

# Software Requirements Specification Document

## **Rate-My-Elective**

**1.0**

**December 2th, 2021**

Spencer Kinney  
Jakob Lybarger  
Shawn O'Brien  
Jacquelyn Simanis  
Gabriel St. Angel

Submitted in partial fulfillment of the requirements of  
IT 326: Principles of Software Engineering

<b>TABLE OF CONTENTS</b>	<b>2</b>
<b>1. INTRODUCTION</b>	<b>4</b>
1.1 PURPOSE	4
1.2 SCOPE	4
1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS	4
1.4 REFERENCES	4
1.5 OVERVIEW	5
<b>2. GENERAL DESCRIPTION</b>	<b>5</b>
2.1 PRODUCT PERSPECTIVE	5
2.2 USER CHARACTERISTICS	5
2.3 SYSTEM ENVIRONMENT	5
2.4 ASSUMPTIONS AND DEPENDENCIES	6
<b>3. SPECIFIC REQUIREMENTS</b>	<b>7</b>
3.1 FUNCTIONAL REQUIREMENTS	7
3.1.1 <i>Remove Class</i>	7
3.1.2 <i>Remove Others' Review</i>	8
3.1.3 <i>Add Department</i>	8
3.1.4 <i>Remove Department</i>	8
3.1.5 <i>View Flagged Reviews</i>	8
3.1.6 <i>Remove Other User's Account</i>	9
3.1.7 <i>View Flagged Classes</i>	9
3.1.8 <i>Write Review</i>	9
3.1.9 <i>Add Class</i>	9
3.1.10 <i>Flag a Review</i>	10
3.1.11 <i>Flag a Class</i>	10
3.1.12 <i>Rate Review</i>	10
3.1.13 <i>Search for a Class</i>	10
3.1.14 <i>Save Data</i>	11
3.1.15 <i>Toggle Dark Mode</i>	11
3.1.16 <i>Contact Admin</i>	11
3.1.17 <i>Remove Own Account</i>	11
3.1.18 <i>Edit Profile</i>	12
3.1.19 <i>Delete Own Review</i>	12
3.1.20 <i>Edit Own Review</i>	12
3.1.21 <i>Log Out of Account</i>	13
3.1.22 <i>Change Password</i>	13
3.1.23 <i>Register for Account</i>	13
3.1.24 <i>Complete Captcha</i>	13
3.1.25 <i>Login to Account</i>	14
3.2 NON-FUNCTIONAL REQUIREMENTS	14
3.2.1 <i>Performance</i>	14
3.2.2 <i>Reliability</i>	14
3.2.3 <i>Availability</i>	14
3.2.4 <i>Security</i>	14
3.2.5 <i>Maintainability</i>	14

<i>3.2.6 Portability</i>	14
<b>4. DESIGN &amp; DEVELOPMENT</b>	14
4.1 SOFTWARE PROCESS MODEL	14
4.2 DELIVERABLE 1	15
4.2.1 Description	15
4.2.2 Design	15
4.3 Deliverable 2	39
4.3.1 Description	39
4.3.2 Design	39
4.3.3 Development	41
4.4 Deliverable 3	43
4.4.1 Development	43
4.4.2 Design	45
4.5 Deliverable 4	52
4.5.1 Description	52

## 1. Introduction

The following subsections of the Software Requirements Specifications (SRS) document will provide an overview of the *Rate My Elective* software project and help the reader understand the project from both a high-level and technical standpoint. The document will begin with some context about the problem that is trying to be solved and what the proposed solution is. It will then go through all of the technical requirements and some analysis on what needs to be completed. Finally, it will go through some data & architectural models to help the reader understand the proposed project.

### 1.1 Purpose

The purpose of this document is to build an online system that allows students easy access to read others' opinions and experiences with certain courses at Illinois State University, as well as share theirs to make the process of choosing electives easier and stress free. This document is useful for gaining a well rounded understanding of what exactly Rate My Elective will allow users to do and how it will benefit the decision making process regarding electives.

### 1.2 Scope

The purpose of *Rate My Elective* is to ease the decision making process that students face when it comes to choosing electives based on the opinions and experiences of peers. The system will allow users to provide and see others opinions on certain electives. Users will also be able to add electives that are not already in the database as well as add reviews on any class and interact with the reviews of others. This system will not allow users to rate professors or perform any of the listed functionality on or about professors.

### 1.3 Definitions, Acronyms, and Abbreviations

At various points throughout this document, we may refer to the user developing a secure connection to the website with HTTPS (Secure Hypertext Transfer Protocol). This will be needed for the project because it will increase security measures for new users connecting to the website and it will create a sign of trust between the website and user. In addition, we will often use 'ISU' throughout the document which you can assume to be Illinois State University. As for technologies, we will also frequently refer to Django and PostgreSQL. Django is a web-based python framework that is used to create dynamically generated web pages. It closely follows the Model-View-Controller (MVC) architecture. PostgreSQL is a relational database management system that utilizes the standard database query language (SQL).

### 1.4 References

The document will frequently reference Illinois State University (ISU) and content that is present on their website. Data including current elective and department offerings will be used. Their website is *illinoisstate.edu*

### 1.5 Overview

Throughout this document, we will present the problem that we are trying to solve as well as what type of user would benefit from using Rate-My-Elective. We will then provide technical

detail and architectural layouts of each of our software requirements and how they will be designed and implemented.

## 2. General Description

Rate-My-Elective was designed to include core functionality to maximize utility and intuitive user experience. The features of the application were created from the perspective of the user group that would be using this application. Rate-My-Elective is intended to provide Illinois State University students quality feedback and insight from peers on elective choices. In this section, we will discuss some of the product features, the profile of the target users, and some assumptions and dependencies for the application.

### 2.1 Product Perspective

This product is similar to *Rate My Professor* which is used for rating professors at various universities and *Rate My Dorm* for rating dorms at various universities. Moreso to *Rate My Professor*. Though similar to *Rate My Professor*, *Rate My Elective* varies due to the fact that there will be no ratings based on professors. The ratings on *Rate My Elective* will solely be based on the information regarding the contents and overall expectations within the class. Ratings will be based on information varying from the amount of time required, difficulty of the overall subject, and more. Overall, it will help students make the most informed decisions possible on what electives to take next semester.

### 2.2 User Characteristics

The intended end user of the Rate My Elective application is generally college students who are currently attending Illinois State University. It is possible that there will be other types of users such as prospective ISU students or students attending a different school, however our target group is currently enrolled ISU students. The application will also be designed to be utilized by someone with limited technical experience which means that we will aim for the most intuitive user experience.

### 2.3 System Environment

The Rate My Elective application is intended to be compatible with any internet-capable device that has a web browser. The window will be able to scale and fit any size of screen and performance will not be an issue on less-powerful devices because of the lightweight design of the application. From any mobile device, the user will be able to execute the same functionalities as they could in a desktop web browser.

The Rate My Elective application will be developed with the Django web framework library. For the persistent storage solution, we will use the PostgreSQL relational database system. The back-end infrastructure will be supported by the Python programming language, and the frontend web pages will be made up of HTML for markup and CSS for styling. The server will run on the LinuxOS operating system.

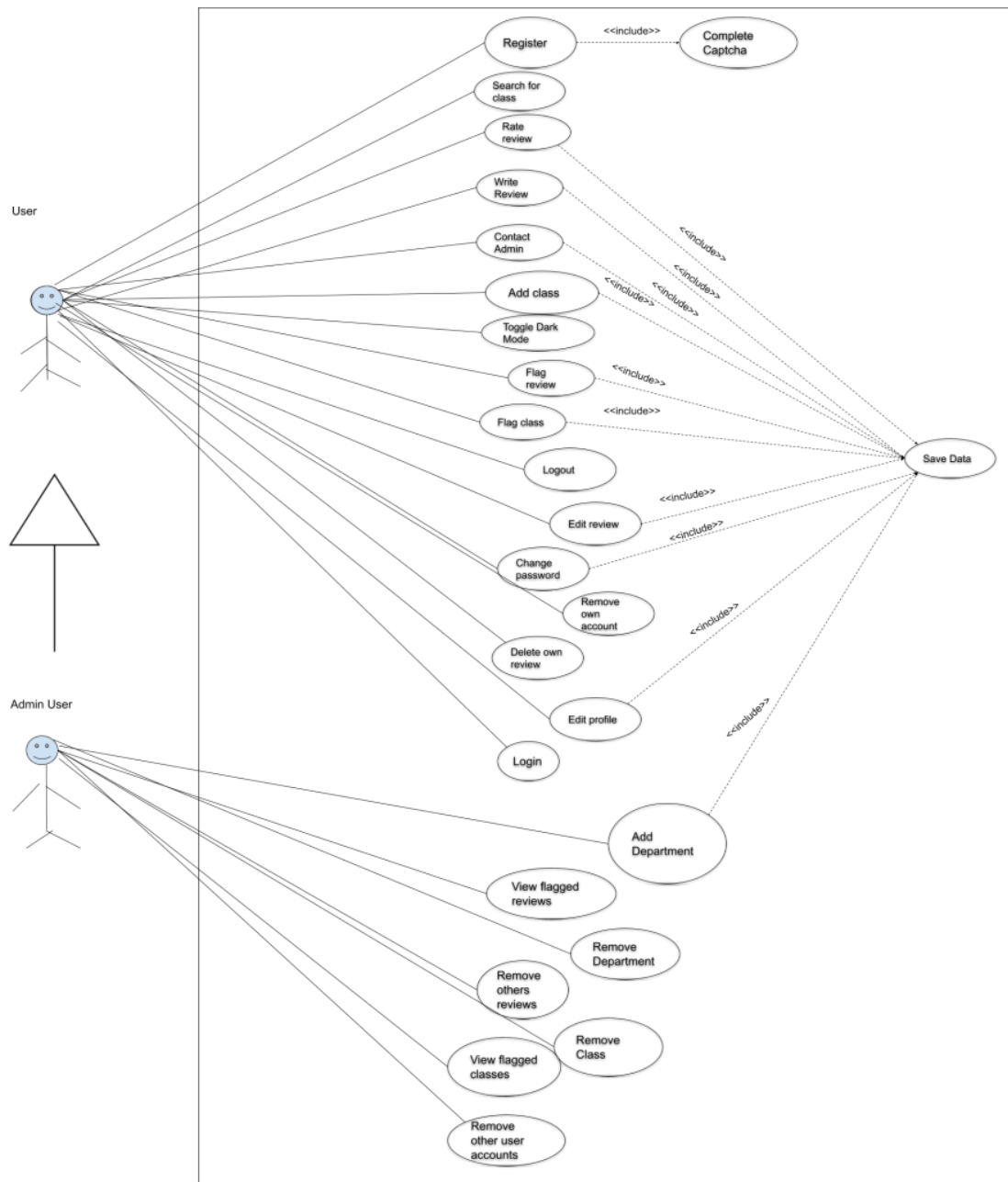
### 2.4 Assumptions and Dependencies

We will use Django for our backend solution along with PostgreSQL for our database. Django will be the back-end application which will serve the dynamic HTML pages as well the API data

from PostgreSQL. Python is the object-oriented language within Django that will create the respective database models. PostgreSQL will be a dependency because the HTML page can not load without the table data. In addition, the Illinois State University website will be parsed to download data about the classes and departments offered, so consistent data will be required for project maintainability. With that said, an assumption can be made that the Illinois State University will remain as an available resource and will not shut down anytime soon.

### 3. Specific Requirements

#### 3.1 Functional Requirements



### **3.1.1 Remove Class**

*a.Description – Allows the Administrators to remove a class*

*b.Actor(s) – Admin*

*c.Trigger – A button press will trigger the removal of the class in the database*

*d.Conditions*

*i. Pre - The class must exist*

*ii.Post – The class must be removed from the database and the UI*

*e.Exceptions:*

*i. Connection to the database was lost.*

### **3.1.2 Remove Others Reviews**

*a.Description – Allows the Administrators to remove a review of a class.*

*b.Actor(s) – Admin*

*c.Trigger – A button press will trigger the removal of the review from the database*

*d.Conditions*

*i.Pre - The review must exist*

*ii.Post – The review must be removed from the database and the UI*

*e. Exceptions*

*i. User is not an admin*

*ii. Connection to database lost*

### **3.1.3 Add Department**

*a.Description - Allows the Administrators to add a school department*

*b.Actor(s) - Admin*

*c.Trigger - A form will allow the entry of a new school department*

*d.Conditions*

*i.Pre- The department must not already exist*

*ii.Post - The department will be added to the database*

*e.Exceptions: None*

### **3.1.4 Remove Department**

*a.Description – Allow the Administrators to remove a department*

*b.Actor(s) – Admin*

*c.Trigger – A button press will trigger the removal of the department from the database*

*d.Conditions*

*i.Pre - The department must exist*

*ii.Post – The department must be removed from the database and the UI*

*e.Exceptions: None*

### **3.1.5 View Flagged Reviews**

*a.Description – Allow the administrators to identify flagged reviews*

*b.Actor(s) – Admin*

*c.Trigger – A button press will open a designated form with all the flagged reviews*

*d.Conditions*

*i.Pre - Reviews must be able to be flagged*

ii. *Post – The flagged reviews will be put in one place for review*

*e. Exceptions:*

i. *Failed to retrieve data from database*

ii. *Unrecognized value used to search for flagged reviews*

### **3.1.6 Remove Other User's Account**

a. *Description – Allow the Administrators to remove a user's account*

b. *Actor(s) - Admin*

c. *Trigger – A button press will trigger the removal of the account from the database*

d. *Conditions*

i. *Pre - The account must exist*

ii. *Post – The account must be removed from the database*

*e. Exceptions:*

i. *Account to be removed belongs to an admin*

ii. *Failed to remove data from database*

### **3.1.7 View Flagged Classes**

a. *Description – Allow the admins to view flagged classes seamlessly*

b. *Actor(s) – Admin*

c. *Trigger – A page designated for the flagged classes will exist*

d. *Conditions*

i. *Pre - The function to flag a class must exist*

ii. *Post – The flagged classes will be collected and displayed for review*

*e. Exceptions:*

i. *Failed to retrieve data from database*

ii. *Unrecognized value used to search for flagged classes*

### **3.1.8 Write Review**

a. *Description – Allow users of the site to write and save a review*

b. *Actor(s) – User*

c. *Trigger – A form will be generated that allows the user to submit and save a review*

d. *Conditions*

i. *Pre - The user must not have written a review for this class before*

*User must be logged in to their account.*

ii. *Post – The review will be posted to the database*

*e. Exceptions:*

i. *Review form not properly filled out*

ii. *Lost connection either to database or internet*

### **3.1.9 Add Class**

a. *Description – Allow users to add a class to the database*

b. *Actor(s) – Login user, Admin*

c. *Trigger – A form will be generated that allows the user to add a class*

d. *Conditions*

i. *Pre - The class must not exist in the database*

*User must be logged in to their account.*

ii. *Post – The class will be posted to the database*



*e. Exceptions:*

*i. Connection to the database was lost.*

### **3.1.10 Flag a Review**

*a. Description – Allow users of the site to flag a review*

*b. Actor(s) – Login user, Admin*

*c. Trigger – A button will exist by a review to trigger a flag*

*d. Conditions*

*i. Pre -*

*1. The user must be logged into their account*

*2. The post must not already be flagged*

*ii. Post – The review will be flagged in the database*

*e. Exceptions:*

*i. Failed to retrieve data from database*

*ii. Unrecognized value used to search for flagged reviews*

### **3.1.11 Flag a Class**

*a. Description – Allow users of the site to flag a class*

*b. Actor(s) – Login user, Admin*

*c. Trigger – A button will exist by a review to trigger a flag*

*d. Conditions*

*i. Pre -*

*1. The user must be logged into their account*

*2. The class must not already be flagged*

*ii. Post – The class will be flagged in the database*

*e. Exceptions:*

*i. Connection to database was lost*

### **3.1.12 Rate Review**

*a. Description – Allow users of the site to rate a review*

*b. Actor(s) – Login user, Admin*

*c. Trigger – Two buttons will exist by the review to trigger a rating*

*d. Conditions*

*i. Pre -*

*1. The user must be logged into their account*

*2. The post must not already have been rated by the user*

*ii. Post – The rating will be added to the review in the database*

*e. Exceptions:*

*i. Connection to database lost*

### **3.1.13 Search for a Class**

*a. Description – Allow users of the site to search for a class by selecting from a drop down menu, and inputting the course number.*

*b. Actor(s) – User*

*c. Trigger – A button click.*

*d. Conditions*

*i.Pre - None*

*ii.Post – The user will be redirected to a page that displays information and reviews for the class selected*

*e.Exceptions:*

*i. Connection to the database lost*

*ii. Incorrect input data*

#### **3.1.14 Save Data**

*a.Description – Allow users to save data after editing or creating a profile information, a new account, a review, class, or department.*

*b.Actor(s) – None*

*c.Trigger – A button press will initiate saving the data being edited or created to the database.*

*d.Conditions*

*i.Pre - The data must be in the “editor” page for the data they wish to save. If the data already exists, the user must be logged into an Admin account or the account that created the data, and a change must have been made in the “editor” page. If the data is being created, all required fields must be filled with valid data.*

*ii.Post – Existing data will be updated in the database. New data will be stored in the database in the appropriate location.*

*e.Exceptions:*

*i.Incorrect form data for data model to be saved*

*ii.Connection to database lost*

#### **3.1.15 Toggle Dark Mode**

*a.Description – Allow users to toggle between Dark Mode and Light mode.*

*b.Actor(s) – User, Admin*

*c.Trigger – A button press will toggle dark mode.*

*d.Conditions*

*i.Pre - No pre-conditions. If it is already in Dark Mode, it will change to Light Mode, the default view.*

*ii.Post – Use case is complete when the current view has been switched ( Dark to Light, or vice versa).*

*e.Exceptions: None*

#### **3.1.16 Contact Admin**

*a.Description – Allow users of the site to contact the Admin*

*b.Actor(s) – User*

*c.Trigger – A form will exist to contact the Admin*

*d.Conditions*

*i.Pre - The user must not have sent a request to an admin recently*

*ii.Post – The form will be put in the database for review by the admin*

*e.Exceptions*

*i. Incorrect input data*

#### **3.1.17 Remove Own Account**

*a.Description – Allows the user to remove their own account.*

*b.Actor(s) – User, Admin*

*c.Trigger – A button press will trigger the removal of the user’s own account from the database.*

*d.Conditions*

*i.Pre –*

*1. The account must exist*

*2. The user must be logged into their account*

*ii. Post – The user's account and written reviews no longer exist in the database*

*e. Exceptions:*

*i. User is not logged in to the account they are trying to remove*

### **3.1.18 Edit Profile**

*a. Description – Allows the user to edit aspects of their profile.*

*b. Actor(s) – User, Admin*

*c. Trigger – A button press will bring the user to a page on which they can edit their profile.*

*d. Conditions*

*i. Pre – The user must be logged into their account.*

*ii. Post – The aspects the user has changed is applied upon the user saving the changes.*

*e. Exceptions:*

*i. Invalid format for email address*

*ii. Password does not meet password requirements*

### **3.1.19 Delete Own Review**

*a. Description – Allows the user to remove a review they previously wrote on that account.*

*b. Actor(s) – User, Admin*

*c. Trigger – A button press will trigger the removal of the selected review from the database.*

*d. Conditions*

*i. Pre –*

*1. The user must be logged into their account*

*2. The user must have written at least one review*

*ii. Post – The selected review is removed from the database.*

*e. Exceptions:*

*i. Try to remove review that is not your own*

*ii. Lost connection to database*

### **3.1.20 Edit Own Review**

*a. Description – Allows the user to edit a review they previously wrote on that account.*

*b. Actor(s) – User, Admin*

*c. Trigger – A button press will trigger the removal of the selected review from the database.*

*d. Conditions*

*i. Pre –*

*1. The user must be logged into their account*

*2. The user must have written at least one review*

*ii. Post – The selected review is removed from the database.*

*e. Exceptions:*

*i. Review has been flagged and removed from database*

### **3.1.21 Log Out of Account**

*a. Description – Allows a user to log out of the currently signed in account.*

*b. Actor(s) – User, Admin*

*c. Trigger – A button press will log the user out of their account.*

*d. Conditions*

*i. Pre – The user must be logged into their account*

*ii. Post – The user will no longer be logged in*

*e. Exceptions: None*

**3.1.22 Change Password**

*a. Description – Allows a user with an account to change their password.*

*b. Actor(s) – Users who have already made an account.*

*c. Trigger – This use case will be triggered by a button press.*

*d. Conditions*

*i. Pre – The user must click the link that is sent to their email, in order to verify they are the owner of the account.*

*User must already have an account.*

*ii. Post – The user's new password will replace the old one in the database.*

*e. Exceptions:*

*i. Account with associated email does not exist*

*ii. New password does not match*

**3.1.23 Register for Account**

*a. Description – Guest user wants to create an account.*

*b. Actor(s) – Guest User*

*c. Trigger – This use case will be triggered by a button press.*

*d. Conditions*

*i. Pre - A captcha must be completed and verified correct before the use case can finish*

*User must not be logged into an account.*

*ii. Post – User's account has been saved to the database.*

*e. Exceptions:*

*i. Account with specified email is already taken*

*ii. Password length does not meet requirement*

**3.1.24 Complete Captcha**

*a. Description – Guest users must complete a captcha in order to save to the database.*

*b. Actor(s) – Guest User*

*c. Trigger – This use case will be triggered when a Guest User tries to write a review, and when they try to register for an account.*

*d. Conditions*

*i. Pre - Guest User has data (Review, Account) that they want to save to the database.*

*ii. Post – Captcha has been verified to be correctly submitted.*

*e. Exceptions:*

*i. Captcha answer incorrect / not attempted at all*

**3.1.25 Login to Account**

*a. Description – Allow users to login to their account.*

*b. Actor(s) – User*

*c. Trigger – Clicking a button*

*d. Conditions*

*i.Pre -*

- 1. User must not already be logged in*
- 2. User must have registered an account*

*ii.Post – User has full access to their account.*

*e. Exceptions:*

*i.Email/password combination is invalid credentials*

## **3.2 Non-Functional Requirements**

### **3.2.1 Performance**

- All pages load within 2 seconds
- Web pages and database must be indexed and cached

### **3.2.2 Reliability**

- A redundant database will be implemented to ensure sufficient reliability.
- Website will be run on a cloud environment with the ability to switch host machines in case of failure.

### **3.2.3 Availability**

- All website functionality must be available for usage when the website is online.
- Website must support concurrency with several users operating the website at the same time.

### **3.2.4 Security**

- Ability to recover accounts.
- Users will securely connect to HTTPS when visiting the site.
- The certificate of the site must be up to date.
- Account passwords must be greater than 8 characters long.

### **3.2.5 Maintainability**

- Application must not have warnings or major design flaws to avoid later technical debt.
- Python classes will provide documentation for future developers for extensibility

### **3.2.6 Portability**

- Can run on more than 2 browsers on all internet connected devices.
- Website is responsive and will resize to fit the screen size of any device.

## **4. Design & Development**

### **4.1 Software Process Model**

We will use the Scrum Model for our project. We choose Scrum as having 2 week sprints would help us prepare for assignment deadlines. We believe that this approach will allow us to deliver our high-priority items first in a quick and efficient way. We plan to have stand-up on Monday through Friday from 3:30PM-3:45PM CST. Sprint reviews will be every other Tuesday.

## 4.2 Deliverable 1

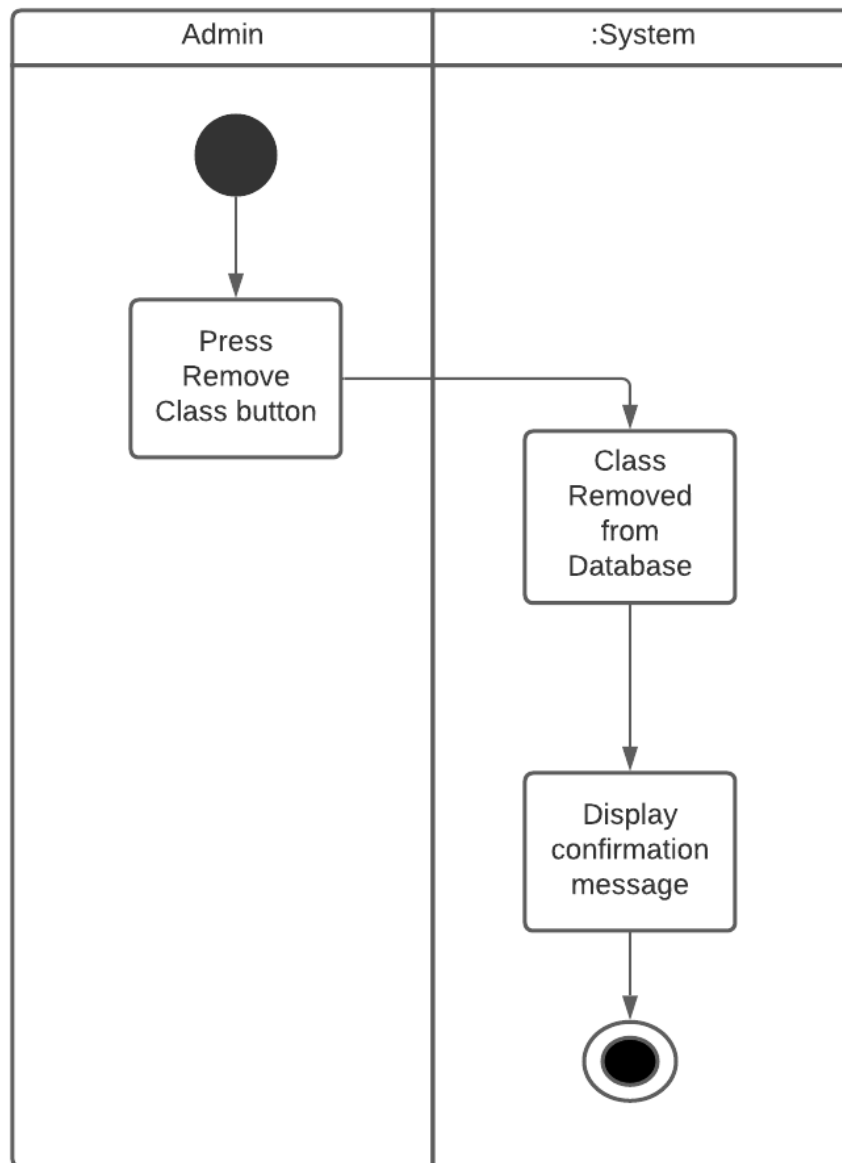
### 4.2.1 Description

For this deliverable, we solidified the agile methodology we are going to use. We chose the requirements each team member will be responsible for, and created activity diagrams for each of those requirements.

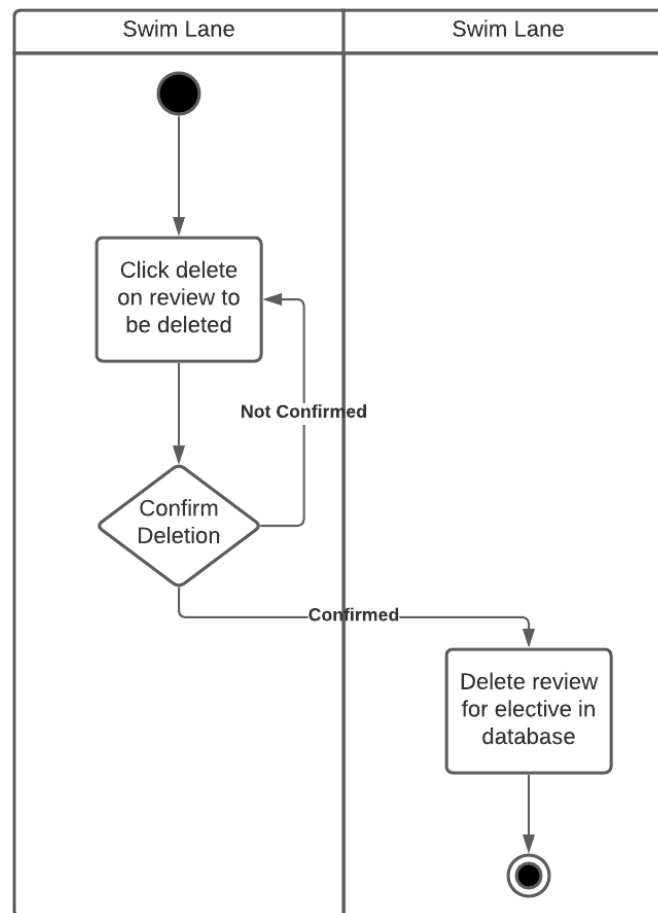
### 4.2.2 Design

#### Activity Diagrams:

##### 4.2.2.1: Remove a Class

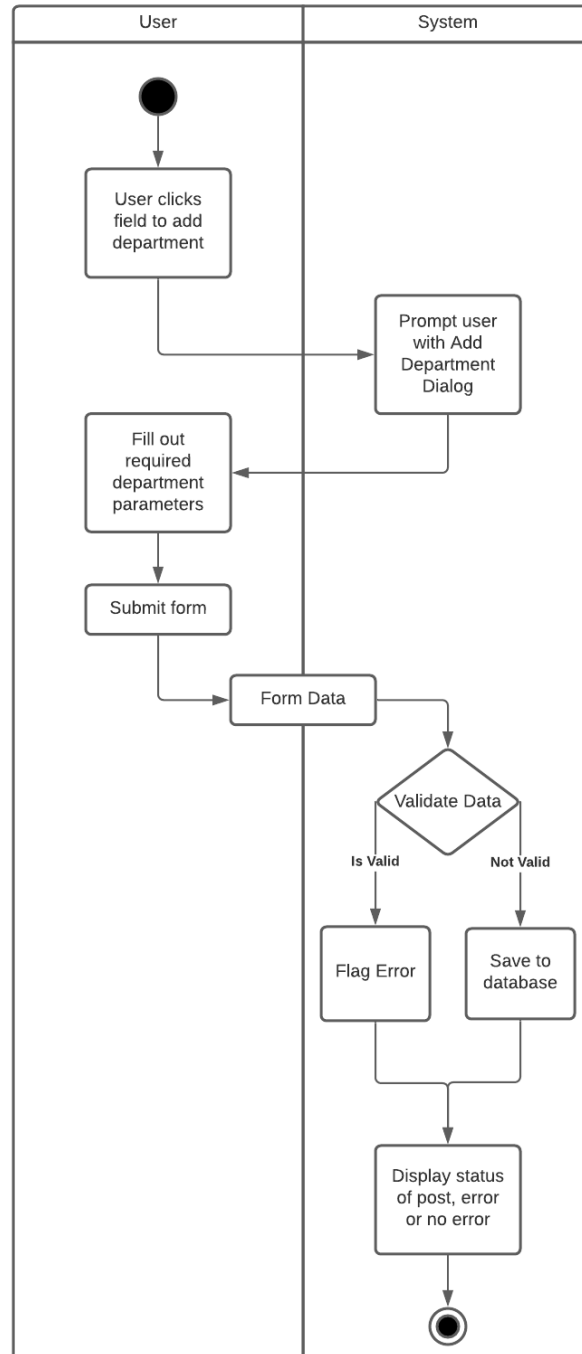


##### 4.2.2.2: Remove Others Reviews



*The activity diagram shown in Figure 2 explains the process of an admin user deleting another user's review.*

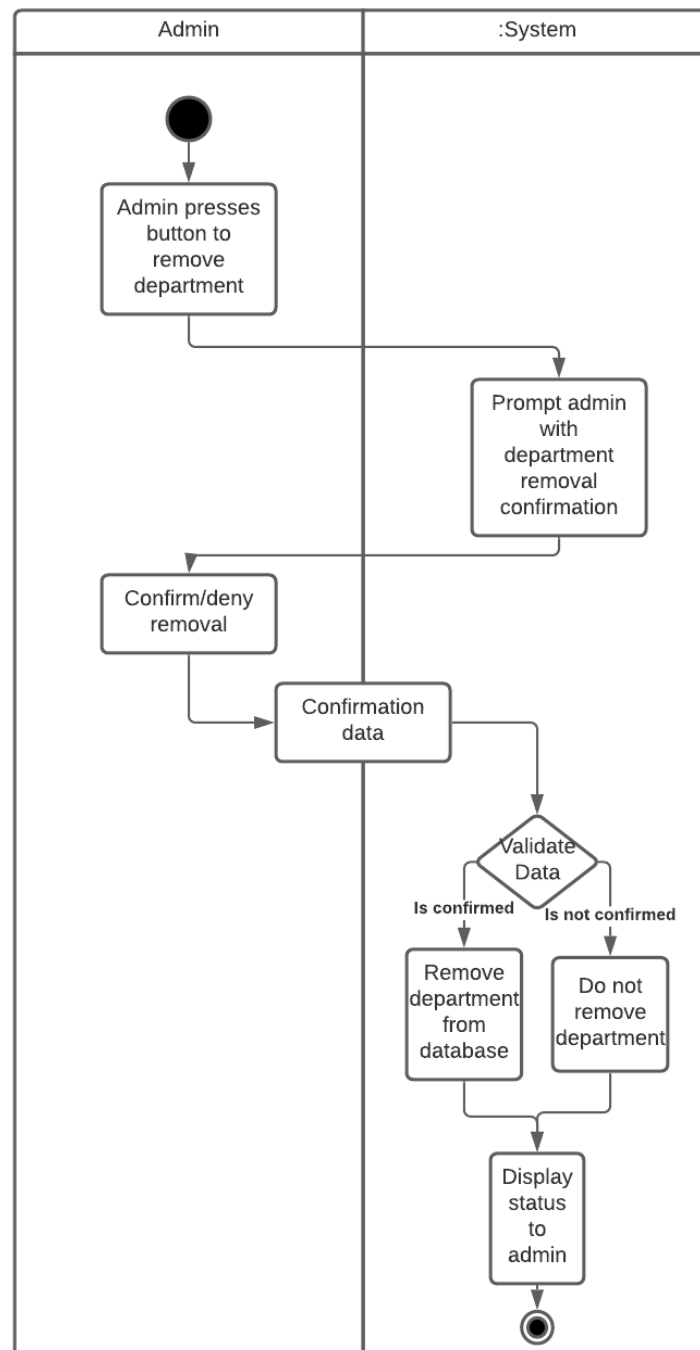
#### 4.2.2.3: Add department Activity Diagram



The activity diagram shown in 4.2.2.3 explains the process of a user adding a department.

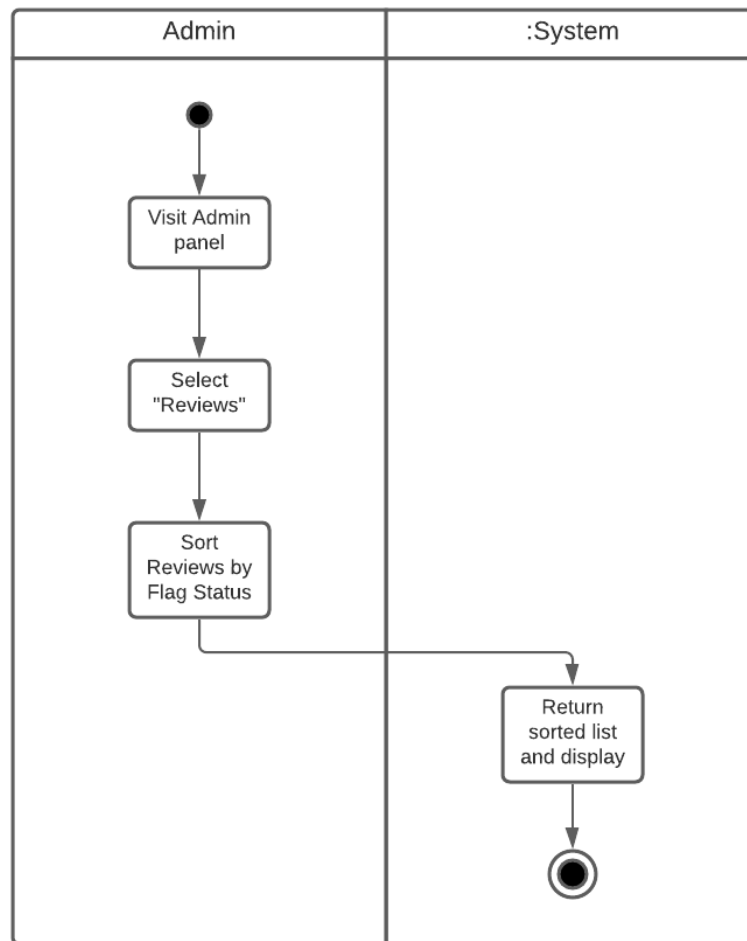
#### 4.2.2.4: Removing A Department Activity Diagram





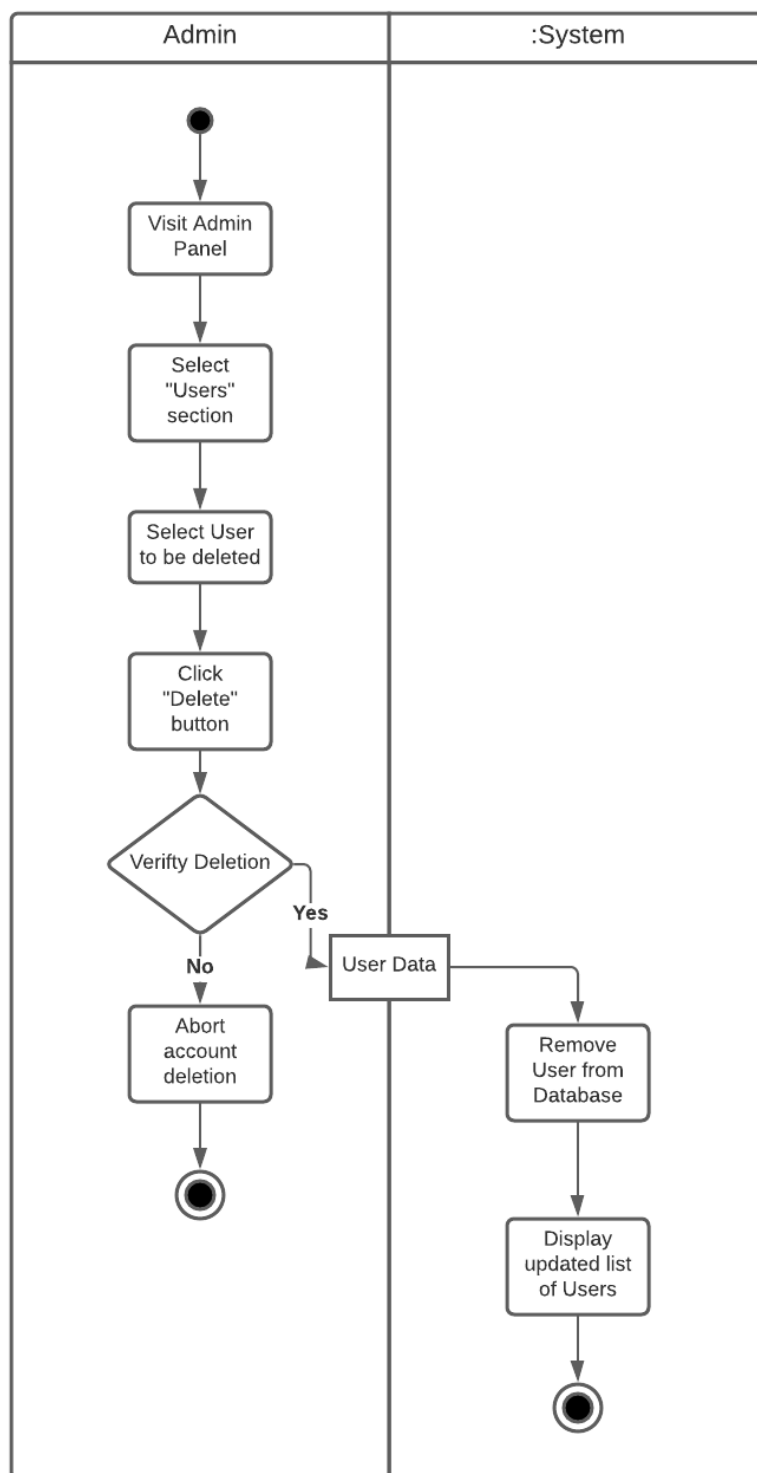
*The activity diagram shown in 4.2.2.4 shows the process of a admin removing a department from the database*

#### 4.2.2.5: View Flagged Reviews



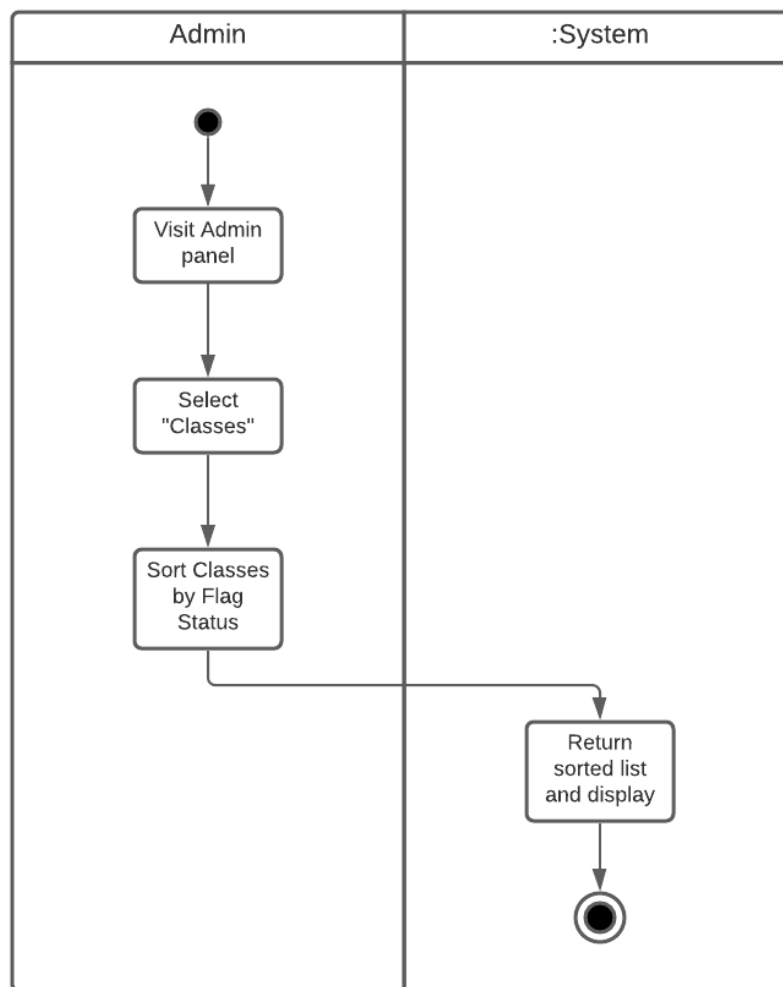
*The activity diagram shown in Figure 5 explains the process of an admin viewing flagged reviews.*

#### 4.2.2.6: Remove Other User's Account



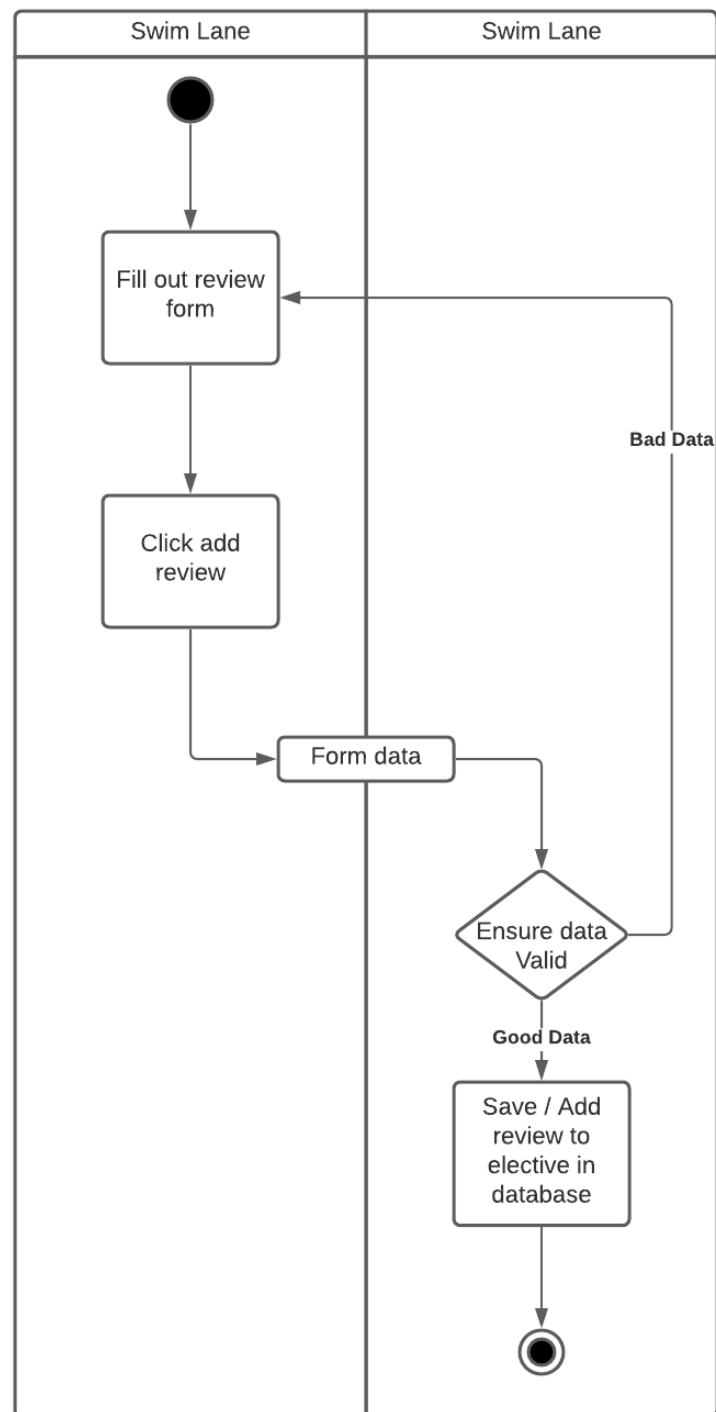
The activity diagram shown in Figure 6 explains the process of an admin removing another user's account.

#### 4.2.2.7: View Flagged Classes



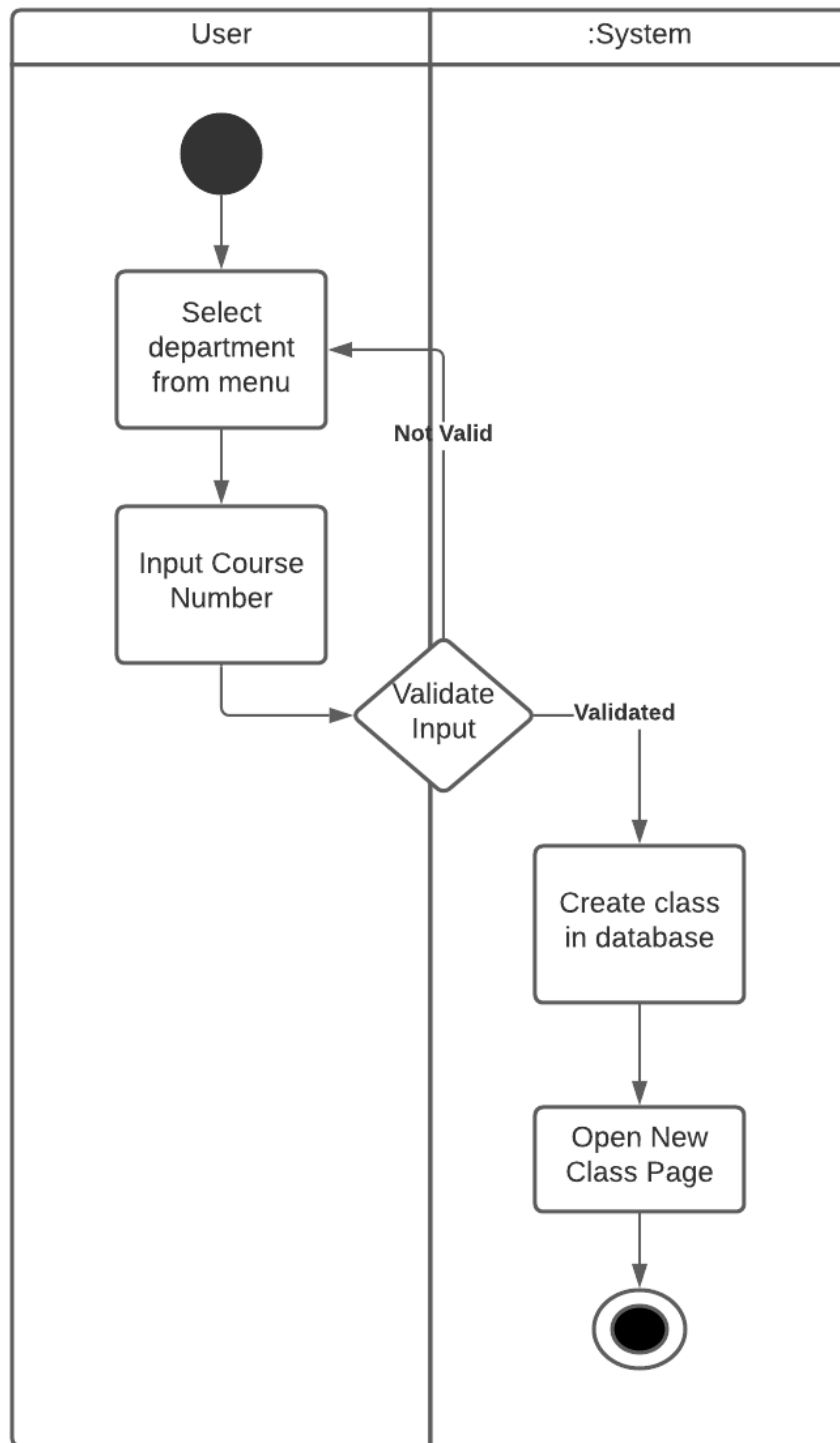
*The activity diagram shown in Figure 7 explains the process of an admin viewing flagged classes.*

#### ***4.2.2.8: Write Review***

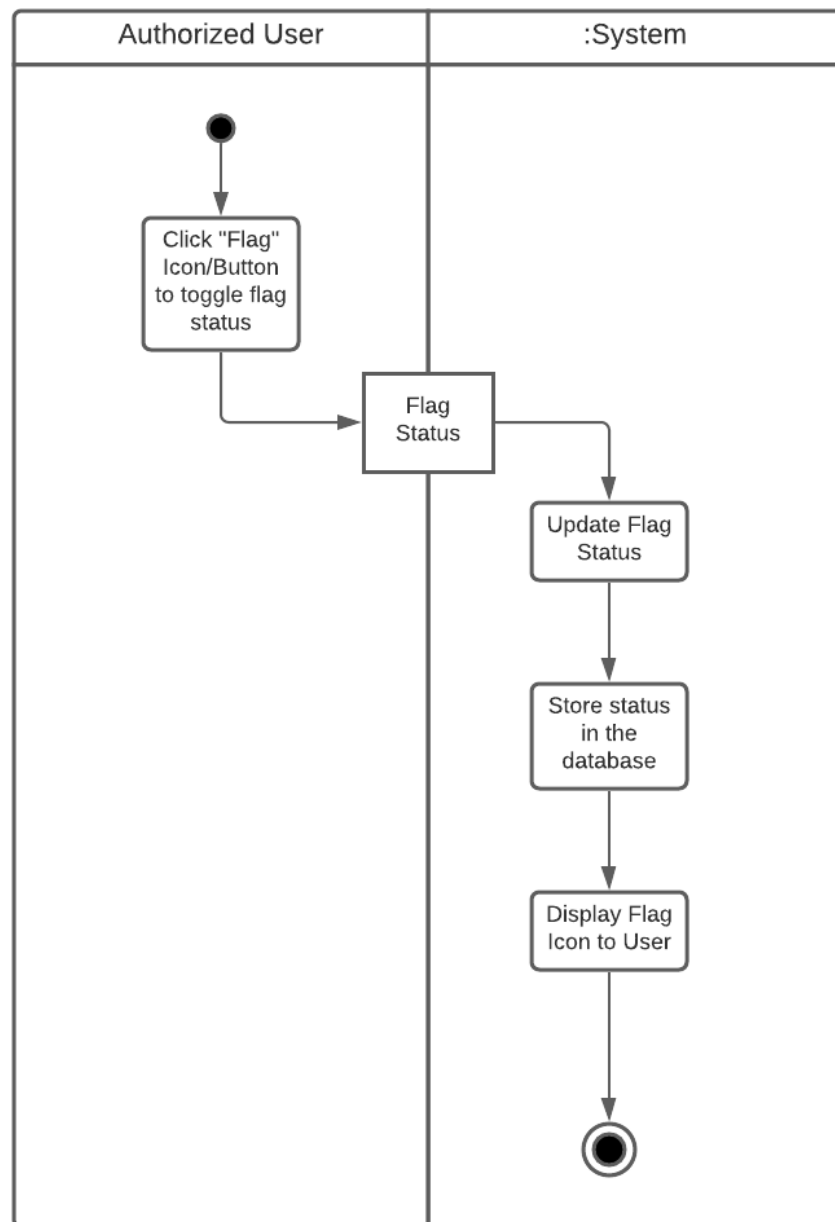


*The activity diagram shown in Figure 8 explains the process of a user writing a review.*

#### **4.2.2.9: Add A Class**

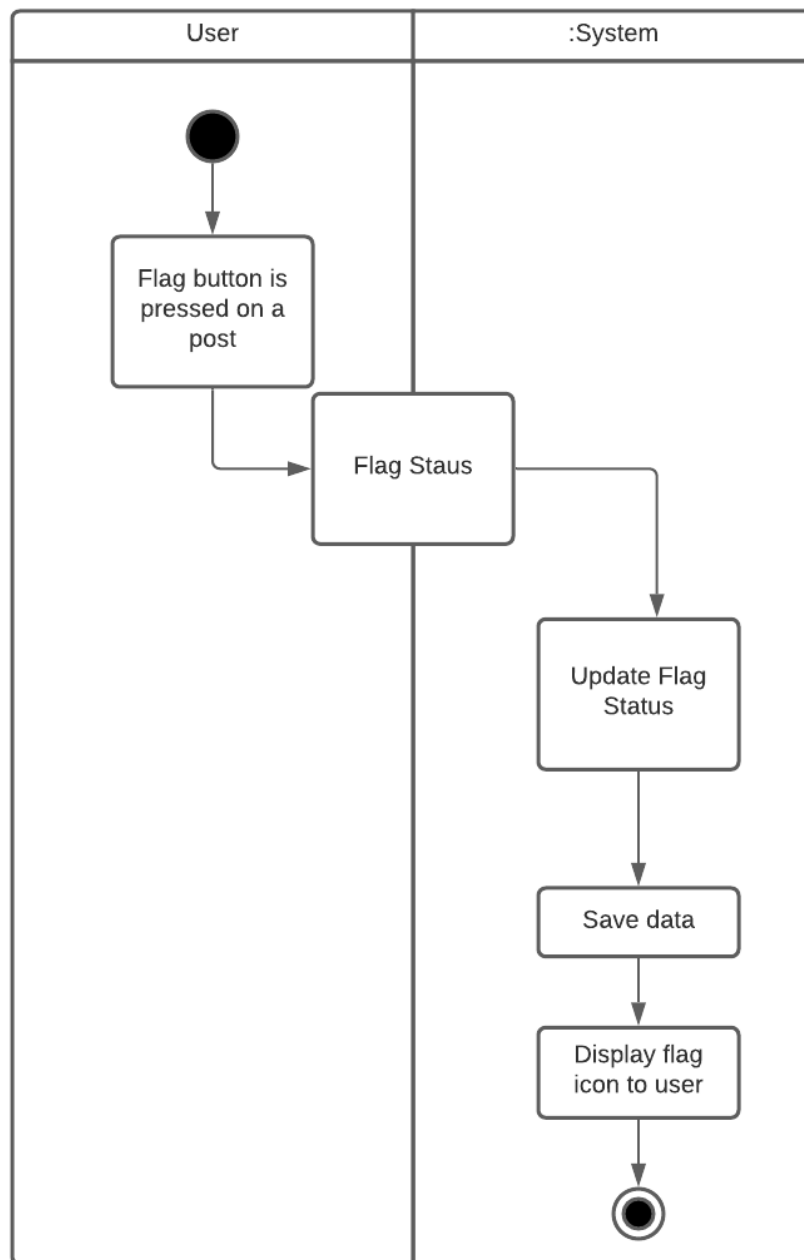


#### 4.2.2.10: Flag a Review



The activity diagram shown in Figure 10 explains the process of a user flagging another user's review.

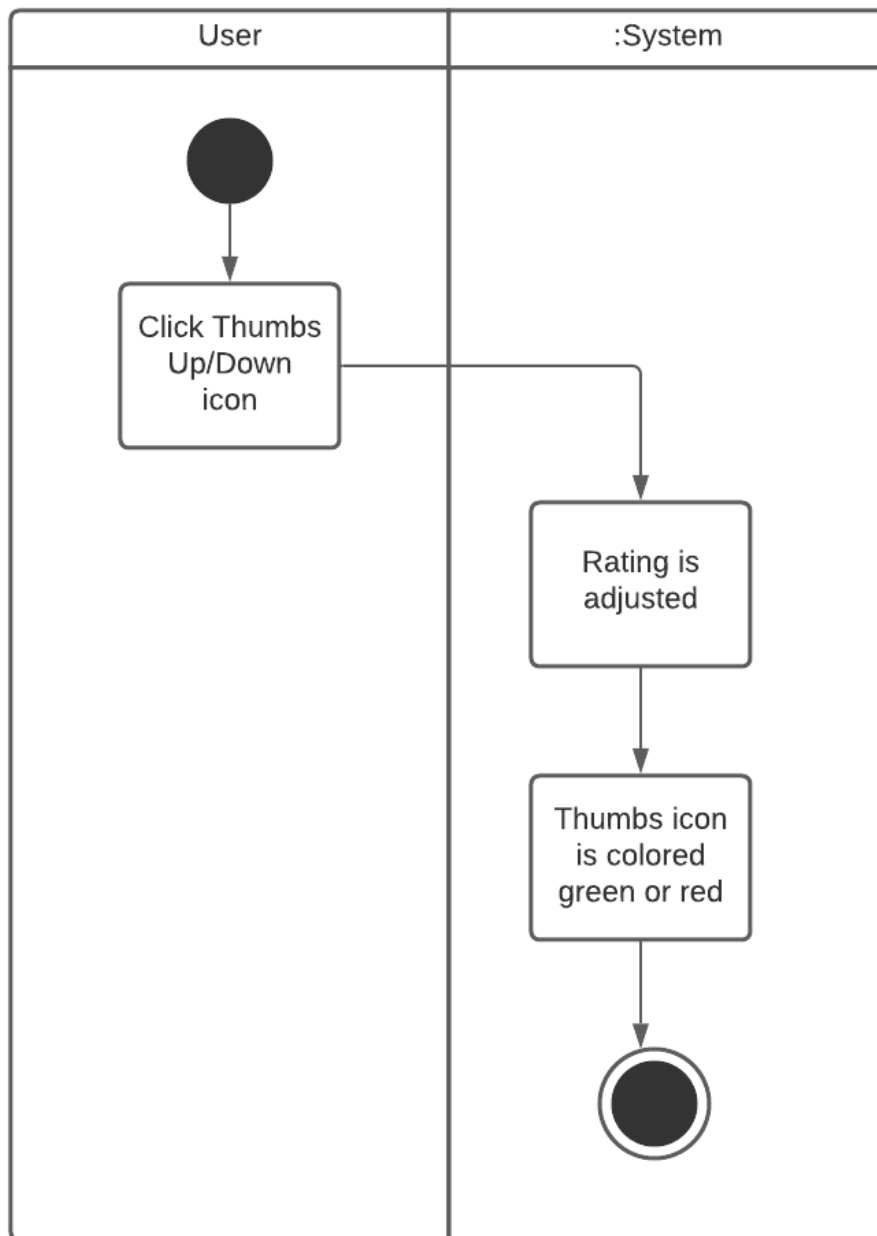
#### 4.2.2.11: *Flagging a Class*



*The activity diagram shown in 4.2.2.10 shows the process of a user flagging a class*

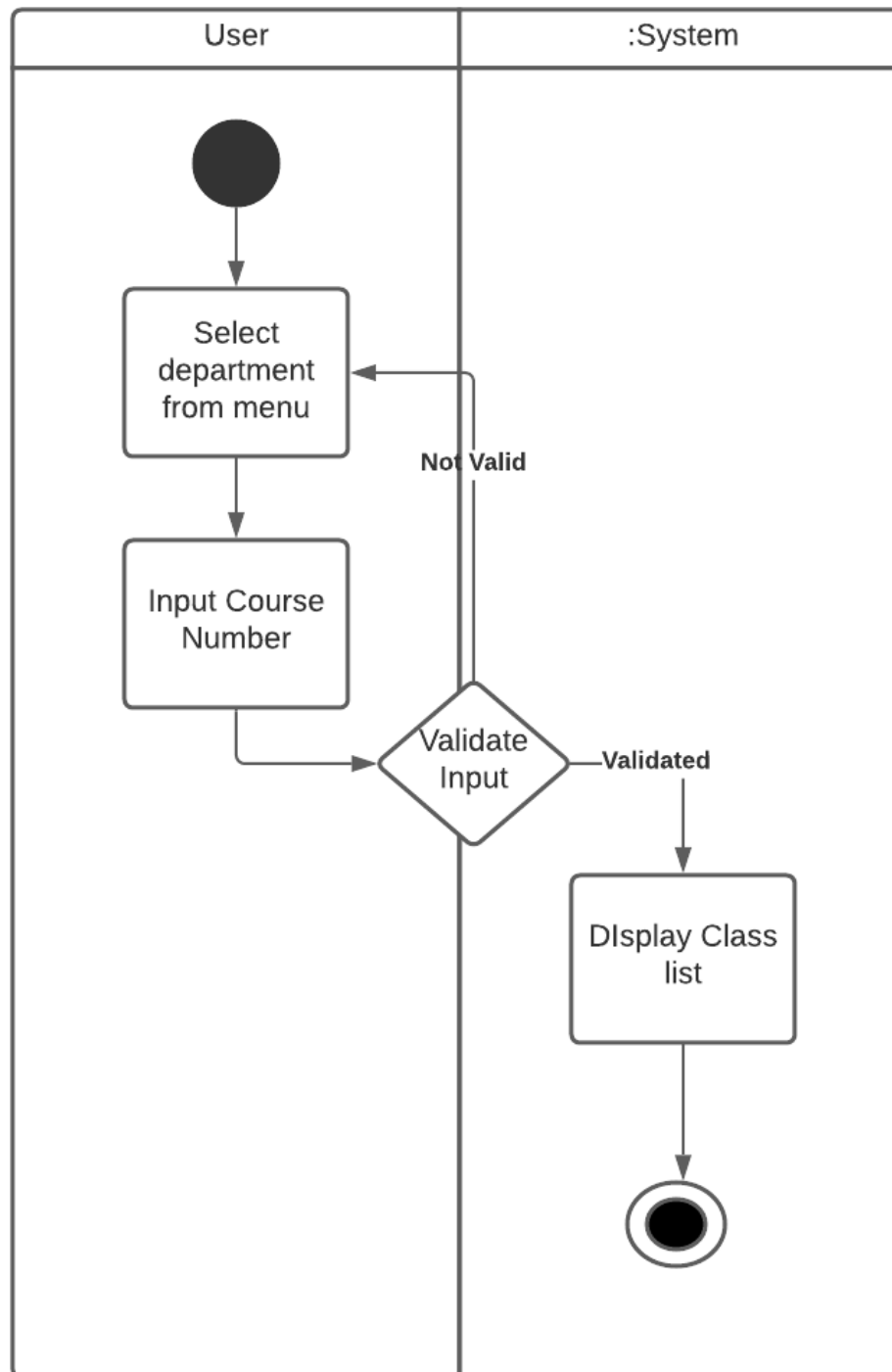
#### 4.2.2.12 *Rate Review*



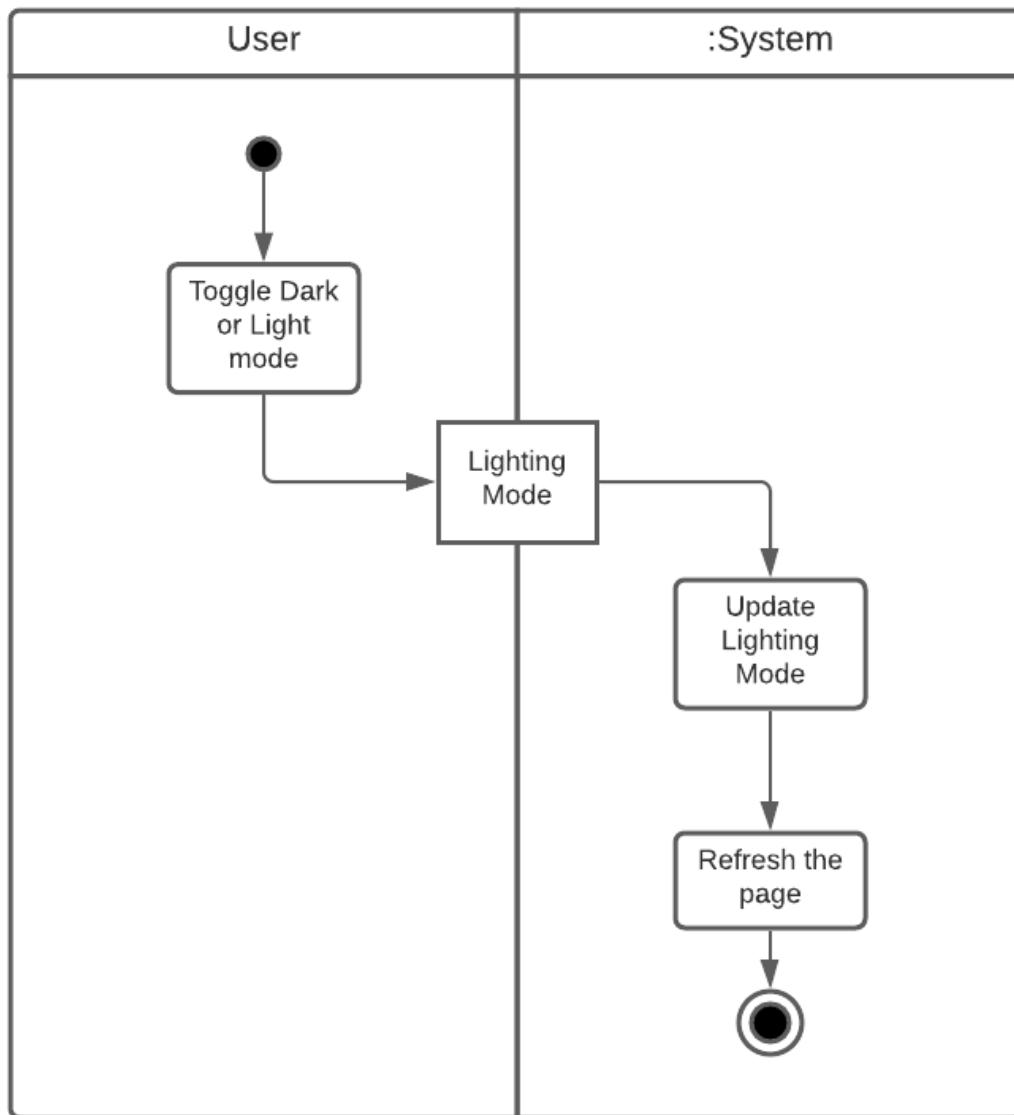


**The Thumbs Up/Down buttons will be located at the bottom right corner of every review.  
The thumbs up button will be colored green if clicked on, and the thumbs down button will be colored red if clicked on.**

#### ***4.2.2.13: Search for a Class***

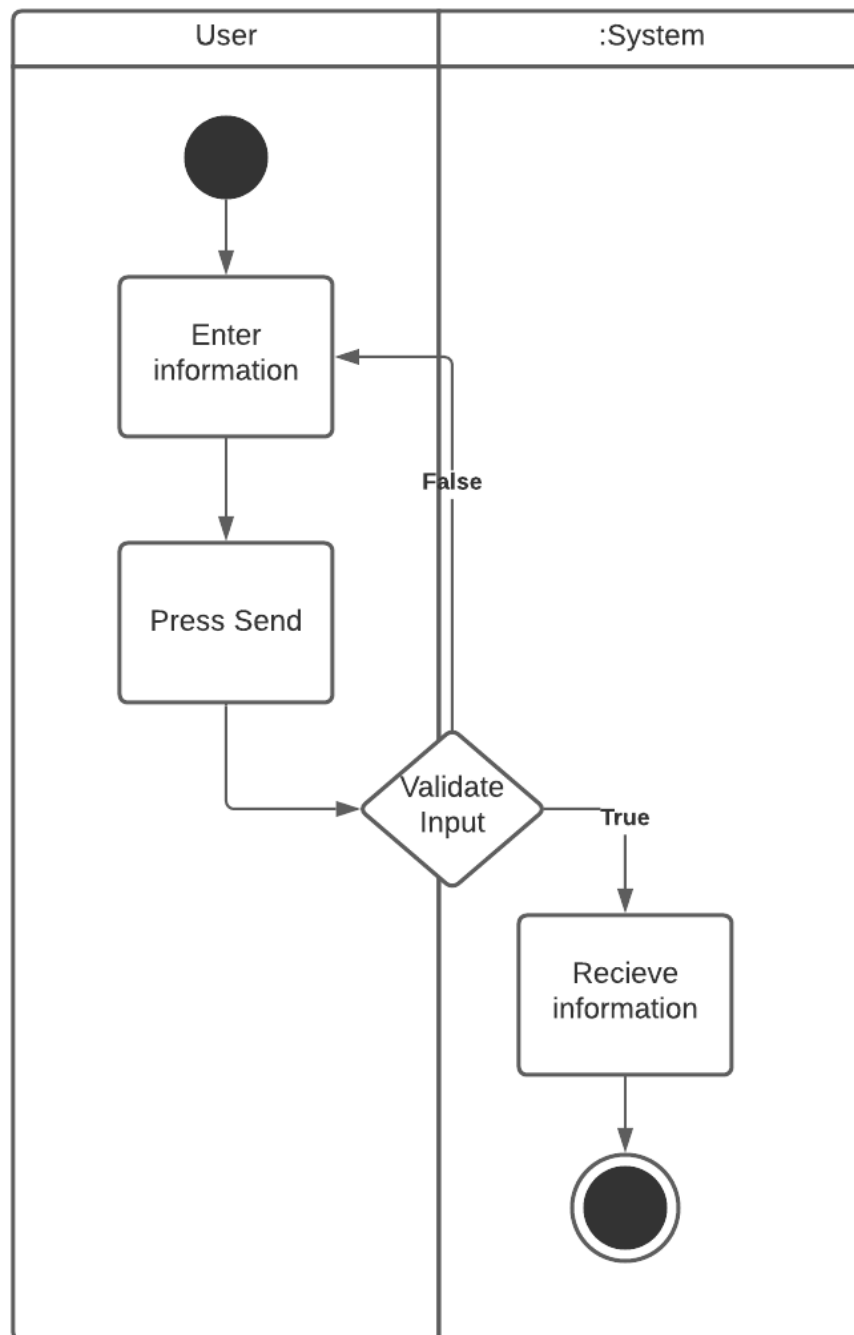


#### 4.2.2.14: Toggle Dark Mode

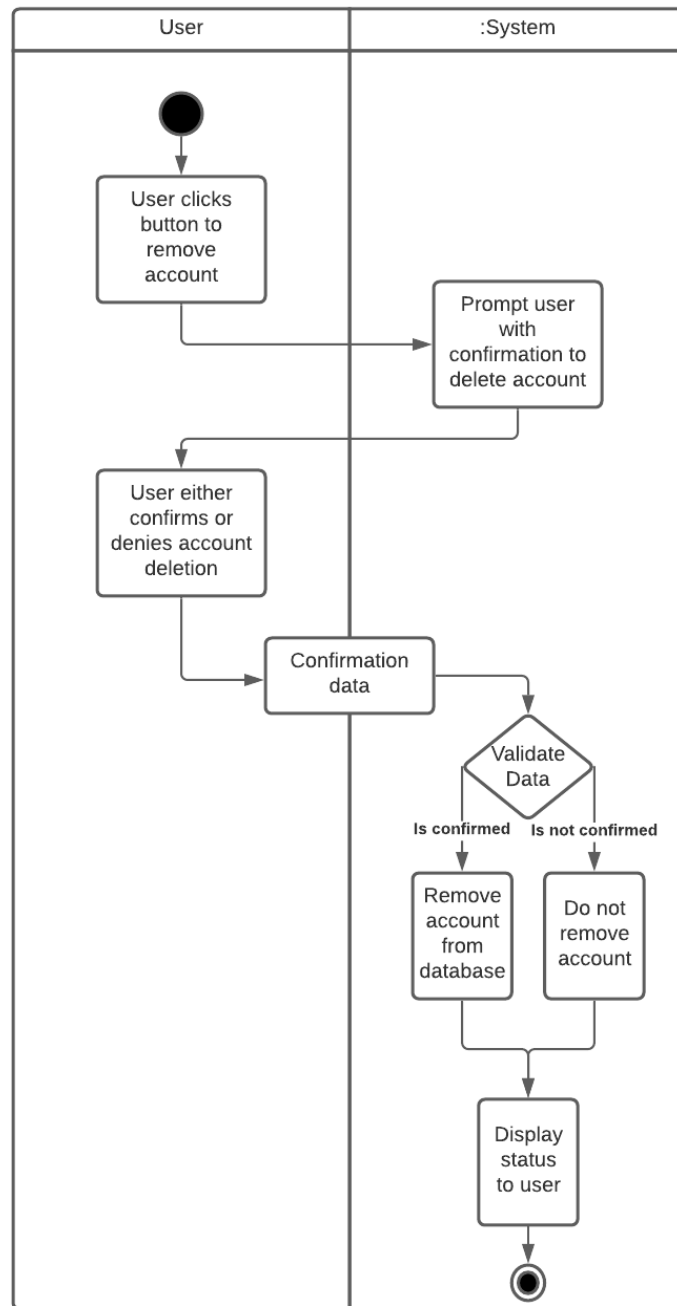


The activity diagram shown in Figure 14 explains the process of a user toggling dark mode.

#### 4.2.2.15: Contact Admin

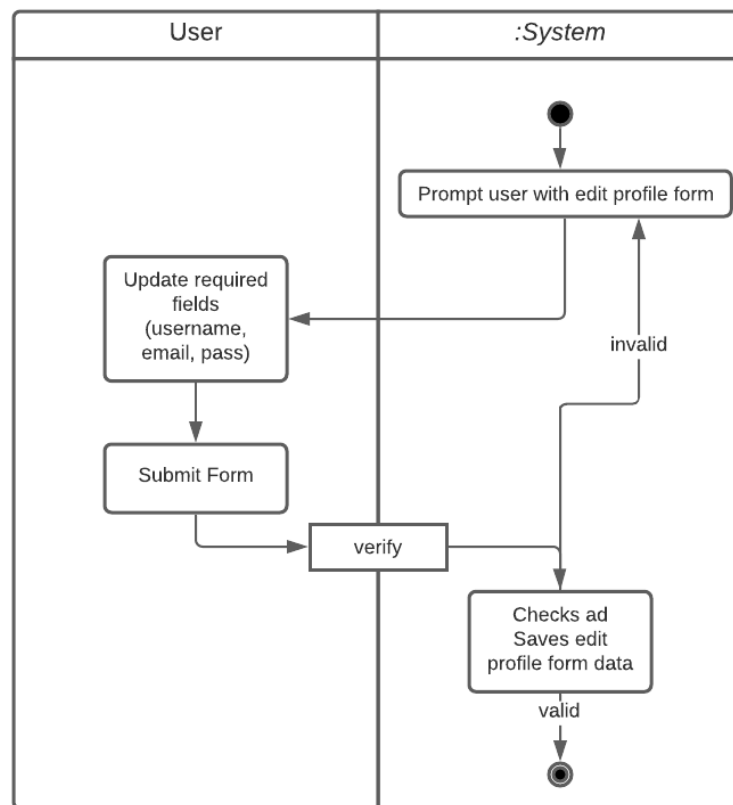


***4.2.2.16: Removing a Users Own account***



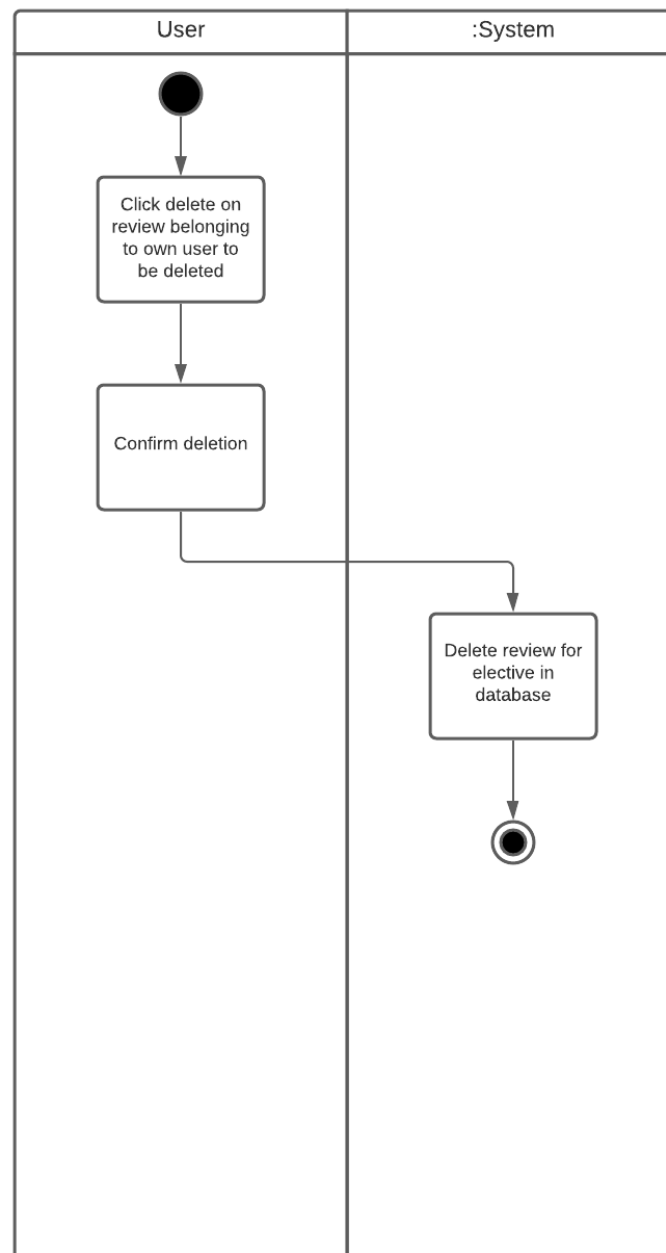
*The activity diagram shown in 4.2.2.16 shows the process of a user deleting their own account*

#### **4.2.2.17: Edit Profile**



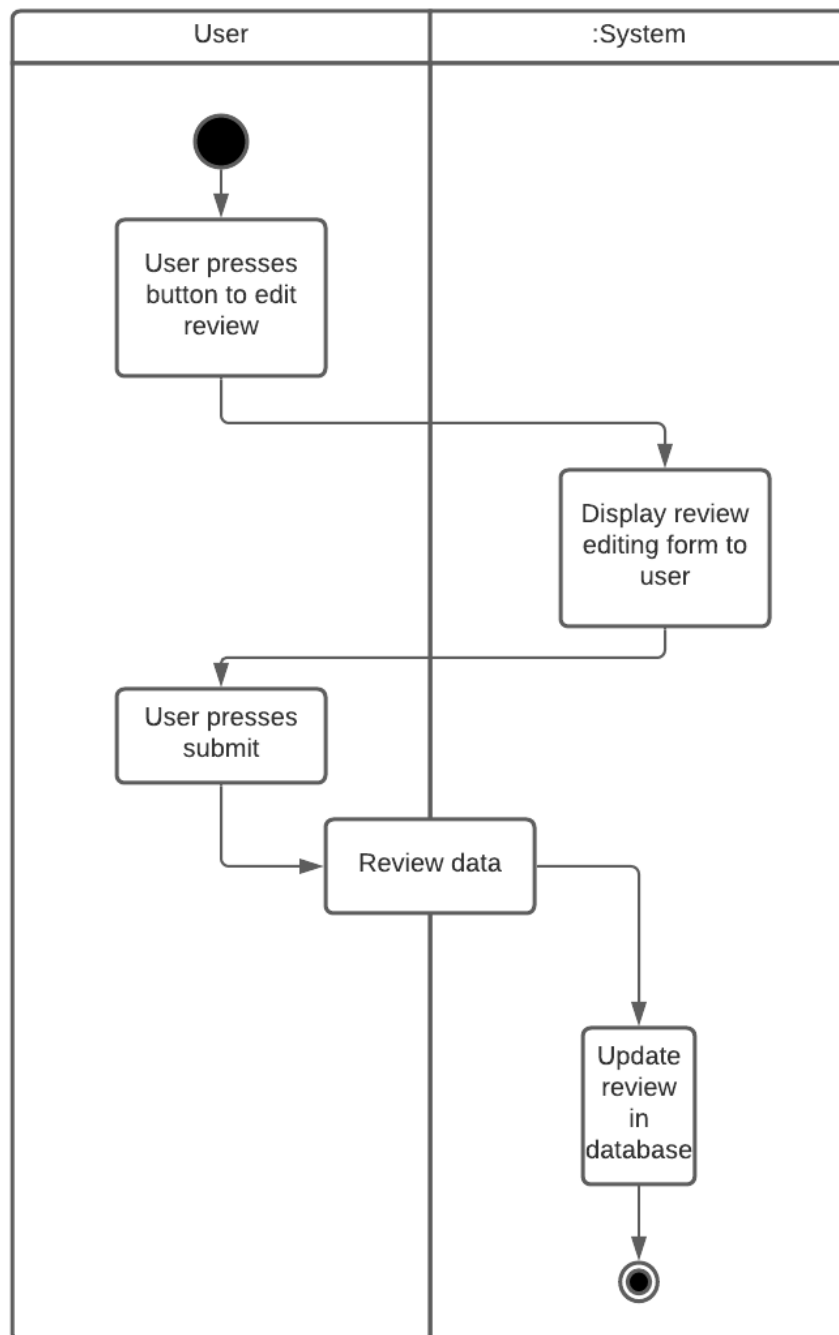
*The activity diagram shown in Figure 17 explains the process of a user updating their account information.*

#### **4.2.2.18: Delete Own Review**



*The activity diagram in Figure 18 explains the process of a user deleting their own review.*

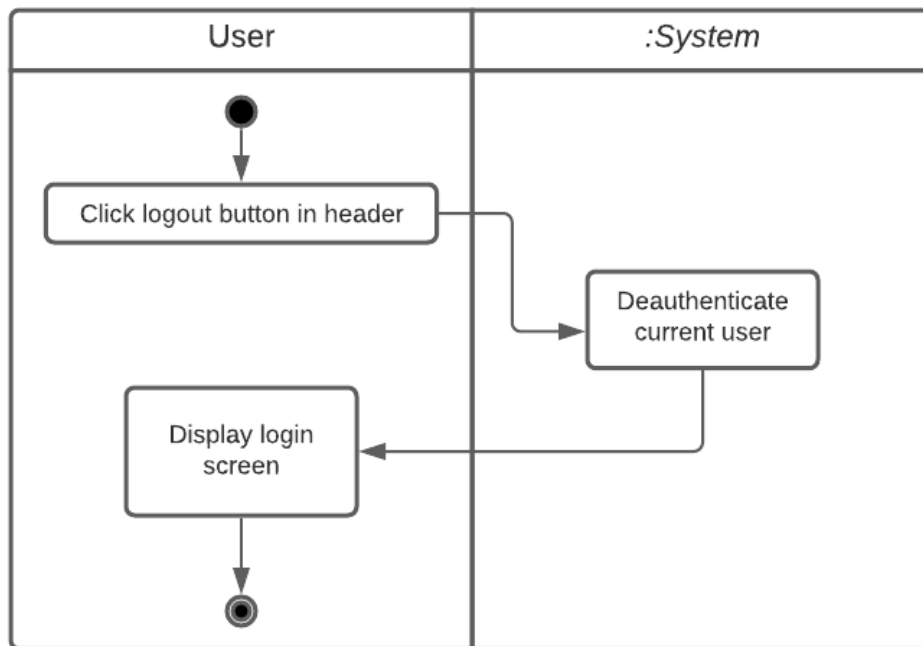
#### ***4.2.2.19: Edit a users own review***



*The activity diagram shown in 4.2.2.19 shows the process of a user editing their own review*

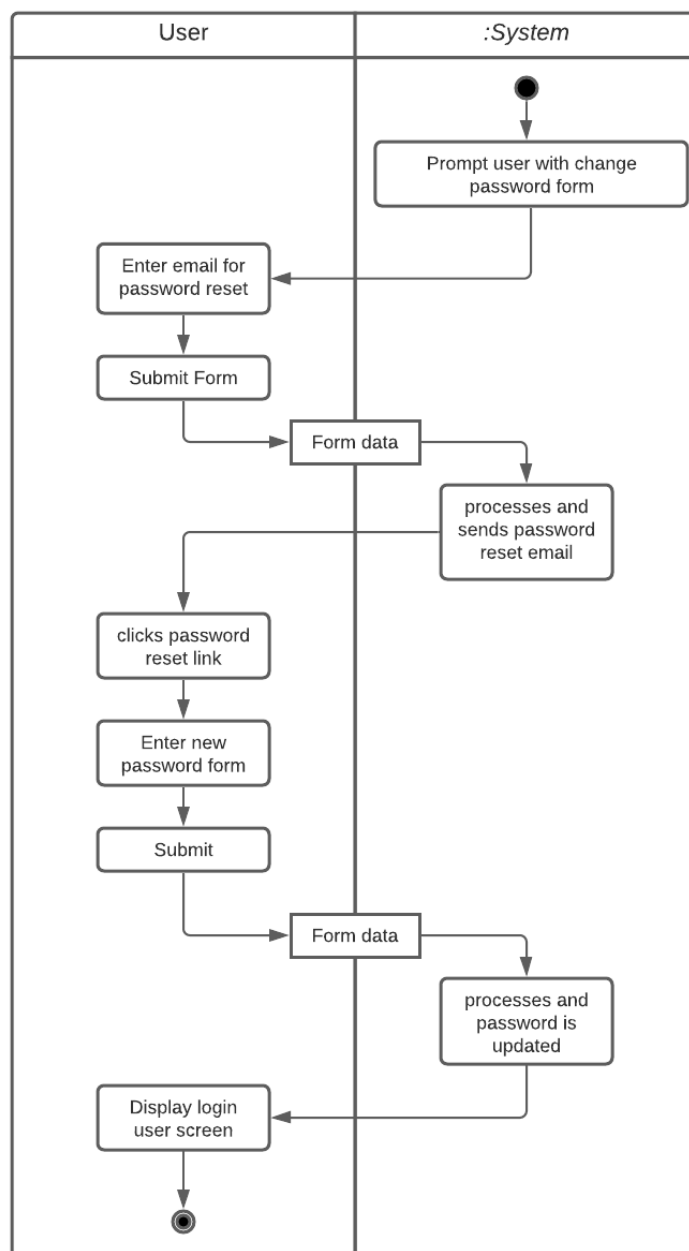


#### 4.2.2.20: Logout



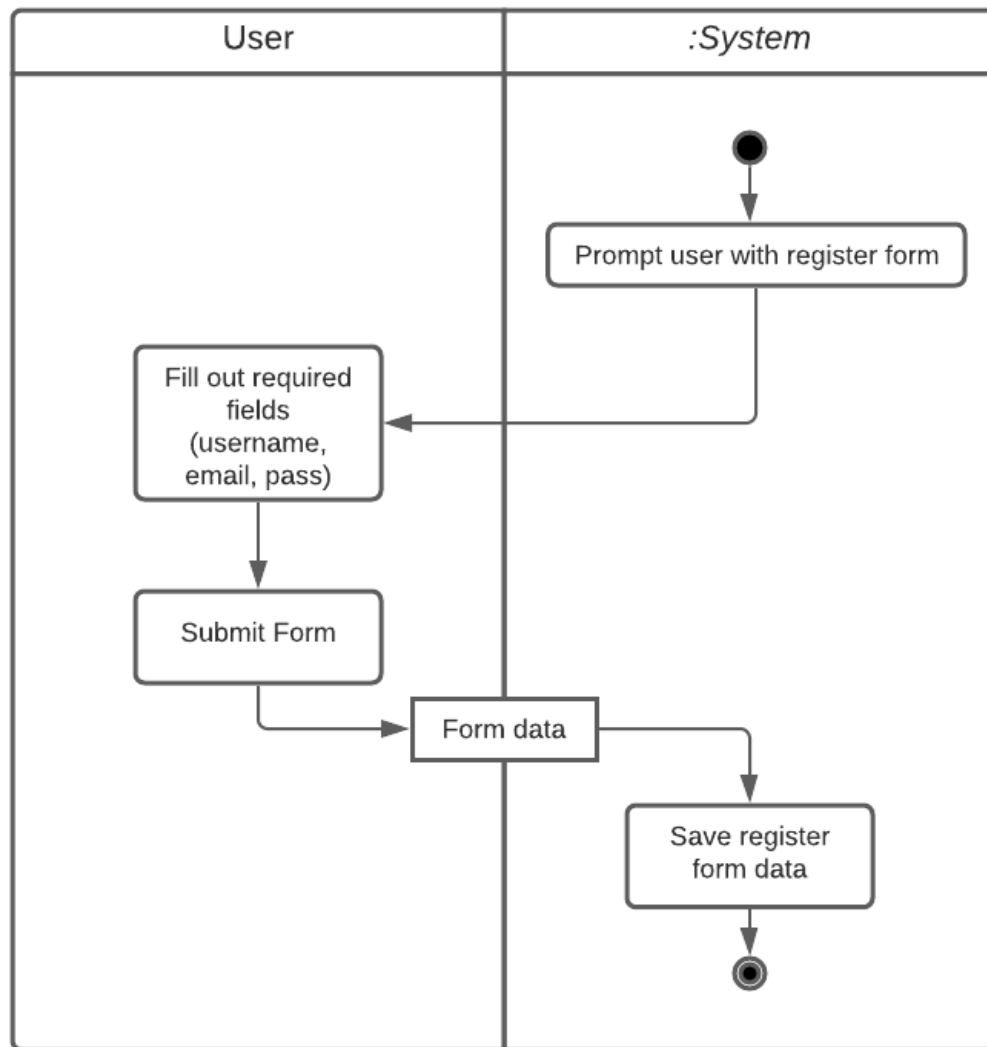
*The activity diagram shown in Figure 20 explains the process of a user logging out of their account.*

#### 4.2.2.21: Change Password



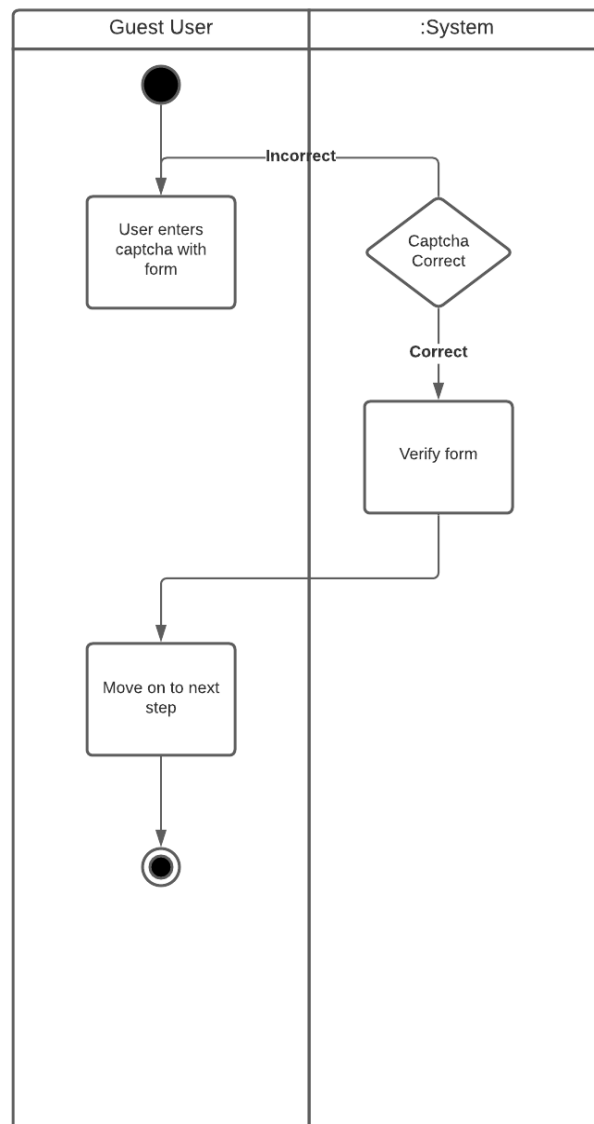
*The activity diagram shown in Figure 21 explains the process of changing their password when they are not able to access their account.*

#### 4.2.2.22: Register Account



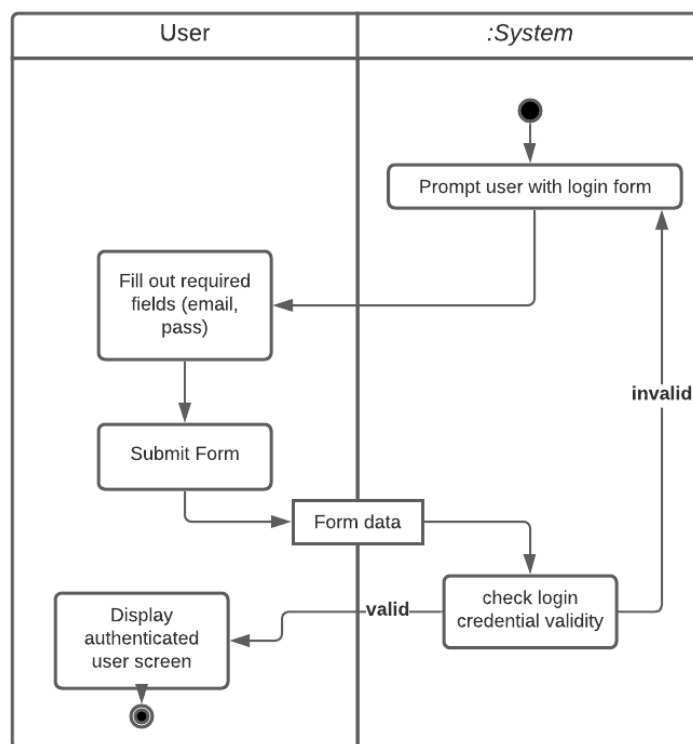
*The activity diagram shown in Figure 22 explains the process of a user completing the account registration form.*

#### ***4.2.2.23: Complete Captcha***



*The activity diagram shown in Figure 23 explains the process of a user completing a captcha.*

#### **4.2.2.24: Log In**



The activity diagram shown in Figure 24 explains the process of a user logging into a current existing account.

## 4.3 Deliverable 2

### 4.3.1 Description

For this Deliverable, we created a class diagram of all our models. In addition, we also started development of our program in Django.

### 4.3.2 Design

#### 4.3.2.1 Class Diagram

The class diagram shown in this section captures the primary entities within the application, as well as their relationships with each other.

##### **Department -**

Contains data relating to the different ISU departments and what courses are offered as electives by each.

##### **Course -**

Captures information of all offered elective courses, what department each course belongs to, and which teacher is teaching that section.

##### **Teacher -**

An abstraction that captures information regarding the teacher of an elective course.

**Professor -**

An implementation of Teacher that contains information for ISU professors that teach elective courses.

**Graduate Student -**

An implementation of Teacher that contains information for graduate students that teach elective courses.

**Review -**

Captures all reviews for a course taught by a specific teacher.

**UserAccount -**

An abstraction that captures information and functionality needed for each user.

**RegularUser -**

An implementation of UserAccount that captures functionalities possible for regular users and not admins.

**Admin -**

An implementation of UserAccount that captures further information and functionalities possible for admins and not regular users.

**Model-**

Contains the methods used to save and delete items from the database.

**Database Gateway-**

Provides a connection to the database to retrieve queries.

**Contact-**

Captures the queries sent by users, along with their name and email.

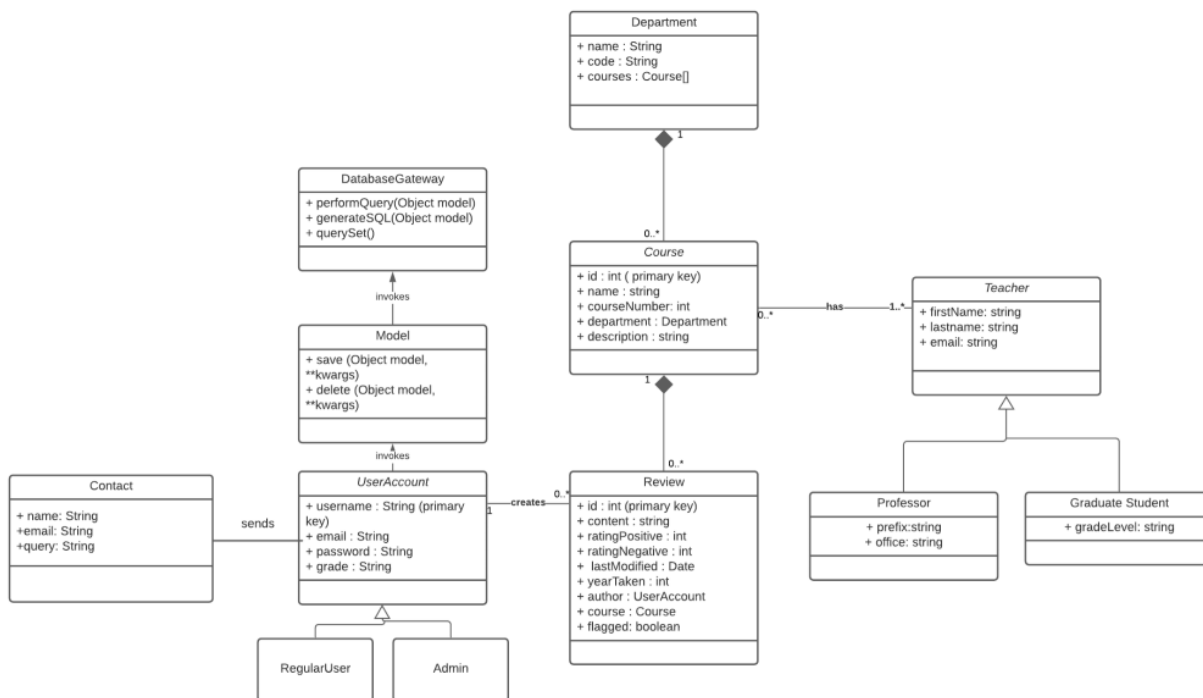


Figure 26: Class diagram

### 4.3.3 Development

#### 4.3.3.1 Description

In Django, we have started to create the classes present in the class diagram shown in section 4.3.2.1, and we have connected a database with PostgreSQL to store the data encapsulated in those classes. We also have completed the requirements related to logging in and logging out, as well as the ability for administrators to access an admin panel.

```
website/models.py ×
1  from django.db import models
2  from django.contrib.auth.models import AbstractUser, User
3  from django.core.validators import MaxValueValidator, MinValueValidator
4
5
6  class UserAccount(AbstractUser):
7      id = models.AutoField(primary_key=True)
8      #built in username and password
9      grade_level = models.IntegerField()
10
11     def __str__(self):
12         return self.username
13
14     class Department(models.Model):
15         code = models.CharField(primary_key=True, max_length=4)
16         name = models.CharField(max_length=40)
17
18         def __str__(self):
19             return self.name
20
21     class Course(models.Model):
22         id = models.AutoField(primary_key=True)
23         name = models.CharField(max_length=69)
24         course_number = models.IntegerField()
25         department = models.ForeignKey(Department, on_delete=models.CASCADE)
26         description = models.TextField()
27
28         def __str__(self):
29             return self.name
30
31
```



```
website/models.py x
30
31
32 class Teacher(models.Model):
33     id = models.AutoField(primary_key=True)
34     first_name = models.CharField(max_length=15)
35     last_name = models.CharField(max_length=20)
36     email = models.CharField(max_length=40)
37     def __str__(self):
38         return str(self.first_name) + " " + str(self.last_name)
39     class Meta:
40         abstract = True
41
42 class Professor(Teacher):
43     prefix = models.BooleanField()
44     office = models.CharField(max_length=15)
45
46 class GraduateStudent(Teacher):
47     grade_level = models.IntegerField()
48
49 class Review(models.Model):
50     id = models.AutoField(primary_key=True)
51     content = models.TextField()
52     rating_positive = models.IntegerField()
53     rating_negative = models.IntegerField()
54     last_modified = models.DateTimeField(auto_now=True)
55     year_taken = models.PositiveSmallIntegerField(blank=True, null=True)
56     author = models.ForeignKey(UserAccount, on_delete=models.CASCADE)
57     course = models.ForeignKey('Course', on_delete=models.CASCADE)
58     flagged = models.BooleanField()
59
60     def __str__(self):
61         return str(self.author.name) + " " + str(self.last_modified)
62
```

Figure 26: Our Models coded in the models.py file on Django

## 4.4 Deliverable 3

### 4.4.1 Description

For this Deliverable, we created 10 sequence diagrams outlining some of the requirements of our project. We also set up the User Interface for many of our requirements.

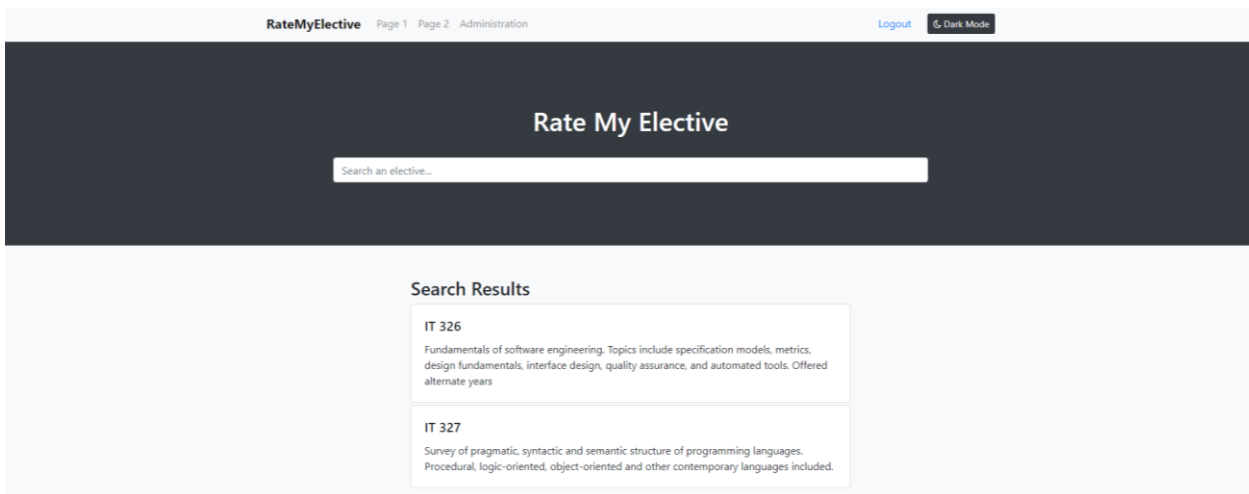
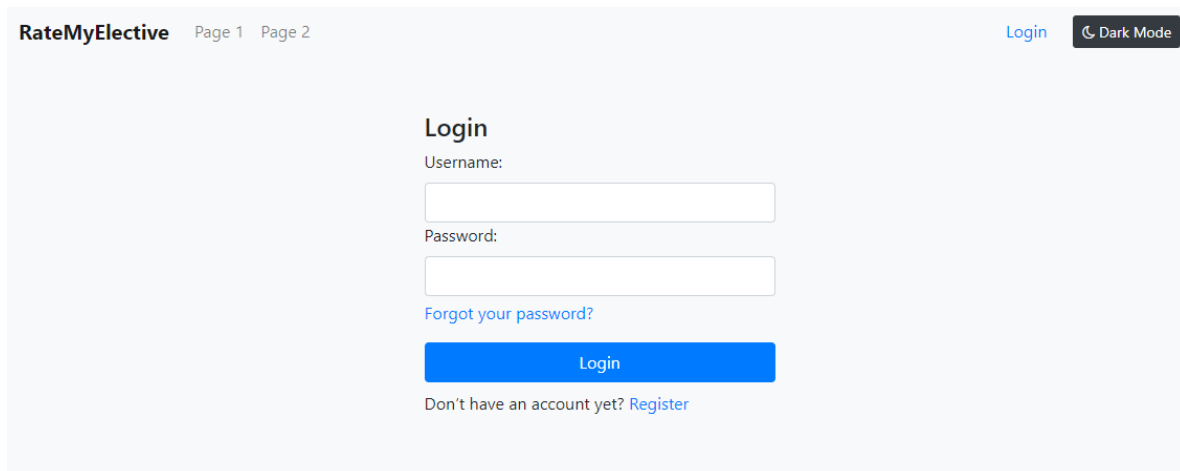


Figure 27: Current user interface home screen for RateMyElective



The login screen for RateMyElective features a header with the logo, page navigation, and a dark mode toggle. The main content area is titled 'Login' and contains input fields for 'Username:' and 'Password:'. A 'Forgot your password?' link is positioned below the password field. A blue 'Login' button is centered below the inputs, and a link to 'Register' is at the bottom for users without an account.

RateMyElective Page 1 Page 2 Login Dark Mode

### Login

Username:

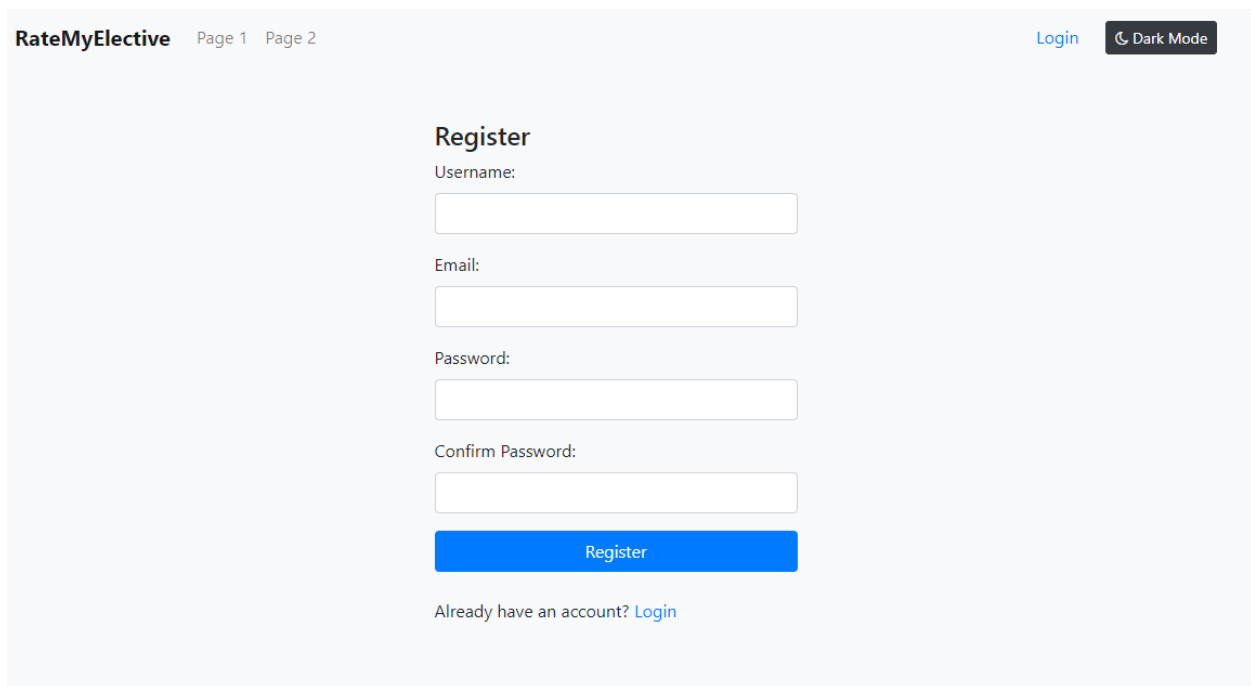
Password:

[Forgot your password?](#)

Login

Don't have an account yet? [Register](#)

Figure 28: Current user interface login screen for RateMyElective



The register screen for RateMyElective features a header with the logo, page navigation, and a dark mode toggle. The main content area is titled 'Register' and contains input fields for 'Username:', 'Email:', 'Password:', and 'Confirm Password:'. A blue 'Register' button is centered below the inputs, and a link to 'Login' is at the bottom for users who already have an account.

RateMyElective Page 1 Page 2 Login Dark Mode

### Register

Username:

Email:

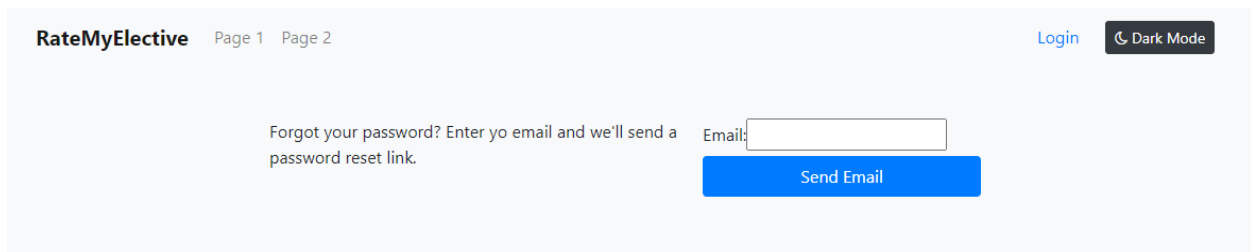
Password:

Confirm Password:

Register

Already have an account? [Login](#)

Figure 29: Current user interface register screen for RateMyElective



The forgot password screen for RateMyElective features a header with the logo, page navigation, and a dark mode toggle. The main content area contains a text prompt asking for an email address to receive a password reset link. An input field for 'Email:' is provided, followed by a blue 'Send Email' button.

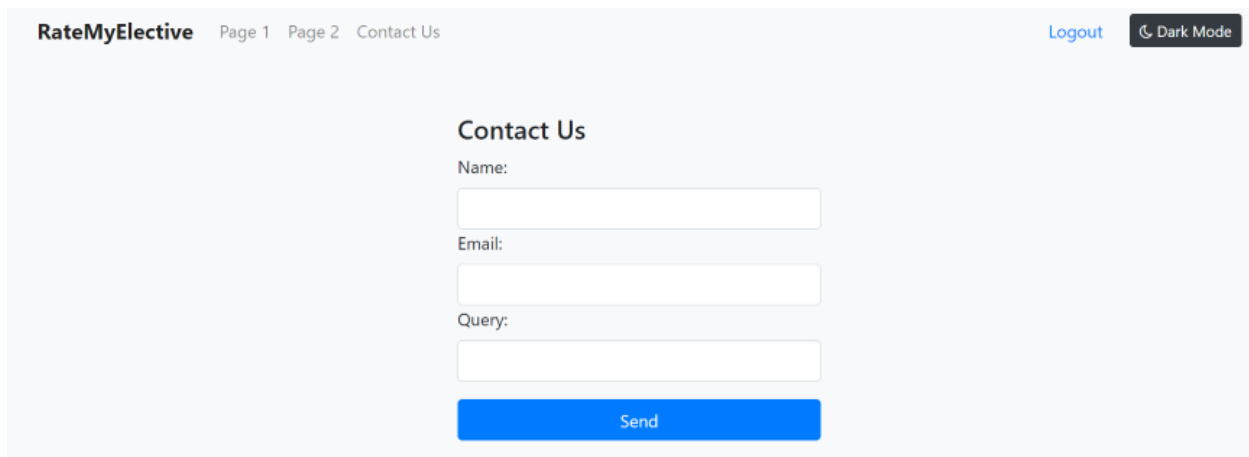
RateMyElective Page 1 Page 2 Login Dark Mode

Forgot your password? Enter yo email and we'll send a password reset link.

Email:

Send Email

Figure 30: Current user interface forgot password screen for RateMyElective



The screenshot shows the 'Contact Us' form on the RateMyElective website. The header includes the site name 'RateMyElective' and navigation links 'Page 1', 'Page 2', and 'Contact Us'. There are also links for 'Logout' and a 'Dark Mode' toggle. The form itself is titled 'Contact Us' and contains three input fields labeled 'Name:', 'Email:', and 'Query:'. A blue 'Send' button is positioned at the bottom of the form.

Figure 31: Form to contact the developers.

### Review Title

Up

Down

Flag

from Jakob Lybarger, taken Fall 2021

This is the text for the review, rishi is a pretty cool dude and I definitely failed califfs test

Date Posted '11-17-2021'

Figure 32: Review card to display a users review of an elective

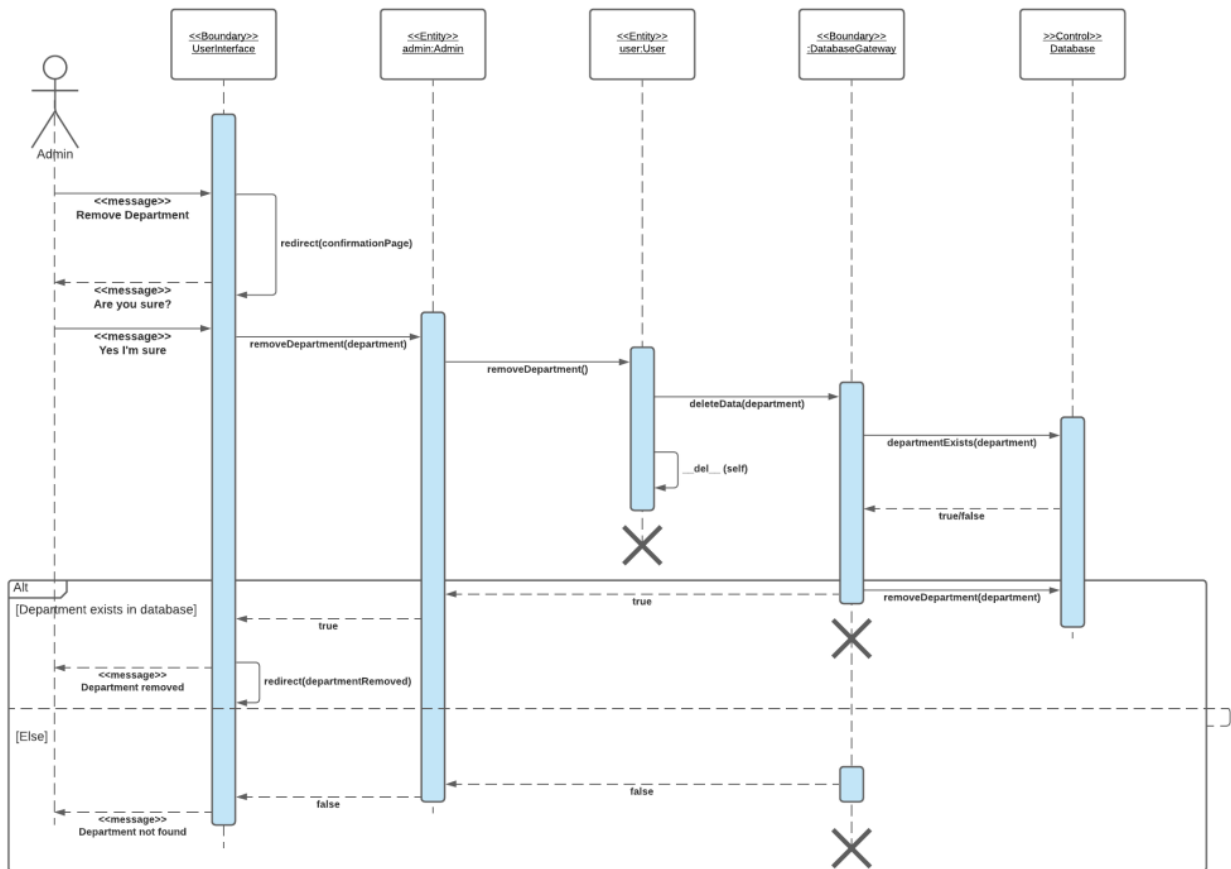
### IT 326

Fundamentals of software engineering. Topics include specification models, metrics, design fundamentals, interface design, quality assurance, and automated tools. Offered alternate years

Figure 33: Elective card to display search results to find an elective

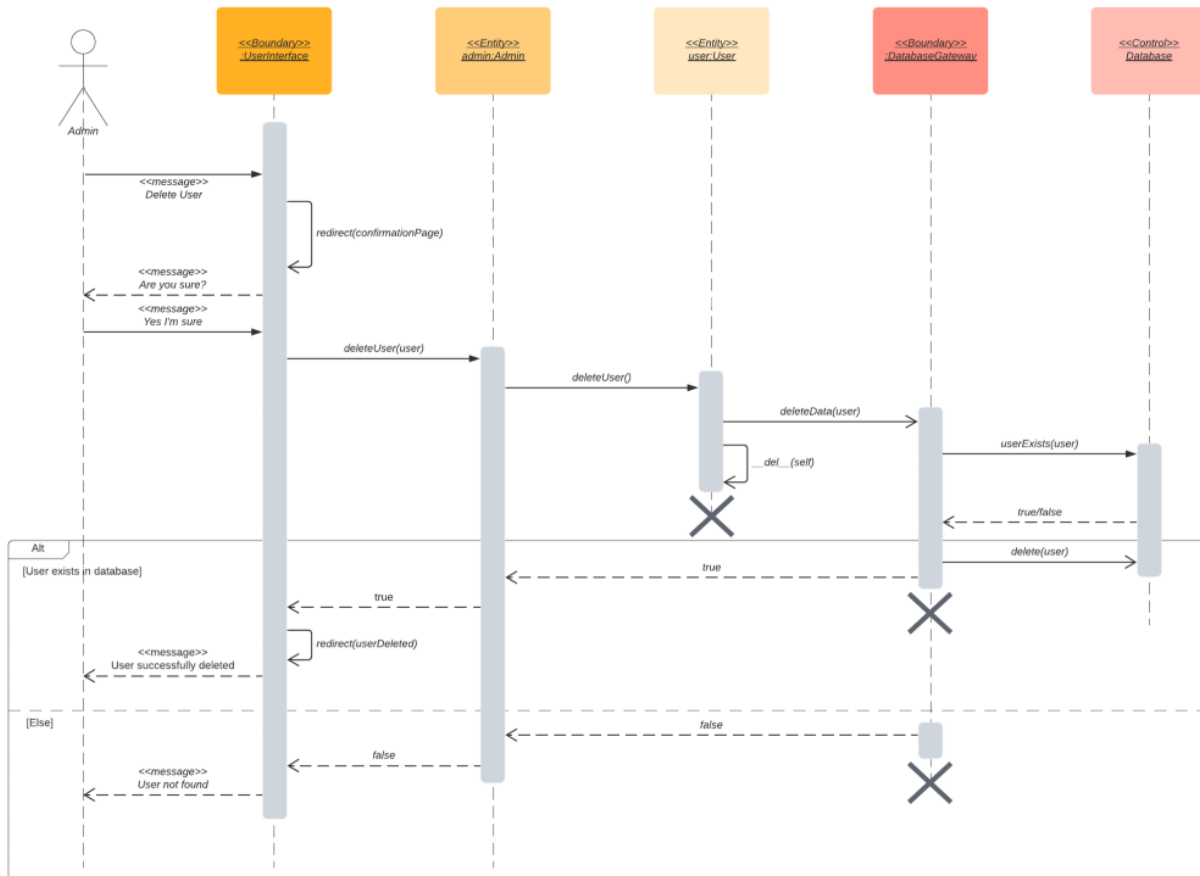
## 4.4.2 Sequence Diagrams

### 4.4.2.1 Remove Department



#### 4.4.2.2 Remove Other User

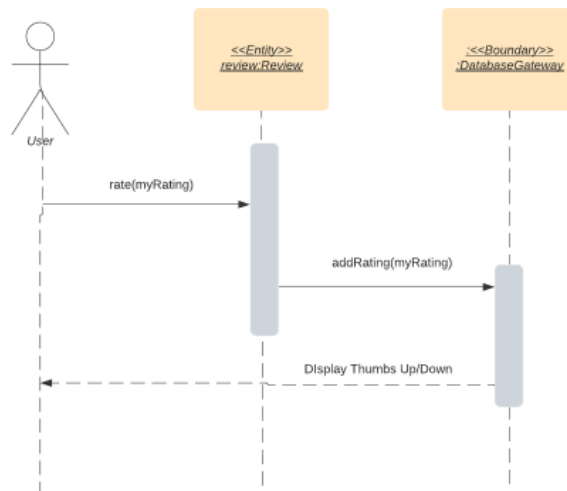
## Rate-My-Elective – Software Requirement Specifications



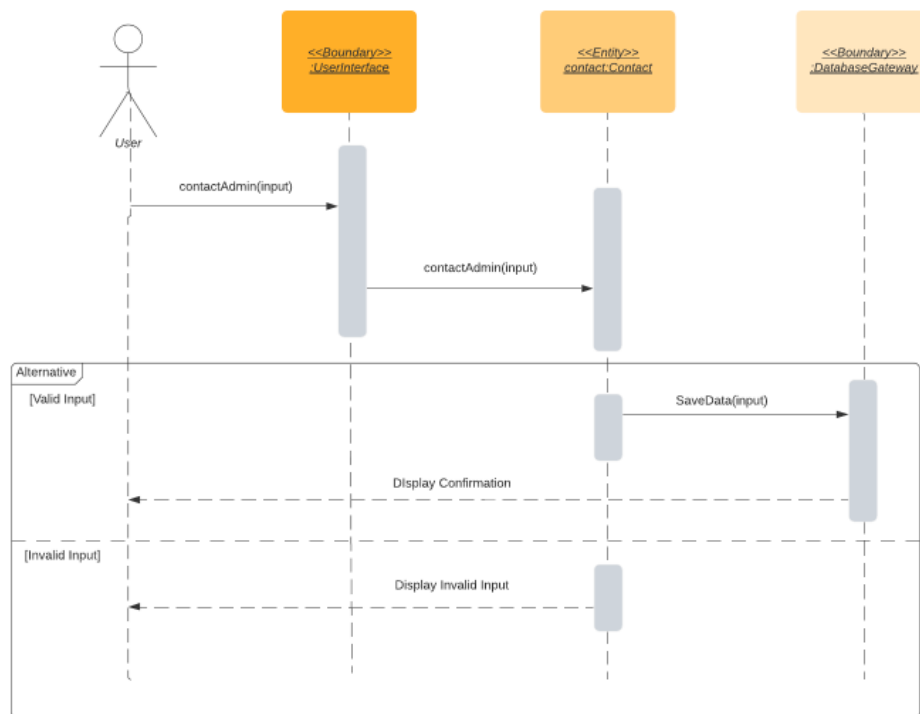
#### 4.4.2.3 Write Review



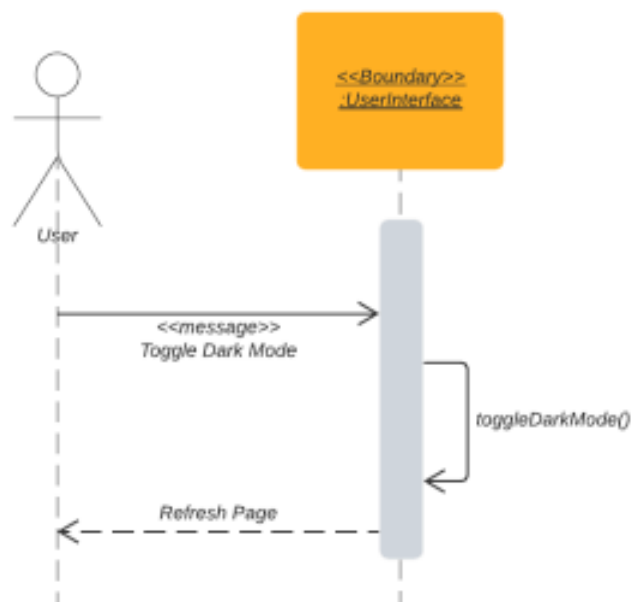
#### 4.4.2.4 Rate Review



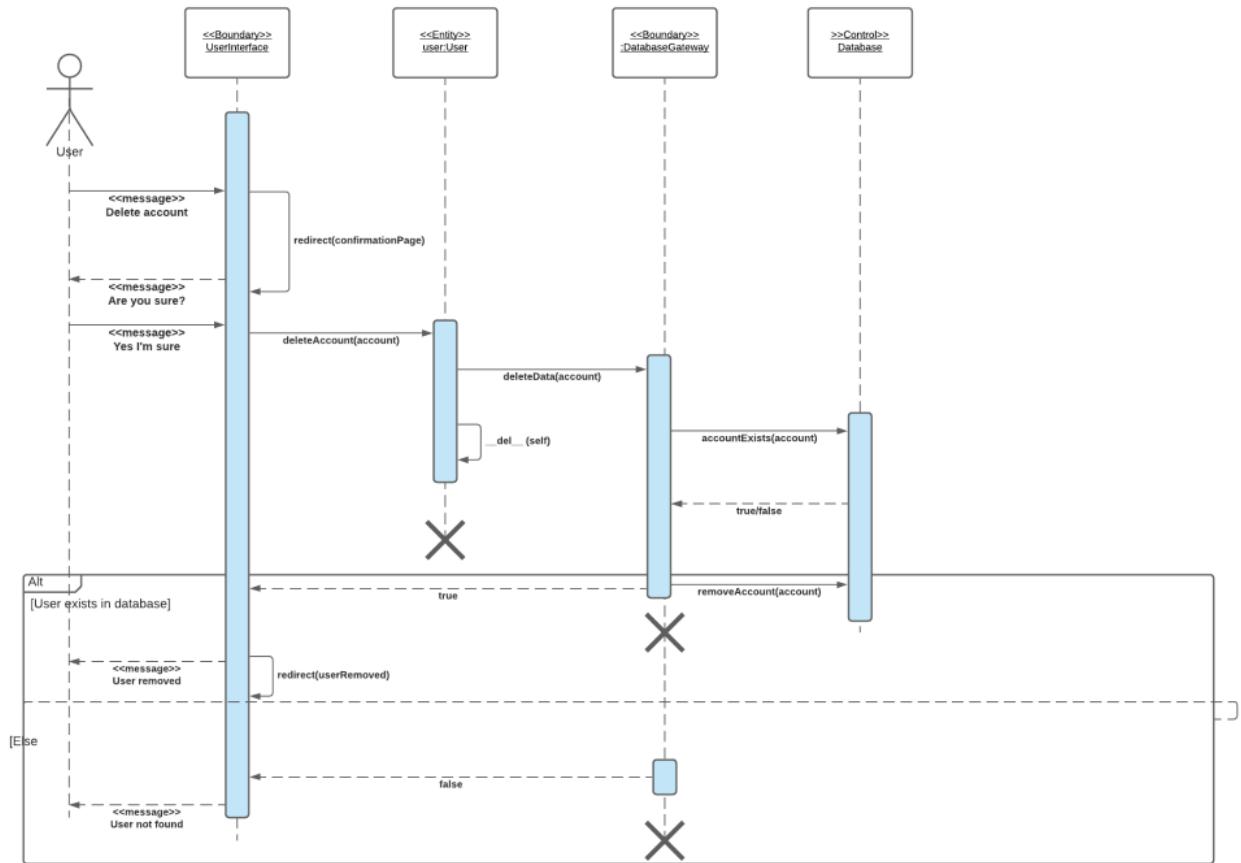
#### 4.4.2.5 Contact Admin



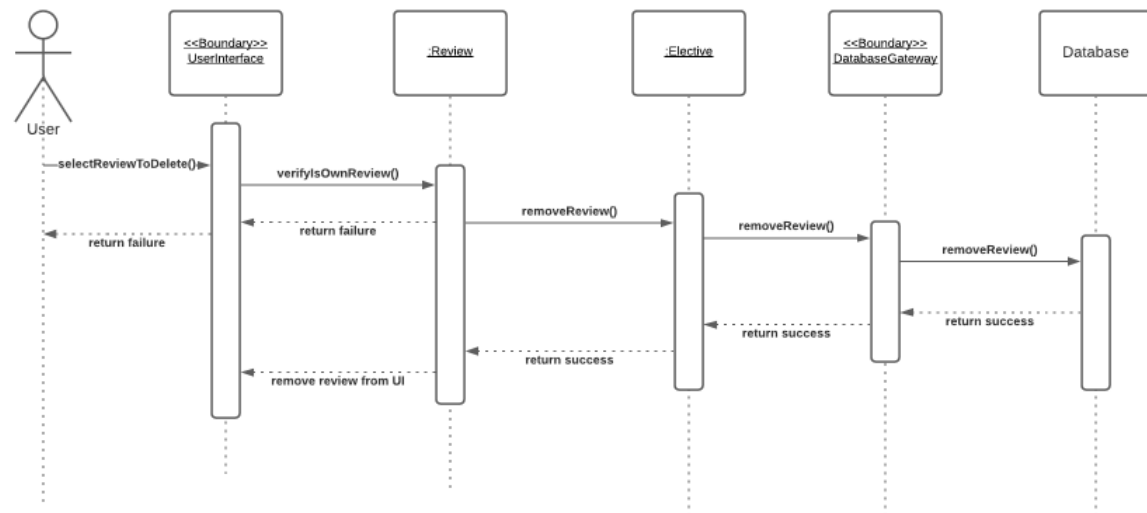
## 4.4.2.6 Toggle Dark Mode



#### 4.4.2.7 Remove Own Account

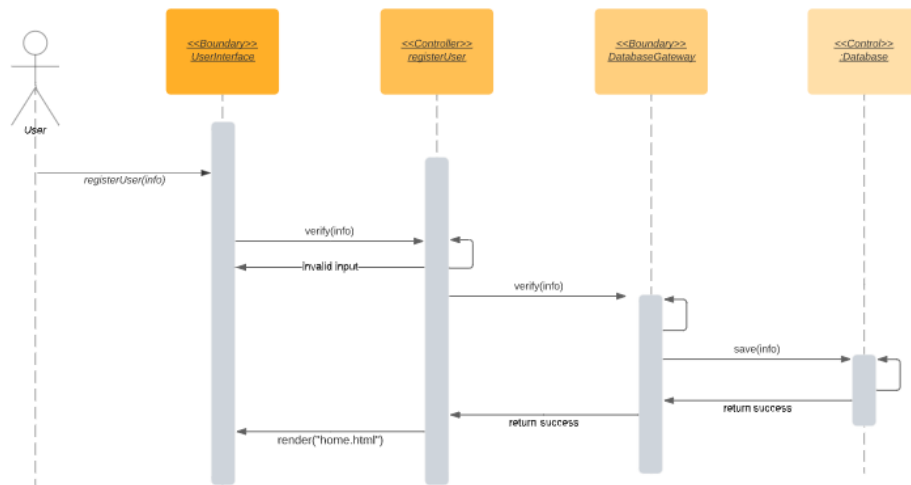


#### 4.4.2.8 Delete Own Review

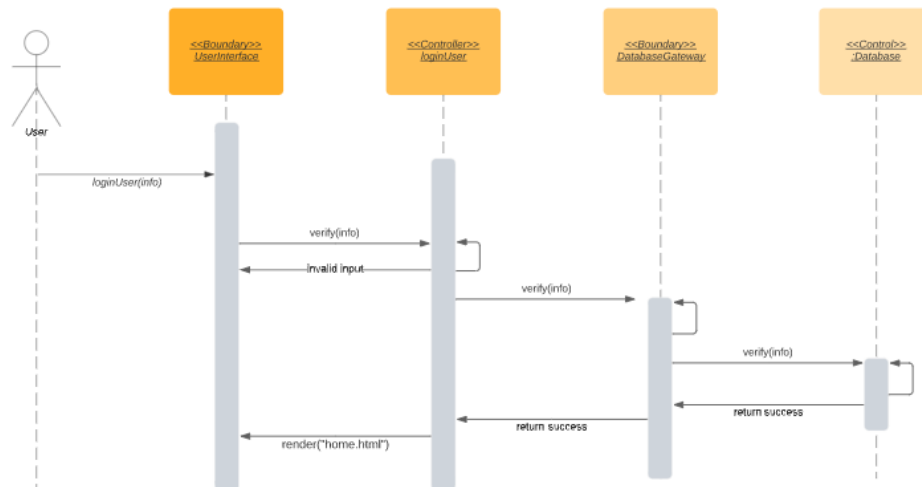


#### 4.4.2.9 Register account





### 4.4.2.10 Login to account



## **4.5 Deliverable 4**

### **4.5.1 Description**

Updated the sequence diagrams seen in Deliverable 3, Section 4.4.

In addition, updated the Class Diagram seen in Deliverable 2, Section 4.3.