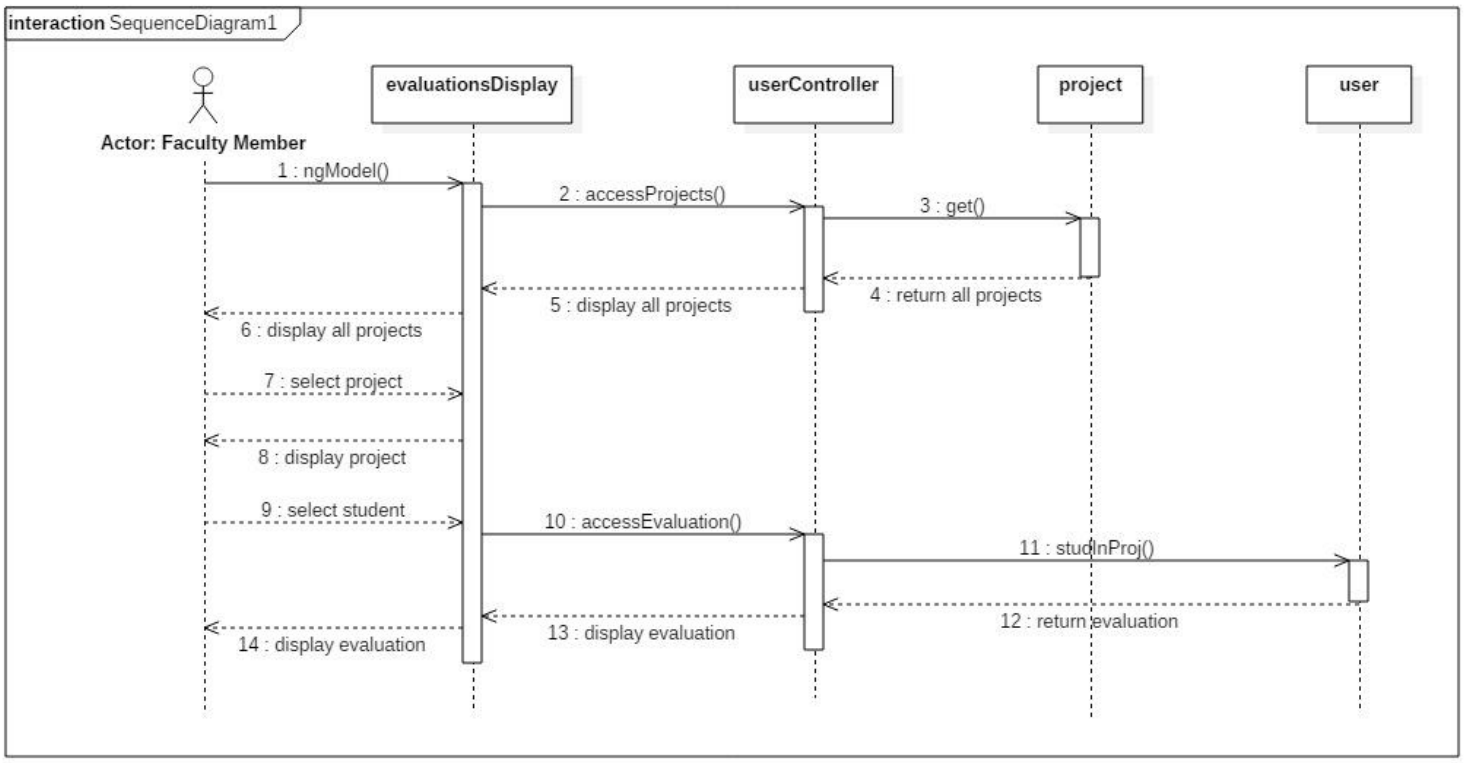

Evaluation Student
Access feedback form
Submit evaluation
Student
Peer

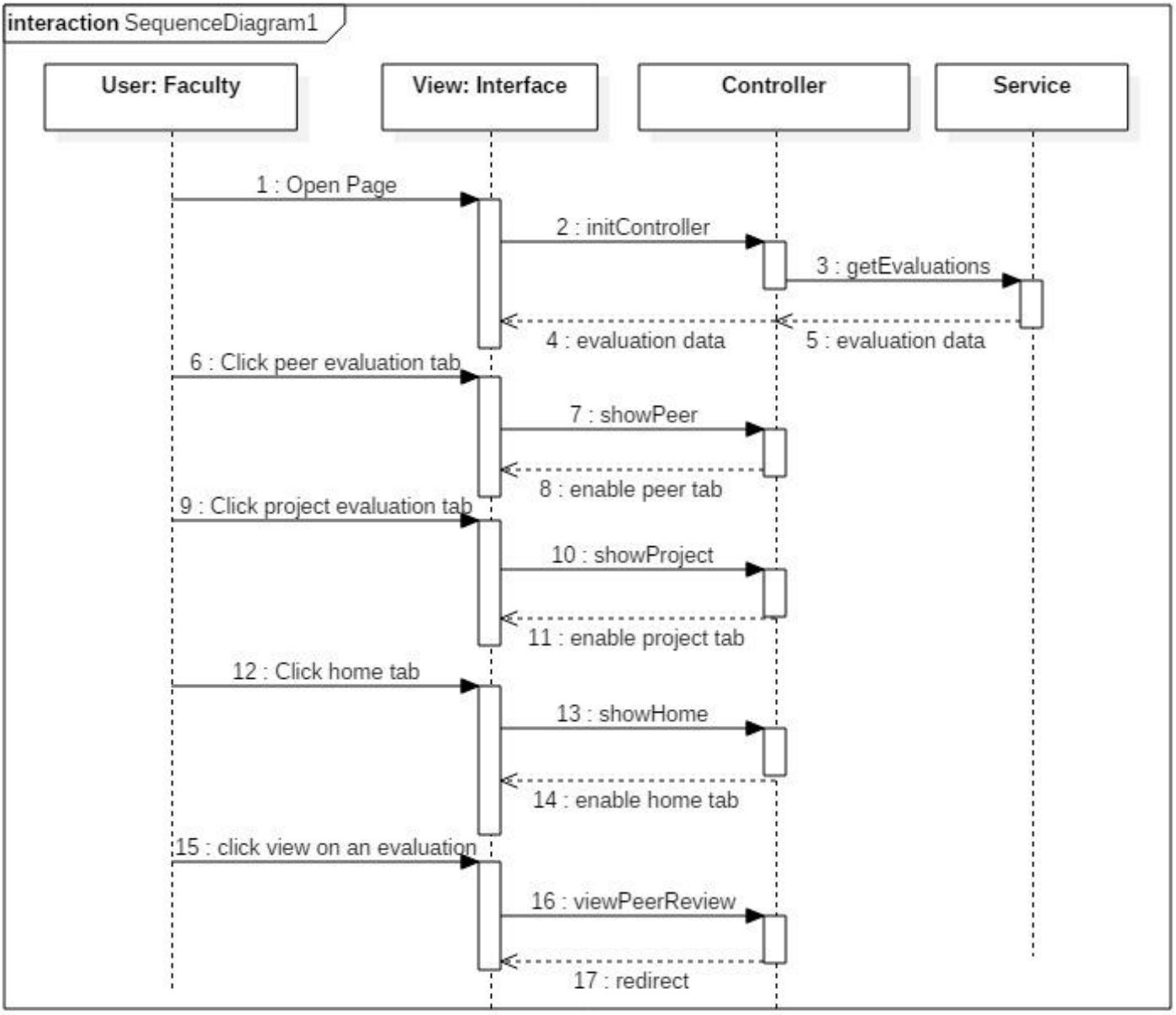
Class Diagram



Sequence Diagram

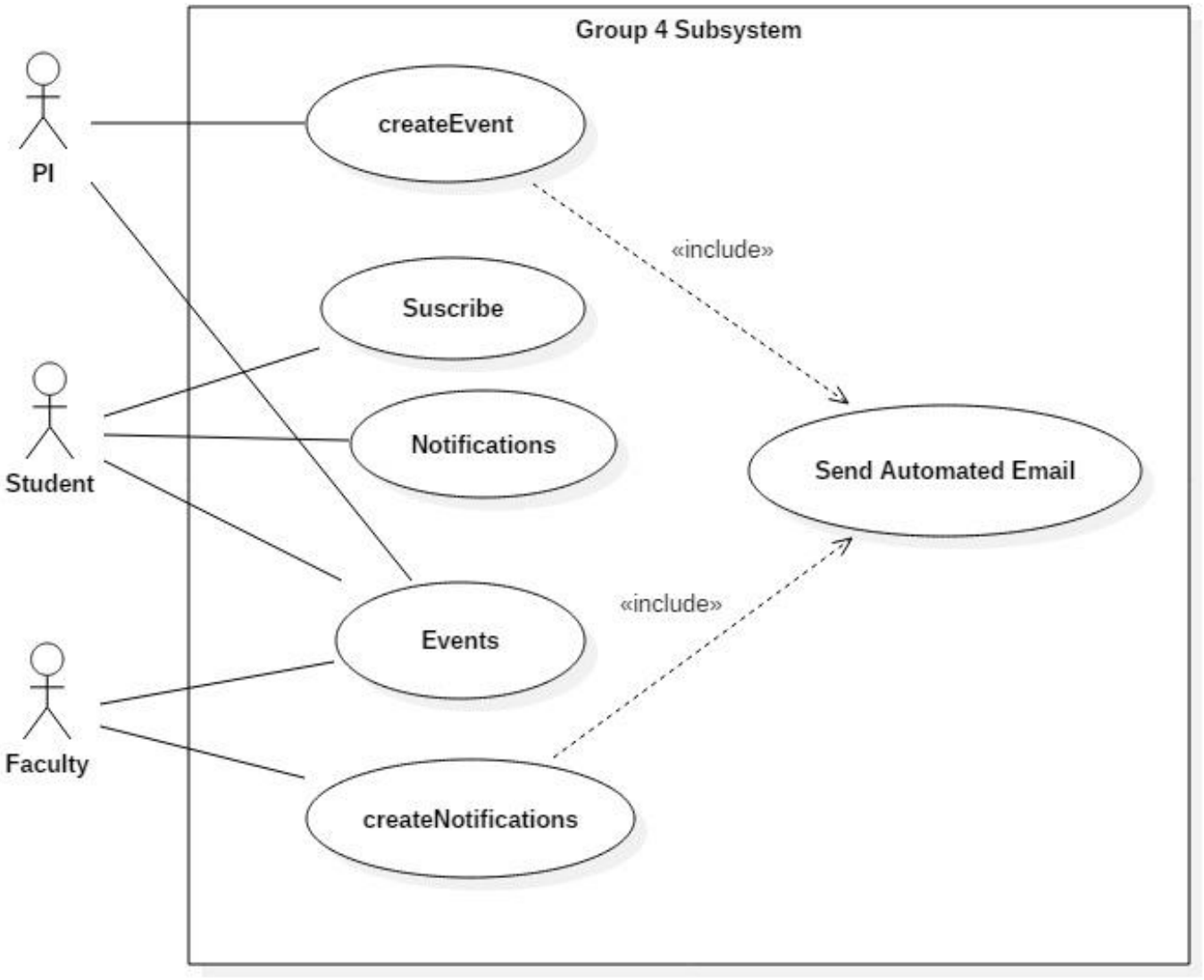



interaction SequenceDiagram1
Actor: Faculty Member
evaluationsDisplay
userController
project
user
1 : ngModel()
2 : accessProjects()
3 : get()
4 : return all projects
5 : display all projects
6 : display all projects
7 : select project
8 : display project
9 : select student
10 : accessEvaluation()
11 : studInProj()
12 : return evaluation
13 : display evaluation
14 : display evaluation

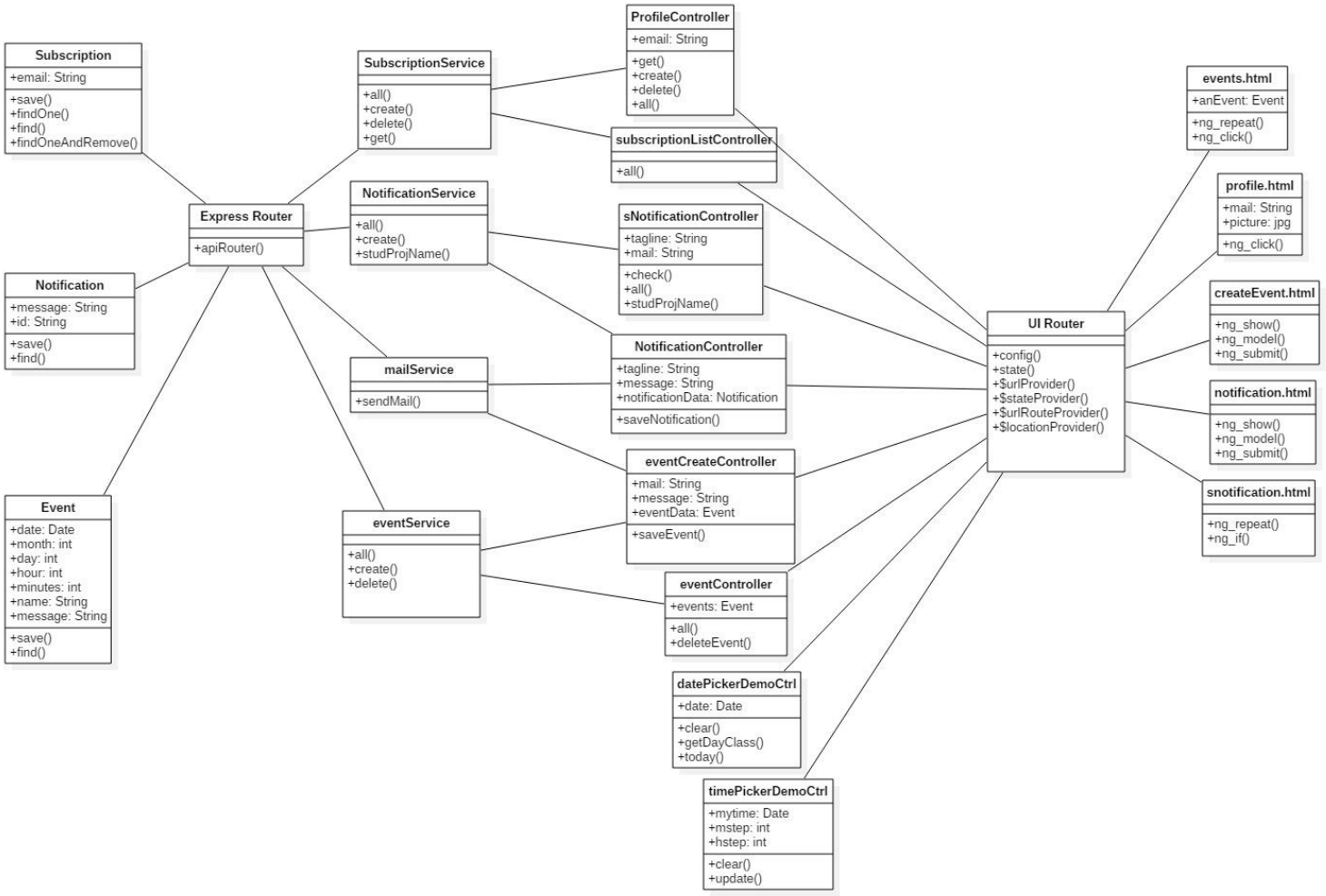

interaction SequenceDiagram1
User: Faculty
View: Interface
Controller
Service
1 : Open Page
2 : initController
3 : getEvaluations
4 : evaluation data
5 : evaluation data
6 : Click peer evaluation tab
7 : showPeer
8 : enable peer tab
9 : Click project evaluation tab
10 : showProject
11 : enable project tab
12 : Click home tab
13 : showHome
14 : enable home tab
15 : click view on an evaluation
16 : viewPeerReview
17 : redirect

Announcements

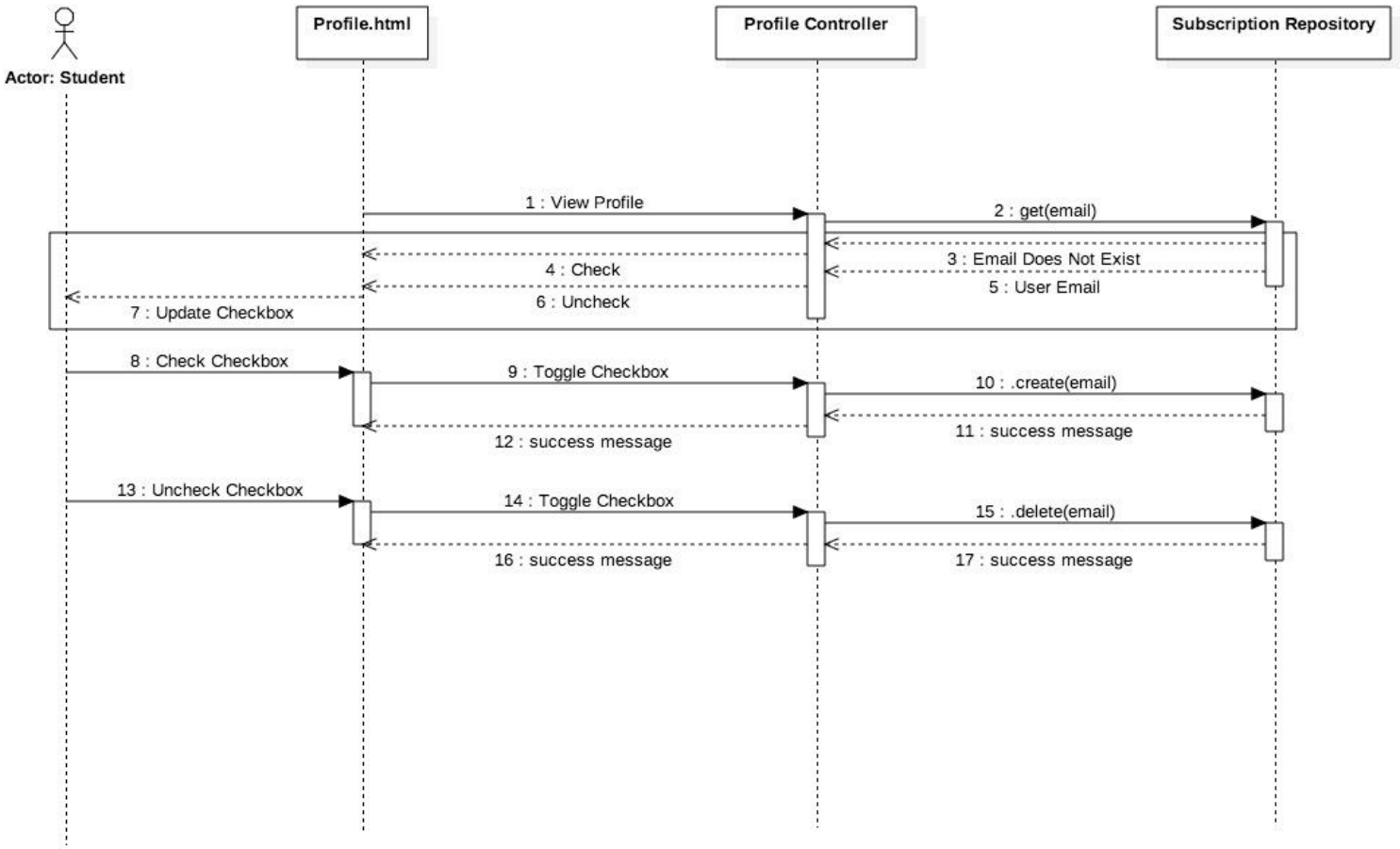
Use Case Diagram

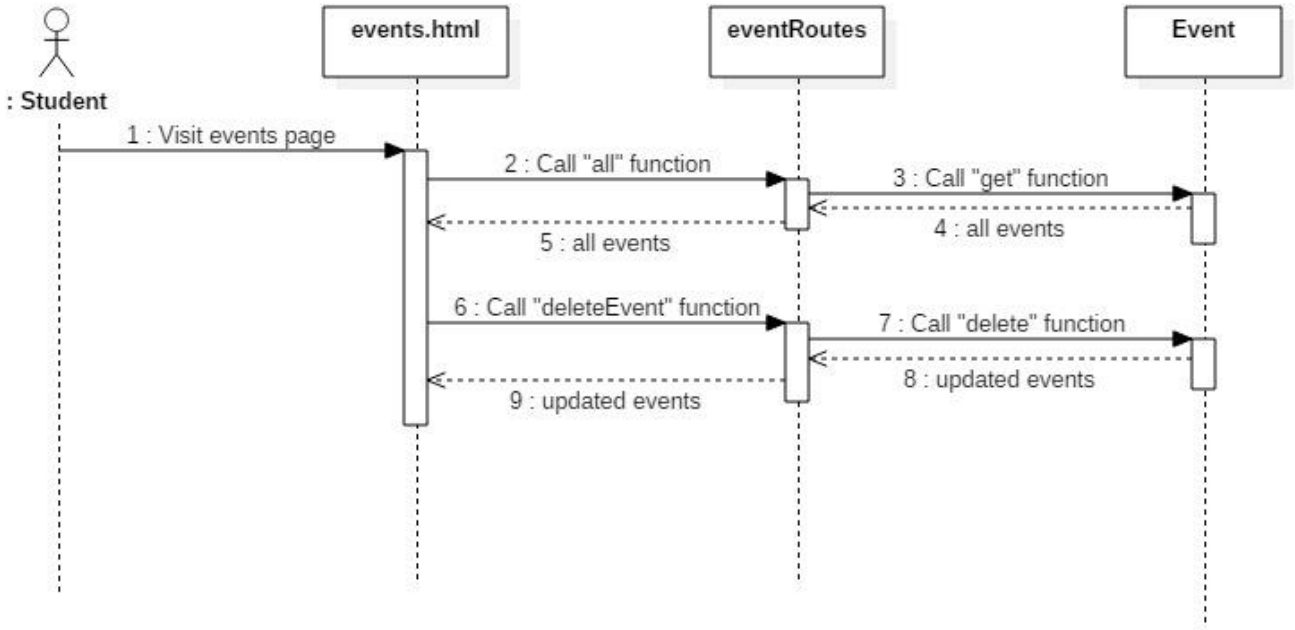

Group 4 Subsystem
PI
Student
Faculty
createEvent
Suscribe
Notifications
Events
createNotifications
«include»
«include»
Send Automated Email

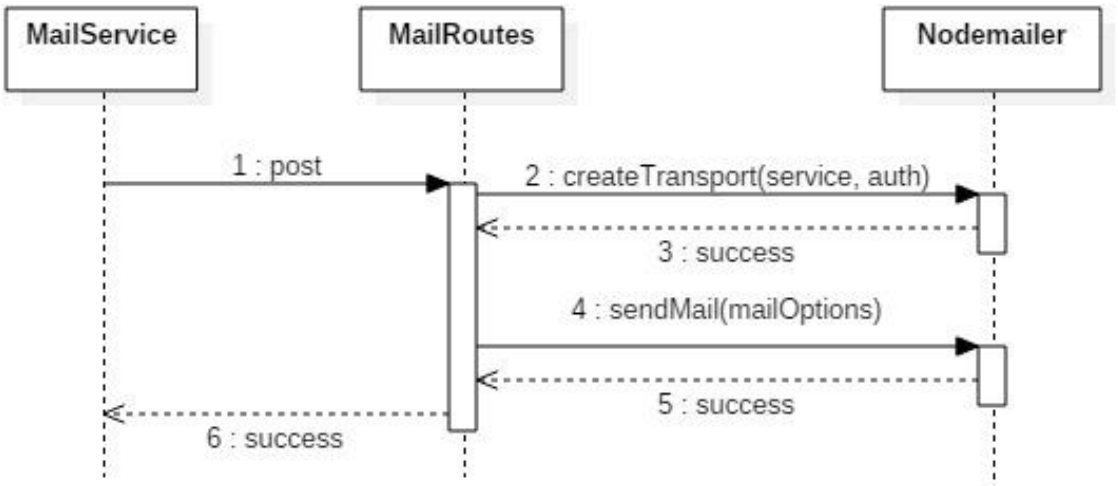
Class Diagram

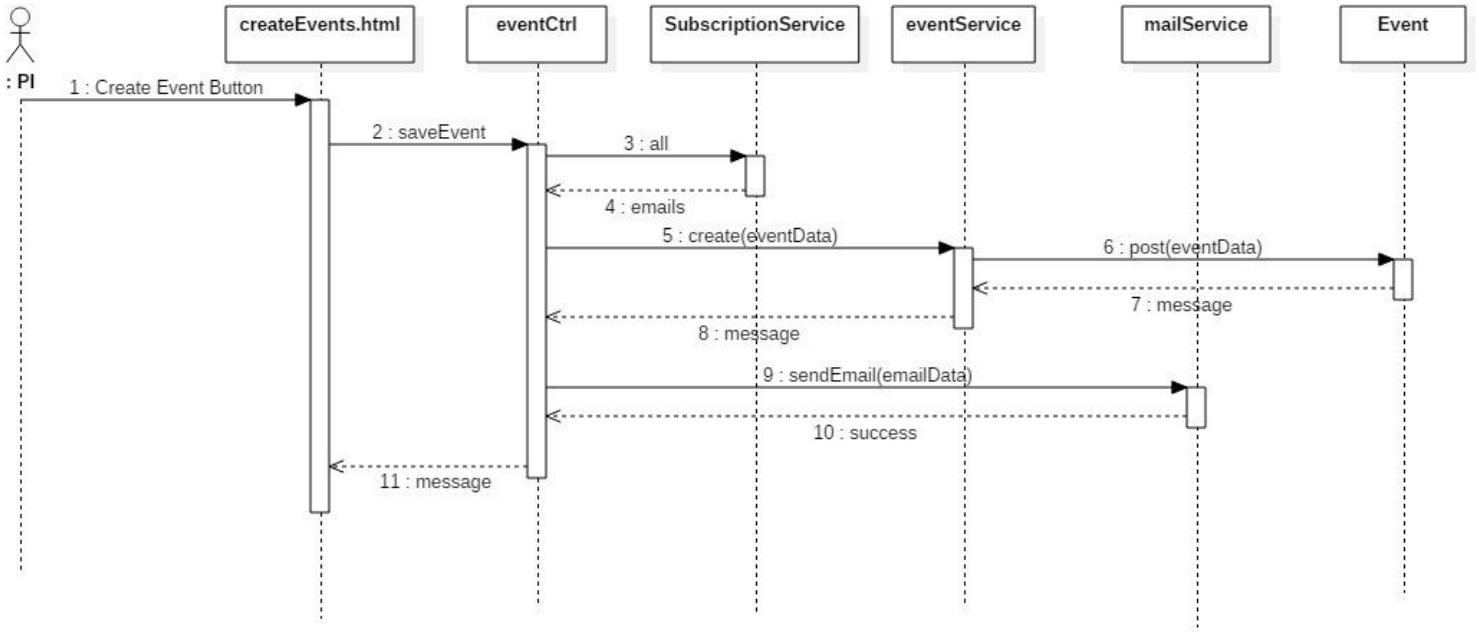


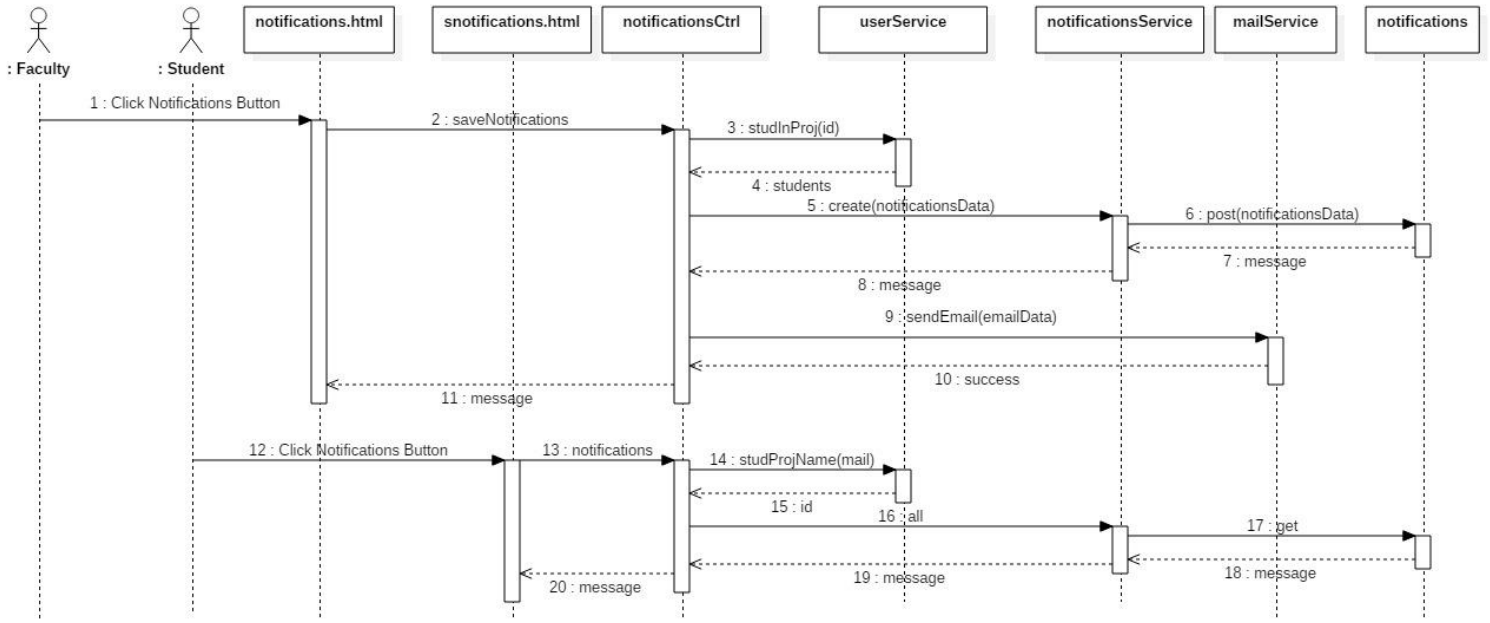
Sequence Diagram




: Student
events.html
eventRoutes
Event
1 : Visit events page
2 : Call "all" function
3 : Call "get" function
4 : all events
5 : all events
6 : Call "deleteEvent" function
7 : Call "delete" function
8 : updated events
9 : updated events


MailService
MailRoutes
Nodemailer
1 : post
2 : createTransport(service, auth)
3 : success
4 : sendMail(mailOptions)
5 : success
6 : success


: PI
createEvents.html
eventCtrl
SubscriptionService
eventService
mailService
Event
1 : Create Event Button
2 : saveEvent
3 : all
4 : emails
5 : create(eventData)
6 : post(eventData)
7 : message
8 : message
9 : sendEmail(emailData)
10 : success
11 : message


: Faculty
: Student
notifications.html
snotifications.html
notificationsCtrl
userService
notificationsService
mailService
notifications
1 : Click Notifications Button
2 : saveNotifications
3 : studInProj(id)
4 : students
5 : create(notificationsData)
6 : post(notificationsData)
7 : message
8 : message
9 : sendEmail(emailData)
10 : success
11 : message
12 : Click Notifications Button
13 : notifications
14 : studProjName(mail)
15 : id
16 : all
17 : get
18 : message
19 : message
20 : message

Appendix B - User Interface Design

The user will visit the FIU-VIP URL and from there will be open to many choices. Prospective students may apply to projects. Current students may apply for specific projects and participate in project events. Faculty and Staff will have several management tools to assign/remove students from projects. PI's and Co-PI's will have global control of many features.

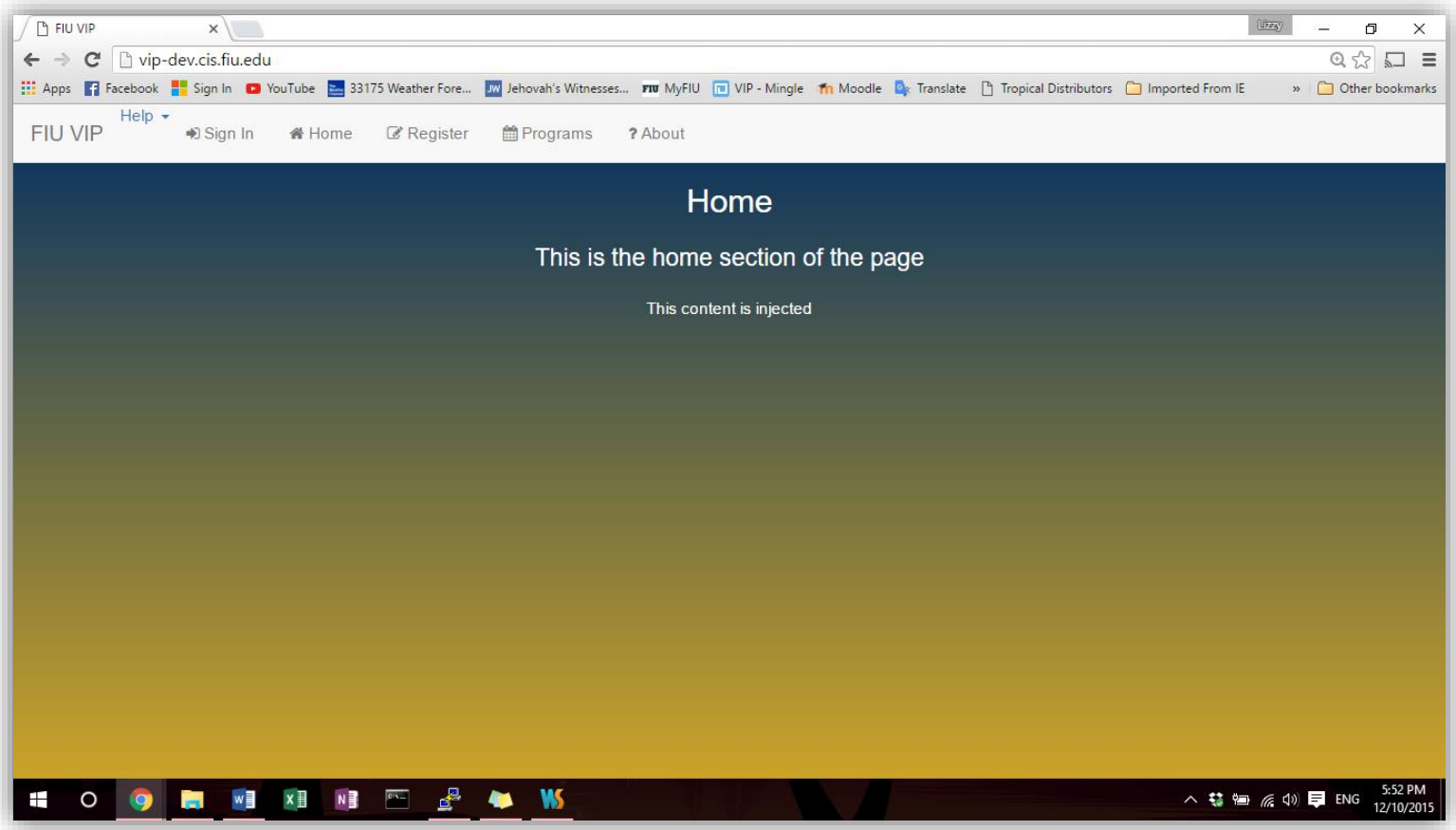


Figure 1: Guest (unregistered user) viewing the VIP home page.

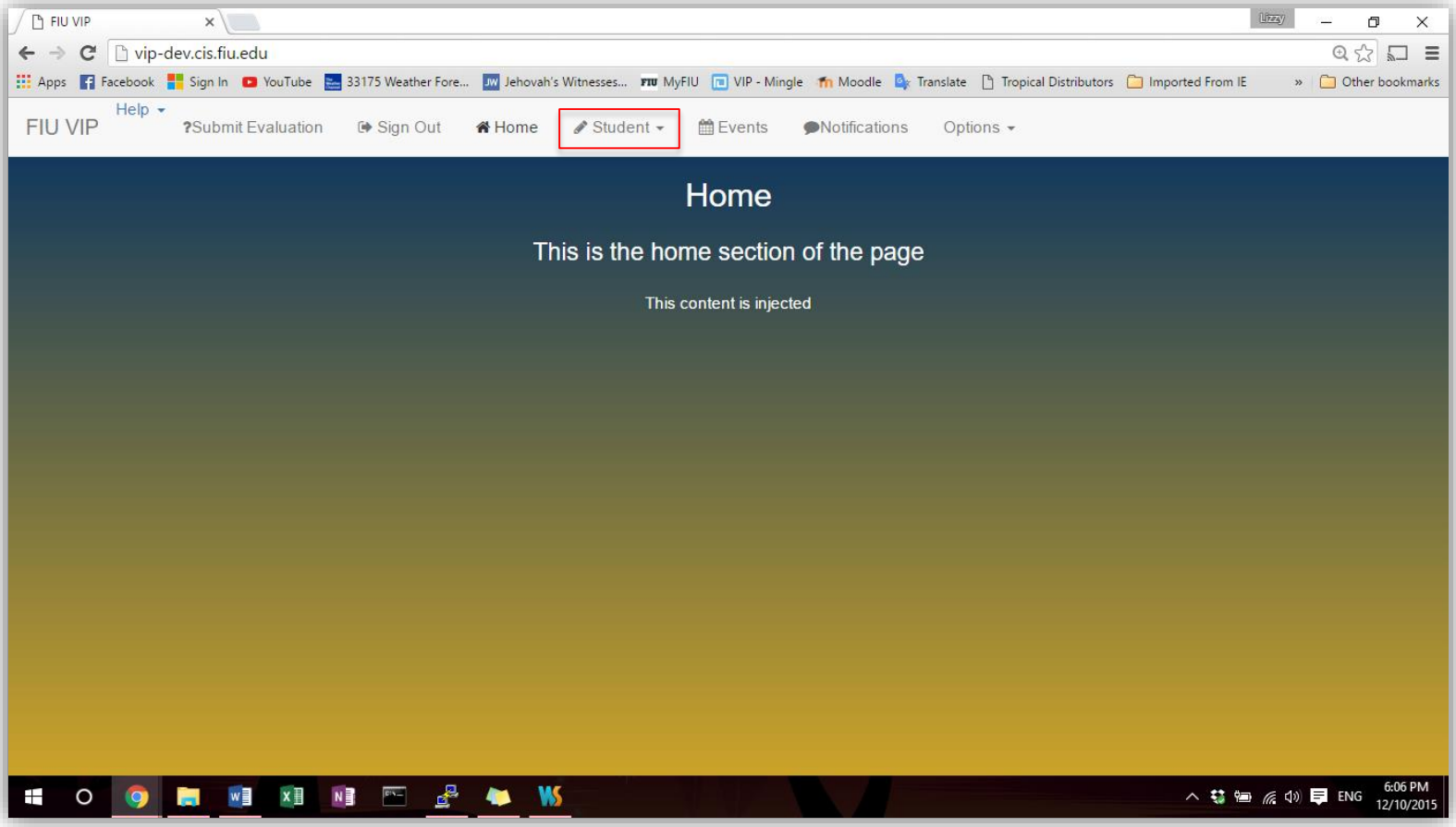


Figure 2: Figure 3: Viewing the page as a Student, which is indicated by the Student button and icon above. Students have access to all the tabs on the notification bar.

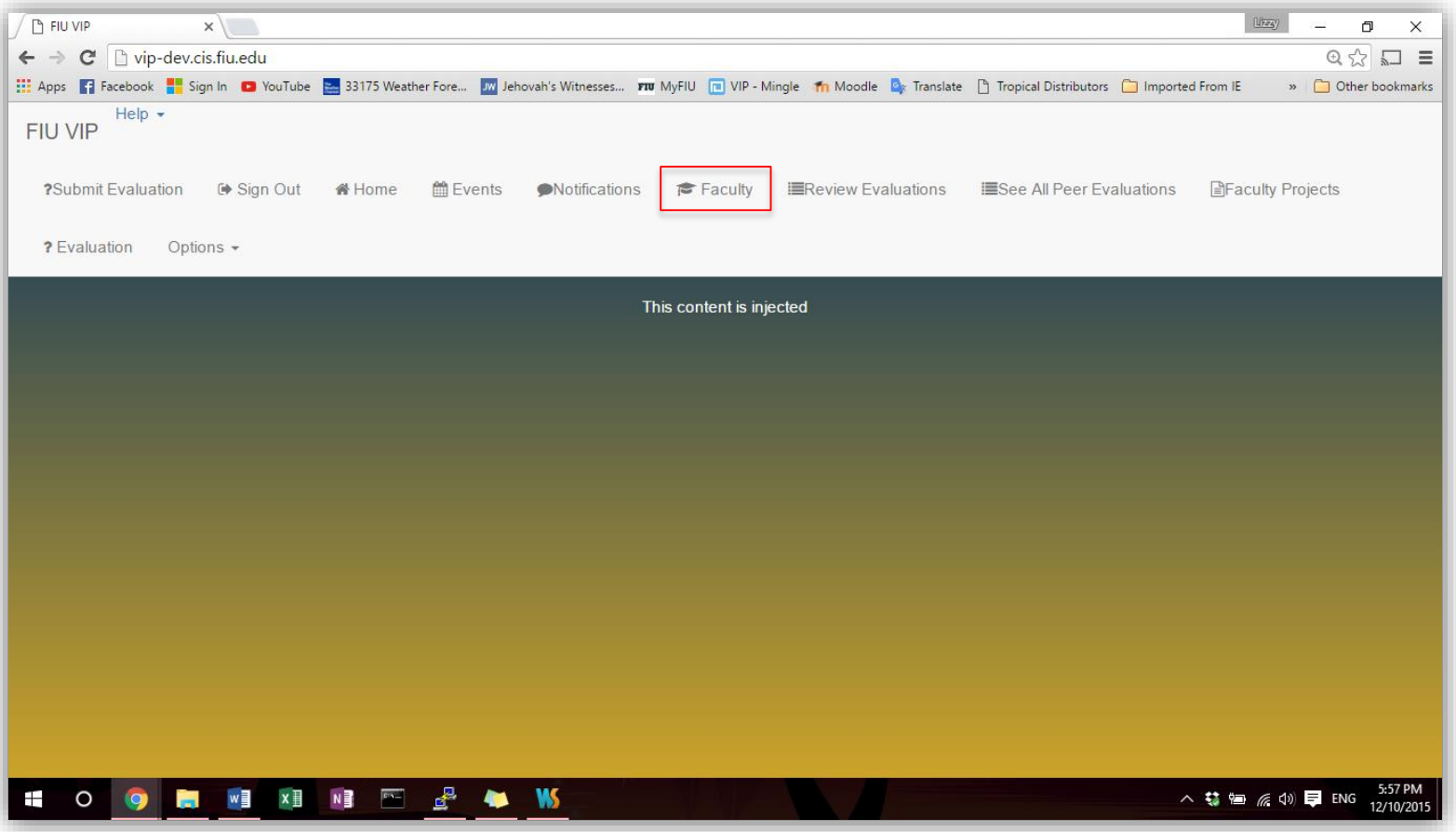


Figure 3: Viewing the page as a Faculty Member, which is indicated by the Faculty button and icon above. Faculty has access to all the tabs on the notification bar.

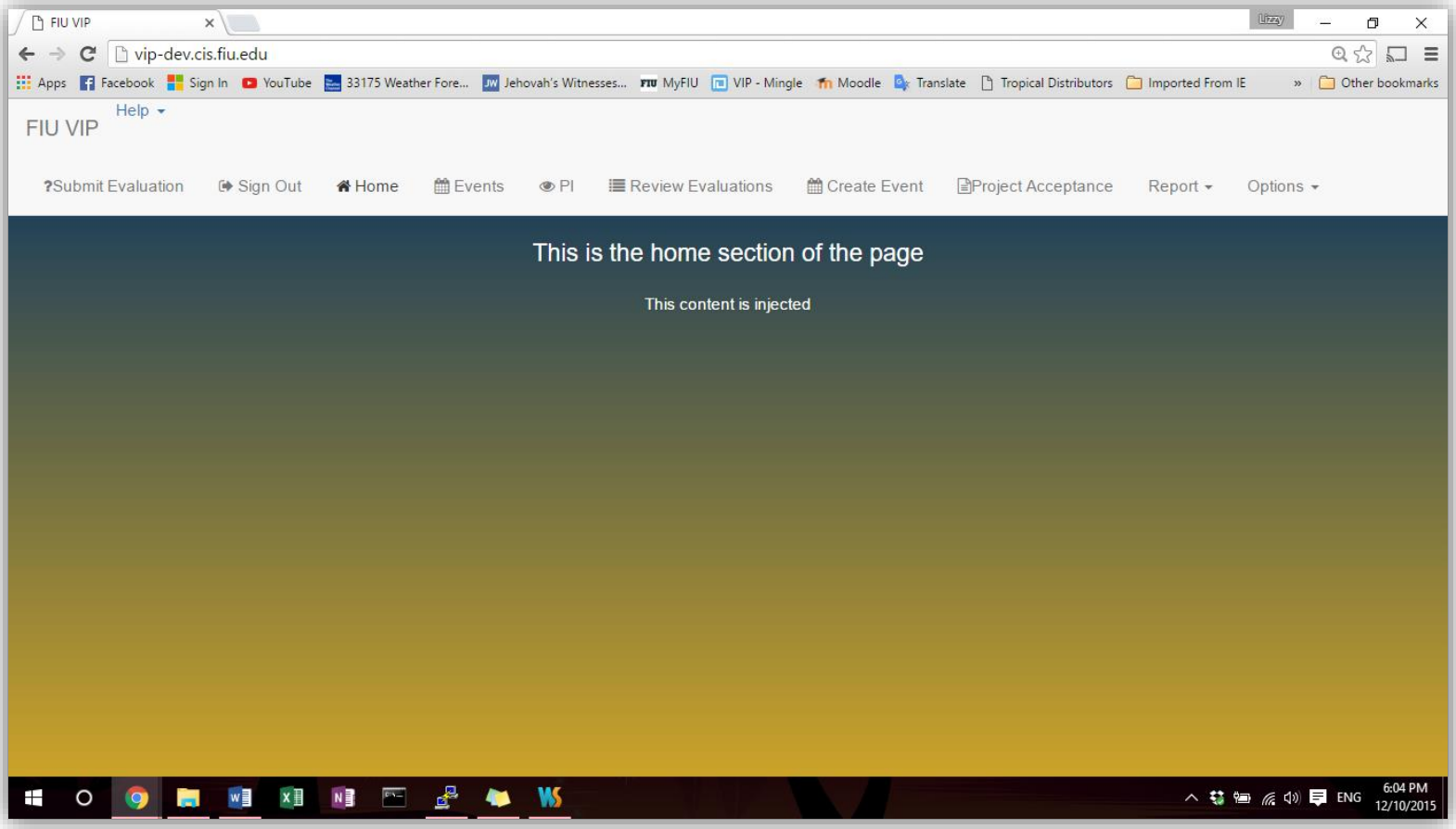


Figure 4: Viewing the page as a PI, which is indicated by the PI and icon above. PI has access to all the tabs on the notification bar.

Appendix C - Sprint Review Reports

Sprint 0 Report

Date: *09/09/2015*
Attendees: *First Name Last Name, First Name Last Name, …*
Discussed Topics:

- Agreed to work *10 hours per week,* outside of lecture time, towards this project
    - 2-5 “daily” scrum meetings per group
- System design: website
- Development software
    - StarUML
    - Webstorm
    - Mozilla Firefox
    - Putty
        - Only if running windows, so they can SSH into the vip-dev servers
    - GitHub is being used as our code repo

Sprint 1 Report

Date: *09/23/2015*
Attendees:
Discussed Topics:

- Split into teams of 4-6 developers.
    - 4-5 teams in total, each assigned which epic they will take charge
- Some stories approved by Sadjadi
    - Posted to Mingle
- Decided to split GitHub into multiple branches, one for each group, and the main master branch, which the website runs off of.
    - Only push to your group’s branch; only scrum masters will integrate the group branch into master

Sprint 2 Report

Date: *10/07/2015*
Attendees:
Discussed Topics:

- Majority of user stories were approved, some developers still without an approved story
- Decided to use MEAN (MongoDB, Express, AngularJS, NodeJS) Stack for our product

    - MEAN Machine textbook was recommended to help us quickly get on board with this stack
- Some groups had their use cases diagrams created
- Issue raised: Some people had problems making diagrams as they did not completely understand MVC (model-view-controller) architecture and the MEAN stack.

Sprint 3 Report

Date: *10/21/2015*
Attendees:
Discussed Topics:

- Development started - basic skeleton of website is up and running on vip-dev
    - Had custom CSS created for the website
- A few people had some basic working code to show on their local environments

Sprint 4 Report

Date: *11/04/2015*
Attendees:
Discussed Topics:

- Issue raised: our custom CSS is causing issues with some features that some people wanted to use (mainly with bootstrap)

Sprint 5 Report

Date: *11/18/2015*
Attendees:
Discussed Topics:

- Some user stories were not finished and needed to be finalized.
    - Most students had miscommunication issues with who had to supply an API for whom and what was needed for the API call
    - Some students had trouble connecting with the database
    - Product backlog was not changed because these were just technical issues.
- Most user stories were finished and Group 2 completely integrated within themselves.

Sprint 6 Report

Date: *11/30/2015*
Attendees:
Discussed Topics:

- Final show and tell with fully integrated website, running off vip-dev.

- Had a mix up when integrating the final project.
    - Group 1, 2, 3 were integrated
    - Group 1, 2, 4 were integrated
    - Did not have a complete class integration.

Sprint 7 Report

Date: *12/11/2015*
Attendees:
Discussed Topics:

- There is no formal sprint 7 review, thanks for the hard work everyone!

REFERENCES

- Sevilleja, Chris, and Holly Lloyd. MEAN MACHINE - The Beginner's Guide to the JavaScript Stack. N.p.: Lean, 2015. Print.
- "Vertically Integrated Projects Program | VIP." Vertically Integrated Projects Program | VIP. N.p., n.d. Web. 10 Dec. 2015.