



**Internship report on:**  
**Online academic information Hub (OAIHUB)**  
**&**  
**Daffodil Idea Hunt**

**Submitted by:**  
**Jashim Uddin Ahmed**  
**ID:181-16-279**  
**Department of Computing and Information System (CIS)**

**Supervised by:**  
**Nayeema Rahman**  
**Senior Lecturer, Department of Computing and Information System**  
**(CIS)**  
**Daffodil international university**

## **APPROVAL**

This Project title “**Online academic information hub (OAIHUB)**” and “**Daffodil Idea Hunt**”, Submitted by **Jashim Uddin Ahmed**, ID No: **181-16-279** to the Department of Computing & Information Systems, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computing & Information Systems and approved as to its style and contents. The presentation has been held on 19-07-2020.

## **BOARD OF EXAMINERS**



---

Mr. Md Sarwar Hossain Mollah

**Chairman**

**Assistant Professor and Head**

Department of Computing & Information Systems

Faculty of Science & Information Technology

Daffodil International University



---

**Ms. Nayeema Rahman**

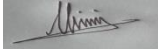
**Internal Examiner**

**Sr. Lecturer**

Department of Computing & Information Systems

Faculty of Science & Information Technology

Daffodil International University



---

**Mr. Minhaj Hosen**

**Lecturer**

Department of Computing & Information Systems

Faculty of Science & Information Technology

Daffodil International University

**Internal Examiner**



---

**Dr. Saifuddin Md. Tareeq**

**Professor**

Department of Computer Science and Engineering

Dhaka University, Dhaka

**External Examiner**

## **Acknowledgement**

In the name of Allah, Most Gracious, Most Merciful. All praise is due to Allah, the Lord of the Worlds. For his blessings I successfully complete the project.

*To our Project Supervisor, **Nayeema Rahman** Senior lecturer, Daffodil International University.* First of all, I would like to thank the teacher who has always been by my side like a shadow and has always inspired me, with his precious time.

Also, I would like to express my sincere gratitude to my family members and other teachers in my department especially my parents who are praying a lot for me to complete this project. I have received a lot of support from the teachers in my department during my project. I would like to say thanks to my big brothers and friends on campus, I could not have done this project without their support.

## Executive Summary

My six-month internship program work was with Daffodil international academy. I was involved in this as a software developer intern with their intern team name “**DIINTERNTEAM**”. This report will cover some background information on the projects was involved in as well as details on how the project was developed. The report also states that which academic courses and projects helped me in overall in internship experience so far.

At the very opening of the internship work I prepared several learning goals, which I wanted to learn about a lot of things. So, many projects they are developing at that time when I was joined there as an intern. I have worked in “**OAIHUB (online academic information HUB)**” which is one of their major projects and I had a significant role in this project. At the same time, I worked with a single projector called “**DAFFODIL IDEA HUNT**”. My task was to do following:

- Understand and working with spring boot framework and Thyme leaf.
- Understand and working with GIT & Rest API.

I obtain so many new technical skills though my work. I acquire new knowledge in front end development using thyme leaf. I also brushed my HTML5, CSS3, BOOTSTRAP4, JavaScript, java skills while working there. I also develop the “Daffodil Idea Hunt” web application using PHP.

This project is going to develop a web-based system name “**DAFFODIL IDEA HUNT**”. This is a web application all users are registration here. This system is allowing all user to share their any kind of idea (Technical or Non-Technical). Admin can manage all user file which is uploaded by user. The special feature of this system is text processing system. In this project I implement database to store end user data and project idea information. The innovation Idea Award policy is intended to provide guidelines on how to evaluate, award and recognize academic along with non-academic, staff to Daffodil Family with best innovation idea that dent positive impacts on the development of the family and national as well.

This internship work also helps me to develop research and analysis skills. That work helps me to enrich my code documentation knowledge too. This report shows

advantages of using spring boot framework and my working capabilities and detailed overview of that project where I was involved.

# Table of Contents

<b>Chapter 1 - Introduction .....</b>	<b>1</b>
<b>1.1 Purpose .....</b>	<b>2</b>
<b>Chapter 2 - Initial Study .....</b>	<b>3</b>
<b>2.1 OAIHUB .....</b>	<b>3</b>
<b>2.1.1 Background of the project .....</b>	<b>3</b>
<b>2.1.2 Problem Area .....</b>	<b>3</b>
<b>2.1.3 Possible solution.....</b>	<b>3</b>
<b>2.2 DAFFODIL IDEA HUNT .....</b>	<b>4</b>
<b>2.2.1 Background of the project .....</b>	<b>4</b>
<b>2.2.2 Problem Area .....</b>	<b>4</b>
<b>2.2.3 The expected solution.....</b>	<b>4</b>
<b>2.2.4 Feasibility analysis.....</b>	<b>5</b>
<b>2.2.5 The motivation and objectives of the project .....</b>	<b>5</b>
<b>2.2.6 Contribution.....</b>	<b>5</b>
<b>Chapter 3 - Literature Review.....</b>	<b>6</b>
<b>3.1 Discussion on problem domain based on published articles .....</b>	<b>6</b>
<b>3.2 Discussion on problem solutions based on published articles .....</b>	<b>7</b>
<b>3.3 Comparison of three/four leading solutions .....</b>	<b>7</b>
<b>3.3.1 Best features .....</b>	<b>7</b>
<b>3.3.2 Limitations .....</b>	<b>7</b>
<b>3.4 Recommended approach .....</b>	<b>8</b>
<b>Chapter 4 - Methodology .....</b>	<b>9</b>
<b>4.1 What to use .....</b>	<b>9</b>
<b>4.2 Why to use.....</b>	<b>11</b>
<b>4.3 Sections of methodology .....</b>	<b>11</b>
<b>4.3.1 Transparency .....</b>	<b>12</b>
<b>4.3.2 Inspection.....</b>	<b>12</b>
<b>4.3.3 Adaption .....</b>	<b>12</b>

4.4	Implementation plans .....	13
<b>Chapter 5 - Planning .....</b>		<b>14</b>
5.1	Project Plan .....	14
5.1.1	Management Plan / Work Breakdown Structure (WBS) .....	14
5.1.2	Time Duration / Time Boxing.....	15
5.1.3	Gantt Chart .....	16
5.2	Test Plan.....	16
5.2.1	Required tests.....	16
5.2.2	Test Case.....	18
5.2.3	User acceptance test plan.....	19
<b>Chapter 6 - Foundation .....</b>		<b>20</b>
6.1	OAIHUB .....	20
6.1.1	Overall Requirement List .....	20
6.1.2	What Technology to be implemented (Client/Web/Standalone).....	21
a.	Technical Languages .....	21
b.	Databases Systems .....	21
c.	Technologies.....	22
d.	Framework.....	22
e.	Build tool .....	22
f.	Server Platforms .....	22
6.2	DAFFODIL IDEA HUNT .....	22
6.2.1	Requirement list .....	22
a.	List of functional requirements.....	22
b.	List of system-wide non-functional requirements.....	23
<b>Chapter 7 - Exploration.....</b>		<b>24</b>
7.1	Old Full System Use Case.....	24
	Description:.....	25
7.2	Old Full System Activity Diagram.....	26
	Description:.....	27



7.3	Prototype of new system (OAIHUB) .....	28
	Description:.....	28
7.4	Diagram of the overall architecture (Daffodil Idea Hunt).....	29
Chapter 8 - Engineering.....		30
8.1	New System Modules .....	30
8.2	Use Case .....	31
	Description:.....	32
8.3	Class Diagram.....	33
	Description:.....	38
8.4	ERD Diagram.....	39
	Description:.....	39
8.5	Sequence Diagram.....	41
a.	As an admin .....	41
	Description:.....	42
b.	As a moderator .....	43
	Description:.....	43
c.	As a user .....	44
	Description:.....	45
8.6	Use Cases of Daffodil Idea Hunt .....	45
8.7	ERD Diagram of Daffodil Idea Hunt .....	46
8.8	Activity diagram of Daffodil Idea Hunt .....	47
Chapter 9 - Deployment / Development.....		48
9.1	OAIHUB.....	48
9.1.1	User Management .....	48
a.	User Class .....	48
b.	User Dao Class .....	53
c.	User Repository Class.....	54
d.	Role Class .....	55
e.	Role Dao Class.....	56

f.	Role Repository Class .....	57
g.	CustomeUserDetailsService Class .....	58
h.	UserDashController Class.....	58
i.	UserController Class.....	59
j.	Signup Controller Class .....	61
k.	Role Controller Class .....	64
l.	HomeController Class .....	65
m.	AdminDashController class.....	66
9.1.2	Configuration.....	66
a.	Login Security Class .....	66
b.	WevMvcConfig Class.....	69
9.1.3	Generic Interface .....	69
a.	This interface is used to hold all similar methods & signature in one place & use them in all the necessary classes & codes where it needs to be used.....	69
9.1.4	Exam Management.....	70
a.	Exam Class.....	70
b.	Exam Repository.....	71
c.	Exam Dao .....	72
d.	Exam Controller .....	73
e.	Comments .....	75
f.	Child Comments .....	77
g.	Comment Repository.....	79
h.	Child Comments Repository .....	79
i.	Child Comments Dao.....	81
j.	Comment Controller .....	82
9.1.5	Forum Code sample .....	87
a.	Post Controller .....	87
b.	Feedback of threads controller.....	89
c.	Thread comments Controller .....	90

d.	Post / thread DAO class .....	91
e.	Feedback of posts .....	92
f.	post comment Dao.....	93
g.	Post Class .....	95
h.	Post Comment Model Class.....	101
i.	Post feedback model class .....	103
j.	Post feedback repository .....	105
k.	Post repository .....	105
l.	Post comment .....	105
9.1.6	Voting code samples.....	105
a.	Votes Controller .....	105
b.	Vote Types Controller.....	107
c.	Suggested Edit Votes .....	108
d.	Suggested edit votes Dao .....	109
e.	Votes Dao .....	110
f.	Vote types Dao.....	111
g.	Suggested edit model class.....	112
h.	Votes Model class.....	115
i.	Vote type model class .....	118
9.2	DAFFODIL IDEA HUNT .....	120
9.2.1	The programming language and framework which I choose and why .....	120
9.2.2	Details paragraph about each working function .....	120
Chapter 10 - Testing.....		127
10.1	OAIHUB.....	127
10.1.1	Unit testing .....	127
10.1.2	Integration Testing:.....	131
10.2	DAFFODIL IDEA HUNT .....	134
10.2.1	Unit, System and module Testing outcome and errors .....	134
10.2.2	Interface Testing: .....	134

10.2.3	Performance Testing:.....	134
10.2.4	Usability Testing: .....	135
10.2.5	Black Box Testing: .....	137
10.2.6	White Box Testing:.....	137
10.2.7	Identify Possible Test Scenarios:.....	137
a.	Test Case for Registration .....	138
b.	Test Case for login.....	139
10.2.8	Quality Assurance technique .....	140
<b>Chapter 11 - Implementation .....</b>		<b>141</b>
11.1	Training .....	141
11.1.1	Assess training needs.....	141
11.1.2	Set organizational training objectives.....	141
11.1.3	Create training action plan.....	142
11.1.4	Implement training initiatives.....	142
11.1.5	Evaluate & revise training .....	142
11.2	Big Bang.....	142
<b>Chapter 12 - Critical Appraisal and Evaluation .....</b>		<b>144</b>
12.1	Objective that could be met .....	144
12.1.1	Success rate against each objective .....	145
12.1.2	How much better it could be done.....	145
12.1.3	How better are the features of the solution? .....	145
12.2	Objectives totally not met / touched.....	146
12.2.1	OAIHUB.....	146
a.	Why it could not be touched .....	146
b.	What could have been done.....	146
12.2.2	DAFFODIL IDEA HUNT .....	147
a.	The requirements that cannot be implemented completely and how I overcome them .....	147
<b>Chapter 13 - Conclusion .....</b>		<b>148</b>

<b>13.1</b>	<b>OAIHUB.....</b>	<b>148</b>
13.1.1	Conclusion.....	148
13.1.2	Summary of the project .....	148
13.1.3	Goal of the project.....	149
13.1.4	Success of the project .....	149
13.1.5	Value of the project .....	149
<b>13.2</b>	<b>DAFFODIL IDEA HUNT .....</b>	<b>150</b>
13.3.1	Strengths and weakness .....	150
13.3.2	Future extension scopes .....	150
13.3.3	Restate contribution precisely .....	150
<b>13.3</b>	<b>My experience.....</b>	<b>150</b>
	<b>References .....</b>	<b>152</b>
	<b>Plagiarism Report:.....</b>	<b>153</b>

<b>Figure 1: Agile Methodology .....</b>	<b>9</b>
<b>Figure 2: Sprint.....</b>	<b>10</b>
<b>Figure 3: Scrum Framework.....</b>	<b>11</b>
<b>Figure 4: Three pillars of Scrum .....</b>	<b>12</b>
<b>Figure 5: WBS Chart. ....</b>	<b>15</b>
<b>Figure 6: Time Boxing.....</b>	<b>15</b>
<b>Figure 7: Gantt Chat.....</b>	<b>16</b>
<b>Figure 8: Test Case table.....</b>	<b>19</b>
<b>Figure 9: User acceptance test plan .....</b>	<b>19</b>
<b>Figure 10: Old Full System Use Case .....</b>	<b>24</b>
<b>Figure 11: Old Full System Activity Diagram .....</b>	<b>26</b>
<b>Figure 12: Architectural design: MVC architecture .....</b>	<b>28</b>
<b>Figure 13: Overall architecture of this web application.....</b>	<b>29</b>
<b>Figure 14: Use case Diagram .....</b>	<b>31</b>
<b>Figure 15: Full system Class Diagram.....</b>	<b>37</b>
<b>Figure 16: ERD Diagram .....</b>	<b>39</b>
<b>Figure 17: Sequence Diagram for admin.....</b>	<b>41</b>
<b>Figure 18:Sequence Diagram for moderator .....</b>	<b>43</b>
<b>Figure 19: Sequence Diagram for user.....</b>	<b>44</b>
<b>Figure 20: Use Cases of Daffodil Idea Hunt .....</b>	<b>45</b>
<b>Figure 21: ERD Diagram of Daffodil Idea Hunt .....</b>	<b>46</b>
<b>Figure 22: Activity diagram of Daffodil Idea Hunt .....</b>	<b>47</b>
<b>Figure 23: This is the login page.....</b>	<b>121</b>
<b>Figure 24: Temporary block user cannot login. ....</b>	<b>121</b>
<b>Figure 25: Student registration. ....</b>	<b>122</b>
<b>Figure 26: Employee registration.....</b>	<b>122</b>
<b>Figure 27: Dashboard page. ....</b>	<b>123</b>
<b>Figure 28: View All user and student.....</b>	<b>123</b>
<b>Figure 29: Admin can view every user’s profile individually.....</b>	<b>124</b>
<b>Figure 30: Admin can manage every user role and permission. ....</b>	<b>124</b>
<b>Figure 31: Admin can add category of idea and sub category of idea. ....</b>	<b>125</b>
<b>Figure 32: User can share their idea.....</b>	<b>125</b>
<b>Figure 33: User can view his idea. ....</b>	<b>126</b>

**Figure 34: Admin can view all idea which was uploaded and have right to give any kind of status..... 126**  
**Figure 35: In this graph we can see that overall performance report..... 135**  
**Figure 36: The usability test report of this site..... 136**

## Chapter 1 - Introduction

In my entire internet time, I have developed two web application projects. These two projects are educational that support the any king of academy. One of these projects is completely developed by me alone which is “**DAFFODIL IDEA HUNT**”, and the other one I developed “**OAIHUB (online academic information HUB)**” by working with my teammates. I will describe my complete documentation with these two web applications.

At present, people are usually depending on modern technology for their daily activities. But the education system in Bangladesh has not been digitized with modern technology yet. Education is one of the most important parts where we need to improve ourselves. There is a lot of gaps in our education system. There is no site where people can get the initial information for their children’s admission or anything else. People are suffering many problems for the information gap from the institution. The guardians and students don’t know about the procedure for their admission to any school, college or university.

So, we want to develop a system where students can get their necessary educational information. The system will contain all the information of all the educational institutions from school to university. They can know about the cost of the individual school, college or university and the procedure for admission to the specific institute. They can know about the facility of the institute, ranking of their desired institute. Each and everything information will be uploaded in this system so that students and guardians can be benefitted. The system will also contain all the public exam questions and answers. Students can share any questions and answer in this system and there are an admin and moderator who moderate the user activity, give them access to share questions or any other. There is a forum where they can discuss the questions or any queries. Any abusive post will be moderated by the moderator. Students can get any update information by this system. So, this system will be helpful for the students.



## 1.1 Purpose

The documentation below contains a detailed discussion about the workflow of the project. The purpose of the documentation is to clear the procedure of the project to any new developer who will be working with this system. As there will be a number of functionalities in this system which might be tough to understand for some developers. This documentation will help them to adopt the system with proper knowledge about the system. There will be some detailed diagrams about the system which will help the user to understand the workflow of the system. The interfaces of different functionalities will be provided here for a better understanding of the developers. As we need to run some testing to find out if the system is running properly, the testing details with the result will be documented here for the reference. This documentation will clarify the design of the system and also the reason behind the way the system is designed. This document contains the algorithm used for the system with proper justification. It also contains the uses of the database. It clarifies the design of the database, the entity-relationship model and how the relations are working here. We can consider this documentation as a clear view of the system. The documentation part is one of the most important parts of the project. As the system will be developed for public uses so the user interface and the functionalities must be explained here so that the system remains understandable to all. With the documentation, the functionalities and designing of the system may not be understood by the users of the system. For the reason, proper documentation is always needed for the project.

## **Chapter 2 - Initial Study**

### **2.1 OAIHUB**

#### **2.1.1 Background of the project**

In our country admission coaching business plays a significant role when a time periods comes to our students to get admission in school, college and university. They are facing so many problems at that moment. At first, they are suffering from information lacking about that institutions. So many students are unable to collect information from their preferred institutions by visiting that institutions. Sometimes they missed their admission test due to lack of valid information. Students can't decide that moment which educational institute is good for them. Most students don't know about payment scheme of an educational institutions. OAIHUB web application will capable to reduce all of these problems in future. It allows user to found all information from remote home.

#### **2.1.2 Problem Area**

Every year in our country students are facing so many problems when they are going to get admitted into a school, college, university, and national university. The student doesn't know which school, college, university and the national university is good for him. which documents are needed if they want to get admitted to their favorite institution? They also don't know about their payment system and payment amount. They also don't have any concept about their admission test exam questions and so many important information about their preferred institutions. Sometimes lack of proper information they missed their admission test exam.

#### **2.1.3 Possible solution**

After analyzing all problems, I saw in our country students are facing so many problems when they are going to take admission to any educational institute. To reduce all these problems OAIHUB web application is the best possible solution. This system brings all academic information to its users. User also can discuss about their academic problems by creating post with other users. A user also can judge or provide

a solution to a post via creating a comment. User can view previous admission test question of various year. They also can participate in those old admission test exams to improve their skills. User never miss any notification of any admission test what user wants to participate. In this system user also can view their educational institute rank and other important information also. OAIHUB also a great feature for pro users only which is a paid feature of this system. By using this feature user will get IELTS, GRE, SAT, etc. questions and answer for that questions and they also participate in online mock test exams.

## **2.2 DAFFODIL IDEA HUNT**

### **2.2.1 Background of the project**

In the project I have developed which is “Daffodil Idea Hunt”. I developed this project as a web base application. In this application is developed for all staff/employee and student of daffodil family. In the project administrator can control all the users of this system. User can do their job in organized way and share their idea through this web application. The administration can control all the submitted idea which was shared by user.

### **2.2.2 Problem Area**

Idea is one of the biggest challenges of the 21st century. Everything is already made in this century but the biggest problem creating something new and creative, something which is different. Each of us strive to solve our everyday obstacles and all we are doing is not anything new, repeating old things over and over again. We are bound in a circle and We have to get out of this circle and develop our thinking.

### **2.2.3 The expected solution**

I have developed this project for an organization “Daffodil Idea Hunt” to solve this problem. In their different types of users are share their idea through this system. If a user wants to submit their idea than they are use this system. In there they analysis all the idea which is submitted by the user. After than they will find out the best thing and give reward the user who submitted the idea.

## 2.2.4 Feasibility analysis

A thorough understanding of all aspects of a project, concept or plan to be aware of any potential problems that may arise during the project implementation.

**Executive summary:** Formulate a description that describes the project, plan, or service details.

**Technical Feasibility:** In this project I need to use some of child organization information for verify the actual user.

**Operational:** The viability of a plan being implemented and run in this section. E.g. In there I will implement the data which I collected before.

**Findings and recommendations:** Divide into technology, organization, and financial sub-sets.

## 2.2.5 The motivation and objectives of the project

I'm encouraged to do it because it's so unique. To developed this project, I have learned about many new things and apply them into this system. The biggest motivation is to developed this project is, it is the first time I'm working with a big organization.

## 2.2.6 Contribution

The organization will be profitable with this system, because the innovation idea will the organization to grow up. This will increase the market value of the organization. The organization will be able to generate new ideas day by day from its staff and student.

## Chapter 3 - Literature Review

### 3.1 Discussion on problem domain based on published articles

In our country people suffer from various kinds of problems in education sector.

- Most of the people does not know how to accommodate with the education system for their child.
- A father doesn't know which school will be suitable for his child and cost friendly for him and also well facilitated for both of them based on their situation.
- Sometimes people don't even know how much money to take out for admission fees.
- Sometimes some institution may show people that they're offering very affordable cost for people but later with time they demand too much high price for completion of their child's study. Which creates a heavy pressure on parents.
- When it comes to the question which college will be good and what type of study a student has to go through nobody knows the proper one.
- Nobody can answer what kind of specific preparation a student should take as there are thousands of coaching centers and book publishers offering their own methodologies which only leads to their own business purposes. Students are greatly suffered there.
- Students cannot find or have to buy previous question banks for high prices for taking preparation in the exams.
- Model test costs are very much high depending on coaching centers advertisements "Getting A+ in God Speed" or "Getting admitted into desired institution with zero study" which is not affordable for all of the students.
- Students who have just graduated don't know which companies to apply based on their skills.
- New graduates can't give proper model tests for their specific job exams or interviews.
- Students cannot find answers of question banks, cannot take suggestion or teachings sometime to have the best answer for his problem.

- We do not have a discussion hub of our country for educational discussion or working purpose.

### **3.2 Discussion on problem solutions based on published articles**

Depending on all these problems various people came up with various solutions.

- Institutes started their own terms of marketing with benefits. Giving various offers to the students.
- A lot of mini coaching centers for school, college, university admission has created.
- Each and individual coaching centers had specified a specific publication of books to read.
- People started using various social media sites to find information about institutions and resources to study for giving exams.
- No specific Solutions has made to solve all the above-mentioned problems.
- Students with low lost budgets are missing thousands of chances and facilities to shine their life.
- With the digital technology people started learning accordingly how to cope up with all of this term by term.

### **3.3 Comparison of three/four leading solutions**

#### **3.3.1 Best features**

- Digitization has made people's life easier. People can easily access to their required information though they have to surf for it too much.
- People can rely on specific institutes for getting admitted into them.
- Various information can be found on various places on internet about the institutions, course curriculum, cost and a lot more.
- Some question answer sheets of various years can be found on internet.

#### **3.3.2 Limitations**

- Information are lack of accuracy. Proper information cannot be found.

- Different sources tell different information. Questions and answers are not found with accurate guideline or answers.
- Job applications or advertisements are not accurate.
- People do not have the ability to speak with the specialists for better solutions.

### **3.4 Recommended approach**

- All the educational information and related things should be brought together.
- All students should be treated equally.
- Parents should know what they are doing. Where their children are getting admitted, is it affordable and maintaining for the parents.
- Students must have the ability to decide where they want to study and grow their future career.
- All educational equipment should be very much affordable so that no one misses their rights.
- Students can give their model test for very much affordable cost for getting prepared for the exams or jobs.
- A place where all the legal information will be found about each and every educational institution to make decisions for the children. Where People can compare between the institutions and decide which will be better for their children and affordable for the parents.
- Nothing should be compromised when it's the question of education and the future of our country.
- Whenever any student is asking a question or in a problem will have the ability to share it in somewhere where all kinds of specialists will be available to give solutions.
- Nobody will miss their education rights to study and brighten their future.

## Chapter 4 - Methodology

Methodology is a set of procedures or a particular procedure. It helps to provide appropriate guideline principle for developing an application or system.

### 4.1 What to use

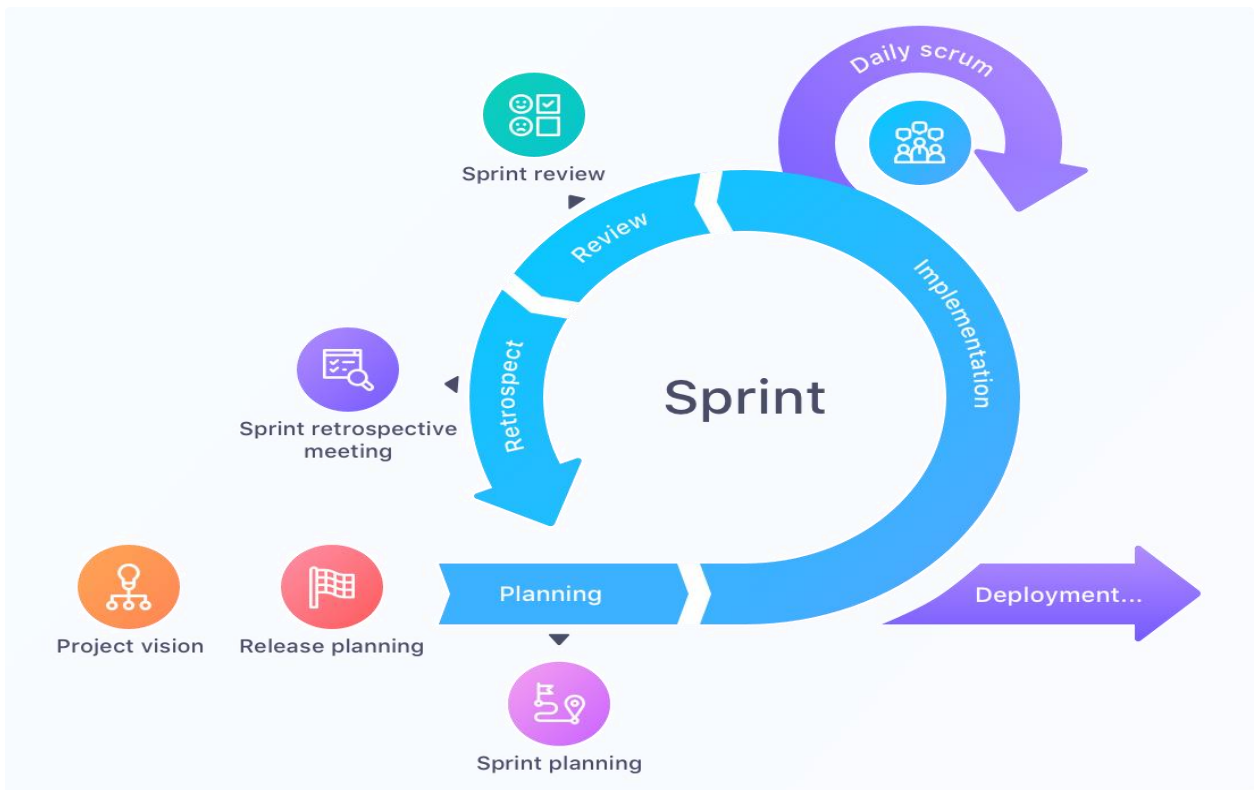
In software development site, there are so many methodologies for developing an application. Agile is one of them. Actually, agile is an evolutionary project management approach under which requirements and solution evolve through the collaborative effort of self-organizing/ cross-functional teams and their customer/end users.



**Figure 1: Agile Methodology**

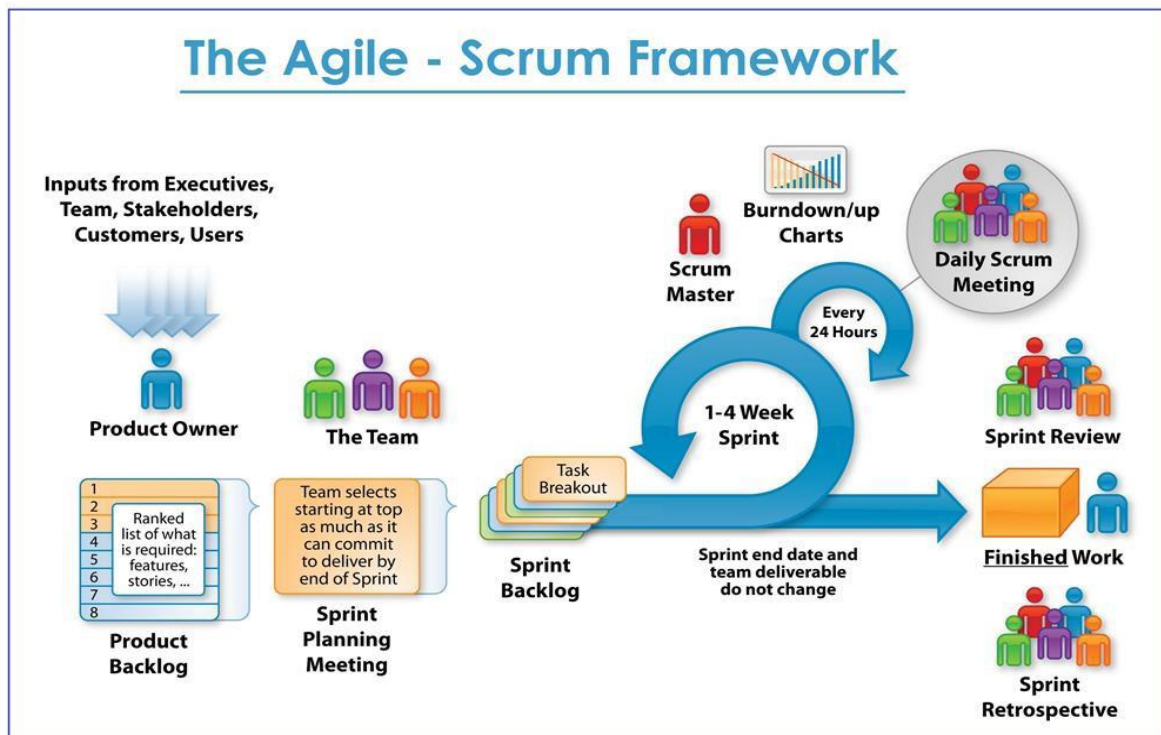
it is a project management methodology what uses small development cycles name “sprints” to attention on continues improvement in the development of an application or a system.





**Figure 2: Sprint**

In this project development our team has been conducted with scrum framework of agile. This framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. Scrum is lightweight, simple to understand and difficult to master



**Figure 3: Scrum Framework**

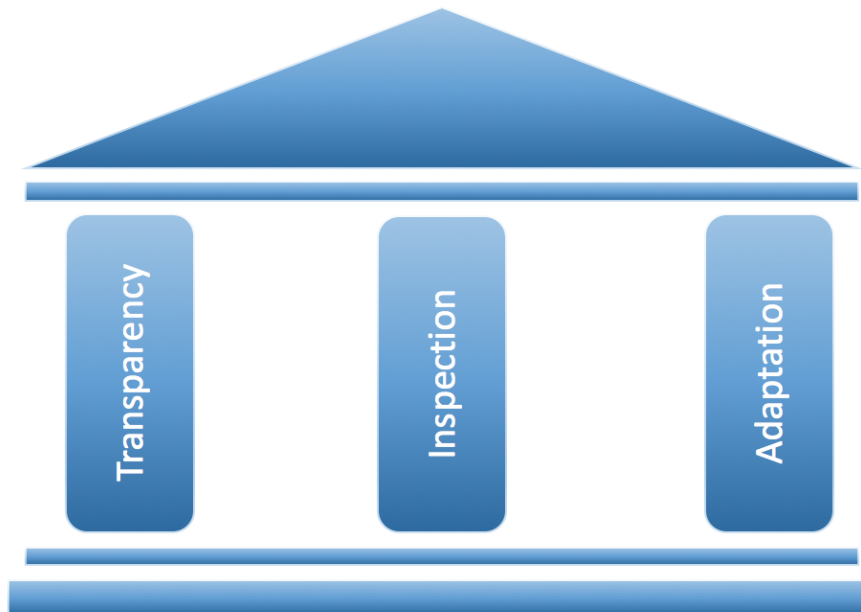
## 4.2 Why to use

The reason of using Scrum framework is given below:

- Higher productivity.
- Better-quality products.
- Reduced time to market.
- Improved stakeholder satisfaction.
- Better team dynamics.
- Happier employees.

## 4.3 Sections of methodology

There are three pillars of Scrum.



**Figure 4: Three pillars of Scrum**

#### **4.3.1 Transparency**

Significant aspects of the process must be visible to those responsible for the outcome. Transparency requires those aspects be defined by a common standard so observers share a common understanding of what is being seen.

#### **4.3.2 Inspection**

Scrum users must frequently inspect Scrum artifacts and progress toward a Sprint Goal to detect undesirable variances. Their inspection should not be so frequent that inspection gets in the way of the work. Inspections are most beneficial when diligently performed by skilled inspectors at the point of work.

#### **4.3.3 Adaption**

If an inspector determines that one or more aspects of a process deviate outside acceptable limits, and that the resulting product will be unacceptable, the process or the material being processed must be adjusted. An adjustment must be made as soon as possible to minimize further deviation.

## 4.4 Implementation plans

Agile implementation is a form of project management that works in small increments and well suited to projects that could be become irreverent once delivered, especially useful in software development. The key to the agile plan is that it provides flexibility for changes to the product as it continues to be developed. Scrum is a framework of agile what delivering product iteratively and incrementally in a timebox fashion. This is simple illustration of what the scrum implementors and others define it, moving with it.

# Chapter 5 - Planning

## 5.1 Project Plan

Here I'm going to show the entire internship work planning in a way that the internship work is being done by me. The whole work is divided in small pieces and those are done within the fixed period of time. In this phase a specific task when will be started and when will be end those things are defined.

### 5.1.1 Management Plan / Work Breakdown Structure (WBS)

WBS (work breakdown structure) is a tool what make the work more manageable and approachable. This approach helps to complete all the task within fixed time duration. The WBS (work breakdown structure) chart of my whole internship work is given below:

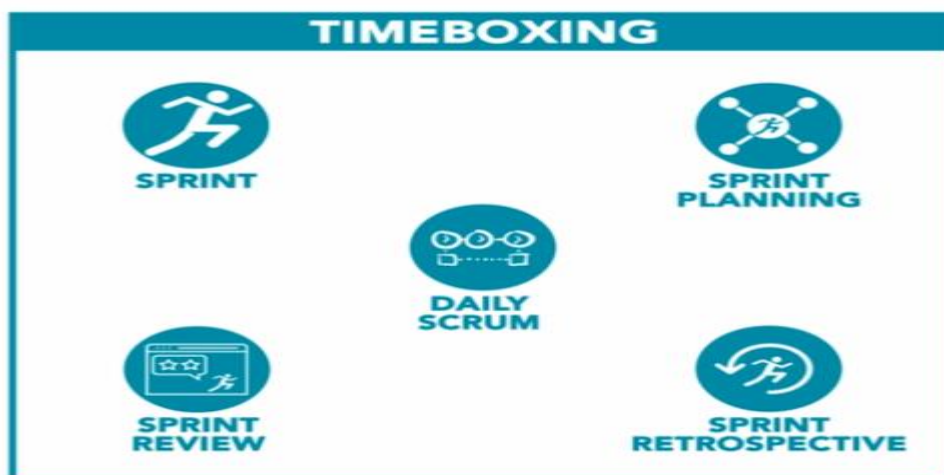
Serial no.	Task title	Start date	End date	Durations (Days)
1	Introductions	01.01.20	02.01.20	2
2	Initial study	03.01.20	09.01.20	7
3	Literature review	10.01.20	14.01.20	5
4	Methodology	15.01.20	18.01.20	4
5	Planning	19.01.20	24.01.20	5
6	Foundation	25.01.20	30.01.20	6
7	Exploration	01.02.20	04.02.20	5
8	Engineering	05.02.20	09.02.20	5
9	Deployment	10.02.20	01.03.20	20

10	Testing	02.03.20	11.03.20	12
11	Implementation	12.03.20	21.03.20	12
12	Critical Appraisal and Evaluation	22.03.20	28.03.20	7
13	Conclusion	29.03.20	30.03.20	2
<b>Total</b>				90 days

**Figure 5: WBS Chart.**

### 5.1.2 Time Duration / Time Boxing

The whole system is being done with agile methodology. All the tasks are iterative in this approach. Timeboxing is one of the useful parts for this project management approach (scruminc, 2019). In this timeboxing process that we follow shown here below:



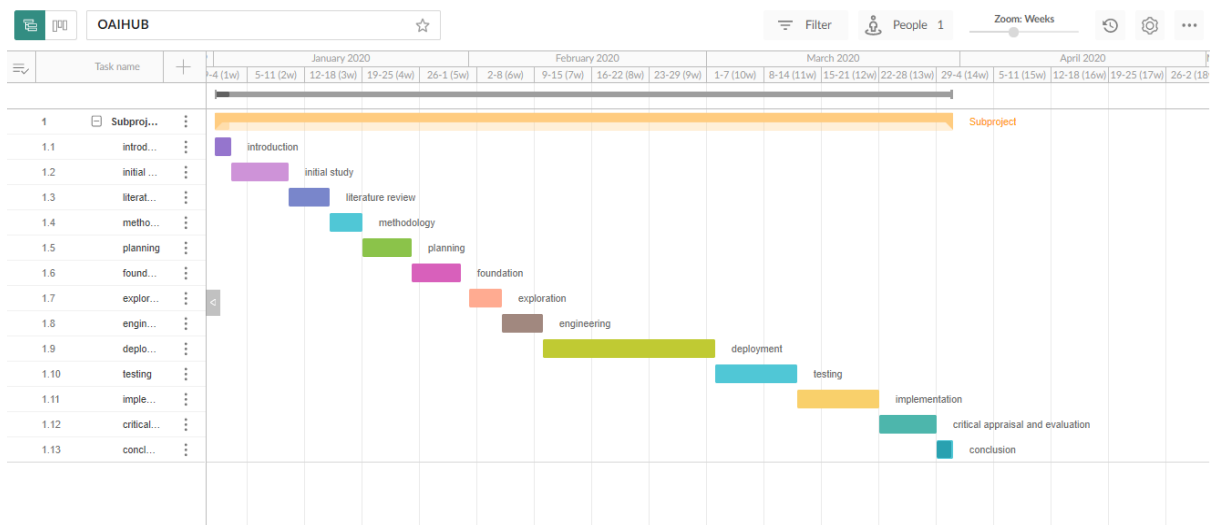
**Figure 6: Time Boxing**

This is one of the critical components of a good scrum. Sprint used for utilize the length of the project. Sprint planning is actually a meeting what timeboxed 8 hour or less for a one month. Daily scrum process is actually a timebox for 15 minutes per day what helps to synchronize teams' activities. Sprint review also a time boxing to adapt the

backlog based on feedback. Sprint retrospectives is an event what inspect itself identifies processes.

### 5.1.3 Gantt Chart

This chart shows activity schedule of the projects and the time duration of specific tasks of the projects. This thing is very important for developing a project. Gantt chart of this projects is given below:



**Figure 7: Gantt Chat**

## 5.2 Test Plan

A test plan is a list of documents which is describes scope and activities for the software testing. It is the foundation on which to properly test any software / product within a project. Amongst other things, it identifies test objects, the features to be tested, the testing tasks, which will do for each task, the test environment, the test design methods and the standards to be used for entry and exit, intendency of the tester and the rationale for their choice and any risks that necessitate contingency planning. It is a record of the procedure of planning tests. (test-plan, 2020)

### 5.2.1 Required tests

There are many types of testing methods and techniques that we will use here in stages. We will test our application here using different testing methods. There are lots

of testing categories for testing web application but in this section, I have followed some popular and efficient testing methods for test this web application. By following these I have fully tested this web application. The types of tests that web application testing should focus on are described below. (guru99, 2020)

#### **a. Unit Testing**

Unit Testing is a type of web application testing where a web application tests individual units or components. The persistence of the application code is to validate that each unit performs as predictable. This testing is completed by developers throughout the development of an application. This test separate and verify the completeness of a unit of code. A unit might be a function, method, procedure, module, or object of its own (unit-testing-guide, 2020).

#### **b. Integration Testing**

System Integration Testing is described as a kind of web application testing performed in a server and system environment to validate the whole application behavior. It is testing which is steered out on a complete, integrated application to assess compliance of the system with its quantified requirement (system-integration-testing, 2020).

#### **c. Module Testing**

This testing focuses mainly on testing web application's programs or sub-program modules, relatively this testing the entire web application at once. This testing in software engineering is very advantageous and at all times suggested since it is very easy to identify, understand and fix the faults and errors at the level of the segment rather than fix that at the section of the application. The main goal of this testing is to confirm that the module is completely tested and functional to contribute in Application Testing (softwaretestingclass, 2020).

#### **d. Performance Testing**

“**OAIHUB**” Is a web application, In the situation of web development, performance testing includes pretending how an application runs through a specific circumstance by consuming the software tools. Measurable performance testing is response time when qualitative testing is about the scalability of this site, stability and interoperability



of this application. Performance testing of this web application allows to identify issues and improve complete performance, which can lead to develop the user experience (UX) and increased expediency. This testing can expose many common problems, like bottlenecks (keycdn, 2020).

### e. Security Testing

This testing is a category of testing which is reveals vulnerabilities, threats, risks in this web application and prevents malicious attacks. The reason of this tests is to finding all possible loopholes and weaknesses that could outcome in information loss, revenue loss etc. The objective of this testing is to finding the threats and measure its potential vulnerabilities, so that the application does not stop working. It also helps to notice all possible security risks inside the application and helps to solve these problems (security-testing, 2020).

### 5.2.2 Test Case

Mainly test case is a set of rules or variables by which a tester will determine that whether a system meets all the requirements of this web application or is functioning acceptably. This can also help to finding problems in an application’s requirements.

**Test Priority:**

**Test Execute by:**

**Unit test No:**

**Test Execute Date:**

**Test case:**

**Objective:**

**Data Source:**

Case No.	Description	Tasks	Result		Status (Pass/Fail)
			Actual result	Expected result	

01		
02		

**Figure 8: Test Case table**

### 5.2.3 User acceptance test plan

User Acceptance Testing (UAT) it can be defining a process that the application is delivered to customer/client in one-word direct user, they use the application for a specified period of time and finding the problem of the web application.

**Unit test No:**

**Test Execute User Name:**

**Test case:**

**Test Execute User Role:**

**Objective:**

**Data Source:**

**Pre-condition:**

**Post-condition:**

Case No.	Steps	Result		Status (Pass/Fail)	Comments
		Actual result	Expected result		
01					
02					

**Figure 9: User acceptance test plan**

# Chapter 6 - Foundation

## 6.1 OAIHUB

### 6.1.1 Overall Requirement List

To Build the preliminary system we need following things to be implemented for constructing the project. The basic requirements will probably able to cut out the edge of the project which has been planned out. The requirements which has been analyzed for so long to develop this system will going to be cover maximum objective requirements of the proposed system. Here is all the overall requirement list given below:

- To register & save new user
- To register & save new moderator
- To register & save new university moderator
- To give control of the whole system in one hand (Admin)
- To register & save pro user
- To upload and download files via specific users
- To upload verified question & answer sheet by moderators
- To Upload & view institute details
- Compare between institutes details
- View & edit user profile
- View & edit moderator profile
- View & edit pro user profile
- View & edit university moderator profile
- To register normal registered user as pro user through payment
- Pro user has access to most of the things
- Pro user can view & download question – answer sheets
- Pro user can give model tests
- Pro user can apply for model tests
- Pro user can apply for specific institutes online
- Education board's various types of exam routines will be shown in the notice board
- UGC notices will be shown through moderators.

- Institute's over all details like cost, facilities, study quality, admission details everything will be shown
- A discussion forum or system is needed
- Every user, requirement-based moderators, pro users, will be able to post the discussion topics as threads
- Other users can comment under those post.
- Important threads can be upvoted via rating system
- Threads will be attached with tags to make it retrievable to specific topics
- Model tests will be examined and moderated via top notch teachers.
- Specific thread publisher names will be visible individually
- Specific comment publisher names will be visible
- Every thread will be viewed by time and date.
- Comments under a thread will be viewed by date and time. Important files will be able to be uploaded through users, pro users in the threads e.g. snapshots, code snippets, word documents, images, and many more.
- A strong secure database system is needed to store all these information part by part and sequentially
- All specific details will be analyzed and saved via the system.

### **6.1.2 What Technology to be implemented (Client/Web/Standalone)**

The technologies and languages which are going to be implemented in this system to develop the proposed system are given below:

#### **a. Technical Languages**

Java, JavaScript, AJAX, XHTML, CSS, Json, JSP, JSTL, HTML, Codemix, NodeJS, Bootstrap, jQuery,

#### **b. Databases Systems**

MySQL, Tomcat, JDBC,

### c. Technologies

ORM (Object Relational Model) tool, Hibernate, REST API, Restful API, Data JPA, Spring Boot, Spring Security, Spring Boot Dev tools,

### d. Framework

Spring, Thyme Leaf

### e. Build tool

Maven, Gradle

### f. Server Platforms

Daffodil Web Server Storage

## 6.2 DAFFODIL IDEA HUNT

### 6.2.1 Requirement list

#### a. List of functional requirements

##### User details:

- **Add employee:** This system will allow to add employee.
- **Add Student:** This system will allow to add student.

**Role management:** In their admin can manage the role of user.

**Login:** Login page for admin, staff/employee and student.

**Dashboard:** See personal activity report on the dashboard.

**User authentication:** In the first login user need to admin permission.

**View user:** In their admin can see all Staff and employee user as well as student.

**Block & Unblock:** Admin can block & unblock faculty user as well as student.

**File upload:** Admin and faculty member can upload file (login require).

**Category:**

- **Add category:** Admin can add any kind of category.
- **Add sub category:** Admin can add sub category under a category.

**Report:** Admin can see the full system birds eye view.

**Search:** Admin can search the any user.

**Download file:** Admin can download uploaded file which is uploaded by the users.

**b. List of system-wide non-functional requirements**

**Security and auditing:**

- **Administrator's rights:** The Administrator's will have all right to add, delete or update any kind of information.
- **System user login ID:** Every system user has individual login id to control the system.
- **Modification:** The system can be modifying if it's needed.

**Reliability:** Create a requirement that data created in the system will be retained for a number of years without the data being changed by the system.

**Performance:** Requirements about properties required, response time, benchmark specifications or anything else having to do with performance.

**Maintainability:** This is the ability of the application to go through changes with a fair degree of smoothness. This quality is the flexibility with which the application can be modified, fixing issues, or to add new functionality.

**Supportability:** Supportability requirements are concerned with the ease of changes to the system after deployment.

**Usability:** Requirements about how problematic it will be to learn and function the system. The requirements are often expressed in learning time or comparable metrics.

# Chapter 7 - Exploration

## 7.1 Old Full System Use Case

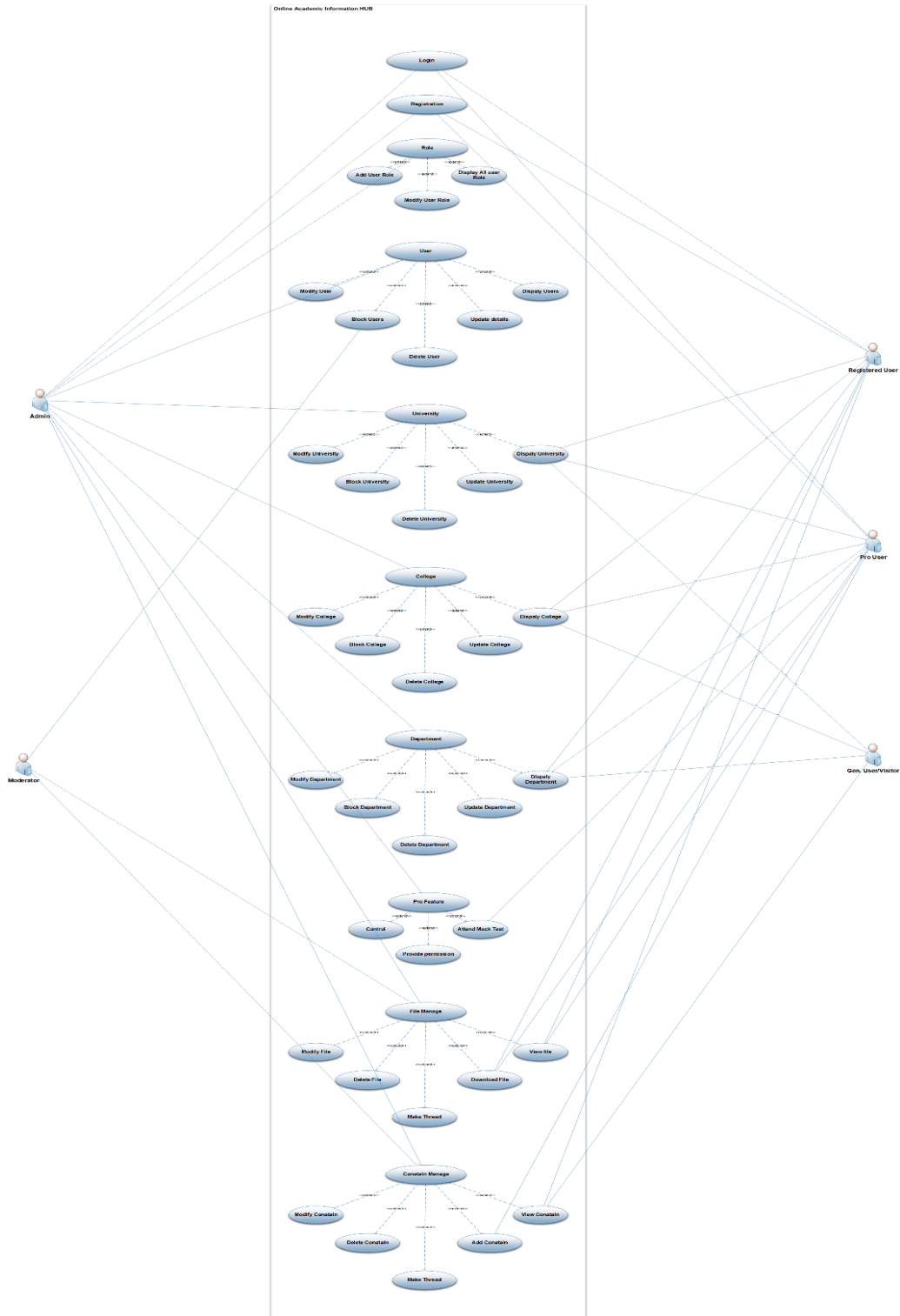


Figure 10: Old Full System Use Case

## **Description:**

In this information hub there are total 5 types of user. Every user has their own rule to use this system. They have various type of data access in this system such as:

**Admin:** He/she can register themselves into the system and can login by their personal information. They can make and modify role in the system, can add user, display all user role. Modify user, block user, delete user, update user data those also can be accessible by admin. All type of University, College and Department related data can be display, blocked, modify or delete by the admin. They also have some pro feature such as control, provide permission and attend mock test (tester). All necessary file can be handled by the admin panel, modifying files, delete files, make thread, download file and view file are the admin panels task. Contain manage like modify, delete, make thread, add and view contain.

**Moderator:** This panel has less power and access then the admin. Modify user, block user, delete user, update user data those also can be accessible by moderator panel. Necessary file can be also handled by the moderator panel. Modifying files, delete files, make thread, download file and view file are the admin panels task. Contain manage like modify, delete, make thread, add and view contain.

**Register user:** This panel they can login and register in the beginning. They can see university, college and department data. From this panel they can also attend mock test. They will have access to view file and download them. They can able to add content and also view others.

**Pro-user:** This panel members are the special then register member. This panel member can login and register in the beginning. They can view university, college and department all data. From this panel they can attend mock test. They will have access to view file and download them. They can also able to add content and also view others.

**General user/visitor:** They have the less ability in this system. Visitor can register themselves. Then they can do many things. Without registration they can only view can view university, college, department data and some content.



## 7.2 Old Full System Activity Diagram

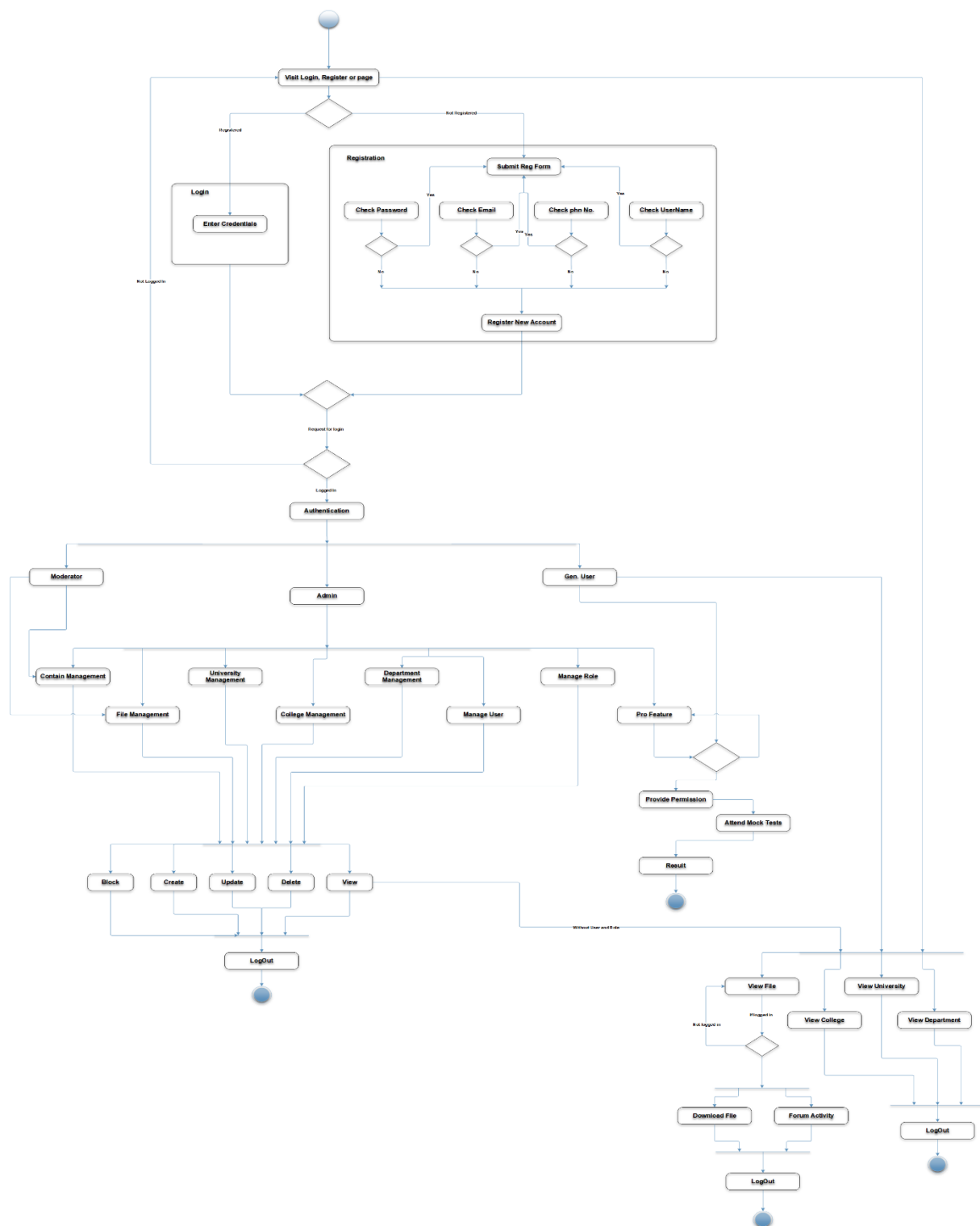
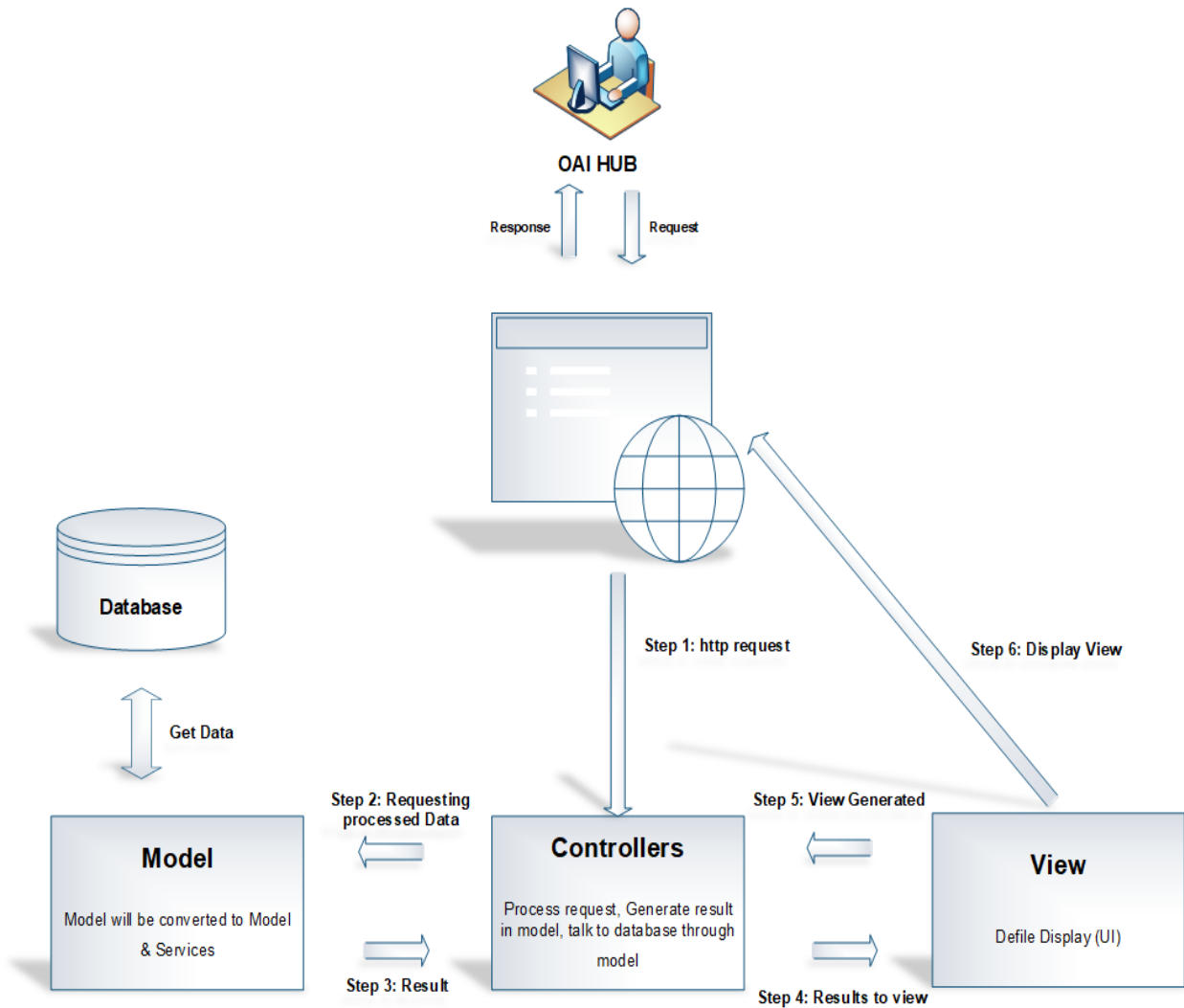


Figure 11: Old Full System Activity Diagram

**Description:**

In the activity diagram above the whole procedure that can be undertaken by a user are shown. If the user is registered, he will go the login page and log into the system providing valid user ID and password. If he is not registered, he will go to the registration or sign up page. He has to input some information like name, password and other relevant information. After signing up he will be able to go to the sign in page to enter the system. After signing in there will be three type of access. Those are admin, moderator and general user. Admin can create, delete, update and retrieve any data or account. Admin will also have to access to block any user. Moderator will have the access to the content management and file management. For general visitor there will be a view access where the user can only view the system and its contents. But if they want to download or download any content, they have to sign in there. After signing in they will have the access to pro features. In pro feature they can attend the mock tests from the question bank we have in our database. The admin will have the access to all the features like content, file management, University management, College management, department management, User management, role management and pro features. After using the system all kind of user will be able to log out from the system using a logout function.

### 7.3 Prototype of new system (OAIHUB)



**Figure 12: Architectural design: MVC architecture**

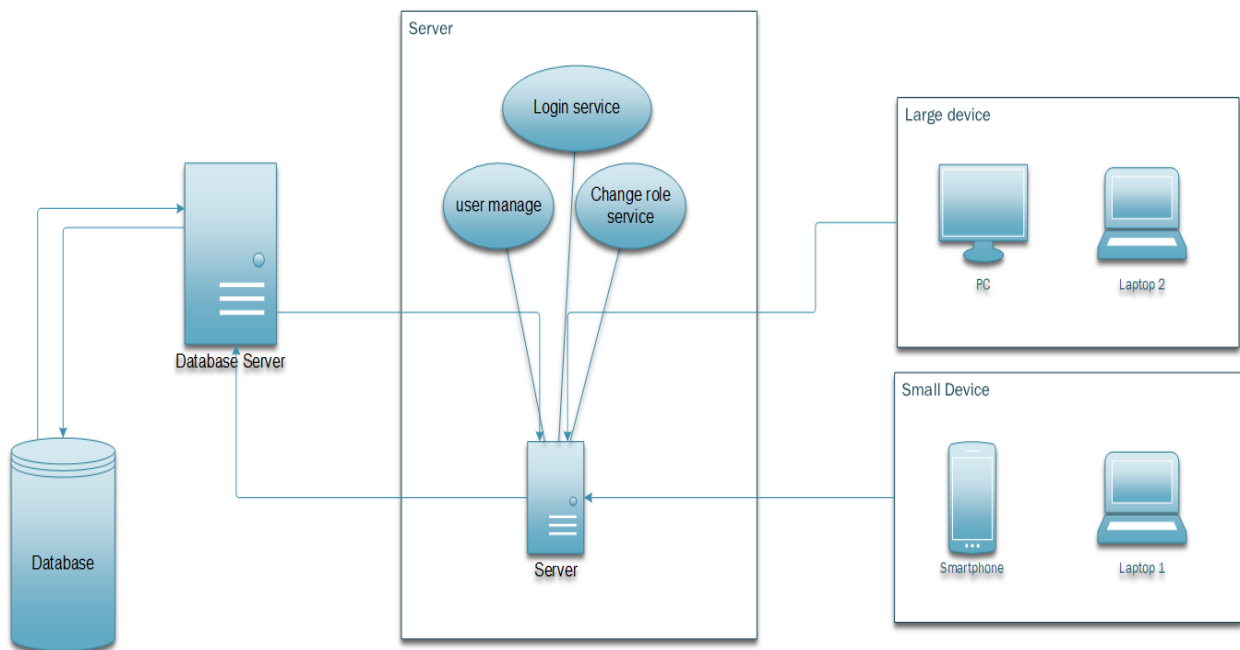
#### Description:

It's a system architecture diagram. We followed the design pattern of MVC for our system design. When a user make request our system through webpage then the webpage make a http request to the controller as a first step. Controller then process the http request and generate result in the model. Model gets data from the database. Here model is acting like a bridge between database and controller. Model does the definition and validation those data which age coming from the database. Then model send data to the controller. Controllers then pass those arrange data to view. View shows those data with a nice user interface through the modern webpage. User

interface which is usually seen by the end user. That request and processing are the backend task. If any user make interaction with UI or make some input then the UI send those inputted data to the controller and controller send them to the model for analysis and arrange and check validation of those data then model send them to database to store those user data safely and securely. Then data base store those data and use shows on the UI that his or her data are securely store in the system database.

#### 7.4 Diagram of the overall architecture (Daffodil Idea Hunt)

Here is the diagram of the overall architecture of “Daffodil Idea Hunt”.



**Figure 13: Overall architecture of this web application.**

## **Chapter 8 - Engineering**

### **8.1 New System Modules**

In this Web Application The newly proposed and under developed modules are fascinating. A total discussion hub, the forum is going to be added and implemented in this system. Some pro features are going to be introduced for which user will have to pay to use. Forum discussion will be much like stack overflow. People can discuss about study, question answers, talk about institutions, post and share job articles and exams, give mock test, find all kinds of resources every student need in every exam, as it will also going to be connected with Education boards of Bangladesh & UGC. So, Students, users will find almost every facility to accommodate.

## 8.2 Use Case

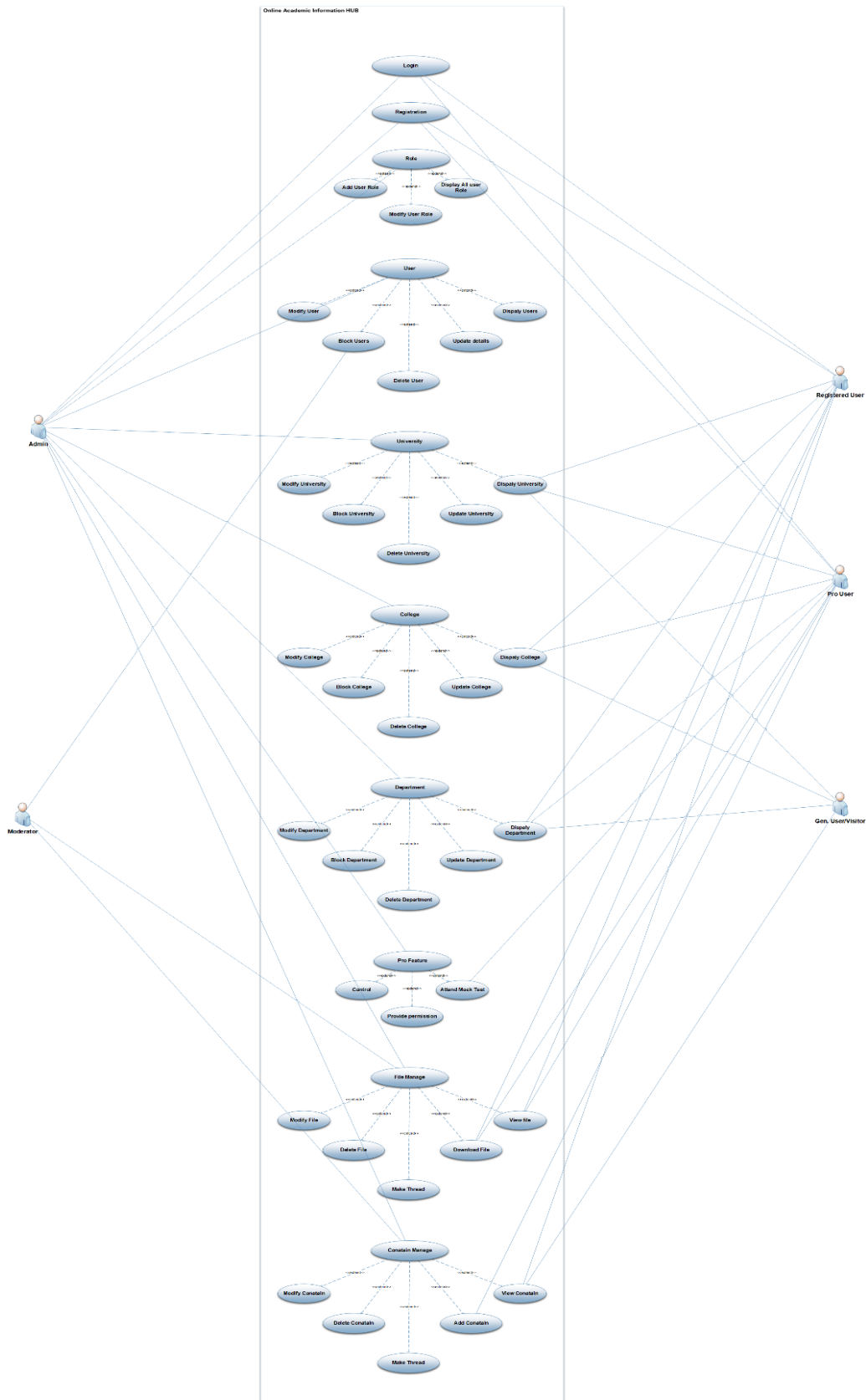


Figure 14: Use case Diagram

## **Description:**

In this information hub there are total 5 types of user. Every user has their own rule to use this system. They have various type of data access in this system such as:

**Admin:** He/she can register themselves into the system and can login by their personal information. They can make and modify role in the system, can add user, display all user role. Modify user, block user, delete user, update user data those also can be accessible by admin. All type of University, College and Department related data can be display, blocked, modify or delete by the admin. They also have some pro feature such as control, provide permission and attend mock test (tester). All necessary file can be handled by the admin panel, modifying files, delete files, make thread, download file and view file are the admin panels task. Contain manage like modify, delete, make thread, add and view contain.

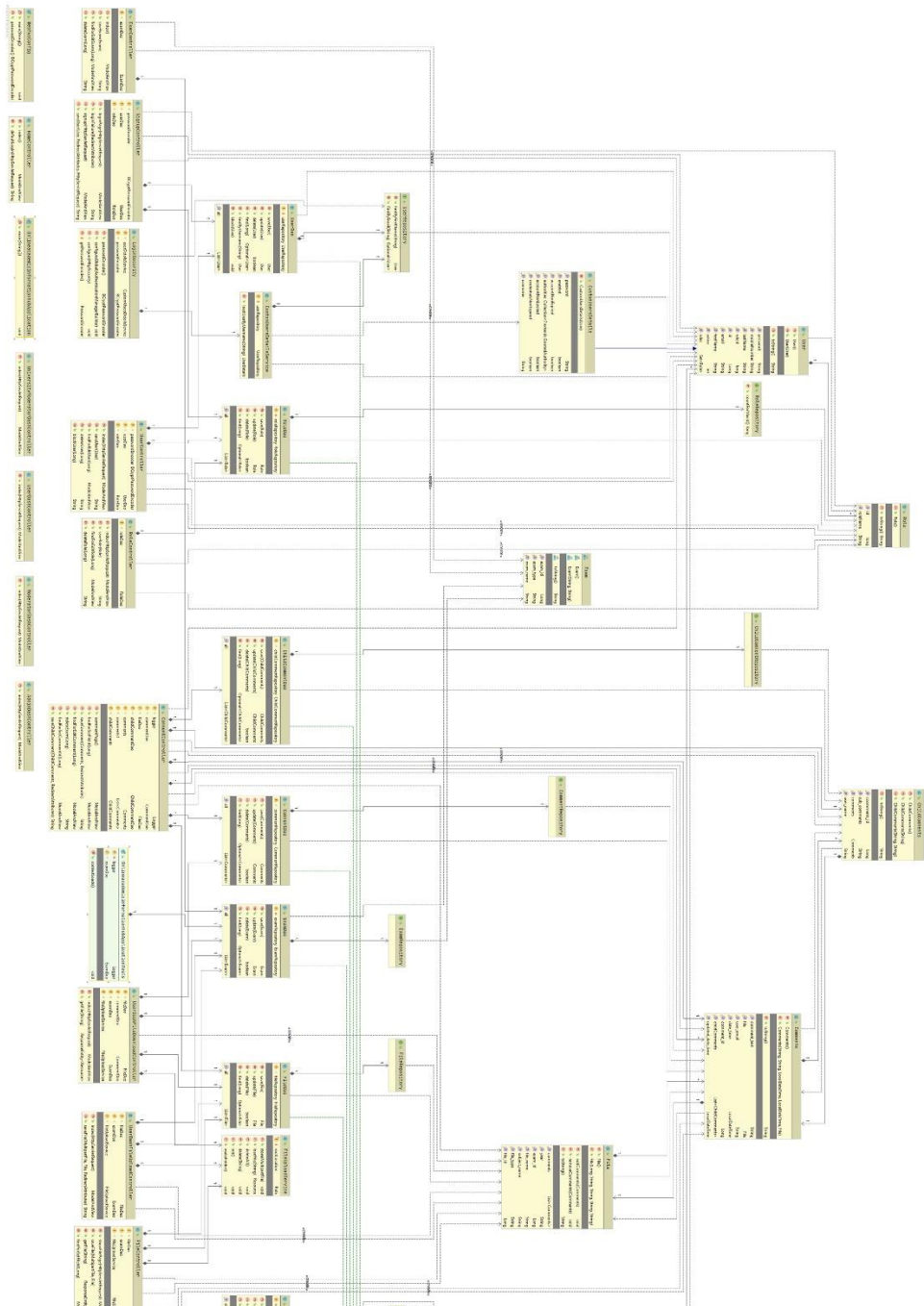
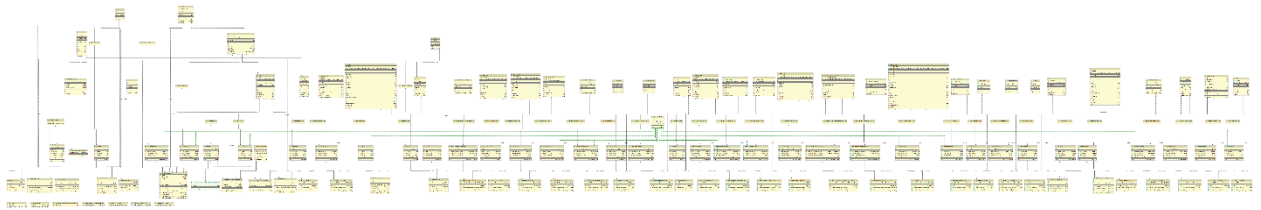
**Moderator:** This panel has less power and access then the admin. Modify user, block user, delete user, update user data those also can be accessible by moderator panel. Necessary file can be also handled by the moderator panel. Modifying files, delete files, make thread, download file and view file are the admin panels task. Contain manage like modify, delete, make thread, add and view contain.

**Register user:** This panel they can login and register in the beginning. They can see university, college and department data. From this panel they can also attend mock test. They will have access to view file and download them. They can able to add content and also view others.

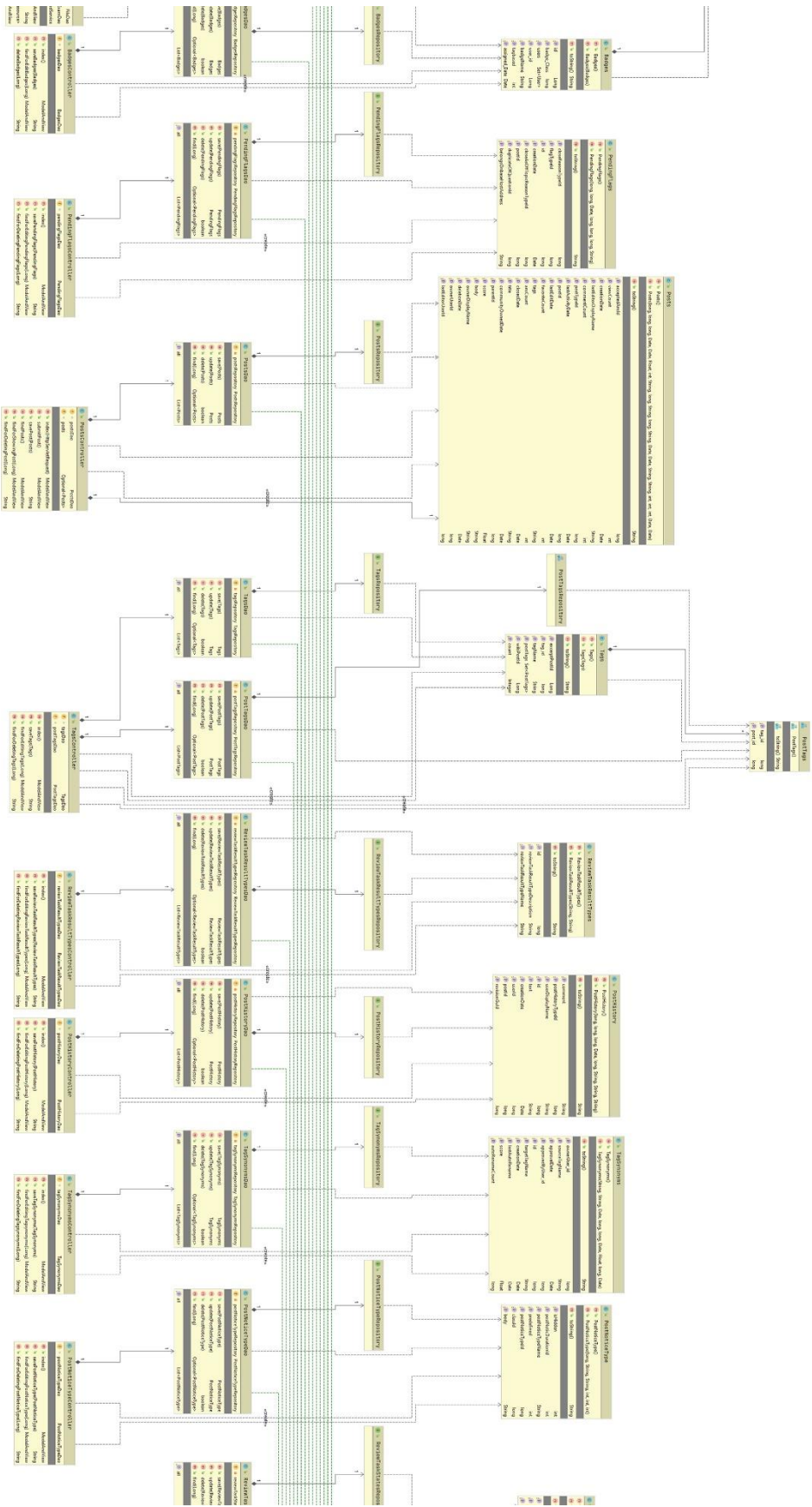
**Pro-user:** This panel members are the special then register member. This panel member can login and register in the beginning. They can view university, college and department all data. From this panel they can attend mock test. They will have access to view file and download them. They can also able to add content and also view others.

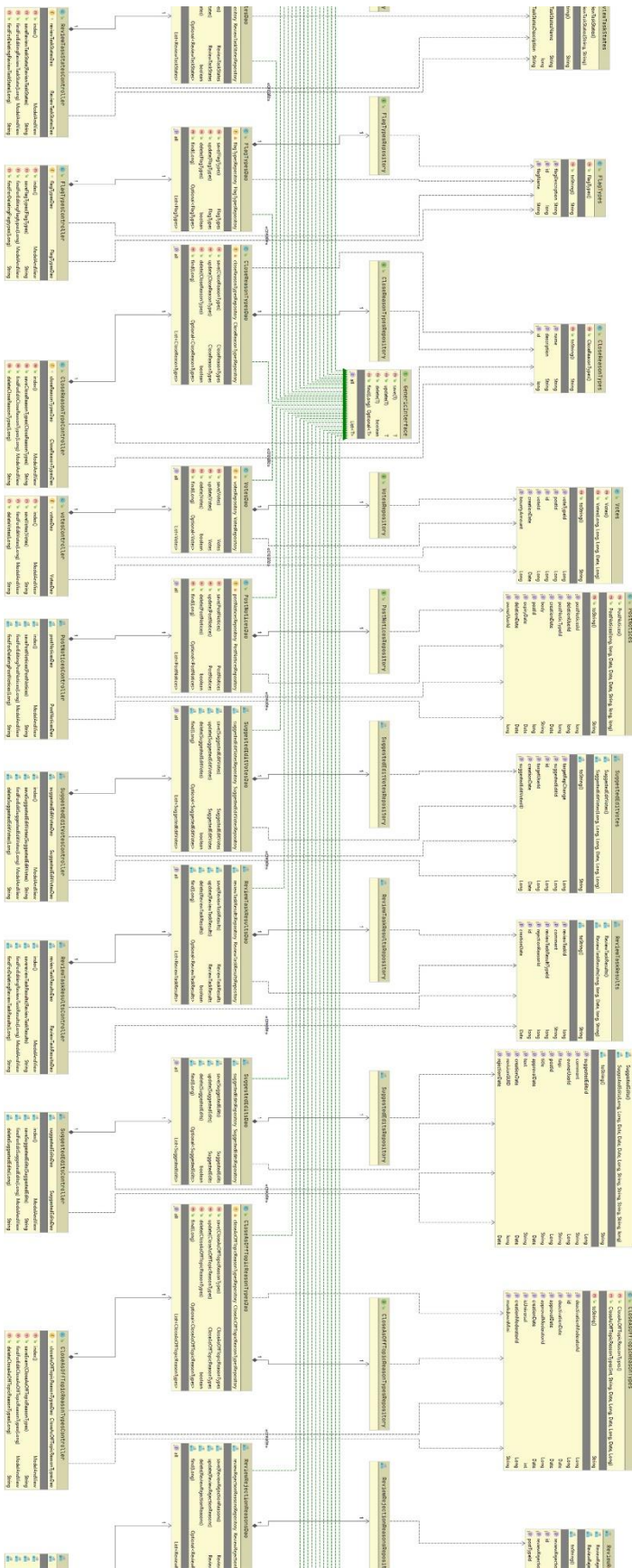
**General user/visitor:** They have the less ability in this system. Visitor can register themselves. Then they can do many things. Without registration they can only view can view university, college, department data and some content.

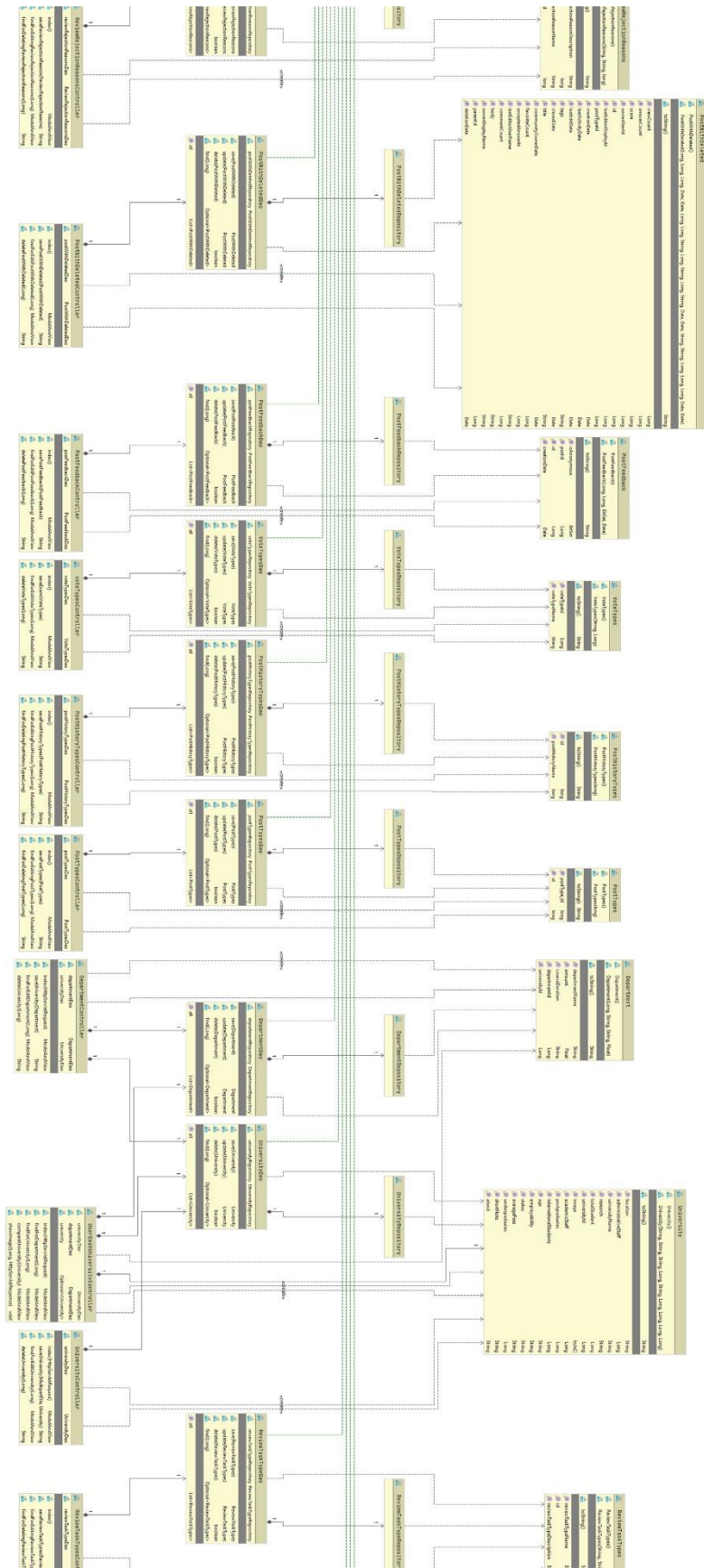
### 8.3 Class Diagram















## **Description:**

Class diagram is that which contains variables, methods, classes, functions, working structure and shows the relationship between each one of them. It has a set of classes, interfaces, collaborations and their relationships. It shows the interactions between the classes that is used in this system. It represents the whole system in a diagram. This class diagram contains many classes like, university, department, user, user detail, files, exam, role etc. and all these class has so many attributes, methods.

Such as, university has university id, university name, location, total student, academic staff etc. Which shows the all information about a university. Department has, department id, department name, course duration, amount and also university id which act as a foreign key here. Department will represent the information of each university's information. User has user id, name, password, email, roles, role id etc. Role has role id, role name. Exam has exam type, exam name. All these classes with their attributes have relationships between them. By using this UML class diagram, we can show the whole system relationship how each of the table interact with each other.

User management system has the accessibility and security layer for every type of user. Admin, moderator, university moderator, user, pro user. All these users have different part of accessibility in the system. Admin has control over the whole system, moderator has access to the part which university and institution is going to be registered in the web application and show their details, they will manage all types of previous questions and answers of all education boards, and all university module information. There will be individual university moderator who will be managing specific university 's information which will be modified based on priority time and demand.

Users can see discussions on the forum, question bank on the forum, talk with people, rate discussion topics, give answers, post job articles, post job exams, and many more. Users who are going to take part in these things will have to register without only seeing things posted in the forum. Everyone can post article in the forum and talk about. Moderators will be monitoring 24 hrs which post will going to be allowed or not in the forum, if there any bad comments are given or not, later which will be put into the AI to monitor each and every second if any bad comments are coming or not, which comments should be given priority or permitted to post. There will be a pro



File has file id. Exam has exam id as primary id. Password reset token has also id has primarily. Organization entity has org id as primary key. Here entity User has many to 1 relation between entity user\_role. Entity user has many to 1 relation between entity role. Entity user role has many to 1 relation to entity password\_reset\_token. Entity hibernate\_sequence has many to 1 relation between persistent\_login. Child\_comments entity has also many to 1 relation between entity comment. entity comment has 1 to many relations to entity files. Entity files has 1 to many relations to exams. Entity University has 1 to many relationships between entity department. Financial entity has 1 to many relations between organization. entity organization has 1 to many relations with entity user\_organization

Now here comes the forum part. There are different and a bunch of entities are utilized in this system. Badge id from badge entity is settled as foreign field in user section. Votes are going to take place through the posts of the forum. People can give an upvote or a downvote to individual post. One user one vote system has designed. Votes are also systemized with the type of vote people can give. The system will show suggestion about editing votes and a result of edited votes. People can give feedback about the post. It's different from comment section. There will be a trash systemization of closed thing topics, post, and more. The 'Close as off topic reason types" will hold the details of users registered and blocked or deleted/banned with specific time calculation and the typical reason the user's banned for. And a post with deleted section where all kinds of specific details of the post/post will be captured. How the post was, post type, comments and everything about it. Posts can be flagged or reposted by users based on some flag types or custom flagging types. When posts are flagged a review section as created for moderators and admins. Posts will be reviewed and a date will be stored for it. The flagged post will be reviewed by a systematic way which will follow some rules. After review a detailed information about the review will be stored for future consultation. Furthermore, a Post History section will be stored and monetized. Every post will be categorized by post history types. Well now a very important part is Tagging posts with specific keywords for making it relevant with the discussion topics. Tags will be connected by post id. A synonym section of tags is also available for posts. Last but not the least the most important part is Post / Threads by which the Forum term is established. All the posts are utilized by user ids,

post types, comments, post links, post notice, post notice types. All the Entities are very much connected to each other in the system.

## 8.5 Sequence Diagram

### a. As an admin

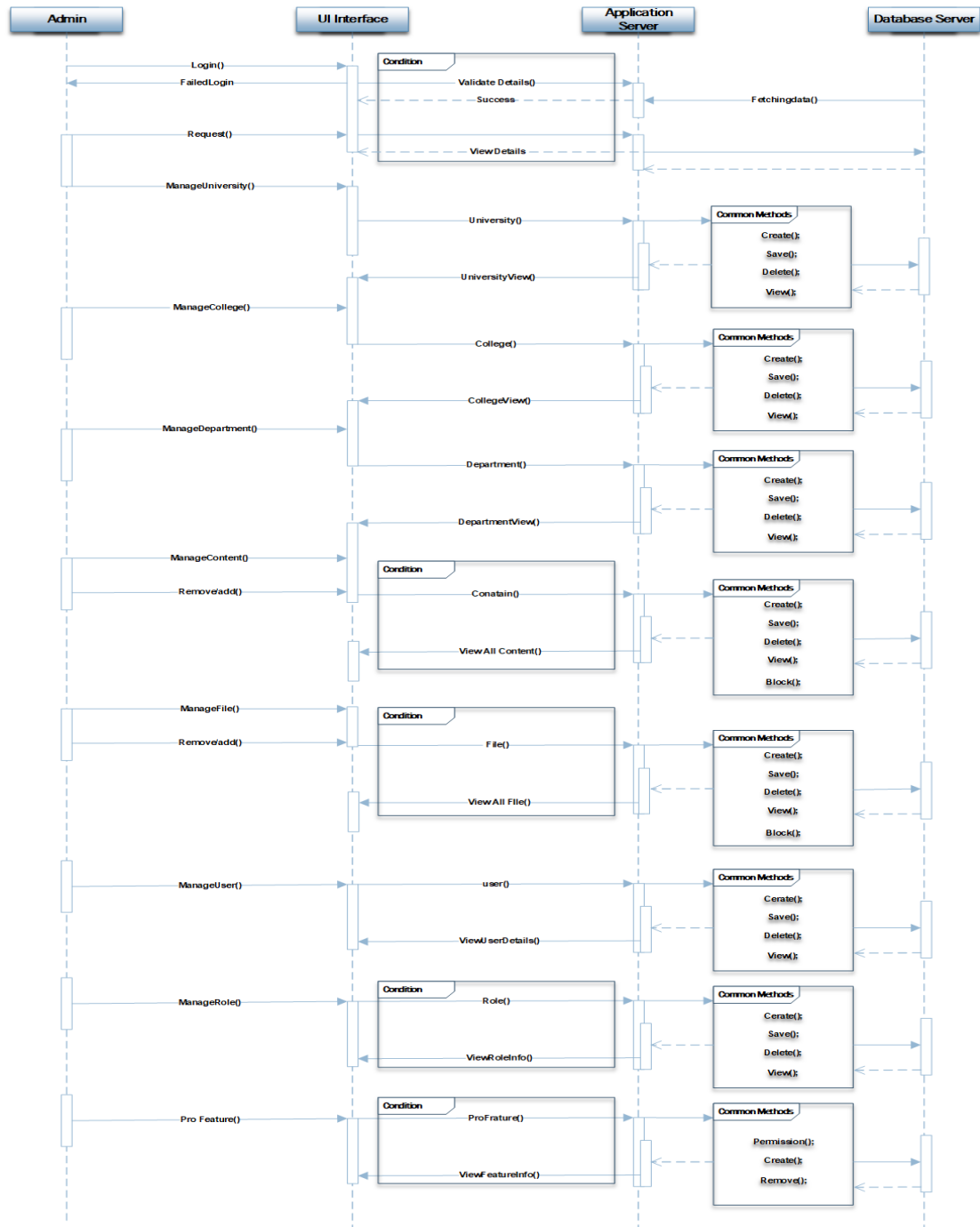


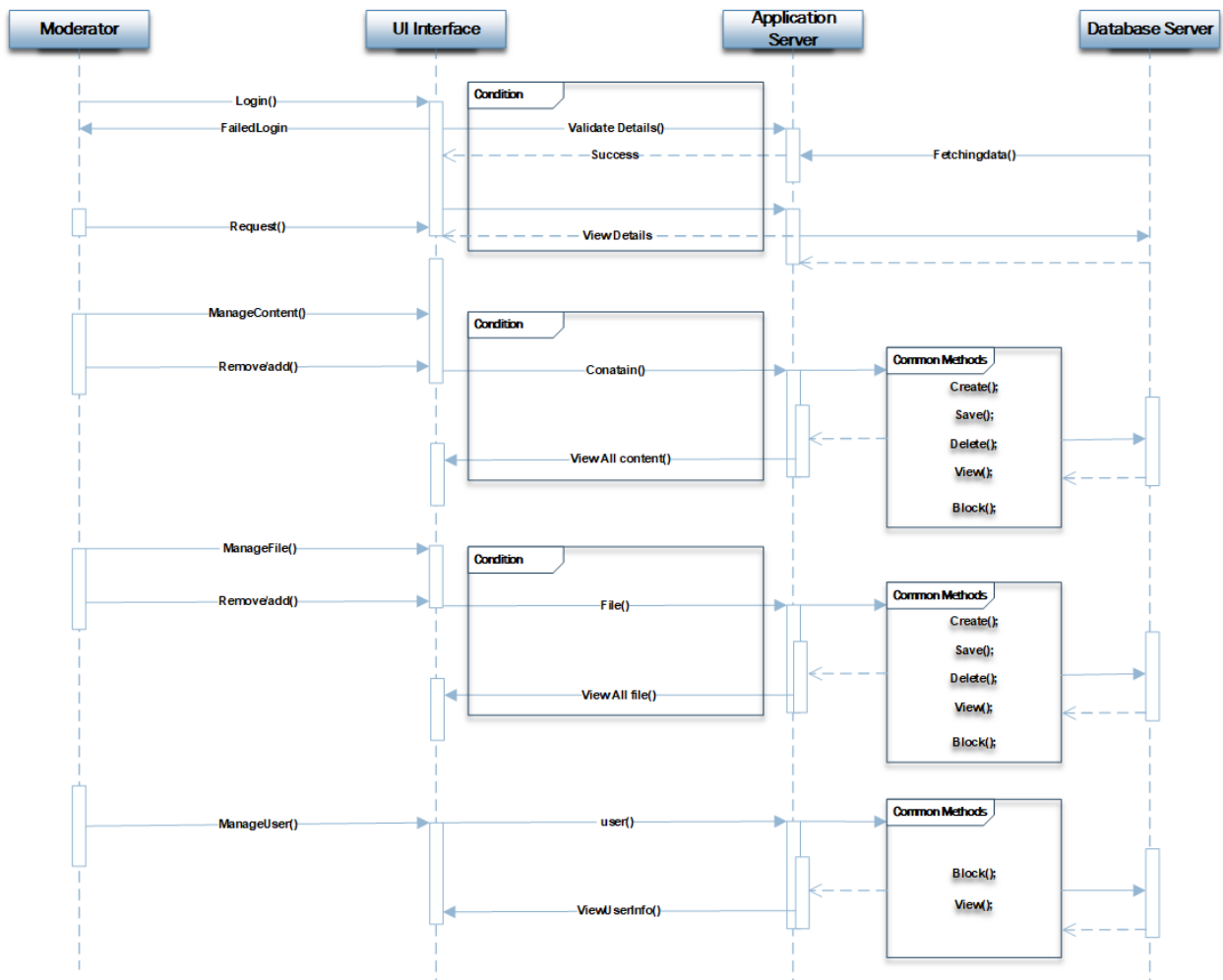
Figure 17: Sequence Diagram for admin



**Description:**

This diagram shows that the admin request for login to the admin portal. It fetches the data which stored on the database server. Database stored the information of the school, college and university. Admin can view the detail information of the institution that are stored on the database server. Admin can manage the university, college, and department. And he can also remove any user from the system. The files which are uploaded by the user are managed by admin and he can manage or remove any files from the database server.

## b. As a moderator



**Figure 18: Sequence Diagram for moderator**

### Description:

Moderator request to login into the system. It fetches data from database and give them login onto the system. Moderator can manage the files, remove any files from the server and he also manage the user. He can approve any user to use the system or block them from the system if any abusive is occurred by them.

c. As a user

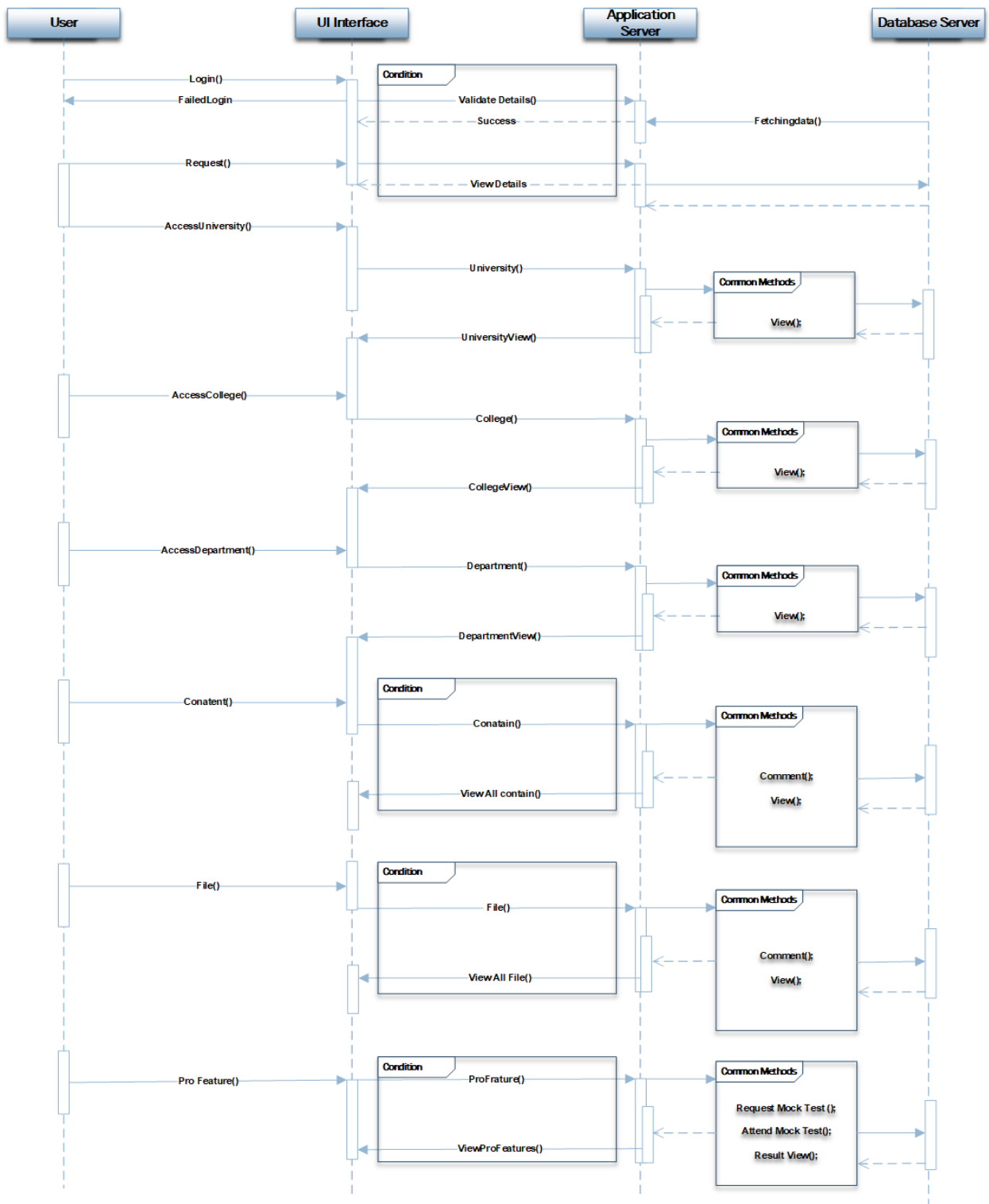


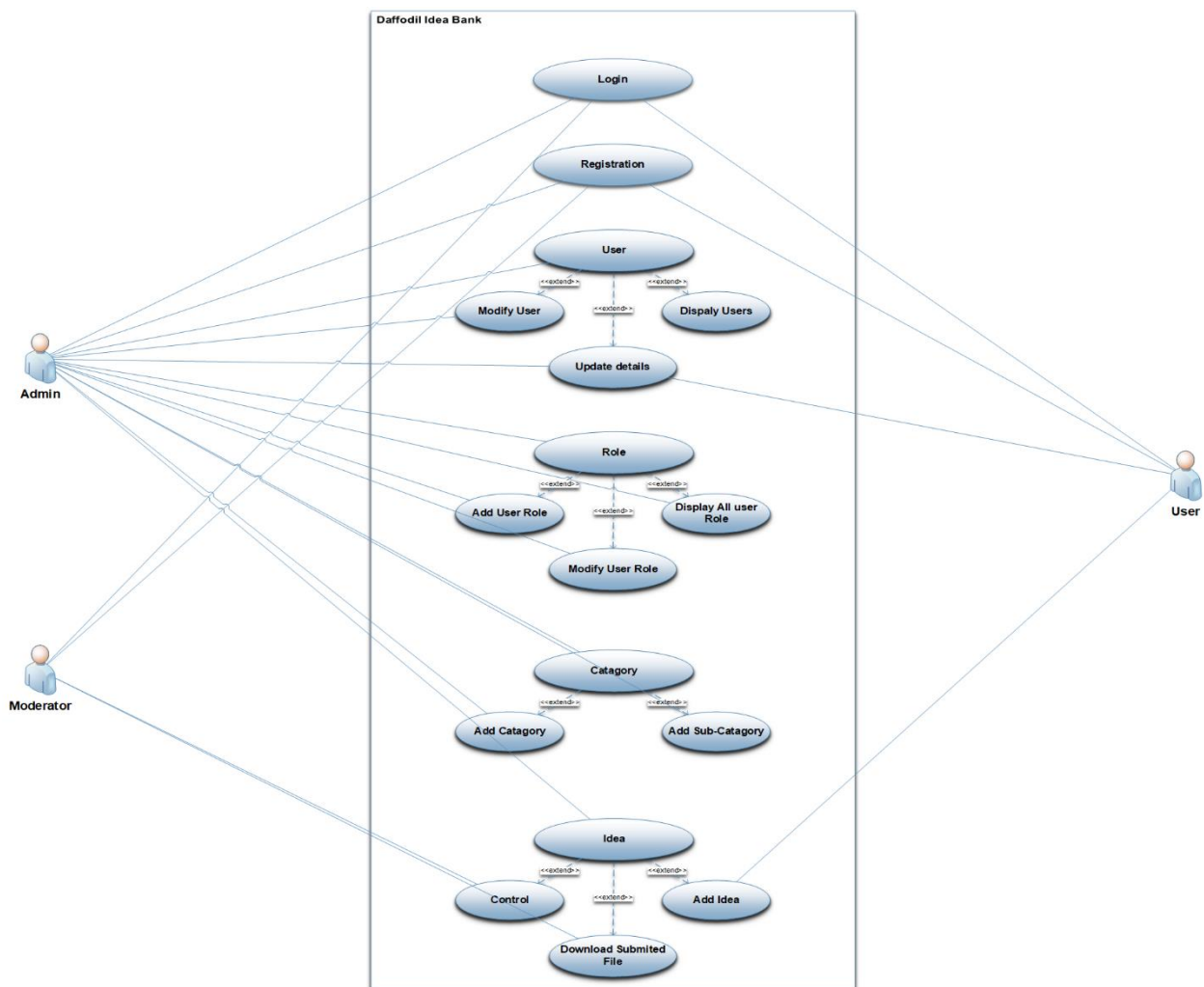
Figure 19: Sequence Diagram for user

## Description:

To login into the system as a user first they have to sign up. After that, the user information will be stored on the database. Every time when user request to login it fetch the data from the database server and give them approval to login. User can view the detail information of university, school or college. They can upload any file which are approved by the moderator. User can request for any test to participate on that. All these information, question bank and answer are store on database server. After any request it fetch the data from database and show the detail information to user.

## 8.6 Use Cases of Daffodil Idea Hunt

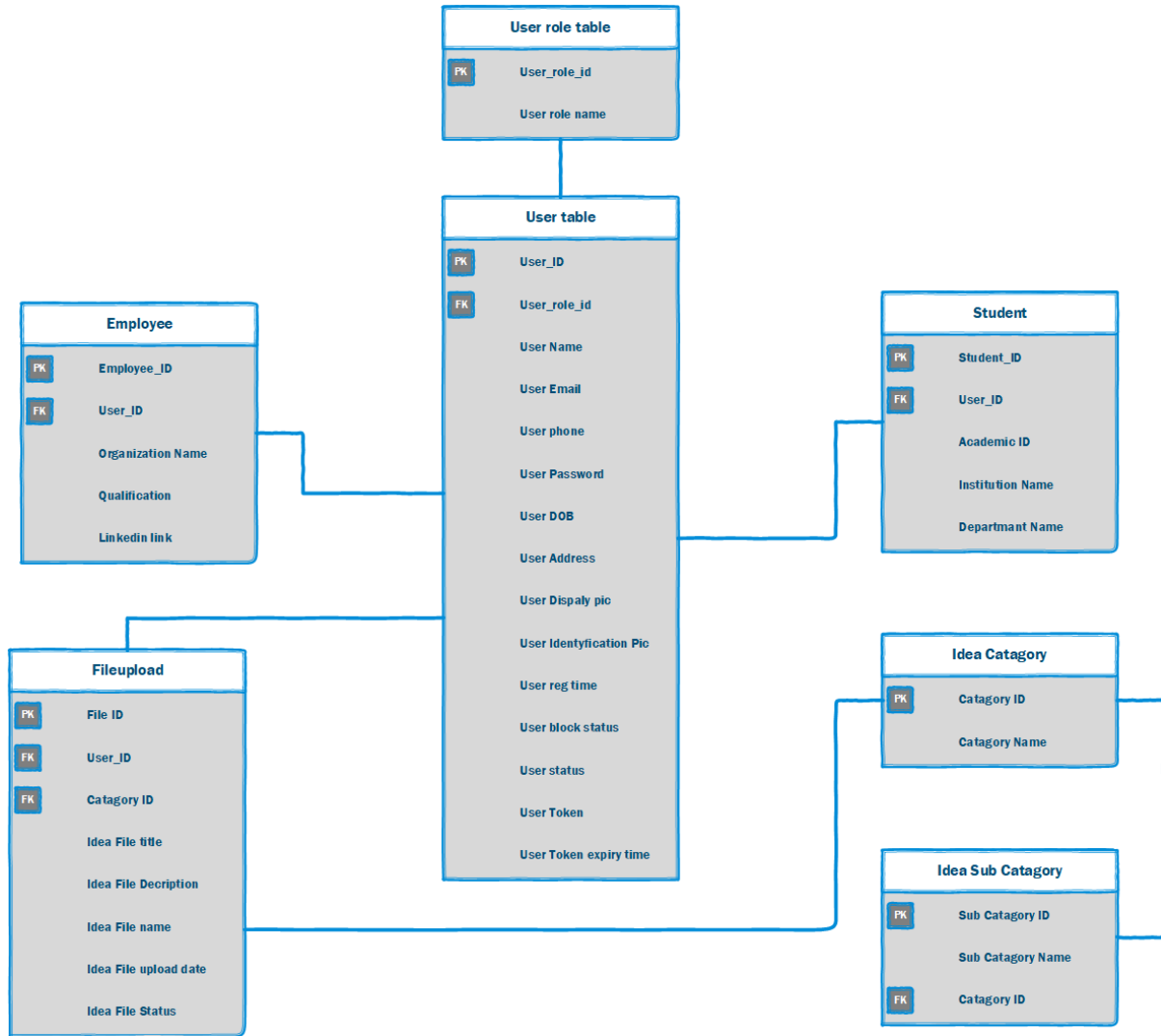
Here is the use case diagram of “Daffodil Idea Hunt”.



**Figure 20: Use Cases of Daffodil Idea Hunt**

## 8.7 ERD Diagram of Daffodil Idea Hunt

Here is the ERD diagram of “Daffodil Idea Hunt”.



**Figure 21: ERD Diagram of Daffodil Idea Hunt**

## 8.8 Activity diagram of Daffodil Idea Hunt

Here is the Activity diagram of “Daffodil Idea Hunt”.

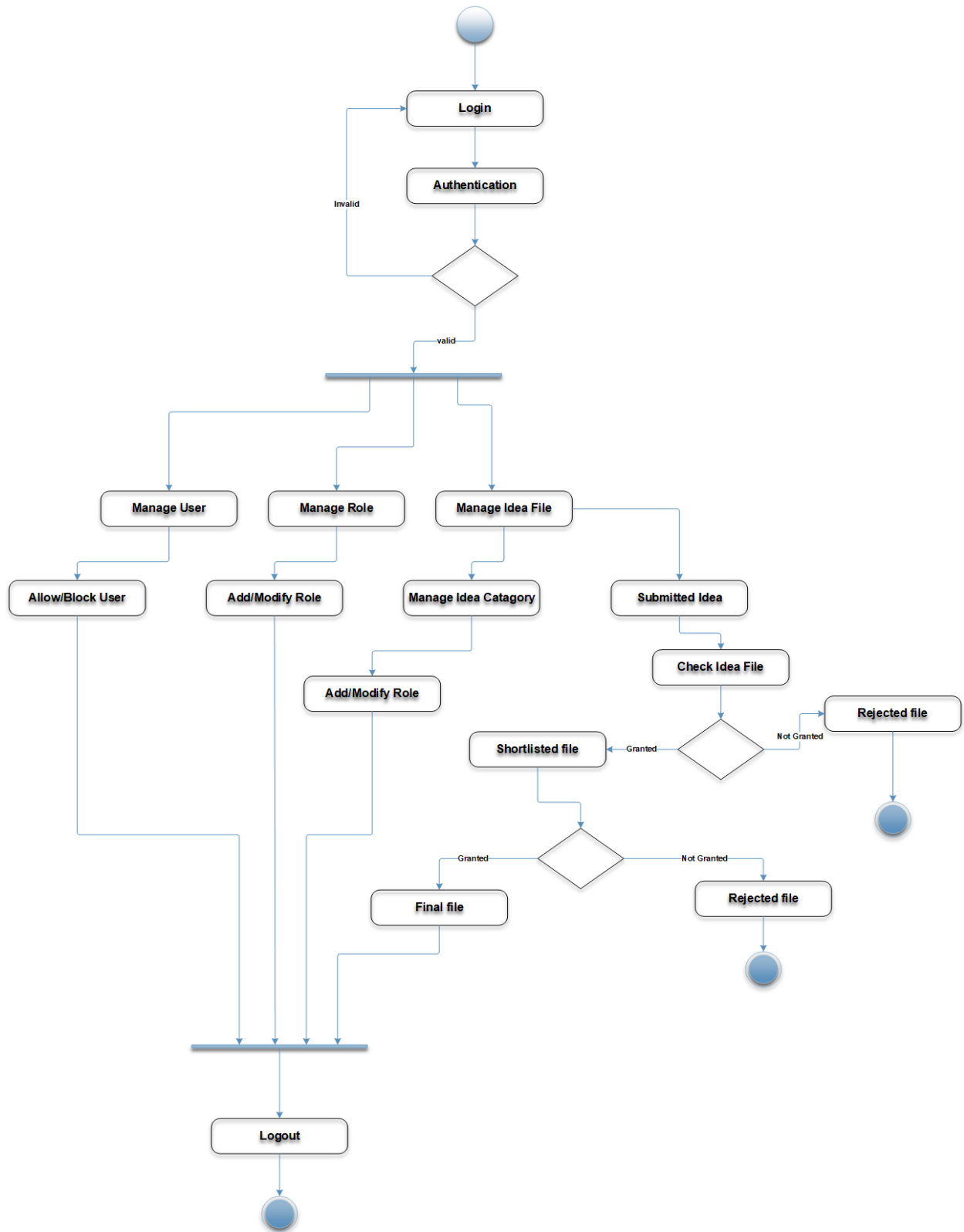


Figure 22: Activity diagram of Daffodil Idea Hunt

# Chapter 9 - Deployment / Development

## 9.1 OAIHUB

Core Module Coding Samples

### 9.1.1 User Management

In this section the user, Admin & Moderator management section Has been coded. The code architecture of user management is given bellow.

#### a. User Class

Here the Class is created to take user information in the system.

#### ▪ Libraries are Imported here

```
package ac.daffodil.model;  
  
import org.hibernate.validator.constraints.Length;  
  
import javax.persistence.*;  
import javax.validation.constraints.Email;  
import javax.validation.constraints.NotEmpty;  
import java.util.Set;
```

- User Entity is Created

```
@Entity
@Table(name = "user")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "user_id")
    private Long id;

    @Column(name = "email")
    @Email(message = "*Please provide a valid Email")
    @NotEmpty(message = "*Please provide an email")
    private String email;

    @Column(name = "password")
    @Length(min = 4, message = "*Your password must have at least 4 characters")
    private String password;

    @Column(name = "firstName")
    @NotEmpty(message = "*Please provide your first name")
    private String firstName;

    @Column(name = "last_name")
    @NotEmpty(message = "*Please provide your last name")
    private String lastName;

    @Column(name = "mobile")
    @NotEmpty(message = "*Please provide your mobile number")
    private String mobileNumber;

    @Column(name = "active")
    private int active;

    @Column(name = "roleId")
    private long roleId;

    @ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    @JoinTable(name = "user_role", joinColumns = @JoinColumn(name = "user_id",
        referencedColumnName = "user_id"), inverseJoinColumns = @JoinColumn(name
= "role_id",
        referencedColumnName = "role_id"))
    private Set<Role> roles;

    public User() {

    }
}
```



- **Variables or Field Data of Entity**

```
public User(User user) {  
    this.id = user.getId();  
    this.email = user.getEmail();  
    this.firstName = user.getFirstName();  
    this.active = user.getActive();  
    this.roleId = user.getRoleId();  
    this.roles = user.getRoles();  
    this.mobileNumber = user.getMobileNumber();  
    this.lastName = user.getLastName();  
    this.password = user.getPassword();  
}
```

- **Getter Setters for taking Data & Saving them into Database**

```
public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public String getMobileNumber() {
    return mobileNumber;
}

public void setMobileNumber(String mobileNumber) {
    this.mobileNumber = mobileNumber;
}

public int getActive() {
    return active;
}

public void setActive(int active) {
```

- A Method is created to Show all those data

```
@Override
public String toString() {
    return "User{" +
        "id=" + id +
        ", email='" + email + '\'' +
        ", password='" + password + '\'' +
        ", firstName='" + firstName + '\'' +
        ", lastName='" + lastName + '\'' +
        ", mobileNumber='" + mobileNumber + '\'' +
        ", active=" + active +
        ", roleId=" + roleId +
        ", roles=" + roles +
        '}';
}
}
```

## b. User Dao Class

- The Data Access object is to fetch data from database of user and modify them.
- This Class implements the generic class where all the method signatures remain same to call upon all the class.

```
@Service
public class UserDao implements GenericInterface<User> {

    @Qualifier("userRepository")
    @Autowired
    private UserRepository userRepository;

    @Override
    public User save(User user) {
        userRepository.save(user);
        return user;
    }

    @Override
    public User update(User user) {
        userRepository.save(user);
        return user;
    }

    @Override
    public boolean delete(User user) {
        userRepository.delete(user);
        return true;
    }

    @Override
    public List<User> getAll() {
        return userRepository.findAll();
    }

    @Override
    public Optional<User> find(Long id) {
        return userRepository.findById(id);
    }

    public Optional<User> findByUsername(String userName) {
        return userRepository.findByFirstName(userName);
    }
}
```

### c. User Repository Class

- The JPA Repository dependency class is inherited

```
package ac.daffodil.repository;

import ac.daffodil.model.University;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository("universityRepository")
public interface UniversityRepository extends JpaRepository<University, Long> {
}
```

#### d. Role Class

- Role will be added by this class code objects

```
package ac.daffodil.model;
import javax.persistence.*;

@Entity
@Table(name = "role")
public class Role {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "role_id")
    private long id;

    @Column(name = "roleName")
    private String roleName;

    public Role() {
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getRoleName() {
        return roleName;
    }

    public void setRoleName(String roleName) {
        this.roleName = roleName;
    }

    @Override
    public String toString() {
        return "Role{" +
```

## e. Role Dao Class

Here Libraries, role class & role repository has been imported to call in action

```
package ac.daffodil.dao;

import ac.daffodil.model.Role;
import ac.daffodil.repository.RoleRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class RoleDao implements GenericInterface<Role> {

    @Qualifier("roleRepository")
    @Autowired
    private RoleRepository roleRepository;

    @Override
    public Role save(Role role) {
        roleRepository.save(role);
        return role;
    }

    @Override
    public Role update(Role role) {
        roleRepository.save(role);
        return role;
    }
}
```

```

@Override
public boolean delete(Role role) {
    roleRepository.delete(role);
    return true;
}

@Override
public List<Role> getAll() {
    return roleRepository.findAll();
}

@Override
public Optional<Role> find(Long id) {
    return roleRepository.findById(id);
}
}

```

#### f. Role Repository Class

- The JPA Repository dependency class is inherited

```

package ac.daffodil.repository;

import ac.daffodil.model.Role;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;

@Repository("roleRepository")
public interface RoleRepository extends JpaRepository<Role, Long> {

    @Query(value = "SELECT MAX(ROLE_ID) FROM ROLE", nativeQuery = true)
    long countForMaxId();
}

```



### g. CustomUserDetailsService Class

- This class fetches all the emails and search for existing emails to login or signup process

```
@Service
public class CustomUsersDetailsService implements UserDetailsService {

    @Autowired
    @Qualifier("userRepository")
    private UserRepository userRepository;

    @Override
    public UserDetails loadUserByUsername(String email) throws
UsernameNotFoundException {
        Optional<User> optionalUsers = userRepository.findByEmail(email);

        optionalUsers
            .orElseThrow(() -> new UsernameNotFoundException("Username not
found"));
        return optionalUsers
            .map(CustomUsersDetails::new).get();
    }
}
```

### h. UserDashController Class

- All information is fetched and shown through this program in the user dashboard

```
@Controller
@RequestMapping("/user")
public class userDashController {

    @RequestMapping(value = {"/userDashPage"}, method = RequestMethod.GET)
    public ModelAndView index(HttpServletRequest request) {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("user/userDash");
        return modelAndView;
    }
}
```

## i. userController Class

- This class is for moderator & admin to control user system

```
@Controller
@RequestMapping("/user")
public class UserController {
    @Autowired
    BCryptPasswordEncoder passwordEncoder;

    @Autowired
    UserDao userDao;

    @Autowired
    RoleDao roleDao;
```

- method for fetching user & role information

```
@RequestMapping(value = {"/userPage"}, method = RequestMethod.GET)
public ModelAndView index(HttpServletRequest request) {
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.addObject("users", userDao.getAll());
    modelAndView.addObject("roles", roleDao.getAll());
    //     for (Role role : roleDao.getAll()) {
    //         System.out.println(role.getRoleName());
    //     }
    modelAndView.addObject("message", request.getParameter("message"));
    modelAndView.addObject("newUser", new User());
    modelAndView.addObject("newRole", new Role());
    modelAndView.setViewName("admin/adminUser");
    return modelAndView;
}
```

- **Method for Saving a user & his/her role**

```
@RequestMapping(value = "/saveUser", method = RequestMethod.POST)
public String saveUser(User user) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<Role> role = roleDao.find(user.getRoleId());
    Set<Role> roles = new HashSet<Role>();
    roles.add(role.get());
    user.setRoles(roles);
    user.setPassword(passwordEncoder.encode(user.getPassword()));
    userDao.save(user);
    modelAndView.addObject("message", " Data Has Been Saved...");
    return "redirect:/user/userPage";
}
```

- **Method for editing a user information**

```
@RequestMapping(value = {"/findForEditUser/{id}"}, method = RequestMethod.GET)
public ModelAndView findForEditUser(@PathVariable(required = true, name = "id")
Long id) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<User> user = userDao.find(id);
    modelAndView.addObject("newUser", user.get());
    modelAndView.addObject("users", userDao.getAll());
    modelAndView.addObject("roles", roleDao.getAll());
    modelAndView.setViewName("admin/adminUser");
    return modelAndView;
}
```

- **Method for deleting a user completely**

```
@RequestMapping(value = "/deleteUser/{id}", method = RequestMethod.GET)
public String deleteUser(@PathVariable(required = true, name = "id") Long id) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<User> user = userDao.find(id);
    userDao.delete(user.get());
    modelAndView.addObject("message", " Data Has Been Deleted...");
    return "redirect:/user/userPage";
}
}
```

## j. Signup Controller Class

- Here all the user classes have been imported
- All the Data Access Object Classes are imported
- All the information is processed to assign a new user information

```
package ac.daffodil.controller;

import ac.daffodil.dao.RoleDao;
import ac.daffodil.dao.UserDao;
import ac.daffodil.model.Role;
import ac.daffodil.model.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import javax.servlet.http.HttpServletRequest;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
```

```
@Controller
public class signupController {

    @Autowired
    BCryptPasswordEncoder passwordEncoder;

    @Autowired
    UserDao userDao;

    @Autowired
    RoleDao roleDao;
```

- Login Page & backend Operation method Attached

```
@RequestMapping(value = {"/login"}, method = RequestMethod.GET)
public ModelAndView loginPage(HttpServletRequest request) {
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.setViewName("fragments/login");
    return modelAndView;
}
```

- **Redirecting to login method**

```
@GetMapping("/loginFailure")
public String loginFailure(RedirectAttributes redirectAttributes) {
    redirectAttributes.addFlashAttribute("message", "Invalid Username or
    Password...");
    redirectAttributes.addFlashAttribute("alertClass", "alert-danger");
    return "redirect:/login";
}
```

- **Signup Method**

```
@RequestMapping(value = {"/signup"}, method = RequestMethod.GET)
public ModelAndView signup(HttpServletRequest request) {
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.addObject("newUser", new User());
    modelAndView.addObject("roles", roleDao.getAll());
    modelAndView.setViewName("fragments/signup");
    return modelAndView;
}
```

- Save the newly signed up user in database

```
@RequestMapping(value = "/saveUser", method = RequestMethod.POST)
public String saveUser(User user, RedirectAttributes redirectAttributes,
HttpServletRequest request) {
    try {
        user.setActive(1);
        List<Role> roles = roleDao.getAll();
        for (Role role : roles) {
            if (role.getRoleName().equals("user")) {
                user.setRoleId(role.getId());
                Set<Role> roleSet = new HashSet<Role>();
                roleSet.add(role);
                user.setRoles(roleSet);
            }
        }
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        userDao.save(user);
        redirectAttributes.addFlashAttribute("message", "User Saved
Successfully... ");
        redirectAttributes.addFlashAttribute("alertClass", "alert-success");
        return "redirect:/signup";
    } catch (Exception e) {
        redirectAttributes.addFlashAttribute("message", "Error... Please Cheack
and input Correct Data.");
        redirectAttributes.addFlashAttribute("alertClass", "alert-danger");
        return "redirect:/signup";
    }
}
```

## k. Role Controller Class

- Role management methods are written here

```
@Controller
@RequestMapping("/role")
public class RoleController {

    @Autowired
    RoleDao roleDao;

    @RequestMapping(value = {"/rolePage"}, method = RequestMethod.GET)
    public ModelAndView index(HttpServletRequest request) {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.addObject("roles", roleDao.getAll());
        modelAndView.addObject("message", request.getParameter("message"));
        modelAndView.addObject("newRole", new Role());
        modelAndView.setViewName("admin/adminRole");
        return modelAndView;
    }
}
```

- Save new role

```
@RequestMapping(value = "/saveRole", method = RequestMethod.POST)
public String saveRole(Role newRole) {
    ModelAndView modelAndView = new ModelAndView();
    roleDao.save(newRole);
    modelAndView.addObject("message", " Data Has Been Saved...");
    return "redirect:/role/rolePage";
}
```

- Edit role

```
@RequestMapping(value = {"/findForEditRole/{id}"}, method = RequestMethod.GET)
public ModelAndView findForEditRole(@PathVariable(required = true, name = "id")
Long id) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<Role> role = roleDao.find(id);
    modelAndView.addObject("newRole", role.get());
    modelAndView.addObject("roles", roleDao.getAll());
    modelAndView.setViewName("admin/adminRole");
    return modelAndView;
}
```

- **Delete role**

```
@RequestMapping(value = "/deleteRole/{id}", method = RequestMethod.GET)
public String deleteRole(@PathVariable(required = true, name = "id") Long id) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<Role> role = roleDao.find(id);
    roleDao.delete(role.get());
    modelAndView.addObject("message", " Data Has Been Deleted...");
    return "redirect:/role/rolePage";
}
}
```

## I. HomeController Class

- **This class controls, models & shows the homepage / dashboard in a designed way**

```
@Controller
public class HomeController {

    @RequestMapping(value = {"/"}, method = RequestMethod.GET)
    public ModelAndView index() {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("fragments/layout");
        return modelAndView;
    }
}
```

- **Admin & user login controller**

```
@RequestMapping(value = {"/defaultLogin"}, method = RequestMethod.GET)
public String defaultLogin(HttpServletRequest request) {
    if (request.isUserInRole("admin")) {
        return "redirect:/admin/adminDashPage";
    }
    return "redirect:/user/userDashPage";
}
}
```



### m. AdminDashController class

- This class controls how the admin dashboard will show up

```
@Controller
@RequestMapping("/admin")
public class adminDashController {
    @RequestMapping(value = {"/adminDashPage"}, method = RequestMethod.GET)
    public ModelAndView index(HttpServletRequest request) {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.addObject("message", request.getParameter("message"));
        modelAndView.setViewName("admin/adminDash");
        return modelAndView;
    }
}
```

## 9.1.2 Configuration

### a. Login Security Class

```
@EnableJpaRepositories(basePackageClasses = UserRepository.class)
@Configuration
@EnableWebSecurity
public class LoginSecurity extends WebSecurityConfigurerAdapter {

    @Autowired
    CustomUsersDetailsService userDetailsService;

    @Autowired
    BCryptPasswordEncoder passwordEncoder;

    @Bean
    public BCryptPasswordEncoder passwordEncoder() {
        BCryptPasswordEncoder bCryptPasswordEncoder = new BCryptPasswordEncoder();
        return bCryptPasswordEncoder;
    }

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder);
    }
}
```

- **Security & Authorization method for users**

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
        .csrf()
        .disable().authorizeRequests()
        .antMatchers("/", "/login").permitAll()

        .antMatchers("/admin/**").hasAnyRole("admin").and().authorizeRequests()
        .antMatchers("/user/**").hasAnyRole("admin", "user")

        .and().authorizeRequests().and().exceptionHandling().accessDeniedPage("/403")
        .and().formLogin()
        .loginPage("/login")
        .defaultSuccessUrl("/defaultLogin")
        .failureUrl("/loginFailure")
        .usernameParameter("username")
        .passwordParameter("password")
        .and().logout()
        .logoutUrl("/logout")
        .logoutSuccessUrl("/");
}
```

- Password encryption Method & Algorithm imported from java library
- If password matches with the stored password in the database, user will be logged in

```
private PasswordEncoder getPasswordEncoder() {  
    return new PasswordEncoder() {  
        @Override  
        public String encode(CharSequence charSequence) {  
            return charSequence.toString();  
        }  
  
        @Override  
        public boolean matches(CharSequence charSequence, String s) {  
            return true;  
        }  
    };  
}
```

## b. WebMvcConfig Class

- Ensures security by using this design pattern

```
@Configuration
public class WebMvcConfig extends WebMvcConfigurerAdapter {

    @Bean
    public BCryptPasswordEncoder passwordEncoder() {
        BCryptPasswordEncoder bCryptPasswordEncoder = new BCryptPasswordEncoder();
        return bCryptPasswordEncoder;
    }

    public static void main(String[] args) {

        BCryptPasswordEncoder bCryptPasswordEncoder = new BCryptPasswordEncoder();
        System.out.println(bCryptPasswordEncoder.encode("1234"));
    }
}
```

## 9.1.3 Generic Interface

- a. This interface is used to hold all similar methods & signature in one place & use them in all the necessary classes & codes where it needs to be used

```
public interface GenericInterface<T> {

    T save(T val);

    T update(T val);

    boolean delete(T val);

    List<T> getAll();

    Optional<T> find(Long id);
}
```

## 9.1.4 Exam Management

### a. Exam Class

- This class is creating for creating an entity named exam and to save exam details in database by following some methods, objects, variables

```
package ac.daffodil.model;

import javax.persistence.*;
import javax.validation.constraints.NotEmpty;

@Entity
@Table(name = "exam")
public class Exam {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "exam_id")
    private Long exam_id;

    @Column(name = "exam_type")
    @NotEmpty(message = "*please select an examtype")
    private String exam_type;

    @Column(name = "exam_name")
    @NotEmpty(message = "*please select exam name")
    private String exam_name;
}
```

- method for selecting exam type

```
public Exam() {
}

public Exam(@NotEmpty(message = "*please select an examtype") String exam_type,
@NotEmpty(message = "*please select exam name") String exam_name) {
    this.exam_type = exam_type;
    this.exam_name = exam_name;
}
```

- **taking inputs for exam files**

```
public Long getExam_id() {
    return exam_id;
}

public void setExam_id(Long exam_id) {
    this.exam_id = exam_id;
}

public String getExam_type() {
    return exam_type;
}

public void setExam_type(String exam_type) {
    this.exam_type = exam_type;
}

public String getExam_name() {
    return exam_name;
}

public void setExam_name(String exam_name) {
    this.exam_name = exam_name;
}
```

- **Showing exam details**

```
@Override
public String toString() {
    return "Exam{" +
        "exam_id=" + exam_id +
        ", exam_type='" + exam_type + '\'' +
        ", exam_name='" + exam_name + '\'' +
        '}';
}
}
```

- b. Exam Repository**

- **Exam JPA dependency repository inheritance**

```

package ac.daffodil.repository;

import ac.daffodil.model.Exam;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ExamRepository extends JpaRepository<Exam, Long> {
}

```

### c. Exam Dao

```

@Repository
@Transactional
public class ExamDao implements GenericInterface<Exam> {
    @Autowired
    private ExamRepository examRepository;
}

```

- Save exam information method

```

@Override
public Exam save(Exam exam) {
    examRepository.save(exam);
    return exam;
}

```

- Update exam information method

```

@Override
public Exam update(Exam exam) {
    examRepository.save(exam);
    return exam;
}

```

- Delete exam information method

```

@Override
public boolean delete(Exam exam) {
    examRepository.delete(exam);
    return true;
}

```

- **Fetch all exam information method**

```
@Override
public List<Exam> getAll() {
    return examRepository.findAll();
}
```

- **Search exam information method**

```
@Override
public Optional<Exam> find(Long id) {
    return examRepository.findById(id);
}
}
```

#### d. Exam Controller

- **Imported examDao Class to access methods from it**

```
@Controller
public class ExamController {

    @Autowired
    ExamDao examDao;
}
```

- **Exam details representation method**

```
@RequestMapping(value = {"/exam"}, method = RequestMethod.GET)
public ModelAndView index() {
    ModelAndView modelAndView = new ModelAndView();
    Exam newExam = new Exam();
    modelAndView.addObject("newExam", newExam);
    modelAndView.addObject("exams", examDao.getAll());
    modelAndView.setViewName("admin/adminExam");
    return modelAndView;
}
```

- **Save Exam details method**

```
@RequestMapping(value = {"/exam/save"}, method = RequestMethod.POST)
public String saveExam(Exam exam) {
    examDao.save(exam);
    return "redirect:/exam";
}
```



- **Editing exam information method**

```
@RequestMapping(value = {"/exam/find/{exam_id}"}, method = RequestMethod.GET)
public ModelAndView findForEditExam(@PathVariable(required = true, name =
"exam_id") Long exam_id) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<Exam> exam = examDao.find(exam_id);
    modelAndView.addObject("newExam", exam.get());
    modelAndView.addObject("exams", examDao.getAll());
    modelAndView.setViewName("admin/adminExam");
    return modelAndView;
}
```

- **Deleting exam information method**

```
@RequestMapping(value = "/exam/delete/{exam_id}", method = RequestMethod.GET)
public String deleteExam(@PathVariable(required = true, name = "exam_id") Long
exam_id) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<Exam> exam = examDao.find(exam_id);
    examDao.delete(exam.get());
    modelAndView.addObject("message", " Data Has Been Deleted...");
    return "redirect:/exam";
}
}
```

## e. Comments

- **Comment Class is for taking comments upon files & Exam**

```
@Entity
@Table(name = "comments")
public class Comments {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long comment_id;

    @Column(nullable = false)
    private String user_email;

    @NotEmpty
    private String comment_text;

    @CreationTimestamp
    private LocalDateTime date_time;

    @UpdateTimestamp
    private LocalDateTime updated_date_time;

    @ManyToOne
    private File file;

    @OneToMany(mappedBy = "comments")
    private List<ChildComments> childComments = new ArrayList<>();
}
```

- Taking comments as input

```
public Comments() {  
}  
  
public Comments(String user_email, String comment_text, LocalDateTime date_time,  
LocalDateTime updated_date_time, File file) {  
    this.user_email = user_email;  
    this.comment_text = comment_text;  
    this.date_time = date_time;  
    this.updated_date_time = updated_date_time;  
    this.file = file;  
}  
  
public Long getComment_id() {  
    return comment_id;  
}  
  
public void setComment_id(Long comment_id) {  
    this.comment_id = comment_id;  
}  
  
public String getComment_text() {  
    return comment_text;  
}  
  
public void setComment_text(String comment_text) {  
    this.comment_text = comment_text;  
}  
  
public LocalDateTime getDate_time() {  
    return date_time;  
}  
  
public void setDate_time(LocalDateTime date_time) {  
    this.date_time = date_time;  
}  
  
public LocalDateTime getUpdated_date_time() {  
    return updated_date_time;  
}  
  
public File getFile() {  
    return file;  
}  
  
public void setFile(File file) {  
    this.file = file;  
}  
  
public String getUser_email() {
```

- **Showing Comments method**

```
@Override
public String toString() {
    return "Comments{" +
        "comment_id=" + comment_id +
        ", user_email='" + user_email + '\'' +
        ", comment_text='" + comment_text + '\'' +
        ", date_time=" + date_time +
        ", updated_date_time=" + updated_date_time +
        ", file=" + file +
        ", childComments=" + childComments +
        '}';
}
}
```

f. **Child Comments**

- **Child Comment Class is for taking Child comments or comments of comments upon files & Exam**
- **Taking Child comments as input**

```
package ac.daffodil.model;
import javax.persistence.*;
@Entity
public class ChildComments {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long ccomments_id;
    private String sub_comments;
    private String user_name
    @ManyToOne
    private Comments comments;
    public ChildComments() {
    }
}
```

```

public ChildComments(String sub_comments) {
    this.sub_comments = sub_comments;
}
public ChildComments(String sub_comments, String user_name) {
    this.sub_comments = sub_comments;
    this.user_name = user_name;
}
public Long getCcomments_id() {
    return ccomments_id;
}
public void setCcomments_id(Long ccomments_id) {
    this.ccomments_id = ccomments_id;
}

public String getSub_comments() {
    return sub_comments;
}
public void setSub_comments(String sub_comments) {
    this.sub_comments = sub_comments;
}
public Comments getComments() {
    return comments;
}
public void setComments(Comments comments) {
    this.comments = comments;
}
public String getUser_name() {
    return user_name;
}
public void setUser_name(String user_name) {
    this.user_name = user_name;
}
}

```

- **Showing child comments method**

```
@Override
public String toString() {
    return "ChildComments{" +
        "ccomments_id=" + ccomments_id +
        ", sub_comments=" + sub_comments + '\n' +
        ", user_name=" + user_name + '\n' +
        ", comments=" + comments +
        '}';
}
}
```

## g. Comment Repository

- **SpringBoot JPA repository**

```
package ac.daffodil.repository;
import ac.daffodil.model.Comments;
import org.springframework.data.jpa.repository.JpaRepository;
public interface CommentRepository extends JpaRepository<Comments, Long> {
}
```

## h. Child Comments Repository

- **SpringBoot JPA repository**

```
package ac.daffodil.repository;
import ac.daffodil.model.ChildComments;
import org.springframework.data.jpa.repository.JpaRepository;
public interface ChildCommentRepository extends JpaRepository<ChildComments, Long> {
}
```

- **Comments Dao**

```
@Repository
@Transactional
public class CommentDao implements GenericInterface<Comments> {
    @Autowired
    CommentRepository commentRepository;
}
```

- **Saving Comments into database method**

```
@Override
public Comments save(Comments comments) {
    commentRepository.save(comments);
    return comments;
}
```

- **updating Comments into database method**

```
@Override
public Comments update(Comments comments) {
    commentRepository.save(comments);
    return comments;
}
```

- **Deleting Comments from database method**

```
@Override
public boolean delete(Comments comments) {
    commentRepository.delete(comments);
    return true;
}
```

- **Showing Comments from database method**

```
@Override
public List<Comments> getAll() {
    return commentRepository.findAll();
}
```

- **Finding Comments from database method**

```
@Override
public Optional<Comments> find(Long id) {
    return commentRepository.findById(id);
}
}
```

## i. Child Comments Dao

```
@Repository
public class ChildCommentDao implements GenericInterface<ChildComments> {

    @Autowired
    ChildCommentRepository childCommentRepository;
```

### ▪ Saving Child Comments into database method

```
@Override
public ChildComments save(ChildComments childComments) {
    childCommentRepository.save(childComments);
    return childComments;
}
```

### ▪ Updating Child Comments into database method

```
@Override
public ChildComments update(ChildComments childComments) {
    childCommentRepository.save(childComments);
    return childComments;
}
```

### ▪ Deleting Child Comments from database method

```
@Override
public boolean delete(ChildComments childComments) {
    childCommentRepository.delete(childComments);
    return true;
}
```

### ▪ Showing Child Comments from database method

```
@Override
public List<ChildComments> getAll() {
    return childCommentRepository.findAll();
}
```



- **Finding Child Comments from database method**

```
@Override
public Optional<ChildComments> find(Long id) {
    return childCommentRepository.findById(id);
}
}
```

## j. **Comment Controller**

```
@Controller
public class CommentController {

    Logger logger = LoggerFactory.getLogger(getClass());

    @Autowired
    CommentDao commentDao;

    @Autowired
    FileDao fileDao;

    @Autowired
    ChildCommentDao childCommentDao;

    Comments comments = new Comments();
    List<Comments> comments1 = new LinkedList<>();

    ChildComments childComments = new ChildComments();
}
```

- **Comment Representation method**

```
@RequestMapping(value = {"/comment"}, method = RequestMethod.GET)
public ModelAndView commentPage() {
    ModelAndView modelAndView = new ModelAndView();
    Comments newComment = new Comments();

    comments1 = new LinkedList<>();
    for (Comments cmt : commentDao.getAll()) {
        if (cmt.getFile().getFile_id() == comments.getFile().getFile_id()) {
            comments1.add(cmt);
        }
    }

    modelAndView.addObject("newComment", comments);
    modelAndView.addObject("commentList", comments1);
    modelAndView.setViewName("user/userDashComment");
    return modelAndView;
}
```

- Find & Get Comment & File ID for resolving problems

```
@RequestMapping(value = {"/comment/findForFile/{file_id}"}, method =
RequestMethod.GET)
public ModelAndView findForSetFileId(@PathVariable(required = true, name =
"file_id") Long file_id) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<File> file = fileDao.find(file_id);
    comments.setFile(file.get());

    Object principal =
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    if (principal instanceof User) {
        String email = ((User) principal).getEmail();
        comments.setUser_email(email);
    }

    comments1 = new LinkedList<>();
    for (Comments cmt : commentDao.getAll()) {
        if (cmt.getFile().getFile_id() == file_id) {
            comments1.add(cmt);
        }
    }

    modelAndView.addObject("commentList", comments1);
    modelAndView.addObject("newComment", comments);
    modelAndView.setViewName("user/userDashComment");
    return modelAndView;
}
```

- **Saving Comments into database method**

```
@RequestMapping(value = {"/comment/saveComment"}, method = RequestMethod.POST)
public String saveComment(Comments comments, RedirectAttributes
redirectAttributes) {
    commentDao.save(comments);
    redirectAttributes.addFlashAttribute("message", "You Comment is= " +
comments.getComment_text());
    redirectAttributes.addFlashAttribute("alertClass", "alert-success");
    return "redirect:/comment";
}
```

- **Child Comment through Id**

```
@RequestMapping(value = {"/comment/find/{comment_id}"}, method =
RequestMethod.GET)
public ModelAndView findForEditComment(@PathVariable(required = true, name =
"comment_id") Long comment_id) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<Comments> comments = commentDao.find(comment_id);
    modelAndView.addObject("newComment", comments.get());
    modelAndView.addObject("commentList", comments1);
    modelAndView.setViewName("user/userDashComment");
    return modelAndView;
}
```

- **Deleting Child Comments from database method**

```
@RequestMapping(value = "/comment/delete/{comment_id}", method =
RequestMethod.GET)
public String deleteExam(@PathVariable(required = true, name =
"comment_id") Long comment_id) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<Comments> comments = commentDao.find(comment_id);
    commentDao.delete(comments.get());
    modelAndView.addObject("message", " Data Has Been Deleted...");
    return "redirect:/comment";
}
```

- **Child Comment through sequential id by finding comment id**

```
@RequestMapping(value = {"/findForComment/{comment_id}"}, method =
RequestMethod.GET)
public ModelAndView findForSetCommentId(@PathVariable(required = true, name =
"comment_id") Long comment_id) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<Comments> comment = commentDao.find(comment_id);
    ChildComments childComments = new ChildComments();
    childComments.setComments(comment.get());
    modelAndView.addObject("newComment", childComments);
    System.out.println(childComments.getComments().getComment_id());
    modelAndView.setViewName("user/childComment");
    return modelAndView;
}
```

- **Saving Child Comment**

```
@RequestMapping(value = {"/comment/saveChildComment"}, method =
RequestMethod.POST)
public String saveChildComment(ChildComments childComments,
RedirectAttributes redirectAttributes) {
    Object principal =
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    if (principal instanceof User) {
        String name = ((User) principal).getFirstName();
        childComments.setUser_name(name);
        childCommentDao.save(childComments);
        redirectAttributes.addFlashAttribute("message", "Your Comment is="
+ comments.getComment_text());
        redirectAttributes.addFlashAttribute("alertClass", "alert-
success");
        return "redirect:/comment";
    }
    return "redirect:/comment";
}
}
```

## 9.1.5 Forum Code sample

### a. Post Controller

```
@Controller
@RequestMapping("/forum")
public class PostsController {

    @Autowired
    PostsDao postsDao;
```

- To view all the threads / post on the forum home page

```
@RequestMapping (value = {"/posts"}, method = RequestMethod.GET)
public ModelAndView index(HttpServletRequest request) {
    ModelAndView modelAndView = new ModelAndView();
    Posts Post = new Posts();
    modelAndView.addObject("Post", Post);
    modelAndView.addObject("posts", postsDao.getAll());
    modelAndView.setViewName("Body/forum");
    return modelAndView;
}
```

- Submit a post or ask a question and press submit

```
@RequestMapping (value = {"/posts/submitPost"}, method = RequestMethod.GET)
public ModelAndView submitPost() {
    ModelAndView modelAndView = new ModelAndView();
    Posts newPost = new Posts();
    modelAndView.addObject("newPost", newPost);
    modelAndView.addObject("posts", postsDao.getAll());
    modelAndView.setViewName("Body/askQuestion");
    return modelAndView;
}
```

- Saving the post into database

```
@RequestMapping(value = {"/posts/savePost"}, method = RequestMethod.POST)
public String savePost(Posts posts){
    ModelAndView modelAndView = new ModelAndView();
    postsDao.save(posts);
    modelAndView.addObject("message","Data Has been saved");
    return "redirect:/forum/posts";
}
```

- **Searching all threads by user**

```
@RequestMapping(value = {"/posts/findAll"}, method = RequestMethod.GET)
public ModelAndView findPosts(){
    ModelAndView modelAndView = new ModelAndView();

    modelAndView.addObject("posts",postsDao.getAll());
    modelAndView.setViewName("Body/postDetails");
    return modelAndView;
}

Optional<Posts> posts = Optional.of(new Posts());
```

- **Searching a single thread**

```
@RequestMapping(value = {"/posts/find/{PostId}"}, method = RequestMethod.GET)
public ModelAndView findForShowingPost(@PathVariable(required = true, name = "PostId")Long PostId){
    ModelAndView modelAndView = new ModelAndView();
    posts = postsDao.find(PostId);
    modelAndView.addObject("posts",posts.get());
    List<Posts> allPosts = new LinkedList<>();
    for (Posts askPost : postsDao.getAll()) {
        if (askPost.getPostId() != posts.get().getPostId()){
            allPosts.add(askPost);
        }
    }
    modelAndView.addObject("newPost", allPosts);
    modelAndView.setViewName("Body/postDetails");
    return modelAndView;
}
```

- **Delete a thread**

```
@RequestMapping(value = {"/posts/delete/{PostId}"}, method = RequestMethod.GET)
public String findForDeletingPost(@PathVariable(required = true,name = "PostId")Long PostId){
    ModelAndView modelAndView = new ModelAndView();
    Optional<Posts> posts = postsDao.find(PostId);
    postsDao.delete(posts.get());
    modelAndView.addObject("message", "Post has been deleted");
    return "redirect:/forum/posts";
}
}
```

## b. Feedback of threads controller

```
@Controller
@RequestMapping("/PostFeedback")
public class PostFeedbackController {
    @Autowired
    PostFeedbackDao postFeedbackDao;
```

### ▪ Viewing feedbacks

```
@RequestMapping(value = { "/postFeedback" }, method = RequestMethod.GET)
public ModelAndView index() {
    ModelAndView modelAndView = new ModelAndView();
    PostFeedback postFeedback = new PostFeedback();
    modelAndView.addObject("newPostFeedback", postFeedback);
    modelAndView.addObject("postFeedback", postFeedbackDao.getAll());
    modelAndView.setViewName("admin/adminPostFeedback");
    modelAndView.setViewName("user/userPostFeedback");
    return modelAndView;
}
```

### ▪ Writing feedback

```
@RequestMapping(value = { "/postFeedback/save" }, method = RequestMethod.POST)
public String savePostFeedback(PostFeedback postFeedback) {
    postFeedbackDao.save(postFeedback);
    return "redirect:/postFeedback";
}
```

### ▪ Find feedback by id

```
@RequestMapping(value={"/postFeedback/find/{PostFeedbackId}"}, method = RequestMethod.GET)
public ModelAndView findForEditPostFeedback(@PathVariable(required = true, name =
"PostFeedbackId") Long PostFeedbackId) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<PostFeedback> postFeedback= postFeedbackDao.find(PostFeedbackId);
    modelAndView.addObject("newPostFeedback", postFeedback.get());
    modelAndView.addObject("postFeedback", postFeedbackDao.getAll());
    modelAndView.setViewName("admin/adminPostFeedback");
    modelAndView.setViewName("user/userPostFeedback");
    return modelAndView;
}
```



- **Delete feedback by id**

```
@RequestMapping(value="/postFeedback/delete/{PostFeedbackId}", method = RequestMethod.GET)
public String deletePostFeedback(@PathVariable(required = true, name = "PostFeedbackId") Long
PostFeedbackId) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<PostFeedback> postFeedback= postFeedbackDao.find(PostFeedbackId);
    postFeedbackDao.delete(postFeedback.get());
    modelAndView.addObject("message", " Data Has Been Deleted...");
    return "redirect:/postFeedback";
}
}
```

### c. Thread comments Controller

```
@Controller
@RequestMapping("/postComment")
public class PostsCommentsController {

    @Autowired
    PostsCommentsDao postsCommentsDao;
```

- **Viewing all comments by post id**

```
@RequestMapping(value = {"/postComments"},method = RequestMethod.GET)
public ModelAndView index(){
    ModelAndView modelAndView = new ModelAndView();
    PostsComments newPostsComment = new PostsComments();
    modelAndView.addObject("newPostsComment",newPostsComment);
    modelAndView.addObject("postsComment",postsCommentsDao.getAll());
    modelAndView.setViewName("admin/adminPendingFlags");
    return modelAndView;
}
```

- **Saving a comment of thread / post**

```
@RequestMapping(value = {"/postComments/save"},method = RequestMethod.POST)
public String savePostComments(PostsComments postsComments){
    postsCommentsDao.save(postsComments);
    return "redirect:/postComments";
}
```

- **Searching a comment or editing**

```
@RequestMapping(value = {"/postComments/find/{postComments_id}"},method = RequestMethod.GET)
public ModelAndView findForEditingPostComments(@PathVariable(required = true,name =
"postComments_id")Long postComments_id){
    ModelAndView modelAndView =new ModelAndView();
    Optional<PostsComments> postsComments = postsCommentsDao.find(postComments_id);
    modelAndView.addObject("newPostComments", postsComments.get());
    modelAndView.addObject("postComments", postsCommentsDao.getAll());
    modelAndView.setViewName("admin/adminPendingFlags");
    return modelAndView;
}
```

- **Deleting a comment**

```
@RequestMapping(value = {"/postComments/delete/{postComments_id}"},method = RequestMethod.GET)
public String findForDeletingPendingFlags(@PathVariable(required = true,name =
"postComments_id")Long postComments_id){
    ModelAndView modelAndView = new ModelAndView();
    Optional<PostsComments> postsComments = postsCommentsDao.find(postComments_id);
    postsCommentsDao.delete(postsComments.get());
    modelAndView.addObject("message","data has been deleted");
    return "redirect:/postComments";
}
}
```

**d. Post / thread DAO class**

- **Imported repository of post**

```
@Repository
@Transactional
public class PostsDao implements GenericInterface<Posts> {
    @Autowired
    private PostsRepository postsRepository;
```

- **All necessary methods to handle posts in controller to view them accurately**

```
@Override
public Posts save(Posts posts) {
    postsRepository.save(posts);
    return posts;
}

@Override
public Posts update(Posts posts) {
    postsRepository.save(posts);
    return posts;
}

@Override
public boolean delete(Posts posts) {
    postsRepository.delete(posts);
    return true;
}

@Override
public List<Posts> getAll() {
    return postsRepository.findAll();
}

@Override
public Optional<Posts> find(Long id) {
    return postsRepository.findById(id);
}
}
```

#### e. Feedback of posts

- **Post feedback & repository has imported**

```
@Repository
@Transactional
public class PostFeedbackDao implements GenericInterface<PostFeedback> {
    @Autowired
    private PostFeedbackRepository postFeedbackRepository;
}
```

- **All necessary methods**

```
@Override
public PostFeedback save(PostFeedback postFeedback) {
    postFeedbackRepository.save(postFeedback);

    return postFeedback;
}

@Override
public PostFeedback update(PostFeedback postFeedback) {
    postFeedbackRepository.save(postFeedback);
    return postFeedback;
}

@Override
public boolean delete(PostFeedback postFeedback) {
    postFeedbackRepository.delete(postFeedback);
    return true;
}

@Override
public List<PostFeedback> getAll() {
    return postFeedbackRepository.findAll();
}

@Override
public Optional<PostFeedback> find(Long id) {
    return postFeedbackRepository.findById(id);
}
}
```

**f. post comment Dao**

- **Post comment model and repository classes are imported**

```
@Repository
@Transactional
public class PostsCommentsDao implements GenericInterface<PostsComments>{

    @Qualifier("postsCommentsRepository")
    @Autowired
    PostsCommentsRepository postsCommentsRepository;
}
```

- All necessary methods are written

```
@Override
public PostsComments save(PostsComments postsComments) {
    postsCommentsRepository.save(postsComments);
    return postsComments;
}
```

```
@Override
public PostsComments update(PostsComments postsComments) {
    postsCommentsRepository.save(postsComments);
    return postsComments;
}
```

```
@Override
public boolean delete(PostsComments postsComments) {
    postsCommentsRepository.delete(postsComments);
    return true;
}
@Override
public List<PostsComments> getAll() {
    return postsCommentsRepository.findAll();
}
@Override
public Optional<PostsComments> find(Long id) {
    return postsCommentsRepository.findById(id);
}
}
```

## g. Post Class

```
package ac.daffodil.model;

import javax.persistence.*;
import java.util.*;

@Entity
@Table(name="posts")
public class Posts {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name= "PostId")
    private long PostId;

    @Column(name = "PostTypeId")
    private long PostTypeId;

    @Column(name = "AcceptedAnsId")
    private long AcceptedAnsId;

    @Column(name = "ParentId")
    private long ParentId;

    @Column(name = "CreationDate")
    private Date CreationDate;

    @Column(name = "DeletionDate")
    private Date DeletionDate;

    @Column(name = "Score")
    private Float Score;

    @Column(name = "ViewCount")
    private int ViewCount;

    @Column(name = "Body")
    private String Body;
```

```

@Column(name = "OwnerUserId")
private long OwnerUserId;

@Column(name = "OwnerDisplayName")
private String OwnerDisplayName;

@Column(name = "LastEditorUserId")
private long LastEditorUserId;

@Column(name = "LastEditorDisplayName")
private String LastEditorDisplayName;

@Column(name = "LastEditDate")
private Date LastEditDate;

@Column(name = "LastActivityDate")
private Date LastActivityDate;

@Column(name = "Title")
private String Title;

@Column(name = "Tags")
private String Tags;

@Column(name = "AnsCount")
private int AnsCount;

@Column(name = "CommentCount")
private int CommentCount;

@Column(name = "FavoriteCount")
private int FavoriteCount;

@Column(name = "ClosedDate")
private Date ClosedDate;

@Column(name = "CommunityOwnedDate")
private Date CommunityOwnedDate;

public Posts() {
}
public Posts(long postTypeId, long acceptedAnsId, long parentId, Date creationDate, Date deletionDate,
Float score, int viewCount, String body, long ownerUserId, String ownerDisplayName, long lastEditorUserId,
String lastEditorDisplayName, Date lastEditDate, Date lastActivityDate, String title, String tags, int ansCount,
int commentCount, int favoriteCount, Date closedDate, Date communityOwnedDate) {
    PostTypeId = postTypeId;
    AcceptedAnsId = acceptedAnsId;

    ParentId = parentId;
    CreationDate = creationDate;
    DeletionDate = deletionDate;

```

```

Score = score;
ViewCount = viewCount;
Body = body;
OwnerUserId = ownerUserId;
OwnerDisplayName = ownerDisplayName;
LastEditorUserId = lastEditorUserId;
LastEditorDisplayName = lastEditorDisplayName;
LastEditDate = lastEditDate;
LastActivityDate = lastActivityDate;
Title = title;
Tags = tags;
AnsCount = ansCount;
CommentCount = commentCount;
FavoriteCount = favoriteCount;
ClosedDate = closedDate;
CommunityOwnedDate = communityOwnedDate;
}

public long getPostId() {
    return PostId;
}

public void setPostId(long postId) {
    PostId = postId;
}

public long getPostTypeId() {
    return PostTypeId;
}

public void setPostTypeId(long postTypeId) {
    PostTypeId = postTypeId;
}

public long getAcceptedAnsId() {
    return AcceptedAnsId;
}

public void setAcceptedAnsId(long acceptedAnsId) {
    AcceptedAnsId = acceptedAnsId;
}

public long getParentId() {
    return ParentId;
}

public void setParentId(long parentId) {
    ParentId = parentId;
}

public Date getCreationDate() {
    return CreationDate;
}
}

```



```

public void setCreationDate(Date creationDate) {
    CreationDate = creationDate;
}
public Date getDeletionDate() {
    return DeletionDate;
}

public void setDeletionDate(Date deletionDate) {
    DeletionDate = deletionDate;
}

public Float getScore() {
    return Score;
}

public void setScore(Float score) {
    Score = score;
}

public int getViewCount() {
    return ViewCount;
}

public void setViewCount(int viewCount) {
    ViewCount = viewCount;
}

public String getBody() {
    return Body;
}

public void setBody(String body) {
    Body = body;
}

public long getOwnerUserId() {
    return OwnerUserId;
}

public void setOwnerUserId(long ownerUserId) {
    OwnerUserId = ownerUserId;
}

public String getOwnerDisplayName() {
    return OwnerDisplayName;
}

public void setOwnerDisplayName(String ownerDisplayName) {
    OwnerDisplayName = ownerDisplayName;
}

public long getLastEditorUserId() {
    return LastEditorUserId;
}
}

```

```

public void setLastEditorUserId(long lastEditorUserId) {
    LastEditorUserId = lastEditorUserId;
}

public String getLastEditorDisplayName() {
    return LastEditorDisplayName;
}

public void setLastEditorDisplayName(String lastEditorDisplayName) {
    LastEditorDisplayName = lastEditorDisplayName;
}

public Date getLastEditDate() {
    return LastEditDate;
}

public void setLastEditDate(Date lastEditDate) {
    LastEditDate = lastEditDate;
}

public Date getLastActivityDate() {
    return LastActivityDate;
}

public void setLastActivityDate(Date lastActivityDate) {
    LastActivityDate = lastActivityDate;
}

public String getTitle() {
    return Title;
}

public void setTitle(String title) {
    Title = title;
}

public String getTags() {
    return Tags;
}

public void setTags(String tags) {
    Tags = tags;
}

public int getAnsCount() {
    return AnsCount;
}

public void setAnsCount(int ansCount) {
    AnsCount = ansCount;
}

public int getCommentCount() {
    return CommentCount;
}

```

```

public void setCommentCount(int commentCount) {
    CommentCount = commentCount;
}
public int getFavoriteCount() {
    return FavoriteCount;
}
public void setFavoriteCount(int favoriteCount) {
    FavoriteCount = favoriteCount;
}
public Date getClosedDate() {
    return ClosedDate;
}
public void setClosedDate(Date closedDate) {
    ClosedDate = closedDate;
}
public Date getCommunityOwnedDate() {
    return CommunityOwnedDate;
}
public void setCommunityOwnedDate(Date communityOwnedDate) {
    CommunityOwnedDate = communityOwnedDate;
}
}

```

- **To view all fields**

```

@Override
public String toString() {
    return "Posts{" +
        "PostId=" + PostId +
        ", PostTypeId=" + PostTypeId +
        ", AcceptedAnsId=" + AcceptedAnsId +
        ", ParentId=" + ParentId +
        ", CreationDate=" + CreationDate +
        ", DeletionDate=" + DeletionDate +
        ", Score=" + Score +
        ", ViewCount=" + ViewCount +
        ", Body=" + Body + "\" +
        ", OwnerUserId=" + OwnerUserId +
        ", OwnerDisplayName=" + OwnerDisplayName + "\" +
        ", LastEditorUserId=" + LastEditorUserId +
        ", LastEditorDispalyName=" + LastEditorDisplayName + "\" +
        ", LastEditDate=" + LastEditDate +
        ", LastActivityDate=" + LastActivityDate +
        ", Title=" + Title + "\" +
        ", Tags=" + Tags + "\" +
        ", AnsCount=" + AnsCount +
        ", CommentCount=" + CommentCount +
        ", FavoriteCount=" + FavoriteCount +
        ", ClosedDate=" + ClosedDate +
        ", CommunityOwnedDate=" + CommunityOwnedDate +
        '}';
}
}

```

## h. Post Comment Model Class

```
package ac.daffodil.model;

import org.springframework.data.annotation.CreatedDate;

import javax.persistence.*;
import java.time.LocalDate;

@Entity
@Table(name = "postsComments")
public class PostsComments {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long postComments_id;

    @Column(name = "Post_Id")
    private long post_id;

    @Column(name = "Score")
    private long score;

    @Column(name = "Text")
    private String text;

    @CreatedDate
    private LocalDate creationDate;

    @Column(name = "UserDisplayName")
    private String userDisplayName;

    @Column(name = "user_id")
    private long user_id;

    public PostsComments() {
    }

    public long getPostComments_id() {
        return postComments_id;
    }

    public void setPostComments_id(long postComments_id) {
        this.postComments_id = postComments_id;
    }

    public long getPost_id() {
        return post_id;
    }

    public void setPost_id(long post_id) {
        this.post_id = post_id;
    }

    public long getScore() {
        return score;
    }
}
```

```

public void setScore(long score) {
    this.score = score;
}

public String getText() {
    return text;
}

public void setText(String text) {
    this.text = text;
}

public LocalDate getCreationDate() {
    return creationDate;
}

public void setCreationDate(LocalDate creationDate) {
    this.creationDate = creationDate;
}

public String getUserDisplayName() {
    return userDisplayName;
}

public void setUserDisplayName(String userDisplayName) {
    this.userDisplayName = userDisplayName;
}

public long getUser_id() {
    return user_id;
}

public void setUser_id(long user_id) {
    this.user_id = user_id;
}

```

- **To view all required fields which have taken**

```

@Override
public String toString() {
    return "PostsComments{" +
        "postComments_id=" + postComments_id +
        ", post_id=" + post_id +
        ", score=" + score +
        ", text=" + text + '\n' +
        ", creationDate=" + creationDate +
        ", userDisplayName=" + userDisplayName + '\n' +
        ", user_id=" + user_id +
        '}';
}
}

```

## i. Post feedback model class

```
package ac.daffodil.model;

import javax.persistence.*;
import java.util.BitSet;
import java.util.Date;

@Entity
@Table(name = "PostFeedback")
```

### ▪ Required input fields

```
public class PostFeedback {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "PostFeedbackId")
    private Long PostFeedbackId;

    @Column(name = "PostId")
    private Long PostId;

    @Column(name = "IsAnonymous")
    private BitSet IsAnonymous;

    @Column(name = "CreationDate")
    private Date CreationDate;
```

- **Building constructors of fields**

```
public PostFeedback() {  
}  
  
public PostFeedback(Long postFeedbackId, Long postId, BitSet isAnonymous, Date creationDate) {  
    PostFeedbackId = postFeedbackId;  
    PostId = postId;  
    IsAnonymous = isAnonymous;  
    CreationDate = creationDate;  
}  
  
public Long getId() {  
    return PostFeedbackId;  
}  
  
public void setId(Long id) {  
    this.PostFeedbackId = id;  
}  
  
public Long getPostId() {  
    return PostId;  
}  
  
public void setPostId(Long postId) {  
    PostId = postId;  
}  
  
public BitSet getIsAnonymous() {  
    return IsAnonymous;  
}
```

```
public void setIsAnonymous(BitSet isAnonymous) {  
    IsAnonymous = isAnonymous;  
}  
  
public Date getCreationDate() {  
    return CreationDate;  
}  
  
public void setCreationDate(Date creationDate) {  
    CreationDate = creationDate;  
}
```

- **To view all the fields specified**

```
@Override  
public String toString() {  
    return "PostFeedback{" +  
        "id=" + PostFeedbackId +  
        ", PostId=" + PostId +  
        ", IsAnonymous=" + IsAnonymous +  
        ", CreationDate=" + CreationDate +  
        '}';  
}  
}
```

## j. Post feedback repository

```
package ac.daffodil.repository;

import ac.daffodil.model.PostFeedback;
import org.springframework.data.jpa.repository.JpaRepository;

public interface PostFeedbackRepository extends JpaRepository<PostFeedback, Long> {
}
```

## k. Post repository

```
package ac.daffodil.repository;

import ac.daffodil.model.Posts;
import org.springframework.data.jpa.repository.JpaRepository;

public interface PostsRepository extends JpaRepository<Posts, Long> {
}
```

## l. Post comment

```
package ac.daffodil.repository;

import ac.daffodil.model.PostsComments;
import org.springframework.data.jpa.repository.JpaRepository;

public interface PostsCommentsRepository extends JpaRepository<PostsComments, Long> {
}
```

## 9.1.6 Voting code samples

### a. Votes Controller

- Request mapper & Controller assigned

```
@Controller
@RequestMapping("/vote")
public class votesController {
    @Autowired
    VotesDao votesDao;
}
```



- **All votes will be viewed here**

```
@RequestMapping(value = { "/votes" }, method = RequestMethod.GET)
public ModelAndView index() {
    ModelAndView modelAndView = new ModelAndView();
    Votes votes = new Votes();
    modelAndView.addObject("newVotes", votes);
    modelAndView.addObject("votes", votesDao.getAll());
    modelAndView.setViewName("admin/adminVotes");
    return modelAndView;
}
```

- **Saving a vote of a user**

```
@RequestMapping(value = { "/votes/save" }, method = RequestMethod.POST)
public String saveVotes(Votes votes) {
    votesDao.save(votes);
    return "redirect:/votes";
}
```

- **Finding a vote of a user (will not be required in later development)**

```
@RequestMapping(value="{ "/votes/find/{VoteId}", method = RequestMethod.GET)
public ModelAndView findForEditVotes(@PathVariable(required = true, name = "VoteId") Long VoteId) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<Votes> votes= votesDao.find(VoteId);
    modelAndView.addObject("newVotes", votes.get());
    modelAndView.addObject("votes", votesDao.getAll());
    modelAndView.setViewName("admin/adminVotes");
    return modelAndView;
}
```

- **Removing vote for the post**

```
@RequestMapping(value="/votes/delete/{VoteId}", method = RequestMethod.GET)
public String deleteVotes(@PathVariable(required = true, name = "VoteId") Long VoteId) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<Votes> votes= votesDao.find(VoteId);
    votesDao.delete(votes.get());
    modelAndView.addObject("message", "Data Has Been Deleted...");
    return "redirect:/votes";
}
}
```

## b. Vote Types Controller

- Request mapper & Controller assigned

```
@Controller
@RequestMapping("/voteType")
public class voteTypesController {
    @Autowired
    VoteTypesDao voteTypesDao;
```

- Type of the votes are viewed

```
@RequestMapping(value = { "/voteTypes" }, method = RequestMethod.GET)
public ModelAndView index() {
    ModelAndView modelAndView = new ModelAndView();
    VoteTypes voteTypes = new VoteTypes();
    modelAndView.addObject("newVoteTypes", voteTypes);
    modelAndView.addObject("voteTypes", voteTypesDao.getAll());
    modelAndView.setViewName("admin/adminVoteTypes");
    return modelAndView;
}
```

- A new vote type is saved

```
@RequestMapping(value = { "/voteTypes/save" }, method = RequestMethod.POST)
public String saveExam(VoteTypes voteTypes) {
    voteTypesDao.save(voteTypes);
    return "redirect:/voteTypes";
}
```

- Finding a saved vote type

```
@RequestMapping(value={"/voteTypes/find/{VoteTypeId}"}, method = RequestMethod.GET)
public ModelAndView findForEditVoteTypes(@PathVariable(required = true, name = "VoteTypeId") Long
VoteTypeId) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<VoteTypes> voteTypes= voteTypesDao.find(VoteTypeId);
    modelAndView.addObject("newVoteTypes", voteTypes.get());
    modelAndView.addObject("voteTypes", voteTypesDao.getAll());
    modelAndView.setViewName("admin/adminVoteTypes");
    return modelAndView;
}
```

- **Deleting a vote type**

```
@RequestMapping(value="/voteTypes/delete/{VoteTypeId}", method = RequestMethod.GET)
public String deleteVoteTypes(@PathVariable(required = true, name = "VoteTypeId") Long VoteTypeId) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<VoteTypes> voteTypes= voteTypesDao.find(VoteTypeId);
    voteTypesDao.delete(voteTypes.get());
    modelAndView.addObject("message", " Data Has Been Deleted...");
    return "redirect:/voteTypes";
}
}
```

- c. **Suggested Edit Votes**

- **Controller & Request mapper assigned**

```
@Controller
@RequestMapping("/suggestedEditVote")
public class SuggestedEditVotesController {
    @Autowired
    SuggestedEditVotesDao suggestedEditVotesDao;
}
```

- **Viewing suggestion for votes**

```
@RequestMapping(value = { "/suggestedEditVotes" }, method = RequestMethod.GET)
public ModelAndView index() {
    ModelAndView modelAndView = new ModelAndView();
    SuggestedEditVotes suggestedEditVotes = new SuggestedEditVotes();
    modelAndView.addObject("newSuggestedEditVotes", suggestedEditVotes);
    modelAndView.addObject("suggestedEditVotes", suggestedEditVotesDao.getAll());
    modelAndView.setViewName("admin/adminSuggestedEditVotes");
    modelAndView.setViewName("user/userSuggestedEditVotes");
    return modelAndView;
}
```

- **Saving a vote**

```
@RequestMapping(value = { "/suggestedEditVotes/save" }, method = RequestMethod.POST)
public String saveSuggestedEditVotes(SuggestedEditVotes suggestedEditVotes) {
    suggestedEditVotesDao.save(suggestedEditVotes);
    return "redirect:/suggestedEditVotes";
}
```

- **Editing a vote**

```
@RequestMapping(value={"/suggestedEditVotes/find/{SuggestedEditVotesID}"}, method =
RequestMethod.GET)
public ModelAndView findForEditSuggestedEditVotes(@PathVariable(required = true, name =
"SuggestedEditVotesID") Long SuggestedEditVotesID) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<SuggestedEditVotes> suggestedEditVotes=
suggestedEditVotesDao.find(SuggestedEditVotesID);
    modelAndView.addObject("newSuggestedEditVotes", suggestedEditVotes.get());
    modelAndView.addObject("suggestedEditVotes", suggestedEditVotesDao.getAll());
    modelAndView.setViewName("admin/adminSuggestedEditVotes");
    modelAndView.setViewName("user/userSuggestedEditVotes");
    return modelAndView;
}
```

- **Deleting a vote**

```
@RequestMapping(value={"/suggestedEditVotes/delete/{SuggestedEditVotesID}"}, method =
RequestMethod.GET)
public String deleteSuggestedEditVotes(@PathVariable(required = true, name = "SuggestedEditVotesID")
Long SuggestedEditVotesID) {
    ModelAndView modelAndView = new ModelAndView();
    Optional<SuggestedEditVotes> suggestedEditVotes=
suggestedEditVotesDao.find(SuggestedEditVotesID);
    suggestedEditVotesDao.delete(suggestedEditVotes.get());
    modelAndView.addObject("message", " Data Has Been Deleted...");
    return "redirect:/suggestedEditVotes";
}
}
```

#### d. Suggested edit votes Dao

- **Repository & model class imported**

```
@Repository
@Transactional
public class SuggestedEditVotesDao implements GenericInterface<SuggestedEditVotes> {
```

- All necessary methods from repository is used

```
@Autowired
private SuggestedEditVotesRepository suggestedEditVotesRepository;
@Override
public SuggestedEditVotes save(SuggestedEditVotes suggestedEditVotes) {
    suggestedEditVotesRepository.save(suggestedEditVotes);
    return suggestedEditVotes;
}

@Override
public SuggestedEditVotes update(SuggestedEditVotes suggestedEditVotes) {
    suggestedEditVotesRepository.save(suggestedEditVotes);
    return suggestedEditVotes;
}

@Override
public boolean delete(SuggestedEditVotes suggestedEditVotes) {
    suggestedEditVotesRepository.delete(suggestedEditVotes);
    return true;
}

@Override
public List<SuggestedEditVotes> getAll() {
    return suggestedEditVotesRepository.findAll();
}

@Override
public Optional<SuggestedEditVotes> find(Long id) {
    return suggestedEditVotesRepository.findById(id);
}
}
```

#### e. Votes Dao

- Repository & Model class imported

```
@Repository
@Transactional
public class VotesDao implements GenericInterface<Votes> {
```

- All necessary methods from repository is used

```
@Autowired
private VotesRepository votesRepository;

@Override
public Votes save(Votes votes) {
    votesRepository.save(votes);
    return votes;
}

@Override
public Votes update(Votes votes) {
    votesRepository.save(votes);
    return votes;
}

@Override
public boolean delete(Votes votes) {
    votesRepository.delete(votes);
    return true;
}

@Override
public List<Votes> getAll() {
    return votesRepository.findAll();
}

@Override
public Optional<Votes> find(Long id) {
    return votesRepository.findById(id);
}
}
```

#### f. Vote types Dao

- Repository & Model class imported

```
@Repository
@Transactional
public class VoteTypesDao implements GenericInterface<VoteTypes> {
```

- **All necessary methods from repository is used**

```
@Autowired
private VoteTypesRepository voteTypesRepository;
@Override
public VoteTypes save(VoteTypes voteTypes) {
    voteTypesRepository.save(voteTypes);
    return voteTypes;
}

@Override
public VoteTypes update(VoteTypes voteTypes) {
    voteTypesRepository.save(voteTypes);
    return voteTypes;
}

@Override
public boolean delete(VoteTypes voteTypes) {
    voteTypesRepository.delete(voteTypes);
    return true;
}

@Override
public List<VoteTypes> getAll() {
    return voteTypesRepository.findAll();
}

@Override
public Optional<VoteTypes> find(Long id) {
    return voteTypesRepository.findById(id);
}
}
```

#### **g. Suggested edit model class**

- **Entity created**

```
@Entity
@Table(name = "SuggestedEditVotes")
```

- **Necessary fields are taken**

```
public class SuggestedEditVotes {  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    @Column(name = "SuggestedEditVotesID")  
    private Long SuggestedEditVotesID;  
  
    @Column(name = "SuggestedEditId")  
    private Long SuggestedEditId;  
  
    @Column(name = "UserId")  
    private Long id;  
  
    @Column(name = "CreationDate")  
    private Date CreationDate;  
  
    @Column(name = "TargetUserId")  
    private Long TargetUserId;  
  
    @Column(name = "TargetRepChange")  
    private Long TargetRepChange;  
  
    public SuggestedEditVotes() {  
    }  
}
```



- **Constructors created**

```
public SuggestedEditVotes(Long suggestedEditVotesID, Long suggestedEditId, Long id, Date creationDate,
Long targetUserId, Long targetRepChange) {
    SuggestedEditVotesID = suggestedEditVotesID;
    SuggestedEditId = suggestedEditId;
    this.id = id;
    CreationDate = creationDate;
    TargetUserId = targetUserId;
    TargetRepChange = targetRepChange;
}

public Long getSuggestedEditVotesID() {
    return SuggestedEditVotesID;
}

public void setSuggestedEditVotesID(Long suggestedEditVotesID) {
    SuggestedEditVotesID = suggestedEditVotesID;
}

public Long getSuggestedEditId() {
    return SuggestedEditId;
}

public void setSuggestedEditId(Long suggestedEditId) {
    SuggestedEditId = suggestedEditId;
}

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public Date getCreationDate() {
    return CreationDate;
}

public void setCreationDate(Date creationDate) {
    CreationDate = creationDate;
}

public Long getTargetUserId() {
    return TargetUserId;
}

public void setTargetUserId(Long targetUserId) {
    TargetUserId = targetUserId;
}
```

```

public Long getTargetRepChange() {
    return TargetRepChange;
}

public void setTargetRepChange(Long targetRepChange) {
    TargetRepChange = targetRepChange;
}

```

- **Method for viewing all fields**

```

@Override
public String toString() {
    return "SuggestedEditVotes{" +
        "SuggestedEditVotesID=" + SuggestedEditVotesID +
        ", SuggestedEditId=" + SuggestedEditId +
        ", id=" + id +
        ", CreationDate=" + CreationDate +
        ", TargetUserId=" + TargetUserId +
        ", TargetRepChange=" + TargetRepChange +
        '}';
}
}

```

#### h. Votes Model class

- **Entity created**

```

@Entity
@Table(name = "Votes")
public class Votes {

```

- **Necessary fields are taken**

```
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
@Column(name = "VoteId")
private Long VoteId;

@Column(name = "PostId")
private Long PostId;

@Column(name = "VoteTypeId")
private Long VoteTypeId;

@Column(name = "UserId")
private Long id;

@Column(name = "CreationDate")
private Date CreationDate;

@Column(name = "BountyAmount")
private Long BountyAmount;
```

- **Constructors created**

```
public Votes() {
}
public Votes(Long postId, Long voteTypeId, Long id, Date creationDate, Long bountyAmount) {
    PostId = postId;
    VoteTypeId = voteTypeId;
    this.id = id;
    CreationDate = creationDate;
    BountyAmount = bountyAmount;
}
public Long getVoteId() {
    return VoteId;
}
public Long getVoteTypeId() {
    return VoteTypeId;
}
public void setVoteTypeId(Long voteTypeId) {
    VoteTypeId = voteTypeId;
}
public void setVoteId(Long voteId) {
    VoteId = voteId;
}
public Long getPostId() {
    return PostId;
}
public void setPostId(Long postId) {
    PostId = postId;
}
public Long getId() {
    return id;
}
public void setId(Long id) {
    this.id = id;
}
public Date getCreationDate() {
    return CreationDate;
}
public void setCreationDate(Date creationDate) {
    CreationDate = creationDate;
}
public Long getBountyAmount() {
    return BountyAmount;
}
public void setBountyAmount(Long bountyAmount) {
    BountyAmount = bountyAmount;
}
}
```

- **Method for viewing all fields**

```
@Override
public String toString() {
    return "Votes{" +
        "VoteId=" + VoteId +
        ", PostId=" + PostId +
        ", VoteTypeId=" + VoteTypeId +
        ", id=" + id +
        ", CreationDate=" + CreationDate +
        ", BountyAmount=" + BountyAmount +
        '}';
}
}
```

#### i. **Vote type model class**

- **Entity created**

```
@Entity
@Table(name = "VoteTypes")
public class VoteTypes {
```

- **Necessary fields are taken**

```
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
@Column(name = "VoteTypeId")
private Long VoteTypeId;

@Column(name = "VoteTypeName")
private String VoteTypeName;
```

- **Constructors created**

```
public VoteTypes() {  
}  
public VoteTypes(String voteTypeName, Long voteTypeId) {  
    VoteTypeName = voteTypeName;  
    VoteTypeId = voteTypeId;  
}  
public Long getVoteTypeId() {  
    return VoteTypeId;  
}  
public void setVoteTypeId(Long voteTypeId) {  
    VoteTypeId = voteTypeId;  
}  
public String getVoteTypeName() {  
    return VoteTypeName;  
}  
public void setVoteTypeName(String voteTypeName) {  
    VoteTypeName = voteTypeName;  
}  
}
```

- **Method for viewing all fields**

```
@Override  
public String toString() {  
    return "VoteTypes{" +  
        "VoteTypeId=" + VoteTypeId +  
        ", VoteTypeName=" + VoteTypeName + '\n' +  
        '}';  
}  
}
```

## 9.2 DAFFODIL IDEA HUNT

### 9.2.1 The programming language and framework which I choose and why

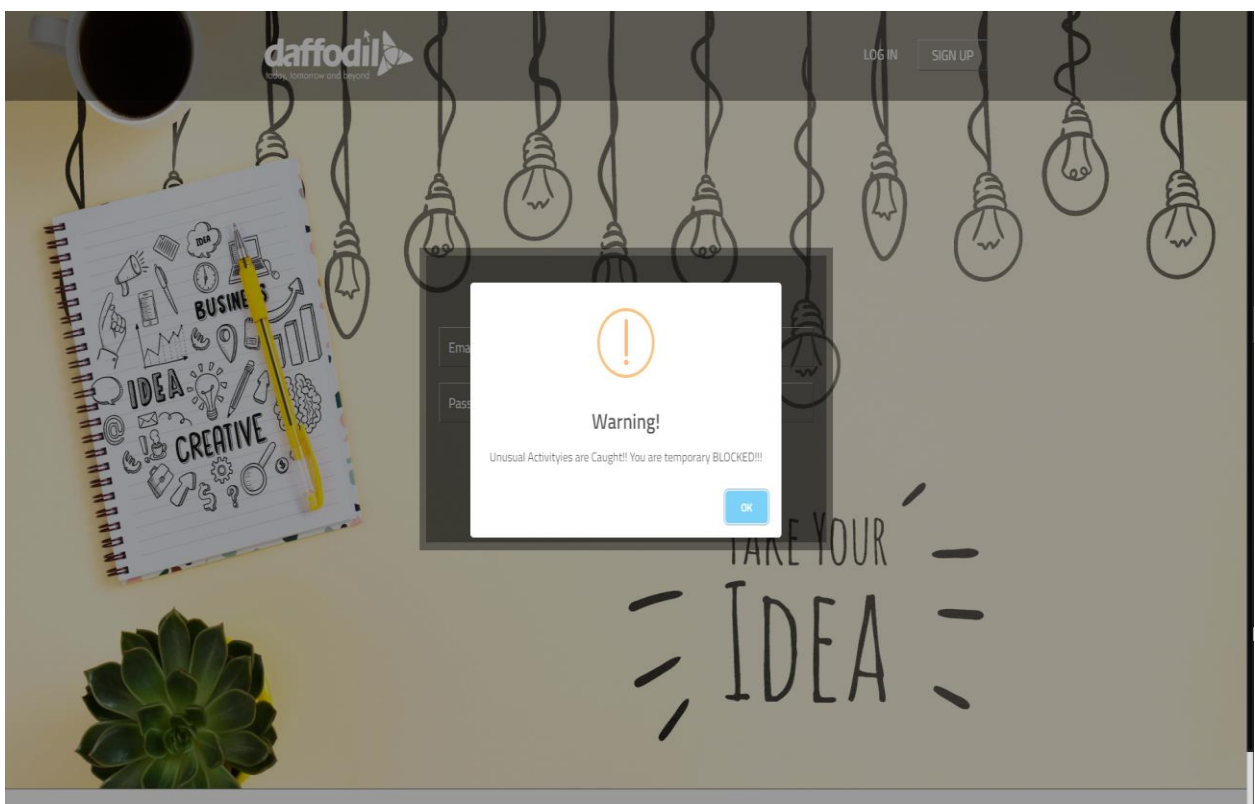
In this project I use PHP, it is a web programming language also it is a scripting language which is interpreted at runtime. I am using PHP in my project because PHP runs on different platforms like Linux, Unix etc. and it is simple and runs efficiently on the server. The main reason for using PHP is that it is executed without compiling. Hypertext Preprocessor (PHP) code can be embedded into HTML code and it can be used in combination with various web content management systems. In my project HTML is added as an extra advantage for using PHP and I am showing the result or output to use HTML. I am manipulating HTML tags by using PHP. I manage the database of my project by using PHP. It makes an easy way for working hand to hand with database servers e.g. MySQL. I make my project dynamic as I can. I take data by using HTML and put this data into the database through the PHP. Collect data from the website and pass it through the server with checking the validation of the data using PHP. I use PHP for sending mail to the user. I encrypt the user data and easily can find the today's date and time by using PHP. In this project I am creating a special area of the website for its members. I create the login and registration with validation by using PHP. Users can add any member, create, write, delete, modify elements within the database through the PHP. Users can block or restrict users to access some pages of this system. In this project PHP handles all forms and its gather data and save data to a file and it can redirect to user in different pages.

### 9.2.2 Details paragraph about each working function

**Login:** I created a login page for all types of users with validation.



**Figure 23: This is the login page.**



**Figure 24: Temporary block user cannot login.**



**Registration:** There are two registration page one is for employee and another is Student

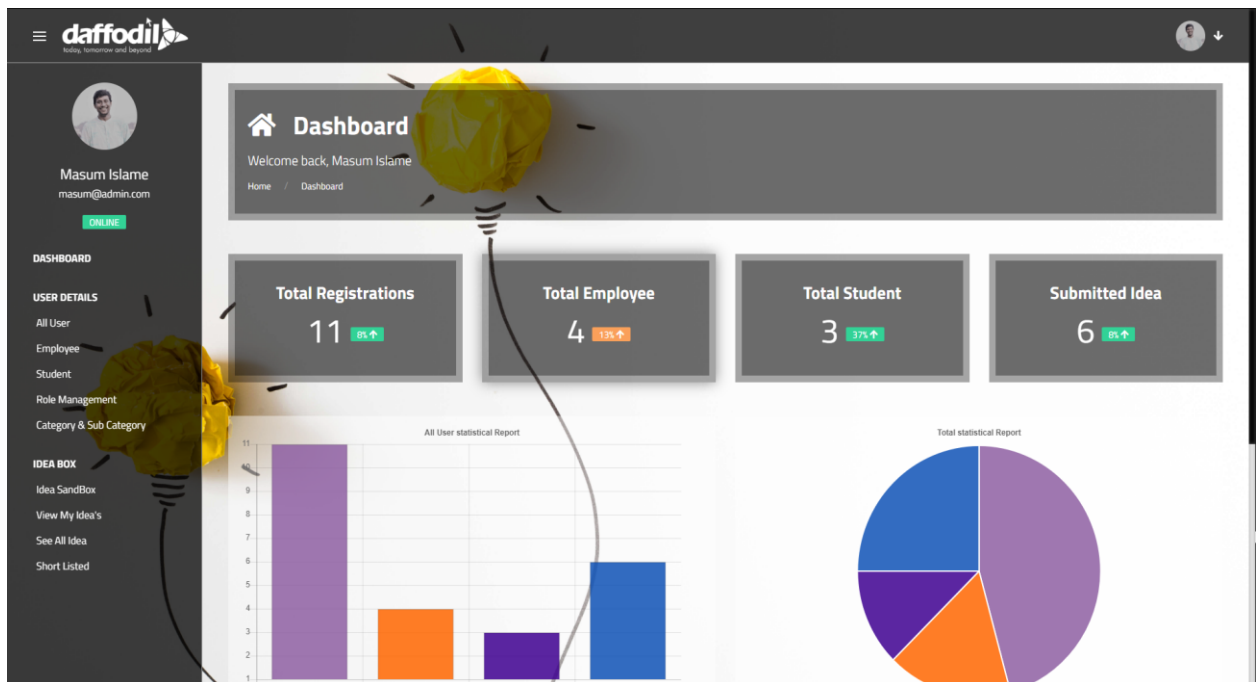
The screenshot shows the 'Student Registration Form' on the Daffodil International University website. The form is located on the right side of the page, with a background image of a hand-drawn brain with an arrow pointing to it and the word 'idea' written below. The form fields include: Full Name, Email, Select Institution Name?, Select Department Name?, Academic ID, Password, Confirm Password, Phone No., mm/dd/yyyy, address, Choose Profile Picture, Choose ID Card Attachment, and a SUBMIT button. The website header includes the Daffodil logo and 'LOG IN' and 'SIGN UP' links. The footer contains the copyright notice '© 2020 DIA Software Team All Rights Reserved' and 'Developed by Jashim Uddin Ahmed'.

**Figure 25: Student registration.**

The screenshot shows the 'Employee Registration Form' on the Daffodil International University website. The form is located on the right side of the page, with a background image of a blue crumpled paper ball. The form fields include: Full Name, Email, Select Organization Name?, Employee ID, Designation, Password, Confirm Password, Phone No., mm/dd/yyyy, LinkedIn link, address, Choose Profile Picture, Choose ID Card Attachment, and a SUBMIT button. The website header includes the Daffodil logo and 'LOG IN' and 'SIGN UP' links. The footer contains the copyright notice '© 2020 DIA Software Team All Rights Reserved' and 'Developed by Jashim Uddin Ahmed'.

**Figure 26: Employee registration.**

**Dashboard:** User can view his/her information with the permission or rights is given. It's works dynamically.



**Figure 27: Dashboard page.**

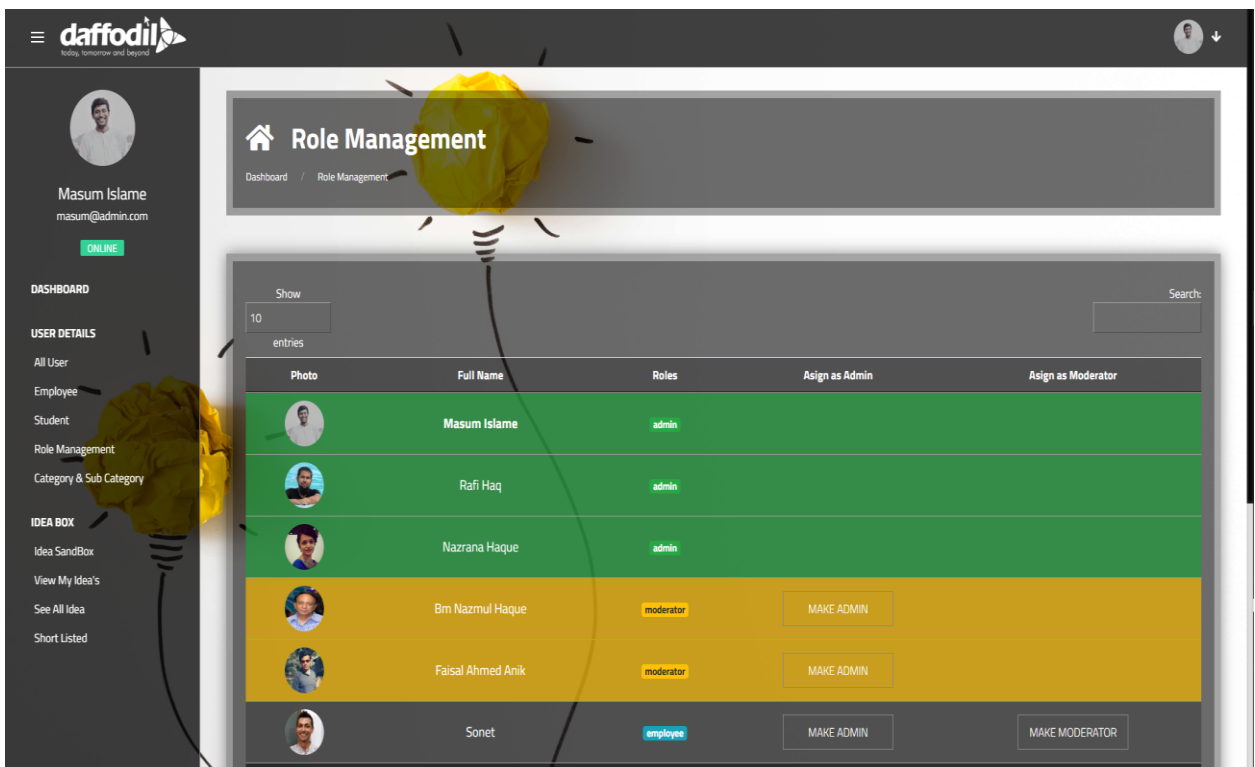
The user management page displays a list of users and students with the following columns: Photo, Full Name, Roles, View Details, and Approve. The table contains the following data:

Photo	Full Name	Roles	View Details	Approve
	Masum Islame	admin	<a href="#">+</a>	
	Rafi Haq	admin	<a href="#">+</a>	
	Nazrana Haque	admin	<a href="#">+</a>	
	Bm Nazmul Haque	employee	<a href="#">+</a>	APPROVED
	Sonet	employee	<a href="#">+</a>	APPROVED
	Rakibul Hasan	employee	<a href="#">+</a>	APPROVED
	Ataul Hasan Sakib	employee	<a href="#">+</a>	APPROVE
	Jashim Ahmed	student	<a href="#">+</a>	APPROVED
	M Rahman Munna	student	<a href="#">+</a>	APPROVED

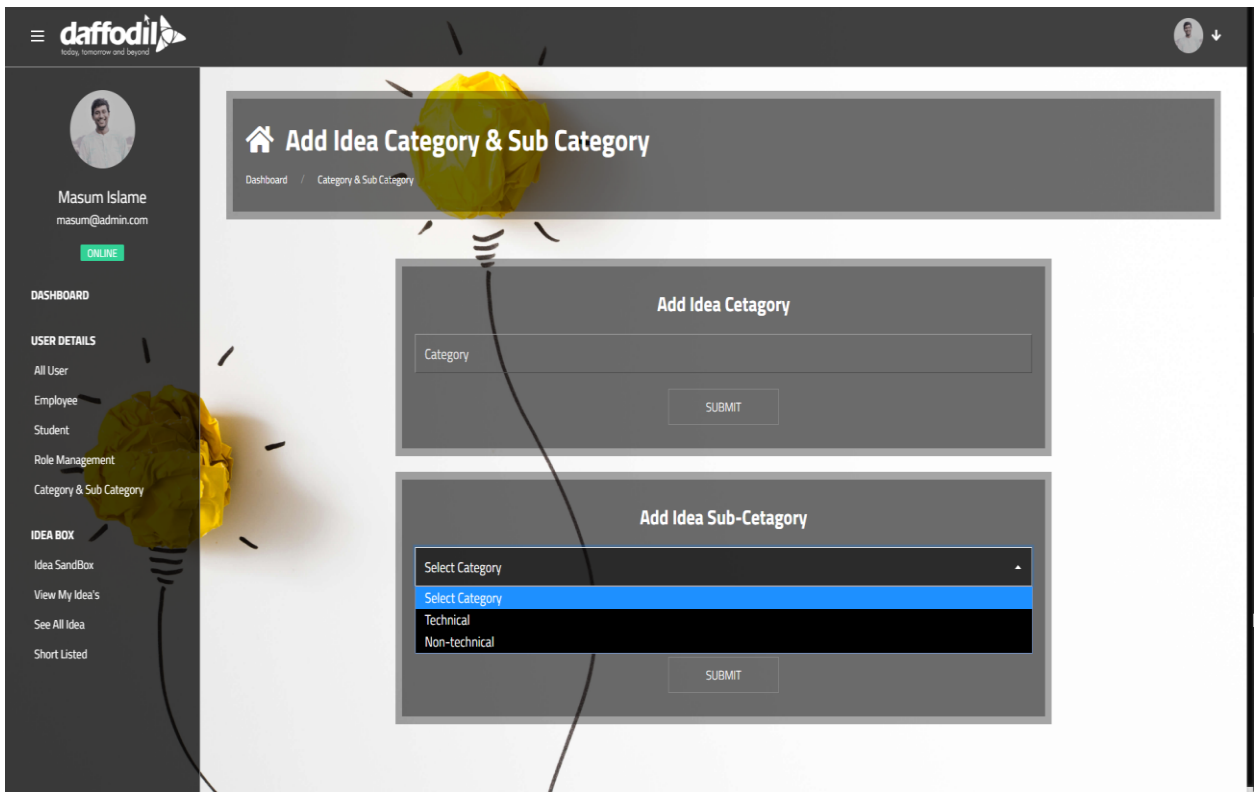
**Figure 28: View All user and student.**



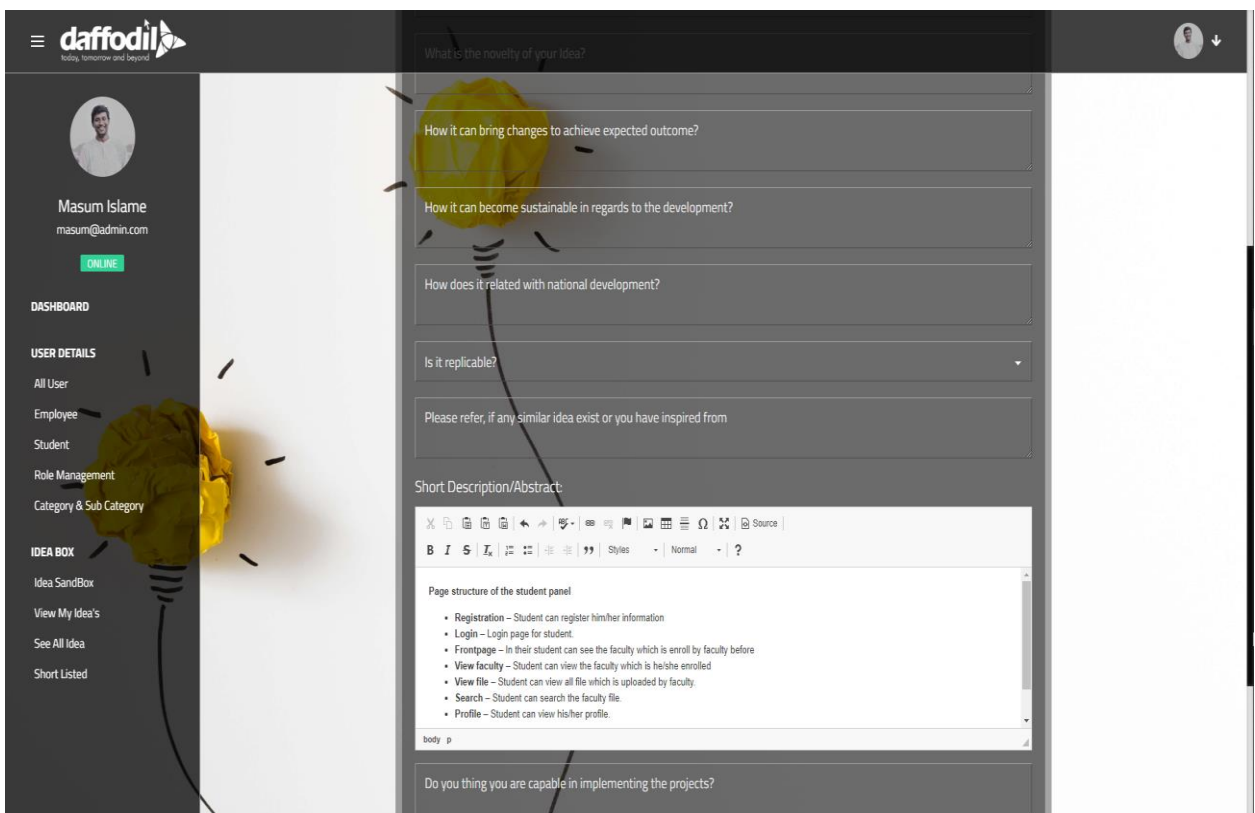
**Figure 29: Admin can view every user's profile individually.**



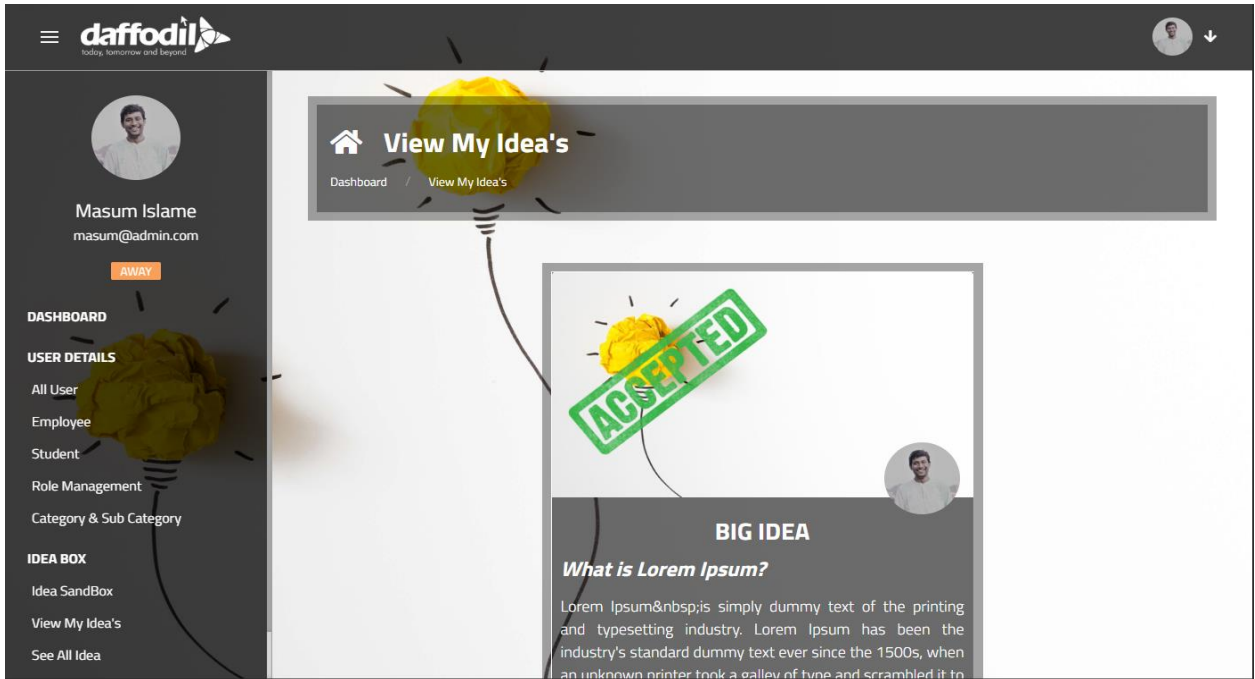
**Figure 30: Admin can manage every user role and permission.**



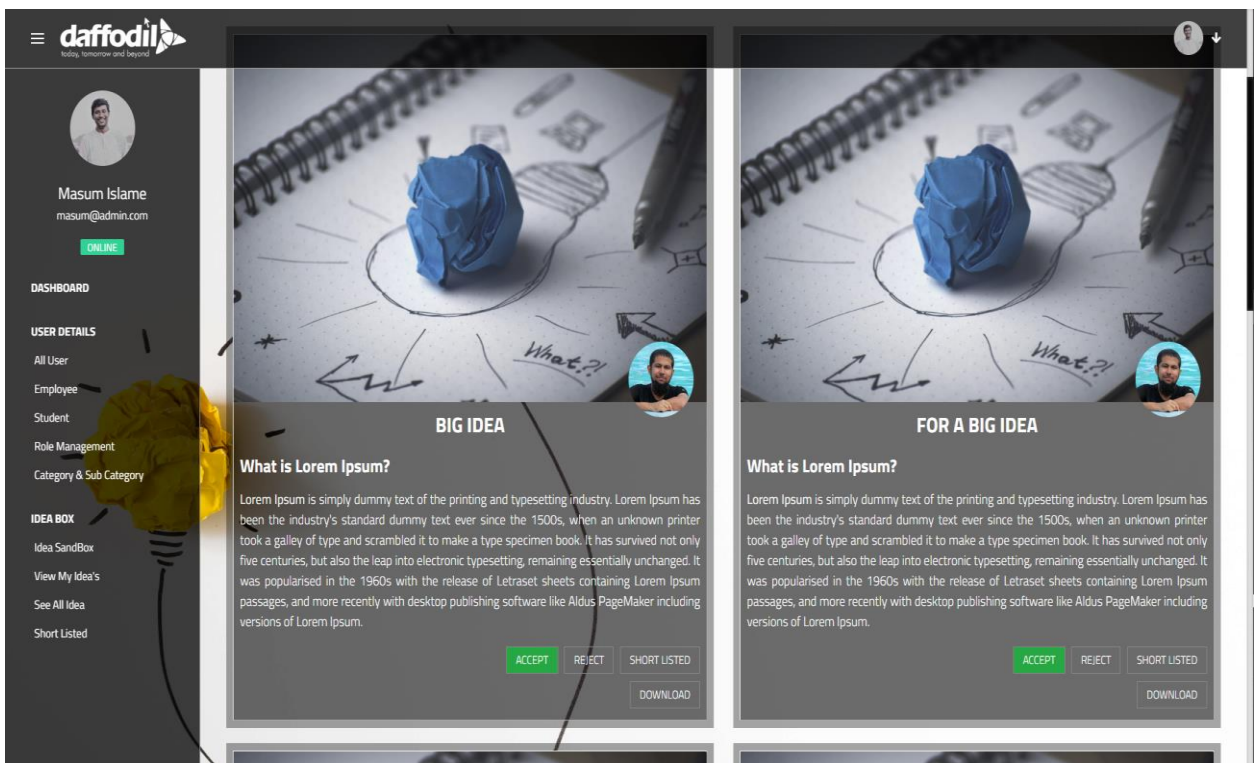
**Figure 31: Admin can add category of idea and sub category of idea.**



**Figure 32: User can share their idea.**



**Figure 33: User can view his idea.**



**Figure 34: Admin can view all idea which was uploaded and have right to give any kind of status.**

# Chapter 10 - Testing

## 10.1 OAIHUB

### 10.1.1 Unit testing

**Test Priority:** High

**Test Execute by:** DIA Intern Team

**Unit test No:** 01

**Test Execute Date:** 10/05/2020

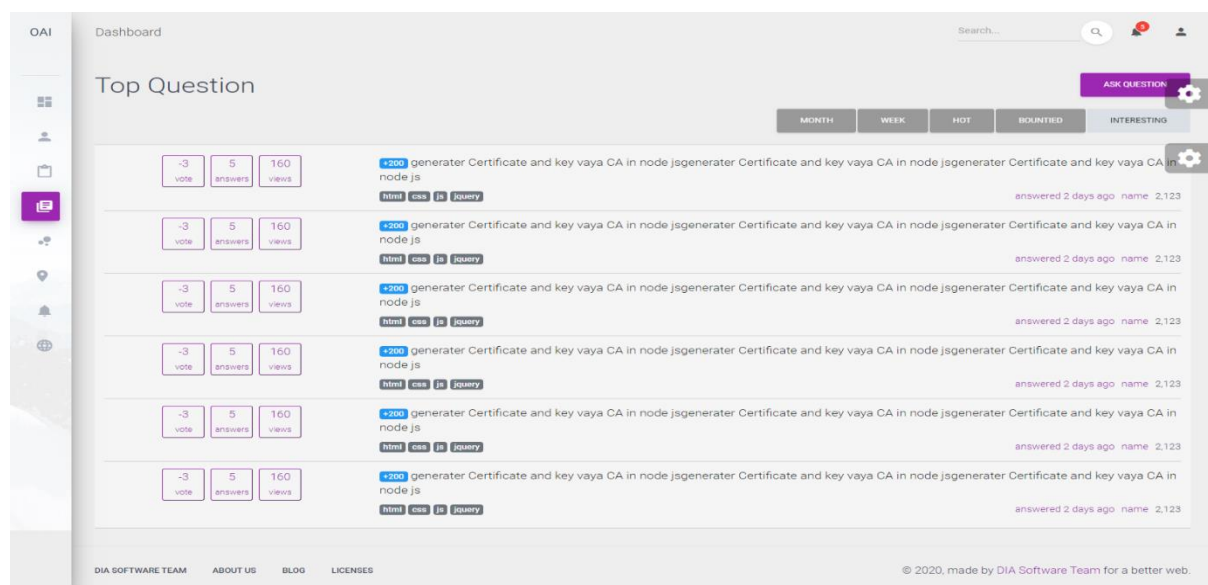
**Test case:** Forum View

**Objective:** The post came up properly on this page

**Data Source:** What the user is posting

Case No.	Description	Tasks	Result		Status (Pass/Fail)
			Actual result	Expected result	
1	We will check if the post is coming to this page properly	Post some question	All post show in the page	All post will be show in the page	Pass

The test result screenshot for Unit Test is given below





Test Priority: High

Test Execute by: DIA Intern Team

Unit test No: 02

Test Execute Date: 10/05/2020

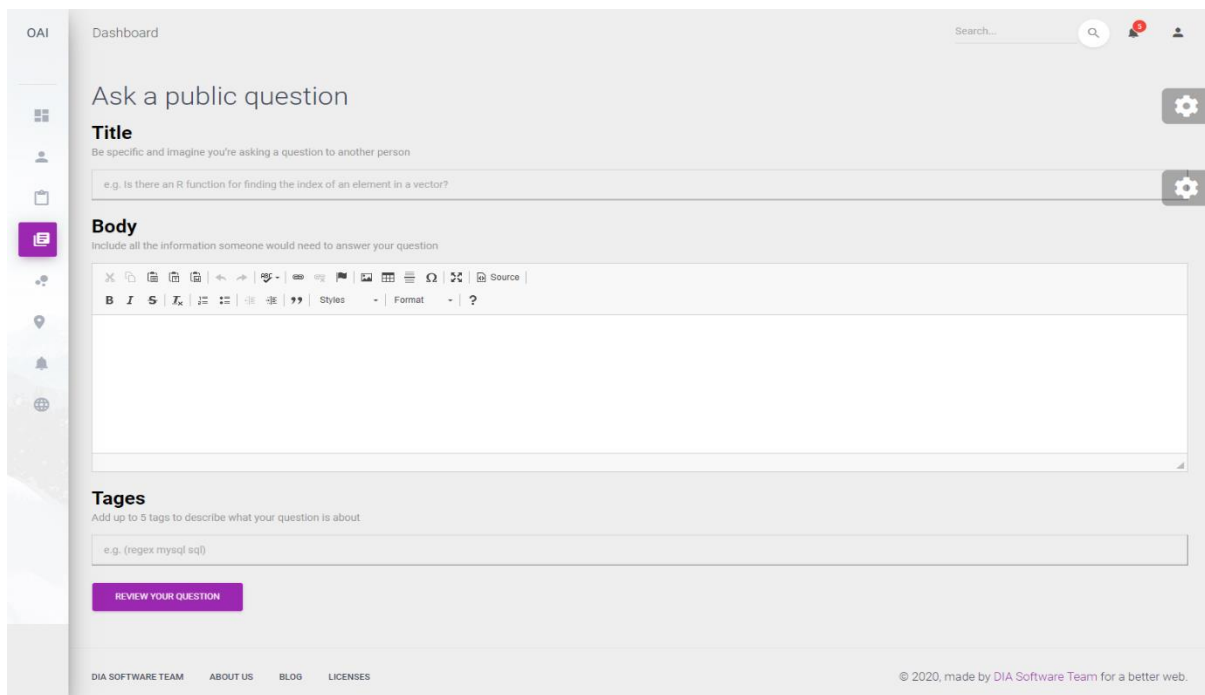
Test case: Public post checking

Objective: The user can post properly

Data Source: User Input

Case No.	Description	Tasks	Result		Status (Pass/Fail)
			Actual result	Expected result	
1	We will check here whether the user can post using all the inputs	Enter the information like title, description and tags	The post is successfully submitted	The post will be successfully submitted	Pass

The test result screenshot for Unit Test is given below



Test Priority: High

Test Execute by: DIA Intern Team

Unit test No: 03

Test Execute Date: 10/05/2020

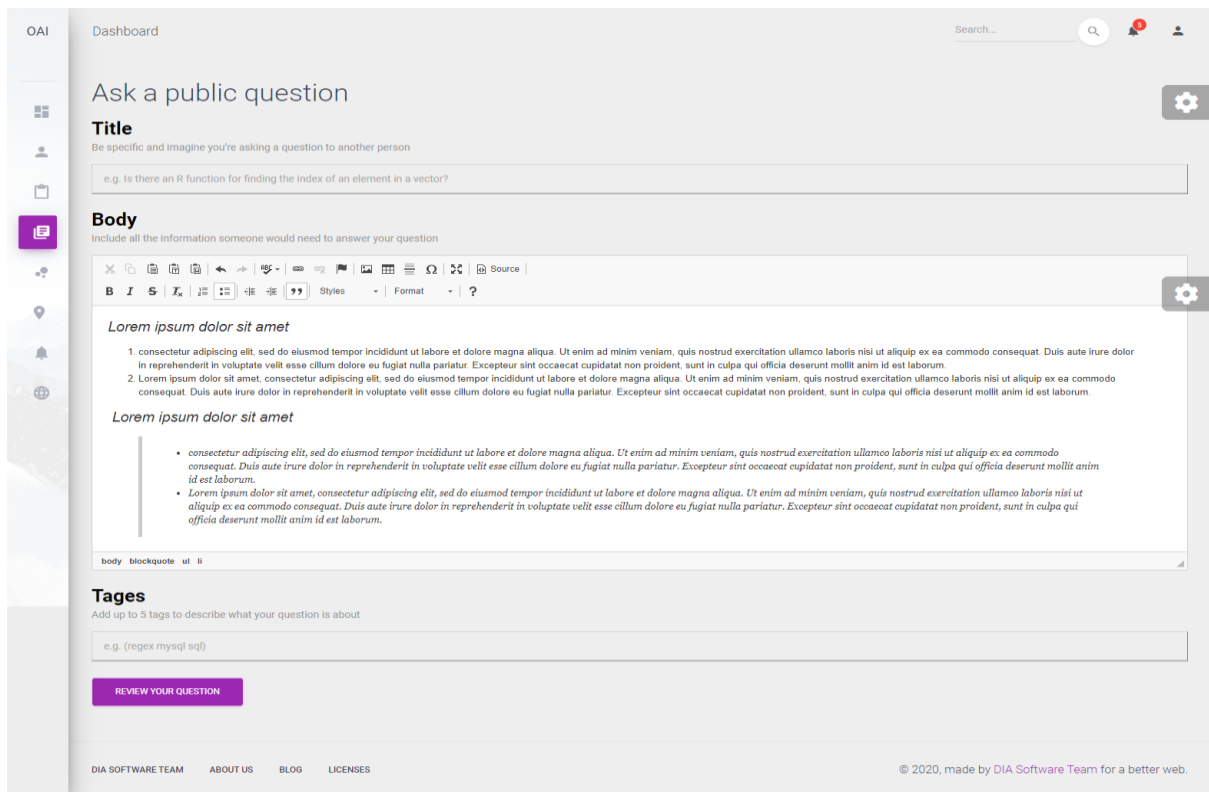
Test case: Text Editor text

Objective: Checking the text editor

Data Source: User Input

Case No.	Description	Tasks	Result		Status (Pass/Fail)
			Actual result	Expected result	
1	We will check the functionality of the text editor	Provide the any kind of data	Text editor work properly	Text editor will be work properly	Pass

The test result screenshot for Unit Test is given below





Test Priority: High

Test Execute by: DIA Intern Team

Unit test No: 04

Test Execute Date: 10/05/2020

Test case: User Login Window

Objective: Password checking test.

Data Source: User input

Case No.	Description	Tasks	Result		Status (Pass/Fail)
			Actual result	Expected result	
1	Test for checking wrong password.	Enter login Info, User Email: Password:	An alert message Shown	An alert message will be Shown	Pass

The test result screenshot for Unit Test is given below

OaiHub

Email  
dity786@gmail.com

Password  
.....

Login

OaiHub

Invalid Username or Password... x

Email  
Enter Email

Password  
Enter Password

Login

## 10.1.2 Integration Testing:

Test Priority: High

Test Execute by: DIA Intern Team

Integration test No: 01

Test Execute Date: 10/05/2020

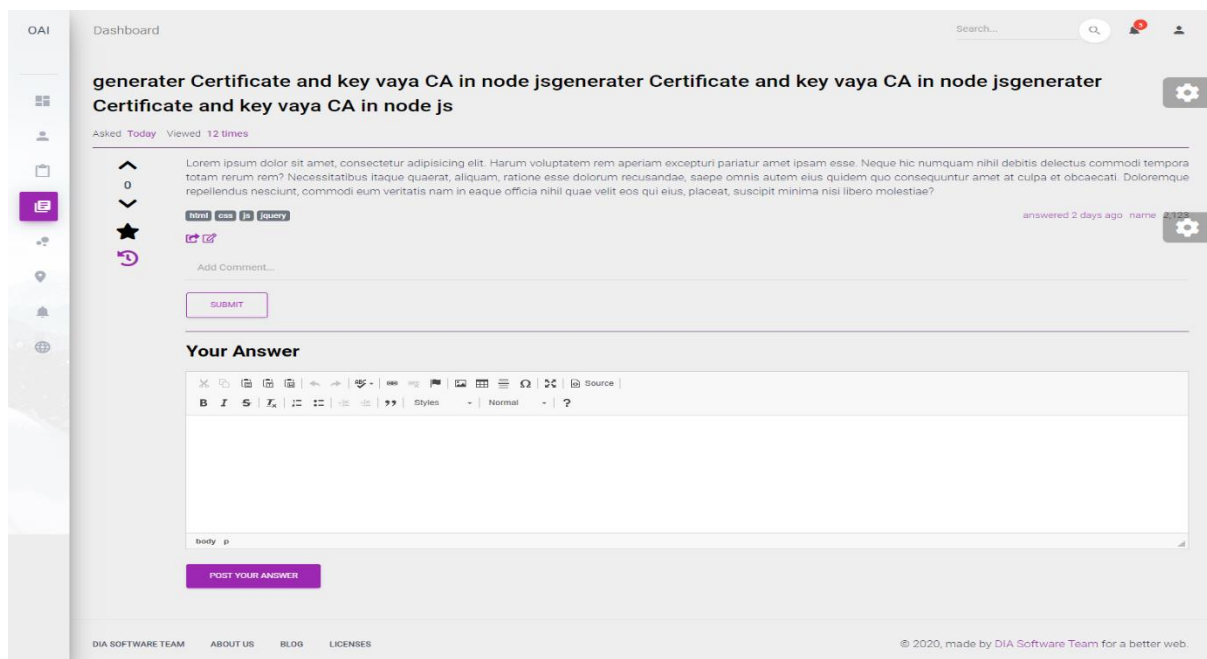
Test case: Comment Testing

Objective: Check the comment part

Data Source: User Input

Case No.	Description	Tasks	Result		Status (Pass/Fail)
			Actual result	Expected result	
1	We will check the comment function here to see it is working properly	Provide the data	The comment is successfully submitted	The comment will be successfully submitted	Pass

The test result screenshot for Integration Test is given below



Test Priority: High

Test Execute by: DIA Intern Team

Integration test No: 02

Test Execute Date: 10/05/2020

Test case: Submitted the Ans

Objective: Check the Ans submission

Data Source: User input

Case No.	Description	Tasks	Result		Status (Pass/Fail)
			Actual result	Expected result	
1	User is able to submit the answer properly. We will check this function	Provide the data	The Ans is successfully submitted	The Ans will be is successfully submitted	Pass

The test result screenshot for Integration Test is given below

The screenshot shows a web interface for a Q&A forum. The question is: "generator Certificate and key vava CA in node js". The post is marked as "Asked Today" and "Viewed 12 times". The question body contains placeholder text: "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Harum voluptatem rem aperiam excepturi pariatur amet ipsam esse. Neque hic numquam nihil debitis delectus commodi tempora totam rerum rem? Necessitatibus itaque quaerat, aliquam, ratione esse dolorum recusandae, saepe omnis autem eius quidem quo consequuntur amet at culpa et obcaecati. Doloremque repellendus nesciunt, commodi eum veritatis nam in eaque officia nihil quae velit eos qui eius, placeat, suscipit minima nisi libero molestiae?". The post has 0 answers and 0 votes. Below the question is a section for "Your Answer" with a rich text editor containing the same placeholder text. A "POST YOUR ANSWER" button is visible at the bottom of the answer section.

Test Priority: High

Test Execute by: DIA Intern Team

Integration test No: 03

Test Execute Date: 10/05/2020

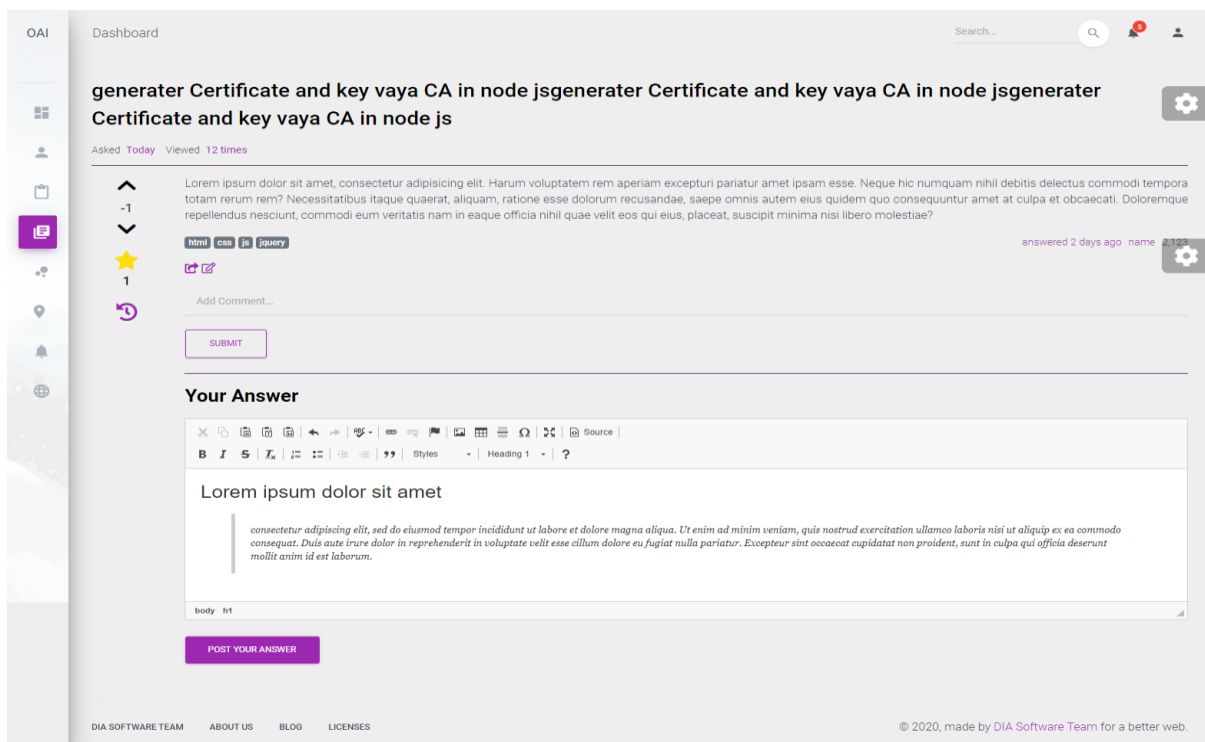
Test case: Bookmark and voting

Objective: Check bookmark and voting functionality

Data Source: User Input

Case No.	Description	Tasks	Result		Status (Pass/Fail)
			Actual result	Expected result	
1	We will check this functionality of bookmarking and voting here	Action by user click	Bookmark and voting successfully happened	Bookmark and voting will be successfully happened	Pass

The test result screenshot for Integration Test is given below



The screenshot displays a web application interface. At the top, there is a search bar and a user profile icon. The main content area shows a question titled "generator Certificate and key vava CA in node js" with a "SUBMIT" button. Below the question, there is a "Your Answer" section with a rich text editor containing the text "Lorem ipsum dolor sit amet...". The footer of the page includes "DIA SOFTWARE TEAM", "ABOUT US", "BLOG", "LICENSES", and a copyright notice: "© 2020, made by DIA Software Team for a better web."

## 10.2 DAFFODIL IDEA HUNT

### 10.2.1 Unit, System and module Testing outcome and errors

Testing is an important part for develop a defect free software. There are different types of testing:

### 10.2.2 Interface Testing:

In this part we will checks this site whether all the transportations between the application server and the web server run efficiently or not. We need to check not only the communication methods but also check the showing of error message. That's means if the web server or database server returns an error message against any query than application server will collect the message from those servers and display the error message accordingly to the users.

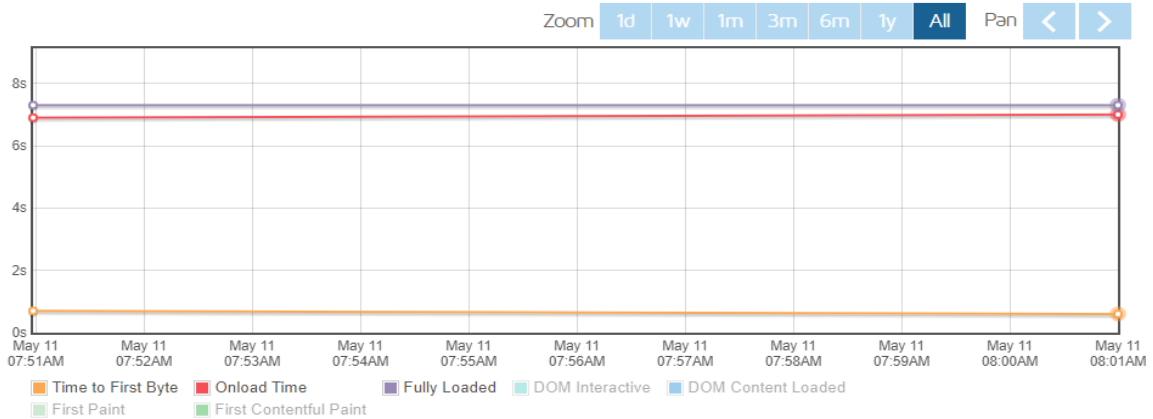
We found few areas of interface testing those are given below:

- **Application:** In here we will test the request which are sent properly to the database and fetch the correct data and display at the user side. If any errors are found in the application it should be display in the admin side not display in the user end.
- **Web Server:** The web server needs to be checked very well because the application has many requests it must be handled those requests without any service denial.
- **Database Server:** We need to test database properly and to be sure that the queries are sent to the database correctly and execute the appropriate results.

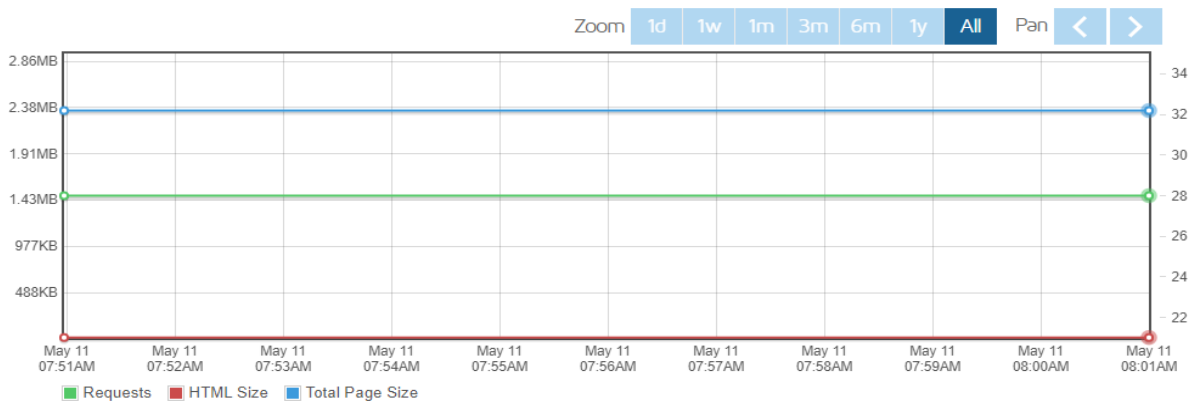
### 10.2.3 Performance Testing:

In this part we will check the performance of this web application. We test the loading time of the page, server response rate, application response time at different connection speeds etc. This testing contains monitoring at several internet speeds and with standard and peak loads. Some points are given below:

## Page timings



## Page sizes and request counts



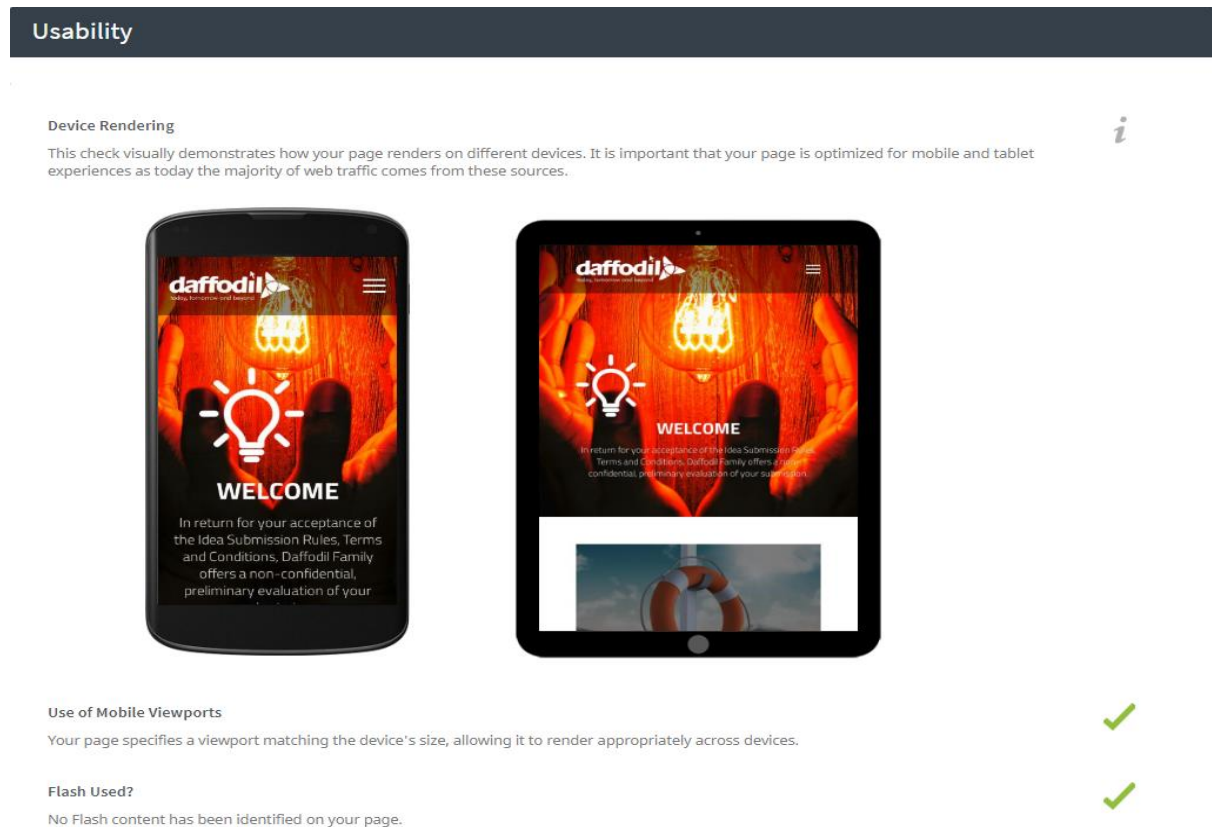
**Figure 35:** In this graph we can see that overall performance report.

- Check the web application data transfer time rate under different connection with different speeds.
- Test the loading time each of the web page for this web applications.
- If the site gets an interruption while loading, does it create any problems or not, if the problem is occurring it should be preventing the problem.
- Stress testing is helpful for estimating the breakpoint of this web applications, which includes placing this applications under-stress until it stops executing.
- Also check the style sheet files, script files to sure that they are minified. And make sure the all images including all the files are optimizes and compressed it reduce the loading time of this application.

### 10.2.4 Usability Testing:

Usability is what makes a web application more acceptable to a user. Day by day usability testing is becoming an important part of a web base project. This testing can

be accomplished by the external testers who represent the intended user base, or the designers can do this testing internally. Usability test is not the same as the user experience testing as the targets are entirely different and so are the phases of the production of the device at which these assessments are carried out. We will also check this website responsive and mobile friendly.



**Figure 36: The usability test report of this site.**

We can divide the usability test through a few processes those are given below:

- Menus, Navbar/Navigation bar, buttons or links to several pages on this website would be simply accessible on all websites and consistent. Color schemes should be user friendly.
- The content of this site should be appropriate. The web page cannot contain any linguistic mistakes or misspellings and also, we have to check if there are any grammatical mistakes exist or not.
- We also check the image alt attribute. The alt attribute helps those users who is physically disable. By using this attribute, they can understand the image easily.

### 10.2.5 Black Box Testing:

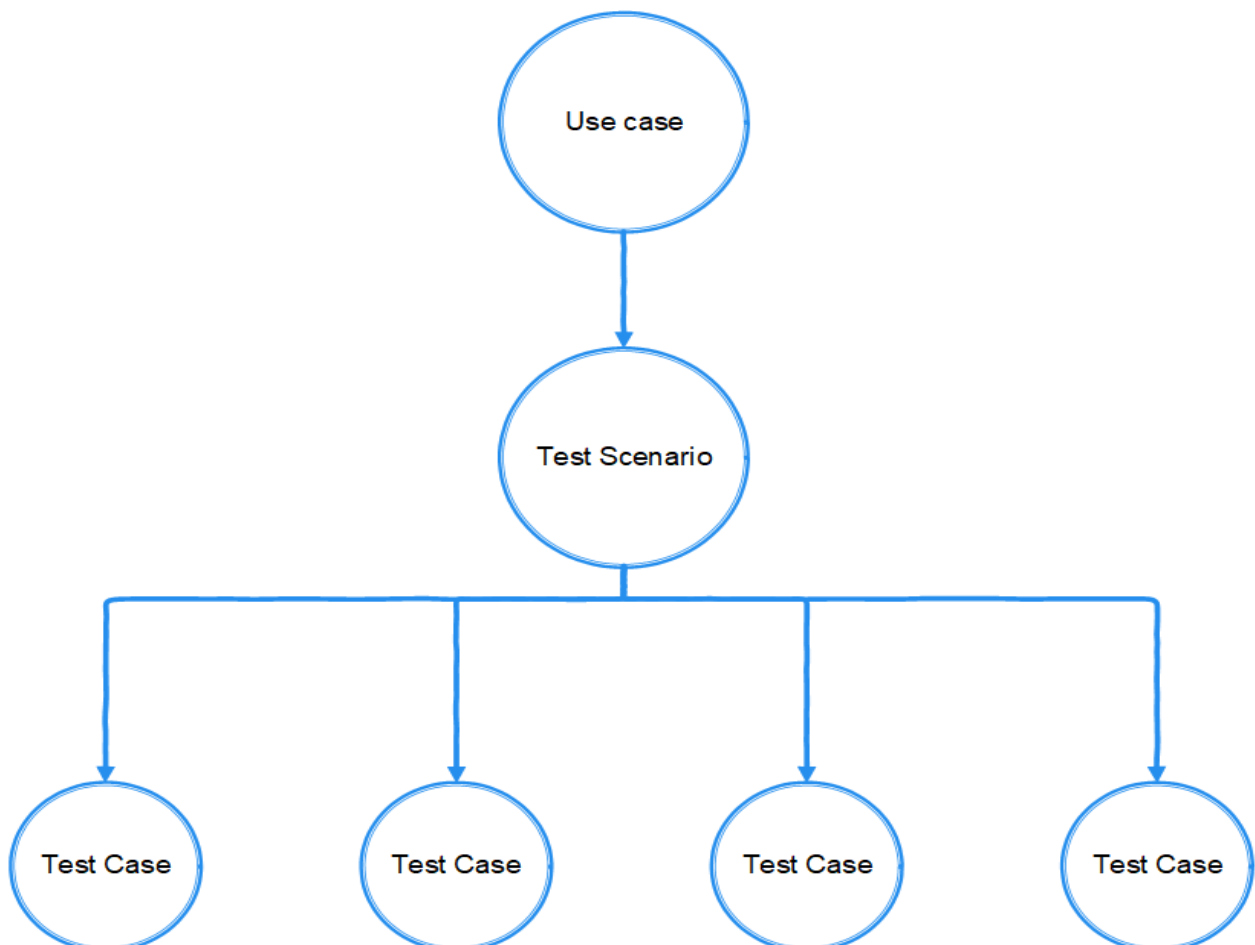
This testing is a testing where examine the functionality of a software without exploratory its interior structure. This type of testing is not investigative by the tester.

### 10.2.6 White Box Testing:

In this testing is a testing where examine the interior structure of a software. This type of testing is investigative by the tester.

### 10.2.7 Identify Possible Test Scenarios:

A test scenario is a document specifying the application's features to be tested. This is mainly used to estimate a function end to end and is usually extracted from the use cases. A single test scenario can concealment a single test case or many. An examination scenario consequently has a one to many relationships with the test cases.





Scenario testing is a type of testing coordinately using scenarios that are derivative from use cases. Complex application-logic can be checked using scenario testing using informal to evaluate test scenarios.

**a. Test Case for Registration**

**Test Priority:** High

**Test Execute by:** Jashim Uddin Ahmed

**Module Name:** Registration part

**Test Execute Date:** 02/05/2020

**Pre-requisite:** Valid Data

**Post-requisite:** NA

**Test Title:** Verify registration with valid email address and data.

**Test Description:** Test the Idea bank registration page.

**Pre-condition:** User has valid email address and data.

Case No.	Test scenario	Test case	Test data	Result		Status (Pass/Fail)
				Actual result	Expected result	
01	Registration Functionality	Registration: Check registration button for correct data	User Name: Test Email: <a href="mailto:test@gmail.com">test@gmail.com</a> Phone: +8801...56 Address: 1/A, Dhaka Password: 12345	Registration successful	Registration will be successful	pass
02	Registration Functionality	Registration: Check registration	User Name: 101 Email: test@gmail.com Phone: test	Registration unsuccessful	Registration will not be	pass

on button action for incorrect data	Address: 0.16 Password: 123@321A	<b>successful</b>
--	--	-------------------

### b. Test Case for login

**Test Priority:** High

**Test Execute by:** Jashim Uddin Ahmed

**Module Name:** Login part

**Test Execute Date:** 02/05/2020

**Pre-requisite:** A valid account

**Post-requisite:** NA

**Test Title:** Verify login with valid email address and password.

**Test Description:** Test the Idea bank login page.

**Pre-condition:** User has valid email address and password.

Case No.	Test scenario	Test case	Test data	Result		Status (Pass/Fail)
				Actual result	Expected result	
01	Login Functionality	Check login to block this site's cookie	Email Address: <a href="mailto:jashimahmed@gmail.com">jashimahmed@gmail.com</a> Password: jashim12345	<b>Login unsuccessful</b>	<b>Login will not be successful</b>	pass
02	Login Functionality	Check login to unblock this site's cookie	Email Address: jashimahmed@gmail.com Password: jashim12345	<b>Login successful</b>	<b>Login will be successful</b>	pass

03	Login Functionality	Check login with invalid email & password	Email Address: <a href="mailto:jashimahme.com">jashimahme.com</a> Password: jashim12345	<b>Login unsuccessful</b>	<b>Login must not be successful</b>	pass
04	Login Functionality	Check login with valid email & password	Email Address: <a href="mailto:jashimahme@gmail.com">jashimahme@gmail.com</a> Password: jashim12345	<b>Login successful</b>	<b>Login must be successful</b>	pass

### 10.2.8 Quality Assurance technique

Quality Assurance is defined as an activity to ensure that users receive the best service possible from an organization. First of all, I am trying to finish all the requirement. Information is collected and categorized on software errors and defects. Developing and testing systems and also "Do" system adjustments. Quality control's main objective is to ensure that the goods follow the user's expectations and requirements.

## Chapter 11 - Implementation

Actually, implementation designates the users process of the organization's or institution's workflow. User attempts to enhance the integrated work using structured method. Implementation is a strategy, method, or any design, idea, model, description, standard or policy for undertaking something. Implementation, as like, is the exploit that must track any introductory thinking so that something actually happens. The project team generates the actual invention during implementing. Implementation of the invention can be an exciting phase for the user, because their project idea becomes something tangible. The project developers start building the software and coding it. In the section I will describe how I have implemented everything about this process.

### 11.1 Training

Implementation includes a sequence of activities, through which training managers bring the course according to the approved design to the learners. It necessitates scheduling of courses, faculties, equipment and service providers aside from arranging ongoing support for the classroom, and ensuring the smooth flow of activities as per the plan. A systematic, step-by-step process is used to build an effective training program. Training initiatives which are standing alone often fail to meet organizational goals and expectations of the participants.

#### 11.1.1 Assess training needs

Identifying and evaluating needs is the first step to developing a training program. Training needs of employees may already be identified in strategic, human resources or individual development plans of the organization.

#### 11.1.2 Set organizational training objectives

Assessments of the training needs (administrative, task & separate) will identify any gaps in your current training initiatives and skill sets for employees. These gaps should be analyzed and prioritized, and transformed into the training goals of the organization.

The ultimate aim is to bridge the gap between current performance and desired performance by developing a training program.

### **11.1.3 Create training action plan**

The next step is to create a comprehensive action plan which includes theories of learning, instructional design, content, materials and any other elements of training. Methods for delivering resources and training should be detailed as well. The level of training and the learning styles of the participants also need to be considered when developing the program.

### **11.1.4 Implement training initiatives**

The phase of implementation is where the workout program comes to life. Organizations need to choose whether in-house or outwardly coordinated training will be provided. Implementation of the program includes scheduling training activities and organizing any associated resources facilities, equipment, etc. Subsequently, the training program is officially launched, promoted and run.

### **11.1.5 Evaluate & revise training**

As mentioned in the last segment, there should be continuous monitoring of the training program. Eventually, the entire program should be evaluated to regulate whether it was successful and meeting the training goals. Feedback from all stakeholders should be obtained in order to determine the effectiveness of the program and instructor and also knowledge or skill acquisition. (training-program, 2020)

## **11.2 Big Bang**

Big bang implementation on a single site is considerably easier to manage over multiple sites than a simultaneous big bang. The usually held view is that implementations with big bang have an inherently higher risk level. In one instance, implementation happens. On a given date, all users move onto the new system. The scope of a big bang implementation can also mean that it is problematic to accomplish

complete end to end system testing and it is only when the system goes live that all interdependencies are fully tested.

## Chapter 12 - Critical Appraisal and Evaluation

Within this chapter the project overview will be explained. In this topic both the success and system failure will be discussed. What's more, what experience we gained throughout the project will also be discussed. Likewise, it includes the success factor, the amount of objective it met, and what features it could not have met and the reason with the justification. In this section we will describe how I can meet my objective goal despite various obstacles.

### 12.1 Objective that could be met

The project proposed has met some objectives outlined below,

- a. Implementing a platform for all student and guardian can get any information about any institution.
- b. This application provides previous question bank with solutions, which can help the students and they can find these easily.
- c. This is the mature platform for both students and the educational organization.
- d. Fix a methodology suitable for implementing a system.
- e. Make the project well documented with maintenance the standard.
- f. The application has to error free as much as possible.
- g. Messaging and commenting system were created for the user (student and guardian) to organization communication.
- h. We are making this application's database which is based on this project requirement.
- i. This application is web-based application which is portable easy to use from anywhere.
- j. We are making this application with met the requirement which is given by our moderator.

- k. We have made this system useful for different types of user and that will be very helpful for the user.

### **12.1.1 Success rate against each objective**

Success rate to this objective is relatively satisfactory. This rate is impartially alright of this objective as both student/guardian and organizations can be able to get their own expected outcome. The organizations or institutions can share their information and student can view their descriptions and get their expected information's this thing will help reduce the hassle of the students also guardian. And it will help more students to collect questions from different years. As a student and as a guardian they will get exactly the information they need from here.

### **12.1.2 How much better it could be done**

We are following many structures to standardize this web application. There were many diagrams in these diagrams that helped us a lot to maintain our sequence like as activity diagram, sequence diagram, ERD diagram, use case diagram and class diagram etc. At the same time, we have tried to do proper documentation of all the diagrams which will help to do more with it in the future. There are some diagrams that we could not add to this documentation because of the security of the organization. And there are a lot of options here that have been made for the student or organizational user in a very convenient way to understand.

### **12.1.3 How better are the features of the solution?**

This web application might be more friendly to the users. If we want to show as an example then we will see that the forum part has been made very interactive. Any student or user can ask his/her questions at any moment. And there is a facility to comment here, if anyone knows the answer, it can be informed through comments which is very much responsive. As a result, it can solve any types of problem very quickly and takes very close to the solution. If the time of the project is extended a bit this system is better than this. The workload was so high, though it was teamwork.



## **12.2 Objectives totally not met / touched**

### **12.2.1 OAIHUB**

In this section we will discuss which things we have not been able to do properly and which we have repeatedly failed to do. We have been able to identify some of the reasons why we have not been able to properly deploy these items and have been discarded and I will give some more reasons here in a specific way so that we can identify very easily. I will write in this section about how we overcome this thing after repeated failures.

#### **a. Why it could not be touched**

For implement this objective we needed proper planning for the work and skills to increment strongly. The reason is that at the beginning of the work we did not get the proper planning and at the same time we lacked some skills. That's why we could not able to meet some requirements in this web application. If we want to say, we will say that we have not done the payment part of the pro feature yet. There are some other user interface functions that are not interactive. And we want to do more with it in the future. Hopefully we can do it all when the future version comes out.

#### **b. What could have been done**

We have tried to reorder our plans properly, if we have to give an example, we have to make a note of what we will do one day. We kept our notes in different box forms on priority basis. What we have to do first, what we have to do later. We decorated those boxes with these things and by doing this we have achieved a fairly good level of success. Then if we want to say we will talk about our skills because we had a lot of lacking's. We used to do a session in between our work every day so that we could develop our skills and that helped us a lot. Time maintenance is the process for properly implementing it and properly overcoming it from the situation also done a lot of security related work in this project.

## 12.2.2 DAFFODIL IDEA HUNT

### a. The requirements that cannot be implemented completely and how I overcome them

In this project I cannot implement one thing that is “draft box”. In this fracture user can save his or her idea as a draft. When the user wishes, he or she will write and he or she can pause it if he or she wants. This feature I cannot implement for the reason due to the shortness of time. This feature was important for this project.

## **Chapter 13 - Conclusion**

### **13.1 OAIHUB**

#### **13.1.1 Conclusion**

To develop this project 'OAIHUB' successfully I have tried our best. The overview of the entire documentation contains the goals and the success inside this section. This section also defines implementing knowledges and project values. I have faced many problems during the development and our mentor help us to overcome the situation. Many more features are not done yet for the short time of the project that will be developed in the next. Here I outlined my summary of the total work such as the main goal, my experience, project value and lots of things. The full project will be very much benefitted for the students. It has so many details those are given below,

#### **13.1.2 Summary of the project**

By following the task, I have appropriately instigated all the project requirements and its works perfectly. I worked on this project for about six months, during this time I have implemented many things in this project. I have worked in many fields while doing this project. I had to collect a lot of in-depth data from various organizations and students and this core data is working very well in our system. For this we did a very large survey.

I have done the engineering part of this system very nicely which can be understood by looking at our documents. I have described all kinds of things in a very stunning way here such as software architecture design, literature analysis, using different methodology, diagrams etc. Here the project standards are measured by means of different categories of assessment techniques. I can say that this documentation does all categories of project stuff. I have implemented this system by succeeding those analyzes and trying to make the system which the fixe the real-world problem. And which will be of great benefit to the students in the future as well as to the benefit of various educational organizations.

### **13.1.3 Goal of the project**

I have created this web application for different students and different educational institutions. I have tried to fulfill all the requirements and I am describing the goal which I have identified in below,

- Using this system, the user can get any information about any institution.
- The previous online question bank can help the students. They can find these easily.
- Students can know the admission procedure in any school, college or university.
- Students will be able to attend various online exams like (IELTS) which they will be able to attend through the pro feature.
- It is an educational web application that will help students in a variety of ways.
- With this we have created a communication system and a system for taking feedback from the user.
- User can discuss anything in the user forum.

### **13.1.4 Success of the project**

The success of each project depends on the acceptance of its respective users and we've been very successful in bringing it to our users. I would like to say that we have done all these fields to fulfill the requirements of this project. I have met all the objectives with data from different students and educational institutions. I have developed a forum through which any student and any user can find a solution to a problem very quickly and this forum is very interactive and user friendly for users of every level. Lastly, I can say that I have been able to fulfill all the objectives in a very efficient way and that is the main success of our project.

### **13.1.5 Value of the project**

The necessities of our life are increasing day by day, and our problems are increasing day by day along with our needs. The students of A level and O level in our country

need a lot of information to get admission in different universities at some stage of their education life, not only do we have to talk to our students but the parents also need a lot of information to enroll their children in different schools and colleges. I have built-up this web application keeping their words in mind. All the information for admission of a student and a parent through the application can be found here which will be very easy for them. I have created a forum here through which any student can gain knowledge by asking questions on any of his topics. Which will bring a lot of good for our country.

## **13.2 DAFFODIL IDEA HUNT**

### **13.3.1 Strengths and weakness**

**Strengths:** My biggest strength is that my system is much more secure and it's much more user-friendly. This system does responsive behavior, and so much interactive with users.

**Weakness:** I miss out on some functions because I couldn't finish everything in a timely manner.

### **13.3.2 Future extension scopes**

This projector has a lot of potential in the future. I will be bringing many big updates for users in the future. I will make the admin panel more beautiful so that admin can maintain everything efficiently. I'll do a little bit better on the management system.

### **13.3.3 Restate contribution precisely**

With this system, the organization will be profitable because the idea of innovation is going to grow the organization. This will enhance the organization's market value. The organization's staff and student will be able to generate new ideas every day.

## **13.3 My experience**

I learned a lot during the internship that was new to me. I have worked on a total of two projectors in my internship and both are web applications. One is “**Daffodil Idea**

**Hunt**” and another is **“OAIHUB”**. I have been developed the **“Daffodil Idea Hunt”** using PHP. When I work on it, I learn a lot about lots of things, such as database design, database architecture also I've done a lot of work on how to connect the back-end with the front-end. We developed **“OAIHUB”** web application using the java Spring Boot Framework. I learn a lot of new topics, new skills I can add as my experience. The biggest thing I have learned is that if I run into a problem, I will find a way to solve it. I have become skilled at the problem solving very well. The thing that I have learned very well from here is that one has to finish a job completely despite the pressure. The requirements that I was able to fulfill in a very efficient way through teamwork. From here I learned how to manage a team and work with the team and how to solve problem by working together. I believe that this experience will be very useful in my future life.

## References

- guru99*. (2020, 5 12). Retrieved from
- keycdn*. (2020, 5 12). Retrieved from <https://www.keycdn.com/blog/performance-testing:https://www.keycdn.com/blog/performance-testing>
- scruminc*. (2019). Retrieved 2020, from <https://www.scruminc.com/what-is-timeboxing/>
- security-testing*. (2020, 5 12). Retrieved from <https://www.guru99.com/what-is-security-testing.html:https://www.guru99.com/what-is-security-testing.html>
- softwaretestingclass*. (2020, 5 12). Retrieved from <https://www.softwaretestingclass.com/what-is-module-testing-definition-and-differences/:https://www.softwaretestingclass.com/what-is-module-testing-definition-and-differences/>
- system-integration-testing*. (2020, 5 12). Retrieved from
- test-plan*. (2020, 05 12). Retrieved from [softwaretestingfundamentals.com:https://softwaretestingfundamentals.com/test-plan/](http://softwaretestingfundamentals.com:https://softwaretestingfundamentals.com/test-plan/)
- training-program*. (2020, 5 7). <https://explorance.com/>. Retrieved from <https://explorance.com/blog/5-steps-to-creating-effective-training-programs/:https://explorance.com/blog/5-steps-to-creating-effective-training-programs/>
- unit-testing-guide*. (2020, 5 12). Retrieved from

# Plagiarism Report:

7/30/2020

Turnitin

<h2>Turnitin Originality Report</h2> <p>Processed on: 30-Jul-2020 15:35 +06                  ID: 1363960022                  Word Count: 22150                  Submitted: 1</p> <p>181-16-279_Jashim_Uddin_Ahmed.pdf                  By Anonymous</p>		<table border="1"> <tr> <td>Similarity Index</td> <td><b>13%</b></td> </tr> </table>	Similarity Index	<b>13%</b>	<table border="1"> <tr> <th colspan="2">Similarity by Source</th> </tr> <tr> <td>Internet Sources:</td> <td>10%</td> </tr> <tr> <td>Publications:</td> <td>7%</td> </tr> <tr> <td>Student Papers:</td> <td>9%</td> </tr> </table>	Similarity by Source		Internet Sources:	10%	Publications:	7%	Student Papers:	9%
Similarity Index	<b>13%</b>												
Similarity by Source													
Internet Sources:	10%												
Publications:	7%												
Student Papers:	9%												

1% match (Internet from 13-Mar-2020) <a href="https://gitlab.fdmci.hva.nl/zuiderh/nopressure/commit/4e2f41281092f04fdd767de17140e0e1b778af40?w=1">https://gitlab.fdmci.hva.nl/zuiderh/nopressure/commit/4e2f41281092f04fdd767de17140e0e1b778af40?w=1</a>
1% match (publications) <a href="#">Carlo Scaroni, Massimo Nardone, "Pro Spring Security", Springer Science and Business Media LLC, 2019</a>
1% match (Internet from 31-May-2015) <a href="http://wepa.mooc.fi/index.html">http://wepa.mooc.fi/index.html</a>
1% match (publications) <a href="#">Paul Tepper Fisher, Brian D. Murphy, "Spring Persistence with Hibernate", Springer Science and Business Media LLC, 2010</a>
1% match (Internet from 31-Aug-2019) <a href="https://gitlab.fdmci.hva.nl/ewa2017/klm-team-1/commit/e8450780e935652bd19b3aa0bac76f1db7e1a1db?expanded=1">https://gitlab.fdmci.hva.nl/ewa2017/klm-team-1/commit/e8450780e935652bd19b3aa0bac76f1db7e1a1db?expanded=1</a>
< 1% match (student papers from 11-Dec-2014) <a href="#">Submitted to University of Derby on 2014-12-11</a>
< 1% match (Internet from 02-Nov-2012) <a href="http://blog.inflinx.com/">http://blog.inflinx.com/</a>
< 1% match (Internet from 14-Aug-2019) <a href="http://techiners.blogspot.com/2015/11/the-scrum-framework.html">http://techiners.blogspot.com/2015/11/the-scrum-framework.html</a>
< 1% match (publications) <a href="#">Juliana Cosmina, "Pivotal Certified Professional Core Spring 5 Developer Exam", Springer Science and Business Media LLC, 2020</a>
< 1% match (student papers from 18-Jul-2020) <a href="#">Submitted to University of Northampton on 2020-07-18</a>
< 1% match (Internet from 07-Feb-2014) <a href="http://thysmichels.com/">http://thysmichels.com/</a>
< 1% match (Internet from 09-Dec-2017) <a href="http://www.nakov.com/blog/2016/08/05/creating-a-blog-system-with-spring-mvc-thymeleaf-jpa-and-mysql/">http://www.nakov.com/blog/2016/08/05/creating-a-blog-system-with-spring-mvc-thymeleaf-jpa-and-mysql/</a>
< 1% match (student papers from 25-Apr-2018) <a href="#">Submitted to University of Hertfordshire on 2018-04-25</a>
< 1% match (Internet from 12-May-2018) <a href="https://o7planning.org/ru/10649/social-login-in-spring-mvc-with-spring-social-security">https://o7planning.org/ru/10649/social-login-in-spring-mvc-with-spring-social-security</a>
< 1% match (Internet from 29-Jun-2019) <a href="https://enhancedscrumguide.com/2016/04/">https://enhancedscrumguide.com/2016/04/</a>
< 1% match (student papers from 22-Mar-2019) <a href="#">Submitted to Oxford Brookes University on 2019-03-22</a>
< 1% match (Internet from 19-May-2014) <a href="http://www.vhuangwucha.com.cn/lovers/2013/9/23/02136.html">http://www.vhuangwucha.com.cn/lovers/2013/9/23/02136.html</a>
< 1% match (Internet from 10-Jul-2006) <a href="http://nono.niaouli.org/2006/05/10/253-hibernate-annotations-mysql-blob-largeblob">http://nono.niaouli.org/2006/05/10/253-hibernate-annotations-mysql-blob-largeblob</a>

[https://www.turnitin.com/newreport\\_printview.asp?eq=0&eb=0&esm=0&oid=1363960022&sid=0&n=0&m=2&svr=56&r=55.33598646084754&lang=en...](https://www.turnitin.com/newreport_printview.asp?eq=0&eb=0&esm=0&oid=1363960022&sid=0&n=0&m=2&svr=56&r=55.33598646084754&lang=en...) 1/32