INTERNSHIP REPORT ON

Online Academic information HUB

Submitted By

Md Sabbir Mehmud
ID: 181-16-273

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computing and Information System

Supervised By

Ms. Nayeema Rahman
Assistant Professor
Department of CIS
Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

Spring 2020

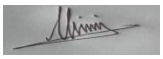
APPROVAL

This Project titled **"Online Academic Information Hub"**, Submitted by **Md Sabbir Mehmud**, ID No: **181-16-273** 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 **Internal Examiner**
Lecturer
Department of Computing & Information Systems
Faculty of Science & Information Technology
Daffodil International University

Dr. Saifuddin Md. Tareeq **External Examiner**
Professor
Department of Computer Science and Engineering
Dhaka University, Dhaka

Acknowledgement

“First and foremost, honor and thanks to the Almighty Allah for His gifts of blessings during our study work in order to successfully complete the project.

To our Project Supervisor, **Miss Nayeema Rahama** Senior Lecturer, Daffodil International University and partially **Md. Mazharul Islam**, Lecturer, Programmer, Daffodil International University, Dhaka, I would like to convey my profound and sincere appreciation for giving me the opportunity to do research and provide valuable guidance throughout this research. During the conversation I had with them on research work and project planning, I extend my heartfelt thanks to them, family for their support, and patience.

I am very much grateful to my family members for their devotion, prayer, consideration, and sacrifice to train and prepare me for my future. I am very much grateful to my other teachers and teammates for their devotion, understanding, prayers and continued support in completing this project work.”

Executive summary

The project is practically based on the total packaged information about education sector. In our country people cannot find proper information anywhere to let their child get admitted to an affordable institution which will both be good for their child and easy cost handler for the parents. Nobody is able to find any detailed information about any institution unless the specific institution is giving their information personally. And we don't know if the information is wrong or right as for business purpose everyone can talk loudly for better revenue, you know what I mean. However, this particular web application I'm working in, is going to have all specific information a guardian and student needs to decide which will be the best for him/her. So, now what exactly is in this web application. Information of every single schools, colleges, universities are included. For example, admission cost, tution fees, and all other stuffs which are needed. All the institution will have to give their information to get themselves known to people of Bangladesh. A very special discussion forum has built to discuss every aspects of educational topics of Bangladesh. All education board all times question and answer bank will be available in the web application rather than finding them into various places or books to practice for next board exam. Almost every kind of job application deadline and the information about it and the examination date can be found here. People for mock test or model test can give online exams by giving a very affordable cost for best preparation to a specific field. The answer sheet will be monitored by best teachers of specific sectors. People can judge their position of knowledge before giving their specific examination and take better preparation ahead. All kinds of questions, answers will be available on the website, almost all exam results will be published maintaining field position throughout the whole country. In one-word Students will get their every specific information from the website as the name of the web system is Online academic information hub.

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 **"DIAINTERNTEAM".** 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 beginning of the internship work I prepared several learning goals, which I wanted to learn about:

• How to understand the functioning and working conditions of the organization.

• How to explore working in a professional environment.

• How to explore the work environment for the possibility of a future career.

• How to utilize my gained skills and knowledge.

• How to find skills and knowledge I still need to work in a professional environment.

• How to acquire knowledge about software development life cycle.

• How to acquire knowledge about the development methodologies.

• How to gain fieldwork experience/collect data in an environment unknown for me.

• How to acquire experience working in multicultural and diverse environment.

• How to improve my interpersonal and technical skills.

• How to create network with professionals in the industry.

Soo 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.my task was to do following:

- Understand and working with spring boot framework.
- Understand and working with ThymeLeaf.
- Understanding Rest API.
- Understand and working with GIT.

I obtain so many new technical skills though my work. I acquire new knowledge in front end development using ThymeLeaf. I also brushed my HTML5, CSS3, BOOTSTRAP3, JavaScript, java skills while working there. 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

Chapter 1 - Introduction

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 the 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.

Overview

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.

Chapter 2 – Initial Study

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.

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.

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.

Chapter 3 – Literature Review

- **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.

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.

Comparison of three/four leading solutions-

- **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.

- **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.

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 compere 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.

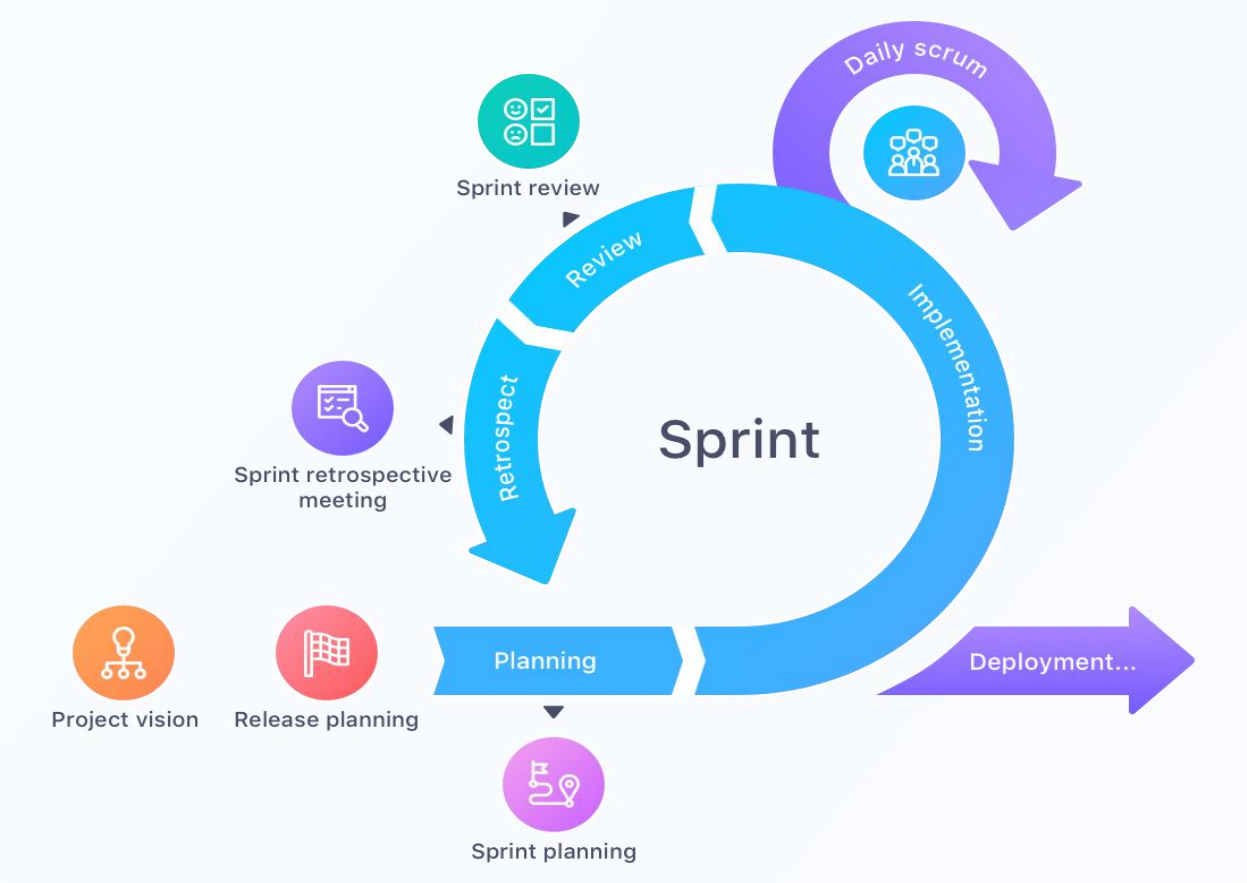
Chapter 4 – Methodology

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

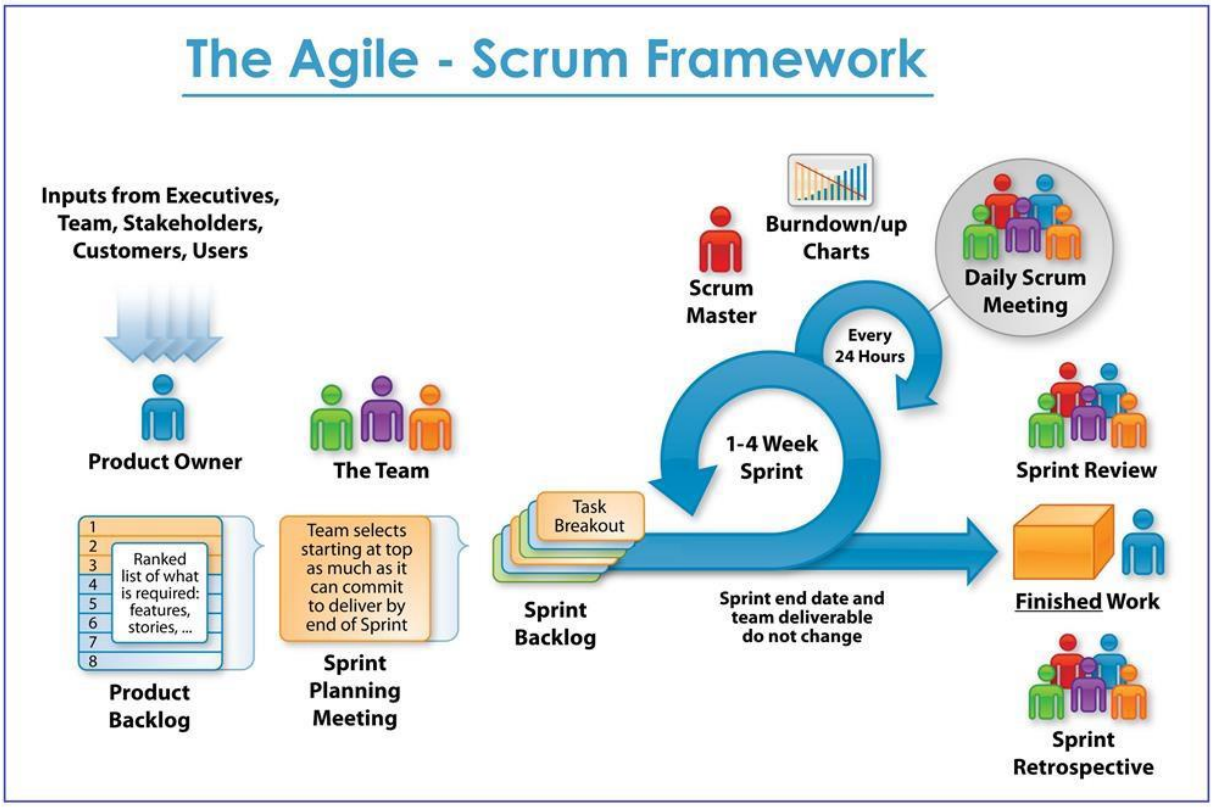
- **What to use**

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.

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.



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.



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.

Sections of methodology

There are three pillars of Scrum.

- Transparency.
- Inspection.
- Adaption.

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.

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.

Adaptation

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.

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

Project plan

The project manager would have to develop a project plan in order to bring the project to completion. The project plan outlines the project cost, magnitude and timetable. It describes exactly which activities and tasks are required and where they can be obtained, and also what resources are required from staff, equipment, and financing. Good project planning will also help to keep all stakeholders up-to - date and integrated in risk and how it should be managed, including contingency plans and a communication strategy.

- **Work Breakdown Structure (WBS)**

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.

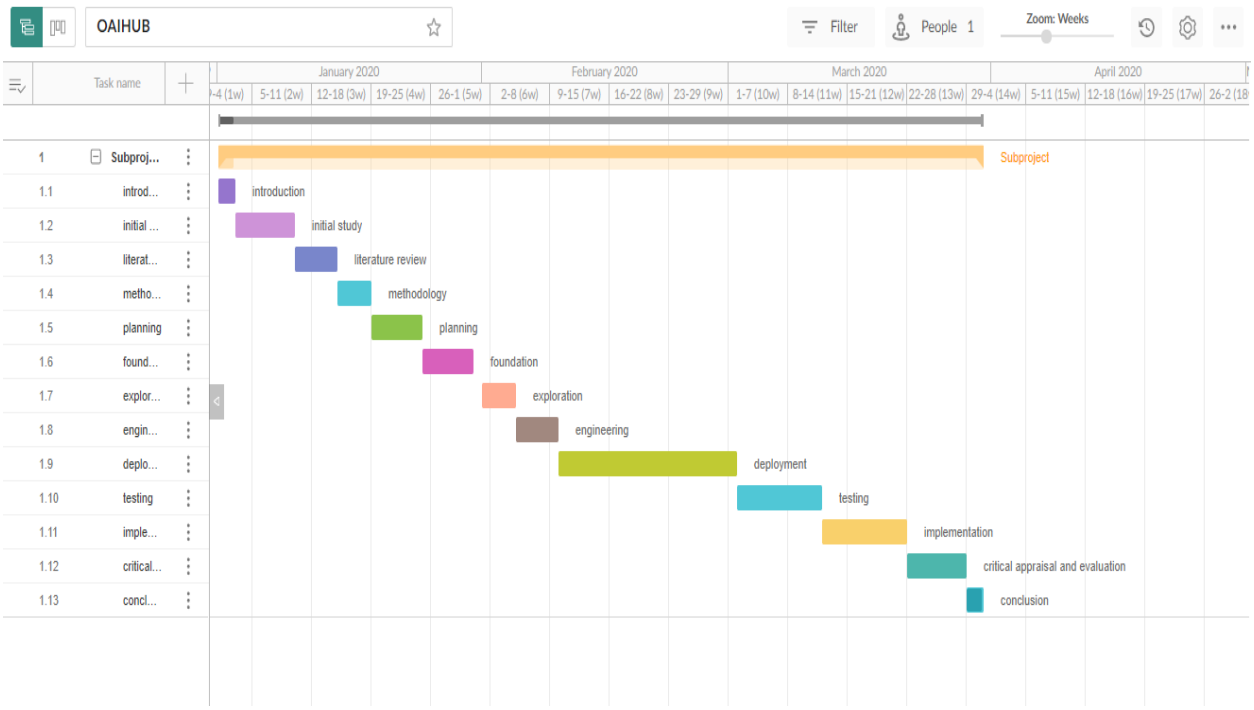
Work breakdown structure:

SL	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
Totals					**90 days**

- **Gantt Chart**

This is chart of our total activity timeline of our whole project. In this Gantt chart all of the specific individual tasks with a timeframe is presented here.



Test plan

Testing planning, the most important activity to ensure a list of tasks and milestones for monitoring project progress is initially included in the baseline plan. The size of the test effort is also defined.

It is the central document sometimes referred to as the Master Test Plan (MIT) or the project evaluation plan.

There are two types of test. Functional & nonfunctional. (Tutorials Point, p. 2019)

Functional testing:

Functional tests provide the guidance, confirmation and inspiration required by QA teams to deliver excellent software products. That is why functional testing is a practice of most successful QA teams. There are several sub types of functional testing. (Simsform, 2019)

Unit testing:

It is a software evaluating method for testing individual software units or components. The goal is to validate the output of every unit of software code. Unit Testing is carried out during a software application creation (coding phase). Tests unit isolates and tests for correctness of a segment of code. A single unit can be a process, method, procedure, module or entity. (2019, p. Guru99)

Integration Testing:

Integration tests determine if software units that have been developed independently work properly when connected. Even diffuse software industry standards have broken up the term, and I was reluctant to use it in my writing. In particular, many people assume that integration testing is of necessity widely spread, while in a smaller scope it can be done more effectively. (fowler, 2018)

Module Testing:

The lower unit of each application is a component. The Testing Component is therefore a technique that allows the smallest or lowest unit of any application to be tested. A synthesis and integration of several small individual modules may be considered as an application. It is imperious that every component OR the smallest unit of the application is thoroughly tested before we test the entire system. (help, 2018)

Non-functional:

Non-functional testing is a kind of software testing term which encompasses several production testing techniques for the evaluation and evaluation of non-functional characteristics of a software application. The main purpose of this evaluation method is to determine, in varying and improvised circumstances, the competence and efficacy of

an application. This kind of software test can be seen as a stop solution for different software issues, such as:

Performance testing:

Performance testing is a method to determine how the device responds with a specific workload in terms of reactivity and stability. Typically, performance testing is performed to check speed, robustness, reliability and size. (Neotyz, p. 2019)

Security testing:

Standard functional testing ensures that the software works accordingly. This allows our customers to assure that their software complies with a list of specifications and requirements. Security tests are a logical extension of negative tests: they concentrate on undesirable inputs and the likelihood of substantial failure of these inputs in relation to the particular specifications of the product being evaluated.

Test case:

A Test case is a series of conditions or variables under which a test system is expected to satisfy or operate correctly. The method of designing test cases can also help to identify issues with program specifications or design. (2019)

Test: 1		Test Class:	Designed By	
Data Source:		Objective:	Tester:	
Test Case	Description	Tasks	Expected Result	Actual Result
1.1		:		

Figure: A test case template

User acceptance test plan:

One of the last steps in software development is to test user acceptance. Once it is released, it's the big check. Sometimes the testing for user acceptance is called "beta testing." You probably know there is more than beta testing for UAT, as we showed you in this article on the 5 types of UAT. All in all, UAT concerns the user and whether the user is working on a certain product or service.

Test Priority:	**Test Execute by:**
Unit test No: 01	**Test Execute Date:**
Test case	
Objective:	
Data Source:	

Case No.	Description	Tasks	Result		Status (Pass/Fail)
			Actual result	Expected result	
1					

Figure: User acceptance test plan template.

Chapter 6 – Foundation

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 retable to specific topics
- ✓ Model tests will be examined and moderated via top notch teachers.
- ✓ Specific thread publisher names will be visual individually
- ✓ Specific comment publisher names will be viewed
- ✓ Every thread will be viewed by time and date.
- ✓ Comments under a thread will be viewed by date and time. Important file will be able to uploaded through users, pro users in the threads e.g. snapshots, code snippets, word docks, 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.

- **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:

Technical Languages

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

Databases Systems

MySQL, Tomcat, JDBC,

Technologies

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

Framework

Spring, Thyme Leaf

Build tool

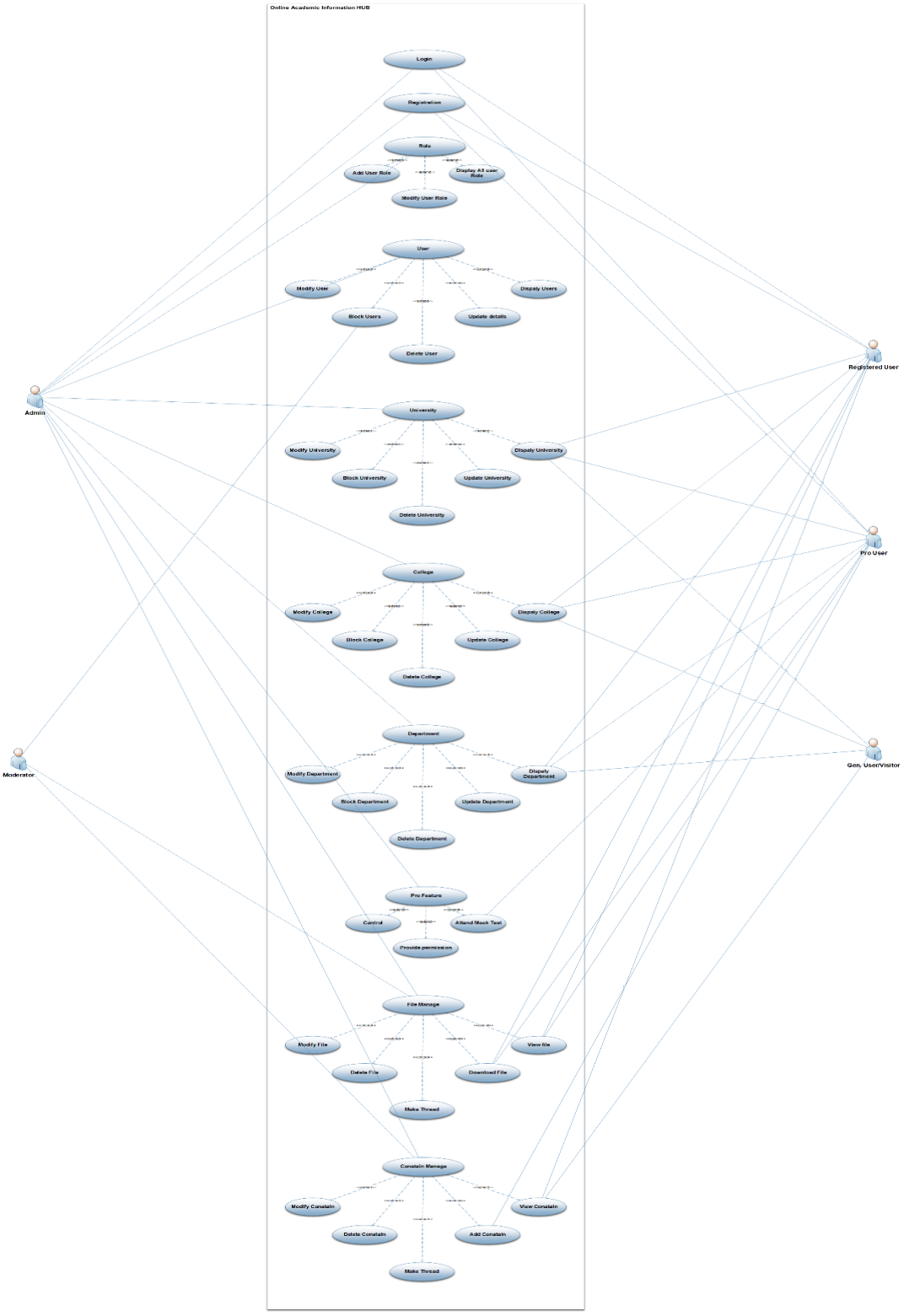
Maven, Gradle

Server Platforms

Daffodil Web Server Storage

Chapter 7 - Exploration

➢ 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:

1. **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 handle 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.

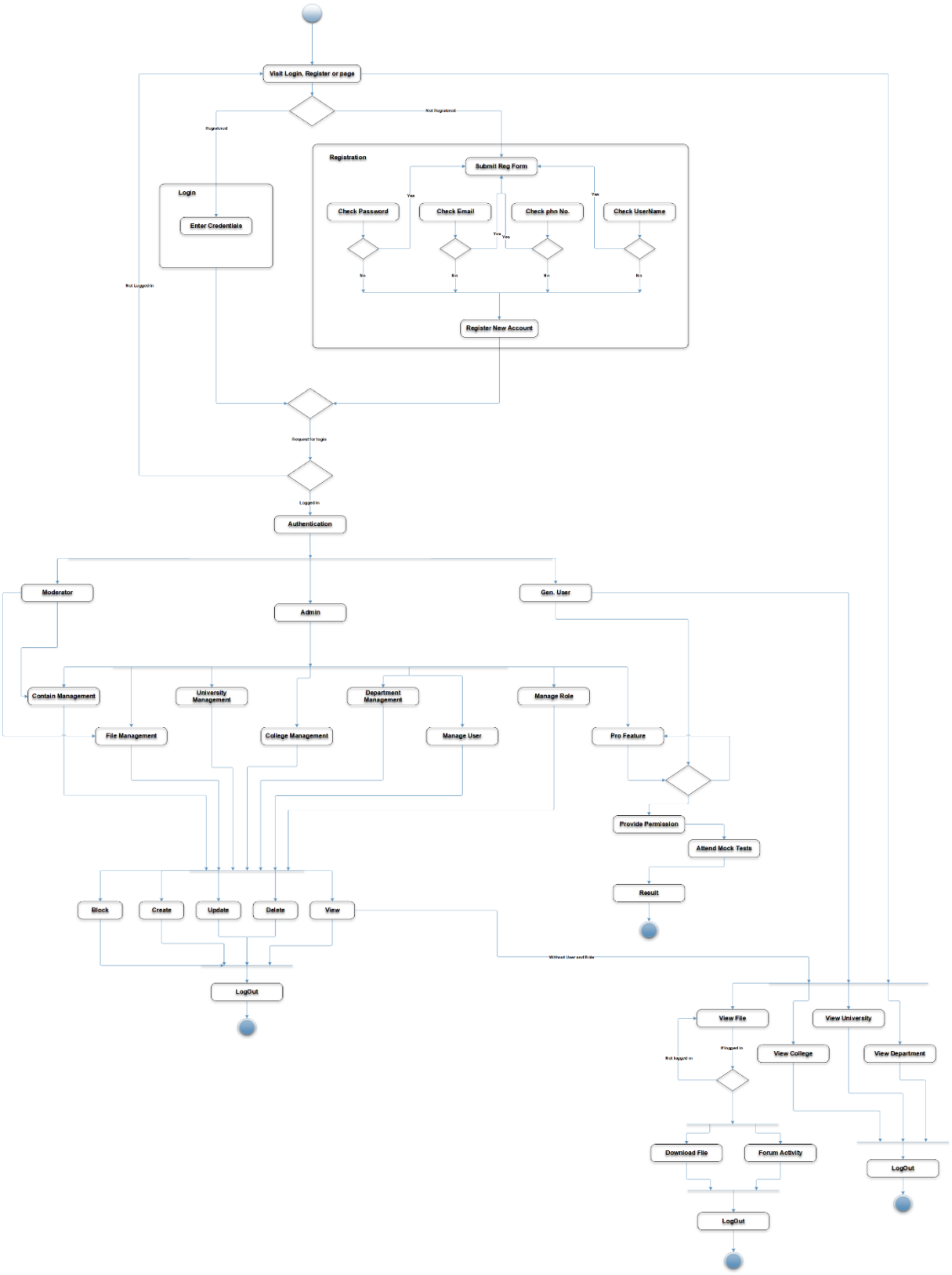
2. **Moderator**: This panel has less power and access then the admin. Modify user, block user, delete user, update user data those also can be accessible by moderator panel. Necessary file can be also handled by the moderator panel. Modifying files, delete files, make thread, download file and view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

3. **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.

4. **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.

5. **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.

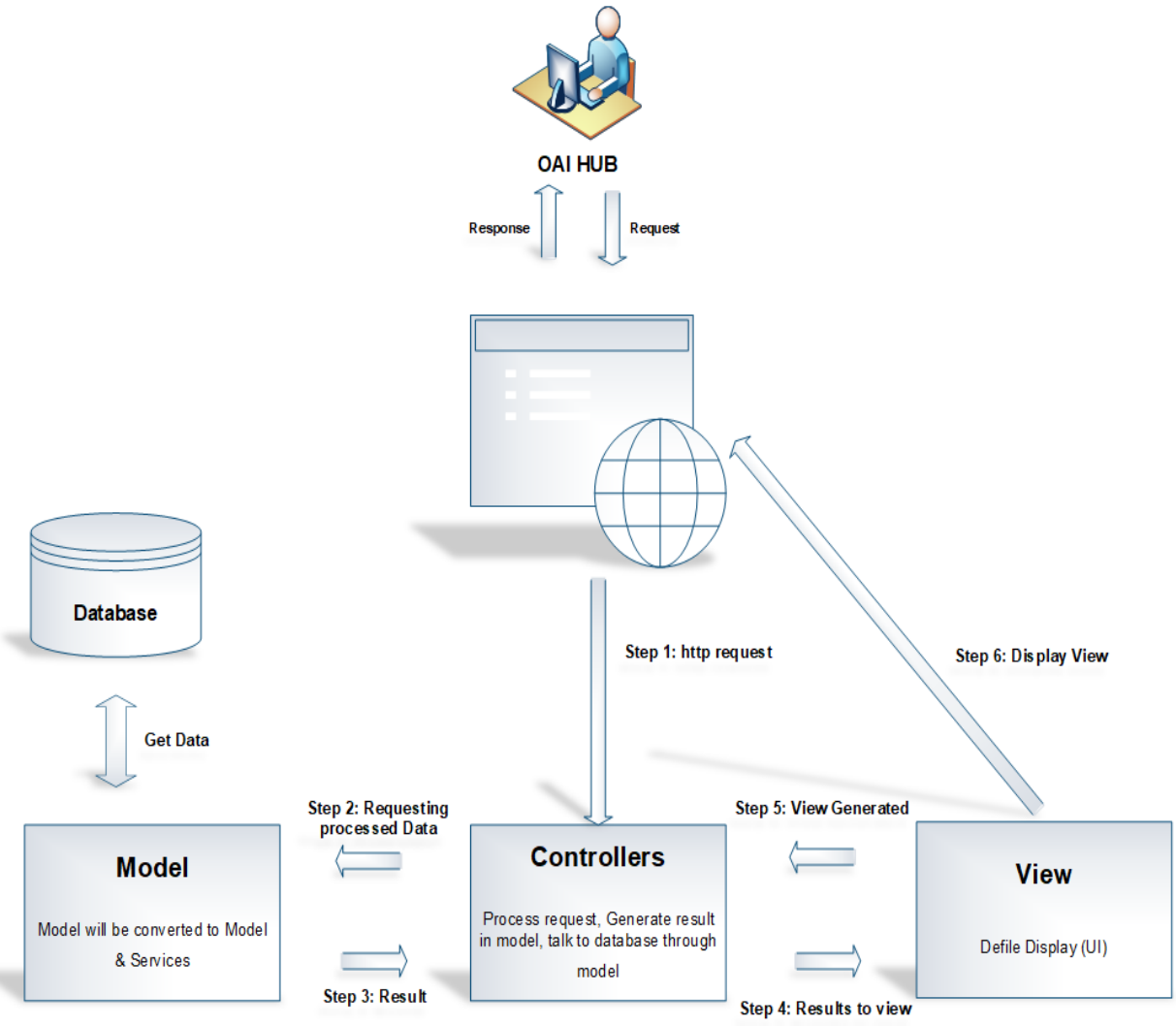
➢ Old Full System Activity Diagram



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.

- **Prototype of new system**



Architectural Design – MVC Architectural

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

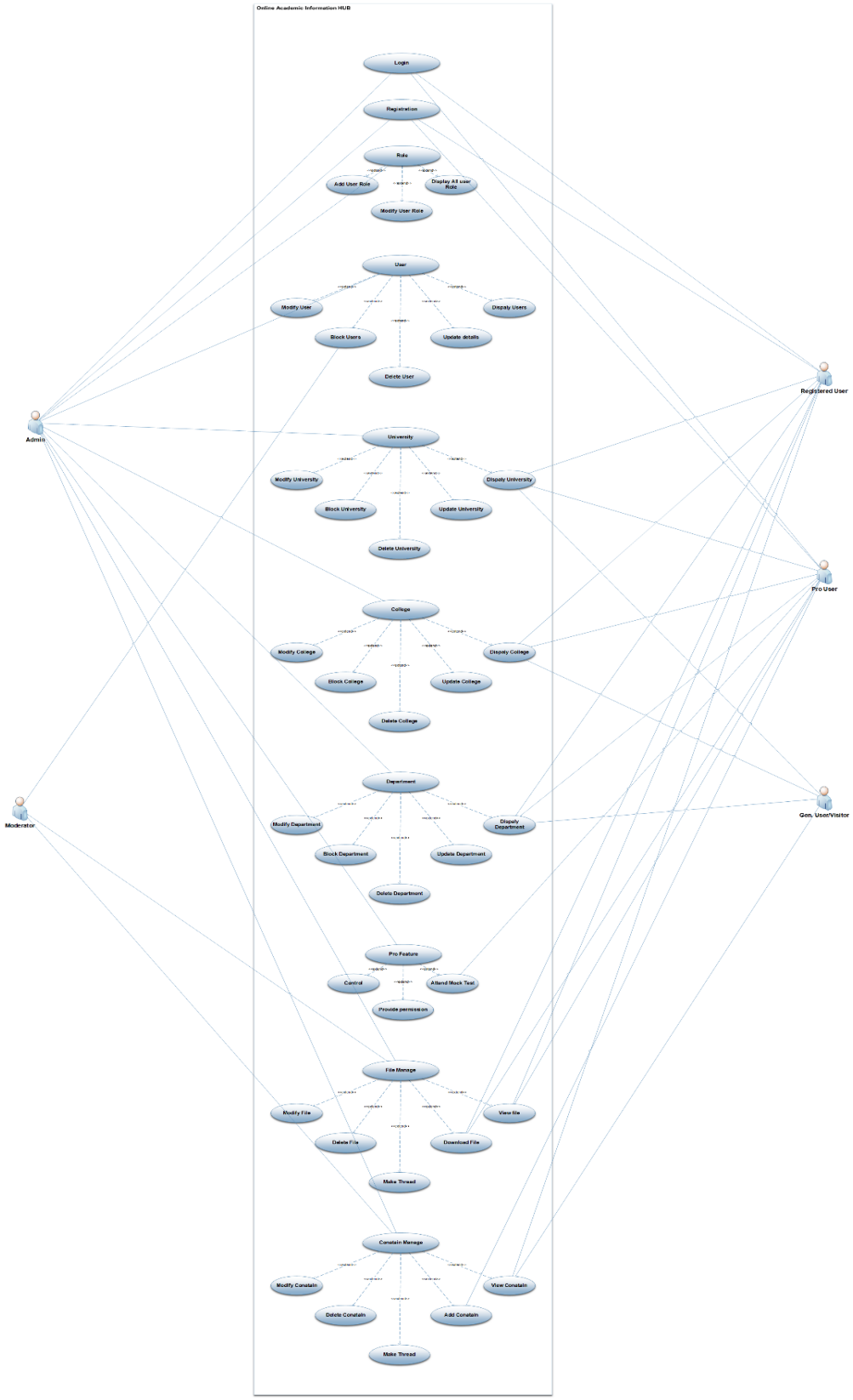
user. That request and processing are the backend task. If any user make interaction with UI or make some input then the UI send those inputted data to the controller and controller send them to the model for analysis and arrange and check validation of those data then model send them to database to 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.

Chapter 8 - Engineering

- **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.

➢ New Use Case


Online Academic Information HUB
Login
Registration
Role
Add User Role
Display All user Role
Modify User Role
User
Modify User
Dispaly Users
Block Users
Update details
Delete User
University
Modify University
Dispaly University
Block University
Update University
Delete University
College
Modify College
Dispaly College
Block College
Update College
Delete College
Department
Modify Department
Dispaly Department
Block Department
Update Department
Delete Department
Pro Feature
Control
Attend Mock Test
Provide permission
File Manage
Modify File
View file
Delete File
Download File
Make Thread
Constain Manage
Modify Constain
View Constain
Delete Constain
Add Constain
Make Thread
Admin
Moderator
Registered User
Pro User
Gen. User/Visitor

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:

1. **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.

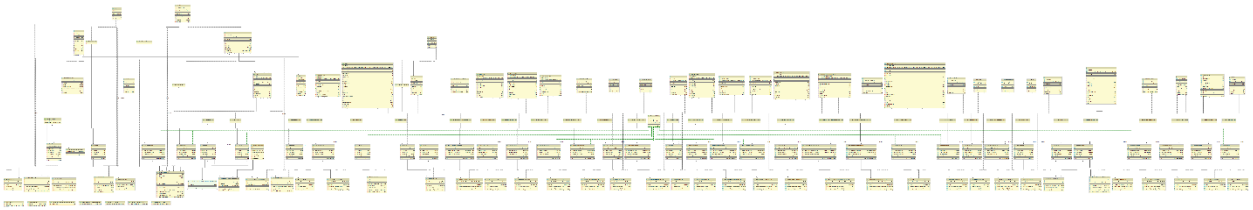
2. **Moderator**: This panel has less power and access then the admin. Modify user, block user, delete user, update user data those also can be accessible by moderator panel. Necessary file can be also handled by the moderator panel. Modifying files, delete files, make thread, download file and view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

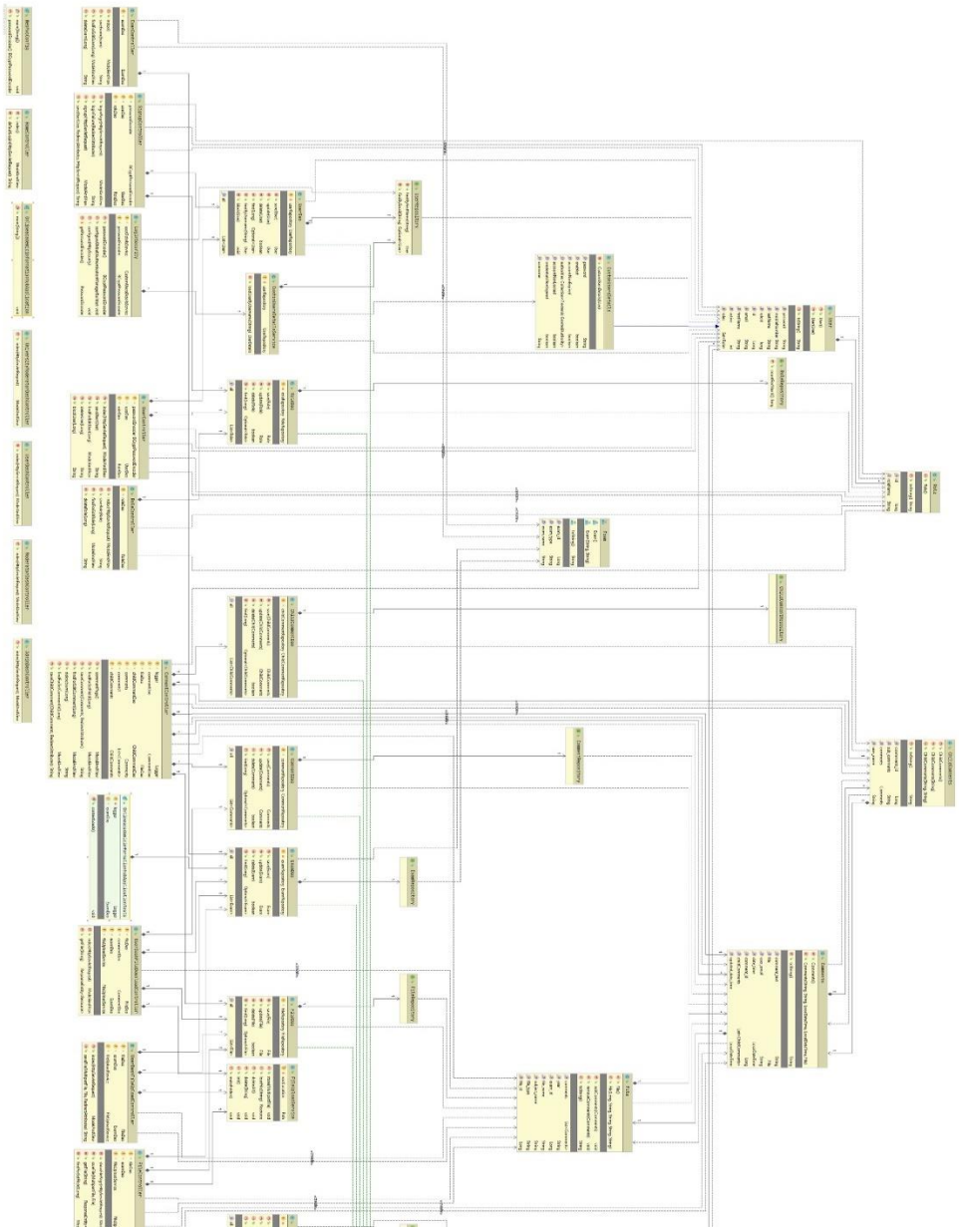
3. **Register user**: This panel they can login and register in the beginning. They can see university, college and department data. From this panel they can also attend mock test. They will have access to view file and download them. They can able to add content and also view others.

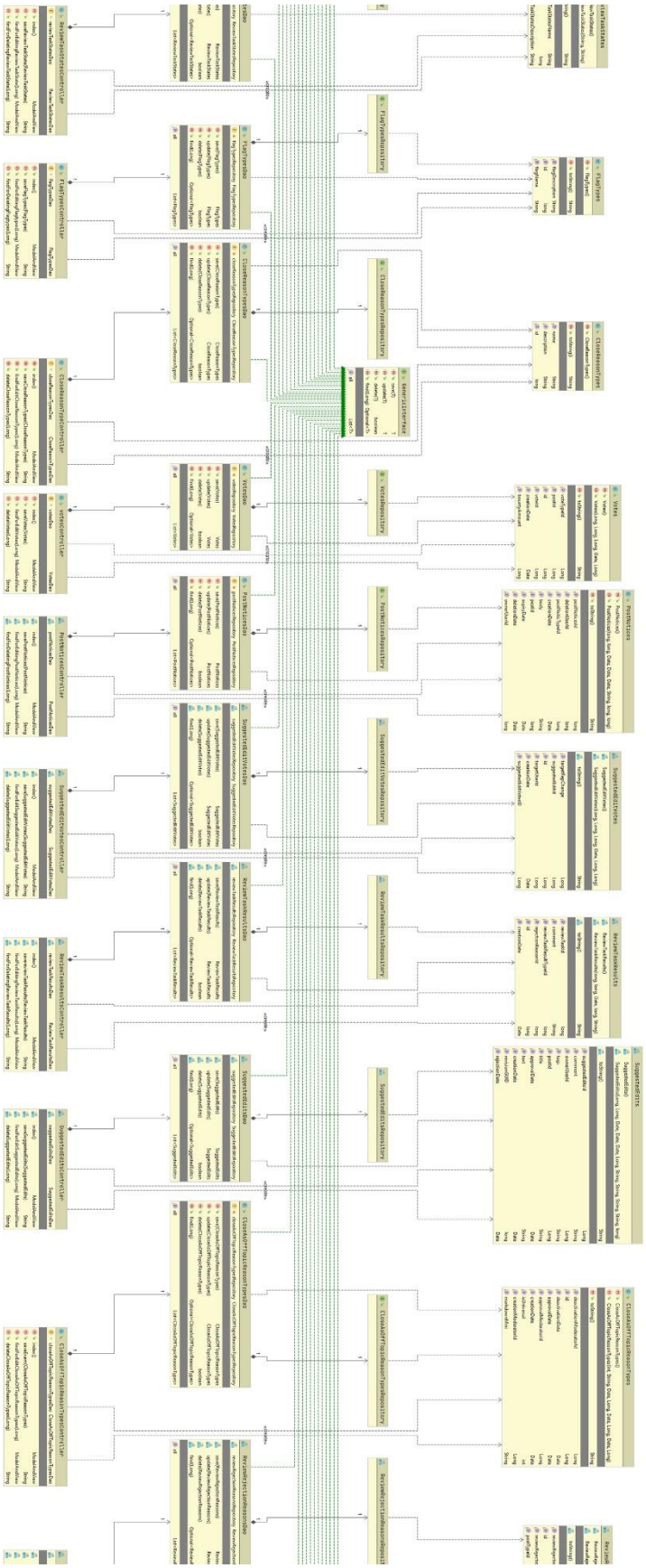
4. **Pro-user**: This panel members are the special then register member. This panel member can login and register in the beginning. They can view university, college and department all data. From this panel they can attend mock test. They will have access to view file and download them. They can also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

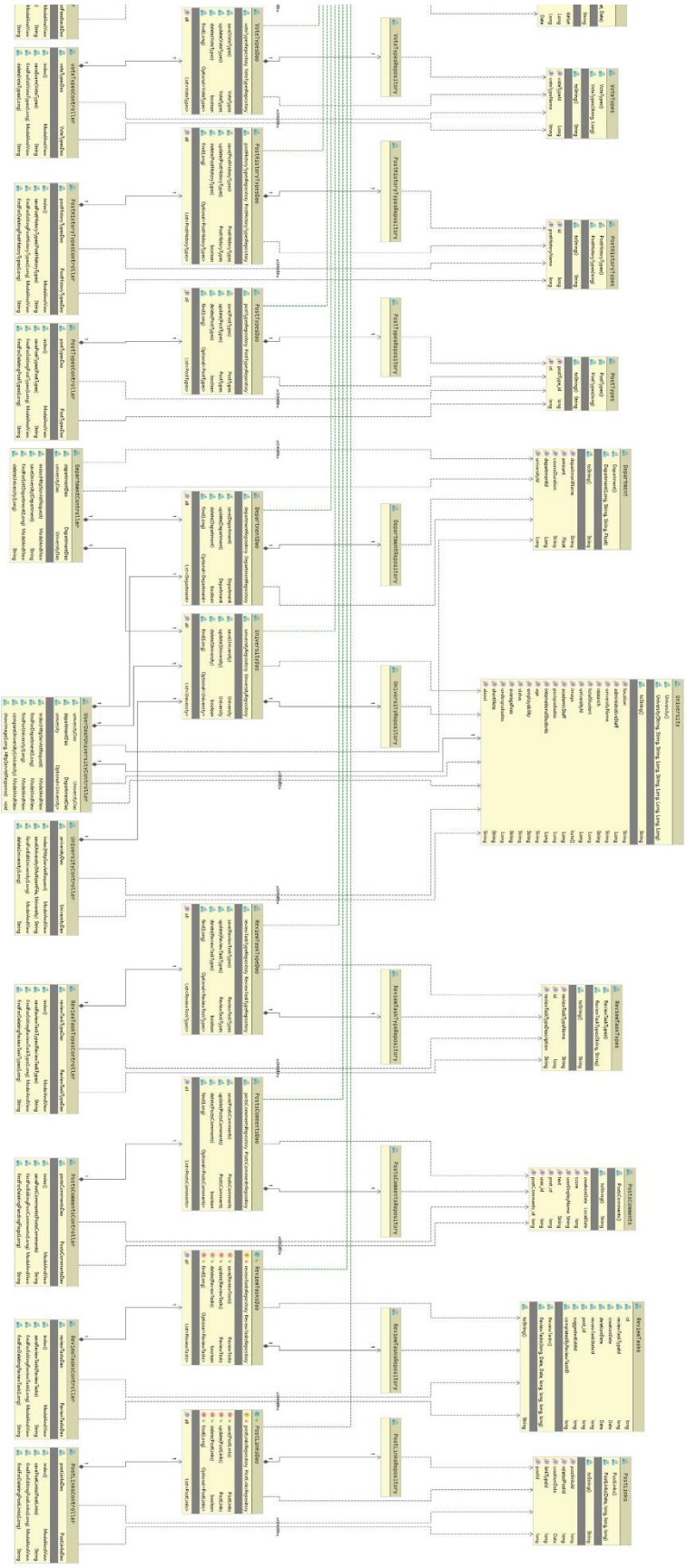
5. **General user/visitor**: They have the less ability in this system. Visitor can register themselves. Then they can do many things. Without registration they can only view can view university, college, department data and some content.

- **New 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.

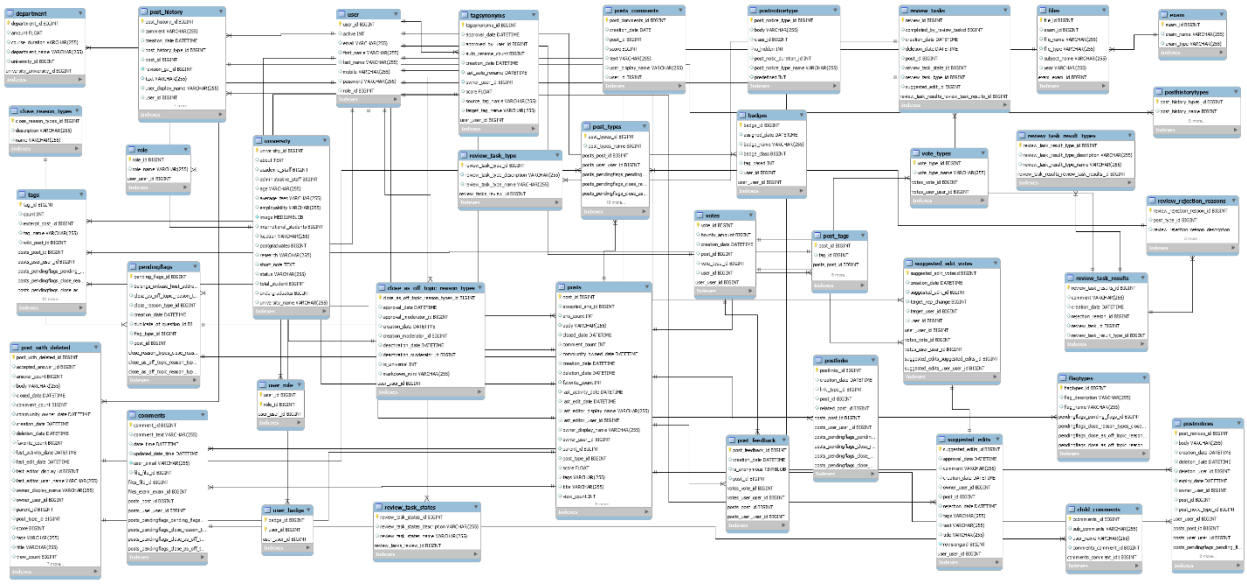
User management system has the accessibility and security layer for every type of user. Admin, moderator, university moderator, user, pro user. All these users have different part of accessibility in the system. Admin has control over the whole system, moderator has access to the part which university 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. Their will be a pro feature item in this web application. In this feature there will be thousand types of model tests to be given for getting prepared for the desired destination. Various types of exams, time duration, exam time schedule, school-college-university admission tests and many more can be given online. Various types of respective teachers from different institutions will be moderating the answer sheets. Pro features will be accessible after paying a short amount of cost.

- **New Entity Relationship Diagram**

The ERD Diagram of the whole system is given below:



Description:

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties. By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases.ER diagrams are used to sketch out the design of a database. Entity user has user_id BIGINT(20) primary key. Child_comments has comments id BIGINT(20) primary key.user role hase two primary key as user id and role id BIGINT(20).persistent logins has series primary key. university has university id. Department, user organization and finance all have their individual id as primary key. Role has role id as primary key. File has file id. Exam has exam id as primary

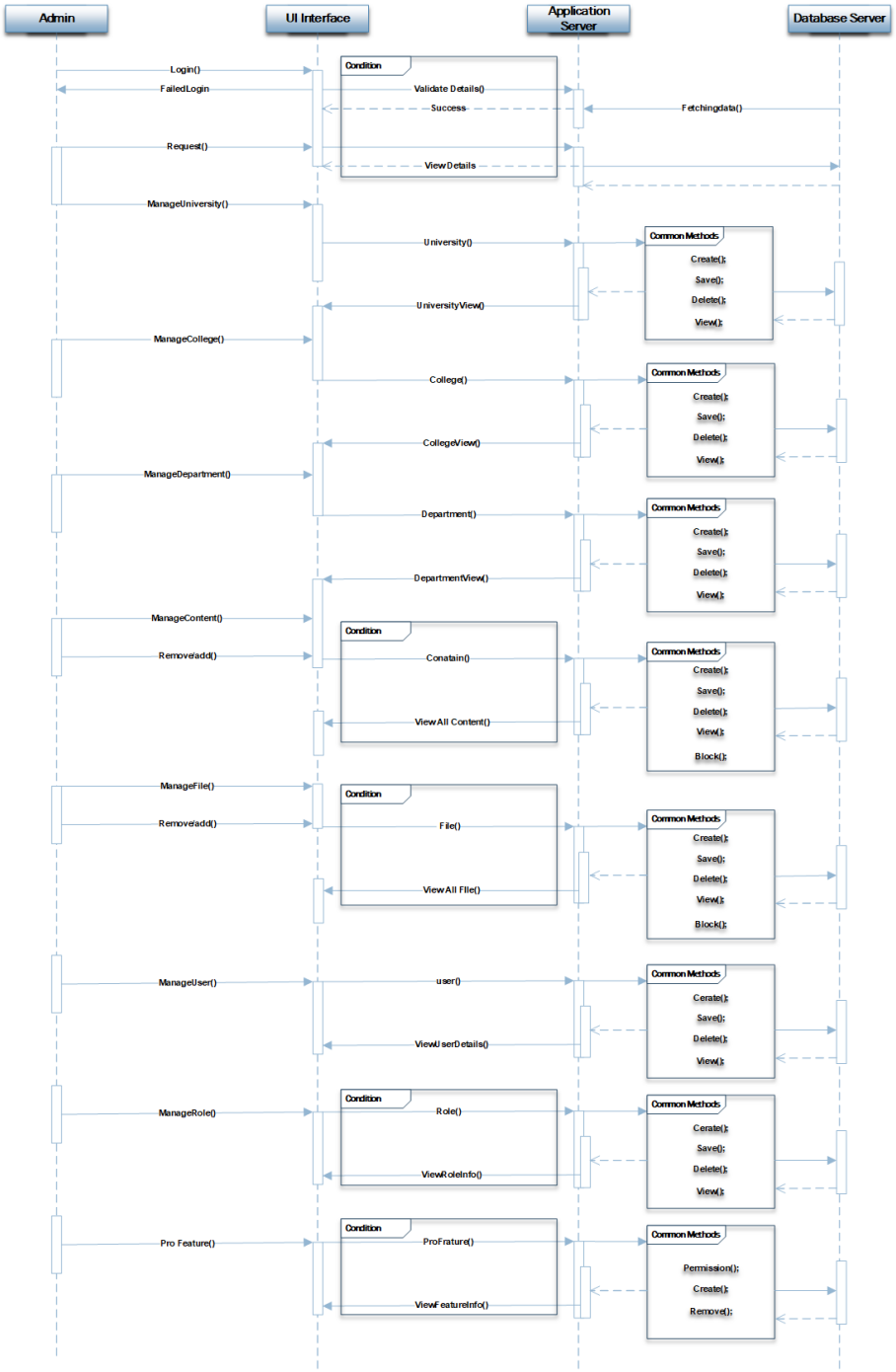
id. Password reset token has also id has primarily. Organization entity has org id as primery key. Here entity User has many to 1 relation between entity user_role. Entity user has many to 1 relation between entity role. Entity user role has many to 1 relation to entity password_reset_token. Entity hibernate_sequence hase 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 relation to entity files. Entity files has 1 to many relation to exams. Entity University has 1 to many relationship between entity department. Financial entity has 1 to many relation between organization.. entity organization has 1 to many relation 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. Its 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 stablished. 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.

- **New Sequence Diagram**

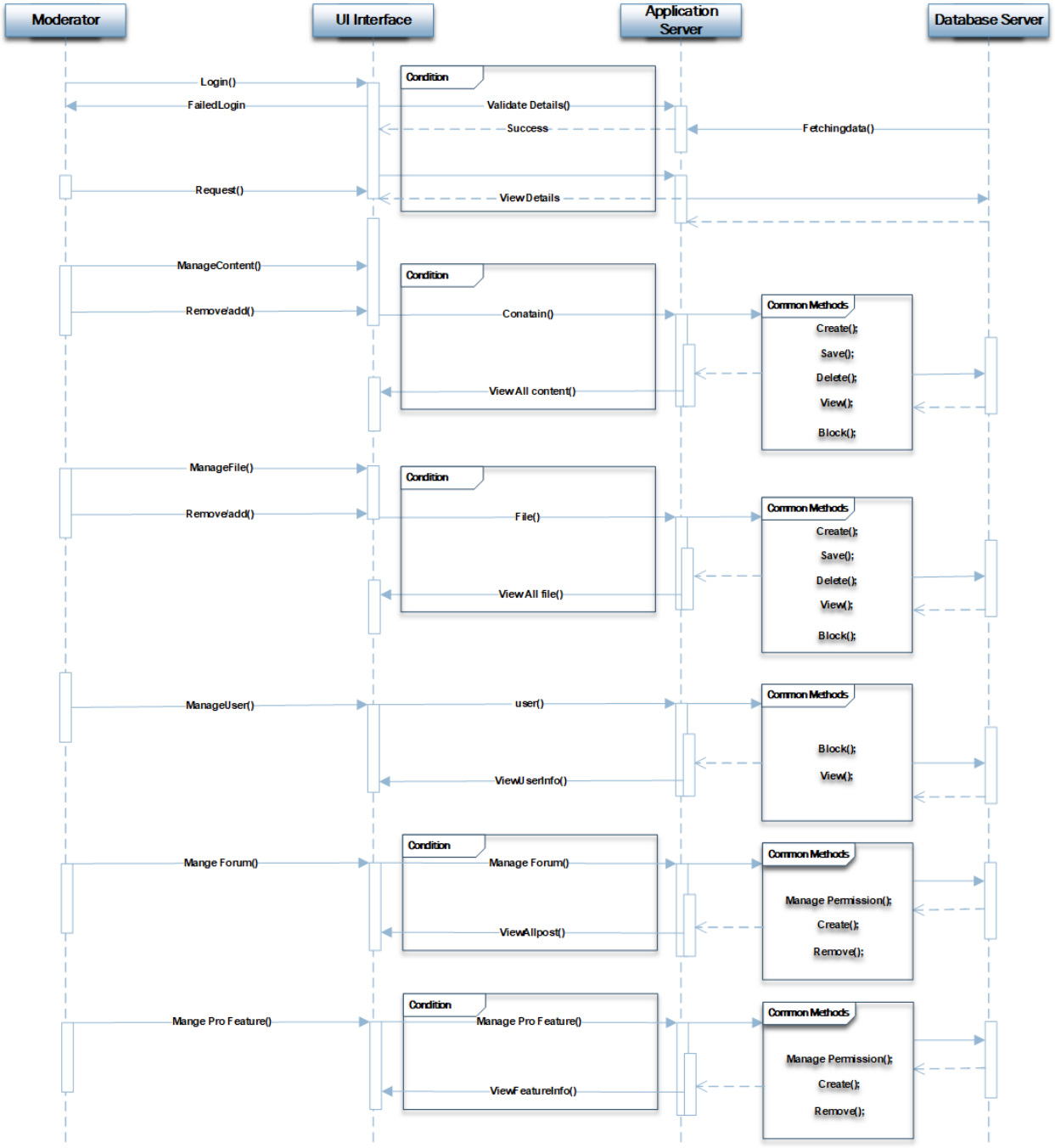
Sequence diagram: As an admin



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.

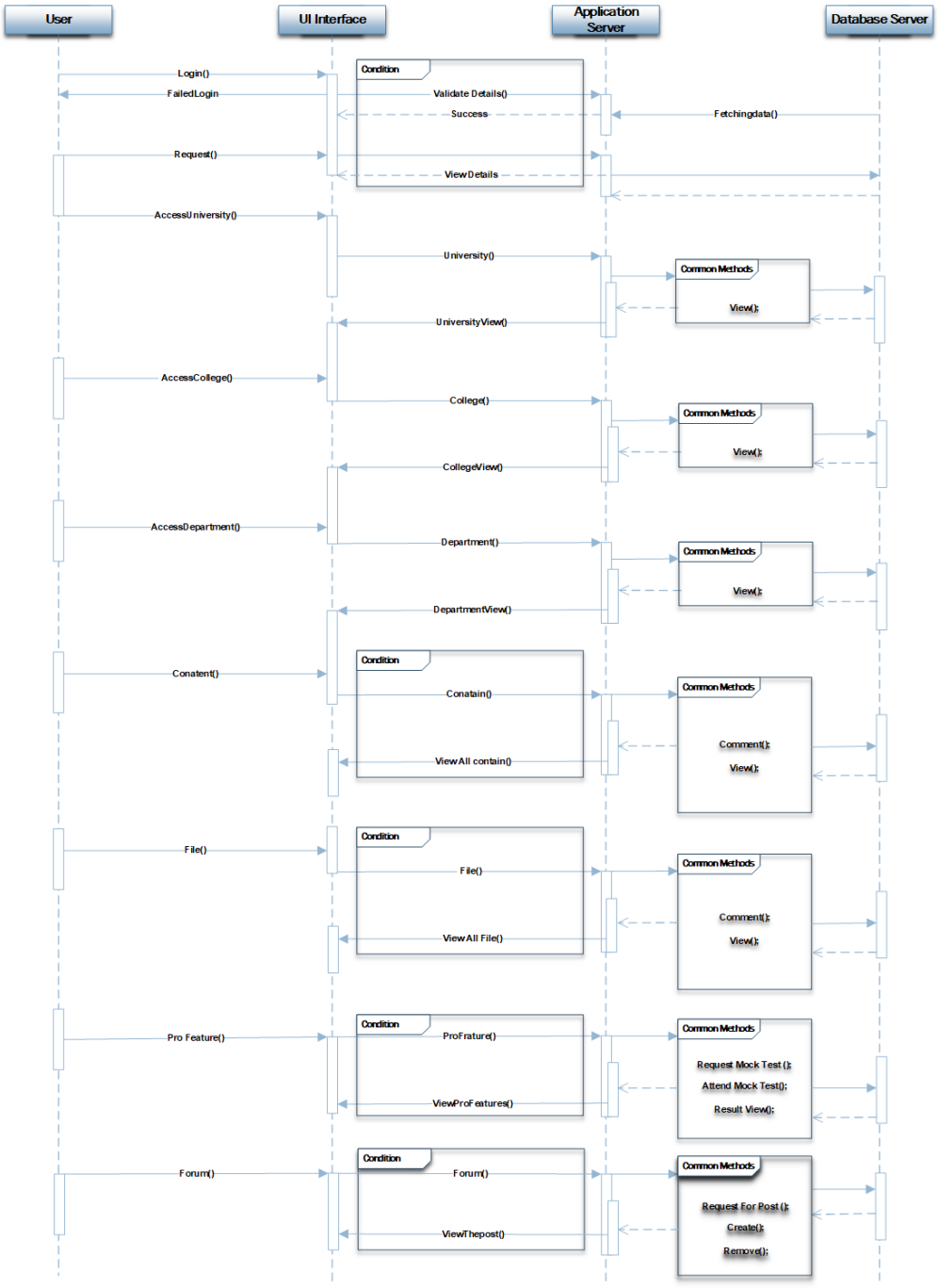
As a moderator:



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.

As a 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. In the forum section All users are permitted to throw discussions individually and others are permitted to comment on those.

Chapter 9 - Deployment / Development

- **Core Module Coding Samples**
- **User Management**

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

1.1 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 +
                '}';
    }
}
```

1.2 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.**

```java
@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);
```

1.3 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> {
}
```

1.4 Role Class

- **Role will be added by this class code objects**

```java
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{" +
```

1.5 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);
    }
}
```

1.6 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();

}
```

1.7 Customer User Details Service 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();
    }
}
```

1.8 User Dash Controller 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;
    }
}
```

1.9 User Controller Class

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

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

    @Autowired
    UserDao userDao;

    @Autowired
    RoleDao roleDao;
```

- **method for fetching user & role information**

```java
    @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**

```java
   @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";
    }
}
```

1.10 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";
        }
    }
}
```

1.11 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";
    }
}
```

1.12 Home Controller 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";
    }
}
```

1.13 Admin Dash Controller 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;
    }
}
```

Configuration

1.14 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;
        }
      };
    }
}
```

1.15 Web MVC Config 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"));
    }

}
```

Generic Interface

- **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);
}
```

- **Exam Management**

2.1 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 + '\'' +
                '}';
    }
}
```

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> {
}
```

2.2 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);
    }
}
```

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";
    }
}
```

2.3 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 +
                '}';
    }
}
```

2.4 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 + '\'' +
                ", user_name='" + user_name + '\'' +
                ", comments=" + comments +
                '}';
    }
}
```

2.5 Comment Repository

- **Spring Boot 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> {
}
```

2.6 Child Comments Repository

- **Spring Boot 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> {
}
```

2.7 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);
    }
}
```

2.8 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);
    }
}
```

2.9 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";
    }
}
```

- **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

```
  @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.save(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

```java
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 + '\'' +
                ", creationDate=" + creationDate +
                ", userDisplayName='" + userDisplayName + '\'' +
                ", 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> {
}
```

- **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 + '\'' +
            '}';
}
}
```

Chapter 10 - Testing

11.1 Unit Testing

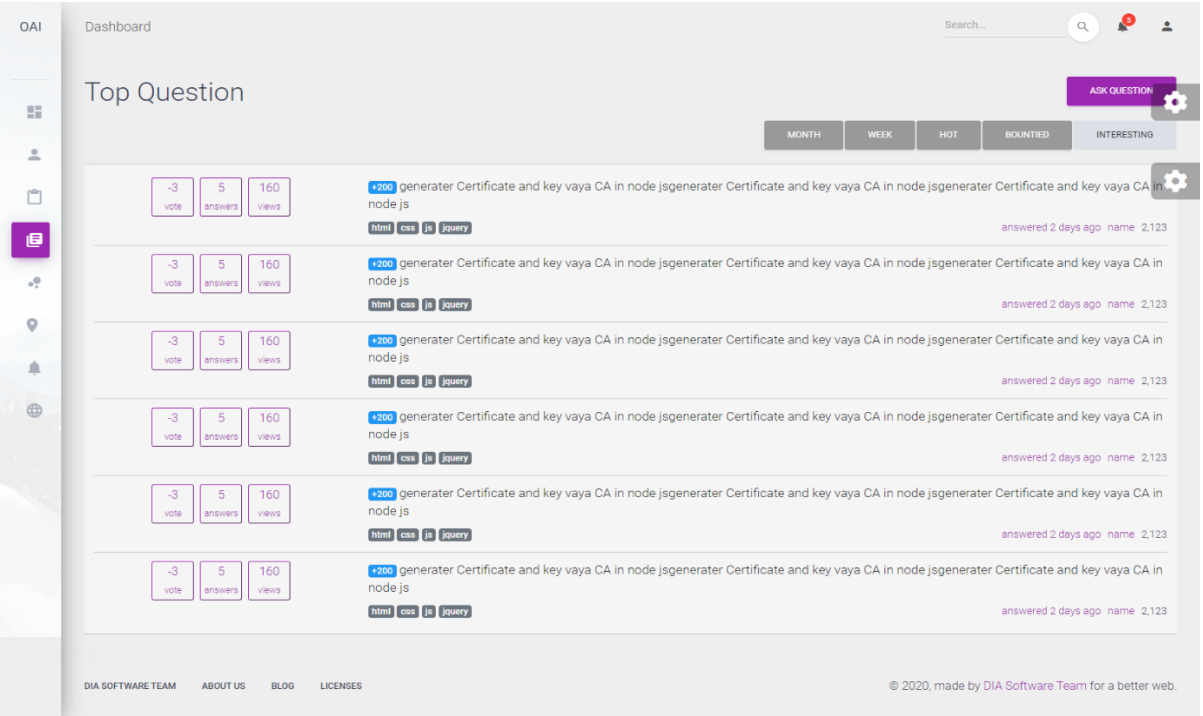
Forum Section

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

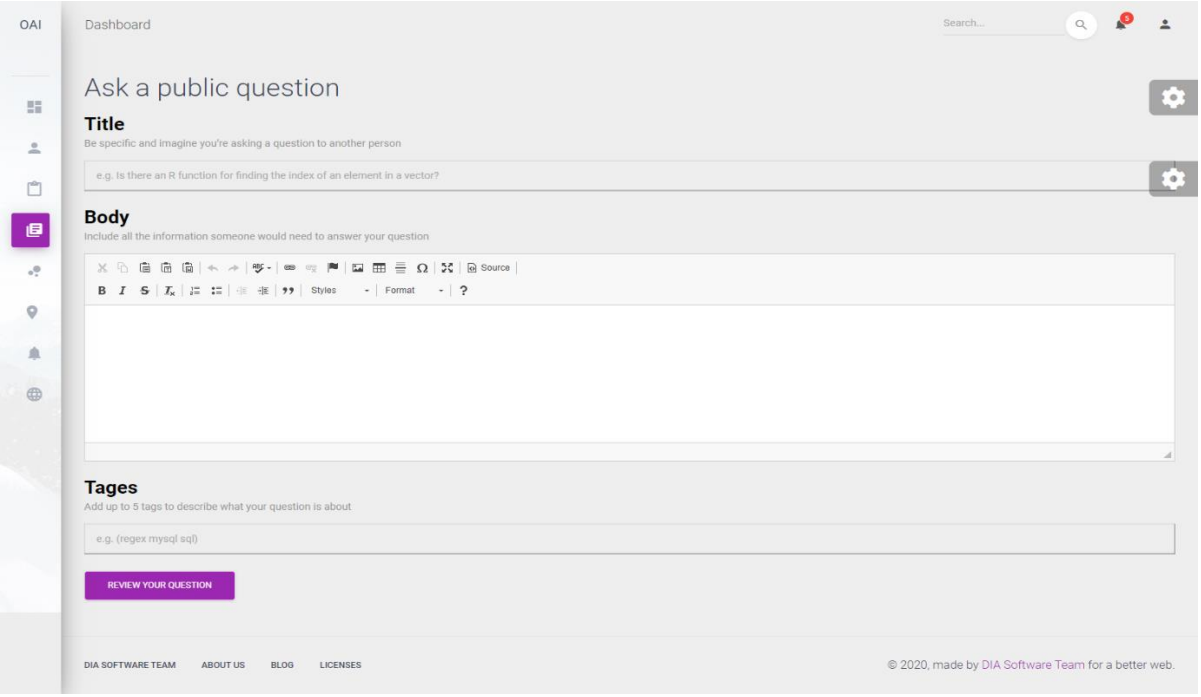
Test case: Public post checking

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

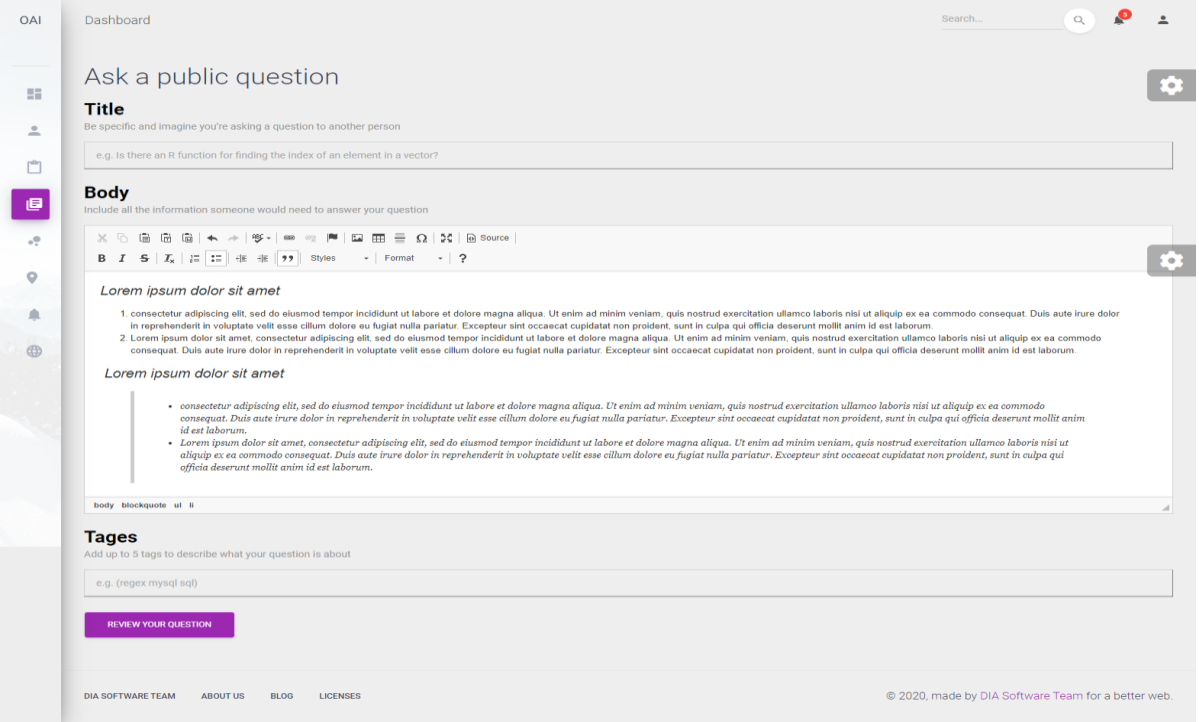
The test result screenshot for Unit Test is given below



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 king of data	Text editor work properly	Text editor will be work properly	Pass

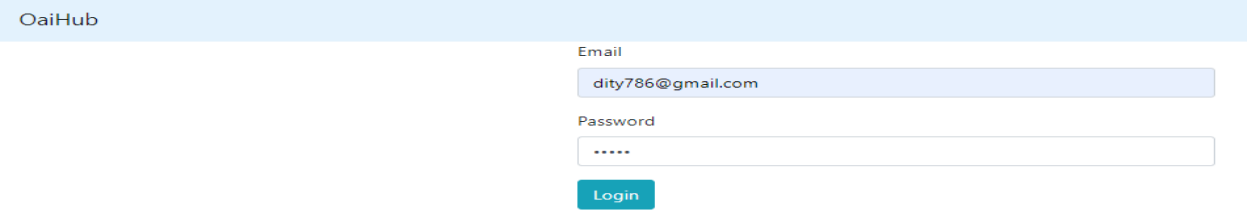
The test result screenshot for Unit Test is given below

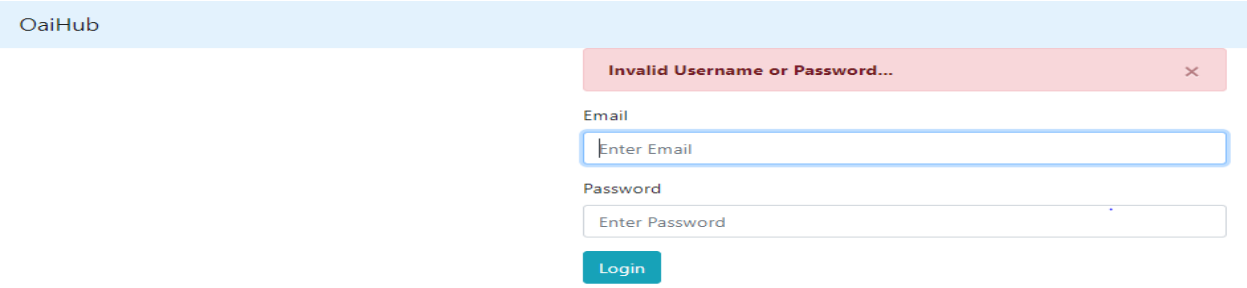


User Management Section

Unit Test: 1		**Test Class:** User Login Window	**Designed By**	
Data Source: User input		**Objective:** Password checking test.	**Tester:** Diainternteam	
Test Case	**Description**	**Tasks**	**Expected Result**	**Actual Result**
1.1	Test for checking wrong password.	Enter login Info, User Email: Password:	Show message “Invalid username or Password”	Perfect

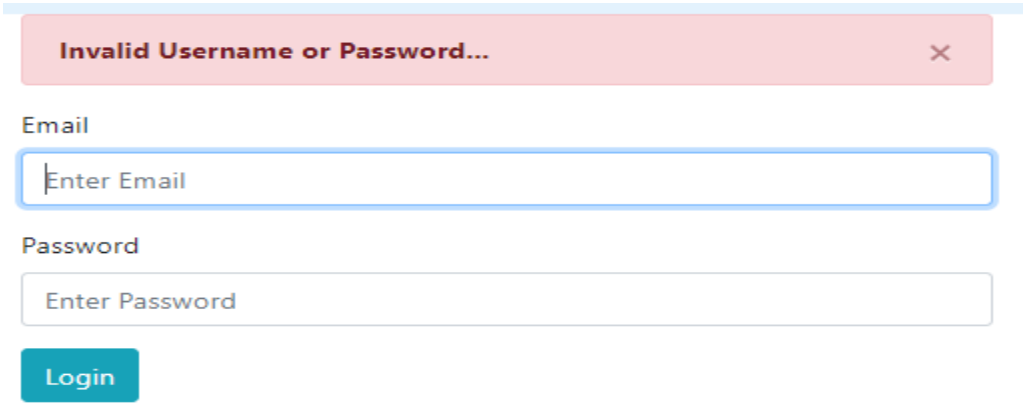
The test result screenshot for Unit Test 1 is given below,





Unit Test: 2		**Test Class:** Admin Login Window	**Designed By**	
Data Source: Admin input		**Objective:** Password checking test.	**Tester:** Diainternteam	
Test Case	**Description**	**Tasks**	**Expected Result**	**Actual Result**
1.1	Test for checking wrong password.	Enter login Info, Admin Email: Password:	Show message “Invalid username or Password”	Perfect

The test result screenshot for Unit Test 2 is given below,



11.2 Integration Testing

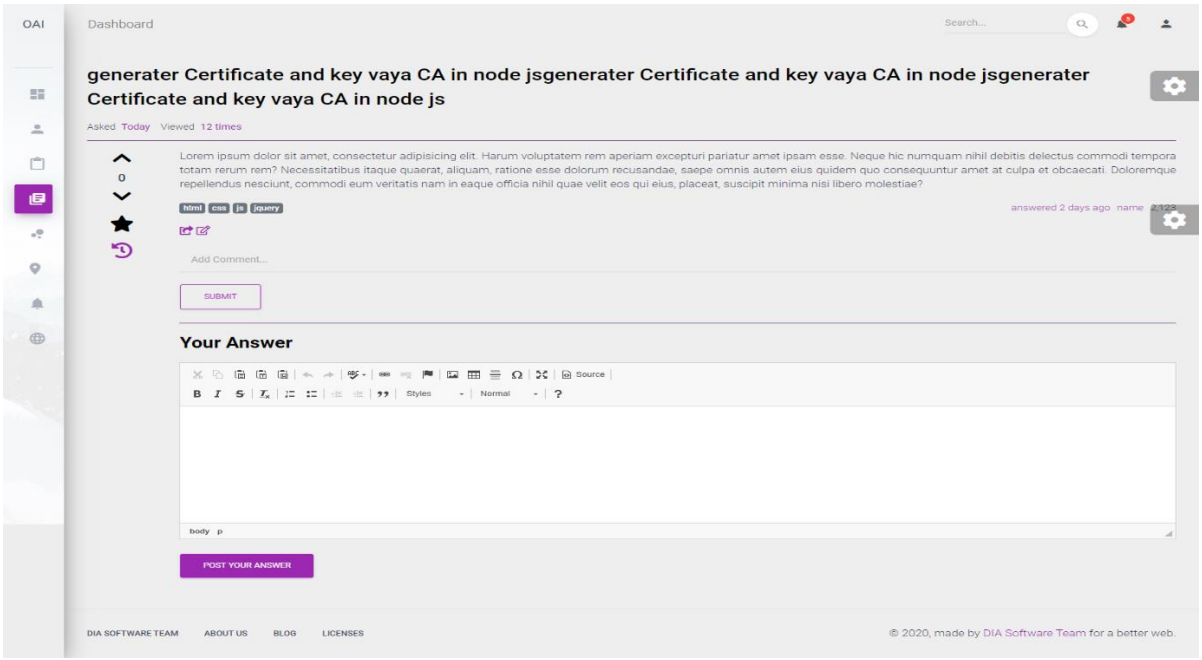
Forum Section

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

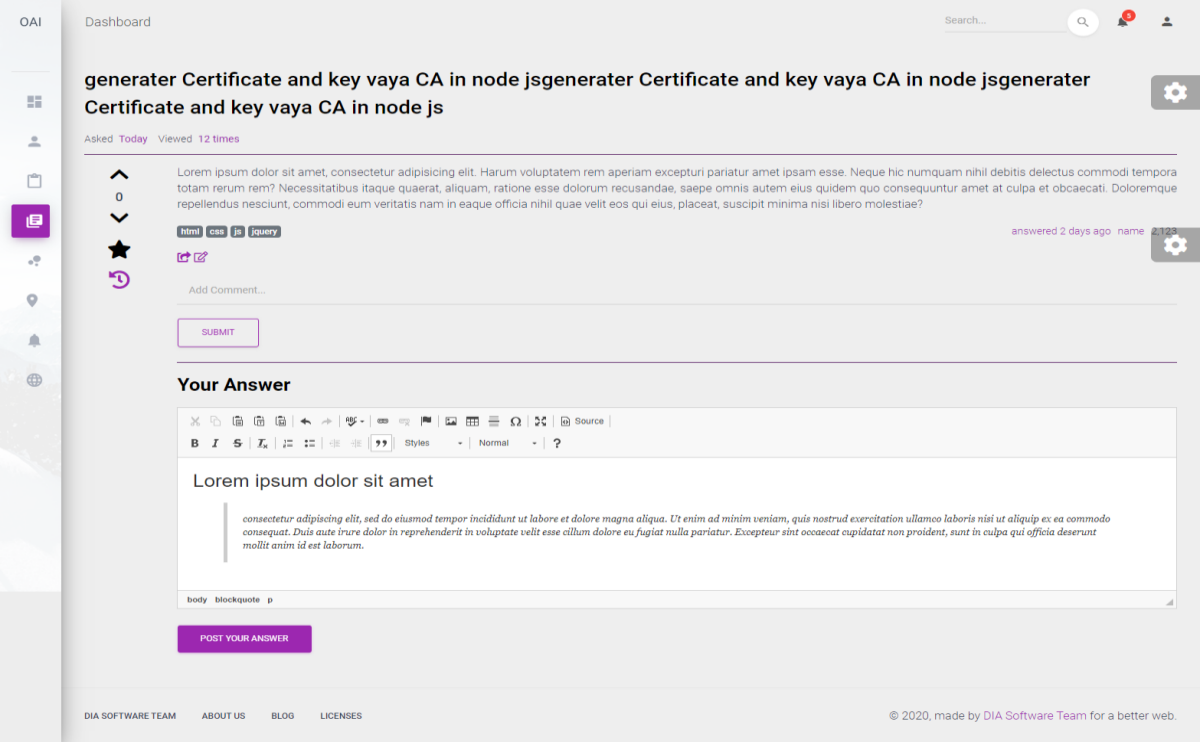


Test Priority: High	**Test Execute by:** DIA Intern Team
Integration test No: 02	**Test Execute Date:** 10/05/2020
Test case: Submitted the Ans	
Objective: Check the Ans submission	
Data Source: User input	

Description	Tasks	Result	

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



Test Priority: High **Test Execute by:** DIA Intern Team

Integration test No: 03 **Test Execute Date:** 10/05/2020

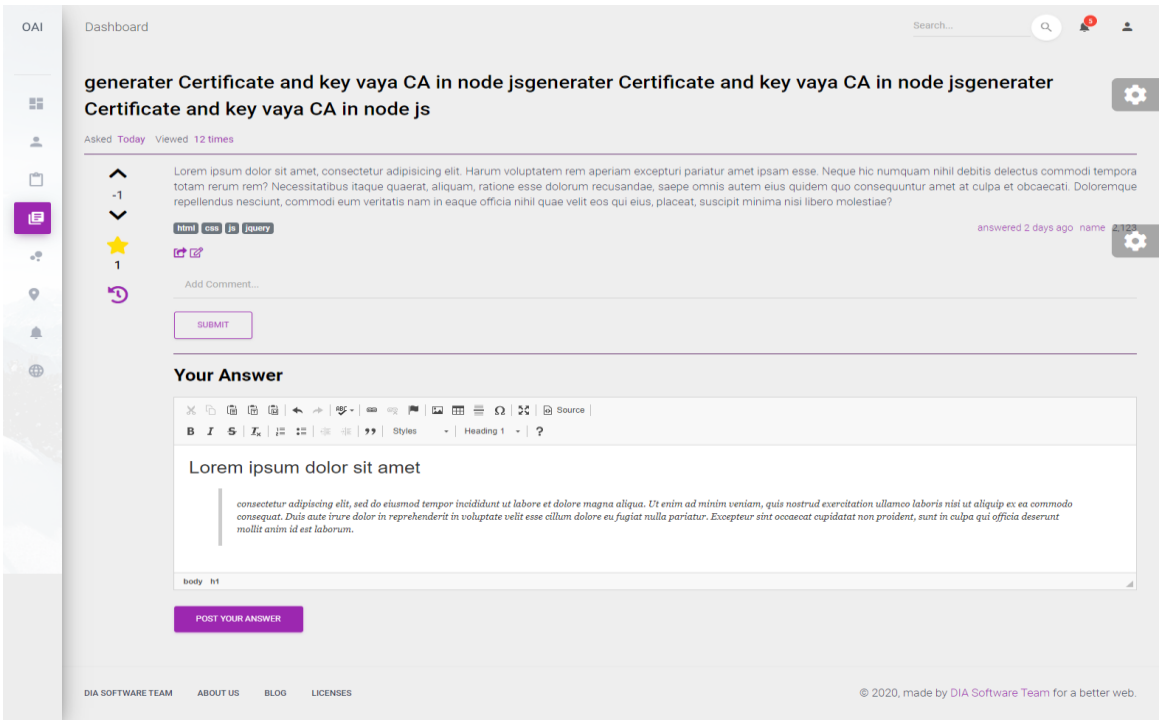
Test case: Bookmark and voting

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

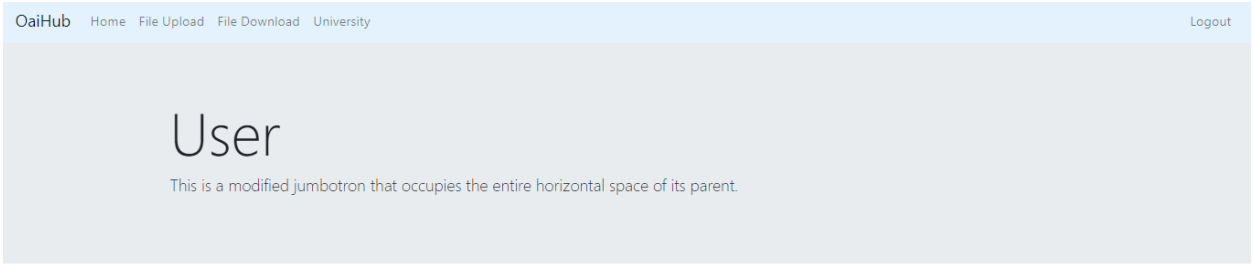
The test result screenshot for Integration Test is given below



User Management Section

Integration Test: 1		**Test Class:** User	**Designed By**	
Data Source: User input		**Objective:** Test for basic functionality.	**Tester:** Diainternteam	
Test Case	**Description**	**Tasks**	**Expected Result**	**Actual Result**
1.1	Test for basic function for user login	Enter Email: sabbir@g.com Enter password: 12345678	Show the user login page	Perfect

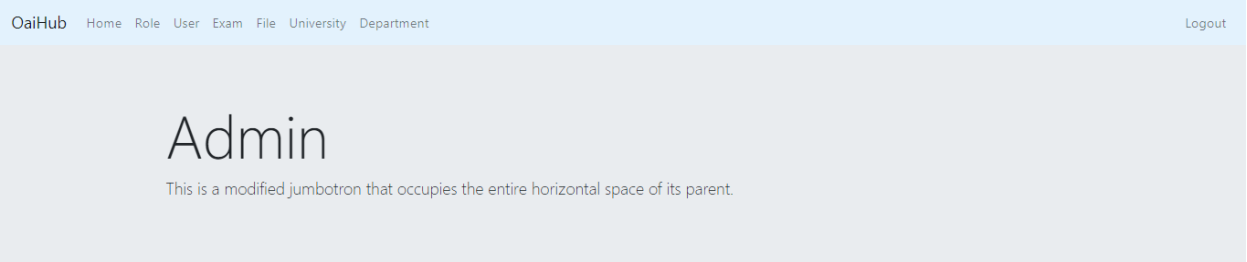
The test result screenshot for Integration Test 1 is given below,



Integration Test: 2		**Test Class:** Admin	**Designed By**	
Data Source: Admin input		**Objective:** Test for basic functionality.	**Tester:** Diainternteam	
Test Case	**Description**	**Tasks**	**Expected Result**	**Actual Result**

1.1	Test for basic function for admin login	Enter Email: dsa@dsa.ds Enter password: 12345678	Show the admin login page	Perfect

The test result screenshot for Integration Test 1 is given below,



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.

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.

- **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.

- **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.

- **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.

- **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.

- **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)

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.

Objective that could be met

The project proposed has met some objectives outlined below,

1. Implementing a platform for all student and guardian can get any information about any institution.
2. This application provides previous question bank with solutions, which can help the students and they can find these easily.
3. This tis the mature platform for both students and the educational organization.
4. Fix a methodology suitable for implementing a system.
5. Make the project well documented with maintenance the standard.
6. The application has to error free as much as possible.
7. Messaging and commenting system were created for the user (student and guardian) to organization communication.
8. We are making this application’s database which is based on this project requirement.
9. This application is web-based application which is portable easy to use from anywhere.
10. We are making this application with met the requirement which is given by our moderator.
11. We have made this system useful for different types of user and that will be very helpful for the user.

- **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.

- **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.

- **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.

Objectives totally not met / touched

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.

- **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.

- **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 lucking'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.

Chapter 13 - Conclusion

Conclusion

We have tried our best to build this 'OAIHUB' project successfully. The full report summary includes the goals and accomplishments in this section. The implementation knowledge and project values are also specified in this section. Over the course of our growth we have had many challenges and our mentor helps us solve this. For the short period of time the project will continue to develop, many more features are not yet done. I outlined here my summary of the total work, including the main objective, my experience, the value of my project and many things. To the students, the whole project is very helpful. There are so many descriptions below.

Summary of the project

I have adequately stimulated all the requirements of the project and its work. I have been working on this project for nearly six months, I have done several things in this project during this period. During this project, I worked in many places. I have had to collect many in-depth data from different organizations and students and these core data work in our system very well. We made an enormous survey of this.

We have done a very nice job of designing this device that is clear when analyzing our papers. We have explained it in a wonderful way, such as software architecture design, literature analysis, different processes, diagrams etc. We explained it. The project standards are measured in this respect through various categories of evaluation techniques. I can guarantee that all types of project material are included in this text. By following these analyzes and attempting to make the system correct the real problem, I have implemented this system. And this will benefit the future students as well as various educational opportunities.

Goal of the project

For various students and various educational institutions, we have created this web application. We tried to meet all the requirements and I explain the target I set out below,

- The user can obtain some knowledge about any organization by using this program.

- Students can help with the previous online question balance. You will quickly locate them.
- The admission procedure can be known to students at any school, university or school.
- Students can take different online exams such as (IELTS), which they can participate through the professional function.
- It is a software application for educational purposes that assists students in various ways.
- We have developed a communications system and a user feedback system with this.
- In the user forum the user can chat about everything.

Success of the project

Every project's success depends on the approval of its individual users and we have succeeded greatly in bringing it to our users. I wish to say that the requirements of this project are met in all these fields. With input from different students and educational institutions, we have achieved all the objectives. We have built an interactive and user-friendly forum through which both student and user can easily find a solution to the issue. Finally, I can say that we have succeeded in very efficiently achieving all the goals, which is our main project success.

Value of the project

The needs of our lives are rising every day and our issues are rising with our needs every day. The A and O students in our country need plenty of information in order to obtain admission at various universities at some point in their educational life, not only are we obliged to speak to our students but also the parents we need plenty of information in order to enroll their children in different schools. Our web-built application takes your words into account. Here you can find all the information you can receive for admission of a student and parent via the procedure. Here we have built a forum from which every student can learn from any of his topics by asking questions. That's going to do our country a lot of success.

My experience

I learnt a lot that was new to me during the internship. In my internship and this is a web application, I have been working on a huge projector. The application name is "OAIHUB." I've also done a lot of work in connection with the background when I worked on it, such as the database design, database architecture. Using the Java Spring Boot Framework, we have developed "OAIHUB" web application. I am learning many new issues, new skills that I can add. The best thing I've known, I'll find a way to fix it if I face a problem. I am very well qualified to solve the problem. What I know very well is that, given the strain, one has to complete a job absolutely. The criteria that I have accomplished by teamwork in a very efficient manner. I have learned from here how to manage a team and work with the team and solve problems through cooperation. I think this experience in my future life will be very useful.

Bibliography

(2019). Retrieved from http://softwaretestingfundamentals.com/test-case/

99, G. (2019). Retrieved from https://www.guru99.com/unit-testing-guide.html

fowler, M. (2018). Retrieved from https://martinfowler.com/bliki/IntegrationTest.html

help, S. t. (2018). Retrieved from https://www.softwaretestinghelp.com/what-is-component-testing-or-module-testing/

Neotyz. (201). Retrieved from https://www.neotys.com/insights/performance-testing

Simsform. (2019). Retrieved from https://www.simform.com/functional-testing/

Tutorials Point. (n.d.). Retrieved from https://www.tutorialspoint.com/software_testing_dictionary/test_plan.htm

Turnitin Originality Report

Processed on: 30-Jul-2020 15:34 +06
ID: 1363960011
Word Count: 18347
Submitted: 1

181-16-273_Md_Sabbir_Mehmud.pdf By Anonymous

Similarity Index	Similarity by Source	
18%	Internet Sources:	13%
	Publications:	9%
	Student Papers:	13%

2% match (student papers from 15-Feb-2013)
Submitted to Cardiff University on 2013-02-15

1% match (Internet from 11-May-2020)
https://labinfo.ing.he-arc.ch/gitlab/luca.verardo/MentorArcMirror/-/commit/ecfbc496e1835d4a98ec1060b76ed96c3e812079?w=1

1% match (publications)
Carlo Scarioni, Massimo Nardone. "Pro Spring Security", Springer Science and Business Media LLC, 2019

1% match (Internet from 29-Jun-2019)
https://enhancedscrumguide.com/2016/04/

1% match (publications)
Iuliana Cosmina, Rob Harrop, Chris Schaefer, Clarence Ho. "Pro Spring 5", Springer Science and Business Media LLC, 2017

1% match (Internet from 08-Jun-2018)
http://repository.stcloudstate.edu/csit_etds/22/

< 1% match (Internet from 31-May-2015)
http://wepa.mooc.fi/index.html

< 1% match (student papers from 18-Jul-2020)
Submitted to University of Northampton on 2020-07-18

< 1% match (Internet from 09-Nov-2018)
https://www.axiomatics.com/blog/pep-sdk-spring-security/

< 1% match (Internet from 02-Nov-2012)
http://blog.inflinx.com/

< 1% match (Internet from 07-Feb-2014)
http://thysmichels.com/

< 1% match (student papers from 11-Dec-2014)
Submitted to University of Derby on 2014-12-11

< 1% match (Internet from 05-Apr-2019)
https://memorynotfound.com/spring-security-forgot-password-send-email-reset-password/

< 1% match (Internet from 12-May-2018)
https://o7planning.org/ru/10649/social-login-in-spring-mvc-with-spring-social-security

< 1% match (student papers from 18-Dec-2017)
Submitted to University of Greenwich on 2017-12-18

< 1% match (Internet from 19-May-2014)
http://www.vhuangwucha.com.cn/lovers/2013/9/23/02136.html

< 1% match (Internet from 10-Jul-2006)
http://nono.niaouli.org/2006/05/10/253-hibernate-annotations-mysql-blob-largeblob

< 1% match (Internet from 23-Oct-2013)
http://lumeniaconsulting.com/blog/john-donagher/big-bang-versus-phased-erp-implementations

https://www.turnitin.com/newreport_printview.asp?eq=0&eb=0&esm=0&oid=1363960011&sid=0&n=0&m=2&svr=50&r=50.8562007200502&lang=en_us