

# **Final Release Documentation**

## **COSC 4P02**

Version 1.0

Due: April 26th, 2021

### **R8Scholar**

[Github](#)

Team Leader (Scrum Master):

Twino Puthiakunnel / [mp13ko@brocku.ca](mailto:mp13ko@brocku.ca) / 5564182 / mp13ko

Product Owner:

Seth Shickluna / [ss16wn@brocku.ca](mailto:ss16wn@brocku.ca) / 6217558 / SethShickluna

Erikas Klimusinas / [ek15kg@brocku.ca](mailto:ek15kg@brocku.ca) / 5903547 / ErikasK

Grant Nike / [gn17az@brocku.ca](mailto:gn17az@brocku.ca) / 6349302 / GrantNike

James Sargent / [js17sy@brocku.ca](mailto:js17sy@brocku.ca) / 6380356 / shorinbonsai

Logan Bell / [lb16tp@brocku.ca](mailto:lb16tp@brocku.ca) / 6047211 / loganbe11

Luciano Ugalde / [lu16xx@brocku.ca](mailto:lu16xx@brocku.ca) / 6102545 / lu16xx

Munashe Masango / [mm16rh@brocku.ca](mailto:mm16rh@brocku.ca) / 6204911 / munashemasango

# Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>Introduction:</b>	<b>7</b>
Project Objective:	7
System Expectations	7
Team Expectations	7
<b>Release:</b>	<b>8</b>
Final Release goals:	8
New Features in Final Release:	8
Iteratives:	9
<b>User stories / Progress report / Backlog:</b>	<b>9</b>
Signup	9
Email Confirmation	9
Log In	10
Sign out	10
Change password	10
Change nickname	11
Delete account	11
Password Reset	12
Create Review	12
Edit Review	13
Report Review	13
Delete Review	13
Create comments	14
Edit Comment	14
Delete Comment	14
View Instructors	15
View Departments	15
View Courses	15
Search	15
View Ratings	16
Thumbs Up / Down	17
Admin: Edit Users	17
Admin: Add Reviews, Courses, Instructors and Departments	18
Admin: Edit Reviews, Courses, Instructors and Departments	18
Admin: Delete Reviews, Courses, Instructors and Departments	18

<b>Updated Requirements</b>	<b>19</b>
1.1 Purpose	19
2.1 Functional Requirements	20
2.1.1 Sign Up	20
2.1.2 Log In	21
2.1.3 Log Out	21
2.1.4 Edit Profile	22
2.1.5 Create Review	22
2.1.6 Edit Review	23
2.1.7 Search For Entity	23
2.1.8 Write Comment	24
2.1.9 Edit Comment	24
2.1.10 View Entity By Category	25
2.1.11 Contact System Administrator	25
2.1.12 Create Forum Post	26
2.1.13 Comment on Forum Post	26
2.1.14 Respond to Review	27
2.1.15 View All Reviews	27
2.1.16 Search Through Reviews	28
2.1.17 Voting System	28
2.1.18 Report Post	29
2.1.19 Remove Review	29
2.1.20 Remove Comment	30
2.1.21 Remove Topic	30
2.1.22 Change Visibility	31
2.1.23 Ban User	31
2.1.24 Add Entity	32
2.1.25 Remove Entity	32
2.2 User Interface Requirements	33
2.4 Stakeholder Requirements	35
2.5 Non-Functional Requirements	35
2.5.1 Product Requirements	35
2.5.2 Organizational Requirements	35
2.5.3 External Requirements	36
<b>3.0 Architectural Design</b>	<b>37</b>
3.1.1 Deployment Diagram	37
3.1.2 Deployment Explanation	37
3.2 Use Case Diagram(s)	38
3.3 Class Diagram(s)	39
3.3 Sequence Diagram(s)	40

3.3.1 Create Review	40
3.3.2 Add Review Item	41
3.3.3 Login	42
3.3.4 Signup	43
3.3.5 Selecting Review Item	44
3.3.6 Search Website	45
3.3.7 Comment on Review	46
3.3.8 Accessing Forum	47
3.3.9 Navigation	48
3.3.10 Edit Review Item	49
<b>4.0. Database Design</b>	<b>50</b>
4.1 ER Diagram	50
4.2 Database Tables	51
4.2.1 User Entity	51
4.2.2 Subject Entity	51
4.2.3 Course Entity	52
4.2.4 Instructor Entity	52
4.2.5 Review Entity	52
4.2.6 Comment Entity	53
4.2.7 Forum Entity	54
4.2.8 Department Entity	54
4.2.9 Ticket	54
<b>Security</b>	<b>55</b>
<b>Host/Domain</b>	<b>55</b>
Development host and domain:	55
Production host and domain:	56
<b>Testing:</b>	<b>56</b>
<b>Informal Function Testing:</b>	<b>56</b>
Admin Functions	56
Top bar	56
Home page	57
Sign up	57
Sign In	57
Reviews	57
Searches	57
<b>Formal UI/Web Testing:</b>	<b>58</b>
Sign Up	58
Log in	59
Log out	61

Edit Nickname	62
Change Password	62
Delete Profile	63
Courses Page:	64
Going to first	64
Going to Last	64
Opening Course	65
Opening Department	65
Sorting System	66
Instructors Page:	66
Going to first	66
Going to Last	67
Opening Instructor	67
Opening Department	68
Sorting System	68
Departments Page:	69
Going to first	69
Going to last	69
Opening Top Instructor	70
Opening Top Course	70
Opening Department	71
Sorting System:	71
Individual Course / Instructors / Departments Page	72
Create Review	73
Edit Review	74
Report Review	74
Search	74
<b>Formal Unittest Testing:</b>	<b>76</b>
Models Test Cases:	76
Test str	76
Email User	76
Has Permission	77
Has Module Permissions	77
Is Staff	77
Update Department Rating	77
Update Instructor Rating within Department	78
Update Course Rating within Department	78
Update Course Rating	78
Update Instructor Rating	78
Generators Test Case	79
Validators Test Case:	79

Validate Brock Email	79
Profanity Validator	79
Password Validator	80
Rating Validator	80
Managers Test Case:	80
Create User	80
Create Superuser	80
Database Test Case	81
<b>Sprints / Teamwork policy:</b>	<b>82</b>
Meeting 1 - First Group Meeting - Jan 11th	82
Meeting 2 - Discuss Requirements Elicitation - Jan 18th	82
Meeting 3 - Share resources and information - Jan 22nd	82
Meeting 4 - Ask Professor Naser questions - Jan 25th	82
Meeting 5 - Finalize Requirements - Jan 29th	83
Meeting 6 - Sprint Planning 1 - Feb 1th	83
Meeting 7 - Sprint Refinement 1 - Feb 8th	83
Meeting 8 - Sprint Review Meeting 1 - Feb 14th	83
Meeting 9 - Sprint Planning 2 - Feb 15th	83
Meeting 10 - Sprint Refinement 2 - Feb 18th	83
Meeting 11 - Sprint Planning 3 (Review 2) - Feb 22th	84
Meeting 12 - Sprint Review 3 (FullStack 3) - Feb 27th	84
Meeting 13 - Sprint Planning 4 (Release 1) - March 1st	84
Meeting 14 - Sprint Review 4 (Release 1) - March 5th	84
Meeting 15 - Release Meeting - March 7th	84
Meeting 16 - Sprint Planning 5 - March 8th 2021	85
Meeting 17 - Sprint Refinement 5 - March 11th 2021	86
Meeting 18 - Sprint Planning 6 (Review 5) - March 16th 2021	87
Meeting 19 - Sprint Planning 7 (Review 6) - March 22th 2021	87
Meeting 20 - Sprint Refinement 7 - March 25th 2021	88
Meeting 21 - Sprint Planning 8 (Review 7) - March 29th 2021	88
Meeting 22 - Sprint Refinement 8 (Release 2) - April 1st 2021	89
Meeting 23 - Sprint Planning 9	89
Meeting 24 - Sprint Refinement 9	90
Meeting 25 - Sprint Planning 10	90
Meeting 26 - Sprint Refinement 10	91
Meeting 27 - Sprint Planning 11 (Review 10)	92
Meeting 28 - Sprint Refinement 11	94
Meeting 29 - Sprint Planning 12 (Review 11)	94
Meeting 30 - Presentation Preparation	94

## Introduction:

COSC 4P02 R8Scholar Final Release Documentation covering software development from date to date. Includes release goals/features, user stories/progress reports/backlog, formal/informal testing, sprints and teamwork policy.

Add security and accessibility(maybe) section.

## Project Objective:

The objective of this project is to deliver a comprehensive experience for Brock students to be able to communicate about their experiences and to give advice/information on potential interactions they might have at the school.

## System Expectations

- Online platform to rate scholarly aspects of Brock University
- Anyone should be able to view all ratings on the site
- Verified brock students should be able to leave ratings on the site
  - Signup / Login
    - Verify Brock Email
- Users must be able to edit or remove their reviews
- Users must be able to delete their account
  - All of users reviews will also be removed
- Needs to display a page for each department, course, and instructor
  - Departments can list staff and courses
    - All of these are dynamically generated, based off database entries
- Search feature on front page
  - Able to search for courses, instructors or departments
- Users must be able to comment on reviews

## Team Expectations

The team will be looking to take the project on using the SCRUM methodology, the reason being that SCRUM offers an agile development and the freedom to work in concentrated teams but with a common goal. The interleaved nature of updated requirements and incrementing of software features seems just right for this type of project.

The team will attend all weekly meetings, and collaborate to develop R8Scholar.

The timing for releases/iterations of the project will be followed according.

## Release:

The final release is focused on making sure all existing features are functioning as expected and hosting R8Scholar on a production ready server with a domain.

### Final Release goals:

- Get user to verify their accounts on account creation
  - Add terms and conditions to sign up
- Increase filtering options
  - by name, by department, by rating (high or low)
- Sending out confirmation emails
- Token authentication
- Allow user to delete their account, cascade delete their reviews, and their comments
- Allow user to delete reviews, cascade delete all comments on review
- List users' reviews on their profile page
- It should be possible to edit and delete reviews
- It should be possible to view, add, edit, and report comments
- Department page should displays all the courses in the department
- API page should not be accessible publicly
- Have a profanity filter to keep site professional

### New Features in Final Release:

- Limited public access to pages. e.g. api pages
- Attribute Reviews to the user (display on profile)
- Attribute Comments to the user
- Ability to edit, report, and delete Reviews
- Token-Based Authentication
- User prompts on verify (junk folder) and signup (don't use brock pass)
- Having professor to rate students (Final Release)
- Ability to search by course name i.e. "Advanced Programming"
- Form validation
- Updated password requirements
- Ability to view all user reviews in profile
- Profanity filter



## Iteratives:

### User stories / Progress report / Backlog:

#### Signup

**Story:** As a user, I would like to be able to sign up for a R8Scholar account.

**Acceptance criteria:**

1. Check to determine if the user has a valid Brock email account
  - a. If there the user provided a valid email address then check to see if the password is valid
    - i. If the password is valid allow user to create account
    - ii. If the password is invalid inform user that password is invalid
  - b. If the email is invalid the user will be informed to provide a Brock email address
2. User will have to agree to terms and conditions
3. The user will have to input a confirmation code which will be sent to their email address
4. User will be brought to login page after entering confirmation code

**Progress report:** Users are able to sign up for a R8Scholar account.

1. Checks if the user has provided a valid main as well as password
2. Makes user agree to terms and conditions
3. User is able to input a confirmation code to verify their account

#### Email Confirmation

**Story:** As a user, I would like to have a confirmation email sent to my email address when I'm creating my account.

**Acceptance criteria:**

1. Check to determine if the user has a valid Brock email account
  - a. If there the user provided a valid email address then a confirmation email will be sent to that address
  - b. If the email is invalid the user will be informed to provide a Brock email address
2. The email will contain a hyperlink which allows the user to jump instantly to the R8Scholar site

**Progress report:** When users create an account an confirmation email is sent with a code.

1. If the user used a valid Brock email account they will receive an email
2. The email contains a hyperlink directing to the R8Scholar site
3. The email has a code which is used for account verification

## Log In

**Story:** As a user, I would like to be able to log in to the site using my email address and password.

**Acceptance criteria:**

1. Check to determine if the email address is valid
  - a. If the email address is valid check to see if the password is valid
    - i. If the password is valid allow user to log in
    - ii. If the password is invalid inform user that password is incorrect
  - b. If the email address is invalid inform user that email is invalid
2. After logging in, the user will be brought to the main homepage

**Progress report:** Able to log into site using Brock University email address, and their password.

1. Their email address is checked when inputted
  - a. Informs them if email address is invalid
2. After logging in they are brought to their home page

## Sign out

**Story:** As a user, I would like to log out from the site from any page.

**Acceptance criteria:**

1. Display logout button on top right of site
2. Allow user to click on log out button and log out immediately
3. User should be brought to home page

**Progress report:**

1. Users are able to see a logout button at the top right of the site while logged in
2. Users are able to click the logout button and sign out immediately
3. User is brought to home page

## Change password

**Story:** As a user, I would like to be able to go to my profile and change my password

**Acceptance criteria:**

1. Check to see if the user is logged in
  - a. If not, prompt log in (see log in story)
2. Enter current password, and enter new password
  - a. If current password is correct, the new password is accepted
3. User will be informed if password was changed or any error occurred

**Progress report:**

1. Profile page is only accessible when user is logged in
2. Profile page has a tab to change password
  - a. User will enter their current password
  - b. They will type in their new password twice
  - c. Password will be changed if their current password is correct and their new password matches and meets all the criteria listed
3. User is not informed if password was changed or any error occurred \*backlog\*

## Change nickname

**Story:** As a user, I would like to be able to go to my profile and change my nickname

**Acceptance criteria:**

1. Check to see if the user is logged in
  - a. If not, prompt log in (see log in story)
2. Enter new nickname
  - a. If a nickname has profanity it will not be allowed
3. Enter current password
4. User will be informed if nickname was changed or any error occurred

**Progress report:**

1. Profile page is only accessible when user is logged in
2. Profile page has a tab to change nickname
  - a. User will enter their new nickname
  - b. User will type in their current password
  - c. Nickname will be changed if their current password is correct
  - d. Site does not check if new nickname contains no profanity \*backlog\*
3. User is not informed if nickname was changed or any error occurred \*backlog\*

## Delete account

**Story:** As a user, I would like to be able to delete my account from the profile page.

**Acceptance criteria:**

1. Allow access to the profile page if the user is logged in
2. User should have to type in password
3. User has to agree to terms of deletion
4. Account deletion should remove all reviews and comments created by that account
5. User should be brought to home page

**Progress report:**

1. Profile page is only accessible when user is logged in
2. Profile page has a tab to delete profile
  - a. User will enter their current password
  - b. User must agree to terms and conditions of deletion
3. User is informed if they have not agreed to terms and conditions
4. User is not informed if password is incorrect \*backlog\*
5. User is brought to home page

**Password Reset**

**Story:** As a user, I would like to receive an email with a temporary password if I request a password reset.

**Acceptance criteria:**

1. User should be able to click a password reset button on the sign in page
2. User should be able to type in a email address and submit password reset request
  - a. If there the user provided a valid user email address then a temporary password email will be sent to that address
  - a. If the email is invalid the user will be informed to provide a Brock email address
3. The email will contain a temporary password which complies with all the password requirements for the site

**Progress report:**

- Password reset not implemented yet \*backlog\*

**Create Review**

**Story:** As a user, I would like to create a review using an input form for a specific course, instructor or department.

**Acceptance criteria:**

1. Only enable user review form if they are a valid user with a verified email address
2. Only allow review to be submitted if all the necessary parts of the form are filled

**Progress report:**

1. If user is not verified then create review tab states that account must be verified
2. User must fill in all necessary information in form for review to be submitted

## Edit Review

**Story:** As a user, I would like to be able to edit any reviews I've created.

**Acceptance criteria:**

1. There should be an edit button available for any reviews created by the account
2. Clicking on the edit button should allow user to edit their review
3. There should be a "submit" or "republish" button to post the edited review
4. The user should be returned to the page they started and be able to see the updated review

**Progress report:**

1. User must be viewing one of their own reviews in order for the edit button to be visible
2. Only allow review to be re-submitted if all parts of the form still meet the criteria for creating a review

## Report Review

**Story:** As a user, I would like to report reviews that may be inappropriate.

**Acceptance criteria:**

1. There should be an report button next to any reviews
2. Clicking on the report button should allow user to type in reason why they are reporting
3. Report should not allowed to be submitted if it contains profanity
4. Submitting the report should inform user that the report was submitted

**Progress report:**

1. There is a red report button to the bottom right of any review
2. Clicking on the report button allows the user to select what is wrong with the review
3. Submitting the report does not inform user that the report was submitted \*backlog\*

## Delete Review

**Story:** As a user, I would like to be able to delete my own reviews.

**Acceptance criteria:**

1. There should be a delete button next to any of the users reviews
2. Clicking the delete button should ask user if they are sure
3. Deleting the review should remove the review and reload the page

**Progress report:**

- Deleting a review is not implemented yet \*backlog\*

## Create comments

**Story:** As a user, I would like to create a comment under any reviews.

**Acceptance criteria:**

1. There should be a comment button under any review
2. Comments should not be allowed to be submitted if they contain profanity
3. Submitting should display the comment posted

**Progress report:**

1. Users are able to comment after clicking the view comments button
2. They are able to write their comment then click submit
3. Comments are displayed in the view comments window

## Edit Comment

**Story:** As a user, I would like to be able to edit any comments I've posted.

**Acceptance criteria:**

1. There should be an edit button available for any comments by the user
2. Clicking on the edit button should allow user to edit their comment
3. There should be a "submit" button to post the edited comment
4. The user should be returned to the page they started and be able to see the updated comment
5. Edited comments should have a indicator that shows that they have been edited

**Progress report:**

- Edit comments not implemented yet \*backlog\*

## Delete Comment

**Story:** As a user, I would like to be able to delete my own comments.

**Acceptance criteria:**

1. There should be a delete button next to any of the users comments
2. Clicking the delete button should ask user if they are sure
3. Deleting the comment should remove the comment and reload the page

**Progress report:**

- Delete comment not implemented yet \*backlog\*

## View Instructors

**Story:** As a user, I would like to be able to view a list of instructors by clicking on the “instructors” tab on the top menu bar.

**Acceptance criteria:** As a user, when I select the instructor tab, it will take me to the instructor page, which will then have a list of instructors that can be further filtered. It will also have instructor rating rank, rating, department the instructor belongs to and the departments rating.

**Progress report:** Users are able to select the instructors tab and view a list of all instructors. Users are able to view instructors rating rank, ratings as well as their department and department rating rank.

## View Departments

**Story:** As a user, I would like to be able to view a list of departments that the university offers.

**Acceptance criteria:** As a user, when I select the department tab, it will take me to the department page, which will then have a list of departments that can be further filtered. It will also display department rating rank, ratings, the top course, and the top instructor.

**Progress report:** Users are able to select the departments tab and view a list of all departments. Users are able to view department rating rank, ratings as well as top course and top instructor.

## View Courses

**Story:** As a user, I would like to be able to view a list of available courses by clicking on the “Courses” tab on the top menu bar.

**Acceptance criteria:** As a user, when I select the courses tab, it will take me to the courses page, which will then have a list of courses. It will have the course rating rank, course code, course name, rating, and department of the course.

**Progress report:** Users are able to select the courses tab and see a list of all courses. Users are able to see the course rating rank, course code, course name, rating and the department of the course.

## Search

**Story:** As a user, I would like to be able to use a search feature to specifically find the information that I want in an efficient manner.

**Acceptance criteria:** As a user, I will be able to input what I am searching for into the search bar, and the system will return all matching results.

**Progress report:** Users are able to input what they want to search into the search bar. The system returns all matching results. The results can be departments, instructors or courses.

## View Ratings

**Story:** As a user, I would like to view ratings of courses, departments and instructors.

### Acceptance criteria:

1. Users must be able to top rated courses, instructors and departments on the homepage
2. Users must be able to see ratings in the all courses, all instructors and all departments lists
  - a. Users must be able to filter ratings by alphabetical or high to low and vice versa
3. Users must be able to view ratings on individual course, instructor, and department pages

### Progress report:

1. Users are able to view the top 5 rated courses, instructors and departments on their homepage
2. Users are able to see ratings in course, instructors, and departments lists
  - a. Users are able to filter ratings by alphabetical or high to low and vice versa
3. Users can view ratings on individual course, instructor, and department pages



## Thumbs Up / Down

**Story:** As a user, I would like to be able to thumbs up good reviews and comments.

**Acceptance criteria:**

1. There must be a thumbs up and down button next to reviews
2. There must be a number next to the thumbs up and down button indicating the number of thumbs up / down a review has received
3. User must be able to click thumbs up button
  - a. Increase the thumbs number by 1
  - b. Highlight the thumbs up button to indicate that thumbs up has been applied
  - c. If a thumbs down had been previously applied, it will be deactivated and thumbs number will increase by 1
4. User must be able to remove their thumbs up by clicking on an highlighted one
  - a. Decrease the thumbs number by 1
  - b. Remove highlight on thumbs up button to indicate that thumbs up has been removed
5. User must be able to click thumbs down button
  - a. Decrease the thumbs number by 1
  - b. Highlight the thumbs down button to indicate that thumbs down has been applied
  - c. If a thumbs up had been previously applied, it will be deactivated and thumbs number will decrease by 1
6. User must be able to remove their thumbs down by clicking on an highlighted one
  - a. Increase the thumbs number by 1
  - b. Remove highlight on thumbs down button to indicate that thumbs down has been removed

**Progress report:**

- Thumbs up / down has not been implemented yet \*backlog\*

## Admin: Edit Users

**Story:** As an admin, I would like to be able to change user email addresses, change user nicknames, and give users administrative privileges.

**Acceptance criteria:**

1. There must be a tab to edit users
2. There must be a form to change email, nickname or permissions
3. There must be a means to save changes

**Progress report:**

1. Admins can use the custom user tab in Django administration
2. Admins are input emails, nicknames or permissions
3. Admins are able to save changes

## Admin: Add Reviews, Courses, Instructors and Departments

**Story:** As an admin, I would like to be able to add review, course, instructor and department information.

**Acceptance criteria:**

1. There must be tabs for users, courses, instructors and departments
2. There must be a button to add users, courses, instructors and departments

**Progress report:**

1. Admins can use the users, courses, instructors and departments tabs in Django administration
2. Admins can choose any tab: user, courses, instructors and departments and add

## Admin: Edit Reviews, Courses, Instructors and Departments

**Story:** As an admin, I would like to be able to change review, course, instructor and department information.

**Acceptance criteria:**

1. There must be a tab to edit reviews, courses, instructors and departments
2. There must be a form to edit their respective information
3. There must be a means to save changes

**Progress report:**

1. Admins can use the reviews, courses, instructors and departments tabs in Django administration
2. Admins can input any changes they wish to make
3. Admins are able to save changes

## Admin: Delete Reviews, Courses, Instructors and Departments

**Story:** As an admin, I would like to be able to delete review, course, instructor and department information.

**Acceptance criteria:**

1. There must be tabs for reviews, courses, instructors and departments
2. There must be a checkbox to select and an action to delete

**Progress report:**

1. Admins can use the reviews, courses, instructors and departments tabs in Django administration
2. Admins can choose any of the reviews, courses, instructors and departments and delete them

# Updated Requirements

## 1.1 Purpose

COSC 4P02 R8Scholar Requirements Elicitation:

Stakeholders:

- Us (Developers)
- Professor Naser
- Users (brock students / people with a brocku email) - Anonymous
- Testers - other brock students
- Potentially the people being rated on the site

Expectation of the system:

- Online platform to rate scholarly aspects of Brock University
- Anyone should be able to view all ratings on the site
- Verified brock students should be able to leave ratings on the site
  - Signup / Login
    - Verify Brock Email
- Needs to display a page for each department
  - Departments can list staff and courses
    - All of these are dynamically generated, based off database entries
  - Search feature on front page

Use Cases:

- Create a review for professors, courses, departments, other
  - Comments
  - Rating
  - Option to add tags about professor characteristics
    - Hard marker
    - Timely
    - Lots of Homework
- Reply to Review
- Create new account
- Sign in to student account
- Students should be able to search up professors or courses
- General forum thread for each topic of discussion
- Viewing professor profile page
- Add Professor/Course/other to site
- Modify Professor/Courses/Other
- Search Feature

#### System Architecture:

- Django based
- Model - Database structure
- View - Site control
- Template - HTML templating

#### Risk Factors of the system:

- Leaked information
  - industry standard password encryption
  - AWS has security features built into the servers they host
- Site crashing
- Unprofessional comments
  - Run comments through a profanity filter before posting
  - Report function

## 2.1 Functional Requirements

### 2.1.1 Sign Up

Use Case Name	Sign Up / Create Profile
Trigger	The user selects the “sign up” on the website
Precondition	The user has not created a profile for their email address
Basic Path	<ol style="list-style-type: none"> <li>1. The system opens a form to enter information</li> <li>2. The user will enter the correct format of data into the form</li> <li>3. The system confirms the information with the database and the user</li> <li>4. The profile is created and the user redirected to the profile page</li> </ol>
Alternative Path	None noted
Postcondition	The database has been updated
Exception Paths	If the user has already created a Profile using that given email address then the use case abandons
Other (Notes)	This can only be used for each email address once

### 2.1.2 Log In

Use Case Name	Log in to profile
Trigger	The user selects the “log in” on the website
Precondition	The user has created a profile previously using a defined email address
Basic Path	<ol style="list-style-type: none"> <li>1. The system opens a log in dialogue</li> <li>2. The user inputs the correct format of data (text)</li> <li>3. The database authenticates whether that user is allowed access</li> <li>4. The user is redirected to the</li> </ol>
Alternative Path	There is the possibility of being directed to the login via a direct link
Postcondition	The user is redirected and the log are updated
Exception Paths	If the user inputs invalid information there will be an error and it will allow you to re-enter the text.
Other (Notes)	

### 2.1.3 Log Out

Use Case Name	Log Out
Trigger	The user selects the log out option on the website
Precondition	The user was already logged in and accepted into the database.
Basic Path	<ol style="list-style-type: none"> <li>1. The system prompts the user to make sure they want to log out</li> <li>2. The system will redirect the user to the homepage of the website</li> </ol>
Alternative Path	None noted so far
Postcondition	The user is no longer authorised within the database.
Exception Paths	The user decides against logging out of the website and thus the use case is abandoned.
Other (Notes)	

### 2.1.4 Edit Profile

Use Case Name	Edit Profile
Trigger	The user selects the Edit feature within the profile.
Precondition	The user must be authorised, be viewing the profile page they created.
Basic Path	<ol style="list-style-type: none"> <li>1. The profile information form is presented in an editable view</li> <li>2. The user edits the desired fields in order make changes</li> <li>3. The system confirms before making any changes to the profile.</li> </ol>
Alternative Path	None noted
Postcondition	The database is updated with the new information from the form.
Exception Paths	If the user updates the information with invalid content it will prompt them to fix it. If the user cancels the changes, there are no new changes and the use case is abandoned.
Other (Notes)	

### 2.1.5 Create Review

Use Case Name	Create Review
Trigger	The selects the review function on the page
Precondition	The user must be authenticated in order to be able to rate an entity.
Basic Path	<ol style="list-style-type: none"> <li>1. The user selects their numeric rating of the entity in question</li> <li>2. The system waits for the user to enter the review text and press the confirmation</li> <li>3. The system prompts the user to confirm the score they have put in and the input text.</li> <li>4. The system shows the user the review they just created</li> </ol>
Alternative Path	The user may opt to not input any text in which case the system will confirm that, then placing a placeholder in the text field(s).
Postcondition	The database is updated and thus calculates the new average score for said entity.
Exception Paths	If the user does not confirm their entry the system will abandon the rating process and revert the database to the previous state.
Other (Notes)	The username of the user is optional to the publishing of a review.

### 2.1.6 Edit Review

Use Case Name	Edit Review
Trigger	The user selects the edit function on the review.
Precondition	The user must be authorised as well as being a review that is created by that user.
Basic Path	<ol style="list-style-type: none"> <li>1. The system makes sure the input is of the correct type (text for a review or numeric for ratings)</li> <li>2. The system updates the entities profile</li> <li>3. The system shows the user the newly edited review</li> </ol>
Alternative Path	None noted
Postcondition	The database is updated with the new information we get from the review section.
Exception Paths	If the user chooses not to publish said edit the system will leave the editing view and the use case abandoned
Other (Notes)	This also includes the ability to remove a review

### 2.1.7 Search For Entity

Use Case Name	Search For Entity
Trigger	The user selects the search field in the website
Precondition	There is a non-empty database of entities and thus can be queried
Basic Path	<ol style="list-style-type: none"> <li>1. The system reads the input from the user</li> <li>2. The database queries for the given conditions of the search parameter</li> <li>3. The results (entities) are displayed by the system</li> <li>4. The user picks an entity from that list</li> <li>5. The system redirects the user to the corresponding profile</li> </ol>
Alternative Path	None noted
Postcondition	None noted
Exception Paths	If the user inputs a search term that does not correspond to anything we have stored in the database, the user will be prompted to change the search term

### 2.1.8 Write Comment

Use Case Name	Write Comment
Trigger	The user selects the comment section below a review that has been posted
Precondition	The user must be authorised and there must exist a comment to use.
Basic Path	<ol style="list-style-type: none"> <li>1. The system awaits the input from the user of correct type (text)</li> <li>2. The system confirms if that text is final</li> <li>3. The system then shows the user the comment they made as well as the original review</li> </ol>
Alternative Path	None noted
Postcondition	The database is updated with the new comment to the given review
Exception Paths	If the user chooses to cancel the comment the system should remove the text and revert to the previous view
Other (Notes)	Username optional.

### 2.1.9 Edit Comment

Use Case Name	Edit Comment
Trigger	The user selects the 'edit'
Precondition	The user must be logged in and there must be a previously made comment by the same user
Basic Path	<ol style="list-style-type: none"> <li>1. The system will wait for the user to enter the text</li> <li>2. The system will confirm the text with the user</li> <li>3. The comment text will be published replacing the old with the new text</li> </ol>
Alternative Path	None noted
Postcondition	The database is updated, changing the previously saved text with the new text.
Exception Paths	If the user wants to, there is the ability to cancel the edit, leaving the old text unchanged in the database.
Other (Notes)	Username optional.



### 2.1.10 View Entity By Category

Use Case Name	View Entity by Category
Trigger	The user selects the option on the website to view “all” the members of some category.
Precondition	The category of entity (Professor, Course etc) should not be empty within the database
Basic Path	<ol style="list-style-type: none"> <li>1. The system waits for the user to select the given category of entity.</li> <li>2. The system will query the database for the given data type</li> <li>3. The system displays the entities in that list</li> <li>4. The system waits for further action from the user</li> </ol>
Alternative Path	None noted
Postcondition	None noted
Exception Paths	If there is nothing within the database that matches the given category the use case should abandon and redirect the user back to the previous page (use case).

### 2.1.11 Contact System Administrator

Use Case Name	Contact System Administrator
Trigger	The user selects the “contact us” section of the website.
Precondition	There are no pre-conditions
Basic Path	<ol style="list-style-type: none"> <li>1. The system presents the user with a form to contact the administrators of the system</li> <li>2. There are text fields for the system to accept inputs from the user</li> <li>3. The system asks the user to confirm the message they are looking to send</li> <li>4. The message is sent to the administrators, who can then respond to any issues raised</li> </ol>
Alternative Path	The user may also find the form through an external hyperlink provided in some other medium i.e. email
Postcondition	There will be an update to users who are already authenticated, there will be an update to the database with the details of the contact the user has had with the administrators.
Exception Paths	The user can cancel any form of communication and not send through the message, in this case the use-case is abandoned and there are no postconditions for registered users.

### 2.1.12 Create Forum Post

Use Case Name	Create Forum Post
Trigger	The user selects the community section on an entity's profile
Precondition	The user should be authorised in order to gain access to the community section of the profile
Basic Path	<ol style="list-style-type: none"> <li>1. The system waits for the user to input the text into the edit field</li> <li>2. The system asks the user to confirm their message for the forum post</li> <li>3. The system publishes the message and then allows for comments and interaction on the forum posts.</li> </ol>
Alternative Path	None noted
Postcondition	The database is updated with all the information within the forum post and stored
Exception Paths	If the user cancels their intention of posting to the forum, the use case is abandoned and there are no postconditions
Other (Notes)	Username optional.

### 2.1.13 Comment on Forum Post

Use Case Name	Comment on Forum Post
Trigger	The user selects the comment function on a previously posted forum post
Precondition	The user must be authenticated within the system and there should also be a forum post already made.
Basic Path	<ol style="list-style-type: none"> <li>1. The system waits for the user to input the text into the edit field</li> <li>2. The system asks the user to confirm their comment for that forum post</li> <li>3. The system publishes the message and then allows for comments and interaction on the forum comment.</li> </ol>
Postcondition	The database is updated with the messages written into the comments on that individual
Exception Paths	If the user cancels their intention to comment on the forum post there should be an abandonment of intention and no post conditions
Other (Notes)	Username optional.

### 2.1.14 Respond to Review

Use Case Name	Respond to Review
Trigger	The Entity selects the reply function on the reviews on the profile.
Precondition	The entity should be authorised and verified to be the responsible party for the review
Basic Path	<ol style="list-style-type: none"> <li>1. The system waits for the user to input the text into the edit field</li> <li>2. The system asks the entity to confirm their comment for that forum post</li> <li>3. The system publishes the message to the review</li> </ol>
Alternative Path	None noted
Postcondition	The database is updated with the message added to the review.
Exception Paths	If the entity decides to cancel their intention of replying to a comment the use case is abandoned.
Other (Notes)	

### 2.1.15 View All Reviews

Use Case Name	View All Reviews
Trigger	The entity can go to the page on which you can view all the reviews and ratings
Precondition	The entity should be authorised and verified to be the responsible party for the review
Basic Path	<ol style="list-style-type: none"> <li>1. The system displays a section with all the ratings of that entity</li> <li>2. The system displays a section with all the review messages written about that entity</li> </ol>
Alternative Path	The entity may view all the ratings and reviews by clicking on their score on the profile.
Postcondition	None noted
Exception Paths	If the entity has no ratings and reviews published at that time, the system should notify them of that and abandon the use case.
Other (Notes)	

### 2.1.16 Search Through Reviews

Use Case Name	Search Through Reviews
Trigger	The entity selects the search feature on website
Precondition	The entity is already authorised and is on the page where they view all the reviews
Basic Path	<ol style="list-style-type: none"> <li>1. The entity enters some form of keyword into the search field</li> <li>2. The system queries the database for that given keyword</li> <li>3. The system presents the results of the search, listing reviews that match the search term</li> </ol>
Alternative Path	None noted
Postcondition	None noted
Exception Paths	If there is nothing matching the search term, the system presents an exception for the user.
Other (Notes)	

### 2.1.17 Voting System

Use Case Name	Voting system (Upvote/Downvote)
Trigger	The user selects the vote they desire on the website post.
Precondition	The user is authorised within the system
Basic Path	<ol style="list-style-type: none"> <li>1. The system awaits the users selection of a vote</li> <li>2. The system calculates the new tally (upvote - downvote)</li> <li>3. The system displays the new tally to the user</li> <li>4. The system indicates which vote is selected and goes back to waiting</li> </ol>
Alternative Path	The user may vote directly from an external link to the post (possibly embedded)
Postcondition	The database is updated with the current tally of the votes
Exception Paths	The user may double click a vote and thus abandoning the use case and reverting all tallies.
Other (Notes)	

### 2.1.18 Report Post

Use Case Name	Report post
Trigger	The user selects the functionality to report
Precondition	The user must be authorised within the system
Basic Path	<ol style="list-style-type: none"> <li>1. The system presents the user with a form for further information</li> <li>2. The system takes the input and asks the user to confirm their report</li> <li>3. The system posts the Report to the Moderator-level users</li> <li>4. The system sends the user a message to confirm the report's status</li> </ol>
Alternative Path	The user uses a direct link to reporting a post on the website
Postcondition	The database is updated with the information contained within the report.
Exception Paths	The user can cancel the intention to report a post and thus the use case is abandoned and the changes reverted.

### 2.1.19 Remove Review

Use Case Name	Remove Review
Trigger	The moderator selects the remove function within the moderation tools on the review
Precondition	The user MUST be of Moderator-level permission within the system or greater.
Basic Path	<ol style="list-style-type: none"> <li>1. The system presents the moderator with a form for post removals</li> <li>2. The system takes the input of the moderator and stores it</li> <li>3. The system confirms with the moderator if they wish to proceed</li> <li>4. The system removes the review and the profiles are refreshed (all calculations)</li> <li>5. The system logs the removal and sends the reporting user the form from the moderator</li> </ol>
Alternative Path	None noted
Postcondition	The database recalculates all the relevant data and updates
Exception Paths	The moderator can cancel their intent to remove the review and the use case is abandoned and there is no change to the database

### 2.1.20 Remove Comment

Use Case Name	Remove Comment
Trigger	The moderator selects the remove function within the moderation tools on the comment
Precondition	The user MUST be of Moderator-level permission within the system or greater.
Basic Path	<ol style="list-style-type: none"> <li>1. The system presents the moderator with a form for post removals</li> <li>2. The system takes the input of the moderator and stores it</li> <li>3. The system confirms with the moderator if they wish to proceed</li> <li>4. The system removes the comment and the profiles are refreshed</li> <li>5. The system logs the removal and sends the reporting user the form from the moderator</li> </ol>
Alternative Path	None noted
Postcondition	The database updates
Exception Paths	The moderator can cancel their intent to remove the comment and the use case is abandoned and there is no change to the database

### 2.1.21 Remove Topic

Use Case Name	Remove Topic
Trigger	The moderator selects the remove function within the moderation tools on the topic
Precondition	The user MUST be of Moderator-level permission within the system or greater.
Basic Path	<ol style="list-style-type: none"> <li>1. The system presents the moderator with a form for post removals</li> <li>2. The system takes the input of the moderator and stores it</li> <li>3. The system confirms with the moderator if they wish to proceed</li> <li>4. The system removes the topic and the profiles are refreshed</li> <li>5. The system logs the removal and sends the reporting user the form from the moderator</li> </ol>
Alternative Path	None noted
Postcondition	The database updates
Exception Paths	The moderator can cancel their intent to remove the topic and the use case is abandoned and there is no change to the database

### 2.1.22 Change Visibility

Use Case Name	Change Visibility
Trigger	The moderator selects the visibility function within the moderation tools on the post
Precondition	The user MUST be of Moderator-level permission within the system.
Basic Path	<ol style="list-style-type: none"> <li>1. The system presents the moderator with a form for post visibility changes</li> <li>2. The system takes the input of the moderator and stores it</li> <li>3. The system confirms with the moderator if they wish to proceed</li> <li>4. The system changes the visibility the post and the profiles are refreshed</li> <li>5. The system logs the change of visibility</li> </ol>
Postcondition	The database updates the status of the post
Exception Paths	The moderator can cancel their intent to remove the comment and the use case is abandoned and there is no change to the database

### 2.1.23 Ban User

Use Case Name	Ban User
Trigger	The Administrator will select the ban function within the Administrator tools on the user's profile
Precondition	The Administrator has the appropriate level of permissions
Basic Path	<ol style="list-style-type: none"> <li>1. The system presents a form for the admin</li> <li>2. The system awaits the confirmation of the input</li> <li>3. The system confirms with the admin that they would like to remove the user</li> <li>4. The user is removed and the change logged.</li> <li>5. The user is notified by the system via a message</li> </ol>
Alternative Path	None noted
Postcondition	<p>The database is updated for the removal of the user and all scores across their reviews recalculated.</p> <p>This also results in the email address used in that user profile being blacklisted from the system.</p>
Exception Paths	The admin may cancel their intent to ban the user from the system and then the use case is abandoned and the database unchanged.

### 2.1.24 Add Entity

Use Case Name	Add Entity (ReviewItem)
Trigger	The Administrator selects the feature to create a new entity on the administrator control panel
Precondition	The Administrator has the appropriate level of permissions within the system
Basic Path	<ol style="list-style-type: none"> <li>1. The admin is presented with a form by the system</li> <li>2. The system waits for the input of the fields necessary for that entity</li> <li>3. The system asks the admin to confirm whether they would like to add the entity as is.</li> <li>4. The system adds the entity to the list and logs the addition</li> </ol>
Alternative Path	None noted
Postcondition	The database is updated with the new entity and its attributes and data.
Exception Paths	The admin can cancel their intent to create a new entity in the system. The use case is then abandoned and the database is unchanged.

### 2.1.25 Remove Entity

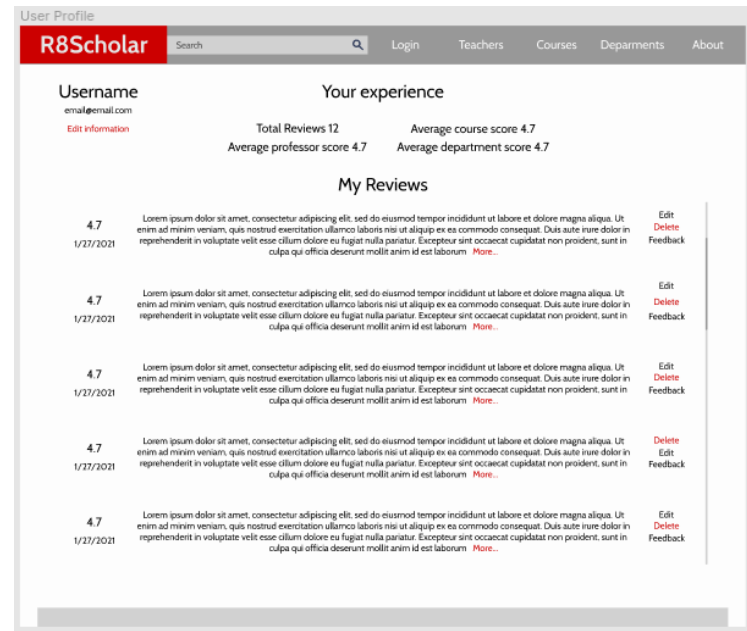
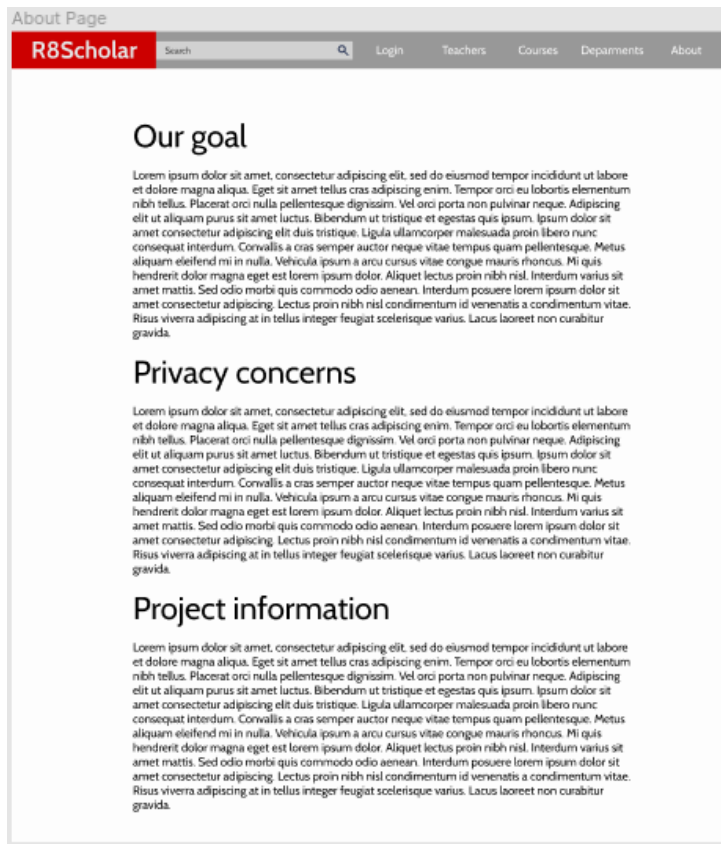
Use Case Name	Remove Entity (ReviewItem)
Trigger	The Administrator selects the remove entity function on the entity's profile.
Precondition	The Administrator has the appropriate level of permissions within the system
Basic Path	<ol style="list-style-type: none"> <li>1. The system present the admin with form for entity removals</li> <li>2. The system confirms that the user would like to remove the entity</li> <li>3. The system removes the entity from the system</li> <li>4. The removal is logged within the system.</li> </ol>
Alternative Path	None noted
Postcondition	The database is updated after the removal of the entity.
Exception Paths	The admin can cancel their intention to remove an entity



## 2.2 User Interface Requirements

- UI must deliver system functions to users in a streamlined and easy-to-use way
  - Options to navigate to each page allowed by their user type
    - User
      - Login
      - Signup
      - Courses
      - Instructors
      - Departments
      - Home
      - About
    - Moderator
      - Moderation Pages
    - Admin
      - Admin menu
- Each page to display navigation and a footer
  - Settings
  - Signout
  - Display courses, departments, instructors
  - Navigation bar with links to each page
    - Navigation: Signed in
      - Display account drop down menu
  - Links to account
    - Display home, about, login, signup
  - Footer: Github link Other Info that would go in a footer such as a contact email or link to about page
- Home page
  - Displays to each of courses, departments, instructors (top ones)
  - Statistics
  - Information
- About page
  - Shows information about the project, the developers and FAQ
- Sign In / Sign up
  - Forms to complete the required actions
  - Login: Link to signup
- User Profile:
  - Show the user's email
  - Give option to edit profile information
  - Show statistics on reviews left by that user such as their average rating among other things
  - In a list, show all reviews left by that user alongside buttons which allow for editing deleting and viewing comments on them

- Courses
  - Display all courses in a list
  - Option to create a review
  - Filter by Course
  - Option to Search Courses
  - Courses/course\_name
    - Show all reviews for course\_name in a list
    - Department which course belongs to
    - Clicking on review shows the in depth review
      - Displays comments
    - Average Rating
- Departments and Instructors should have the same UI design as courses
- Creating Reviews
  - Large text box to write content
  - Boxes to input score out of 5 on predefined criteria
    - Lectures
    - Feedback
    - Organizational Skills
- Visual Example of UI Design



## 2.4 Stakeholder Requirements

- Minimal ads.
- Professional visuals.
- Searchable using names, course names, departments with autocomplete.
- Contact to get reviews removed.
- Be able to rate staff, and departments. May be able to rate TAs as well.
- Review students that professors worked with in Project Courses.

## 2.5 Non-Functional Requirements

### 2.5.1 Product Requirements

Requirements which specify that the delivered product must behave in a particular way.

#### Usability

- There should be an option for colourblind or other colour scheme themes in order to have the site be usable by as many different types of users to be given access.

#### Efficiency

- The search should be efficient (auto completing or querying as they type).
- Any course can only have one course page.

#### Security

- The authentication is only for brocku.ca domain accounts and accounts should be validated as real Brock accounts
  - User account information should be handled and stored securely(i.e hash passwords before storing)
- All emails are stored within R8Scholar.
- All passwords are hashed.

#### Dependability

- Only one account for each email address.

### 2.5.2 Organizational Requirements

Requirements which are a consequence of organizational policies and procedures e.g. process standards used, implementation requirements, etc.

#### Operational

- Site must update courses and professors as they are added
- Must have the ability for a moderator to remove comments
- Must update displayed statistics each time a review is made

#### Developmental

- Ability to add entities to database
- Remove entities
- Must look professional

### 2.5.3 External Requirements

Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

#### Regulatory

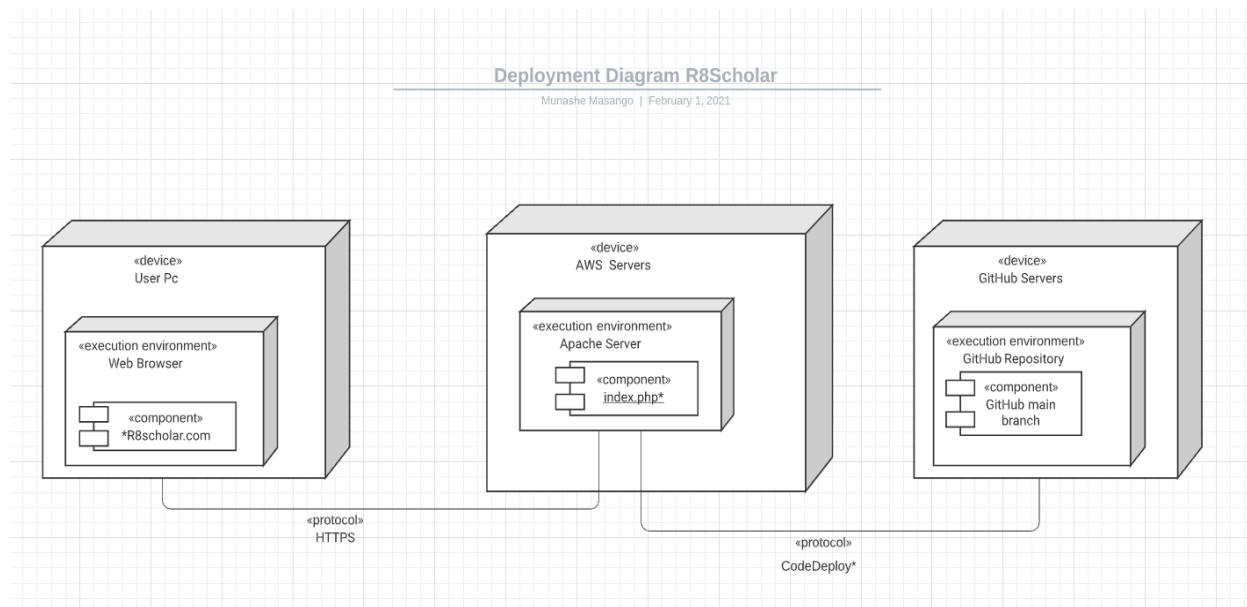
- All information on site comes from publicly available data.
- Any comments/reviews on site are not reflective of R8Scholar or it's teams opinions.

#### Ethical

- All software used is Open Source and used as allowed by their respective licenses.
- R8Scholar reserves the right to remove any posts it deems unethical or abusive.

## 3.0 Architectural Design

### 3.1.1 Deployment Diagram



### 3.1.2 Deployment Explanation

#### User PC

- Web Browser
  - The Web Browser is the primary interfacing method for the users of the product specifically. There is a connection from the users' computer to the AWS servers via HTTP(S) protocols. This connection with the server will then be further secured within the server using the user's authentication details (which they specify and should remember).
- R8Scholar website
  - Is an artifact that represents the instantiation of the website for the user. This is presented via web browser given that the browser supports all the requisite technologies.

#### AWS Servers

- Apache Server
  - Barebones Linux server supported by Amazon Web Services. The Apache technologies involved include those that AWS supports (like Amazon Keyspaces). The reasoning for using AWS and Apache is that there are advantages in performance when scaled. This website is ideally going to be used by many people in the future, this allows us to grow the tables in the required databases. Other useful features include being highly available and secure while

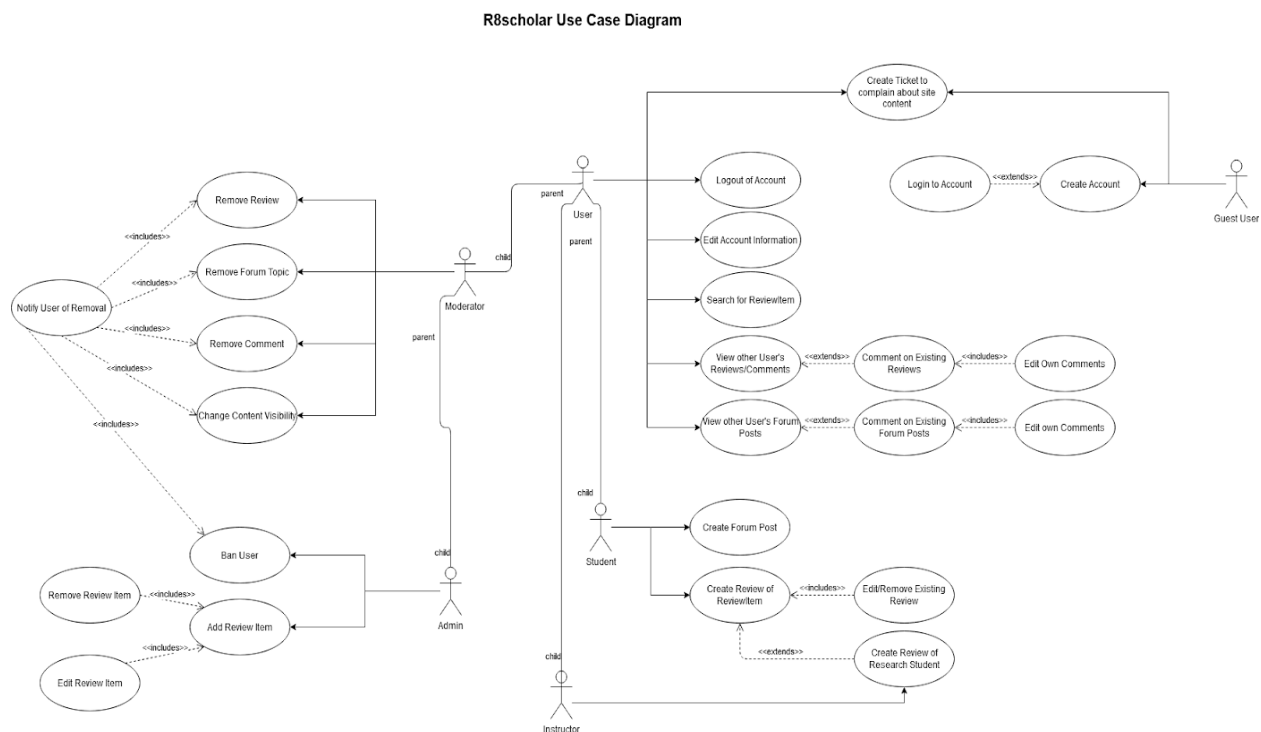
still allowing the development team to minimise costs by using open-source softwares.

- Database files (index.php\*) - is an example file that demonstrates that the Apache server can access instances of files within the database to present as web pages to the user, via the connection method above.

### GitHub Servers

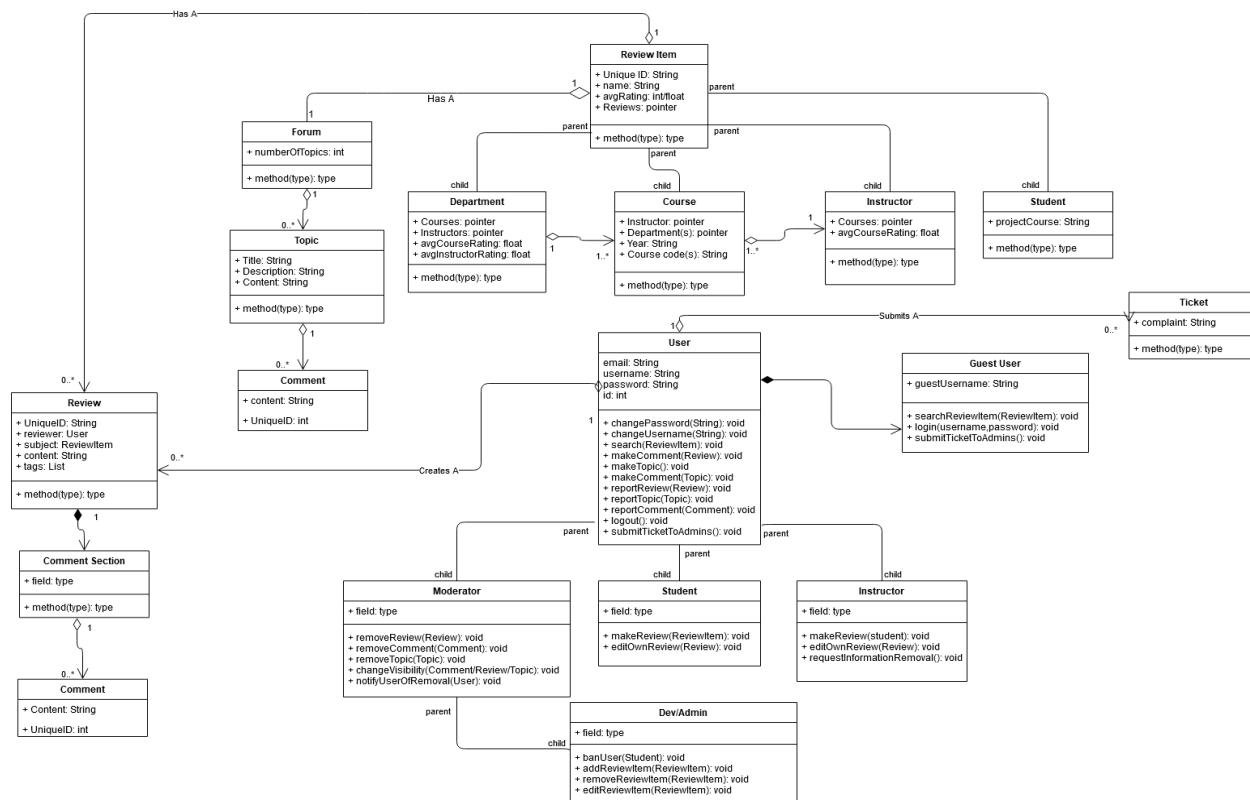
- Repositories
  - The GitHub repository is used for code version control within the product life cycle the AWS server is connected to the GitHub Servers via technologies supported by the server and the like (CodeDeploy). The repository
- Main Branch
  - The main branch will be the instance of the current (or latest deployment) version of the website. The AWS server will pull the versions of the code from this branch. The use of this method for version control and updating the server are useful as they provide some key features to the developers of the product. The developer is able to minimise downtime on the website, centralise any control and collaborate with a group effectively, while maintaining the integrity of the data.

## 3.2 Use Case Diagram(s)



### 3.3 Class Diagram(s)

R8scholar Class Diagram

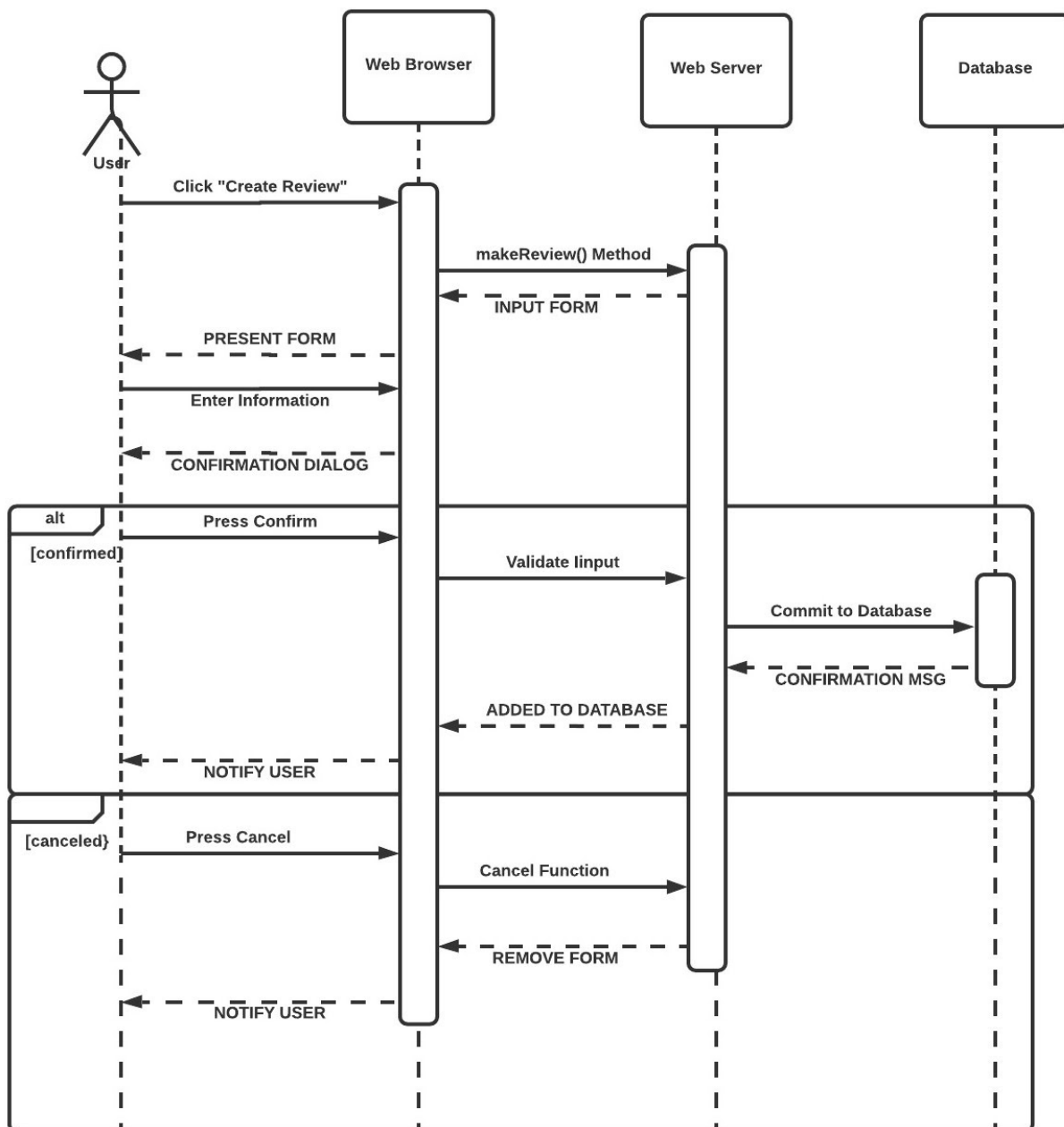


## 3.3 Sequence Diagram(s)

### 3.3.1 Create Review

#### Sequence Diagram - Create Review

Munashe Masango | February 4, 2021

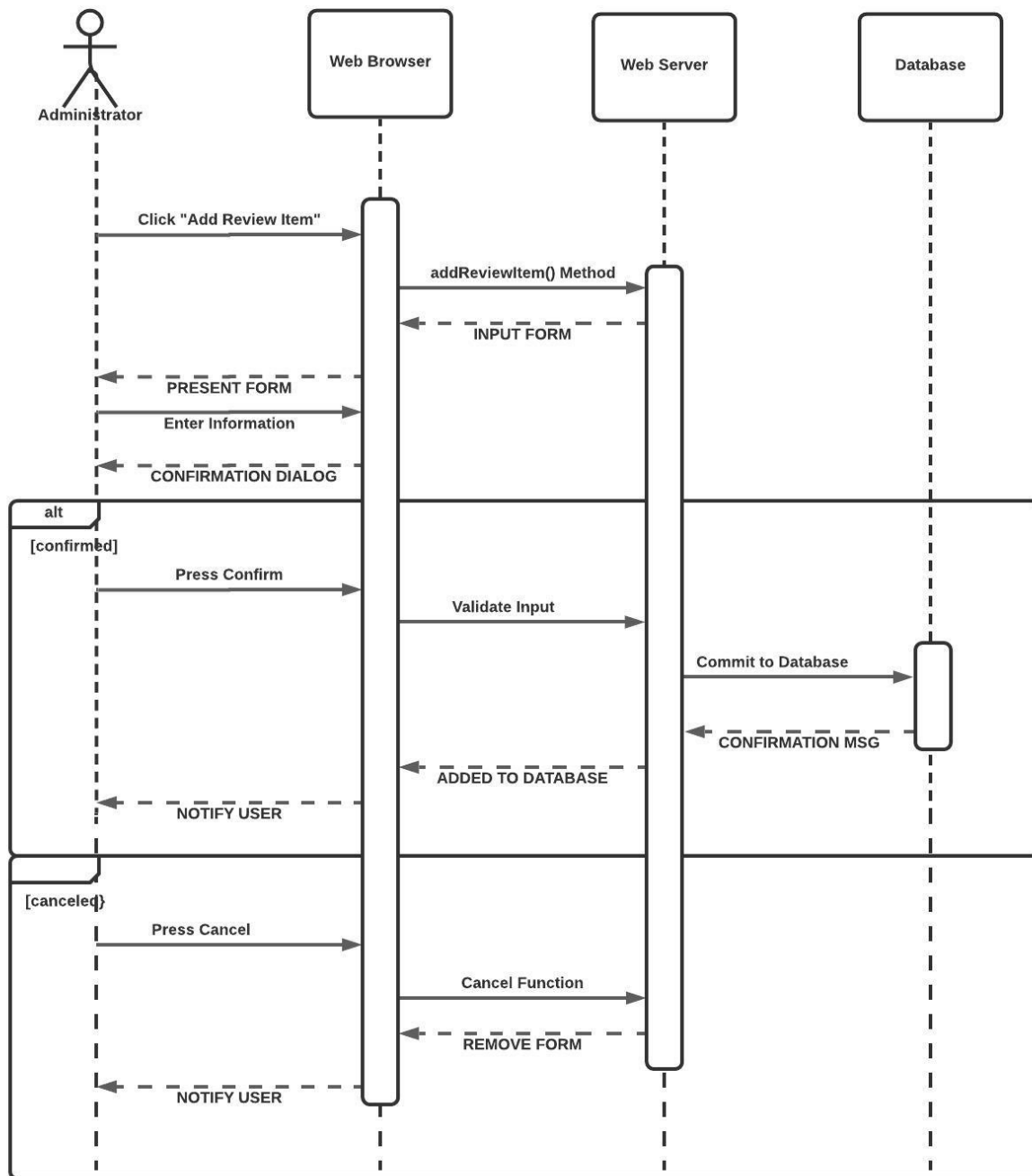




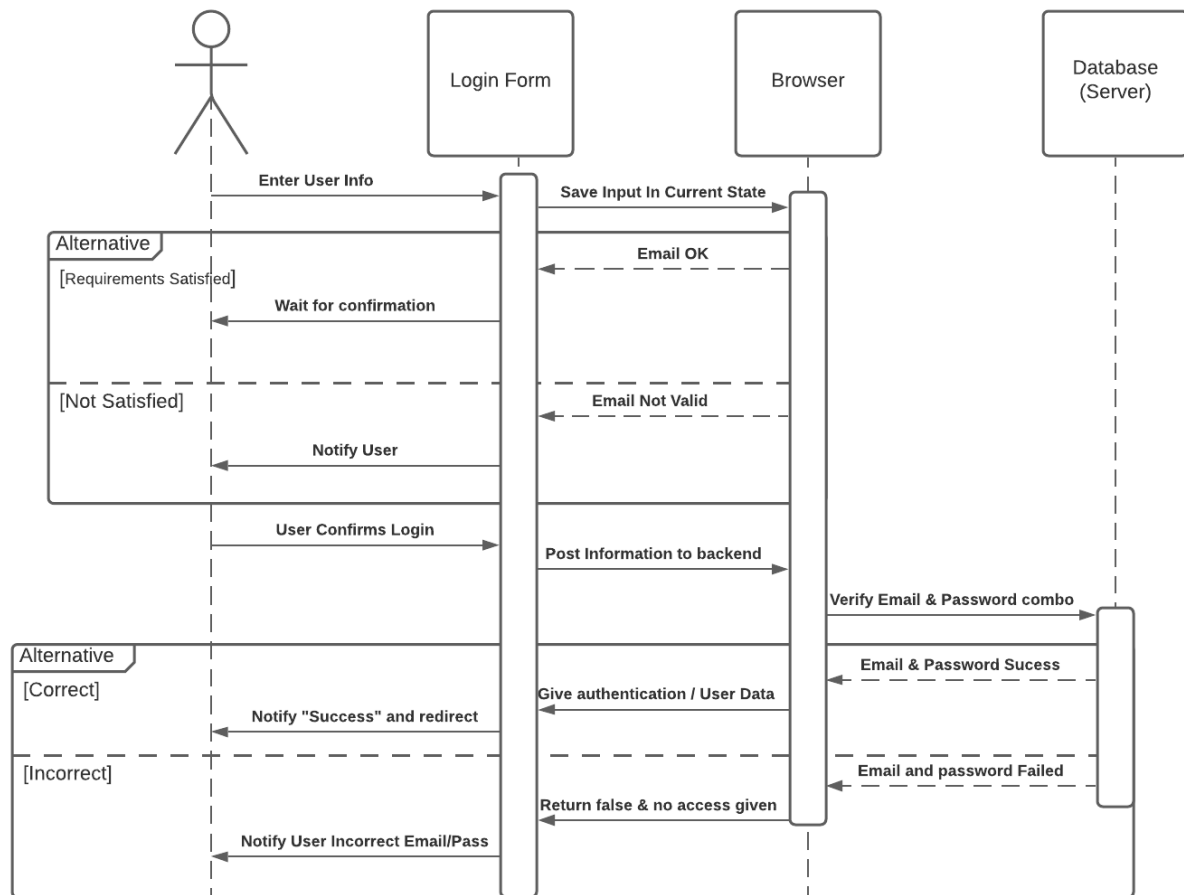
## 3.3.2 Add Review Item

Sequence Diagram - Add Entity

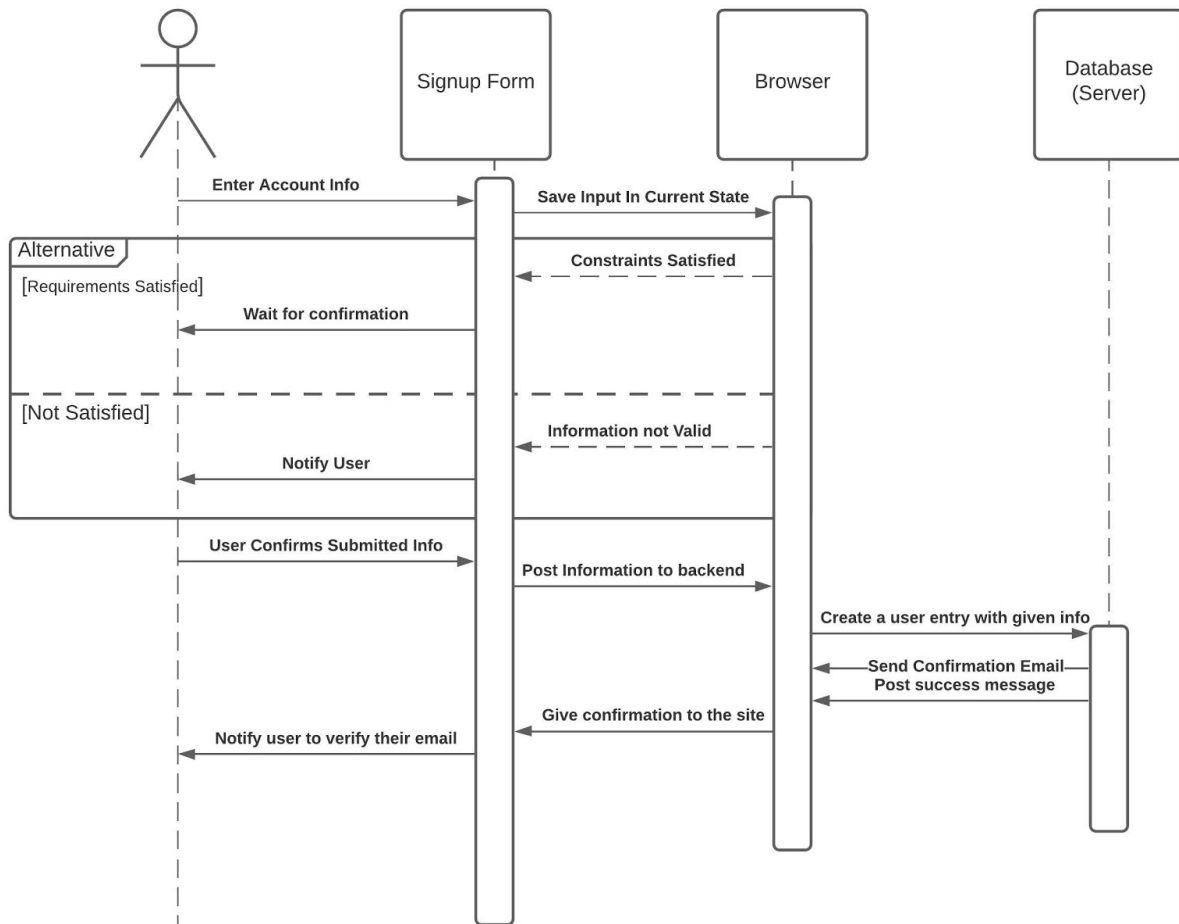
Munashe Masango | February 4, 2021



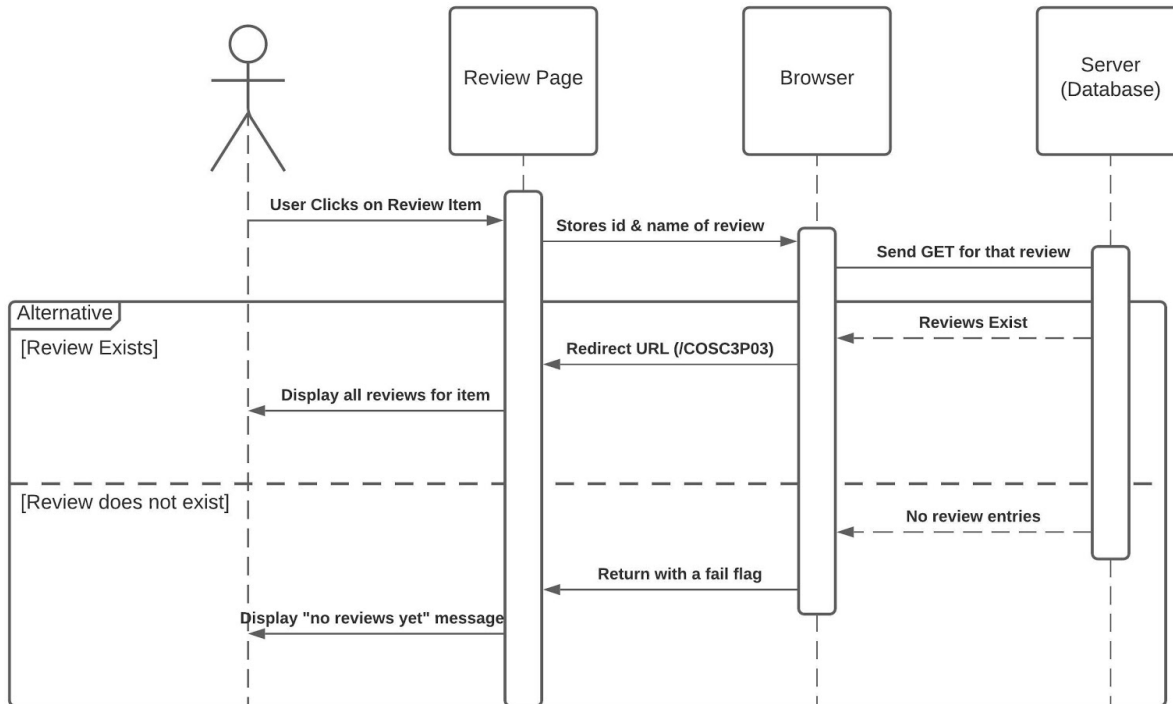
### 3.3.3 Login



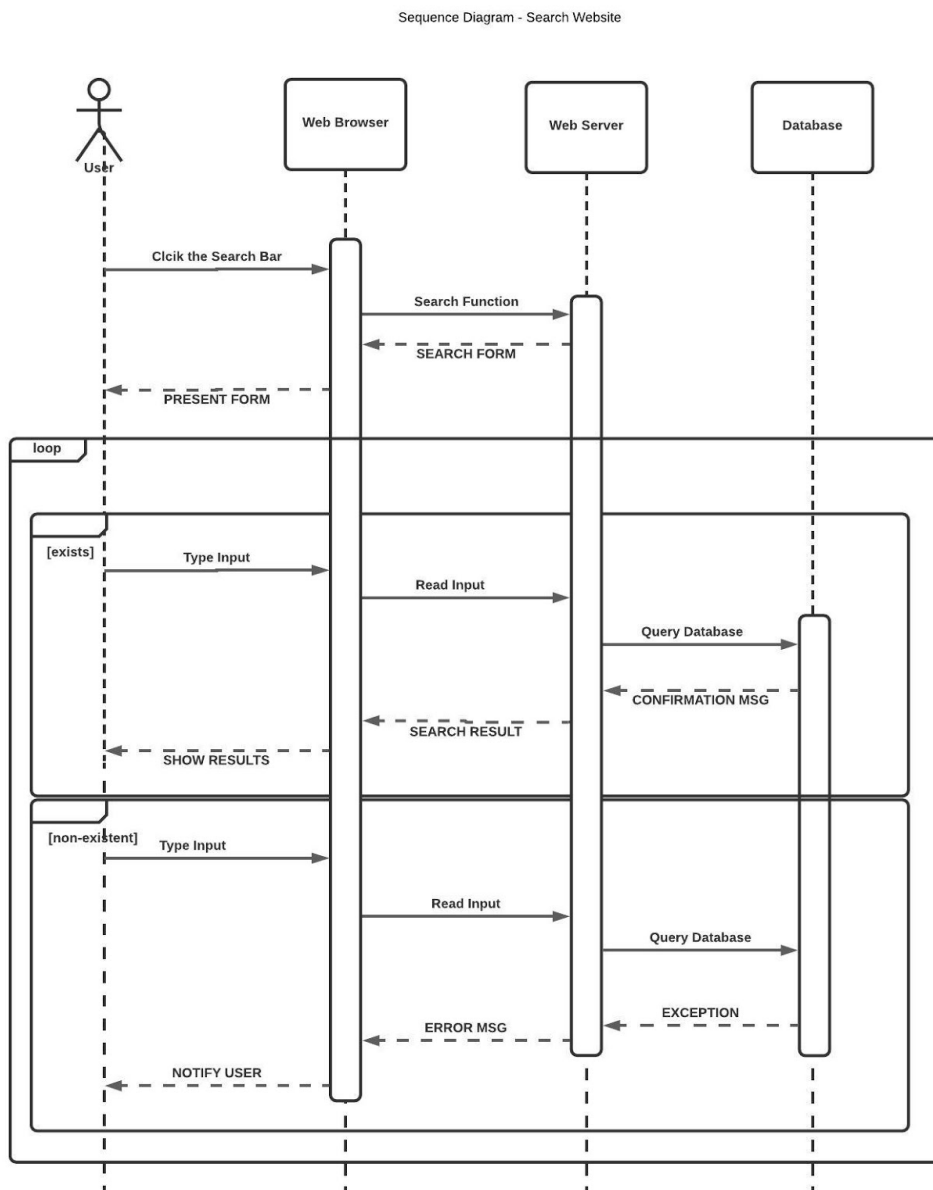
### 3.3.4 Signup



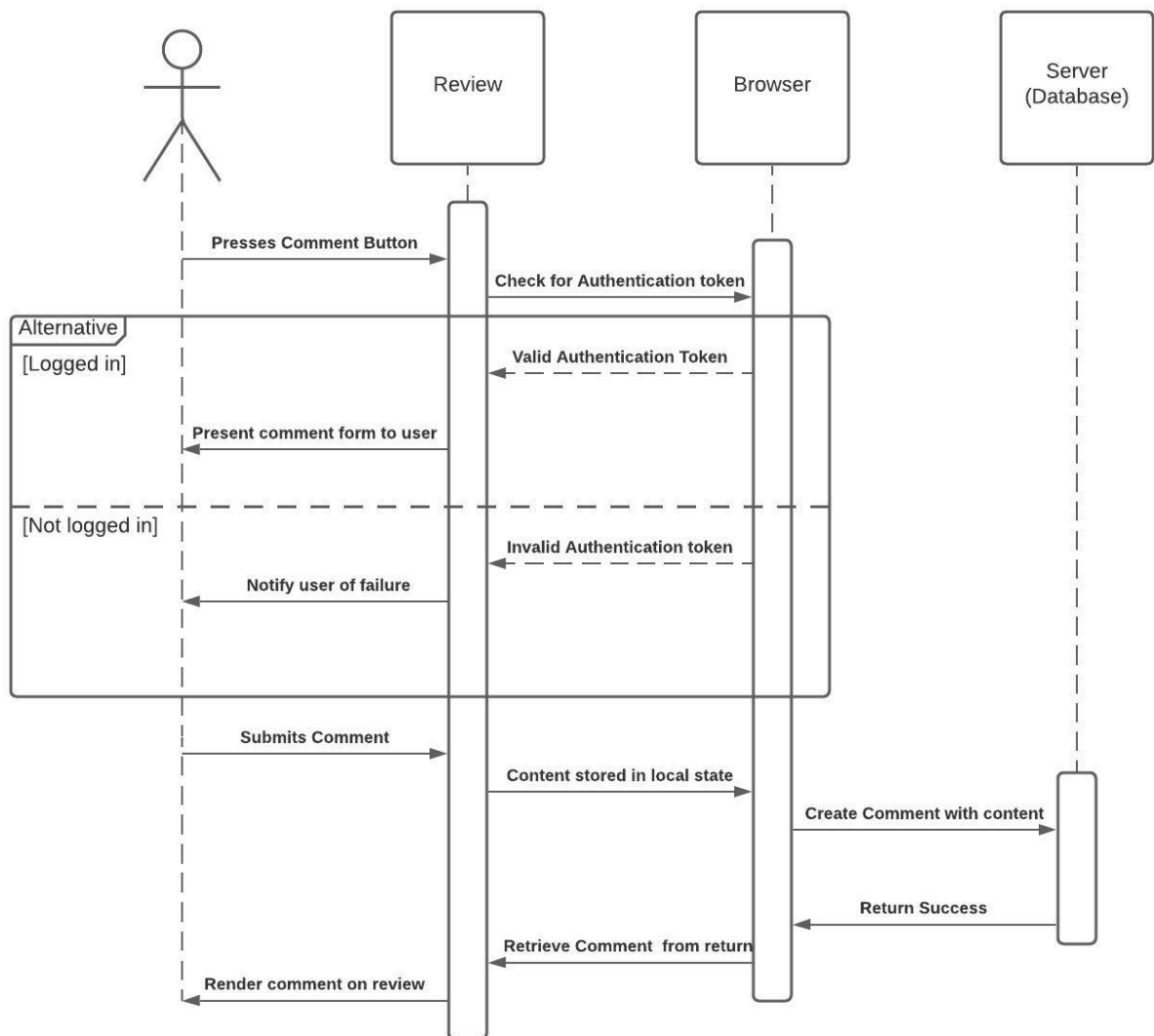
### 3.3.5 Selecting Review Item



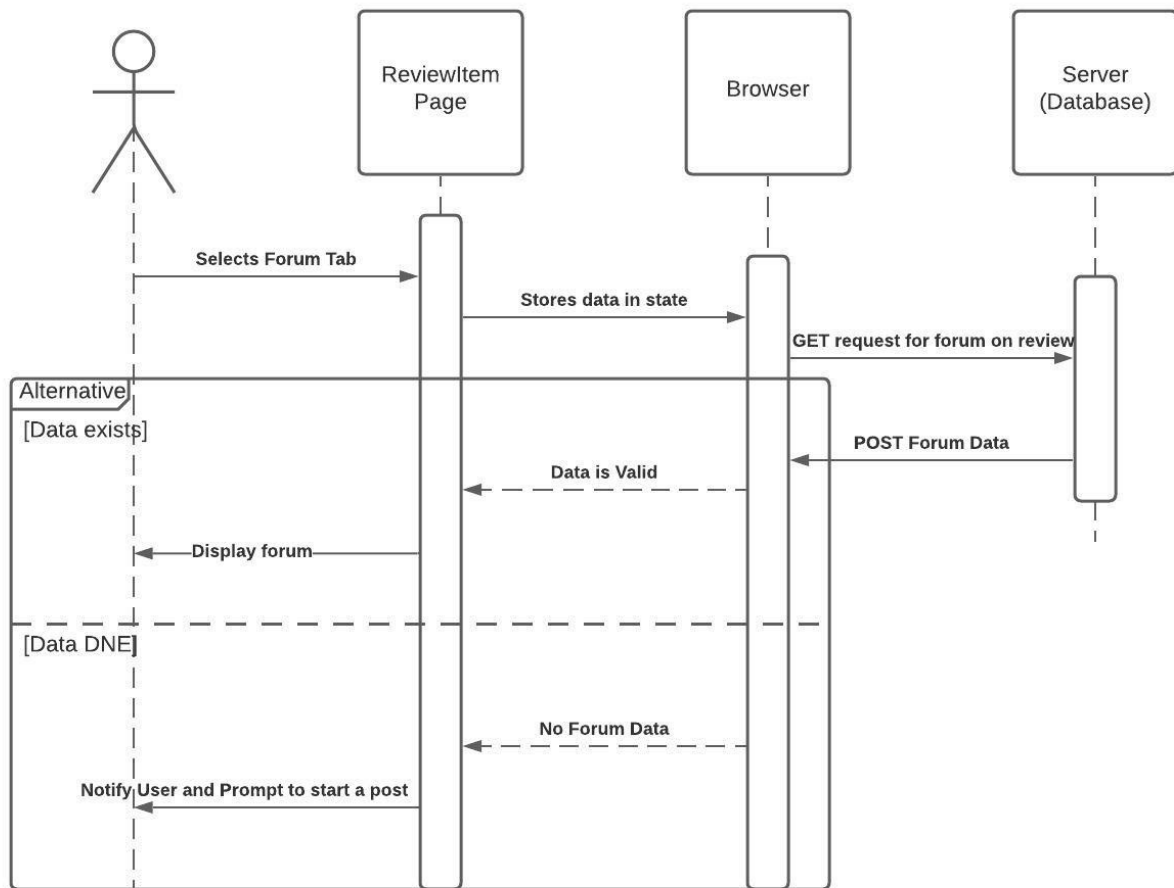
### 3.3.6 Search Website



## 3.3.7 Comment on Review



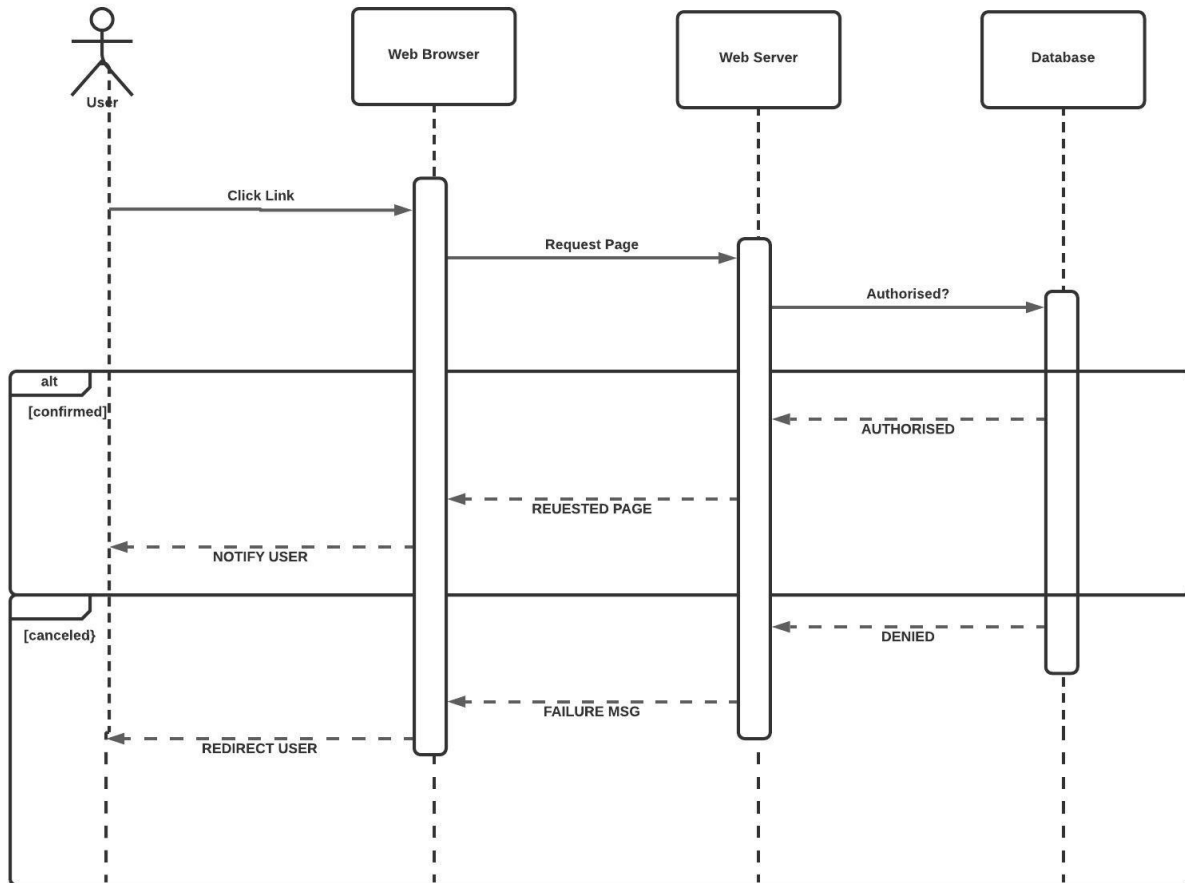
### 3.3.8 Accessing Forum



### 3.3.9 Navigation

#### Sequence Diagram - Website Navigation

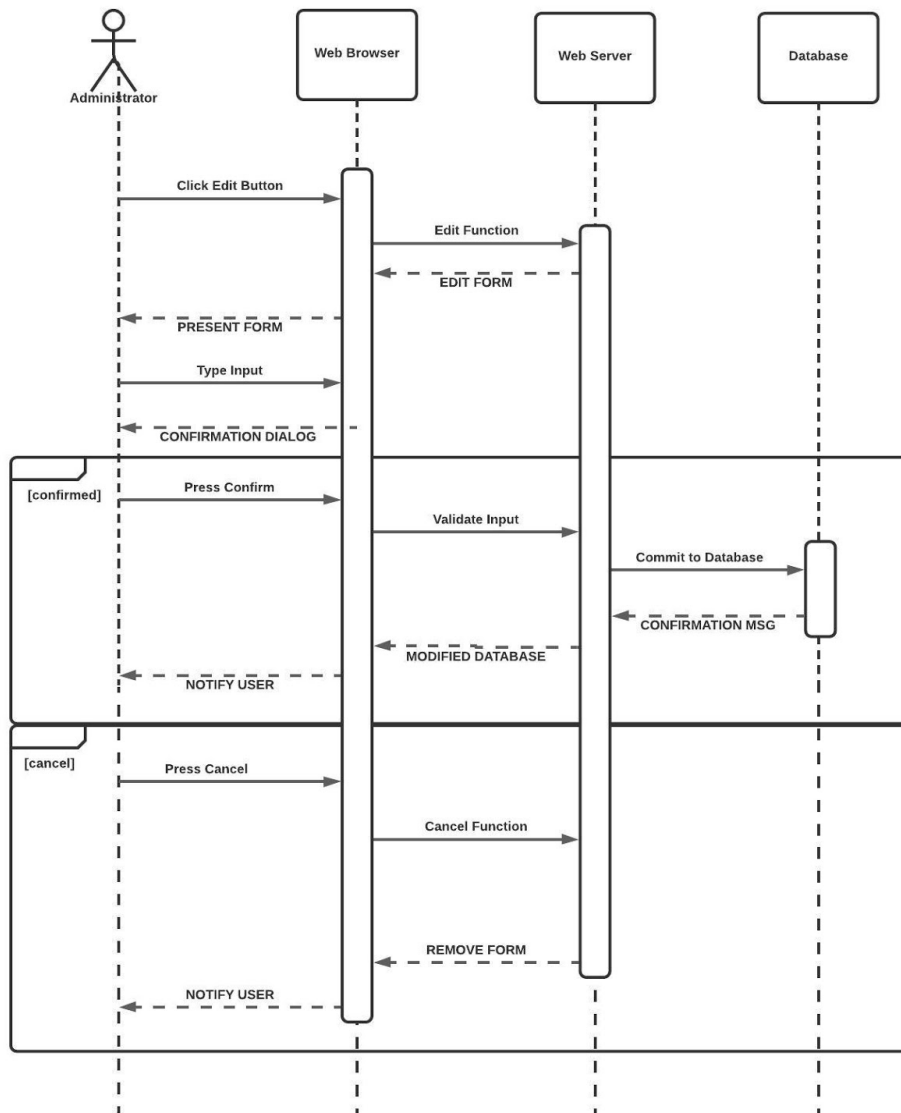
Munashe Masango | February 6, 2021





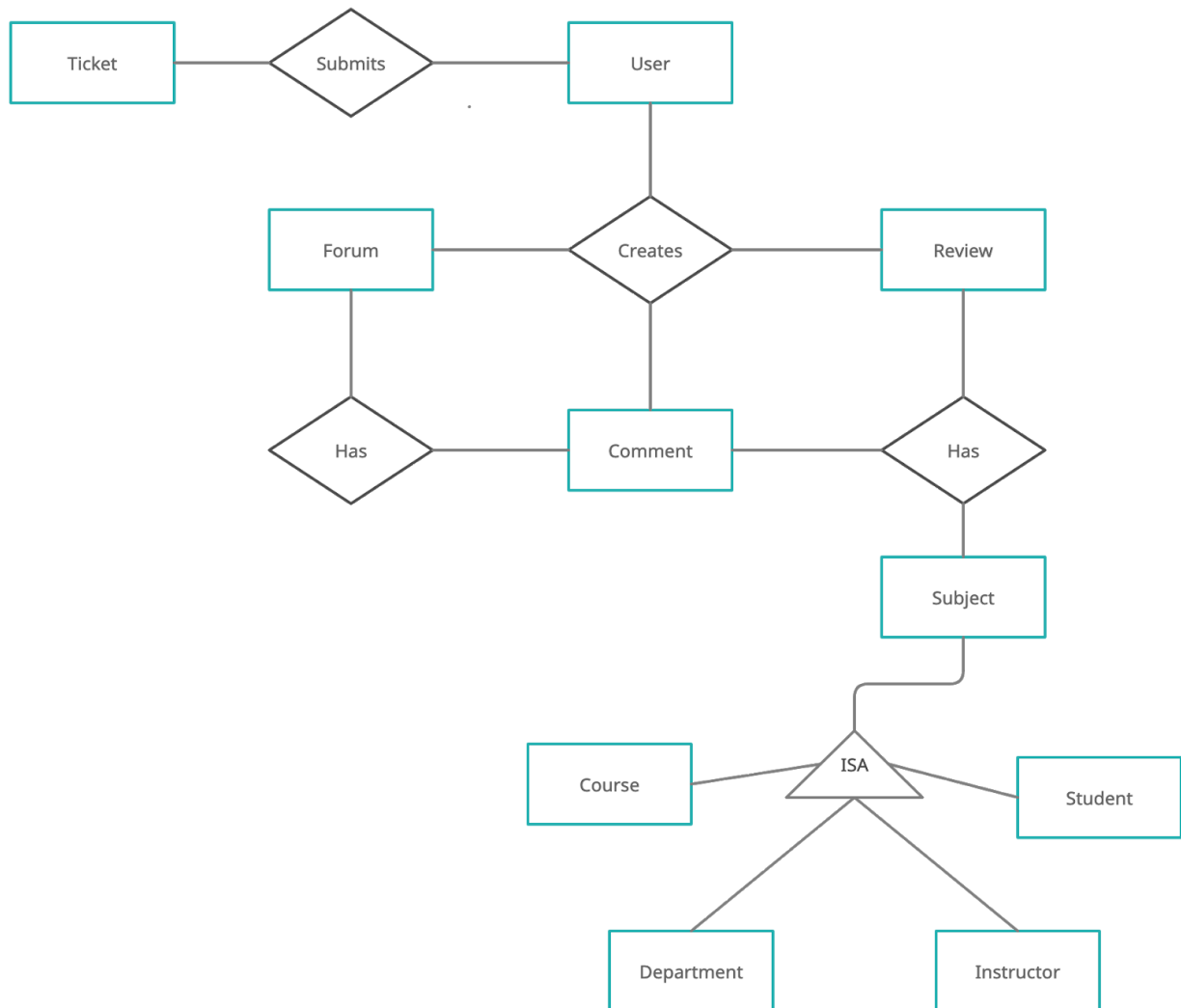
## 3.3.10 Edit Review Item

Sequence Diagram - Modify ReviewItem



## 4.0. Database Design

### 4.1 ER Diagram



## 4.2 Database Tables

### 4.2.1 User Entity

Data Item	Type	Description	Comment
Email	Text/String	Email of the user	Primary Key
Username	Text/String	Username submitted on signup	
Password	Encrypted text/String	Account password	This would be encrypted (only the user themselves would know what it is)
Reviews	Pointer	Review Entity	This would point to each review this user left
Comments	Pointer	Comment Entity	
Forums	Pointer	Forum Entity	
UserType	String	Type of user (dev, student, admin, instructor)	Handles permissions, etc
isVerified	boolean	Used to determine if the user has verified their email	Probably done by sending a code to the provided email
verificationCode	string	Generated upon account creation to verify the given email	Code is emailed to user then they input it into a first time page
DateCreated	Date	Date account was created on	

### 4.2.2 Subject Entity

Data Item	Type	Description	Comment
name	String	Name of dept/instructor/student/course	Course full name [dept+code], primary key

### 4.2.3 Course Entity

Data Item	Type	Description	Comment
Department	Text/String	Department in which the course is from	ex: COSC
Code	Text/String	Course code	ex: 2P03
Avg Rating	int	The average score this course was rated	
Reviews	Pointer	Review Entity	reviews for course
Instructor	Pointer	Instructor Entity	Instructor who taught this course
Aliases	Pointer	Course entity → cross-listed courses	

### 4.2.4 Instructor Entity

Data Item	Type	Description	Comment
department	Text/String	Department in which the course is from	ex: COSC
Avg Rating	int	The average score this course was rated	
Reviews	Pointer	Review Entity	reviews for this prof
overallCourseScore	int	Overall score from course reviews	

### 4.2.5 Review Entity

Data Item	Type	Description	Comment
Id	number	Id number of the review (for book keeping), primary key	Generate from date + random string?
reviewer	User	Points to the user entity who wrote this review	
subject	Subject Entity	Points to the subject (professor / course)	

		of the review	
content	Text/String	Text review written by a user	
Rating	int	Score this rating gave	
reports	int	Amount of times this review has been reported	
tags	text/String	Pre-defined tags about the course/prof for ease of use	
comments	Comment Entity	Points to each comment that has been left on the review	This could be any number $\geq 0$
Date	Date	Date the review was made	
visibleTo	string	Denotes who this review is visible to	Ex: visibleTo: 'public'

#### 4.2.6 Comment Entity

Data Item	Type	Description	Comment
Comment Id	int	Id number of the review	Combined key
Review Id	int		Combined key
Name	string	Name of commenter	
Content	string	Comment nested in review	
Child	Comment entity	Points to all replies to this comment	
DatePosted	Date	Date which the comment was made	
reports	int	Amount of times this review has been reported	

### 4.2.7 Forum Entity

Data Item	Type	Description	Comment
id	Int	Primary key	Not sure maybe date + number?
Subject	Subject Entity	Course, prof, dept, student that the forum is regarding	
Title	String	Basically the top level content of the forum. Question/statement user addressed in forum creation	
Comments	Comment item	Top level comment on a forum	

### 4.2.8 Department Entity

Data Item	Type	Description	Comment
Id	number	Id number of the review	
Name	string	Name of department	
Avg Rating (Overall score)	int/Numerical	The average score this course was rated	
Reviews	Pointer	Review Entity	reviews for this dept
Overall course score	Int		
Overall Professor Score	Int		

### 4.2.9 Ticket

Data Item	Type	Description	Comment
Id	int	Primary key	
user	User	Person submitting the ticket	
content	text/string	Information	
DateSent	Date		

## Security

- Passwords are not stored in the database in plaintext, Django hashes and salts them using the SHA-256 hash algorithm
- Any messages passed from the frontend to the backend containing sensitive information, such as login credentials, are passed using POST and they disallow GET requests
- Password requirements conform to basic modern standards; 12 character minimum, must have at least one uppercase, one lowercase, one digit, and one special character
- Users are verified via a six digit code sent to their provided email, ensuring only users with a valid Brock email can post reviews. This also ensures that a user cannot sign up with an email that does not belong to them
- We strongly recommend to users on the sign up page that they do not use the same password that they use for their Brock accounts to ensure our security does not affect their Brock accounts' security. We also have stronger password requirements than Brock does
- The only data stored on users is their email account, the nickname that they choose and a hash of their password. This means we do not store enough information about a user to identify them, in case of any data leak

## Host/Domain

### Development host and domain:

Development server specifications:

- 2 vCPU Core
- IPv4/6 static public IP
- 4 GB RAM
- 40 GB standard HDD (40MB/s)
- 1 Gbit Network "Unmetered" (non-prioritized)
- Datacenter location: Toronto
- OS: Ubuntu 18.04 LTS 64bit
- Remote access via RDP

Test domain: thedengroup.ca

Server: Apache2 with reverse proxy and redirect HTTP to HTTPS

Was used during development to prepare the project for production. Used to test security and apply secure networking practices in preparation to later switch to production domain.

## Production host and domain:

Production server specifications:

- 2 vCPU Core
- IPv4/6 static public IP
- 4 GB RAM
- 40 GB premium SSD (900MB/s)
- 512 MB scalable RAMDISK (12000MB/s)
- 1 Gbit Network "Unmetered" (prioritized)
- Datacenter location: Toronto
- OS: Ubuntu 20.04.2 LTS 64bit
- 99.99% SLA network uptime
- Remote access via RDP

Production domain: r8scholar.ca

Server: Apache2 with reverse proxy and redirect HTTP to HTTPS

## Testing:

### Informal Function Testing:

#### Admin Functions

Issue:

- Unexpected password error prevents saving. Users cannot be modified due to this. Field length seems to be a recurring issue. Also occurs with departments and courses, while instructors seem to be fine

Expected fix: Correct maximum field length sizes

Issue:

- Some instructors appear as "merged" with other instructors, such as 2 names being treated as one instructor.

Expected fix: Properly separate instructors with a line break

#### Top bar

Issue:

- The bar flickers before displaying properly
- At half size, top bar is shifted to the right side of the screen

Expected fixes:

- Burger menu opens on the left side listing out courses, instructors, department, signup, and login.



Issue: When clicking the R8 SCHOLAR button from the main page it opens a new tab.

Expected fix: R8 SCHOLAR button press results in refresh of page

Minor issue: YouTube icon/button on top bar leads to twitter account

## Home page

Issue:

- Stars are easily mis-aligned, from large course/department/instructor names or window size.

Expected fix: Modify the scaling

## Sign up

Issue:

- When creating an account, it can be very unclear what the user is doing wrong in the event of an error.

Expected fix:

- There should be a way to let the user know whether they have an email error, a username error, a password error, and what the error is, whether by a help button or tooltips, or more descriptive/specific error messages.

## Sign In

Minor issue:

- After entering in the email and password, pressing the enter key doesn't do anything.

Expected fix:

- Pressing enter should have the same effect as pressing the Sign in button.

## Reviews

Minor issue:

- Users should only be able to edit their review for the course.

Expected fix:

- "Create Review" should change to "Edit Review"

Issue:

- When you change your nickname it doesn't update the old reviews with the new nickname.

Expected fix:

- Old reviews should be having your current nickname.

Issue:

- Clicking "VIEW FULL REVIEW" does nothing

Expected fix:

- Add functionality

## Searches

Minor issue:

- Queries sometimes glitch out when typed in too fast, and the display doesn't update in time

Issue:

- When clicking on the highlighted area when selecting, selection does not occur and the area stays dark even after the cursor is not on the search results.

Expected fix:

- Clicking on the highlighted area of the search result should be the same as clicking on the result word itself.

Issue:

- Clicking on the area directly around the search bar results in a visible outline (not the same as the outline you get when clicking on the actual search bar).



Expected fix:

- It should only be possible to click inside the search bar.

## Formal UI/Web Testing:




Formal UI/Web testing done using Katalon Studio, an all-in-one test automation solution. Tested for web, using Google Chrome web browser as testbed with automated tests generated using Katalon Studio. Functionality testing was used for link testing, form validation, cookie testing, and database connection checkup. Performance testing was done by simulating multiple users using functions at the same time via automation and multiple Katalon runtimes. Microsoft Edge was also used in conjunction with Google Chrome for compatibility testing.

## Sign Up


Use Case Name	Sign Up / Create Profile
Trigger	The user clicks the "sign up" button on the website
Precondition	The user has not created a profile for their email address (not in use)
Basic Path	<ol style="list-style-type: none"> <li>1. The system opens a form to enter information</li> <li>2. The user will enter the correct format of data into the form</li> <li>3. The system confirms the information with the database and the user</li> <li>4. The profile is created and the user redirected to the profile page</li> </ol>
Alternative Path	None noted
Postcondition	User is able to log in with the specified email and password

Exception Paths	If the user has already created a Profile using that given email address then the use case abandons
Other (Notes)	This can only be used for each email address once

The first test case was run successfully using a test email:

Job Progress		 
	<b>Test Cases/Sign Up - Chrome - 20210328_135101</b>	1/1
	<Passed> - Chrome	

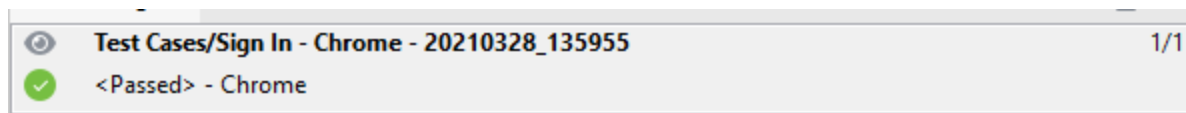
The second test case was also successful using the same test email used previously, and followed the exception path of abandoning the sign up, as a profile was already created with the email.

	<b>Test Cases/Sign Up - Chrome - 20210328_134919</b>	1/1
	<Passed> - Chrome	

## Log in

Use Case Name	Log in to profile
Trigger	The user selects the “log in” on the website
Precondition	The user has created a profile previously using a defined email address
Basic Path	<ol style="list-style-type: none"> <li>1. The system opens a log in dialogue</li> <li>2. The user inputs the correct format of data (text)</li> <li>3. The database authenticates whether that user is allowed access</li> <li>4. The user is redirected to the</li> </ol>
Alternative Path	There is the possibility of being directed to the login via a direct link
Postcondition	The user is redirected and is logged in
Exception Paths	If the user inputs invalid information there will be an error and it will allow you to re-enter the text.

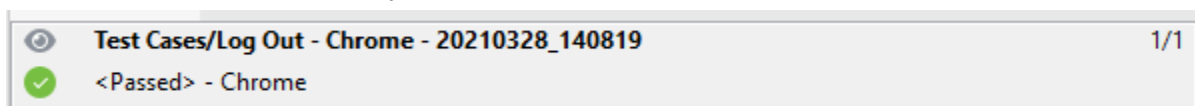
Test case was run twice - one with a valid email and once without. Both cases were handled correctly, and the user was able to log in on the first one, and the user was prompted with an “invalid email” response on the second one.



## Log out

Use Case Name	Log Out
Trigger	The user selects the log out option on the website
Precondition	The user was already logged in and accepted into the database.
Basic Path	<ol style="list-style-type: none"> <li>1. The system prompts the user to make sure they want to log out</li> <li>2. The system will redirect the user to the homepage of the website</li> </ol>
Alternative Path	None noted so far
Postcondition	The user is no longer logged in and able to write reviews
Exception Paths	The user decides against logging out of the website and thus the use case is abandoned.


Test case was run successfully.



## Edit Nickname

Use Case Name	Edit Nickname
Trigger	The user selects the Edit feature within the profile.
Precondition	The user must be authorised, be viewing the profile page they created.
Basic Path	<ol style="list-style-type: none"> <li>1. The profile information form is presented in an editable view</li> <li>2. The user edits the desired fields in order make changes</li> <li>3. The system confirms before making any changes to the profile.</li> </ol>
Alternative Path	None noted
Postcondition	The nickname changes successfully
Exception Paths	If the user updates the information with invalid content it will prompt them to fix it. If the user cancels the changes, there are no new changes and the use case is abandoned.


Username changes were tested successfully.

Test Cases/Edit Profile - Chrome - 20210328_141704		1/1
	<Passed> - Chrome	

## Change Password

Use Case Name	Edit Password
Trigger	The user selects the Edit feature within the profile.
Precondition	The user must be authorised, be viewing the profile page they created.
Basic Path	<ol style="list-style-type: none"> <li>1. The profile information form is presented in an editable view</li> <li>2. The user edits the desired fields in order make changes</li> <li>3. The system confirms before making any changes to the profile.</li> </ol>
Alternative Path	None noted
Postcondition	The user is able to use the new password to log in
Exception Paths	If the user updates the information with invalid content it will prompt them to fix it. If the user cancels the changes, there are no new changes and the use case is abandoned.



Password changing was tested successfully, with both an incorrect and correct password.

Test Cases/Edit Profile - Chrome - 20210328_141928		1/1
	<Passed> - Chrome	



## Delete Profile

Use Case Name	Delete Profile
Trigger	The user clicks delete profile button after accepting terms and conditions and entering their current password
Precondition	The user must be logged in.
Basic Path	<ol style="list-style-type: none"> <li>1. User clicks on "PROFILE" button on top bar</li> <li>2. User clicks on "Delete Profile" tab</li> <li>3. User enters their current password</li> <li>4. User accepts terms and conditions</li> </ol>
Alternative Path	None noted
Postcondition	Account and all associated reviews and comments will be deleted
Exception Paths	If user inputs invalid password or does not accept terms and conditions then the profile will not be deleted, and user will be informed



The case where user inputs correct password and accepts terms and conditions:

 <b>Test Cases/Delete Profile - Chrome - 20210402_204807</b> <span>1/1</span>
 <Passed> - Chrome

The case where user inputs invalid password:

 <b>Test Cases/Delete Profile/Invalid Password - Chrome - 20210402_205202</b> <span>1/1</span>
 <Passed> - Chrome



The case where user does not accept terms and conditions:

 <b>Test Cases/Delete Profile/Unacceptable Conditons - Chrome - 20210402_205606</b> <span>1/1</span>
 <Passed> - Chrome

## Courses Page:



### Going to first

Use Case Name	Going to first
Trigger	The user clicks on the first button on the Courses page
Precondition	None
Basic Path	<ul style="list-style-type: none"> <li>• Checks if first button keeps you on first page when on first page</li> <li>• Checks if first button brings you on first page when on second page</li> <li>• Checks if first button brings you to first page when on a page far away from first page</li> <li>• Checks that clicking on 1 when not on first page brings you to the same page as first page</li> </ul>
Postcondition	The user is brought to the first page
Other (Notes)	Checks all ways of getting to first page

 <b>Test Cases/courses/going to first - Chrome - 20210402_201452</b>	1/1
 <Passed> - Chrome	

### Going to Last

Use Case Name	Going to last
Trigger	The user clicks on the last button on the Courses page
Precondition	None
Basic Path	<ul style="list-style-type: none"> <li>• Checks if last button brings you on last page when on first page</li> <li>• Checks if last button brings you on last page when on second last page</li> <li>• Checks if last button keeps you to last page when on the last page</li> <li>• Checks that clicking on the last listed number when on the second last page brings you to the same page as the last page</li> </ul>
Postcondition	The user is brought to the last page
Other (Notes)	Checks all ways of getting to last page

 <b>Test Cases/courses/going to last - Chrome - 20210402_202725</b>	1/1
 <Passed> - Chrome	



## Opening Course

Use Case Name	Opening Course
Trigger	The user clicks on a course listed one the Courses page
Precondition	None
Basic Path	<ol style="list-style-type: none"> <li>1. User goes to home page</li> <li>2. User clicks on courses</li> <li>3. User clicks on course name</li> <li>4. User brought to individual course page which is the same as the course name clicked on</li> </ol>
Alternative Path	<ol style="list-style-type: none"> <li>1. User goes to home page</li> <li>2. User clicks on courses</li> <li>3. User clicks on course code</li> <li>4. User brought to individual course page which is the same as the course code clicked on</li> </ol>
Postcondition	The user is brought to the last page
Other (Notes)	Checks all ways of getting to last page

**Test Cases/courses/opening course - Chrome - 20210402\_201514**

1/1



&lt;Passed&gt; - Chrome

## Opening Department

Use Case Name	Opening Department
Trigger	The user clicks on a Department listed one the Courses page
Precondition	None
Basic Path	<ol style="list-style-type: none"> <li>1. User goes to home page</li> <li>2. User clicks on courses</li> <li>3. User clicks on department for a course</li> <li>4. User brought to individual department page which matches department clicked on</li> </ol>
Postcondition	The user is brought to individual department page

**Test Cases/courses/opening department - Chrome - 20210402\_203334**



1/1



&lt;Passed&gt; - Chrome

## Sorting System


Use Case Name	Sorting System
Trigger	The user clicks on the sorting options for Courses
Precondition	Sorting is currently on A-Z
Basic Path	<ul style="list-style-type: none"> <li>• Checks if Sorting by Z-A causes a difference</li> <li>• Clicks on highest to lowest rating to sorts by highest rating</li> <li>• Sorts by lowest to highest rating to sort by lowest rating first</li> </ul>
Postcondition	The ratings are sorted based on the selected sorting option

 **Test Cases/courses/sorting system - Chrome - 20210402\_203032** 1/1  
 <Passed> - Chrome

## Instructors Page:



## Going to first

Use Case Name	Going to first
Trigger	The user clicks on the first button on the Instructors page
Precondition	None
Basic Path	<ul style="list-style-type: none"> <li>• Checks if first button keeps you on first page when on first page</li> <li>• Checks if first button brings you on first page when on second page</li> <li>• Checks if first button brings you to first page when on a page far away from first page</li> <li>• Checks that clicking on 1 when not on first page brings you to the same page as first page</li> </ul>
Postcondition	The user is brought to the first page
Other (Notes)	Checks all ways of getting to first page

**Test Cases/instructors/Going to first - Chrome - 20210402\_203540** 1/1  
 <Passed> - Chrome


## Going to Last

Use Case Name	Going to last
Trigger	The user clicks on the last button on the Instructors page
Precondition	None
Basic Path	<ul style="list-style-type: none"> <li>• Checks if last button brings you on last page when on first page</li> <li>• Checks if last button brings you on last page when on second last page</li> <li>• Checks if last button keeps you to last page when on the last page</li> <li>• Checks that clicking on the last listed number when on the second last page brings you to the same page as the last page</li> </ul>
Postcondition	The user is brought to the last page
Other (Notes)	Checks all ways of getting to last page

 **Test Cases/instructors/Going to last - Chrome - 20210402\_204047** 1/1  
 <Passed> - Chrome

## Opening Instructor

Use Case Name	Opening Instructor
Trigger	The user clicks on a instructor listed one the Instructors page
Precondition	None
Basic Path	<ol style="list-style-type: none"> <li>1. User goes to home page</li> <li>2. User clicks on instructors</li> <li>3. User clicks on instructor's name</li> <li>4. User brought to individual instructor page which is the same as the instructor's name clicked on</li> </ol>
Alternative Path	<ol style="list-style-type: none"> <li>1. User goes to home page</li> <li>2. User clicks on instructors</li> <li>3. User changes instructors page</li> <li>4. User clicks on instructor's name</li> <li>5. User brought to individual instructor's page which is the same as the instructor's name clicked on</li> </ol>
Postcondition	The user is brought to the individual instructor's page

**Test Cases/instructors/opening instructor - Chrome - 20210402\_203557** 1/1  
 <Passed> - Chrome

## Opening Department

Use Case Name	Opening Department
Trigger	The user clicks on a department listed one the Instructors page
Precondition	None
Basic Path	<ol style="list-style-type: none"> <li>1. User goes to home page</li> <li>2. User clicks on instructors</li> <li>3. User clicks on instructor's department</li> <li>4. User brought to individual department page which is the same as the department name clicked on</li> </ol>
Alternative Path	<ol style="list-style-type: none"> <li>1. User goes to home page</li> <li>2. User clicks on instructors</li> <li>3. User changes instructors page</li> <li>4. User clicks on instructor's department</li> <li>5. User brought to individual department page which is the same as the instructor's department clicked on</li> </ol>
Postcondition	The user is brought to the individual department page



**Test Cases/instructors/opening departments - Chrome - 20210402\_204153**

1/1



<Passed> - Chrome

## Sorting System

Use Case Name	Sorting System
Trigger	The user clicks on the sorting options for instructors
Precondition	Sorting is currently on A-Z
Basic Path	<ul style="list-style-type: none"> <li>• Checks if Sorting by Z-A causes a difference</li> <li>• Clicks on highest to lowest rating to sorts by highest rating</li> <li>• Sorts by lowest to highest rating to sort by lowest rating first</li> </ul>
Postcondition	The ratings are sorted based on the selected sorting option



**Test Cases/instructors/Sorting systems - Chrome - 20210402\_204324**

1/1




<Passed> - Chrome

## Departments Page:


### Going to first

Use Case Name	Going to first
Trigger	The user clicks on the first button on the departments page
Precondition	None
Basic Path	<ul style="list-style-type: none"> <li>• Checks if first button keeps you on first page when on first page</li> <li>• Checks if first button brings you on first page when on second page</li> <li>• Checks if first button brings you to first page when on a page far away from first page</li> <li>• Checks that clicking on 1 when not on first page brings you to the same page as first page</li> </ul>
Postcondition	The user is brought to the first page
Other (Notes)	Checks all ways of getting to first page

<b>Test Cases/departments/Going to first - Chrome - 20210402_204430</b>		1/1
	<Passed> - Chrome	

### Going to last

Use Case Name	Going to last
Trigger	The user clicks on the last button on the departments page
Precondition	None
Basic Path	<ul style="list-style-type: none"> <li>• Checks if last button brings you on last page when on first page</li> <li>• Checks if last button brings you on last page when on second last page</li> <li>• Checks if last button keeps you to last page when on the last page</li> <li>• Checks that clicking on the last listed number when on the second last page brings you to the same page as the last page</li> </ul>
Postcondition	The user is brought to the last page
Other (Notes)	Checks all ways of getting to last page

<b>Test Cases/departments/going to last - Chrome - 20210402_204436</b>		1/1
	<Passed> - Chrome	

## Opening Top Instructor

Use Case Name	Opening top Instructor
Trigger	The user clicks on a instructor listed one the Instructors page
Precondition	None
Basic Path	<ol style="list-style-type: none"> <li>1. User goes to home page</li> <li>2. User clicks on departments</li> <li>3. User clicks on top instructor for department</li> <li>4. User brought to individual instructor page which is the same as the instructor's name clicked on</li> </ol>
Alternative Path	<ol style="list-style-type: none"> <li>1. User goes to home page</li> <li>2. User clicks on departments</li> <li>3. User clicks on department that doesn't have any instructors</li> <li>4. User clicks on No Data</li> <li>5. User brought to page that can return user to instructors page</li> </ol>
Postcondition	The user is brought to the individual instructor's page if it exists
Other (Notes)	A department with no instructors will have a top instructor of "No Data"

**Test Cases/departments/opening top instructor - Chrome - 20210402\_204459**

1/1



<Passed> - Chrome

## Opening Top Course

Use Case Name	Opening top Course
Trigger	The user clicks on a top Course listed one the Departments page
Precondition	None
Basic Path	<ol style="list-style-type: none"> <li>1. User goes to home page</li> <li>2. User clicks on departments</li> <li>3. User clicks on top courses name</li> <li>4. User brought to individual course page which is the same as the courses name clicked on</li> </ol>
Postcondition	The user is brought to the individual instructor's page



**Test Cases/departments/opening top course - Chrome - 20210402\_205138**

1/1



<Passed> - Chrome

## Opening Department

Use Case Name	Opening Department
Trigger	The user clicks on a department listed one the departments page
Precondition	None
Basic Path	<ol style="list-style-type: none"> <li>1. User goes to home page</li> <li>2. User clicks on departments</li> <li>3. User clicks on department name</li> <li>4. User brought to individual department page which is the same as the department name clicked on</li> </ol>
Postcondition	The user is brought to the individual department page

**Test Cases/departments/opening departments - Chrome - 20210402\_204443**

1/1



<Passed> - Chrome

## Sorting System:

Use Case Name	Sorting System
Trigger	The user clicks on the sorting options for departments
Precondition	Sorting is currently on A-Z
Basic Path	<ul style="list-style-type: none"> <li>• Checks if Sorting by Z-A causes a difference</li> <li>• Clicks on highest to lowest rating to sorts by highest rating</li> <li>• Sorts by lowest to highest rating to sort by lowest rating first</li> </ul>
Postcondition	The ratings are sorted based on the selected sorting option



**Test Cases/departments/sorting system - Chrome - 20210402\_205306**

1/1


<Passed> - Chrome

## Individual Course / Instructors / Departments Page

### Individual Course Page:


Use Case Name	Individual Course testing
Trigger	The user clicks to go to an individual Course Page
Precondition	Course Exists
Basic Path	<ul style="list-style-type: none"> <li>• Checks if name matches course name listed on courses</li> <li>• Checks if course code matches course code listed on courses</li> <li>• Checks if department listed brings you expected individual department</li> </ul>
Postcondition	All links work as intended


**Test Cases/Individual course/individual course testing - Chrome - 20210402\_205423**
1/1


 <Passed> - Chrome

### Individual Instructors Page:

Use Case Name	Individual Instructor testing
Trigger	The user clicks to go to an individual Instructors Page
Precondition	Instructor Exists
Basic Path	<ul style="list-style-type: none"> <li>• Checks if instructors name matches instructors name listed on instructors</li> <li>• Checks if popular instructors brings you to expected individual instructors pages</li> <li>• Checks if popular courses brings you to expected individual courses pages</li> <li>• Checks if department listed brings you expected individual department</li> </ul>
Postcondition	All links work as intended


**Test Cases/individual instructor/individual instructor testing - Chrome - 20210402\_205617**
1/1



 <Passed> - Chrome



**Individual Departments Page:**


Use Case Name	Individual department testing
Trigger	The user clicks to go to an individual Departments Page
Precondition	Department Exists
Basic Path	<ul style="list-style-type: none"> <li>• Checks if department name matches department name listed</li> <li>• Checks if popular instructors brings you to expected individual instructors pages</li> <li>• If there are no instructors for the department, checks that no instructors are listed</li> <li>• Checks if popular courses goes to expected individual course</li> </ul>
Postcondition	All links work as intended
Other (Notes)	Departments are known to have no instructors if there is no top instructor listed on department page (will have "No Data" in its place)


**Test Cases/individual department/individual department testing - Chrome - 20210402\_205437**
1/1


<Passed> - Chrome

**Create Review**



Use Case Name	Create Review
Trigger	Goes to create review tab
Precondition	Signed in user on specific course, instructor or department page
Basic Path	<ol style="list-style-type: none"> <li>1. User selects "Create Review" tab</li> <li>2. User inputs a Review Title</li> <li>3. User selects a rating for each category listed</li> <li>4. User writes down what they think about the respective course, instructor or department</li> <li>5. User clicks on "SUBMIT" button to submit review</li> </ol>
Alternative Path	None noted
Postcondition	Review will be published and publicly available
Exception Paths	If title and detail are not included their review will not be submitted


**Test Cases/Review/Create Review - Chrome - 20210403\_002613**
1/1


<Passed> - Chrome


## Edit Review

Use Case Name	Edit Review
Trigger	User wishes to edit their review
Precondition	The user must be logged in and on a specific course, instructor, department page, or their profile
Basic Path	<ol style="list-style-type: none"> <li>1. User selects "Edit" button on their review</li> <li>2. User edits their Review Title if they want to</li> <li>3. User changes a rating for each category listed if they want to</li> <li>4. User edits what they think about the respective course, instructor or department if they want to</li> <li>5. User clicks on "SUBMIT" button to submit edited review</li> </ol>
Alternative Path	None noted
Postcondition	Edited Review will be published and publicly available
Exception Paths	If review has no title and detail, their review will not be submitted

	Test Cases/Review/Edit Review - Chrome - 20210403_003606	1/1
	<Passed> - Chrome	

## Report Review



Use Case Name	Report Review
Trigger	User wishes to report a review
Precondition	None
Basic Path	<ol style="list-style-type: none"> <li>1. User clicks on "REPORT" button on review</li> <li>2. User selects what's wrong with review</li> <li>3. User enters explanation if they choose to</li> <li>4. User clicks "REPORT" button to submit</li> </ol>
Alternative Path	None noted
Postcondition	User will return to page viewing review
Exception Paths	If user makes no selection clicking submit will return user to review

	Test Cases/Review/Report Review - Chrome - 20210403_004643	1/1
	<Passed> - Chrome	



## Search

Use Case Name	Search
Trigger	User starts a search
Precondition	None
Basic Path	<ol style="list-style-type: none"> <li>1. User clicks on R8Scholar search bar</li> <li>2. User types desired course, instructor or department name or code</li> <li>3. User presses enter</li> <li>4. User clicks on their chosen course, instructor or department</li> <li>5. User is brought to their chosen course, instructor or department page</li> </ol>
Alternative Path	None noted
Postcondition	User will be on their chosen course, instructor or department page
Exception Paths	If user inputs name or code incorrectly they will not be able to reach their desired course, instructor or department page



Tests searching course names and course codes:

	<b>Test Cases/Search/Search Courses - Chrome - 20210403_022618</b>	1/1
	<Passed> - Chrome	

Tests searching instructor names:

	<b>Test Cases/Search/Search Instructors - Chrome - 20210403_022700</b>	1/1
	<Passed> - Chrome	

Tests searching department names:

	<b>Test Cases/Search/Search Departments - Chrome - 20210403_022740</b>	1/1
	<Passed> - Chrome	

## Formal Unittest Testing:

Formal Unit testing for python/Django framework of R8Scholar was accomplished using unittest unit testing framework. Unittest was used for test automation by creating a test suite with test cases covering a range of functions in the python/Django framework. A test runner is used to orchestrate the tests and to provide a text interface indicating the results from executing the tests.

### Models Test Cases:

#### Imports:

```
from django.test import TestCase
from api.models import CustomUser
from api.models import Department
from api.models import Instructor
from api.models import Course
from api.models import Review
```

#### Setup:

```
def setUp(self):
    CustomUser.objects.create(email="lb16tp@brocku.ca",nickname="logan",password="G*Zfb2UcV6l0")
    CustomUser.objects.create(email="lb16tp2@brocku.ca",nickname="logan2",password="z^!ZeJ$d%8hO",is_admin=True)
    Department.objects.create(name="departmenttest",courses_rating=3.555,instructors_rating=2.444,rating=3.00)
    Dept=Department.objects.get(name="departmenttest")
    Cuser=CustomUser.objects.get(email="lb16tp@brocku.ca")
    Instructor.objects.create(name="instructortest",department=Dept,rating=3.44,)
    Inst=Instructor.objects.get(name="instructortest")
    Course.objects.create(name="coursetest",department=Dept,rating=4.22,course_full_name="fullnametest")
    Cor=Course.objects.get(name="coursetest")

    Review.objects.create(reviewer=Cuser,nickname="testnick",subject="tester",title="test",rating=2.5,department_name=Dept,instructor_name=Inst,course_name=Cor)
```

#### Test str

##### Test code:

```
def test_str(self):
    CUser=CustomUser.objects.get(email="lb16tp@brocku.ca")
    self.assertEqual(CUser._str(), 'lb16tp@brocku.ca')
```

##### Results:

```
TestCase_test_str (__main__.ModelsTestCase) ... ok
```

#### Email User

##### Test code:

```
def test_email_user(self):
    CUser=CustomUser.objects.get(email="lb16tp@brocku.ca")
```

##### Results:

```
TestCase_test_email_user (__main__.ModelsTestCase) ... ok
```

## Has Permission

Test code:

```
def test_has_perm(self):
    CUser=CustomUser.objects.get(email="lb16tp@brocku.ca")
    self.assertTrue(CUser.has_perm(CUser.is_active))
```

Results:

```
TestCase_test_has_perm (__main__.ModelsTestCase) ... ok
```

## Has Module Permissions

Test code:

```
def test_has_module_perms(self):
    CUser=CustomUser.objects.get(email="lb16tp@brocku.ca")
    self.assertTrue(CUser.has_module_perms(CUser.is_active))
```

Results:

```
TestCase_test_has_module_perms (__main__.ModelsTestCase) ... ok
```

## Is Staff

Test code:

```
def test_is_staff(self):
    CUser=CustomUser.objects.get(email="lb16tp@brocku.ca")
    CUser2=CustomUser.objects.get(email="lb16tp2@brocku.ca")
    self.assertFalse(CUser.is_staff)
    self.assertTrue(CUser2.is_staff)
```

Results:

```
TestCase_test_is_staff (__main__.ModelsTestCase) ... ok
```

## Update Department Rating

Test code:

```
def test_dept_update_rating(self):
    Dept=Department.objects.get(name="departmenttest")
    Dept.update_rating()
    self.assertEqual(Dept.rating,2.5)
```

Results:

```
TestCase_test_dept-update (__main__.ModelsTestCase) ... ok
```

## Update Instructor Rating within Department

Test code:

```
def test_update_instructor_rating(self):
    Dept=Department.objects.get(name="departmenttest")
    Dept.update_instructor_rating()
    self.assertEqual(Dept.instructors_rating,2.5)
```

Results:

```
TestCase_test_update_instructor_rating (__main__.ModelsTestCase) ... ok
```

## Update Course Rating within Department

Test code:

```
def test_update_course_rating(self):
    Dept=Department.objects.get(name="departmenttest")
    Dept.update_course_rating()
    self.assertEqual(Dept.courses_rating,2.5)
```

Results:

```
TestCase_test_update_course_rating (__main__.ModelsTestCase) ... ok
```

## Update Course Rating

Test code:

```
def test_course_update_rating(self):
    Cor=Course.objects.get(name="coursetest")
    Cor.update_rating()
    self.assertEqual(Cor.rating,2.5)
```

Results:

```
TestCase_test_course_update_rating (__main__.ModelsTestCase) ... ok
```

## Update Instructor Rating

Test code:

```
def test_instructor_update_rating(self):
    Inst=Instructor.objects.get(name="instructortest")
    Inst.update_rating()
    self.assertEqual(Inst.rating,2.5)
```

Results:

```
TestCase_test_instructor_update_rating (__main__.ModelsTestCase) ... ok
```

## Generators Test Case

### Imports:

```
from django.test import TestCase
from api.generators import generate_validation_code
```

### Test code:

```
class generatorsTestCase(TestCase):

    def test_generate_validation_code(self):
        self.assertEqual(len(generate_validation_code(6)),6)
        self.assertEqual(len(generate_validation_code(8)),8)
        self.assertEqual(len(generate_validation_code()),6)
```

### Results:

```
TestCase_test_generate_validation_code (__main__.GeneratorsTestCase) ... ok
```

## Validators Test Case:

### Imports:

```
from django.test import TestCase
from api.validators import validate_brock_mail
from api.validators import profanity_validator
from api.validators import rating_validator
from django.core.exceptions import ValidationError
```

### Validate Brock Email

### Test code:

```
def test_validate_brock_mail(self):
    mail="lb16tp@brocku.ca"
    fakemail="somemail@notgood.ca"
    self.assertEqual(validate_brock_mail(mail), 'lb16tp@brocku.ca')
    with self.assertRaises(ValidationError):
        validate_brock_mail(fakemail)
```

### Results:

```
TestCase_test_validate_brock_mail (__main__.ValidatorsTestCase) ... ok
```

## Profanity Validator

### Test code:

```
def test_profanity_validator(self):
    goodword="good"
    badword="nasty"
    self.assertEqual(profanity_validator(goodword), "good")
    with self.assertRaises(ValidationError):
        profanity_validator(badword)
```

### Results:

```
TestCase_test_profanity_validator (__main__.ValidatorsTestCase) ... ok
```

## Password Validator

Test code:

```
def password_validator(self):
    strongpassword="This#isAstrongpassword!132"
    weakpassword="thisisnt"
    weakpassword2="neitheristhispassword"
    weakpassword3="Neitheristhispassword2"
    self.assertEqual(password_validator(strongpassword),"This#isAstrongpassword!132")
    self.assertNotEqual(password_validator(weakpassword),"thisisnt")
    self.assertNotEqual(password_validator(weakpassword1),"neitheristhispassword")
    self.assertNotEqual(password_validator(weakpassword2),"Neitheristhispassword2")
```

Results:

```
TestCase_test_password_validator (__main__.ValidatorsTestCase) ... ok
```

## Rating Validator

Test code:

```
def rating_validator(self):
    rating=3.5
    rating2=5.5
    self.assertEqual(rating_validator(rating),3.5)
    self.assertNotEqual(rating_validator(rating2),5.5)
```

Results:

```
TestCase_test_rating_validator (__main__.ValidatorsTestCase) ... ok
```

## Managers Test Case:

### Create User

Test code:

```
def test_create_user(self):
    Cuser = CustomUser.objects.create_user("sometmail@brocku.ca","nickname23","Asecurepassword##234")
    self.assertEqual(Cuser.nickname, 'nickname23')
    self.assertEqual(Cuser.email,'sometmail@brocku.ca')
    self.assertFalse(Cuser.is_admin)
```

Results:

```
TestCase_test_create_user (__main__.ValidatorsTestCase) ... ok
```

### Create Superuser

Test code:

```
def test_create_superuser(self):
    Cuser = CustomUser.objects.create_superuser("someothermail@brocku.ca","nickname2345","Asecurepassword##2345")
    self.assertEqual(Cuser.nickname, 'nickname2345')
    self.assertEqual(Cuser.email,'someothermail@brocku.ca')
    self.assertTrue(Cuser.is_admin)
```

Results:

```
TestCase_test_create_superuser (__main__.ValidatorsTestCase) ... ok
```



## Database Test Case

### Imports:

```
from django.test import TestCase
from api.models import Instructor, Course, Department
from api.db_entries import ModelGenerator
```

### Test code:

```
class db_entriesTestCase(TestCase):
    def test_Department_db_entries(self):
        ModelGenerator('./api/data/departments.txt', "Department")
        self.assertTrue(True)
        ModelGenerator('./api/data/instructors.txt', "Instructor")
        self.assertTrue(True)
        ModelGenerator('./api/data/courses.txt', "Course")
        self.assertTrue(True)
```

### Results:

```
TestCase_test_Department_db_entries (__main__.ModelGeneratorTestCase) ... ok
```

```
-----
Ran 16 tests in 20.527s

OK
Destroying test database for alias 'default'...
```

## Sprints / Teamwork policy:

### Meeting 1 - First Group Meeting - Jan 11th

- Everyone attended and contributed, Meeting 1 summary includes more details
- James and Seth feel comfortable with web development.
- James, Seth, Twino(Mathew) and Erikas have some app development experience.
- James and Grant have some database experience.
- Everyone is familiar with GitHub.
- Seth offered to make a Youtube video to promote the project

### Meeting 2 - Discuss Requirements Elicitation - Jan 18th

- Twino(Mathew) Puthiakunnel emailed and invited Professor Naser
- Everyone in the group attended the meeting
- Twino(Mathew) Puthiakunnel created shared Requirements Elicitation document
- James Sargent suggested using Django for databases
- Seth Shickluna-Pierce decided to work on database as well
- Logan Bell and Erikas Klimusinas decided to work on the UI using Figma
- Grant Nike and Luciano Ugalde decided to work on diagrams and may use draw.io
- Munashe Mansango agreed to work on requirements
- Twino(Mathew) Puthiakunnel decided to work on requirements as well

### Meeting 3 - Share resources and information - Jan 22nd

- Figma was shared by Logan Bell
- PDF of UI development was shared by Erikas Klimusinas
- James Sargent shared tutorial on django web development
- Seth Shickluna-Pierce demonstrated Amazon AWS deployment at <http://3.96.54.189/>
- Seth Shickluna-Pierce registered the domain r8scholar.ca
- PDF of UI development was shared by Logan Bell
- Erikas Klimusinas started a discussion if there is a need for a registration page
- James Sargent suggested that github should be set up with the basic project soon
- Seth Shickluna-Pierce suggested using React for the front end of the site
- Luciano Ugalde and Grant Nike made a diagrams channel in Teams

### Meeting 4 - Ask Professor Naser questions - Jan 25th

- Twino(Mathew) Puthiakunnel asked questions to Professor Naser on behalf of the team
- Logan Bell and Erikas Klimusinas Suggested Changes to be made in order for the back end to be compatible with UI
- Twino(Mathew) Puthiakunnel offered to be in charge of full-stack coordination

- Seth Shickluna-Pierce offered to be in charge of front end coordination
- James Sargent offered to be in charge of back end coordination
- Group decided on updated requirements for system entities
- James Sargent begun the setup of the back end project
- Seth Shickluna-Pierce begun the setup of the front end

## Meeting 5 - Finalize Requirements - Jan 29th

- Seth Shickluna-Pierce, James Sargent, and Twino(Mathew) Puthiakunnel discussed how the teams should be split up and what topics they should cover during sprints
- Twino(Mathew) Puthiakunnel made channels for Sprint Planning and Sprint Review

## Meeting 6 - Sprint Planning 1 - Feb 1th

- Twino(Mathew) Puthiakunnel conducted the meeting
- M.Masango updated the GitHub Readme

## Meeting 7 - Sprint Refinement 1 - Feb 8th

- (Feb 11) M.Masango added About page content, added footer component
- (Feb 12) M.Masango added basic forum posting functionality

## Meeting 8 - Sprint Review Meeting 1 - Feb 14th

- Scheduled FullStack Meeting 2
- Seth began discussion on hosting situation

## Meeting 9 - Sprint Planning 2 - Feb 15th

- Group completed backlog
- Seth Shickluna-Pierce shared current hosting situation
- James Sargent shared the current status of the REST api
  - Initial models setup

## Meeting 10 - Sprint Refinement 2 - Feb 18th

- Seth did fetching from API
- Grant has been working with user creation forms, and is working on email verification
- (Feb 18) M.Masango began work on the forum home page.
- (Feb 19) M.Masango added a basic Profile Page
- (Feb 20) M.Masango added custom forms and profile picture editing to Profile page

## Meeting 11 - Sprint Planning 3 (Review 2) - Feb 22th

- Seth made signup with accounts work
  - Seth demonstrated account signup
- James fixing the database
  - Cleaning up models.py, admin.py and serializers.py
- Munashe added to account page, and settings page

## Meeting 12 - Sprint Review 3 (FullStack 3) - Feb 27th

- Seth demonstrated new project structure, used webpack and babel
- Grant worked on new models, rewrote error messages
- (Feb 27) M.Masango added an admin settings page
- (Feb 28) M.Masango added the custom forms for additional functionality on the settings page.

## Meeting 13 - Sprint Planning 4 (Release 1) - March 1st

Split tasks:

- API calls, get-create views (Luciao, Grant, James)
- Populate database, text files, run scripts (Twino, Logan)
- Styles (Seth, Munashe)
- Functional components (all buttons should work)
  - Login/logout (Grant)
  - Changing password/nickname (Grant)
  - Course display page (frontend team)
- API calls (Grant, Luciano, James)
- Database (James)
- Grant and James went over setting up local backend with Luciano

## Meeting 14 - Sprint Review 4 (Release 1) - March 5th

- Seth demonstrated the new looks for the R8Scholar site

## Meeting 15 - Release Meeting - March 7th

- Grant suggested changes
- James removed migrations from github

## Meeting 16 - Sprint Planning 5 - March 8th 2021

### Objectives/Backlog:

- Get user to verify their account as soon as they create
- Add terms and conditions to sign up
- Fix text that's hard to read on some pages
- Change Edit Profile into Change Nickname and Change Password
- Implement searching using api calls
- filtering implementation (user supplied)
  - by name
  - department
  - rating (high or low)
- Finish scraping list of instructors
- (later) Email confirmation

### Edit profile page:

1. Change username and change password should be separate.
2. Shouldn't need to ever change email.
3. Terms and conditions should be agreed to when the user signs up.
4. Github and youtube links not fixed

### Create Review page(Course,Department,Instructor):

1. Wording is redundant.
2. Shouldn't be asking to rate instructors in course reviews and courses in instructor reviews.

### Sign Up Page:

1. Doesn't prompt users to validate their account after signing up.
2. Doesn't login after they create an account.
3. Terms and conditions should be agreed to when the user signs up.

### Misc:

- Some fonts can't be seen well(example:the home page)
- Nav bar icons could be larger

### Progress:

- Grant: Fixed top instructors, worked on review and email confirmation
- James: Worked on search implementation and demonstrated it during sprint refinement
- Seth: Tested out filtering results in the backend and shared his conclusions
- Logan: Worked on setting up the necessary data of which instructors are in which courses

### Things different compared to release 1

- API pages should not be accessible openly

## Meeting 17 - Sprint Refinement 5 - March 11th 2021

- Everyone attended the meeting
- Grant
  - Documenting testing, (issues) on Release 2 document
  - Encouraged more communication
- James
  - review, comment linked with users (after the demo)
    - displaying user's reviews
  - Issue with current api being exposed (after the demo)
- Logan
  - Has to finish up instructors db
  - Bug: Z-A, on top of A-Z
- Twino demonstrated inconsistency with top bar
  - Switching pages had issues
  - Nickname changed, but page needs to be refreshed
  - "First" button causes page to be blank after changing multiple pages
  - Mobile (optional)
- Seth
  - Authentication token (after the demo)
  - Report and auditing feature (after the demo)
  - Heroku
- Distributed tasks
- Front end:
  - Login/Signup Page: Make return button an actual "Button Component" (Nash)
  - Login Page: List requirements on creating an account (Nash)
  - Courses/Instructors/Departments: Page Filtering working and displaying. (Seth)
  - Profile Page: Redirect if Not logged in (Erikas)
  - Nav Bar: Change Spacing, rearrange features (Seth).
- Backend:
  - Issue with reviews being associated with user (James)
  - Authentication token (Grant & Seth)
  - Exposed API page (Grant)
- Testing:
  - Twino(Mathew) - Top Bar, Bottom Bar, Reviews, Searches
  - Luciano - Front page, Top 5, Sign up, Login page
  - Logan - Course, Instructor and Department lists and individual pages

## Meeting 18 - Sprint Planning 6 (Review 5) - March 16th 2021

- Everyone attended the meeting

Release 2 work:

- Remove Forums
- Attribute Reviews to the user (display on profile)
- Attribute Comments to the user
- Edit Reviews
- Report Reviews
- Delete Reviews
- Token-Based Authentication
- User prompts on verify (junk folder) and signup (don't use brock pass)
- Having professor to rate students (Final Release)

Full Stack/Test:

- work on formal testing for UI
- use Katalon Studio

## Meeting 19 - Sprint Planning 7 (Review 6) - March 22th 2021

- Everyone except Erikas attended (was given explanation later)
- James
  - Connected reviews to user
    - Demonstrated
  - Added searches for course names
    - Had to add an extra dummy column
  - Removed forums model
- Consider making a way to generate reviews
- Seth and Grant working on Token-Based Authentication
  - Profile Page: Redirect if Not logged in
  - Grant and Seth decided to meet for token auth on Tuesday 23rd
- James will work on hiding API page
- Grant will work on edit reviews, report review, and delete review views
- Erikas will work on help functions for create account, login
- Discussed Heroku
  - Security flaws must be addressed before deployment
  - Seth will handle deploying on Heroku
- FullStack will work on UI testing automation
  - Should have most major use cases ready by March 29th

## Meeting 20 - Sprint Refinement 7 - March 25th 2021

- James was unavailable for meeting, had informed earlier
- Demonstrate any changes
  - Login/sign up ui changes
  - Delete profile
  - Added verify account requirement on all pages needing it
  - Optimized hrefs
  - Backend: getuserreviews gets all the reviews for a specific user
- Testing progress
  - Showed documentation
    - need to correct some post conditions
  - Showed scripting and explained testing functionality
- Discussed how to document backlog and concluded that user stories with progress reports are what make up backlog
- Discussed possible features to implement
  - Add all reviews to profile page
  - Add tags to review e.g. tough marker, approachable

## Meeting 21 - Sprint Planning 8 (Review 7) - March 29th 2021

- Everyone except Luciano attended meeting, Luciano informed about absence beforehand
- Seth demonstrated updates to profile pages
- Twino(Mathew) and Logan demonstrated formal UI testing
- Created list of all functions that need user stories and progress reports as a group
- Grant listed all the features that can worked on for final release
  - Having two ratings: Quality and difficulty
  - Tags e.g. lost of homework, tough marker, approachable
  - Thumbs up/down for reviews/comments
  - Ask users if they'd recommend a course
  - Show number of ratings
- Split tasks
  - Grant will be working on testing future functions
  - James has a heavy workload in other courses this week
  - Seth will work on bug fixes
  - Twino(Mathew), Logan, and Luciano will work on formal testing and documentation
  - Erikas will be given tasks by Seth



## Meeting 22 - Sprint Refinement 8 (Release 2) - April 1st 2021

- Erikas, Luciano, and Logan were not in the meeting
  - Luciano and Logan had informed ahead of time
- Seth demonstrated changes
  - Changes with account creation page
  - New notification when changing password, changing nickname
  - Demonstrated viewing and making comments
  - Demonstrated how courses are now visible in department page
- Discussed testing
  - Finalized formal UI testing
  - Researched python testing documentation format
- Discussed teamwork policy and instructor intervention
- Grant demonstrated new functions he made in final release testing branch
  - tags
  - thumbs up/down
  - profanity filter
  - increased password security
  - reset password email
- Twino(Mathew) discussed bugs and issues
- Decided to not have a professor demonstration 2

## Meeting 23 - Sprint Planning 9

- Plans for final release
  - Documentation - Email Prof - Twino(Mathew)
    - Which reports are necessary
      - Would a website need a user/technical manual or an installation manual?
    - Updated Requirements document or Final Report which includes updated requirements?
    - Should the document include a section about security
  - Presentation
    - 3 to 5 people
    - Should have a powerpoint
    - Should practice and have a list of talking points
  - Demonstration (5 to 10 min)
    - Seth or Grant with screen sharing
  - Extra marks
    - Deployment to Heroku
      - Seth will be working on it this sprint
    - Getting real users (if we get production up and running)
- New Features

- Dark mode - Erikas worked on it
- Password reset backend - Grant and James
- Backlog
  - tags - Logan will be creating more tags
  - thumbs up/down - Erikas created front end, Grant will continue to work on backend
  - would take course again, needs to be added as a question in front end when making review
  - number of ratings
  - password reset frontend
  - Prof should be able to rate students
  - Formal UI testing - Luciano
    - tags, thumbs up/down, number of ratings, password reset

## Meeting 24 - Sprint Refinement 9

- Discussed final document requirements
  - No user/technical manual or an installation manual
  - Final report should have everything together
  - Document should cover security and hosting/domain/certificate
- Final release document will be shared by Sprint Planning 10
- Progress
  - Grant fixed reporting, worked on tags (limit of 3 per review)
  - James worked on filtering options
  - Seth worked on hosting
  - Erikas worked on tags,
- Backlog
  - Thumbs up/down for comments
  - Show when a review was created
  - Tags should be in little pill boxes on the reviews
  - "Would take again" needs to be placed correctly
  - Work out kinks with hosting
- Preferred presentation date April 29th, 2021 at 4:00 pm

## Meeting 25 - Sprint Planning 10

- Everyone except Logan and Seth attended (both gave reasons ahead of time)
- Erikas may not be able to attend next sprint review
- Accessibility (mention briefly) - Erikas
  - Increased font size, darker, across the board
  - Division between categories
  - Extend width of search bar
- James worked on filtering for difficulty rating
- Grant - finishing up with things from backlog

- Munashe setting and getting tads finished
- Erikas worked more on dark mode, worked on tags (pills), fixed top 5 scaling
- Backlog
  - Reviewers recommendations, bold, add color
  - Add more separation for overall ratings
  - Add pagination to search results page
  - Pressing enter on search bar should bring you to search results page
  - Display difficulty on course/instructor/department page
  - Thumbs down and thumbs up indicators individually i.e 5 upvotes, 2 downvotes
  - When verifying account pressing enter doesn't work, only clicking on button

## Meeting 26 - Sprint Refinement 10

- Final report
  - Grant will work on Security section
  - Twino(Mathew) will work on Hosting section
  - Logan will get started on updated requirements
  - Decided to create new backlog section
- Hosting
  - Final hosting will be on Twino(Mathew)'s new remote server
  - r8scholar.ca domain needs to be pointed to new server
  - r8scholar.ca needs a certificate for SSL
- Split tasks/schedule
  - Erikas - Free
    - Styling, font, division between categories, extend search bar
    - Work on merging changes
    - Profile page when logging in, redirects to /signin, should be /login
  - Grant - Free after 17th
    - Merging new type of rating, thumbs up/down backend
  - Nash - Free after 17th - Decide tasks in next sprint
  - James - Free after 23rd - Decide tasks in next sprint
  - Luciano - Free after 17th - Decide tasks in next sprint
  - Logan - Free after 17th
    - Updated requirements
  - Twino(Mathew) - Busy but better after 17th
    - Add axios to server
    - Confirm domain is pointed at server
    - Add r8scholar.ca as an allowed host in apache server
    - SSL Certificate (later)
    - Push updated install.sh
  - Seth - Busy till 18th
    - Get domain to point at server

## Meeting 27 - Sprint Planning 11 (Review 10)

- Discussed backlog
- Previous sprint
  - Erikas
    - styling, font, division between categories, extend search bar
    - added number of reviews
    - Profile page when logging in, redirects to /signin, should be /login
  - Grant
    - Thumbs up/down (front end calls need to change)
  - Seth
    - Pointed r8scholar at server
  - Twino(Mathew)
    - Add axios to server
    - Confirm domain is pointed at server
    - Updated install.sh
  - Logan
    - Updated requirements
- Demo (3 people) (Finish by 22nd)
  - Grant, what we need to cover, order, script, assign
- Presentation (Google Slides) (30 min) (Finish by 26th)
  - Intro (Twino(Mathew))
  - Proposal/Requirements (Munashe)
  - UI Design phase (Erikas)
  - Database Design/Architecture (James)
  - Sprints/Agile (selected software development process) (Twino(Mathew))
    - Sprint ceremonies
  - (Seth, Luciano, Grant(refinement), James(refinement))
  - Release 1 (Luciano)
    - Backlog (What we planned on doing)
    - Backlog priority/ and why
    - Design/requirements changes
    - Problems we faced
      - Formal Testing
      - Longer sprints
      - Missing features
    - Completed backlog (What got done)
    - Release summary + document
  - Release 2 (Seth or Grant)
    - Backlog remaining (What we planned on doing)
    - Backlog priority/ and why
    - Who was assigned what
    - Design/requirements changes
    - What was added/removed to backlog

- Problems we faced
    - Bringing Erikas up to speed
    - Features with no front end ready
  - Completed backlog (What got done)
  - Release summary + document
- Final Release (Seth or Grant)
  - Backlog remaining (What we planned on doing)
  - Backlog priority/ and why
  - Who was assigned what
  - Design/requirements changes
  - What was added/removed to backlog
  - Problems we faced
    - Heroku
    - Deployment settings
    - SSL Certification for production domain
  - Completed backlog (What got done)
  - Release summary + document
- Formal Testing (Logan)
  - Automation
- Deployment (Twino(Mathew))
  - Server
  - SSL Cert
  - Installation
- Questions (15 min)
  - Front end (Seth)
  - Backend (Grant)
  - Documentation, testing
    - Testing (Logan)
  - Server (Twino(Mathew))
  - Security (Grant)
  - UI Design (Erikas)
  - Database Design (James)
  - Architecture (Luciano)
  - Fullstack (Munashe)
- Schedule
  - Intro speech (Finished by 22nd)
  - Demo (Finish by 22nd)
  - Presentation (Google Slides) (Finish by 26th)
- Backlog
  - settings.py (Seth, Grant)
    - Create user, localhost
  - SSL cert (Twino(Mathew))
  - Merging rating code (Grant)
  - Front end thumbs up/down (Erikas)

- Merging frontend Erikas branch (Erikas)
- Tests need to be run on r8scholar.ca (after deployment)

## Meeting 28 - Sprint Refinement 11

- settings.py
  - Seth will test out deployment ready settings.py locally
  - Twino(Mathew) will test the new setting.py on server by 24th
- SSL Certification will be ready by 23rd night
- Grant
  - Merged rating code
  - Rough script for demo ready
- Seth
  - Created list of main features per release
- Erikas
  - Thumbs up/down front end ready
- Demo
  - Demo should be using deployed r8scholar.ca if possible
  - Two people present demo
  - Need to create sample reviews prior to demo
- Final Presentation Slides
  - Discussed formatting
  - Discussed/decided who does what
  - Choose team tree version
  - Discussed backlog slides
  - Slide finalization meeting on 25th at 2:00 PM

## Meeting 29 - Sprint Planning 12 (Review 11)

- Planned out slide timings
- Checked all slides
- Checked documentation requirements
- Discussed production issues

## Meeting 30 - Presentation Preparation

- Decided how we want to allocate time between demo and presentation
- Scripted outline of presentation, rescripted demo as needed
- Practiced presentation and demo twice
- Decided if questions slide should include who would talk about what