

# **Grad Apps 2.0 Design Documentation**

**Edward Vaisman      Sadman Sakib Hasan**

April 28, 2018

## Revisions

Date	Revision	Description
22 November, 2017	1.0	Initial Document Draft
4 December, 2017	1.1	Included UML Class Diagram and Rationale
12 December, 2017	1.2	Updated the Class diagram (Model section Role separation), Added Entity-Model Diagram, Added Data descriptions
14 December, 2017	1.3	Updated Class diagram to match the new model, Updated Entity-Model Diagram, Edited Data descriptions
16 December, 2017	1.4	Added Interface Design and Traceability Matrix
17 December, 2017	1.5	Added Sequence Diagrams for the Component Design
19 January, 2018	1.6	Updated Data Dictionary and Model Diagram
28 April, 2018	1.7	Updated Data Dictionary, ER Diagram, and added File Structure. Removed Requirements Matrix & Appendices

## Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	Purpose . . . . .	5
1.2	Scope . . . . .	5
1.3	Overview . . . . .	6
<b>2</b>	<b>System Overview</b>	<b>7</b>
<b>3</b>	<b>System Architecture</b>	<b>8</b>
3.1	Architectural Design . . . . .	8
3.2	File Structure . . . . .	9
3.3	Decomposition Description . . . . .	12
3.4	Design Rationale . . . . .	12
3.4.1	MVC . . . . .	12
3.4.2	Roles . . . . .	12
<b>4</b>	<b>Data Design</b>	<b>13</b>
4.1	Data Description . . . . .	13
4.2	Data Dictionary . . . . .	14
<b>5</b>	<b>Component Design</b>	<b>16</b>
5.1	Model . . . . .	16
5.2	View . . . . .	16
5.3	Controller . . . . .	16
5.3.1	Sequence Diagrams . . . . .	16
<b>6</b>	<b>Human Interface Design</b>	<b>25</b>
6.1	Overview of User Interface . . . . .	25
6.2	Screen Images . . . . .	25
6.3	Screen Objects and Actions . . . . .	32

## List of Figures

1	UML Class Diagram . . . . .	8
2	Entity Relation Diagram . . . . .	13
3	Data Dictionary Part.1 . . . . .	14
4	Data Dictionary Part.2 . . . . .	15
5	Sequence Diagram: Login to the system . . . . .	17

---

6	Sequence Diagram: Role Selection . . . . .	18
7	Sequence Diagram: Upload an Application . . . . .	19
8	Sequence Diagram: Update an Application . . . . .	20
9	Sequence Diagram: Assigning a Role . . . . .	20
10	Sequence Diagram: Applying filter . . . . .	21
11	Sequence Diagram: Notification . . . . .	22
12	Sequence Diagram: Review an Application . . . . .	23
13	Sequence Diagram: Applying University Assessment . . . . .	24
14	Login Page View . . . . .	25
15	Role Selection View . . . . .	26
16	Admin View - Committee Member Status . . . . .	27
17	Admin View - List Faculty Members . . . . .	27
18	Admin and Professor View - List of Student Applications . . . . .	28
19	Filters available for the Admin and Professor role . . . . .	29
20	Committee View - List of review applications . . . . .	30
21	Filters available for the Committee role . . . . .	31

# 1 Introduction

## 1.1 Purpose

This software design document is intended to provide information regarding the system and architecture on the design of the new graduate application system for the EECS graduate program.

## 1.2 Scope

Gradapps v2.0, a graduate application system, is intended to assist the EECS graduate program better locate and maintain potential graduate student applications from the moment it is submitted until the moment a decision is made. It is intended to replace the current system in use for the EECS graduate program. It will be accessible for use as an online portal on the web.

Gradapps v2.0 promises the following:

- *Administrators* will be able to manage *Committee Members*, assign applications to *Committee Members* to review, edit/manage student applications, grant/revoke privileges from faculty members and add/remove members from the system.
- *Committee Members* will be able to view and manage a list of student applications assigned to them and be able to save applications as a draft for future edits.
- *Professors* will be able to search for applications and indicate their interest in a student to other professors and for personal use indicate whether or not an application has been reviewed.
- All roles will be able to apply advanced filtering methods relative to their given roles to help search for applications.

## 1.3 Overview

This document is divided into 9 sections,

- Introduction
- System Overview
- System Architecture
- Data Design
- Component Design
- Human Interface Design

## 2 System Overview

Having a postgraduate degree on one's resume is always appealing to their future employer, let it be working in the industry or continuing studies to achieve a doctoral status. When applying for a postgraduate program, applicants spend a lot of their time gathering different levels of information (transcripts, letter of recommendation, resume etc.) required for admission. Once an application has been submitted, the graduate program analyses the information to find the best candidate for each program.

Analysing that level of dense information can be challenging at times and to avoid loss of any information, it is best practice to automate this process as much as possible. In order to achieve that goal, a *concise* and *simple* Business System is required that will reduce the manual work.

The graduate program in Electrical Engineering and Computer Science (EECS) is facing a very similar situation. The current system our client has involves a lot of manual work. The centre of this process are the Graduate Program Director (GPD) and Graduate Program Assistant (GPA) who plays a major role in all applications regardless of the applicant being admitted or rejected.

*Administrators* have the task of managing *Committee Members*, acquiring documents from the admissions office for upload, assigning applications to *Committee Members* for review, assigning roles to faculty members, uploading applications for faculty members to view, and lastly manage student applications.

*Committee Members* have the task of reviewing an application provided by the Administrator(s) and send it back for the *Administrators* to upload once all reviews are in for Faculty Members to see.

*Professors* then have the ability to search for applications that best suit their criteria for a good graduate student.

Given the current restrictions with York admissions office some things such as accessing different levels of information must be done manually by the *GPA* other types of accessing information such as, looking up a university description could be done automatically. Overall, our client requires a more robust and concise system that will enable them to automate the selection of the best candidate into the program minimizing the manual work.

## 3 System Architecture

### 3.1 Architectural Design

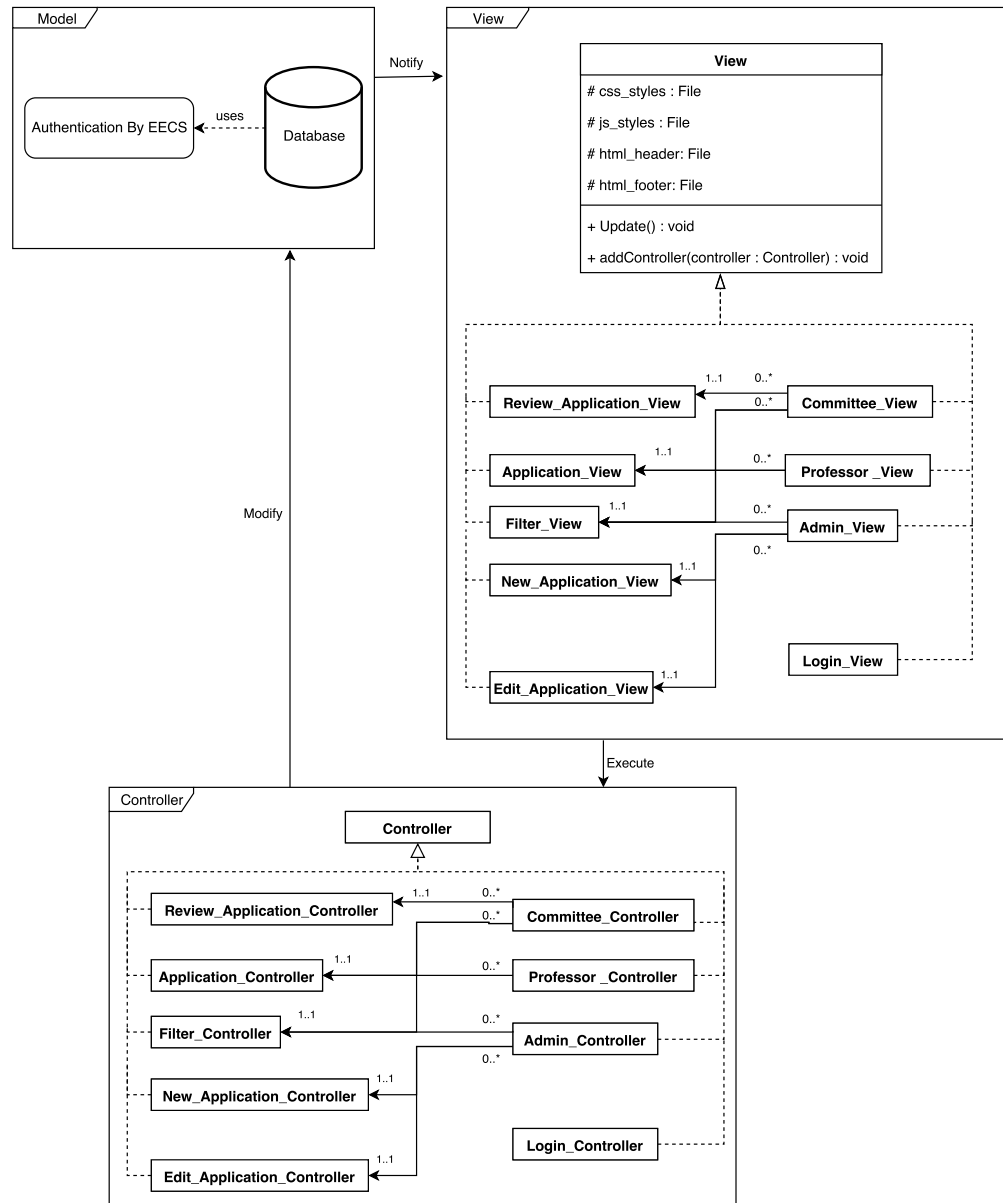


Figure 1: UML Class Diagram



## 3.2 File Structure

The next page contains a condensed version of the current file structure of our application. The system is currently run on Node.js and uses the app.js file to start the application. Authentication is done through passport.js.

To view all *Controller* related the main folders of focus will be the *routes* and *controller* folder. The routes handles the management of users that traverse to the website and separates what the user can and cannot do based on their role. Each route makes use of the files in the controller that will directly interact with our model. This is all done on the server.

To view all *Model* related files you will need to see the mysql database in addition to the .private folder. the .private folder contains a .htpasswd file of all our users in the system with the files in the model folder to handle the manipulation of the file.

Lastly, the *View* related files can be seen in the views folder. All the files contain a html template that will change according to the data received by the model and will provide forms and buttons for the user to communicate with the controller via GET and POST requests. It's stylings can be found in the public folder which contains any client sided javascript, css, images, fonts, and documents.

```

1 src :
2 |
3 +---+ .private
4 +---+ .config
5 |
6 |       database.js
7 |       db-type.js
8 |       passport.js
9 +---+ .controller
10 |       application.js
11 |       fm.js
12 |       review.js
13 |       utils.js
14 +---+ .model
15 |       user.js
16 +---+ .public
17 |       +---+ docs
18 |       +---+ fonts
19 |       +---+ image
20 |       +---+ js
21 |       \---+ stylesheets
22 +---+ .routes
23 |       admin.js
24 |       auth.js
25 |       committee.js
26 |       index.js
27 |       professor.js
28 |       roles.js
29 |       routes.js
30 |       settings.js
31 |       view-app.js
32 |       \---+ admin
33 |       |       manage-app.js
34 |       |       manage-review.js
35 |       |       manage-user.js
36 |       |       view-review.js
37 |       +---+ manage-app
38 |       |       edit-app.js
39 |       |       new-app.js
40 |       \---+ manage-user
41 |       |       edit-user.js
42 |       |       new-user.js
43 |       \---+ views
44 |       |       admin.ejs
45 |       |       committee.ejs
46 |       |       edit-app.ejs
47 |       |       edit-user.ejs
48 |       |       error.ejs
49 |       |       index.ejs
50 |       |       login.ejs
51 |       |       manage-app.ejs
52 |       |       manage-review.ejs
53 |       |       manage-user.ejs
54 |       |       new-app.ejs
55 |       |       new-user.ejs
56 |       |       professor.ejs
57 |       |       review.ejs
58 |       |       roles.ejs
59 |       |       settings.ejs
60 |       |       view-review.ejs
61 |       \---+ partials
62 |       |       filter-application.ejs
63 |       |       filter-committee.ejs
64 |       |       filter-review.ejs
65 |       |       footer.ejs
66 |       |       header.ejs
67 |       |       navbar.ejs

```

Listing 1: File Structure

### 3.3 Decomposition Description

The System Architecture was designed with the MVC Pattern in mind.

- The *Model* represents the database and acts as a data handler for all the data to be stored.
- The *View* represents the front-end web application using web technologies such as html, css and javascript.
- The *Controller* represents the back-end system that triggers the model and updates the view.

### 3.4 Design Rationale

#### 3.4.1 MVC

Given that this is a web application designing it using the MVC pattern makes the most sense here for the following reason:

- Simultaneous development
- Ease of modification
- Multiple Views for our one model

As a result of MVC, developers will be able to work on parts of the system concurrently as opposed to having to wait for others to finish their part and the ability to add or remove a feature later on will be easier and the overall system will be modular.

#### 3.4.2 Roles

To satisfy the specification requirements of faculty members being able to have more than one role, there is no inheritance/generalization between faculty members and all the other roles. This design choice allows for an easy separation of information across roles so that changes in one does not affect changes in another.

An alternative to this design would have been to split the roles into their own classes. This choice is not beneficial for our system since all of our roles share common things such as id, name, email etc and would create a lot of duplication in the system. See Section 4 for diagram and data description.

## 4 Data Design

### 4.1 Data Description

The following diagram illustrates the UML Entity Relationship. The design of each entity will be represented in the database during the implementation phase.

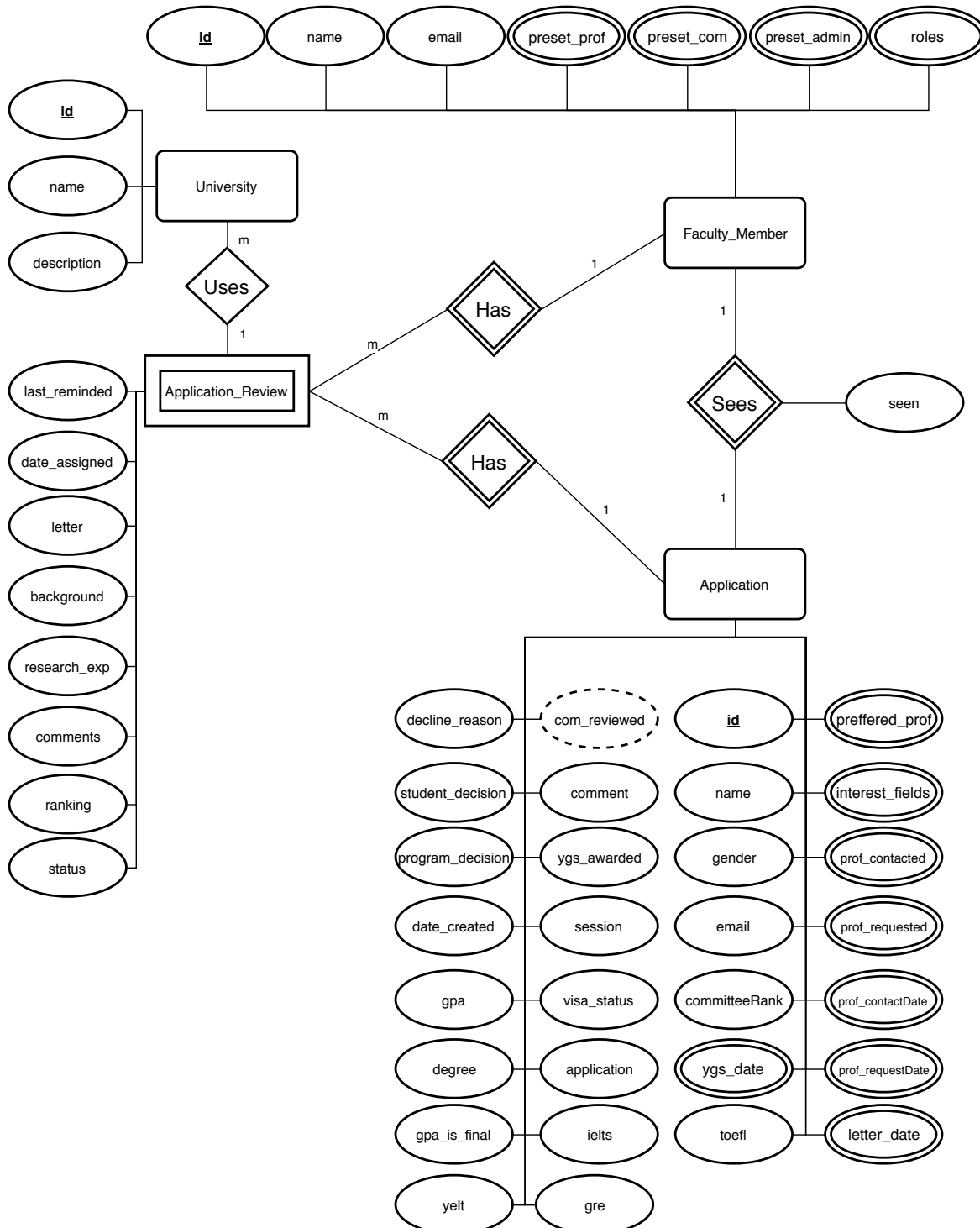


Figure 2: Entity Relation Diagram

## 4.2 Data Dictionary

Table Name	Field	Type	Null	Default
University	id	PK int	no	
	name	varchar(xx)	no	
	description	array of varchar(xx)	yes	NULL
Application_Review	com_id	FK, PK int	no	
	app_id	FK, PK int	no	
	letter	varchar(xx)	yes	NULL
	background	varchar(xx)	yes	NULL
	research_exp	varchar(xx)	yes	NULL
	comments	varchar(xx)	yes	NULL
	ranking	enum('A+', 'A', 'B+', 'B', 'C+', 'C')	yes	NULL
	status	enum('New', 'Draft', 'Submitted')	no	FALSE
	last_reminded	Date	no	current time
	date_assigned	Date	no	current time
Application	id	PK int	no	
	session	varchar(xx)	no	
	student_id	bigint(9)	no	
	name	varchar(xx)	no	
	gender	enum('male', 'female')	yes	NULL
	email	varchar(xx)	no	
	gpa_final	bool	no	FALSE
	gre	varchar(xx)	yes	NULL
	toefl	int	yes	NULL
	ielts	float	yes	NULL
	yelt	int	yes	NULL
	gpa	enum('A+', 'A', 'B+', 'B', 'C+', 'C', 'D+', 'D', 'E', 'F', 'NC')	yes	NULL
	committeeRank	enum('A+', 'A', 'B+', 'B', 'C+', 'C')	yes	NULL
	degree	enum('PHD', 'Msc', 'MASC')	yes	NULL
	visa_status	enum('Domestic', 'Visa')	no	
	application	File	no	
	session	enum('Fall', 'Winter', 'Summer')	no	
	interest_fields	array of varchar(xx)	yes	NULL
	preferred_prof	array of varchar(xx)	yes	NULL
	date_created	Date	no	current time
	prof_contacted	array of varchar(xx)	yes	NULL
	prof_requested	array of varchar(xx)	yes	NULL
	prof_contactDate	array of Date	yes	NULL
	prof_requestDate	array of Date	yes	NULL
	letter_date	array of Date	yes	NULL
	ygs_date	array of Date	yes	NULL
	program_decision	varchar(xx)	yes	NULL
	student_decision	varchar(xx)	yes	NULL
	decline_reason	varchar(xx)	yes	NULL
	ygs_awarded	bool	no	FALSE
	comments	varchar(xx)	yes	NULL
	com_reviewed	bool	no	FALSE

Figure 3: Data Dictionary Part.1

The below figure shows two tables *Fields\_of\_Interest* and *GPA* that are not displayed in our ER-diagram. These two tables are beneficial for the internal logic of our system such as being able to properly sort GPA and to have a more flexibility for things we expect to change throughout the years like fields of interest.

Table Name	Field	Type	Null	Default
Application_Seen	<b>prof_id</b>	FK, PK int	no	
	<b>app_id</b>	FK, PK int	no	
	seen	bool	no	FALSE
Faculty_Member	<b>id</b>	PK int	no	
	username	unique varchar(50)	no	
	name	varchar(100)	no	
	email	varchar(255)	yes	NULL
	preset_prof	array of varchar(xx)	yes	NULL
	preset_com	array of varchar(xx)	yes	NULL
	preset_admin	array of varchar(xx)	yes	NULL
	roles	array of enum('Admin','Committee','Professor')	yes	NULL
Fields_of_Interest	<b>id</b>	PK int	no	
	<b>name</b>	PK varchar(50)	no	
GPA	<b>grade</b>	PK enum('A+','A','B+','B','C+','C','D+','D','E','F','NC')	no	
	grade_point	int	no	

Figure 4: Data Dictionary Part.2

## 5 Component Design

### 5.1 Model

The model acts as a data storage handler for the GradApps business system. Most of which is already discussed in Section 4. The database will use MySQL version: 5.7.20 and Authentication By EECS as per the requirements elicited from the client. The design chosen was in favor of simplicity instead of efficiency. This decision was based on the relatively low amount of student applications gradapps will store (roughly 1000 applications per year) thus eliminating the need for faster query calls.

### 5.2 View

Some views will be able to display other views depending on the role selected.

Here are a few:

- Admin View includes the Filter View, New Application View, Edit Application View, and the Application View.
- Committee View includes the Filter View, and the Review Application View.
- Professor View includes the Application View, and the Filter View.

Based on what the main view is (Admin, Committee, Professor) the filter view will adjust accordingly to allow for their respective controllers.

### 5.3 Controller

Since some roles have different privileges than others, it is important to restrict what one user could do based on the view that they are in. For example, if someone is in the professor view they cannot access methods from the admin controller.

#### 5.3.1 Sequence Diagrams

The following sequence diagrams describe the communication between our actors and the system under description. In short our system behaves as follows.

- Actor acts on the system
- Client side validation
- Appropriate updates to database

- Return updated results and error message if necessary.

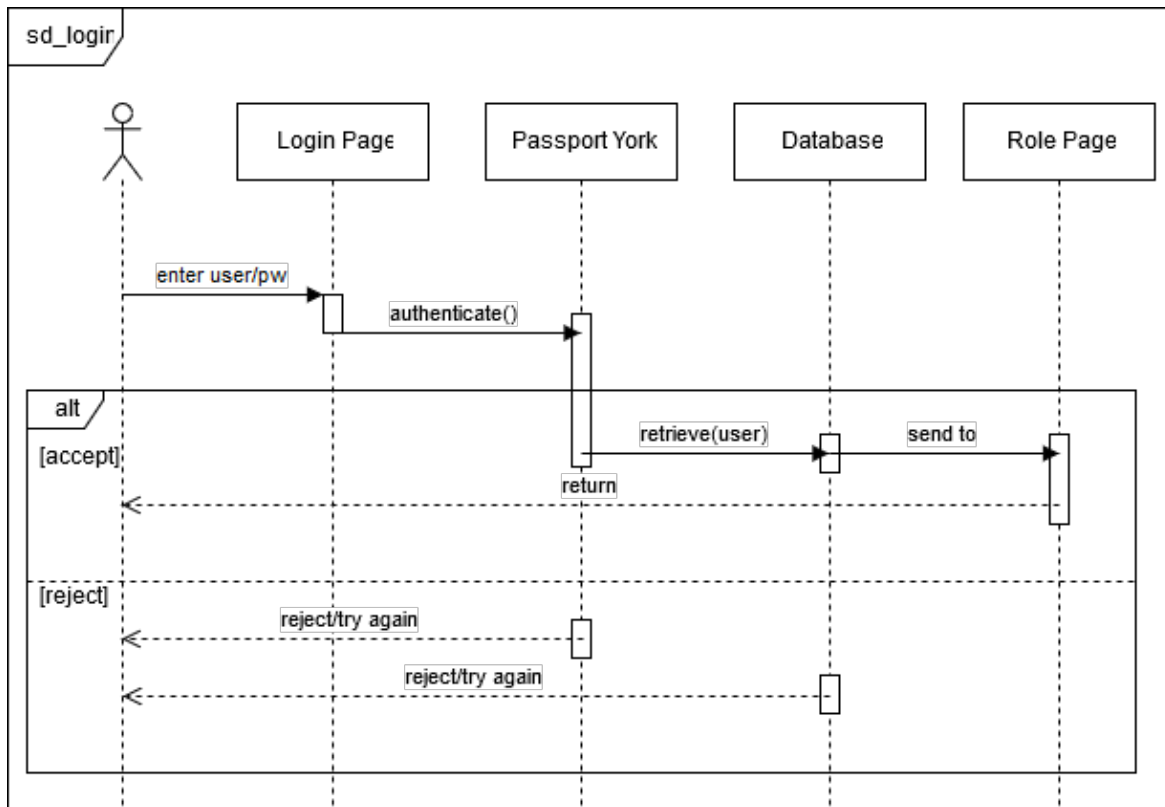


Figure 5: Sequence Diagram: Login to the system



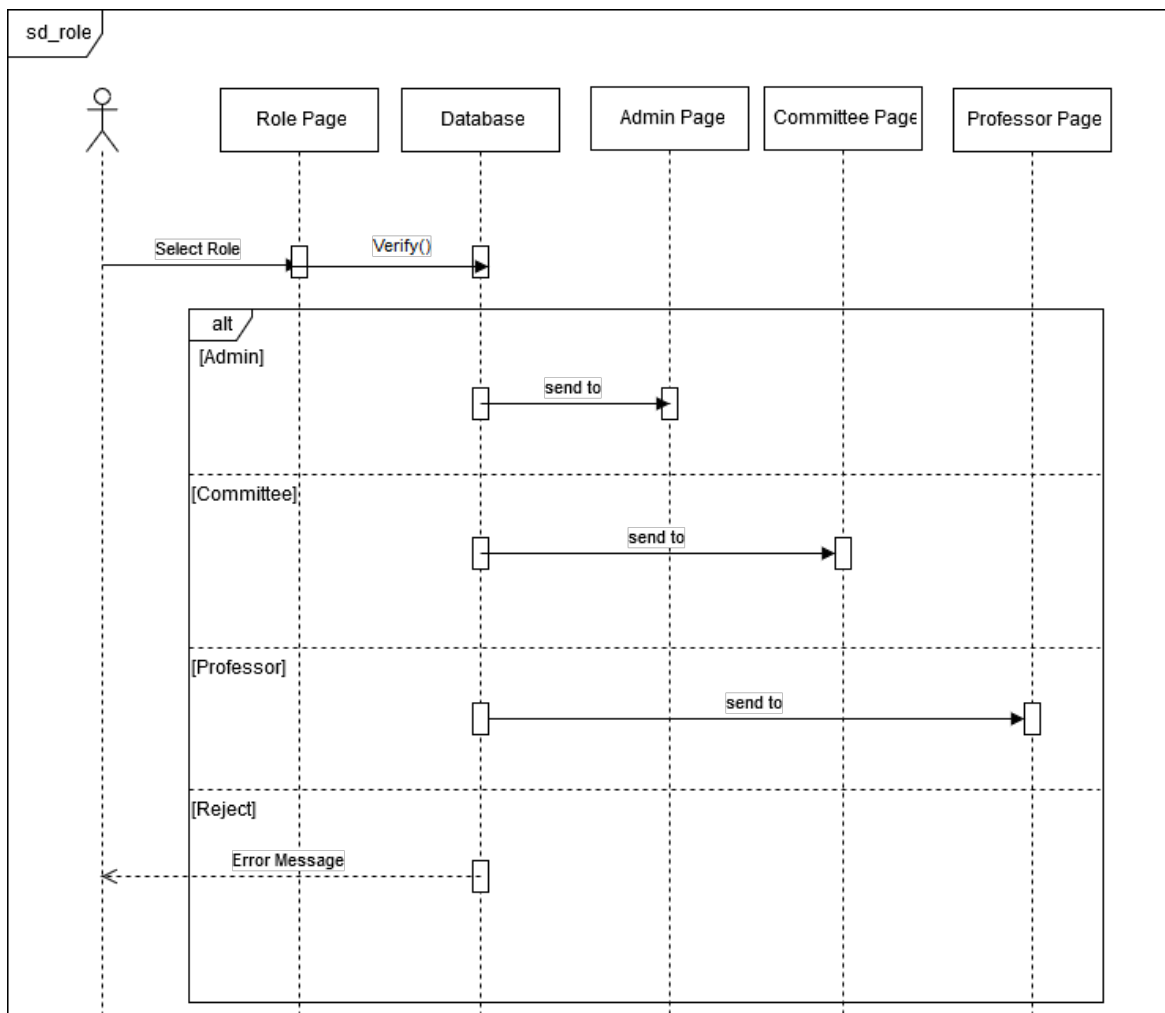


Figure 6: Sequence Diagram: Role Selection

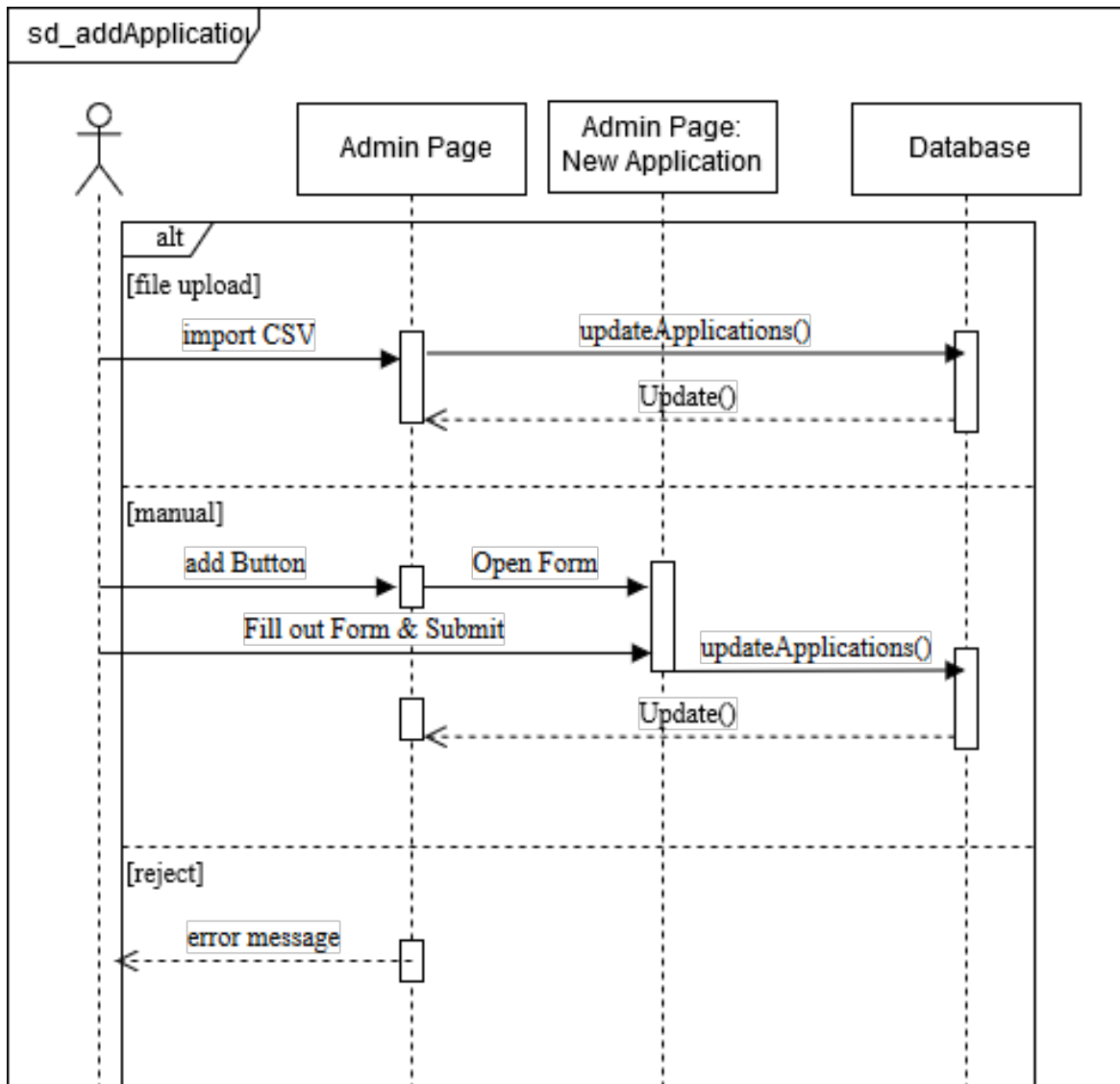


Figure 7: Sequence Diagram: Upload an Application

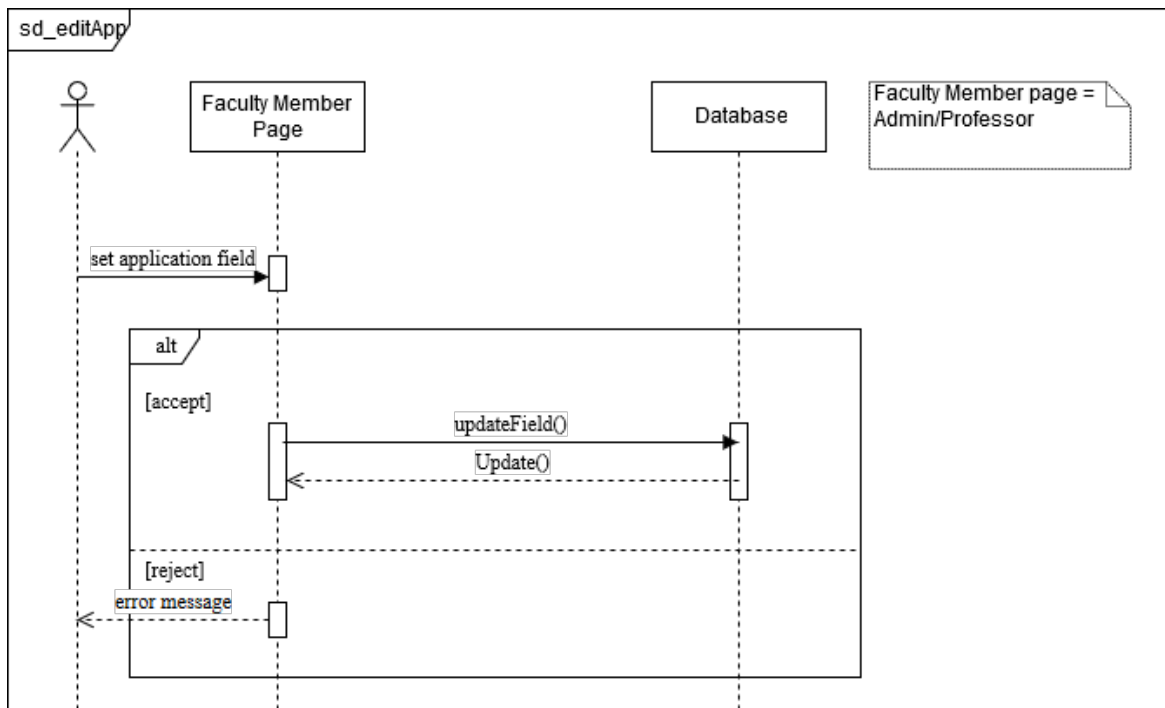


Figure 8: Sequence Diagram: Update an Application

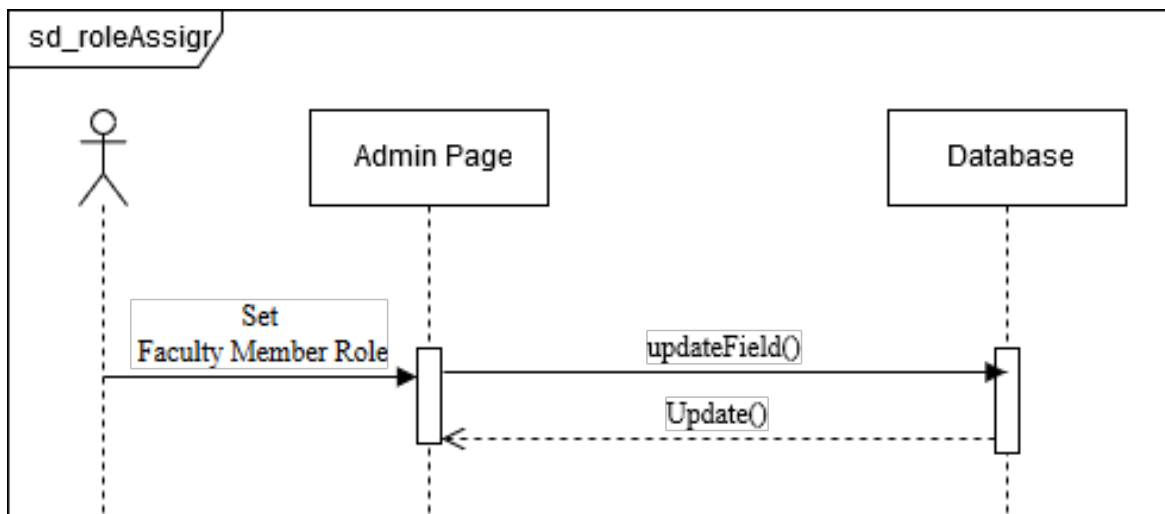


Figure 9: Sequence Diagram: Assigning a Role

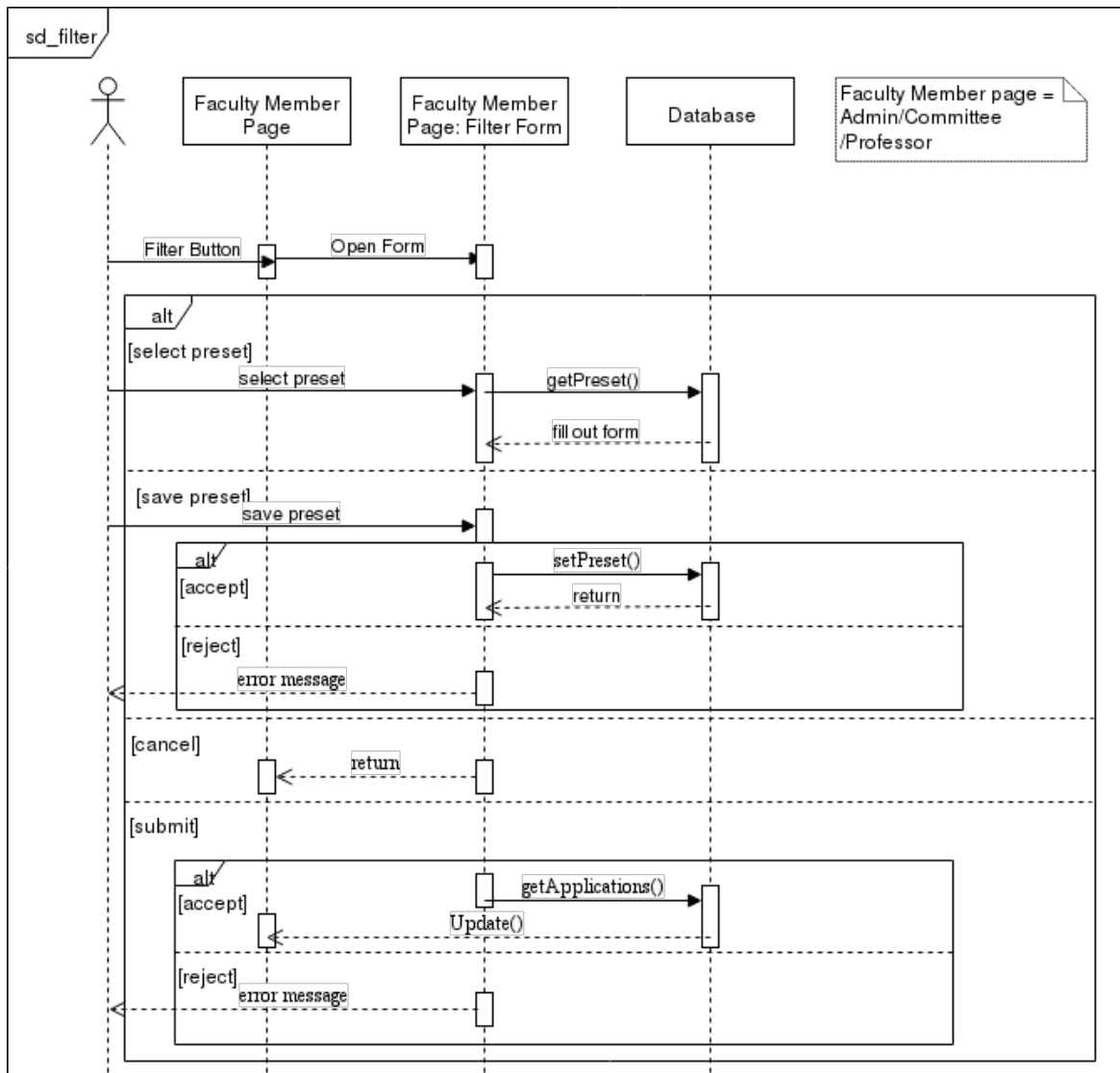


Figure 10: Sequence Diagram: Applying filter

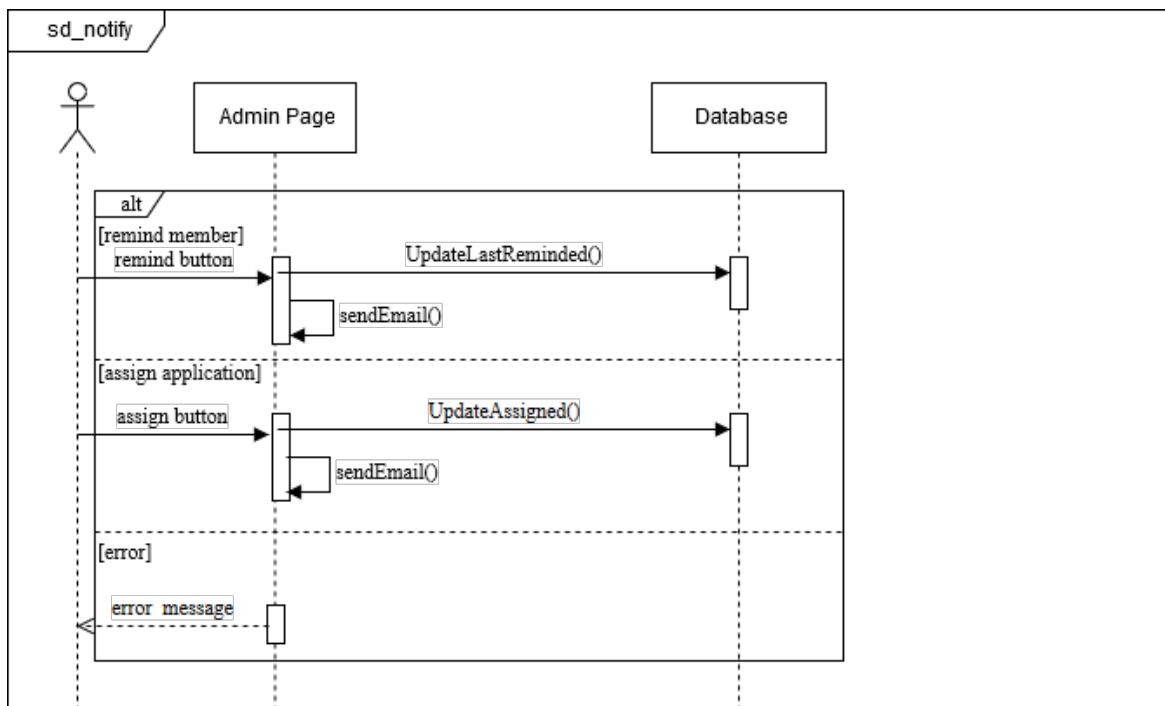


Figure 11: Sequence Diagram: Notification

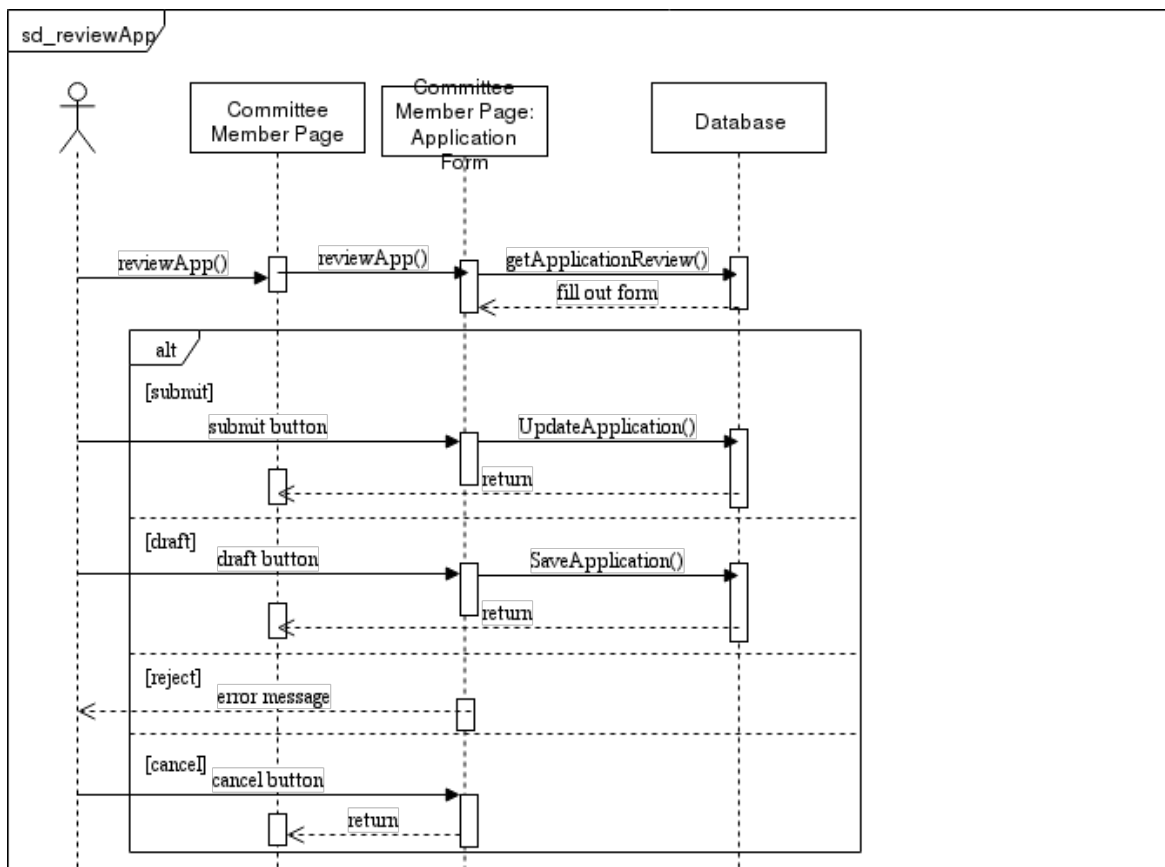


Figure 12: Sequence Diagram: Review an Application

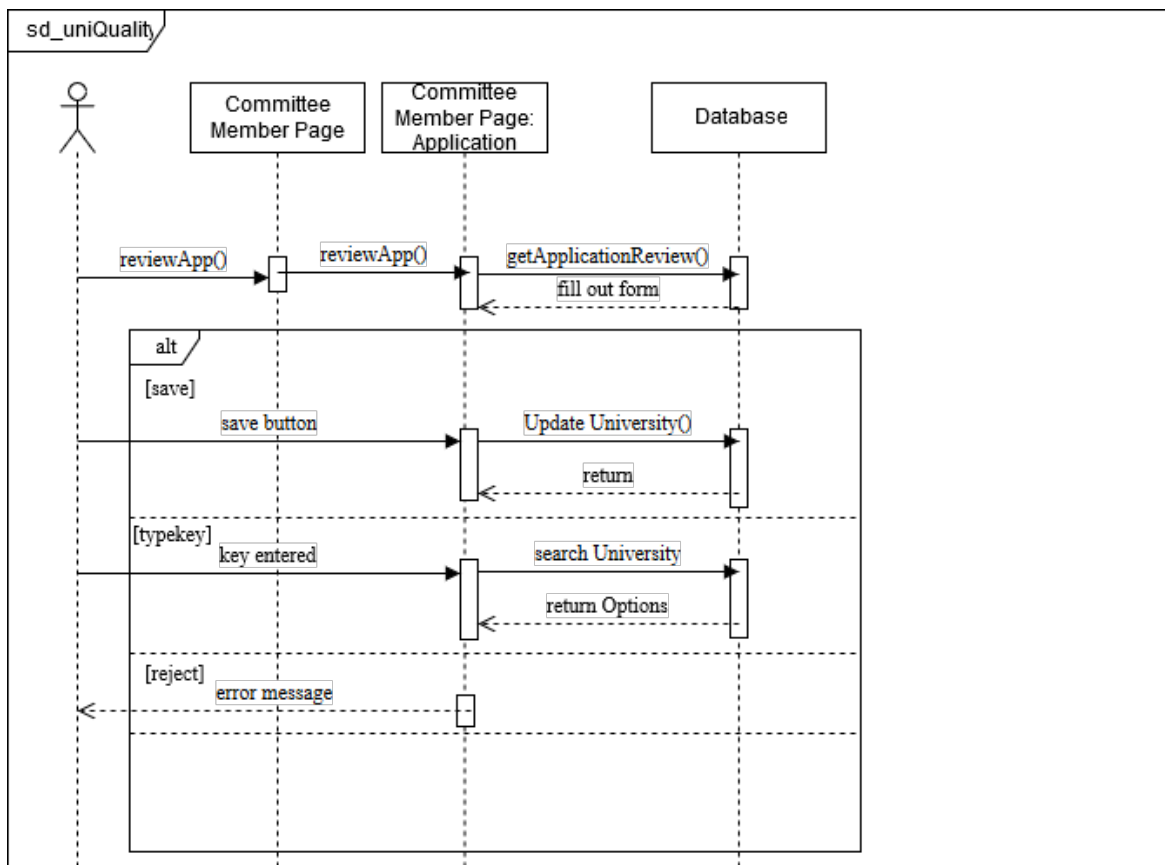


Figure 13: Sequence Diagram: Applying University Assessment

## 6 Human Interface Design

### 6.1 Overview of User Interface

Refer to <http://cambridge.eecs.yorku.ca:3000/>

### 6.2 Screen Images

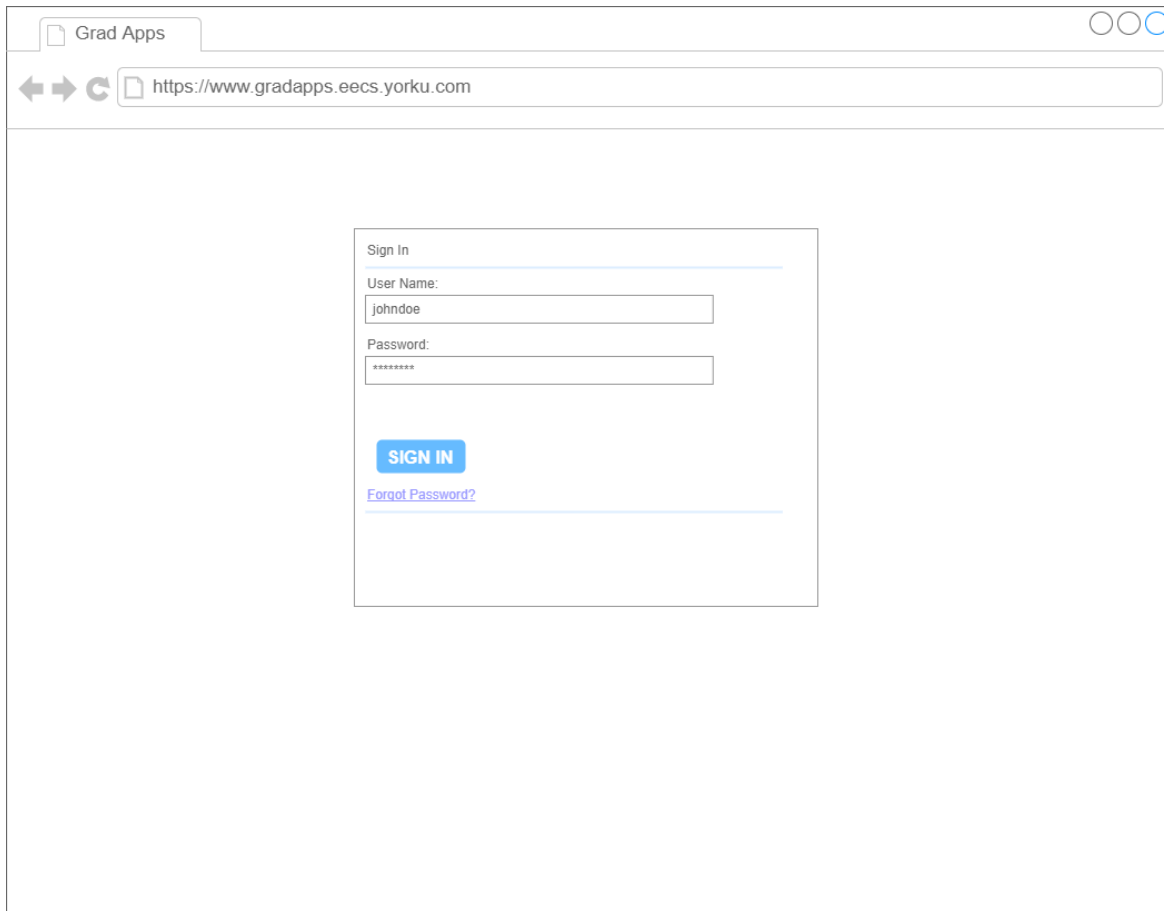


Figure 14: Login Page View



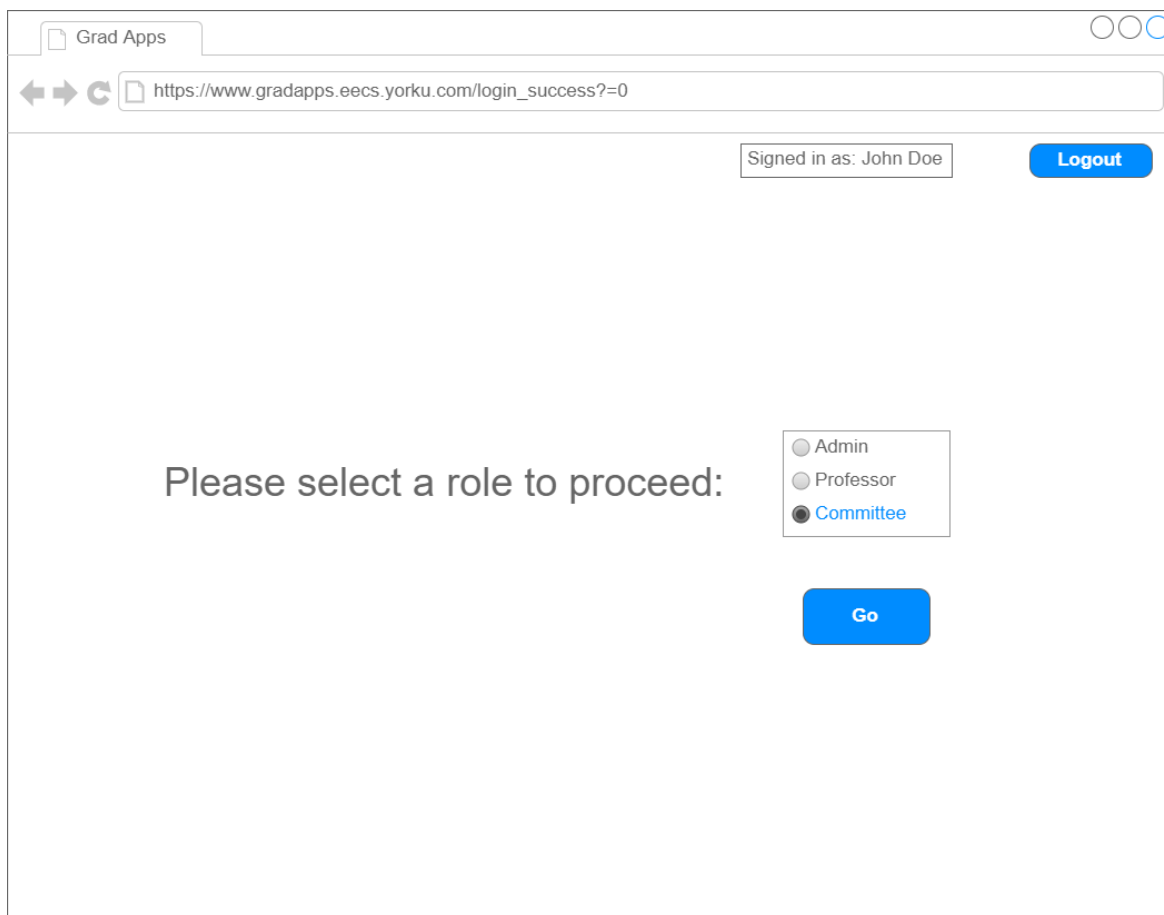


Figure 15: Role Selection View




<div>  Signed in as: Professor Awesome <div>Filter </div> <div>Role: Admin </div> <div>Sign Out</div> </div>			
Committee Member Status	Faculty Members	Applications	
Name	Total Applications	# of Applications to Review	Actions
Michael R. M. Jenkin	10	6	<button>Remind</button>
Franck van Breugel	5	2	<button>Remind</button>
Andrew Eckford	3	1	<button>Remind</button>
Jonathan S. Ostroff	7	3	<button>Remind</button>
Melanie Baljko	5	2	<button>Remind</button>
Matthew Kyan	6	0	<button>Remind</button>

Figure 16: Admin View - Committee Member Status






<div>  Signed in as: Professor Awesome <div>Filter </div> <div>Role: Admin </div> <div>Sign Out</div> </div>		
Committee Member Status	Faculty Members	Applications
Name	Role(s)	Action
Michael R. M. Jenkin	Professor	<button>Edit Role</button>
Franck van Breugel	Admin, Professor, Committee	<button>Edit Role</button>
Andrew Eckford	Professor	<button>Edit Role</button>
Jonathan S. Ostroff	Professor	<button>Edit Role</button>
Melanie Baljko	Committee	<button>Edit Role</button>
Matthew Kyan	Professor	<button>Edit Role</button>
Ourma	Admin	<button>Edit Role</button>

Figure 17: Admin View - List Faculty Members

EECS  
ELECTRICAL ENGINEERING  
AND COMPUTER SCIENCE

Signed in as: Professor Awesome

Filter 

Role: Admin 

Sign Out

Committee Member Status

Faculty Members

Applications

# of applications with no reviewers: 3

Create New Application


Field of Interest	Committee Ranking	Degree Applied For	Visa Status 	Application Status	Actions
AI	A+	PHD (CS)	CAD	Contacted by Michael R. M. Jenkin	<div>View Application</div> <div>Set to Contacted</div> <div>Set to Reviewed</div>
Theory of Computation, Networks, AI	A+	MSc	CAD	Accepted	<div>View Application</div> <div>Set to Contacted</div> <div>Set to Reviewed</div>
Biomedical Engineering, Software Engineering, VR	C	MASc	CAD	Rejected	<div>View Application</div> <div>Set to Contacted</div> <div>Set to Reviewed</div>
Theory of Computation, Networks	A+	PHD (CS)	VISA	Submitted	<div>View Application</div> <div>Set to Contacted</div> <div>Set to Reviewed</div>

Figure 18: Admin and Professor View - List of Student Applications

## Filter

### Choose Your Columns

GPA

Degree Applied For

Visa Status

Application Status

Reviewed Status

### Selected Columns

Field of Interest ✕

Preferred Professor(s) ✕

Name ✕

Committee Ranking ✕

### Choose Your Filters

Name ▾

Field of Interest ▾

Preferred Professor(s) ▾

Committee Ranking ▾

GPA ▾

Degree Applied For ▾

Visa Status ▾

Application Status ▾

Reviewed Status ▾



### Selected Filter

Visa Status = VISA AND GPA >= C AND Committee Ranking > C

✕ Cancel

✔ Submit

Figure 19: Filters available for the Admin and Professor role

 **EECS** ENGINEERING, ELECTRONICS, AND COMPUTER SCIENCE Signed in as: Professor Awesome Filter  Role: Committee ▾ Sign Out

### Example Tables

Date of application	Student #	Status of application	Actions
31/12/2017	1	Reviewed	<a href="#">View Application</a>
25/10/2017	2	Not Reviewed	<a href="#">Edit Application</a>
20/10/2017	3	Draft Available	<a href="#">Edit Application</a>
03/09/2017	4	Reviewed	<a href="#">View Application</a>
04/04/2017	5	Not Reviewed	<a href="#">Edit Application</a>
1/1/2017	6	Reviewed	<a href="#">View Application</a>

Figure 20: Committee View - List of review applications

# Filter

Choose Your Columns

Preset 1 ▼

Student # Application Status

Selected Columns

Date of application ✕

Choose Your Filters

Date of application - Application Status -

Selected Filter

Field of Interest = AI AND Preferred Professor = Michael R. M. Jenkin  
AND Status of application = Not Reviewed



✕ Cancel  Submit  Save Filter

Figure 21: Filters available for the Committee role

### 6.3 Screen Objects and Actions

The screenshots un the previous section from Figure 14 to 21 describes the flow of action an user may take for the System Under Description. The following describes each figure from the previous section.

- Figure 14 specifies the login view for the system. The login credentials shall be authenticated by Passport York. The user is expected to have a valid Passport York account and be part of the system database in order to gain access to the system.
- Figure 15 specifies the role selection view upon logging into the system. Once the user has been successfully authenticated into the system, the user needs to select a role to proceed further.
- Figure 16 specifies the admin view of checking the committee member status. This view includes reminding a committee member who have been assigned a review and has not completed it yet.
- Figure 17 specifies the admin view that lists all the members in the system. An admin can remove or change a role of a member. An admin can further add a new member into the system.
- Figure 18 specifies the admin and professor view of all student applications in the system.
- Figure 19 specifies the admin and professor view of the filters available on student applications.
- Figure 20 specifies the committee member view of all applications needed for review in the system.
- Figure 21 specifies the committee member view of the filters available on review applications.