Grad Apps 2.0 Requirements Documentation

Edward Vaisman **Sadman Sakib Hasan**

January 10, 2018

Revisions

Date	Revision	Description
9 October, 2017	1.0	• Fix grammars • Look into each research group for field of interest(s)
17 October, 2017	1.1	Fix additional grammars
23 October, 2017	2.0	• Apply new list of field of interest(s) conducted from the questionnaire • Further elicitation from the GPA and GPD
30 November, 2017	3.0	Add complete use case diagrams, textual description for all specs
5 December, 2017	4.0	• Add use case diagrams for each roles • Include additional R descriptions for optional deliverables • Include additional Use Case Textual Descriptions
9 December, 2017	5.0	Add abstract UI grammar and acceptance tests
29 December, 2017	6.0	• Add system invariants • Reorganize E/R descriptions (seperated into usecases) • Fix UI grammar and acceptance tests
2 January, 2017	7.0	• Updated Use Case Textual Descriptions • Re-organized E/R descriptions • Added error list and success list messages • General typo fixes • Added more system goals

Contents

List of Figures

List of Tables

1. System Overview

Having a post graduate 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 and 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 can ease out the manual work.

The situation of the graduate program in Electrical Engineering and Computer Science (EECS) is very similar. The current system our client has involves a lot of manual work. The centre of this process is 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.

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 to be done. The following diagram outlines the institutions faculty member hierarchy. Our client are the members of **The EECS Graduate Program**.

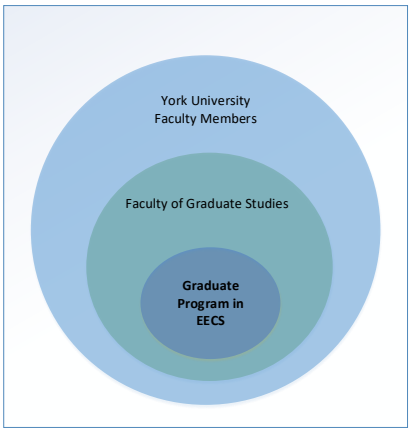

Figure 1: Faculty Members Venn Diagram

2. System Goals

The current system in place for the the EECS Graduate Program completes the bare minimum requirement of displaying student applications to professors. However, faculty members express their frustration about the amount of time that is required to find a suitable graduate student ranging from uploading a student application to requesting a student for admission.

As a result we have set the following high-level goals (**G**) to reduce frustration and improve efficiency:

- **G1**—An Admin should be able to add/remove/modify student applications in the system.
- **G2**—An Admin should be able to assign applications to be reviewed by Committee Members.
- **G3**—An Admin should be able to manage Committee Members.
- **G4**– An Admin should be able to assign/remove roles of other faculty members.
- **G5**—A Committee Member should be able to view applications to be reviewed.
- **G6**—A Committee Member should be able to use/modify/add university descriptions in the system for their reviews.
- **G7**—A Professor and Admin should be able to search for desired student applications by applying a filter (See Appendix B) on application attributes.
- **G8**—A Professor should be able to indicate Admins to request a student for admission.

3. Set Dictionary

The following table denotes a set dictionary describing the mathematical sets defined in the system:

Set	**Notation**	**Description**
$STAFF$		Set of all staffs at York University
FM	$FM \subseteq STAFF$	Set of all faculty members in the EECS graduate program
$APPS$		Set of all student applications in the EECS graduate program
$ADMIN$	$ADMIN \subseteq FM$	Set of all admins
GPA	$(GPA \subseteq STAFF)$ $\wedge$ $(GPA \neq FM)$	Grad Program Assistant in the EECS graduate program
GC	$GC \subseteq FM$	Set of all graduate committee members
$PROF$	$PROF \subseteq FM$	Set of all professors
$UPLOADED$	$UPLOADED \subseteq APPS$	Set of all uploaded applications
$ASSIGNED$	$ASSIGNED \subseteq APPS$	Set of all assigned applications
$REVIEWED$	$REVIEWED \subseteq APPS$	Set of all reviewed applications
TBR	$TBR \subseteq ASSIGNED$	Set of all to be reviewed applications
$IN\_PROGRESS$	$IN\_PROGRESS \subseteq (ASSIGNED \setminus TBR)$	Set of all applications in progress
$DRAFT$	$DRAFT \subseteq IN\_PROGRESS$	Set of all applications saved as draft
$ATTR$	Refer to table: 2 and 3	Set of all application attributes
APP_STATUS	$APP\_STATUS \subseteq (ATTR = "ApplicationStatus")$	Set of all application status
REV_STATUS	$REV\_STATUS \subseteq (ATTR = "ReviewStatus")$	Set of all review status

Table 1: Description table for mathematical sets used in the System Under Description

4. Roles

The **three** major roles in our business system are: *administrator*, *committee member* and *professor*. In other words, a faculty member, fm, such that $fm \in FM$ needs to have at least one role assigned to them in order to access the system. The diagram Fig. 2 depicts the roles along with descriptions in each subsection:

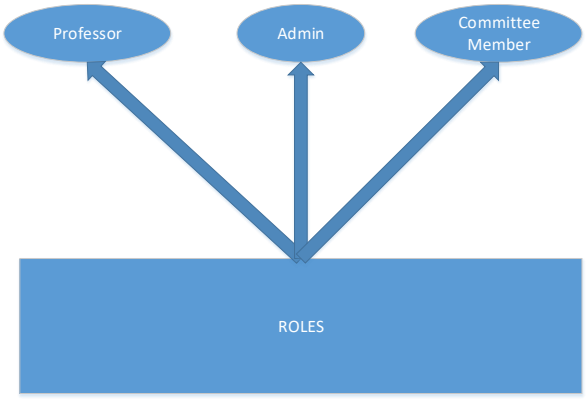

Figure 2: Roles in System Under Description

4.1. Admin

The *admin* in our system under description are the *Graduate Program Director (GPD)* and the *Graduate Program Assistant (GPA)*. The GPA is an **exception** case for an Admin despite not being a member of the graduate program refer to Table 1. The admins have the master control of the overall system starting from adding/removing graduate program members into the system to sending out the final decision for an applicant. A breakdown of an *admin's* permissible actions are listed below.

An *admin*:

- Can add a new member to the list of EECS graduate program members

- Can remove a member from the list of EECS graduate program members except for themselves
- Can assign a new role to an existing member
- Can assign applications to be reviewed by a graduate committee member
- Can send email reminders to committee members about assigned applications
- Can upload a student application to the portal
- Can update all attributes of a student application

4.2. Committee Member

The *committee member* in our system under description is a subset of EECS graduate program members who are in charge of reviewing new applications. The role of a graduate committee member, gcm, such that $gcm \in GC$, is to review the student application(s) that are assigned to them by an *admin*. A breakdown of a *committee member's* permissible actions are listed below.

A *committee member*:

- Can review an assigned application(s)
- Can view past reviewed application(s)
- Can add/use/modify university descriptions in the system to be used for all applications.

4.3. Professor

The *professors* in our system are a subset of EECS graduate program members who are in charge of choosing a student once the final application with reviews have been uploaded to the portal. The role of a professor, $prof$, such that $prof \in PROF$, is to select a student who is interested in their field of studies satisfying the admission requirements. Once a professor is interested in a student, they can contact the student as well as request the student for admission to the *admin*. A breakdown of a *professor's* permissible actions are listed below.

A *professor*:

- Can contact a student, once interested
- Can request a student for admission, once contacted

5. Deliverables

The deliverables for this project upon reaching an agreement with our clients were broken into two parts: **Required** 5.1 and **Optional** 5.2.

The *Required* deliverables are required to have fully functional services and on the other hand, *Optional* deliverables are optional services that could be achieved if and only if all the required deliverables have been completed.

5.1. Required Deliverables

The *Required* deliverables were decided by the majority vote on a list of questionnaires conducted on the EECS graduate program members. Following are the ***Required*** deliverables for the System Under Description:

- Any user shall be able to log into the system if and only if they are a member of the EECS graduate program
- Any user shall be able to log out of the system if and only if they are logged in
- Any user shall be logged out of the system after a maximum of 15 minutes of idleness
- An *admin* shall be able to add a new member to the list of EECS graduate program members
- An *admin* shall be able to remove a member from the list of EECS graduate program members except for themselves
- An *admin* shall be able to assign a new role to an existing faculty member
- An *admin* shall be able to assign applications to be reviewed by committee members
- An *admin* shall be able to upload a student application to the portal
- An *admin* shall be able to update all attributes of a student application
- An *admin* who assigned an application shall be notified when reviews for that application has been completed
- An *admin* shall be able to export the applications to CSV format
- A *committee member* shall be notified when a batch of applications come in for review
- A *committee member* shall be able to see the list of new and previously reviewed application(s)
- A *committee member* shall be able to apply filtering only on previously reviewed application(s)
- A *committee member* shall be able to save applications as a draft for future completion

- A *committee member* shall be able to view, use or modify a university assessment if it has already been used when reviewing applications
- An *admin* and *professor* shall be able to see details about an application in an organized table with the following attributes:
    1. Date of Application
    2. Year
    3. Session
        a) Fall
        b) Winter
        c) Summer
    4. Student ID
    5. Name
    6. Gender
        a) M
        b) F
    7. Field of Interest(s)
        a) Artificial Intelligence
        b) Bioinformatics
        c) Biomedical Engineering
        d) Computational Neuroscience
        e) Computational Biology
        f) Computer Graphics and Media
        g) Computer Security and Networks
        h) Computer Vision
        i) Data Science
        j) Data Mining
        k) Distributed Computing
        l) Embedded Systems
        m) History of Computing
        n) Human-Computer Interaction
        o) Graph Mining
        p) Integrated Circuits and Systems
        q) Large-scale Software Systems

   r) Micro/Nano Electronic Systems
   s) Machine Learning
   t) Performance Engineering
   u) Power and Renewable Energy Systems
   v) Robotics
   w) Signal Processing
   x) Software Engineering
   y) Theory of Computation
8. Preferred Professor(s)
   a) Provide with list of the members of the EECS graduate program
9. Committee Ranking
   a) A+ (definite admit)
   b) A (likely admit)
   c) B+ (possible admit)
   d) B (reserve list)
   e) C (reject)
10. GPA
11. Degree Applied For
   a) MSc
   b) MASc
   c) PhD
12. Visa Status
   a) Visa
   b) Domestic
13. Committee Reviewer
   a) Provide with list of the members of the EECS graduate program who are in the graduate committee
14. Application Status
   a) Contacted by *Professor* - once the student has been contacted by one or more professor
   b) Requested by *Professor* - once the student has been requested by one or more professor
   c) Decision by Program

i. Accepted
ii. Declined
iii. No Decision Made Yet

d) Decision by Student
i. Accepted
ii. Declined
iii. No Decision Made Yet

e) Date letter sent
f) YGS Awarded
g) Reason for Decline
h) Comment

15. Reviewed Status (Similar to application status but is user specific)
    a) Reviewed - once a professor has seen a student application and decided to not move on to contact them
    b) Not Reviewed - not looked into the application yet

- An *admin* and *professor* shall be able to perform advanced filtering on all of the above attributes
- In addition, *Application Status* attribute can only be filtered on:
    1. The date of application uploaded
    2. The professor(s) who has been in contact with the student
    3. The professor(s) who has requested the student
    4. The decision by the program
    5. The decision by the student

Rationale on Field of Interest(s): The initial field of interest(s) had too much overlapping between fields. After conducting a questionnaire on the graduate program members, we came up with a much more distinguished set of field interest(s). Some of the changes from the initial list were:

- Adding important fields that were missing, e.g
    1. Data Mining
    2. Embedded Systems
    3. Machine Learning
    4. Software Engineering
- Combine fields that has the same set of researchers into one, e.g

1. Merge Big Data and Information Systems into just Data Science
2. Merge Adaptive Systems into just Artificial Intelligence

- Separate fields that have too much overlapping into two different fields, e.g
  1. Computer Vision and Robotics were broken down into two fields
     a) Computer Vision
     b) Robotics

Application Attributes

An *admin* or a *professor*, can see a list of student applications once uploaded to the system and a review has been completed. The following table denotes the attributes of a student application:

Attribute	**Type**	**Range**
Date of Application	Date	
Year	Int	0000...9999
Session	Enumerated	$\{Fall, Winter, Summer\}$
Student ID	Big Int	000000000...999999999
Name	String	
Gender	Enumerated	$\{M, F\}$
Field of Interest(s)	Enumerated	Refer to Section 5.1
Preferred Professor(s)	Enumerated	FM
Committee Ranking	Enumerated	$\{A+, A, B+, B, C\}$
GPA	Any	
Degree Applied For	Enumerated	$\{MSc, MASc, PhD\}$
Visa Status	Enumerated	$\{Visa, Domestic\}$
Committee Reviewer	Enumerated	GC
Application Status	Enumerated	Refer to Section 5.1
Reviewed Status	Enumerated	$\{Reviewed, NotReviewed\}$

Table 2: Attributes on uploaded applications

A *committee member*, can see a list of student applications due for review or previously reviewed applications. The following table denotes the attributes of a reviewed student application:

Rationale on initial attributes: On the grad apps webpage, the system should be listing the **top 7** attributes that were voted for when seeing the application. This to ensure a reasonable amount of information about an application is contained initially

Attribute	**Type**	**Range**
Date of Application	Date	
Student ID	Big Int	000000000...999999999
Application Status	Enumerated	$\{Reviewed, NotReviewed, DraftAvailable\}$

Table 3: Attributes on uploaded applications

to view an application. With the use of filtering, one can always see more or less fields than the initial view.

5.2. Optional Deliverables

The *Optional* deliverables were decided by the minority vote on a list of questionnaires conducted on the EECS graduate program staffs. Following are the ***Optional*** deliverables for the System Under Description:

- An *admin* may be able to import a CSV file and extend the database with the new data
- A *committee member* completing a review form may be able to have a head start on the form with information automatically extracted and filled such as Name, Student No., GPA
- A *committee member* may be able to download offline forms (XML format) and upload it back into the server
- A *professor* might be interested in other attributes that are to be provided in an organized table and used for advanced filtering
    1. Previous University
    2. GRE
    3. Other English proficiency test scores
        a) IELTS
        b) TOEFL
        c) YELT
- A *professor* may be able to contact students on behalf of a research group (i.e instead of multiple professor from the same field contacting a student, they might want to contact the student as part of a group eg. vision, databases, etc.)
- A *professor* may be able to contact student (email) directly from the application
- A *professor* may be able to leave notes on an application for personal use or for other professors to view when going through the same application

6. System Invariants

INV1	There must be at least one admin in the system.

INV2	User private attributes such as reviewed status and drafted applications can only be used by that user.

INV3	Committee members only have access to the applications assigned to them.

INV4	An application must be reviewed before Professors can access them.

INV5	A user must be logged in with exactly one role.

7. Use Case Diagram

The following is an abstract Use Case Diagram for the System Under Description. The actors in the Use Case Diagram are the: *admin*, *professor* and *committee member* as discussed in Section 4.

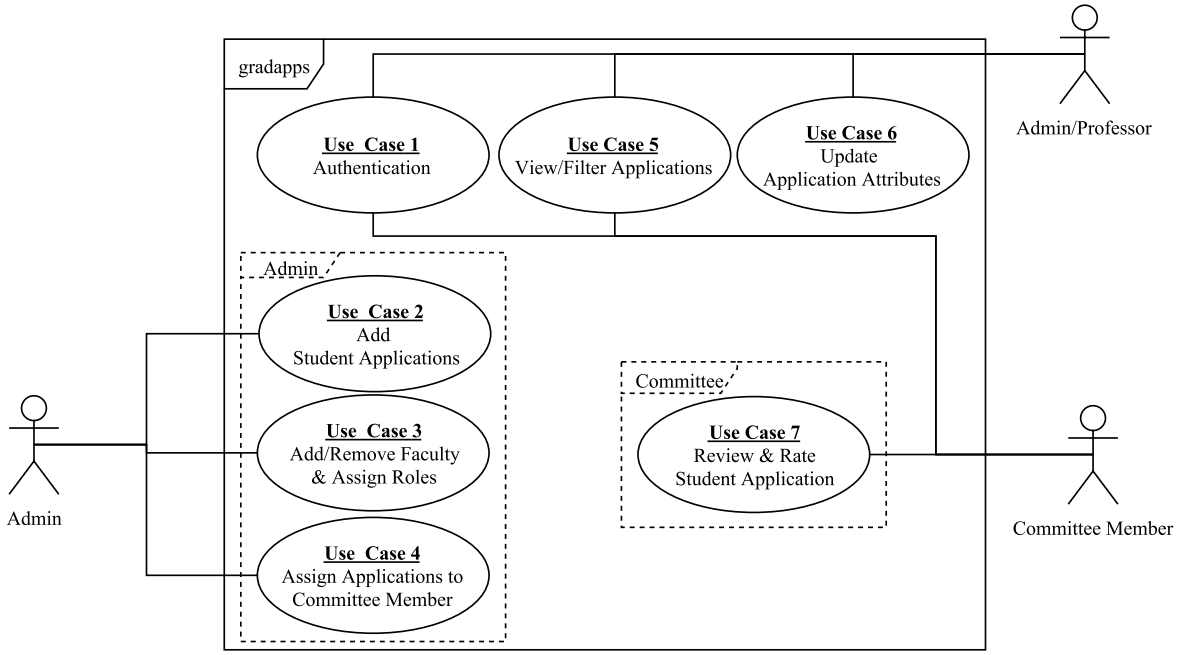

Figure 3: Use Case Diagram for the System Under Description

8. Use Case Textual Description

The following are the Use Case Textual Description for the diagram provided in Section 7. A list of success and error messages are provided in Appendix C and D respectively.

8.1. Use Case 1 - Authentication

The following three use case textual representation describes the *Authentication* use case.

- Table 4 describes a user is logging into the system
- Table 5 describes a user once validated, selecting a role to log in as.
- Table 6 describes a user logging out of the system.

Use Case ID: UC-1A
Use Case Name: Login
Primary Actor: Admin, Professor, Committee Member
Description: A user opens the browser and goes to the portal webpage. The user enters their Passport York username along with the password associated with it. The user clicks the login button, the credentials get authenticated and succeeds. The user is now logged into the system.
Trigger: User indicates to login to the system.
Precondition: PRE-1. The user has a PPY account. PRE-2. The user is not logged into the system.
Postcondition: POST-1. The user is logged into the system.
Normal Flow: 1A.0 Login to the system 1. User opens a web browser and visits the grad apps webpage. 2. System prompts the user to enter their PPY username and password. 3. User enters the username and password and clicks enter. 4. System sends authentication request to PPY and gets a valid response back. 5. The user is redirected to the next page for selecting the role. Extension point to UC-3.
Exception Flow: 1A.0.E1 User ID or Password is not valid 1. System displays error **e1** 2. System awaits for the user to re-enter a correct ID and password (3a) or to exit (3b). 3a. System starts normal flow over. 3b. System terminates the use case.

Table 4: Use Case Textual Description for Login

Use Case ID: UC-1B
Use Case Name: Select Role
Primary Actor: Admin, Professor, Committee Member
Description: A user has logged into the system. A prompt is shown to the user to select a role they are assigned to. The user selects one of the roles and proceeds into the next page.
Precondition: PRE-1. The user is logged in to the system.
Postcondition: POST-1. The user logs in with the selected a role.
Normal Flow: 1B.0 Select Role 1. User selects the role they want to use the system with. 2. System prompts the user to confirm the selection of the role. 3. User chooses yes. 4. System requests all access for the user in the role and succeeds. 5. The user is now able to access the system with a particular role.
Exception Flow: 1B.0.E1 Role selected is invalid 1. System displays error **e2**. 2. System awaits for the user to select another role (3a) or to exit (3b). 3a. System starts normal flow over. 3b. System terminates the use case.

Table 5: Use Case Textual Description for Role Selection

Use Case ID: UC-1C
Use Case Name: Logout
Primary Actor: Admin, Professor, Committee Member
Description: A user is already logged into the system. The user clicks on the logout button. The user's logged in session is successfully terminated by PPY and the user is bought back into the login page.
Trigger: User indicates to logout of the system.
Precondition: PRE-1. The user is logged in to the system. PRE-2. The user has a selected role.
Postcondition: POST-1. The user is logged out of the system.
Normal Flow: 1C.0 Logout of the system 1. User clicks on the logout button. 2. System prompts the user to confirm logging out of the system. 3. User chooses ok. 4. System sends session termination request to PPY and gets a valid response back. 5. The user is logged out and redirected to the login page.
Exception Flow: 1C.0.E1 Already logged out 1. System displays error **e3** 2. User is redirected to the login pages 3. System terminates the use case.

Table 6: Use Case Textual Description for Logout

8.2. Use Case 2 - Add Student Application

Use Case ID: UC-2
Use Case Name: Add Student Application
Primary Actor: Admin
Description: A user has logged into the system. The user wants to add an application to the system. The user selects the option to add an application, a prompt is shown to enter the details about the application and select a PDF formatted file of the application. The user enters the information and submits a request. The system confirms success in uploading the application.
Trigger: User indicates to add an application to the system.
Precondition: PRE-1. The user is logged in. PRE-2. The role selected by the user is admin.
Postcondition: POST-1. The user has added an application to the system.
Normal Flow: 2.0 Add a Student Application 1. User selects the option to add an application. 2. System prompts the user to enter details about the application and select a PDF file. 3. User enters the information and clicks enter. 4. System validates the information provided and succeeds. 5. The user is shown success message **s5**.
Exception Flow: 2.0.E1 File uploaded is not of the correct type 1. System displays error **e9**. 2. System prompts the user to try uploading file of PDF type (3a) or to exit (3b).
Exception Flow: 2.0.E2 Incorrect type of value entered 1. System displays error **e10**. 2. System awaits for the user to re-enter a new value (3a) or to exit (3b).
Exception Flow: 2.0.E3 Not an Admin 1. System displays error **e12**. 2. System exits (3b).
Exception Flow 3a. User requests to try again. System starts normal flow over. 3b. User asks to exit. System terminates the use case.

Table 7: Use Case Textual Description for Adding an Application

8.3. Use Case 3 - Add/Remove Members and Assign Roles

The following four use case textual representation describes the *Add/Remove Members and Assign Roles* use case.

- Table 8 describes adding a new faculty member to the system.
- Table 9 describes removing a faculty member from the system.
- Table 10 describes assigning a role to a faculty member in the system.
- Table 11 describes unassigning a role from a faculty member in the system.

Use Case ID: UC-3A
Use Case Name: Add a Member
Primary Actor: Admin
Description: A user has logged into the system. The user wants to add a new user. The user selects the option to add a new member, a prompt is shown to enter the member's PPY id. The user enters all the information and submits a request. The system confirms success in adding the new member.
Precondition: PRE-1. The user is logged in. PRE-2. The role selected by the user is admin. PRE-3. The member to be added is not in the system.
Postcondition: POST-1. The member has been added to the system.
Normal Flow: 3A.0 Add a new member 1. User selects the option to add a new member. 2. System prompts the user to enter the member ID. 3. User enters the information and clicks enter. 4. System validates the information provided and succeeds. 5. The user is shown a success message **s1**.
Exception Flow: 3A.0.E1 Member with the ID does not exist 1. System displays error **e4**. 2. System prompts the user to try adding another member (3a) or to exit (3b).
Exception Flow: 3A.0.E2 Not an Admin 1. System displays error **e12**. 2. System exits (3b).
Exception Flow 3a. User requests to add another member. System starts normal flow over 3b. User asks to exit. System terminates the use case.

Table 8: Use Case Textual Description for Add a New Member

Use Case ID: UC-3B
Use Case Name: Remove a Member
Primary Actor: Admin
Description: A user has logged into the system. The user wants to remove a user from the system. The user selects the option to remove the member, a prompt is shown to enter the member's PPY id. The user enters all the information and submits a request. The system confirms success in removing the member.
Precondition: PRE-1. The user is logged in. PRE-2. The role selected by the user is admin. PRE-3. The member to be removed is in the system.
Postcondition: POST-1. The member is removed from the system. POST-2. The member is logged out if logged in.
Normal Flow: 3B.0 Remove an existing member 1. User selects the option to remove an existing member. 2. System prompts the user to enter the member ID. 3. User enters the information and clicks enter. 4. System validates the information provided and succeeds. 5. The user is shown a success message **s2**. .
Exception Flow: 3B.0.E1 Member with the ID does not exist 1. System displays error **e5**. 2. System prompts the user to try removing another member (3a) or to exit (3b).
Exception Flow: 3B.0.E2 Not an Admin 1. System displays error **e12**. 2. System exits (3b).
Exception Flow 3a. User requests to remove another member. System starts normal flow over 3b. User asks to exit. System terminates the use case.

Table 9: Use Case Textual Description for Remove an Existing Member

Use Case ID: UC-3C
Use Case Name: Assign a Role
Primary Actor: Admin
Description: A user has logged into the system. The user wants to assign a new role to an existing user. The user selects the option to assign the role of a member, a prompt is shown to enter the member's PPY id and the role to be assigned. The user enters the information and submits a request. The system confirms success in adding the members role.
Precondition: PRE-1. The user is logged in. PRE-2. The role selected by the user is admin. PRE-3. The member is in the system. PRE-4. The role to be assigned is a valid role. Refer to Section 4.
Postcondition: POST-1. The member has a new role associated.
Normal Flow: 3C.0 Assign a new role to an existing member 1. User selects the option to assign a new role to an existing member. 2. System prompts the user to enter the member ID and the role to be assigned. 3. User enters the information and clicks enter. 4. System validates the information provided and succeeds. 5. The user is shown a success message **s3**.
Exception Flow: 3C.0.E1 Member with the ID does not exist 1. System displays error **e5**. 2. System prompts the user to try again on another member (3a) or exit (3b).
Exception Flow: 3C.0.E2 Member already has the role assigned 1. System displays error **e6**. 2. System prompts the user to try again on another member (3a) or exit (3b).
Exception Flow: 3C.0.E3 Not an Admin 1. System displays error **e12**. 2. System exits (3b).
Exception Flow: 3C.0.E4 Role is invalid 1. System displays error **e13**. 2. System prompts the user to try again on another member (3a) or exit (3b).
Exception Flow 3a. User requests to assign a role to another member. System starts normal flow over 3b. User asks to exit. System terminates the use case.

Table 10: Use Case Textual Description for Assign a New Role

Use Case ID: UC-3D
Use Case Name: Unassign a Role
Primary Actor: Admin
Description: A user has logged into the system. The user wants to unassign a role for an existing user. The user selects the option to unassign the role of a member, a prompt is shown to enter the member's PPY id and the role to be unassigned. The user enters the information and submits a request. The system confirms success in removing the members role.
Precondition: PRE-1. The user is logged in. PRE-2. The role selected by the user is admin. PRE-3. The member is in the system. PRE-4. The member is assigned the role to be removed.
Postcondition: POST-1. The member is not associated to the role. POST-2. The member is logged out if using the role removed.
Normal Flow: 3D.0 Unassign a role from an existing member 1. User selects the option to unassign a role of an existing member. 2. System prompts the admin to enter the member ID and the role to be unassigned.user 3. User enters the information and clicks enter. 4. System validates the information provided and succeeds. 5. The user is shown a success message **s4**.
Exception Flow: 3D.0.E1 Member with the ID does not exist 1. System displays error **e5**. 2. System prompts the user to try again on member (3a) or to exit (3b).
Exception Flow: 3D.0.E2 Member already has the role assigned 1. System displays error **e6**. 2. System prompts the user to try again on another member (3a) or to exit (3b).
Exception Flow: 3D.0.E3 Not an Admin 1. System displays error **e12**. 2. System exits (3b).
Exception Flow: 3D.0.E4 Member doesn't have the selected role assigned 1. System displays error **e7**. 2. System prompts the user to try again on another member (3a) or to exit (3b)
Exception Flow 3a. User requests to unassign a role of another member. System starts normal flow over. 3b. User asks to exit. System terminates the use case.

Table 11: Use Case Textual Description for Unassign a Role

8.4. Use Case 4 - Assign Application to Committee Member

Use Case ID: UC-4
Use Case Name: Assign Application
Primary Actor: Admin
Description: A user has logged into the system. The user wants to assign an application to a committee member for review. The user selects the option to assign an application to a committee member, a prompt is shown to enter the member's PPY id and the application to be assigned. The user enters the information and submits a request.
Precondition: PRE-1. The user is logged in with the role admin. PRE-2. The committee member is in the system. PRE-3. The application is not already assigned to the committee member. PRE-4. Number of reviewers are ≤ 2
Postcondition: POST-1. The application is assigned to the committee member. POST-2. Number of reviewers for the application increase by 1.
Normal Flow: 4.0 Unassign a role from an existing member 1. User selects the option to assign an application for review. 2. System prompts the user to enter the member ID and select the application to be reviewed. 3. User enters the information and clicks enter. 4. System validates the information provided and succeeds. 5. The user is shown a success message **s8**. 6. System notifies the committee member an application needs to be reviewed.
Exception Flow: 4.0.E1 Member with the ID does not exist 1. System displays error **e5**. 2. System prompts the user to try again on member (3a) or to exit (3b).
Exception Flow: 4.0.E2 Member already has the application assigned 1. System displays error **e8**. 2. System prompts the user to try again on another member (3a) or to exit (3b).
Exception Flow: 4.0.E3 Not an Admin 1. System displays error **e12**. System exits (3b).
Exception Flow: 4.0.E4 Too many reviewers 1. System displays error **e14**. System exits (3b).
Exception Flow 3a. User requests to try again on another member. System starts normal flow over. 3b. User asks to exit. System terminates the use case.

Table 12: Use Case Textual Description for Assigning Applications For Review

8.5. Use Case 5 - View/Filter Applications

The following two use case textual representation describes the *View/Filter Applications* use case.

- Table 13 describes viewing an application.
- Table 14 describes filtering applications.

Use Case ID: UC-5A
Use Case Name: View Application
Primary Actor: Admin, Committee Member, Professor
Description: A user has logged into the system. The user selects the option to view an application. The system confirms success in opening the application and the application is viewed.
Precondition: PRE-1. The user is logged in. PRE-2. The user has selected a role.
Postcondition: POST-1. The user has viewed an application.
Normal Flow: 5A.0 View Application 1. User selects the option to view an application. 2. System processes the request and opens the application in PDF view. 3. User views the application and closes the window. 4. System successfully closes the application as the PDF view.
Exception Flow: 5A.0.E1 Application is not reviewed (Professor Only) 1. System displays error **e11**. Go to (3a)
Exception Flow: 5A.0.E2 Application is not assigned (Committee Only) 1. System displays error **e15**. 2. System exits (3b).
Exception Flow 3a. System starts normal flow over. 3b. User asks to exit. System terminates the use case.

Table 13: Use Case Textual Description for Viewing Applications

Use Case ID: UC-5B
Use Case Name: Filter Applications
Primary Actor: Admin, Committee Member, Professor
Description: A user has logged into the system. The user sees a list of applications and wants to only see applications that meet a criteria. The user creates a filter (see Appendix B) and the system processes the filter and returns a set of new applications satisfying the filter.
Precondition: PRE-1. The user is logged in. PRE-2. The user has selected a role.
Postcondition: POST-1. The user has filtered a new set of applications. POST-2. The user is displayed the new set of applications.
Normal Flow: 5B.0 Filter Applications 1. User selects the option to filter applications. 2. System prompts the user with a set of available filters. 3. User selects the desired filters and clicks enter. 4. System processes the request and succeeds returning a new set of applications satisfying the filter.
Exception Flow: 5B.0.E1 Incorrect type of value entered 1. System displays error **e10**. 2. System awaits for the user to re-enter a new value (3a) or to exit (3b).
Exception Flow 3a. System starts normal flow over. 3b. System terminates the use case.

Table 14: Use Case Textual Description for Filtering Applications

8.6. Use Case 6 - Update Application

Use Case ID: UC-6
Use Case Name: Update Application
Primary Actor: Admin, Professor
Description: A user has logged into the system. The user wants to update an application. The user selects the attribute to update and clicks enter. The system processes the request updates the attribute on the application if their role permits it. Valid attributes for professors to update could be seen at REQ23 and REQ24. Valid attributes for admins to update could be seen at REQ20.
Precondition: PRE-1. The user is logged in. PRE-2. The user has selected a role.
Postcondition: POST-1. The user has updated an application.
Normal Flow: 6.0 Update an Application 1. User selects an application to update. 2. User selects an attribute to update and enters a new value. see REQ20, REQ23 and REQ24 2. System processes the input and succeeds. 3. System displays a success message **s6**
Exception Flow: 6.0.E1 Incorrect type of value entered 1. System displays error **e10**. 2. System awaits for the user to re-enter a new value (3a) or to exit (3b).
Exception Flow: 6.0.E2 Application is not reviewed (Professor Only) 1. System displays error **e11**. 2. System exits (3b).
Exception Flow 3a. System starts normal flow over. 3b. System terminates the use case.

Table 15: Use Case Textual Description for Updating an Application

8.7. Use Case 7 - Review Student Application

The following four use case textual representation describes the *Review Student Application* use case.

- Table 16 describes saving an ongoing review as a draft.
- Table 17 describes resuming a draft review.
- Table 18 describes applying university assessment on a review.
- Table 19 describes submitting a review.

Use Case ID: UC-7A
Use Case Name: Save Review
Primary Actor: Committee Member
Description: A user has logged into the system. The user is reviewing an application and decides to save the review as a draft for future review. User clicks on the save as a draft option.
Precondition: PRE-1. The user is logged in. PRE-2. The role selected by the user is committee member. PRE-3. The user has an ongoing review.
Postcondition: POST-1. The user has saved the review as draft.
Normal Flow: 7A.0 Save a Review 1. User selects the option to save the review as a draft. 2. System process the request and saves the review. 3. The user is shown a success message **s8**
Exception Flow: 7A.0.E1 Not a Committee Member 1. System displays error **e12**. 2. System exits (3a).
Exception Flow: 7A.0.E2 Application is not assigned 1. System displays error **e15**. 2. System exits (3a).
Exception Flow 3a. User asks to exit. System terminates the use case.

Table 16: Use Case Textual Description for Saving an Ongoing Review Application

Use Case ID: UC-7B
Use Case Name: Resume Review
Primary Actor: Committee Member
Description: A user has logged into the system. The user has some saved reviews. The user selects one of the drafted reviews and selects resume. System processes the request and reopens the review.
Precondition: PRE-1. The user is logged in. PRE-2. The role selected by the user is committee member. PRE-3. The application selected has a draft.
Postcondition: POST-1. The user has resumed a review.
Normal Flow: 7B.0 Resume a Review 1. User selects a review that has been saved. 2. System prompts the user to resume the review. 3. User chooses ok and clicks enter. 4. System processes the request and reopens the review.
Exception Flow: 7B.0.E1 Not a Committee Member 1. System displays error **e12**. 2. System exits (3a).
Exception Flow: 7B.0.E2 Draft doesn't exist 1. System displays error **e17**. 2. System exits (3a).
Exception Flow: 7B.0.E3 Application is not assigned 1. System displays error **e15**. 2. System exits (3a).
Exception Flow 3a. User asks to exit. System terminates the use case.

Table 17: Use Case Textual Description for Resume a Saved Application

Use Case ID: UC-7C
Use Case Name: Apply University Assessment
Primary Actor: Committee Member
Description: A user has logged into the system. The user has reviews to complete. The user clicks on one review and fills out the form. The user types the name of the university the student has attended, the system processes the name and fetches previously assessed comments on that institution. The user is prompted to save, use or modify the assessment for the review. The user selects to use the assessment. The system fetches the assessment and applies it on the review.
Precondition: PRE-1. The user is logged in. PRE-2. The role selected by the user is committee member. PRE-3. The user has an ongoing review.
Postcondition: POST-1. The user has used a previously specified assessment.
Normal Flow: 7C.0 Apply University Assessment 1. User enters the institution name on the assessment field. 2. System processes the text to show previous assessment on that institution. 3. User selects the assessment to use it (4a) or selects save to save it (4b). 4a. System autofills the data for that review with the previous assessment. 4b. System saves that assessment for that review.
Exception Flow: 7C.0.E1 Not a Committee Member 1. System displays error **e12**. 2. System exits (3a).
Exception Flow: 7C.0.E2 Application is not assigned 1. System displays error **e15**. 2. System exits (3a).
Exception Flow 3a. User asks to exit. System terminates the use case.

Table 18: Use Case Textual Description for Apply Previously Used University Assessments

Use Case ID: UC-7D
Use Case Name: Submit Review
Primary Actor: Committee Member
Description: A user has logged into the system. The user has completed some reviews and decides to submit the review. User clicks on the submit option, the system processes the request and succeeds.
Precondition: PRE-1. The user is logged in. PRE-2. The role selected by the user is committee member. PRE-3. The user has completed a review.
Postcondition: POST-1. The user has submitted a review.
Normal Flow: 7D.0 Submit a Review 1. User selects the option to submit the review. 2. System process the request and submits the review. 3. The user is shown a success message **s9**. 4. System notifies the GPA an application has been reviewed.
Exception Flow: 7D.0.E1 Not a Committee Member 1. System displays error **e12**. 2. System exits (3a).
Exception Flow: 7D.0.E2 Application is not assigned 1. System displays error **e15**. 2. System exits (3a).
Exception Flow 3a. User asks to exit. System terminates the use case.

Table 19: Use Case Textual Description for Submit a Review

9. E/R Descriptions

9.1. Use Case 1 - E/R Descriptions

The following are the E/R Descriptions elicited from Use Case 1, refer to Section 8.1

ENV1	Each EECS graduate program member must have an unique username and password associated with their account via PPY.

Rationale: The login credentials will be handled through the Passport York. This is to simplify login uses and since this business system is under York University, PPY is the most secure option.

ENV2	If the PPY is down for maintenance service, the system must not be accessible.	See Env. 1

Rationale: Since the system heavily relies on PPY authentication, if PPY is scheduled down for maintenance, the system becomes inaccessible.

REQ1	A user shall be able to login to the system *iff* they are a member of the EECS graduate program with a role assigned.	$user \in FM$

Rationale: A user shall be able to log into the system if they are a member of the EECS graduate program and have a role assigned to them. The login will be completed using Passport York Authentication (PPY).

REQ2	A user shall be able to logout of the system *iff* they are already logged in.

Rationale: A user shall be able to log out of the system iff they are logged into the system through PPY.

REQ3	A user shall be not both logged in and logged out at the same time.	$(m\_loggedIn) \iff \neg(m\_loggedOut)$

Rationale: For security reasons a user cannot be both logged in and logged out at the same time. If a logged in account gets idle after 15 minutes, the user is automatically logged out (refer to Req. 4).

REQ4	A user shall be logged out of the system after a maximum of 15 minutes of idleness.

Rationale: It is best practice to log out an idle user from a business critical system. This is to make sure the user does not stay logged on forever and to ensure liveness property on a user account.

REQ5	A user shall select a role from the list of roles they are assigned to once logged into the system.	List of all the roles are as specified in Section 4.

Rationale: The roles available to the user upon logging into the system are the roles the user have been assigned to. A user cannot select a role that has not been assigned to them.

REQ6	A user shall be logged in with exactly *one* role at any given time.	$\forall role1, role2 : (user.role = role1 \wedge user.role = role2) \implies (role1 = role2)$

Rationale: A logged in user cannot be logged in with two different roles. This is to avoid conflicting access control between two different roles.

9.2. Use Case 2 - E/R Descriptions

Following are the E/R Descriptions elicited from Use Case 2, refer to Section 8.2

ENV3	The *GPA* must manually compile an application received from the graduate office before sending it for review.

Rationale: This is the manual work that needs to be done by the Graduate Program Assistant. The GPA has agreed it is in their best interest to manually compile all bits of information into one file before proceeding with the application.

REQ7	An *admin* shall be able to upload a student application to the portal.

Rationale: Only an *admin* user shall be able to assign applications for review to a graduate committee member. For simplicity, there is no maximum cap enforced for reviewing an application by a committee member.

9.3. Use Case 3 - E/R Descriptions

Following are the E/R Descriptions elicited from Use Case 3, refer to Section 8.3

REQ8	An *admin* shall be able to add a new member to the list of EECS graduate program staff.

Rationale: Only an *admin* user shall be able to add a new member to the list of EECS graduate program staffs and assign one or more role(s) to them. In fact, an *admin* shall be able to add a new member and assign them to the role of *admin* as well.

REQ9	An *admin* shall be able to remove a member from the list of EECS graduate program staff except themselves.

Rationale: Only an *admin* user shall be able to remove an existing member from the list of EECS graduate program staffs. An *admin* **cannot** remove themselves from the system.

REQ10	An *admin* shall be able to assign a new role to an existing faculty member with an old role except for themselves.

Rationale: Only an *admin* user shall be able to change existing roles of a registered user. An *admin* **cannot** change their own role in the system.

REQ11	An *admin* shall be able to remove a role from an existing graduate program member with a at least one role assigned.

Rationale: A *admin* shall be able to remove a role assigned to an existing graduate member. This is to take away privileges on performing certain actions in the system.

9.4. Use Case 4 - E/R Descriptions

Following are the E/R Descriptions elicited from Use Case 4, refer to Section 8.4

REQ12	An *admin* shall be able to assign applications for review to a graduate committee member.	The set GC denotes the set of all graduate committee members which is a subset of all members in the system, $GC \subseteq FM$.

Rationale: Only an *admin* user shall be able to assign applications for review to a graduate committee member. For simplicity, there is no maximum cap enforced for reviewing an application by a committee member.

REQ13	A *committee member* shall be notified when a batch of applications come in for review.

Rationale: After an *admin* sends out application(s) for review to a committee member, the *committee member* gets an in app notification associated with their PPY.

9.5. Use Case 5 - E/R Descriptions

Following are the E/R Descriptions elicited from Use Case 5, refer to Section 8.5

REQ14	An *admin* shall be able to export the applications to CSV format.

Rationale: Only an *admin* user shall be able to download a CSV format of the applications and make changes to the file. Being able to import the changed CSV file is an optional requirement (refer to Section: 5.2) that may be implemented if time permits and all required deliverables have been achieved.

REQ15	A *committee member* shall be able to see the list of assigned application(s).

Rationale: A *committee member* shall be allowed to view a list of new (to be reviewed) and previously reviewed application(s). This will allow the committee members to have an organized view of the applications left to review and the applications that have been reviewed already.

REQ16	A *committee member* shall be able to apply filtering only on selected attributes.	Refer to Table 3

Rationale: Allowing extra filtering on assigned application will cause favouritism of some student X while student Y will never get reviewed. Thus, it has been decided to only allow filtering on already reviewed application(s).

REQ17	An *admin* shall be able to view a list of all student application(s).	refer to Req. 19

REQ18	An *professor* shall be able to view a list of student application(s) approved by an admin.	refer to Req. 19

Rationale: An *admin* and *professor* shall be allowed to view a list of student application(s). The list of students shown can be narrowed/expanded using filters (refer to Req. 19).

REQ19	An *admin* and *professor* shall be able to filter on all student applications including personal attributes such as review status.

Rationale: An *admin* and a *professor* shall be able to apply filters on a list of student applications. This is to allow narrow/expand searches of student on various attributes.

9.6. Use Case 6 - E/R Descriptions

Following are the E/R Descriptions elicited from Use Case 6, refer to Section 8.6

ENV4	The *GPA* must contact the institutions graduate admission office whenever there needs to be a form of communication.

Rationale: The GPA can contact the institutions graduate admission office for further information on application if needed.

ENV5	The *GPA* must send the final decision of a student application to the institutions graduate admission office.

Rationale: The GPA must send the final decision of an application to the institutions graduate admission office.

REQ20	An *admin* shall be able to update all attributes of a student application.	refer to Table 2

Rationale: Only an *admin* user shall be able to update all attributes of an application (refer to Table 2). This to allow updating secured information on the application that maybe confidential.

REQ21	The *GPA* shall be notified when a *professor* has requested a student for admission.	refer to Env. 5

Rationale: The *GPA* shall be notified when a *professor* requests a student for admission. The graduate program assistant can then contact the graduate office with further information (refer to Env. 5).

REQ22	A *professor* shall be able to review a student application.

Rationale: A *professor* shall be allowed to view a posted student application. Once an application has been viewed by the professor, it is automatically checked **Reviewed** and this status is only visible to the professor user who have viewed it. For all not reviewed applications, the **Reviewed Status** field is left empty.

REQ23	A *professor* shall be able to contact a student.

Rationale: A *professor* shall be allowed to contact a student once their application satisfies the professor's need. The contact shall need to be made through external email and the professor needs to update the application to **Contacted**. We leave this upto the user to use this feature correctly. The contacted status is visible to all graduate program members who have access to the portal (i.e all *admins* and *professors*).

REQ24	A *professor* shall be able to request a student for admission.	refer to Req. 21 and Env. 5

Rationale: A *professor* shall be allowed to request a student for admission. Once the request has been sent in, the *GPA* will receive a notification (refer to Req. 21) and then can reach out the institutions graduate office with a request for admission (refer to Env. 5).

9.7. Use Case 7 - E/R Descriptions

Following are the E/R Descriptions elicited from Use Case 7, refer to Section 8.7

REQ25	The *GPA* shall be notified when reviews for an application have been completed.

Rationale: Only the *GPA* shall be notified when the review has been completed. It shall be a 1-to-1 relation to avoid multiple levels of notification between users.

REQ26	A *committee member* shall be able to save ongoing reviews as a draft for future completion.

Rationale: A *committee member* shall be allowed to save ongoing reviews as a draft. A draft can be later resumed for completion. This gives the committee member an opportunity to not rush through the review.

REQ27	A *committee member* shall be able to view, use, add or modify a university assessment if it has already been provided.

Rationale: A *committee member* shall be allowed to view, use or modify a previous university assessment used in a review. This removes the work of typing in the same university assessment or even a modified assessment of a previously assessed university.

REQ28	A *committee member* shall be able to submit a completed review to the *GPA*.	Refer to Req. 25

Rationale: A *committee member* shall be able to submit a completed review to the *GPA*. The *GPA* can then upload the application to the portal for professors to access the application.

10. Abstract UI Grammar for Monitored Events

The following user inputs are defined as the abstract UI grammar for The System Under Description.

```
-- business system for the graduate department of EECS --
system gradapps

/************************ definitions ************************/

-- set of all PPY ids at York University --
type PPY_IDS = STRING

-- set of roles --
-- the empty role is to hold users who can be --
-- added to the system without a role --
type ROLES = {"admin", "professor", "committee member"}

-- users who have access to the system is --
-- a function from PPY_IDS to an array of roles --
type USERS = PPY_IDS |-> [ROLES]

-- logged in users are the users who --
-- are logged into the system --
-- i.e a user can be only logged in with one role --
typed LOGGED_IN_USERS = PPY_IDS |-> ROLES

-- the fields of an application --
-- as specified in Table 3 --
type ATTRIBUTES
-- set of all graduate applications submitted to EECS --
type APPLICATIONS = TUPLE[id: NUMBER, attr: ATTRIBUTES]
-- set of all uploaded applications into the system --
type UPLOADED_APPS = TUPLE[id: NUMBER, attr: ATTRIBUTES]

-- definition of filter from the glossary --
-- a filter can be organized with an array --
-- of attributes with a given order --
type FILTERS = TUPLE[[attr: ATTRIBUTES], order: {ASC, DESC}]
```

```
-- review form is a tuple containing --
-- all needed information for reviewing --
-- an application --
type REVIEW_FORM = TUPLE[STUD_ID: NUMBER, ASSESSMENT: STRING, ...]
type REVIEWS = APPLICATIONS |-> REVIEW_FORM

-- status of application review --
type REV_STATUS = {"new", "draft", "in progress", "completed",
"submitted"}

-- set of all review applications in different phases --
type REV_APPLS = REVIEWS |-> [TUPLE[PPY_IDS, REV_STATUS]]

-- set of all actions when applying an assessment --
type ACTIONS = {save, modify, use}

login(mid: PPY_IDS, mrole: ROLES)
        -- pre:
                1. USERS(mid) must return at least one role
                2. LOGGED_IN_USERS(mid) must return null
                3. mrole must be a valid role
        -- post:
                1. LOGGED_IN_USERS(mid) must return mrole

logout(mid: PPY_IDS, mrole: ROLES)
        -- pre:
                1. USERS(mid) must return at least one role
                2. LOGGED_IN_USERS(mid) must return mrole
        -- post:
                1. LOGGED_IN_USERS(mid) must return null
```

```
add_member(mid: PPY_IDS, role: ARRAY[ROLES])
        -- pre:
                1. USERS(mid) must return null
                2. role must contain all valid roles
        -- post:
                1. USERS(mid) must return role

remove_member(mid: PPY_IDS)
        -- pre:
                1. USERS(mid) must return at least one role
                2. LOGGED_IN_USERS(mid) must return null
        -- post:
                1. USERS(mid) must return null

assign_role(mid: PPY_IDS, role: ROLES)
        -- pre:
                1. USERS(mid) must return at least one role
                2. USERS(mid) must not include role
        -- post:
                1. USERS(mid) must include role

unassign_role(mid: PPY_IDS, role: ROLES)
        -- pre:
                1. USERS(mid) must return at least one role
                2. USERS(mid) must include role
                3. LOGGED_IN_USERS(mid) must not include role
        -- post:
                1. USERS(mid) must not include role

upload_application(mid: PPY_IDS, appl: APPLICATIONS)
        -- pre:
                1. USERS(mid) must return at least one role
                2. LOGGED_IN_USERS(mid) must return admin
                3. appl is not an element of UPLOADED_APPS
        -- post:
                1. appl is an element of UPLOADED_APPS
```

```
view_application(mid: PPY_IDS, appl: APPLICATIONS)
        -- pre:
                1. USERS(mid) must return at least one role
                2. LOGGED_IN_USERS(mid) must return a role
                3. appl is an element of UPLOADED_APPS

filter_applications(mid: PPY_IDS, appl: APPLICATIONS,
        filter: FILTERS, new_appls: APPLICATIONS)
        -- pre:
                1. USERS(mid) must return at least one role
                2. LOGGED_IN_USERS(mid) must return a role
                3. appl is a subset of UPLOADED_APPS
                4. filter is an element of FILTERS
        -- post:
                1. new_appls is a subset of UPLOADED_APPS

update_application(mid: PPY_IDS, appl: APPLICATIONS,
        attr: ATTRIBUTES; data: VALUE)
        -- pre:
                1. USERS(mid) must return at least one role
                2. LOGGED_IN_USERS(mid) must return admin or
                professor
                3. appl is a subset of UPLOADED_APPS
                4. attr is a subset of ATTRIBUTES
        -- post:
                1. appl[attr] must return data

assign_review(mid: IDS, rev: REVIEWS)
        -- pre:
                1. USERS(mid) must return at least one role
                2. USERS(mid) must include committee member
                3. REV_APPLS(rev) must not include mid
        -- post:
                1. REV_APPLS(rev) must include a tuple of mid
                and status of newly assigned
```

```
save_review(mid: PPY_IDS, rev: REVIEWS)
        -- pre:
                1. LOGGED_IN_USERS(mid) must return
                committee member
                2. REV_APPLS(rev) must include a tuple of mid
                and status of in progress
        -- post:
                1. REV_APPLS(rev) must include a tuple of mid
                and status of draft

resume_review(mid: PPY_IDS, rev: REVIEWS)
        -- pre:
                1. LOGGED_IN_USERS(mid) must return
                committee member
                2. REV_APPLS(rev) must include a tuple of mid
                and status of draft
        -- post:
                1. REV_APPLS(rev) must include a tuple of mid
                and status of in progress

submit_review(mid: PPY_IDS, rev: REVIEWS)
        -- pre:
                1. LOGGED_IN_USERS(mid) must return
                committee member
                2. REV_APPLS(rev) must include a tuple of mid
                and status of completed
        -- post:
                1. REV_APPLS(rev) must include a tuple of mid
                and status of submitted
```

```
apply_assessment(mid: PPY_IDS, rev: REVIEWS,
        appl: APPLICATIONS, uni: STRING,
        asmt: STRING, act: ACTIONS)
        -- pre:
                1. LOGGED_IN_USERS(mid) must return
                committee member
                2. REV_APPLS(rev) must include a tuple of mid
                and status of in progress
                3. act is a valid action
        -- post:
                1. rev[appl][ASSESSMENT] must return asmt
```

11. ASCII encoded output of the abstract state

A sequence of user commands, e.g. at0.txt below, shall conform to the abstract UI grammar specified in Section 10.

```
-- Acceptance test for GradApps: at0.txt
-- Very basic Use Case to:
        -- An admin logs into the system:
                -- upload some applications
                -- add some new members
                -- assign them to review applications
                -- logs out of the system

-- login to the system
login("user1", "admin")

-- upload some applications
upload_application("user1", app1)
upload_application("user1", app2)

-- add some new members
add_member("user2", ["professor", "committee member"])
add_member("user3", ["committee member"])
add_member("user4", ["professor", "admin"])

-- assign an application for review
assign_review("user2", app1)
assign_review("user3", app2)
assign_review("user2", app2)

-- logout of the system
logout("user1", "admin")
```

The output of the acceptance test shall display as follows:

```
  \** NOTE: The definitions that are not mentioned in the state are
  not changed in this specific use case. As a result we do not
  specify them. \**

  state 0 ok
  USERS = {"user1" |-> ["admin"]}
  LOGGED_IN_USERS = {}
  UPLOADED_APPS = {}
  REVIEWS = {}
  REV_APPLS = {}

->login("user1", "admin")
  state 1 ok
  USERS = {"user1" |-> ["admin"]}
  LOGGED_IN_USERS = {"user1" |-> "admin"}
  UPLOADED_APPS = {}
  REVIEWS = {}
  REV_APPLS = {}

->upload_application("user1", app1)
  state 2 ok
  USERS = {"user1" |-> ["admin"]}
  LOGGED_IN_USERS = {"user1" |-> "admin"}
  UPLOADED_APPS = {app1}
  REVIEWS = {}
  REV_APPLS = {}

->upload_application("user1", app2)
  state 3 ok
  USERS = {"user1" |-> ["admin"]}
  LOGGED_IN_USERS = {"user1" |-> "admin"}
  UPLOADED_APPS = {app1, app2}
  REVIEWS = {}
  REV_APPLS = {}
```

```
->add_member("user2", ["professor", "committee member"])
  state 4 ok
  USERS = {"user1" |-> ["admin"], "user2" |->
  ["professor", "committee member"]}
  LOGGED_IN_USERS = {"user1" |-> "admin"}
  UPLOADED_APPS = {app1, app2}
  REVIEWS = {}
  REV_APPLS = {}

->add_member("user3", ["committee member"])
  state 5 ok
  USERS = {"user1" |-> ["admin"], "user2" |->
  ["professor", "committee member"], "user3" |->
  ["committee member"]}
  LOGGED_IN_USERS = {"user1" |-> "admin"}
  UPLOADED_APPS = {app1, app2}
  REVIEWS = {}
  REV_APPLS = {}

->add_member("user4", ["professor", "admin"])
  state 6 ok
  USERS = {"user1" |-> ["admin"], "user2" |->
  ["professor", "committee member"], "user3" |->
  ["committee member"], "user4" |-> ["professor", "admin"]}
  LOGGED_IN_USERS = {"user1" |-> "admin"}
  UPLOADED_APPS = {app1, app2}
  REVIEWS = {}
  REV_APPLS = {}

->assign_review("user2", app1)
  state 7 ok
  USERS = {"user1" |-> ["admin"], "user2" |->
  ["professor", "committee member"], "user3" |->
  ["committee member"], "user4" |-> ["professor", "admin"]}
  LOGGED_IN_USERS = {"user1" |-> "admin"}
  UPLOADED_APPS = {app1, app2}
  REVIEWS = {app1 |-> app1_form}
  REV_APPLS = {(app1 |-> app1_form) |-> ("user2", "new")}
```

```
->assign_review("user3", app2)
  state 8 ok
  USERS = {"user1" |-> ["admin"], "user2" |->
  ["professor", "committee member"], "user3" |->
  ["committee member"], "user4" |-> ["professor", "admin"]}
  LOGGED_IN_USERS = {"user1" |-> "admin"}
  UPLOADED_APPS = {app1, app2}
  REVIEWS = {app1 |-> app1_form, app2 |-> app2_form}
  REV_APPLS = {(app1 |-> app1_form) |-> ("user2", "new"),
  (app2 |-> app2_form) |-> ("user3", "new")}

->assign_review("user2", app2)
  state 9 ok
  USERS = {"user1" |-> ["admin"], "user2" |->
  ["professor", "committee member"], "user3" |->
  ["committee member"], "user4" |-> ["professor", "admin"]}
  LOGGED_IN_USERS = {"user1" |-> "admin"}
  UPLOADED_APPS = {app1, app2}
  REVIEWS = {app1 |-> app1_form, app2 |-> app2_form}
  REV_APPLS = {(app1 |-> app1_form) |-> ("user2", "new"),
  (app2 |-> app2_form) |-> ("user3", "new"),
  (app2 |-> app2_form) |-> ("user2", "new")}

->logout("user1", "admin")
  state 10 ok
  USERS = {"user1" |-> ["admin"], "user2" |-> ["professor",
  "committee member"], "user3" |-> ["committee member"],
  "user4" |-> ["professor", "admin"]}
  LOGGED_IN_USERS = {}
  UPLOADED_APPS = {app1, app2}
  REVIEWS = {app1 |-> app1_form, app2 |-> app2_form}
  REV_APPLS = {(app1 |-> app1_form) |-> ("user2", "new"),
  (app2 |-> app2_form) |-> ("user3", "new"),
  (app2 |-> app2_form) |-> ("user2", "new")}
```

12. Appendices

A. Additional Use Case Textual Descriptions

Use Case ID: UC-8
Use Case Name: Read Notification
Primary Actor: Admin, Committee Member
Description: A user has logged into the system. The user sees a notification and clicks on it. The system processes the notification and marks it being read.
Precondition: PRE-1. The user is logged in. PRE-2. The user has a selected role. PRE-2. The user has a notification.
Postcondition: POST-1. The notification is marked as read.
Normal Flow: 8.0 Read a notification 1. User sees a new notification and clicks to expand the notification. 2. System displays the notification message to the user and marks it as read.
Exception Flow: 8.0.E1 Not a Committee Member 1. System displays error **e12**. 2. System exits (3a).
Exception Flow: 8.0.E2 Not an Admin 1. System displays error **e12**. 2. System exits (3a).
Exception Flow: 8.0.E3 Application is not assigned (Committee) 1. System displays error **e15**. 2. System exits (3a).
Exception Flow 3a. User asks to exit. System terminates the use case.

Table 20: Use Case Textual Description for Notification

Use Case ID: UC-9
Use Case Name: Export Applications
Primary Actor: Admin
Description: A user has logged into the system. The user wants to export all the applications into CSV format. The user chooses the option to export application, selects all the applications and clicks enter. The system processes the request and succeeds. A CSV formatted file is downloaded to the user.
Precondition: PRE-1. The user is logged in. PRE-2. The role selected by the user is admin. PRE-3. The user has selected all the applications to export.
Postcondition: POST-1. The user has exported all the applications in CSV format.
Normal Flow: 9.0 Export Applications 1. User selects the option to export applications. 2. System prompts the user to select all the fields or specific fields. 3. User selects all fields and clicks enter. 4. System processes the request and succeeds. 5. The user is presented with a CSV formatted file of applications.

Table 21: Use Case Textual Description for Exporting Application(s)

B. Glossary

The following are the definition of fuzzy terms used in this document:

- **Export Applications**—The ability to download all or selected applications in CSV format.
- **Filter**—The ability to select multiple application attributes, limit their values and build a query using *intersection* of those attributes.
- **Filtering Applications**—The ability to use a filter as described above and apply it on all applications avaliable to you.
- **Application Attributes**—Valid Application Attributes for Admins/Professors: Table 2, Committee Members: Table: 3, Explanation: Section 5.1

C. Success Message List

The following is a list of success messages to be displayed in the system when applicable. See Section 8

s1: Member, M, has been successfully added.
s2: Member, M, has been successfully removed
s3: Member, M, has been assigned a new role R
s4: Member, M, has been unassigned a new role R
s5: Application successfully added.
s6: Application Attribute, A, has been updated.
s7: Member, M, has been assigned application A for review.
s8: Review has been saved as a draft for Member, M.
s9: Review has been submitted by Member, M.

D. Error Message List

The following is a list of errors to be displayed in the system when applicable. See Section 8

e1: Incorrect User ID or Password.
e2: You don't have permission to select this role. Please contact your administrator.
e3: You are already logged out of the system.
e4: Member, M, does not exist.
e5: Member, M, does not exist in the system.
e6: Member, M, is already assigned to role R.
e7: Member, M, does not have role, R, assigned.
e8: Application, A, has already been assigned to member, M.
e9: File chosen is not of the correct type.
e10: Value for <attribute name>is not of the correct type.
e11: Application is not reviewed.
e12: You don't have permission to perform this action. Please contact your administrator.
e13: Role, R, is not a valid role..
e14: Too many reviewers for the application.
e15: Application is not assigned to member.
e16: Filter is invalid.
e17: Draft does not exist for this application.