nEdCom

(Web-based Educational Networking Platform for University Students)

Course: CSCI441_VA_F2018: 2018F_CSCI441_VA_Software Engineering

Team Name: HKMS
Team Members: Heanh Sok, Kanika Montha, Molika Meas, Sokchamroen Riem
GitHub Link: https://github.com/heanhsok/hkms-fhsu-se-project

Team Member Contribution

All team members contributed equally to this project.

Table of Contents

nEdCom 1

Team Member Contribution 2
1. Interaction Diagram 4
2. Class Diagrams And Interface Specification 11
3. System Architecture And System Design 15
4. Algorithm And Data Structures 20
5. User Interface Design And Implementation 21
6. Design of Tests 26
Project Management 31
Resources 32

1. Interaction Diagram

Interaction diagram shows the responsibility of system objects and how they interact with each other. Below are the interaction diagrams of the 6 use cases specified in the fully dressed description in report 1.

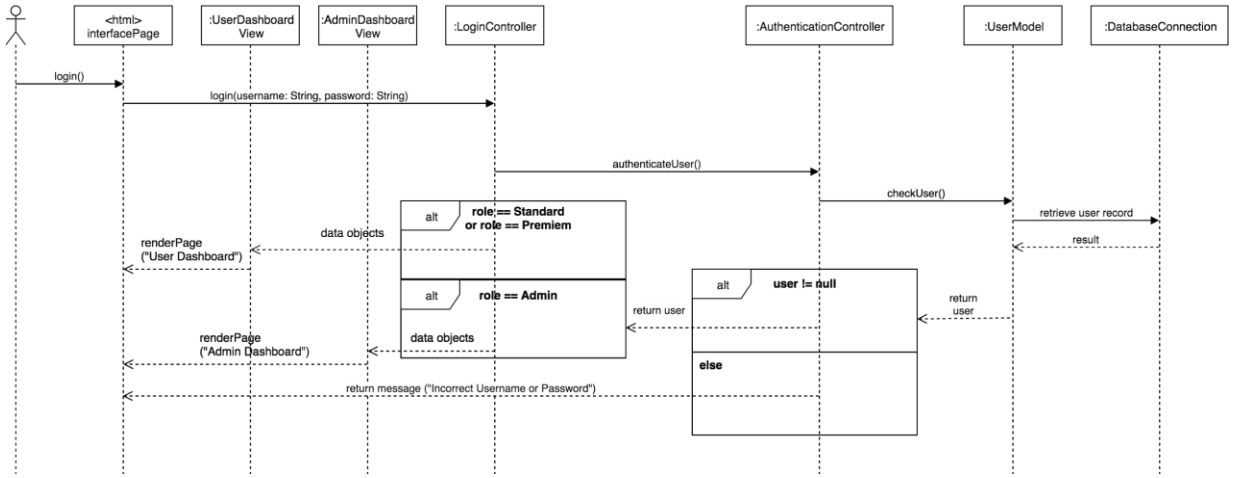
Design Architecture

It's worth noting that nEdCom application is built in laravel, a PHP framework that uses MVC (Model, View, Controller) architecture. View layer is responsible for user interface pages and displaying object data to the user in an appropriate format. Controller takes requests from the user, performs computations and logic, and returns objects to View. Model retrieves and send data to the database as needed by the Controller.

Design Principle

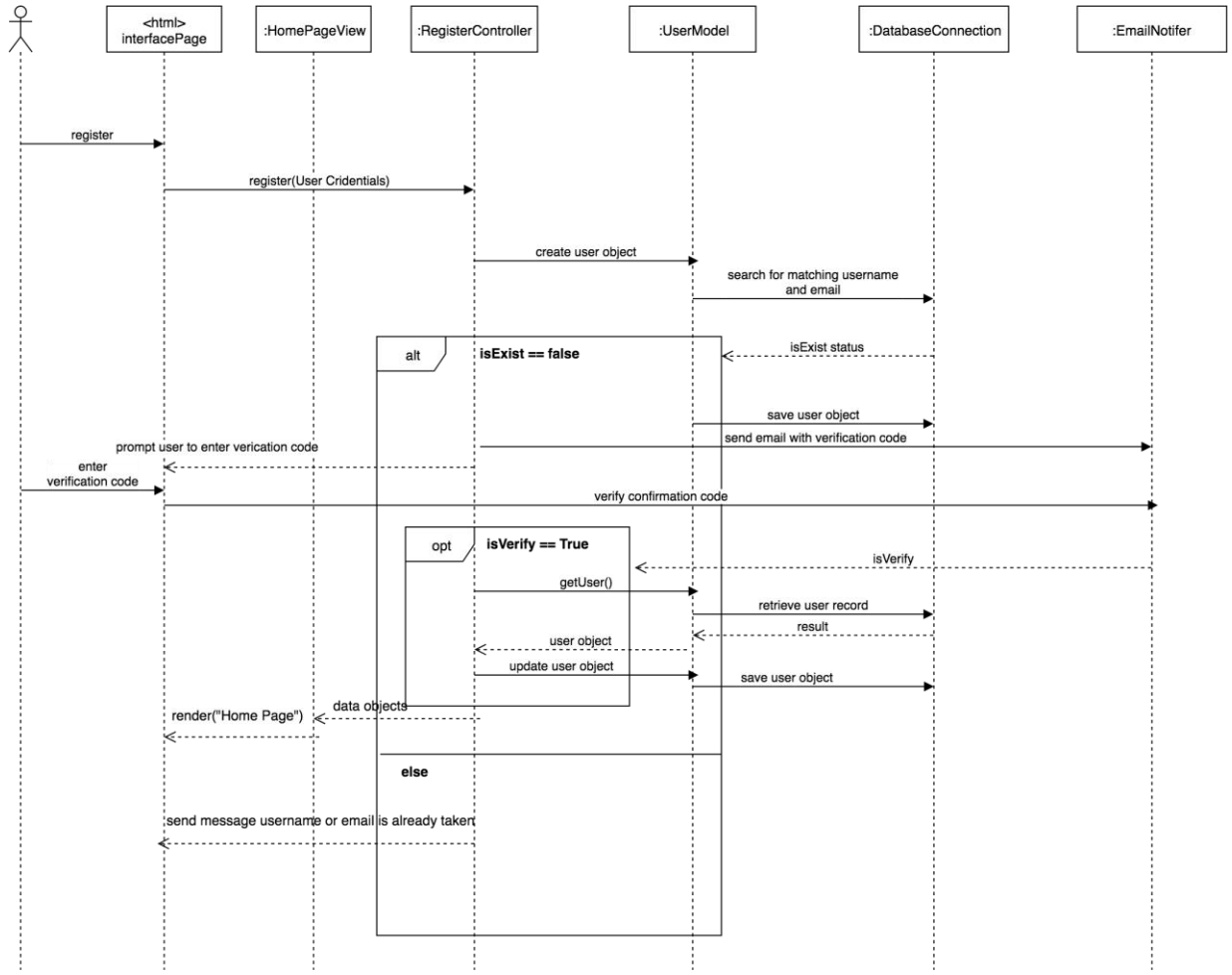
- Expert Doer: View layers such as UserDashboardView, AdminDashboardView, OpportunityPageView, ForumPageView focus rendering html pages. Authentication controller focuses on authenticating the user. Model layers such as UserModel, OpportunityPagePostModel, ForumQuestionModel, ForumAnswerModel focuses on retrieving records from database.
- High Cohesion: Each object is assigned specific computation responsibility. Example: LoginController checks role in user object and call the corresponding view layer file accordingly to render the html page.
- Low Coupling: Each object does not take on many communication responsibilities. For example, Authentication controller only communicates with LoginController and UserModel. UserModel only communicates with database and controller.

Login



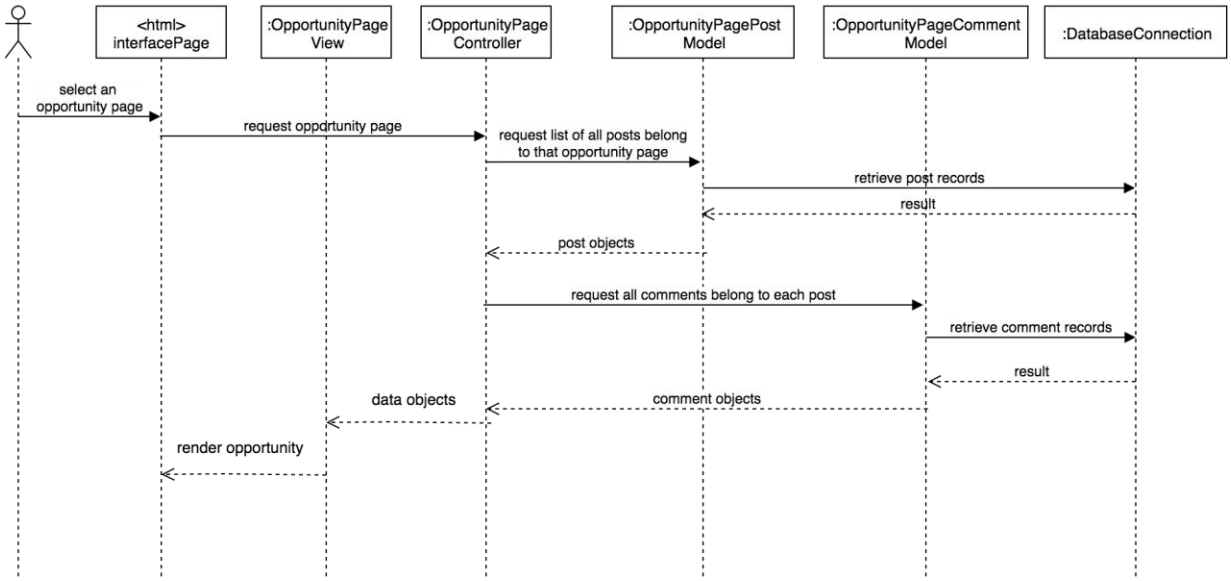
- The login function with username and password as input parameters in loginController is called when the user inputs the credentials and clicks on login.
- The loginController then calls the authenticateUser() function in AuthenticationController to authenticate the user.
- The AuthenticationController then calls the UserModel to find user with the given credential. The UserModel makes call to database to retrieve user record that matches the credential.
- The user object is returned to the AuthenticationController.
    - If the user object is not null, that means the user exists in our database. Therefore, the user object is returned to the LoginController.
        - Once the LoginController receives the user object, it checks for user role.
        - If user role is ‘standard’ or ‘premium’, it makes call to UserDashboardView to render the User Dashboard html page for user to see
        - If user role is ‘admin’, it makes call to AdminDashboardView to render the Admin Dashboard html page for user to see.
    - However, if the user object is null, that means the user does not exist in our database. The message “Incorrect Username or Password” is shown to the user.

Registration



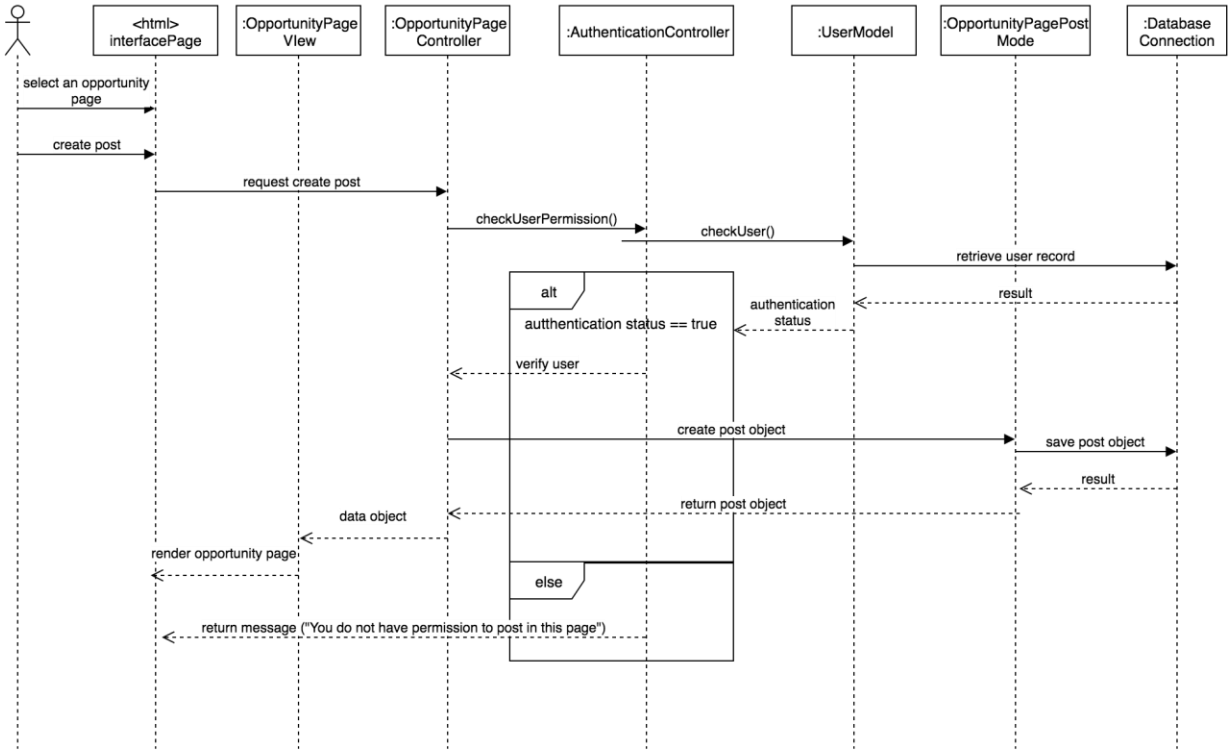
- RegisterController creates user object with the input information and pass it to UserModel. The user model then search for matching username and email to see if they are already taken.
    - If username and email are not taken. The UserModel saves the user object in the database. The RegisterController then requests the EmailNotifier to send verification code via email to the user. After the user submit the correct verification code, their verification status in the database will be updated.
    - If username or email are already taken, the message “Username or email are already taken” is shown to the user.

View Opportunity Page



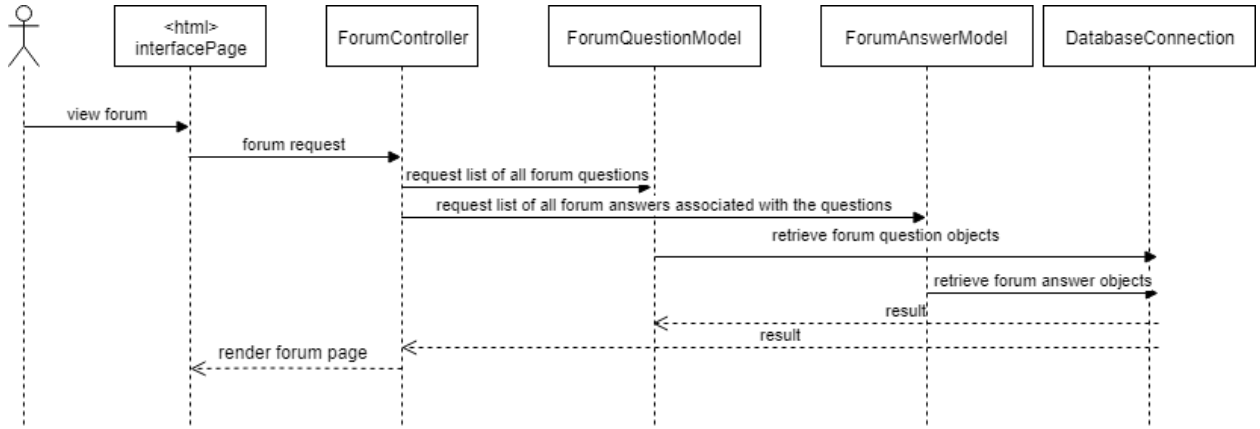
- Once the user selects on a specific opportunity page (competitions, scholarships, internships, career, or volunteering opportunities), the OpportunityPageController make calls to OpportunityPagePostModel to get all the posts belong to that opportunity page. OpportunityPageController also makes call to OpportunityPageCommentModel to get all comments belong to the each of the post. The two models retrieve the requested data from the database and return them to OpportunityPageController. The controller then returns the data to the OpportunityPageView to render the html page for user to see.

Post in Opportunity Page



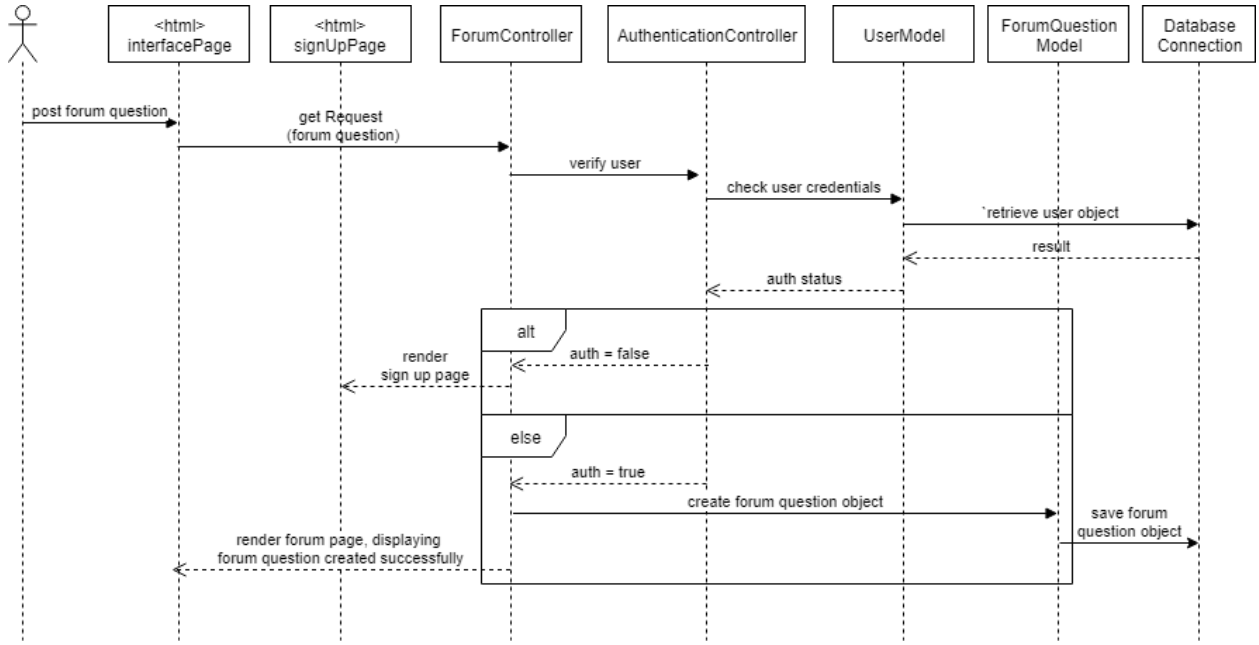
- Once the user selects a specific opportunity page (competitions, scholarships, internships, career, or volunteering opportunities) and select create post, the OpportunityPageController will make calls to the AuthenticationController to check whether the user has permission to post or not.
    - If the user has permission to post, the OpportunityPage creates the post object and let the OpportunityPagePostModel save the record in the database.After that, the OpportunityPageController returns the successfully created post to the OpportunityPageView to render the html page for user to see.
    - If the user does not have permission to post, the message “You do not have permission to post in this page” is shown to the user.

View Forum



- Once the user opens the forum page, the ForumController make query call to the ForumQuestionModel to request list of all forum questions and the ForumAnswerModel to request list of all forum answers associated with each forum questions. Then both the models retrieve the data from the database. The database return results as queried, and the ForumController take the data objects to display to the forum page.

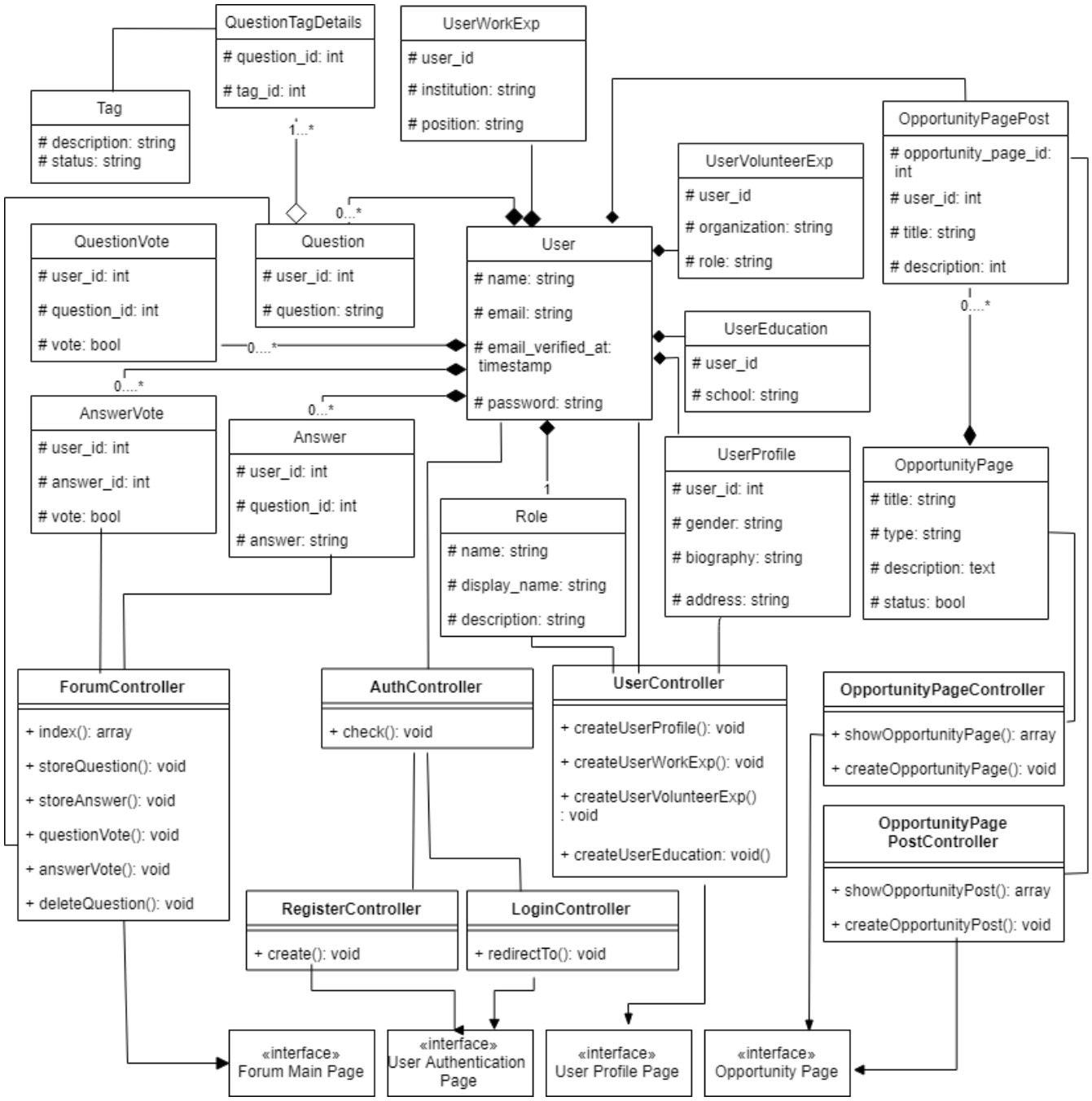
Post in forum



- When user submits a forum question, the ForumController obtains the input data from the interface page, then checks with the AuthenticationController to verify the user. The AuthenticationController asks the UserModel if user already has an account. The UserModel take the request and retrieve user object from the database. Then the database return the result as queried. The UserModel take the result and return the user's auth status to the AuthenticationController.
    - If auth is false, user doesn't have an account yet. The ForumController will render the signup page for user to register for an account.
    - If auth is true, user already has an account and is eligible to post forum question. ForumController takes the input data and request ForumQuestionModel to create new forum question object which is saved to the database. Then ForumController renders the forum page so that user can see the forum question they have posted.

2. Class Diagrams And Interface Specification

a. Class Diagram



b. Data Types And Operation Signatures

User: a class that contains user account information

- Attributes:
  # string name
  # string email
  # string password

Role: a class that contains details associated with user roles.

- Attributes
  # string name
  # string display_name
  # string description

UserProfile: a class that keeps details of user profile information that displays in user profile page

- Attributes
  # int user_id
  # string gender
  # string address
  # string biography

Question: a class that keeps details of forum questions

- Attributes
  # int user_id: id associated with user who post the question; related to user.user_id
  # string question

Tag: a class that contains all forum tags

- Attributes
  # string description
  # string status: tag status, active/inactive

QuestionTagDetails: a class that matches forum question and tags

- Attributes
  # int question_id: id associated with forum question; related to question.question_id
  # int tag_id: id of tag; related to tag.tag_id

Answer: a class that keeps details of forum answers

- Attributes
  # int user_id: id of user who posted the answer; related to user.user_id
  # int question_id: id of question to which the answer belongs; related to question.question_id
  # string answer

QuestionVote: a class that keeps upvote/downvote details for forum questions

- Attributes
  # int question_id: id of the question that gets voted
  # int user_id: id of user who clicks upvote/downvote on the question
  # bool vote: vote status; 1 if upvote; 0 if downvote

AnswerVote: a class that contains upvote/downvote details for forum answers

- Attributes
  # int answer_id: id of the answer that gets voted
  # int user_id: id of user who clicks upvote/downvote on the answer
  # bool vote: vote status; 1 if upvote; 0 if downvote

OpportunityPage: a class that contains information about opportunity page

- Attributes
  # string title
  # string type
  # string description
  # string status

OpportunityPageController: a class that handles opportunity page operations

- Operations:
  +array showOpportunityPage(): return list of all opportunity pages to the interface page
  + void createOpportunityPage(): create a new opportunity page and inserts to the database

ForumController: a class that handles forum operations

- Operations:
  + array index(): function that return list of all forum questions to display in forum page
  + void storeQuestion(): function to store question details
  + void storeAnswer(): function to store answer details
  + void questionVote(question_id; int): function to store user votes associated with the forum question with the given id
  + void answerVote(answer_id: int): function to store user votes associated with the forum answer with the given id
  + void deleteQuestion(question_id: int): function to delete forum question associated with the given id

UserController: a class that handles user profile operations

- Operations
  + void createUserProfile(): function to create user profile information
  + void createUserWorkExp(): function to create user work experience information
  + void createUserVolunteerExp(): function to create user volunteer experience information
  + void createUserEducation(): function to create user education information

RegisterController: a class that handles user account registration

- Operations
  + void create(): function to register new user account

c. Traceability Matrix

	Classes												
Domain Concepts	DC	RC	UC	QC	AC	QS	AS	TS	OS	OC	FC	UC	RC
Database Connection	X	X	X	X	X	X	X	X	X				
User Account Storage	X												
User Profile Storage		X	X										
Forum Question Storage				X									
Forum Answer Storage					X								
Point System						X	X						
Forum Tag Storage								X					
Opportunity Page Storage									X				
Opportunity Page creator										X			
Forum Question Creator											X		
User Profile Creator												X	
User Account Creator													X

The diagram traces how the classes evolve from the domain concept model. Some domain concepts that have low priority weight in requirement have not yet been implemented as a class and to be deferred for future implementation.

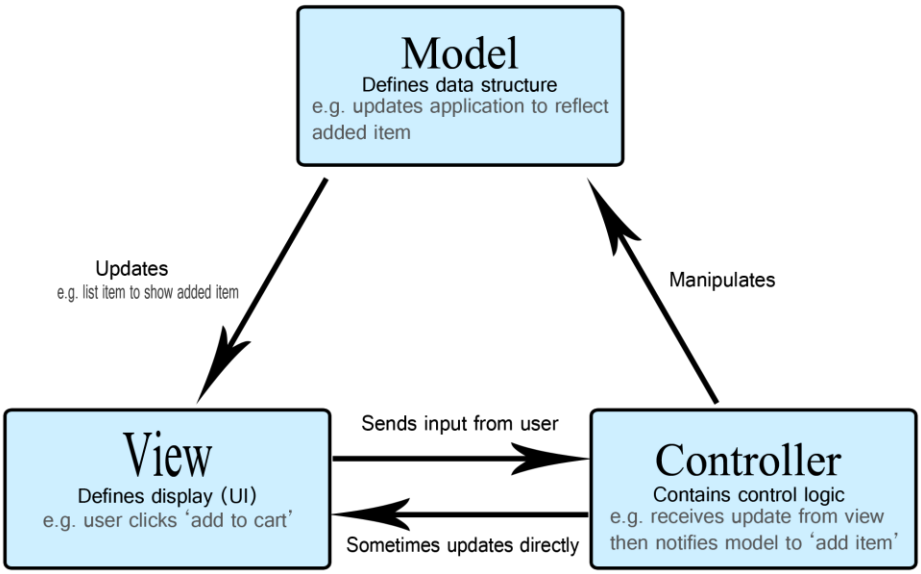
Model classes made to be correspond with the concept model includes User, Role, UserProfile, Question, Answer, and the OpportunityPage. For the point system, it requires

classes to be split into QuestionVote and AnswerVote to keep track of votes (points) user obtains from both their forum questions and answers.

3. System Architecture And System Design

a. Architectural Styles

nEdComm web application is implemented using Model-View-Controller (MVC) architectural pattern. As the name itself suggests, the MVC architectural pattern separates the application logic into three layers such as Model layer, View layer, and Controller layer.



Source: [developer.mozilla.org]

The View layer defines the presentation of data. It is responsible for rendering user interface pages and displaying data objects to the users in an appropriate format.The Controller defines logics that update the model or view in response to user interaction. It takes requests from the user, performs computations and logic, and returns data objects to View layer. The Model layer defines the data structure of the application. It retrieves data from database and persist data to the database. It transfers data to and from controller as object instances.

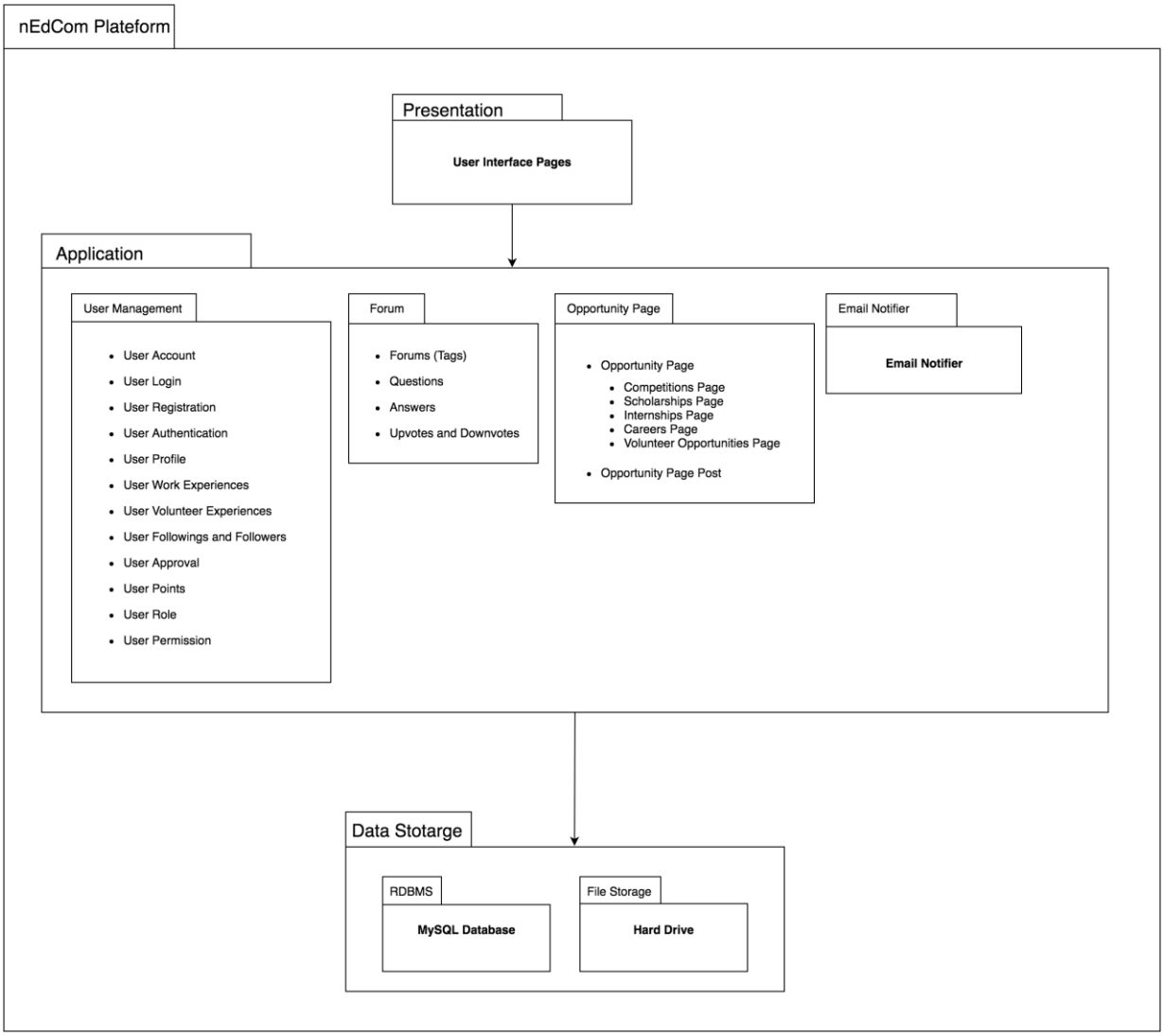
In addition to the Model, View, Controller layers, we also have additional components such as Router, Database Migration, Database Seeder, and Database Factories in building this applications. Router is used for managing all the URLs (Universal Resource Locator) in our application. Database Migration is used for defining database schema and creating all the tables in our database. Database Seeder is used for add initial set of data or testing data into our database. In our case, we use Database Seeder to insert user roles such as 'standard', 'premium', and 'admin' to our database so we can use one of the role when creating new user. Database Factory is used for generating large records of dummy data into our database.

This architectural pattern enable our code base to be modular and reusable. Moreover, it promotes good design principles such as Expert Doer (each layer is expert in their responsibilities), High Cohesion (each controller is assigned specific computation responsibility), Low Coupling (each component does not take on many communication responsibilities). Additionally, this architectural pattern enables software engineer to efficiently split roles for developer in the project and enable them to work simultaneously. Backend developers can work on database schema in Model layer and programming the application and business logic in the controller layer. At the same time, frontend developers can work on designing and programming the user interface in the view layer.

b. Identifying Subsystems

Our system contains the following subsystems:

- User Management is used for managing user account, user profile, user activities on the platform.
- Forum is used for managing all the forums page (tags), all the posts/questions by users in each of the forums as well as all the answers/comments to posts/questions. It also manages all the upvotes and downvotes made by users to posts/questions and answers/comments.
- Opportunity Page is used for managing all type of opportunity pages such as Competitions Page, Scholarships Page, Internships Page, Careers Page , Volunteer Opportunities Page as well as all the posts in each of those opportunity page.
- Email Notifier is used for sending verification code to users via email,



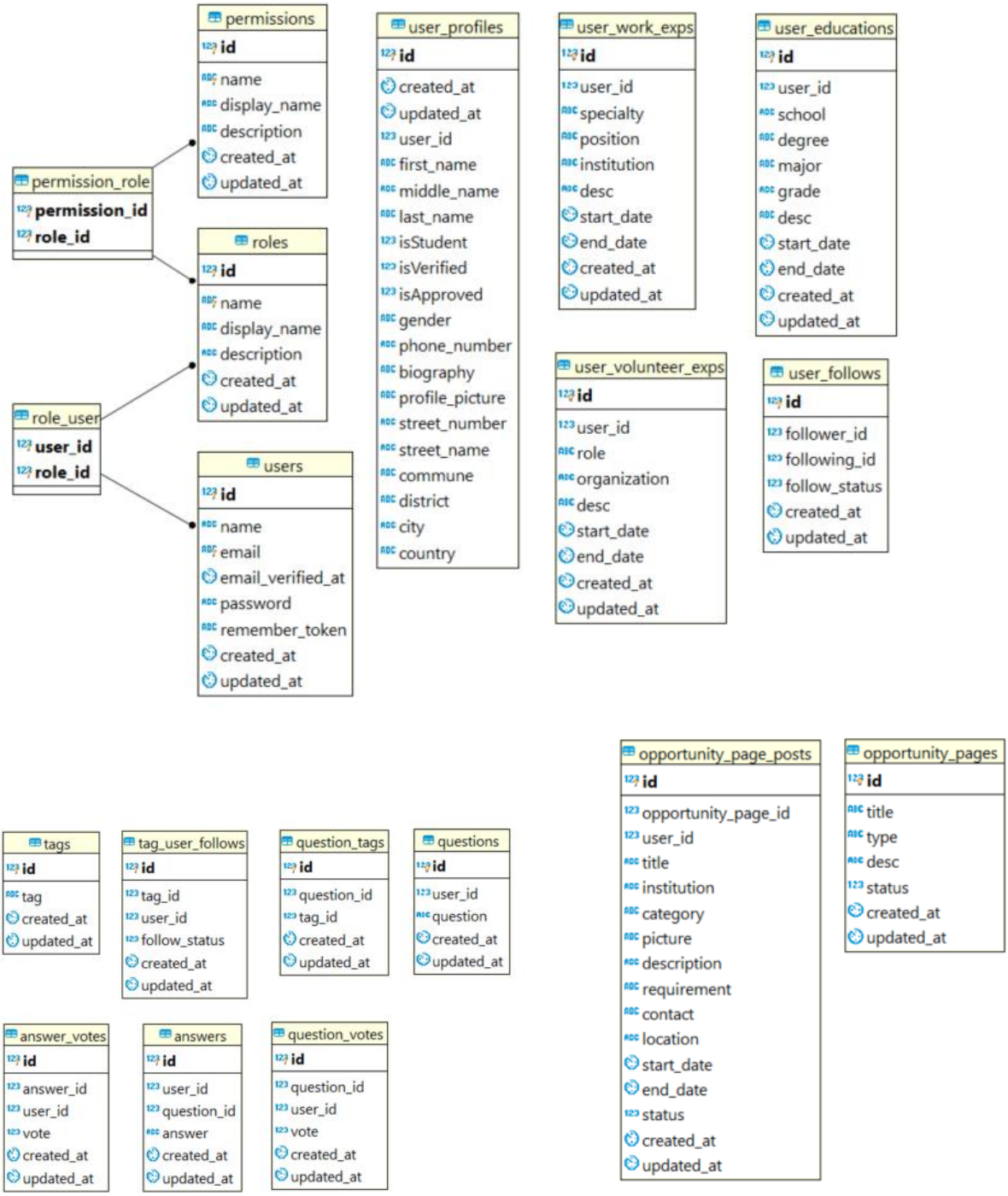
c. Mapping Subsystems to Hardware

All subsystems except Email Notifier will be running on the Apache Web Server that runs on a Linux Server. For Email Notifier subsystem, it will be implemented through a third party email service API (Application Programming Interface) such as mailgun that will be run on a server separate from our main server.

d. Persistent Data Storage

In term of storage, all textual data are stored in MySQL relational database which runs on the same Linux server as the application. All the photos such as JPEG, PNG, GIF and other multimedia formats are stored as files on the same Linux Server as the main application.

Database Schema



e. Network Protocol

Our Application uses HTTP (Hypertext Transfer Protocol) to enable communication between clients and server. This network will allow request and response to be send back and forth between client (user internet browsers such as Chrome, Firefox, Edge, Safari,..etc running

on devices like Smartphone, Computer, Tablets...etc) and server (nEdCom laravel web application). HTTP methods such as GET, POST, PUT, DELETE are used.

f. Global Control Flow

a. Execution Orderliness: The system is event driven because it performs certain actions in response to the user interaction or input.
.

b. Time Dependency: The system is designed to be real time. When users input information into our platform (for example, posting a question in a forum), those information are recorded into the database immediately. However, user needs to refresh their browser in order for the system to retrieve the latest information and display to them.

c. Concurrency: Multiple threading will be implement in future version of the platform especially when we scale the platform to support large number of users accessing the platform simultaneously.

g. Hardware Requirements

a. For client side, End Users need to have electronic devices capable of running modern internet browsers in order to user nEdComm platform.
b. For server side, nEdComm will be running on a Linux server
    i. At testing stage, the server needs to have at least
        - 1GB of Memory
        - 20GB of Storage Space
        - High speed Internet connection with large bandwidth
        - 2 Core CPU
    ii. At production stage, the server needs to have at least
        - 8GB of Memory
        - 100GB of Storage Space
        - 4 Core CPU
        - High speed Internet connection with large bandwidth

4. Algorithm And Data Structures

a. Algorithms

The system implements the following algorithms:

- User Points: Point is a critical part in our platform for determining the credibility of each user. We want to encourage users to actively engage in our platform so system award 1 point to the user when the user's post/comment get upvoted by other users. At the same time, we do realize that sometime user make mistake, so we don't want to penalize too much point when the user's post/comment get downvoted. Thus, we only deduct 1 multiple by weight (in this case 0.1) point when that happens. The algorithm works the following way:

  ```
  if user_post get upvoted
      add 1 point to the user
  if user_post get downvoted
      minus (1 x 0.1) point from the user
  ```

  The algorithm is the same for user_comment.
  For simplicity, even though points are stored with all digits in our database, when displaying the point to user, we show them the number that is rounded to the nearest integer, eg. 1, 2, 3, 4.
- Permission to access each page based on User Points
  - To provide incentive for users to acquire more points on our platform, some exclusive opportunity pages and forum pages also have minimum point for entry. Therefore, before allowing user to view each of those pages, we check the following condition first:

    ```
    if user_point >= page_minimun_entry_point
        grant access to user
    else
        deny access
    ```
- Encryption: Our system use Bcrypt hashing algorithm (provided by Laravel) to hash users' password before recording them into the database.
- Serialization: We also serialization algorithm such as toArray and toJSON (built-in with Laravel) to convert Model objects to array or json format respectively.
- Sort: For sorting records, the system uses mysql ORDER BY clause to sort the records ascending or descending.
  Ex: SELECT FNAME, LNAME FROM USER ORDER BY LNAME DESC;
- Search: For searching, the system mysql WHERE clause to select records that match specific criteria.
  EX: SELECT FNAME, LNAME FROM USER WHERE FNAME like '%am% AND LNAME = 'Cruise';

b. Data Structures

We use data structures such Arrays, Object, Stack, Set, Map, all of which are provided natively by PHP and Laravel. On the database side, we store data in a form of rows and columns inside tables of mysql relational database with primary keys and/or foreign keys in each of the table.

5. User Interface Design And Implementation

5.1 User Interface:

5.1.a. Homepage

Homepage interface for login and non-login users is the same. The difference is only when users want to interact with posts; those includes comment, vote, report, post question or other opportunities. In order to get access to those functions, non-login users will be asked to login (if already have an account) or sign up (if no account).

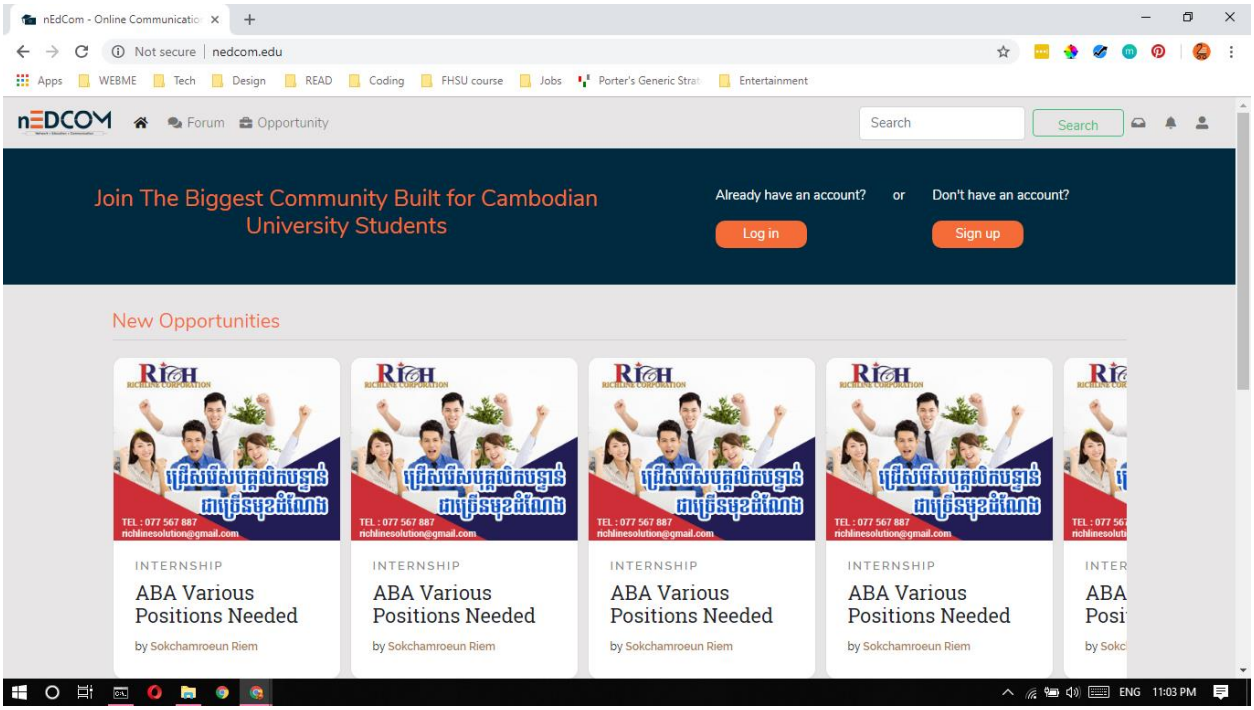


Figure 5.1.a(1): Homepage

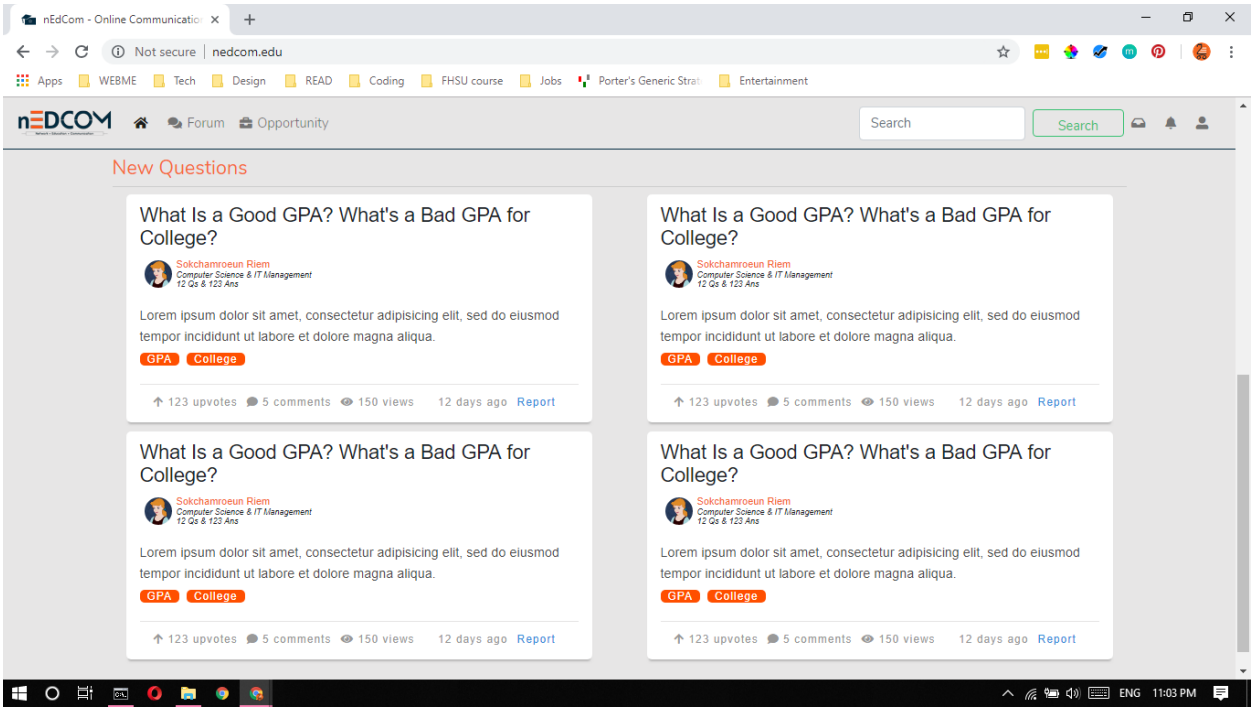


Figure 5.1.a(2): Homepage

5.1.b. Signup Pages

5.1.b.1. Standard User

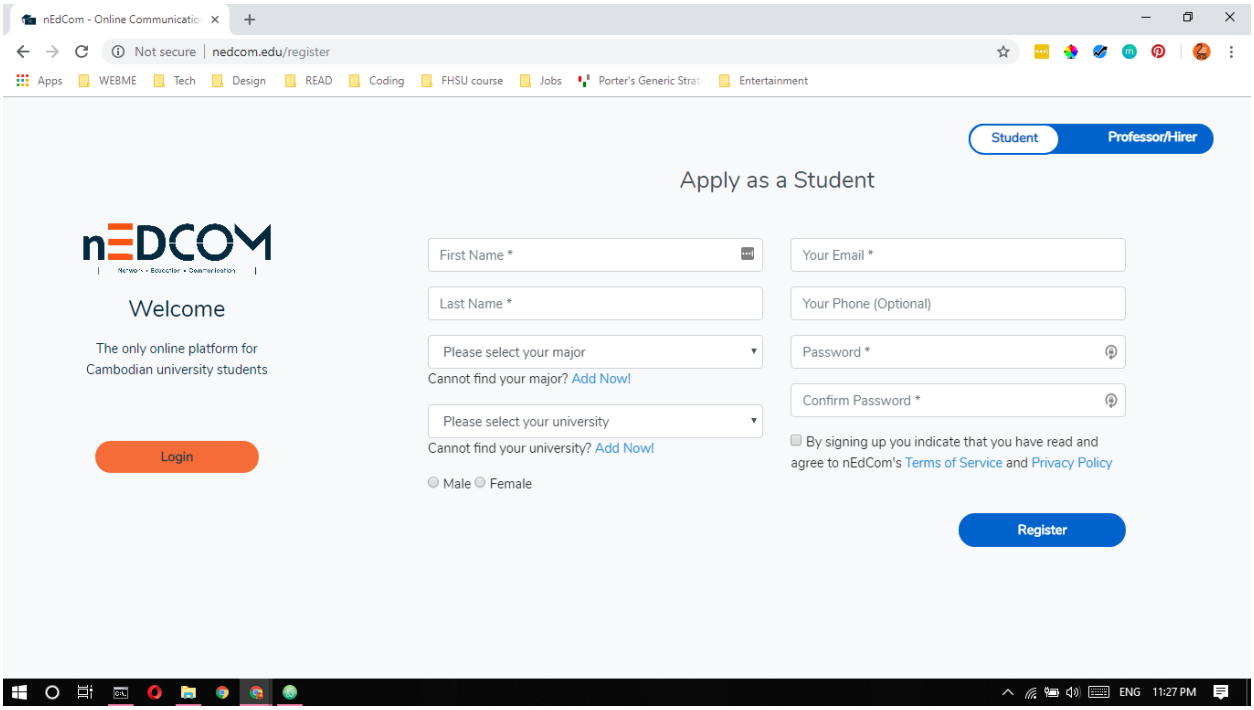


Figure 5.1.b(1): Register page for standard users

5.1.b.2. Premium User

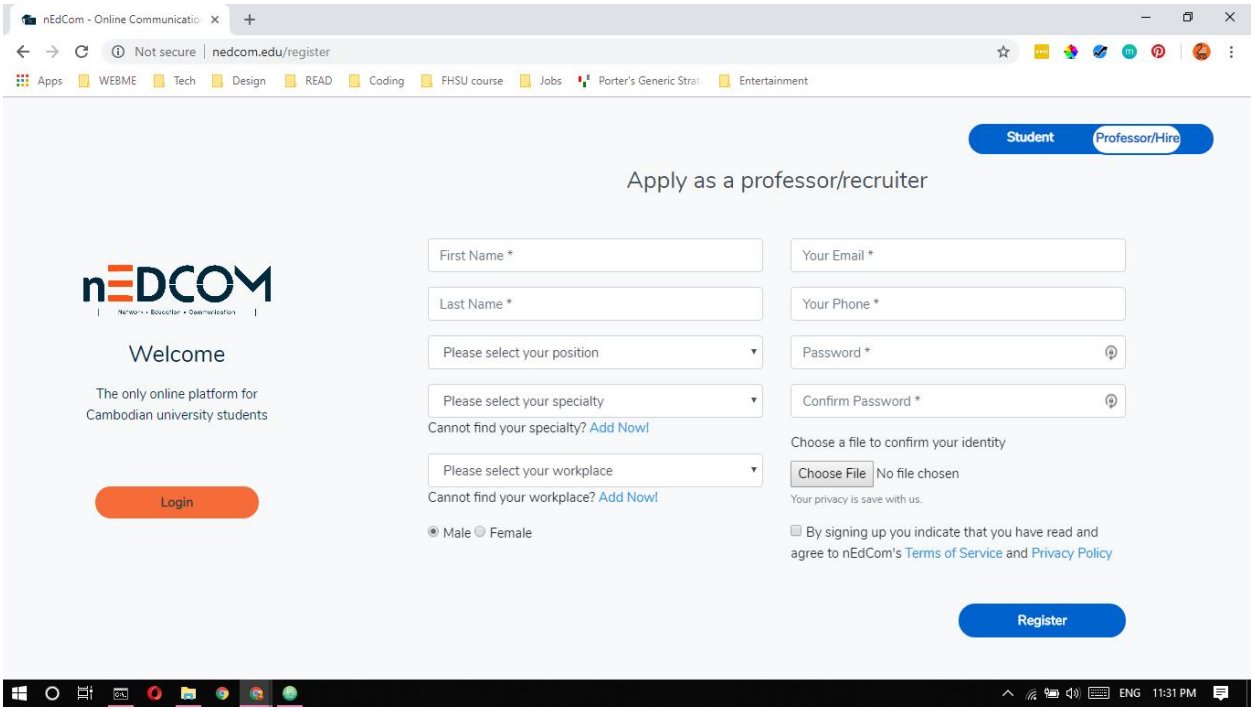


Figure 5.1.b(2): Register page for standard premium users

5.1.c. Login Page

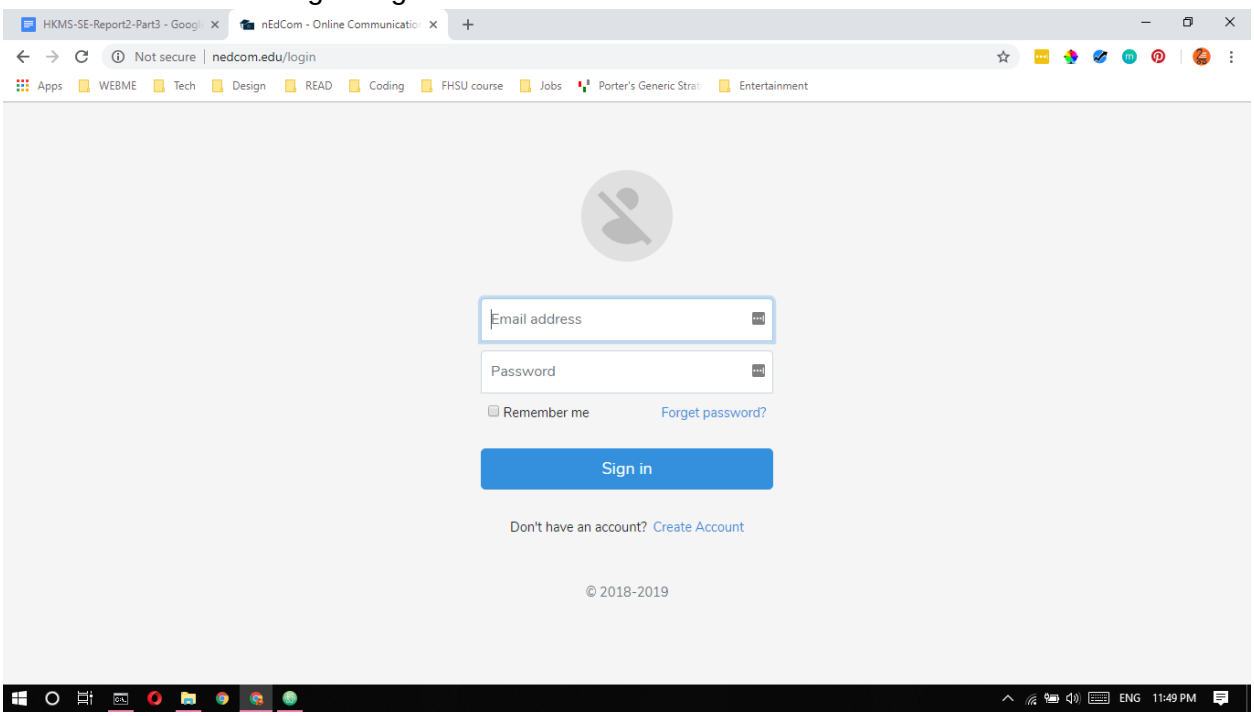


Figure 5.1.c: Login page for all users

5.1.d. Forum homepage

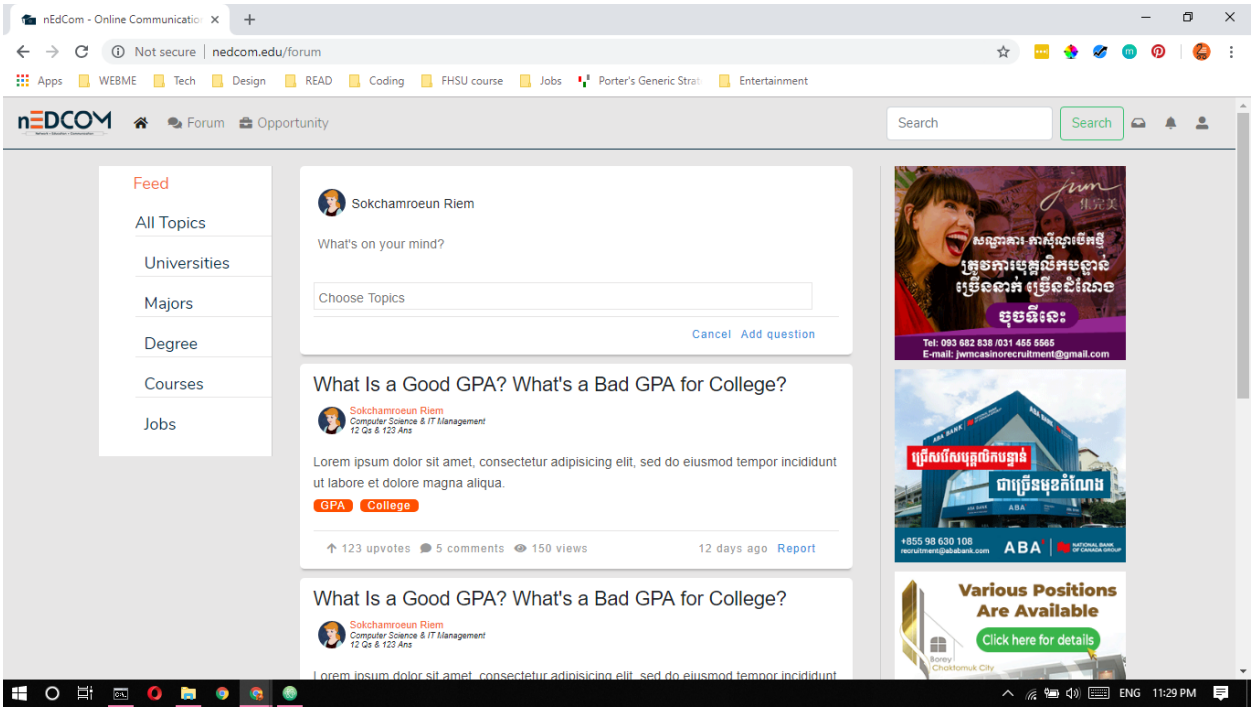


Figure 5.1.d: Forum homepage

5.1.e. Opportunity homepage

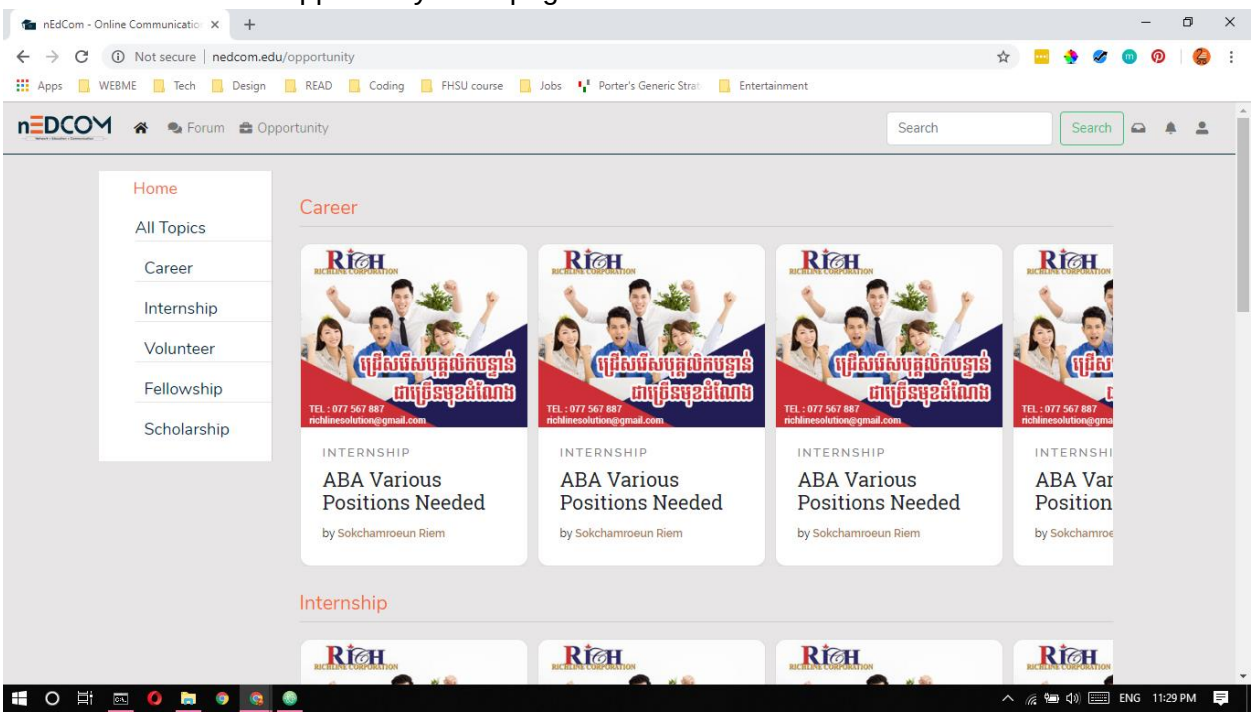


Figure 5.1.e: Opportunity homepage

User Profile Page

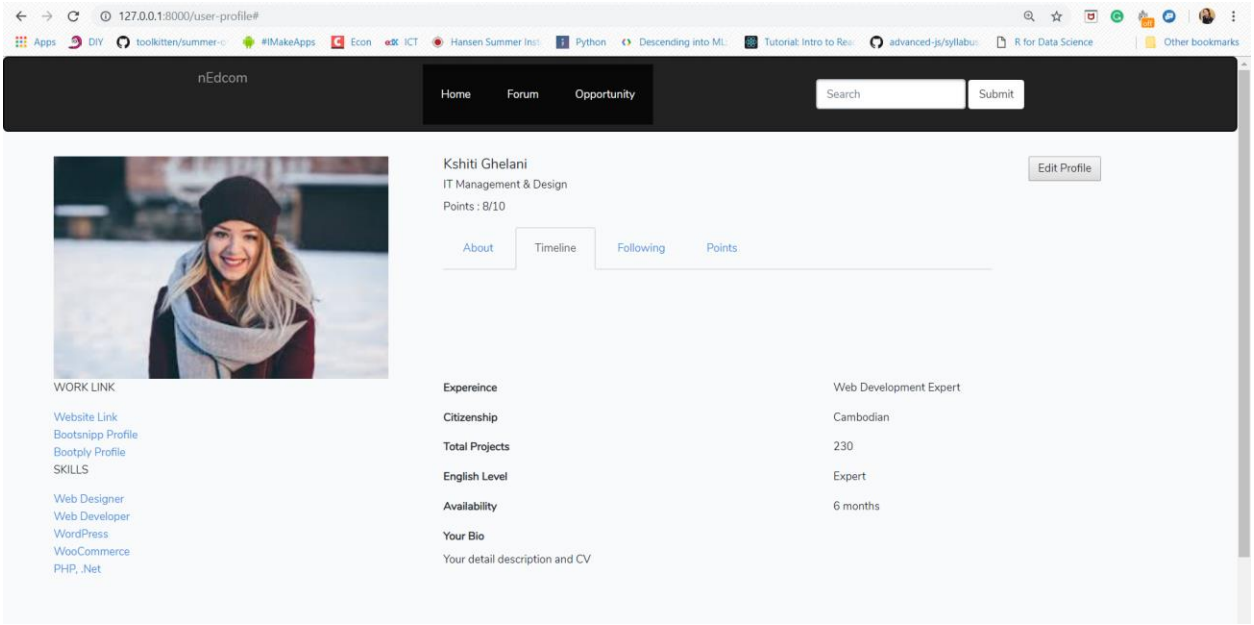


Figure 5.1.f: User Profile Page

Edit Profile Page

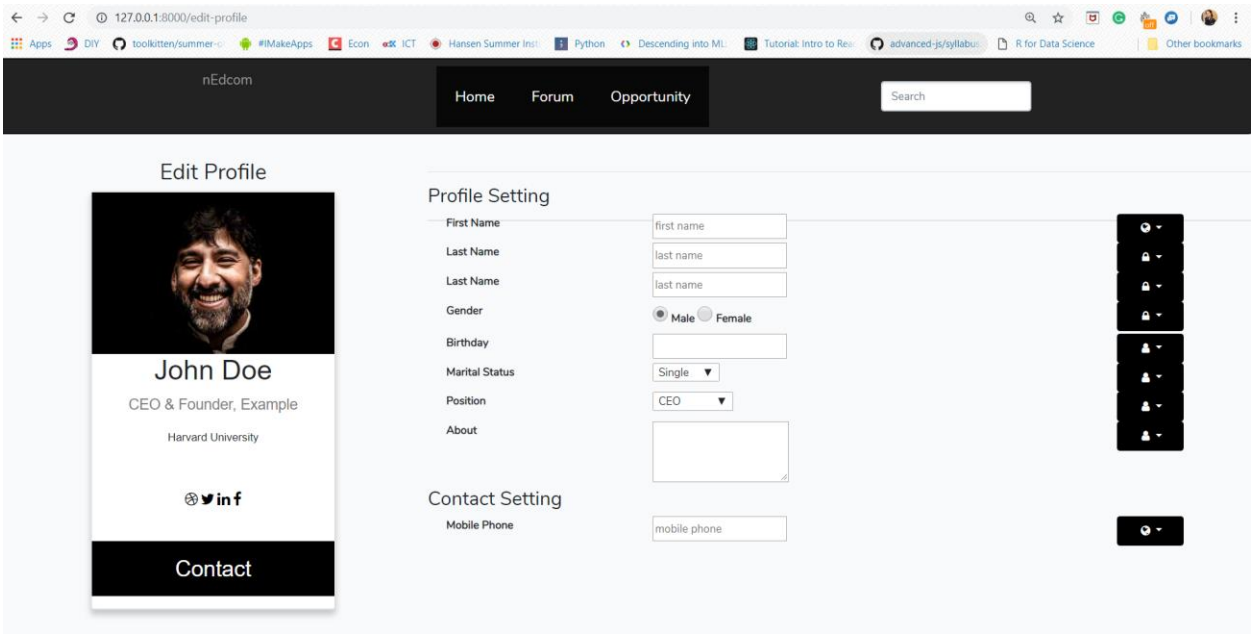


Figure 5.1.g: User Profile Editor

6. Design of Tests

Integration Testing

Following the test-driven development practice, tests are designed on the system to find bugs and identify them. For this system, integration testing strategy is used. Small units are tested first, with all the scenarios. Upon finish testing each module, integration test will be performed by combining all testing units into one system component to ensure all the smaller components work properly together.

User Acceptance Test Cases

The following tests are created based on the use cases, which are part of the system requirements from report 1:

- Register Standard User
- Register Premium User
- Login
- Post in Forum
- Post in Opportunity Page

Test Case Identifier: TC1-Register Standard User
Pass/Fail Criteria: The test passes if the user enters an email that is not existed in the database yet and has a correct email format, a first name, last name, password consisting of 6 or more characters and the confirm password matches
Input Data: First name, last name, email, password, retype of password

Test Procedure	**Expected Result**
Step 1. Enter valid email, first name, last name and incorrect confirm password	System alerts user of wrong confirm password and prompts user to retype the password
Step 2. Enter valid email, first name, last name and matching confirm password	System sends confirmation email to user and notify user to click on the link in the email
Step 3. Click on the confirmation email	System saves the account to the database, notifies user that account has been created successfully and directs user to homepage

Test Case Identifier: TC2-Register Premium User
Pass/Fail Criteria: The test passes if the user enters an email that is not existed in the database yet and has a correct email format, a first name, last name, institution, profession, specialty, password consisting of 6 or more characters, a matching confirm password, upload of proof of employment
Input Data: First name, last name, institution, profession, specialty, email, password, retype

of password, upload of proof of employment	
Test Procedure	**Expected Result**
Step 1. Enter valid email, first name, last name, institution, profession, specialty, and incorrect confirm password Step 2. Enter valid email, first name, last name, institution, profession, specialty, and matching confirm password Step 3. Clicks on the confirmation email Step 4. Admin clicks approve user	System alerts user of wrong confirm password and prompts user to retype the password System sends confirmation email to user and notify user to click on the link in the email System saves the account to the database, notifies admin of premium account in need of approval, alerts user that an account has been created yet awaiting for admin approval and directs user to homepage System update user account to premium in the database, alerts user that premium registration has been approved

Test Case Identifier: TC3-Login **Pass/Fail Criteria:** The test passes if user enters credentials contained in the database with less than number of allowed unsuccessful attempts. **Input Data:** Email, Password	
Test Procedure	**Expected Result**
Step 1. Enter valid email and incorrect password Step 2. Enter valid email and matching password	System alerts user of authentication failure, record number of unsuccessful attempt and prompt user to retype login credential System directs user to homepage

Test Case Identifier: TC4-View Forum **Pass/Fail Criteria:** The test passes if user clicks on the forum page and see list of all forum questions **Input Data:** Forum page link	
Test Procedure	**Expected Result**
Step 1. Click on the forum link	System display forum page to user with

	listing of all forum question

Test Case Identifier: TC5-Post In Forum
Pass/Fail Criteria: The test passes if user fills in a forum question header, forum question detail, select one or more tags
Input Data: Forum question header, forum question detail, question tag(s)

Test Procedure	**Expected Result**
Step 1. Enter question header, question detail, but select no tag Step 2. Enter question header, question detail, and select one tag	System alert user of unsuccessful question post and prompt user to select one or more tags System saves the question to the database, notifies user that question has been created successfully and direct user to forum homepage

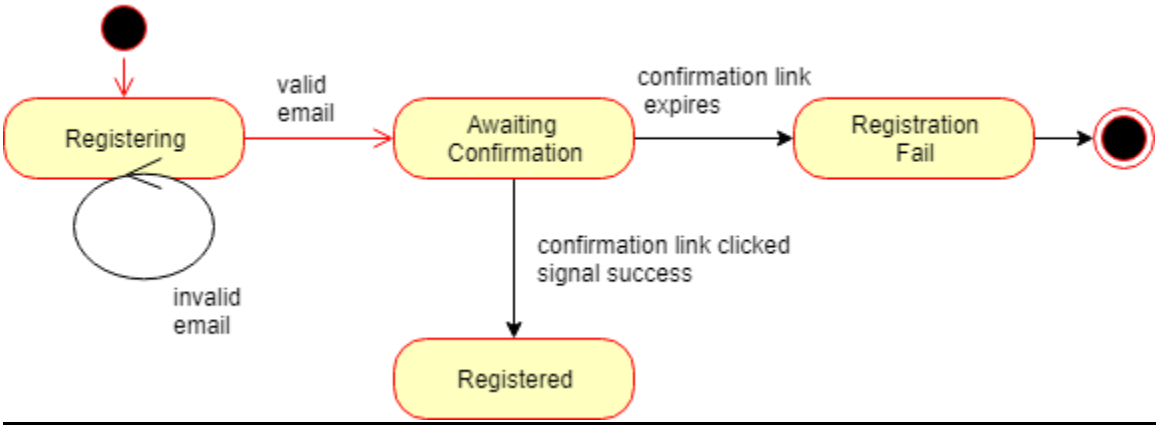
Test Case Identifier: TC6-Post In Opportunity Page
Pass/Fail Criteria: The test passes if user fills in opportunity header, opportunity description, image, and selected a category
Input Data: Opportunity header, opportunity description, image, category

Test Procedure	**Expected Result**
Step 1. Enter header, description, select a category, upload file of wrong format Step 2. Enter header, description, select a category and upload image file with correct format	System alert user of wrong image format and prompts user to select again System saves the post to the database, notifies user of the post creation and direct user to opportunity homepage

State Diagram

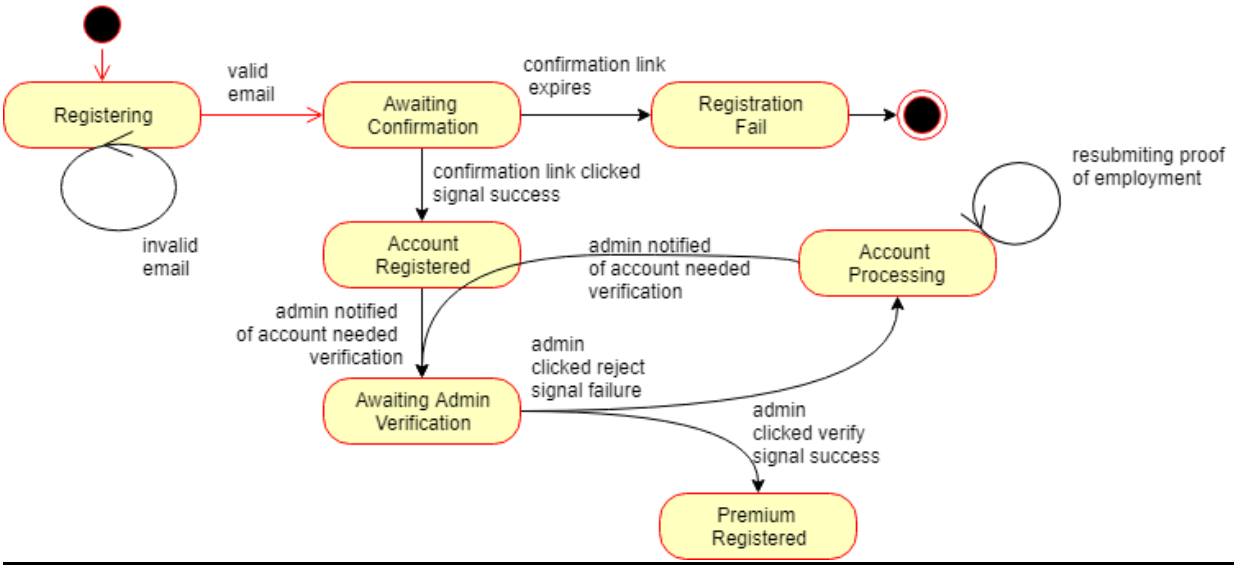
- Registering Standard Account

The RegisterController class will be tested. The Controller is responsible for verifying and creating all user role account. The user will need to enter required personal information and a valid email, then click on the confirmation link that they receive. If user enters an invalid email, they won't get to the next state of confirmation in progress, or if user fail to click on the link within a predetermined period, the account won't be created.



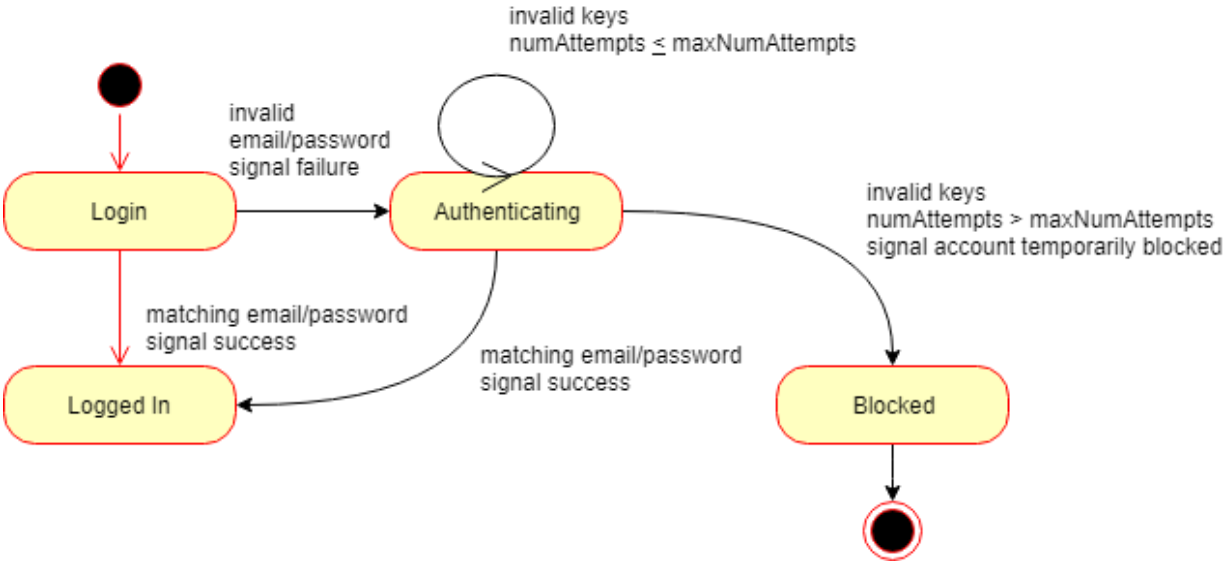
- Registering Premium Account

This test requires more additional steps to Registering Standard User, since it needs the system admin to review and approve the request before the user account get registered as premium. Again, RegisterController will be tested. It should update user role to premium once the admi click on approve of premium registration.



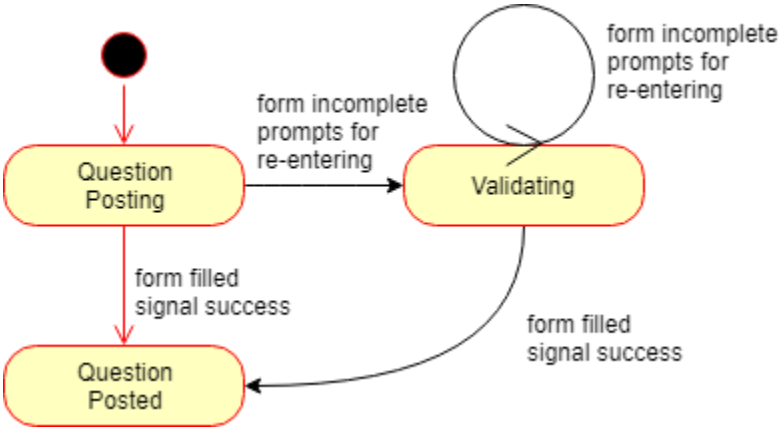
- Login

This test covers authentication of existing user account. LoginController class will be tested for verifying user while adding a layer of security by preventing user from entering wrong login information for a predetermined number of times. After series of unsuccessful attempts to login, the system will automatically block user from the system for a predetermined period of time.



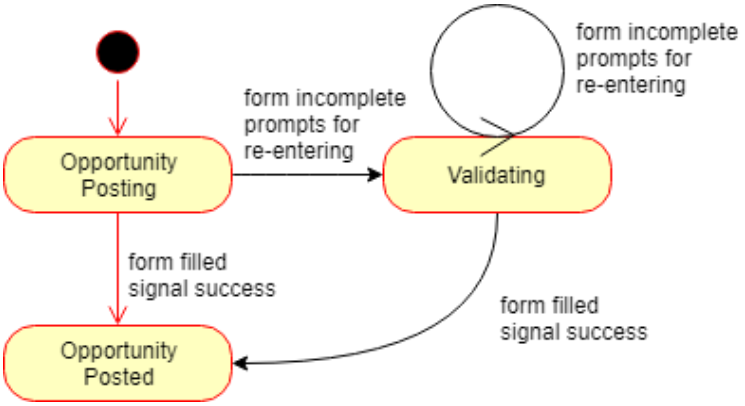
- Post Forum Question

ForumController class will be tested for the workings of forum. The Controller validate the form filled by user and prompts user to enter again if not all required fields are completed. The class saves the data into the database once it validates the data.



- Create Opportunity Post

In Creating Opportunity Post, OpportunityPagePostController will be tested. It receives data from user, validate that all required fields follows the required length and format, then the post shall save to the database. If user upload files that are too big or file that is not in the range of accepted format, user will need to select a new file.



Project Management

- Molika Meas
  Backend – Forum Management
- Heanh Sok
  Backend – User Management, Opportunity Page Management
- Sokchamroeun Riem
  Frontend – User Account, Forum, and Opportunity Page Interface
- Kanika Montha
  Frontend – User Profile, Forum Interface

In the process of developing nEdCom all team members had put out equal contribution. The team conducts weekly meetings to ensure update of work progress and responsibility division. Two members focus on front-end design, crafting logos and producing responsive, interactive webpages. In the meantime, the other two work on the backend components such as designing the computing logic, classes, functions, and the database structure. Finally, the front-end and back-end codes are integrated, and we run unit tests on each component, making sure that each small module work fine, before testing the whole system. As we build the platform, do more testing, and analyzing, we might add, remove, or change priority of some functionality, so the system will evolve overtime.

Implementation Progress & Future work

For the beginning stage of the system, we decide to focus more on the user account registration, login, user account management because it is fundamental for our system to work. Once we have a solid user management part working, we progress to work on the forum management.

Due to time constraint, the Opportunity Page Management part is to saved for future work.

Tasks	27-Sep	2-Oct	9-Oct	16-Oct	5-Nov	12-Nov	26-Nov	11-Dec
Report #2 part 1								
Interaction Diagrams								
Report #2 part 2								
Interaction diagrams revision								
class diagram								
System architecture								
Report #2								
Interaction diagrams revision								
class diagram								
System architecture								
algorithms and data structures								
user interface design and implementation								
design of tests								
Demo #1								
1. database design								
2. user Experience and User Interface Design								
3. Coding the platform								
4. Unit testing and evaluation								
5. Integration testing and evaluation								
technical + user documentation								
presentation materials								
Report #3 part 1								
Report #3								
Demo #2								

Resources

- Report Guideline:
    - Marsic, Ivan. “Software Engineering Project Report.” *Book: Computer Networking - Textbook by Ivan Marsic*, www.ece.rutgers.edu/~marsic/Teaching/SE/report2.html
- Course Book: Software Engineering by Ivan Marsic http://www.ece.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf