

School Code 10722

Student ID 1915034308

Classification No

Secret classification

咸陽師範學院

## Undergraduate Thesis ( Design )

Title Design and implementation of

Online exam system based on

Springboot framework and

Micorservices

Name of Author Gwenani James

Major Name Software Engineering

Instructor OuYang Hong Ji

Date of Submission June 2023

Performance Evaluation







## Abstract

This paper attempts at streamlining the exam process by making it easier for both examiners and examinees. Many endeavours have been made to solve this problem; but most end up falling short. The research contained in this paper was the result of rigorous study of similar systems that failed; explicitly finding out why they failed and how to improve on what they achieved and for intents and purposes avoiding the pitfalls that lead to their failure.

The solution that was derived after laborious research is an innovative approach to online exams by making them distributed systems that utilize microservices architecture. The proposed system leverages the advantages of microservices such as scalability, modularity, and flexibility to provide a reliable and efficient online exam platform. By breaking down the system into smaller, independent services, the system can be easily maintained and updated without disrupting the overall system. Additionally, this approach allows for greater customization and flexibility for both students and administrators. The system also provides enhanced security measures, such as authentication and authorization, to ensure the integrity of the exam process. The system was developed using Java as the main programming language and MS SQL as the preferred database. The Springboot framework was used to create the microservices.

**Keywords:** Microservices, Distributed system, Java, Springboot, MS SQL.

# Table of Contents

<b>Abstract.....</b>	<b>I</b>
<b>Chapter1:Introduction .....</b>	<b>1</b>
1.1Research overview .....	1
1.2Research Significance .....	1
1.3 Overall Structure .....	2
<b>Chapter2:Fundamentals/Environment and Related Works.....</b>	<b>3</b>
2.1 key technologies .....	3
2.2 General requirements .....	3
2.2.1 Hardware interface .....	3
2.2.2 Software interface .....	4
2.2.3 Development environment.....	4
2.3 Feasibility analysis.....	4
2.3.1 Technical Feasibility .....	4
2.3.2 Operational feasibility.....	5
2.4 Use cases .....	5
<b>Chapter3:Developed Architecture/System Design/Implementation .....</b>	<b>10</b>
3.1 System architecture .....	10
3.1.1 Functional Module .....	10
3.1.2 Database design .....	11
3.1.3 Database logic design .....	13
3.1.4 Database physical design .....	13
3.2 System implementation.....	24
3.2.1 System development operating environment.....	24
3.2.2 System security .....	24
3.2.3 API Gateway .....	25
3.2.4 Faculty microservice.....	26
3.2.5 Student microservice.....	34
<b>Chapter4:Result Analysis and Measurements .....</b>	<b>41</b>
4.1 Test purpose.....	41
4.2 Test environment .....	41
4.3 Main contents of test.....	41
4.4 White box test methods.....	41
4.5 Black box testing method.....	44
4.6 Test cases .....	45
<b>Chapter5:Conclusion .....</b>	<b>48</b>
5.1 Conclusion .....	48
5.2 Acknowledgements .....	49
<b>References.....</b>	<b>50</b>







# **Chapter1 Introduction**

## **1.1 Research overview**

Online exams have become increasingly popular in recent years due to their convenience and accessibility. With the rise of remote education and work, online exams have become an essential tool for assessing learning outcomes and skills. However, conducting online exams can be a challenging task due to various factors, including scalability issues, security concerns, and technical difficulties.

To address these challenges, many educational institutions and organizations are turning towards distributed systems[1] using microservices[2]. A distributed system is a collection of independent computers that communicate and coordinate their actions to achieve a common goal. Microservices architecture is an approach to building distributed systems that decomposes large applications into small, independent services that can be developed, deployed, and scaled independently.

## **1.2 Research Significance**

Adopting a microservices-based approach for online exam distribution allows for better scalability, reliability, and maintainability of the system. The primary goal of this research paper is to explore the benefits, challenges, and best practices of implementing a microservices-based distributed system for online exam distribution.

The paper will also examine how such a system can enhance the overall user experience for both students and instructors. One of the main advantages of utilizing microservices architecture for online exams is scalability. By breaking down the system into small, independent services, it becomes easier to scale up or down specific components of the system as needed. This modularity ensures that there is no single point of failure and makes it easier to add new features or functionalities to the system without disrupting the entire system. Additionally, microservices-based distributed systems ensure high availability and fault tolerance since they are designed to handle failures gracefully. If one service fails, the system can continue to function, providing a seamless experience for users. This is especially important for online exams, where any downtime could result in significant consequences for students, instructors, and schools.

However, implementing a microservices-based distributed system for online exams presents its own set of challenges. One of the primary challenges is ensuring security since exams contain

sensitive and confidential information. The system must be designed to prevent unauthorized access, ensure data confidentiality, and integrity. Another challenge is maintaining consistency across different services. With each service operating independently, it can become challenging to ensure data consistency across the system. This requires careful coordination between services to prevent data discrepancies, especially when handling high-stakes exams. Despite these challenges, there are many best practices that organizations can follow to ensure a successful implementation of a microservices-based distributed system for online exam distribution.

These include designing with loose coupling[3] in mind, using robust testing and monitoring tools, and enabling continuous delivery and deployment to ensure quick and seamless updates. In conclusion, this research paper will explore how microservices-based distributed systems can enhance the experience of online exam distribution for students and instructors while addressing scalability, reliability, and maintainability concerns. By examining the benefits, challenges, and best practices of implementing such a system, we can better understand its potential for transforming the way we conduct online exams in the future.

### **1.3 Overall Structure**

This paper contains six chapters that critically analyse data and present results in a technical but comprehensive manner.

The first chapter gives a brief introduction; offering to provide rational behind the research methodology. It also gives short but crisp overview of the entire paper's structure.

The second chapter presents the technologies that were utilized and makes comparisons with similar works. It dives deeper into the requirement and feasibility analysis.

The third chapter which forms the bulk of the research, encompasses the detailed design and implementation. Shading light on the security aspect of the software architecture.

The fourth chapter shows us the robust tests cases conducted on the system and the results derived from them.

The fifth chapter wraps everything up, by providing a swift but concise summation.

## **Chapter2 Fundamentals/Environment and Related Works**

### **2.1 Key technologies**

The systems will be developed using Java as the main programming language. Spring Tool Suite 4 will be the running environment of choice and Mysql4.5 server[4] will be utilized as the preferred database.

The system will make use of the SSM framework. The technical architecture of every system will go according to the Springboot framework[5], using the MVC design pattern. Mybatis framework[6] will be used at the model level helping with the java database connection; the Springboot native methods will help in processing the requests, and transfer data to where it is needed. Jstl[7] and Bootstrap will be used at the view level.

### **2.2 General requirements**

The application leverages the power of microservices to provide an efficient online exam platform. This section gives a detailed account of how the system operates and how it should be used. The platform allows students to take attend exam and Teacher can create exams, while moderators are allowed to audit the exam process.

The system must have an easy to use interactive interface for students and instructors. The system should be responsive and work on multiple devices, including but not limited to personal computers and tablets.

It should be able to handle concurrent users without significantly affecting the overall system performance. The system should authenticate users based on their roles and permissions. The system must encrypt all data transmitted over the network and store it in the database. Unauthorized access to exam data must be prevented at all costs.

#### **2.2 .1 Hardware interface**

The application can run on any electronic device but it is much better to run it on a computer for a much better experience.

#### **2.2.2 Software interface**

The necessary software of the system are listed below.

1. Operating System
2. Mysql
3. Web Browser

#### 4. Spring Tool Suite 4

### **2.2.3 Development environment**

#### 1. Website development environment

Operating system: Win 10

Development language: Java

Development Frameworks: Mybatis, Springboot

Development tools: Spring Tool Suite 4

#### 2. Server side running environment.

Mysql

#### 3. Client running environment.

Web Browser.

### **2.3 Feasibility analysis**

#### **2.3.1 Technical Feasibility**

High Scalability[8]: Microservices allow individual component to scale horizontally or vertically based on specific needs.

Easy Maintenance[9]: The fact the any particular microservice can be maintained individually lets developers to make changes without affecting the other components.

High Availability: If one service fails it does not impact the entire system because each service runs it own separate operation.

Security: The most vital part of the distributed exam system is the security. It must ensure that only authorised personnel can access the system and it must deny access to unauthorised personnel. The user data must be encrypted for confidentiality purposes.

### 2.3.2 Operational feasibility

There are a few innovative online exam systems that use micorservices such us; ExamSoft[10], Proctorio[11] and Honorlock. The table below shows a side by side comparison.

Table 1. Comparison table

<b>Feature</b>	Distributed online exam using microservices	ExamSoft	Proctorio	Honorlock
<b>Scalability</b>	High	High	High	High
<b>Security</b>	High	High	High	High
<b>Ease of maintenance</b>	High	Moderate	Moderate	Moderate
<b>User Experience</b>	High	Moderate	Moderate	Moderate
<b>Cost</b>	N/A	High	High	High

According to the technical and operational analysis it is clear that developing a distributed online exam system using microservices architecture is feasible. The proposed system can offer scalable, secure and easy to maintain solution for student to take exams remotely.

## 2.4 Use cases

The system has the four roles; faculty admin, student admin, student and faculty members. The faculty admin can do the following: register faculty members, view all faculty members, create new departments and courses, reset user passwords, schedule exams and login.

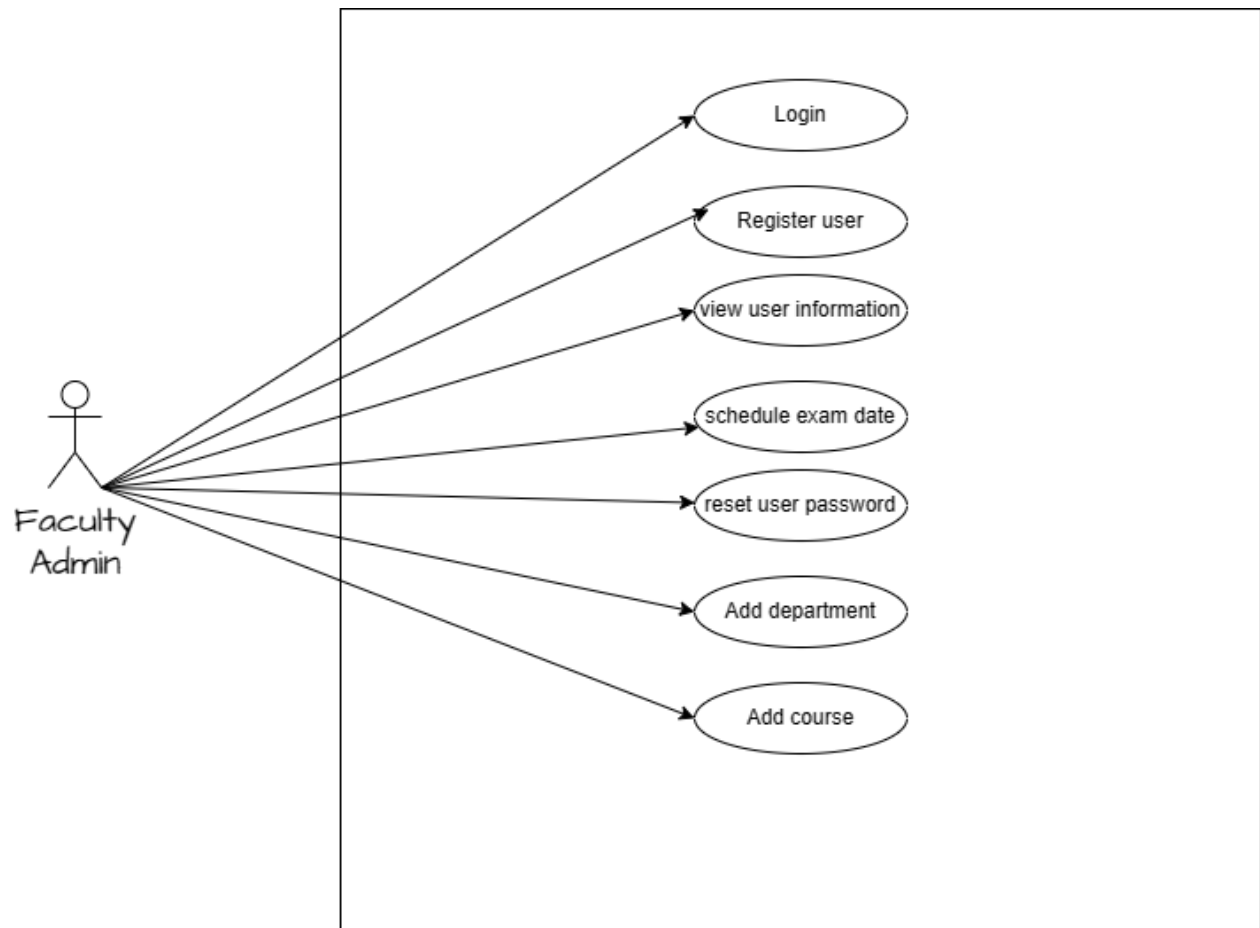


Figure 2.1 faculty admin use case diagram

The student admin do the following: login register new students, view all students information, reset students passwords and assign student credits.

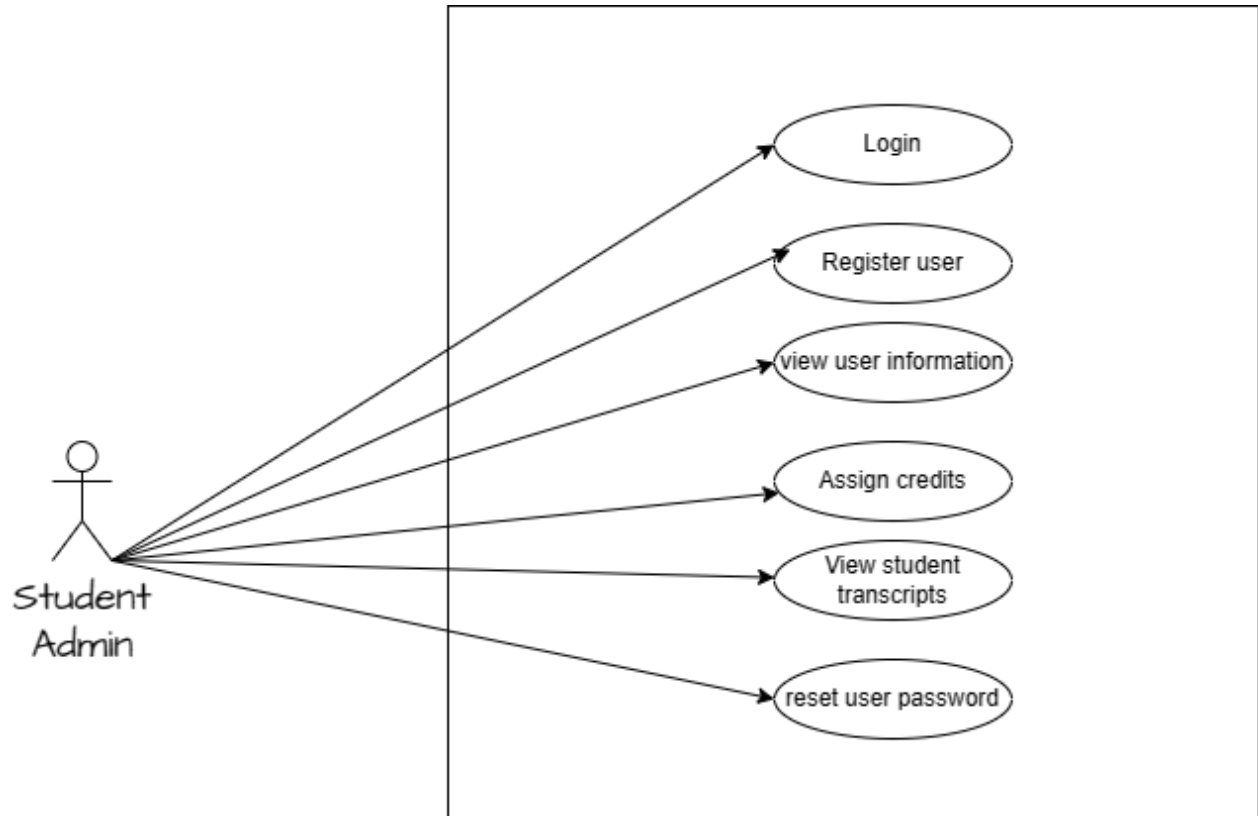


Figure 2.2 Student admin use case diagram

Faculty members can do the following: login, create exams, edit questions, delete exam questions, publish exams, grade students exams and view students transcripts.

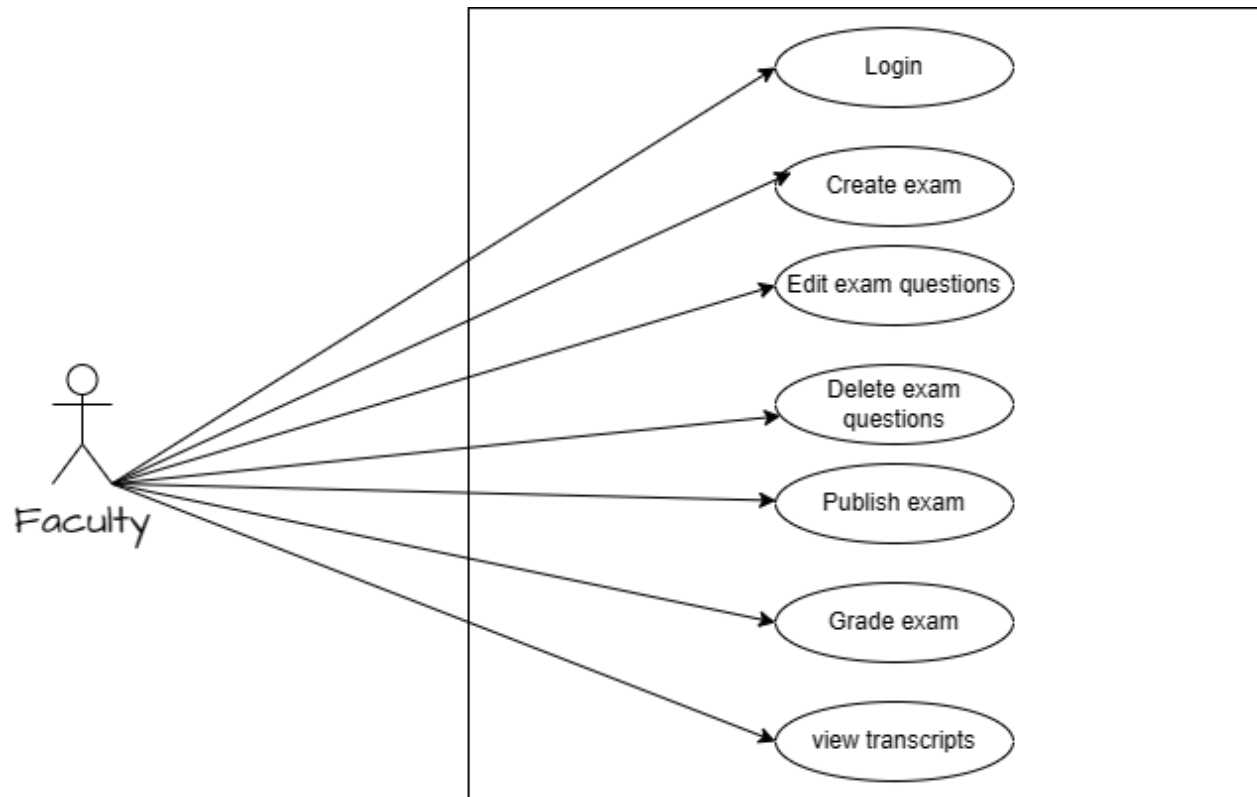


Figure 2.3 faculty use case diagram



Students can do the following: login, take exams and view their transcripts. Below is the system's use case diagram.

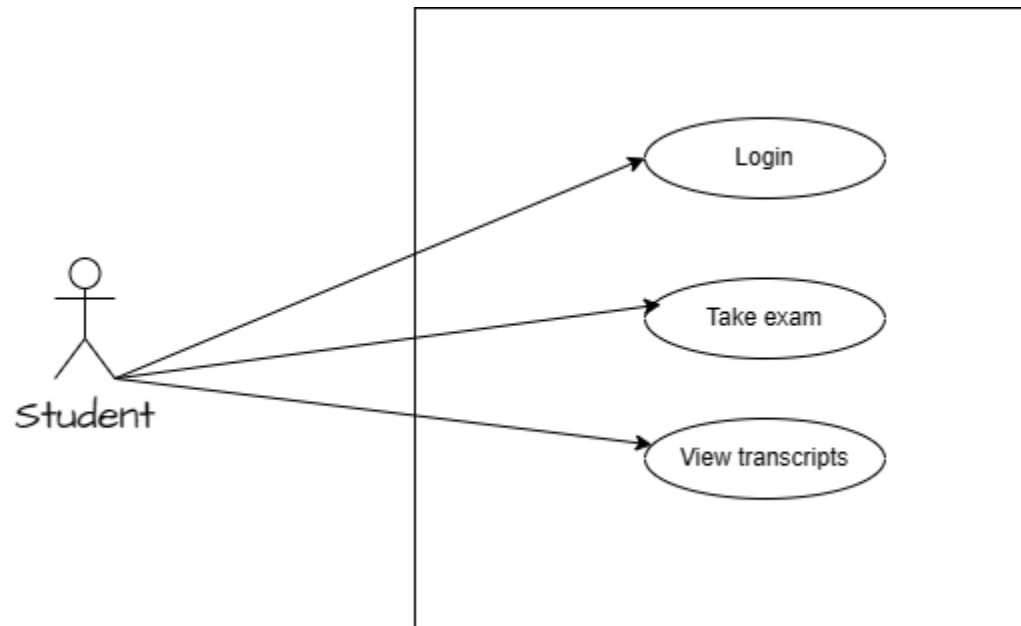


Figure 2.4 student use case diagram

## **Chapter3 Developed Architecture/System Design/Implementation**

### **3.1 System architecture**

The distributed online exam makes use of two microservices; the student portal microservice and faculty portal microservice. Each microservice has its own database because each service is its own process. Changes to an individual database don't impact other services. Thus, there isn't a single point of failure in the application. So to speak, the application is more resilient. This design mechanism is also advantageous if in the future we wanted use polyglot persistence[12]. The figures below show the two services' entity relation diagrams.

#### **3.1.1 Functional Module**

The distributed online exam is made up of several modules that work together to offer uniform exam experience to students and lessen the instructors' work load.

The exam creation module is responsible for creating and managing exams. The instructor can create new exams; set the questions and publish the exam when ready.

The grade exam module allows the instructors to inspect students' papers and grade them accordingly. This module automates the grading of multiple choice questions and lets the instructors grade the rest.

The track students' performance provides a report, of students' performance across a department. The register faculty module permits the faculty admin to register faculty members. The add department and courses module allows the faculty admin to add departments and courses across the whole system.

The set exam schedule awards the faculty admin the authority to set the time of an exam. It also grants the faculty admin the privilege of setting the time for when the exam will start and end.

The take exam module lets the student take a scheduled exam in the department. It makes sure that the student takes the exam on a section basis. Each section has its own designated time, after which the section closes.

The view transcripts enable the student to see his grades after they have been graded and the download transcripts make it possible for the student to download the transcripts.

The assign credits module authorizes the student admin to assign credits to students who took the exam. There is another register module that is used by the student admin to register new students.

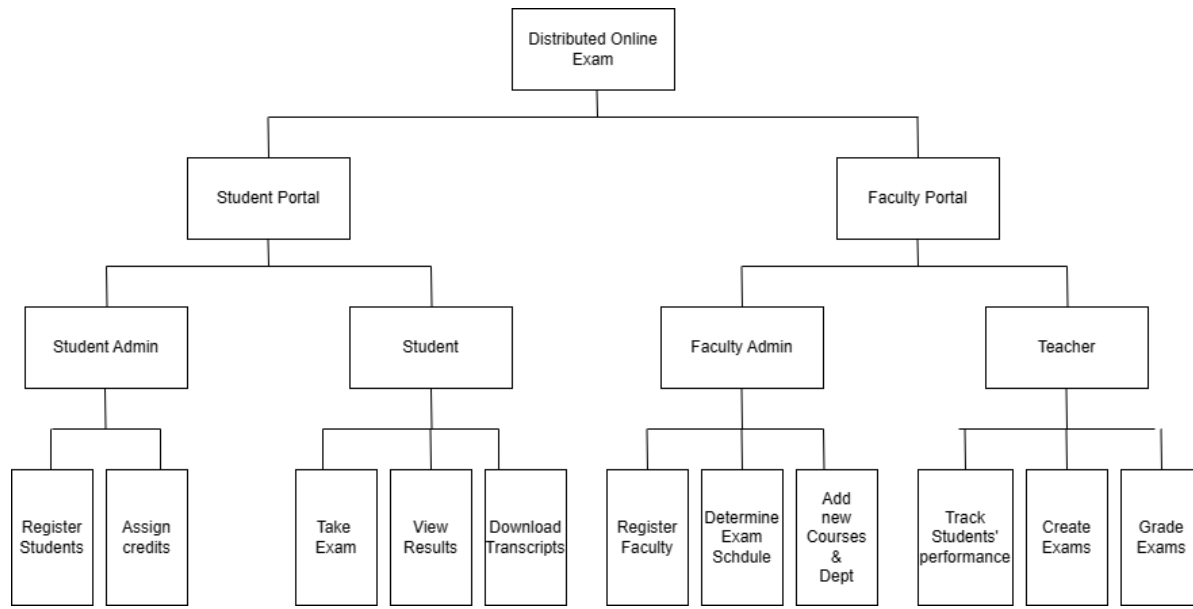


Figure 3.1 functional modules Diagram.

### 3.1.2 Database Design

The design of the database stems from the first forming the designing of entities and the relationships they share. Below are figures of the entity relation diagrams.

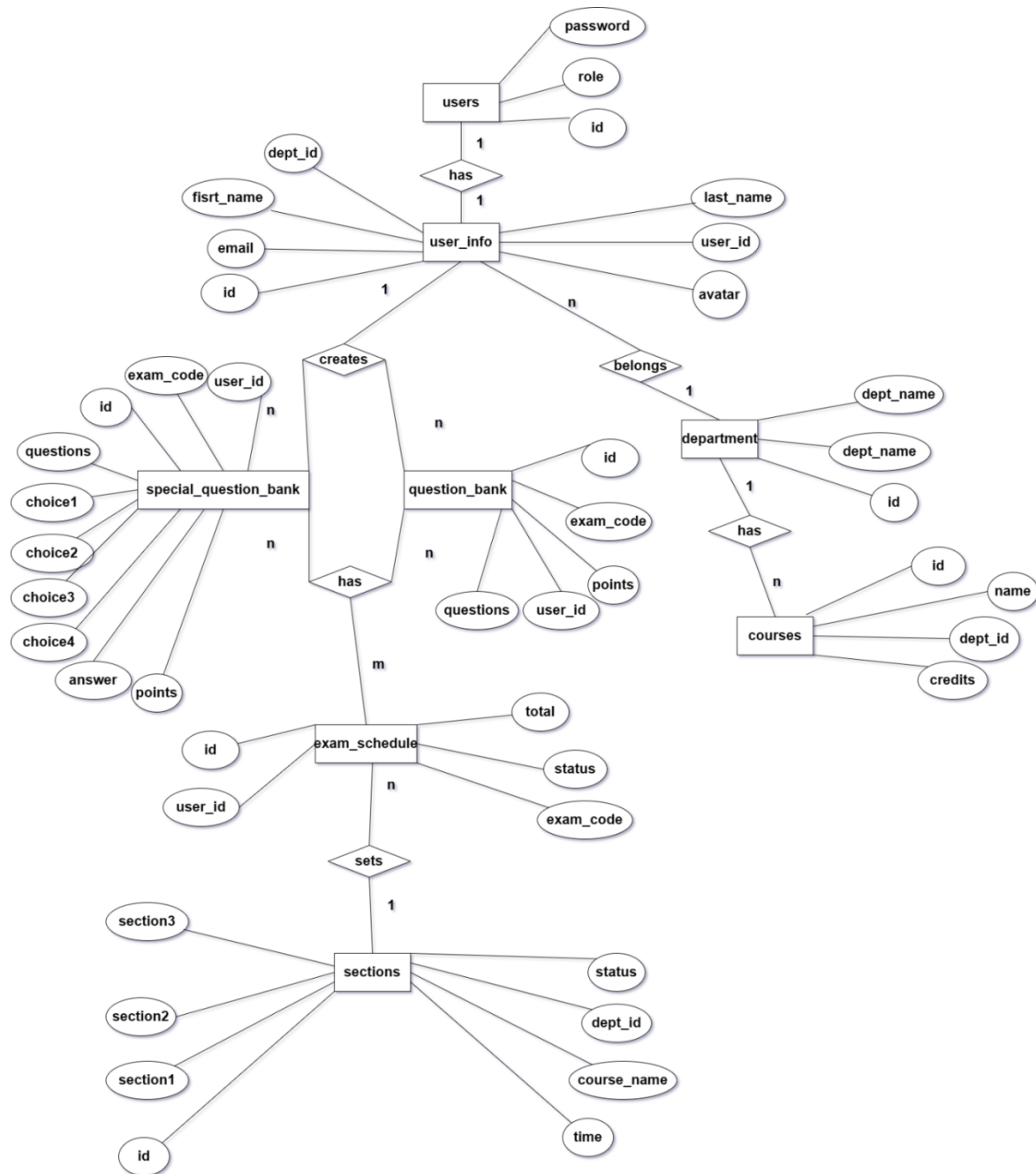


Figure 3.2 Faculty E-R Diagram.

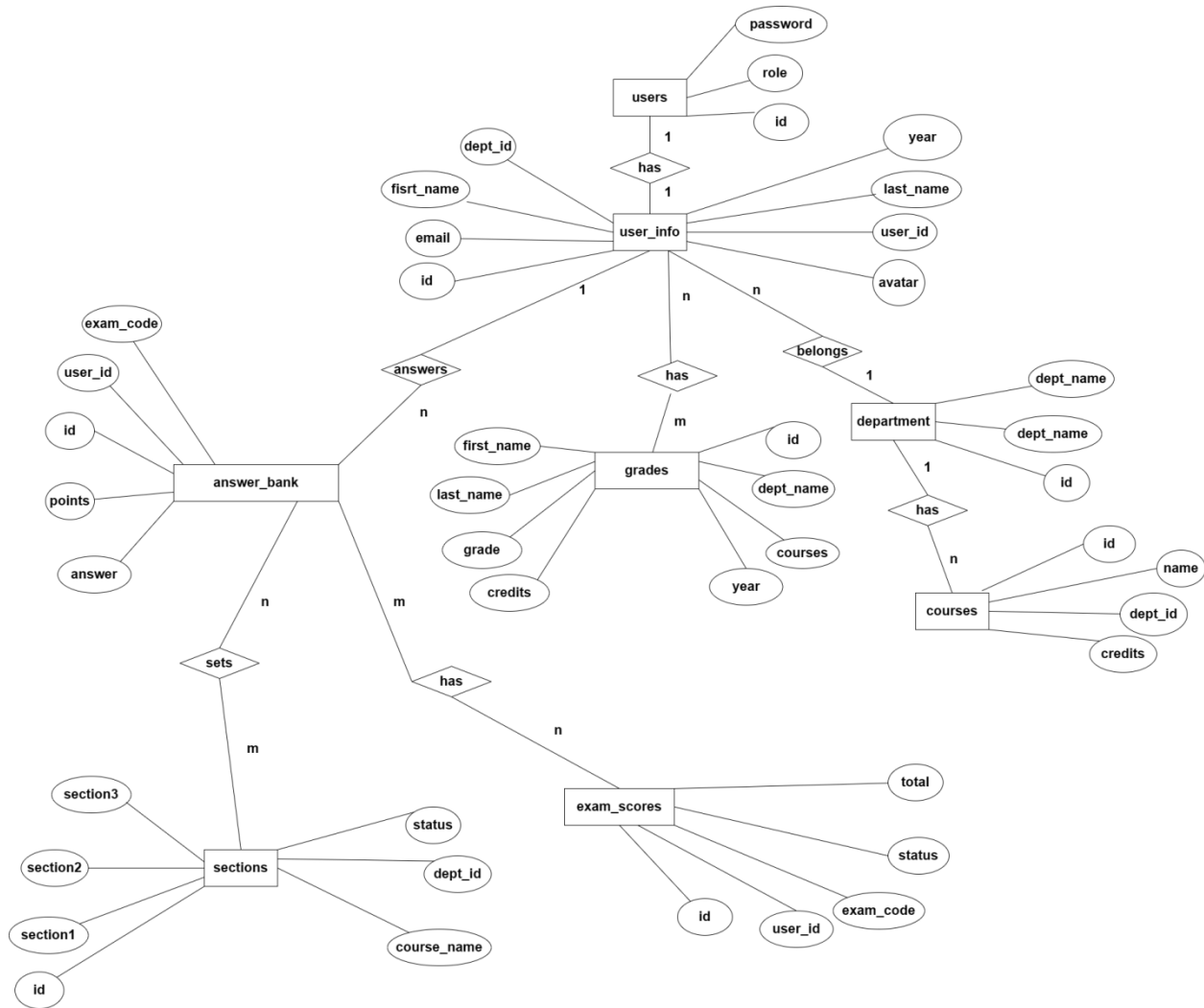


Figure 3.3 students E-R Diagram.

### 3.1.3 Database logic Design

The database per service approach is perfect because it meets the requirements that we trying to strive for; in that we are trying to develop a scalable loosely and independent system in terms of deployment.

One of the main drawbacks of this approach is that each service can only directly access its own database, we need to facilitate some sort of communication between services. The solution to this problem to will be addressed in the coming chapter.

### 3.1.4 Database physical Design

The following are tables for the Student schema.

Users Table. Stores important user credentials used to identify the type of user.

Table 3.1 Student users table.

id	Name	Datatype	length	Primary key	Not null	Expression
1	id	int	11	yes	Not null	User id
2	role	int	11	no	Not null	User role
3	password	varchar	150	no	Not null	User password

User Info Table. Houses more generic user information.

Table 3.2 Student users info table.

id	Name	Datatype	length	Primary key	Not null	Expression
1	id	int	11	yes	Not null	Primary id
2	User_id	int	11	no	Not null	User id
3	First_name	varchar	50	no	null	First name
4	Last_name	varchar	50	no	null	Last name
5	email	varchar	150	no	null	email
6	Dept_id	int	11	no	Not null	Department id
7	year	varchar	11	no	null	year
8	avatar	varchar	500	no	null	User photo

Answer Bank Table. Holds student answers.

Table 3.3 Student answer bank table.

id	Name	Datatype	length	Primary key	Not null	Expression
1	id	Int	11	yes	Not null	Primary id
2	answers	Varchar	1000	no	null	Student answers
3	points	Int	11	no	null	Student score
4	exam_code	Varchar	50	no	null	Exam code
5	user_id	int	11	no	null	User id

Department Table. Stores the available departments.

Table 3.4 Student department table.

id	Name	Datatype	length	Primary key	Not null	Expression
1	id	int	11	yes	Not null	Primary id
2	Dept_name	varchar	150	no	Not null	Department name

Courses Table. Stores the available courses

Table 3.5 Student courses table.

id	Name	Datatype	length	Primary key	Not null	Expression
1	id	int	11	yes	Not null	Primary id
2	Dept_id	int	11	no	Not null	Department id
3	name	varchar	150	no	Not null	Course name
4	credits	int	11	no	null	credits

Exam scores Table. Stores student exam scores for each section.

Table 3.6 Student scores table.

id	Name	Datatype	length	Primary key	Not null	Expression
1	Id	int	11	yes	Not null	Primary id
2	Exam_code	Varchar	150	no	Not null	Exam code
3	Total	Int	11	no	null	Total
4	User_id	Int	11	no	null	User id
5	status	Int	11	no	null	Status



Sections Table. Stores already created available exams.

Table 3.7 Student sections table.

id	Name	Datatype	length	Primary key	Not null	Expression
1	Id	Int	11	yes	Not null	Primary id
2	Section1	Varchar	50	no	Not null	Section A
3	Section2	Varchar	50	no	Not null	Section B
4	Section3	Varchar	50	no	Not null	Section C
5	Course_name	Varchar	150	no	Not null	Course name
6	Status	Int	11	no	null	Status
7	Dept_id	Int	11	no	Not null	Department id

Grades Table. Contains student transcript information.

Table 3.8 Student sections table.

id	Name	Datatype	length	Primary key	Not null	Expression
1	Id	Int	11	yes	Not null	Primary id
2	Dept_name	Varchar	150	no	Not null	Department name
3	Courses	Varchar	150	no	Not null	Course name
4	First_name	Varchar	50	no	Not null	First name
5	Last_name	Varchar	50	no	Not null	Last name
6	Year	Varchar	50	no	Not null	Year
7	Credits	Int	11	no	Not null	Credits
8	grade	varchar	50	no	Not null	grade

The following are tables for the Faculty schema.

Users Table. Users Table. Stores important user credentials used to identify the type of user.

Table 3.9 Faculty users table.

id	Name	Datatype	length	Primary key	Not null	Expression
1	id	int	11	yes	Not null	User id
2	role	int	11	no	Not null	User role
3	password	varchar	150	no	Not null	User password

User Info Table. Houses more generic user information.

Table 3.10 Faculty users info table.

id	Name	Datatype	length	Primary key	Not null	Expression
1	id	int	11	yes	Not null	Primary id
2	User_id	int	11	no	Not null	User id
3	First_name	varchar	50	no	null	First name
4	Last_name	varchar	50	no	null	Last name
5	email	varchar	150	no	null	email
6	Dept_id	int	11	no	null	Department id
8	avatar	varchar	500	no	null	User photo

Special Question Bank Table. Stores multiple choice questions.

Table 3.11 Faculty special questions table.

id	Name	Datatype	length	Primary key	Not null	Expression
1	id	Int	11	yes	Not null	Primary id
2	Questions	Varchar	1000	no	null	Questions
3	Choice1	Varchar	100	no	null	Option A
4	Choice2	Varchar	100	no	null	Option B
5	Choice3	Varchar	100	no	null	Option C
6	Choice4	Varchar	100	no	null	Option D
7	Answer	Varchar	100	no	null	Answer
8	Exam_code	varchar	150	no	null	Exam code
9	Points	Int	11	no	null	Score
10	User_id	int	11	no	null	User id

Question Bank Table. Stores ordinary questions.

Table 3.12 Faculty questions table.

id	Name	Datatype	length	Primary key	Not null	Expression
1	id	Int	11	yes	Not null	Primary id
2	Questions	Varchar	1000	no	null	Questions
3	Exam_code	varchar	150	no	null	Exam code
4	Points	Int	11	no	null	Score
5	User_id	int	11	no	null	User id

Department Table. Stores available departments.

Table 3.13 Faculty department table.

id	Name	Datatype	length	Primary key	Not null	Expression
1	id	int	11	yes	Not null	Primary id
2	Dept_name	varchar	150	no	Not null	Department name

1) Courses Table. Stores available courses.

Table 3.14 Faculty courses table.

id	Name	Datatype	length	Primary key	Not null	Expression
1	id	int	11	yes	Not null	Primary id
2	Dept_id	int	11	no	Not null	Department id
3	name	varchar	150	no	Not null	Course name
4	credits	int	11	no	null	credits

Exam Schedule Table. Stores all the exams created by teachers.

Table 3.15 Faculty exam schedule table.

id	Name	Datatype	length	Primary key	Not null	Expression
1	Id	int	11	yes	Not null	Primary id
2	Exam_code	Varchar	150	no	Not null	Exam code
3	Total	Int	11	no	null	Total
4	User_id	Int	11	no	Not null	User id
5	status	Int	11	no	null	Status

Sections Table. Shows published exams.

Table 3.16 Faculty sections table.

id	Name	Datatype	length	Primary key	Not null	Expression
1	Id	Int	11	yes	Not null	Primary id
2	Section1	Varchar	50	no	Not null	Section A
3	Section2	Varchar	50	no	Not null	Section B
4	Section3	Varchar	50	no	Not null	Section C
5	Course_name	Varchar	150	no	Not null	Course name
6	Status	Int	11	no	null	Status
7	Dept_id	Int	11	no	Not null	Department id
8	time	DATE		no	Not null	Exam date

## **3.2 System implementation**

### **3.2.1 System development and operation environment**

(1) Hardware environment

CPU: intel(R)core (TM) i3-5250U CPU @1.8GHz

RAM:8GB

(2) Software environment.

Operating system: windows 10

Database: Mysql

Main Tech: Springboot, java, Mybatis.

### **3.2.2 System security**

The system makes use of spring security[13] to define the roles of all the users and assign them adequate permissions. There are essentially two types of roles, an admin role and user role. Once the user successfully logs in; spring security forwards them to the relevant page. Spring security prohibits any one user to view pages that are beyond their role's domain.



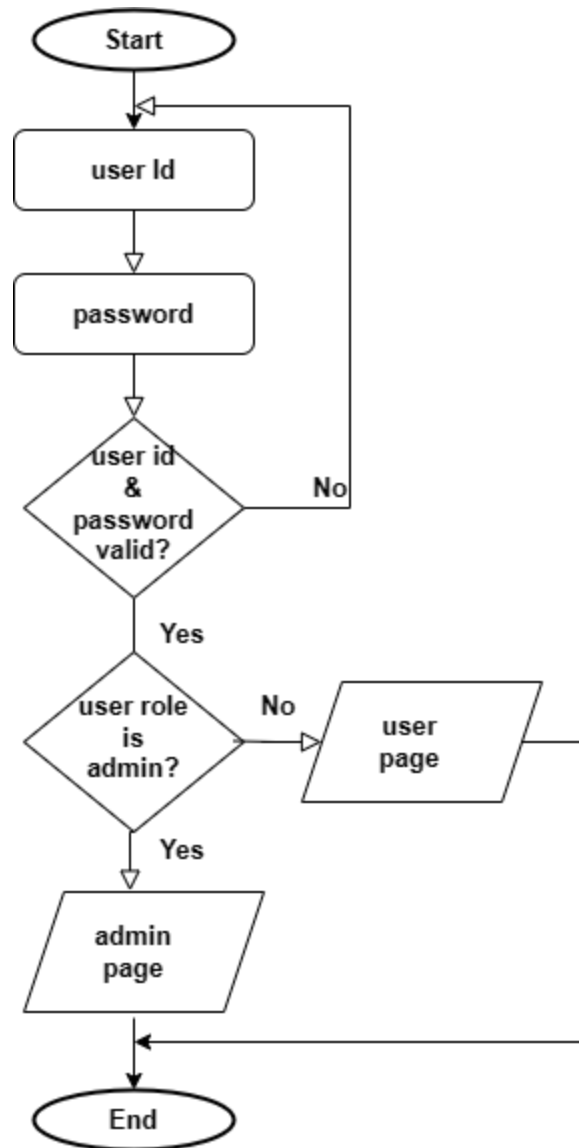


Figure 3.4 User authentication flow chart diagram.

### 3.2.3 API gateway

The user can access the application using a simulated API gateway[14] made possible by Spring REST template[15], which aggregates different API's from different services. Services can communicate asynchronously, which helps avoid low latency. This solves the problem that was mentioned earlier in chapter2.

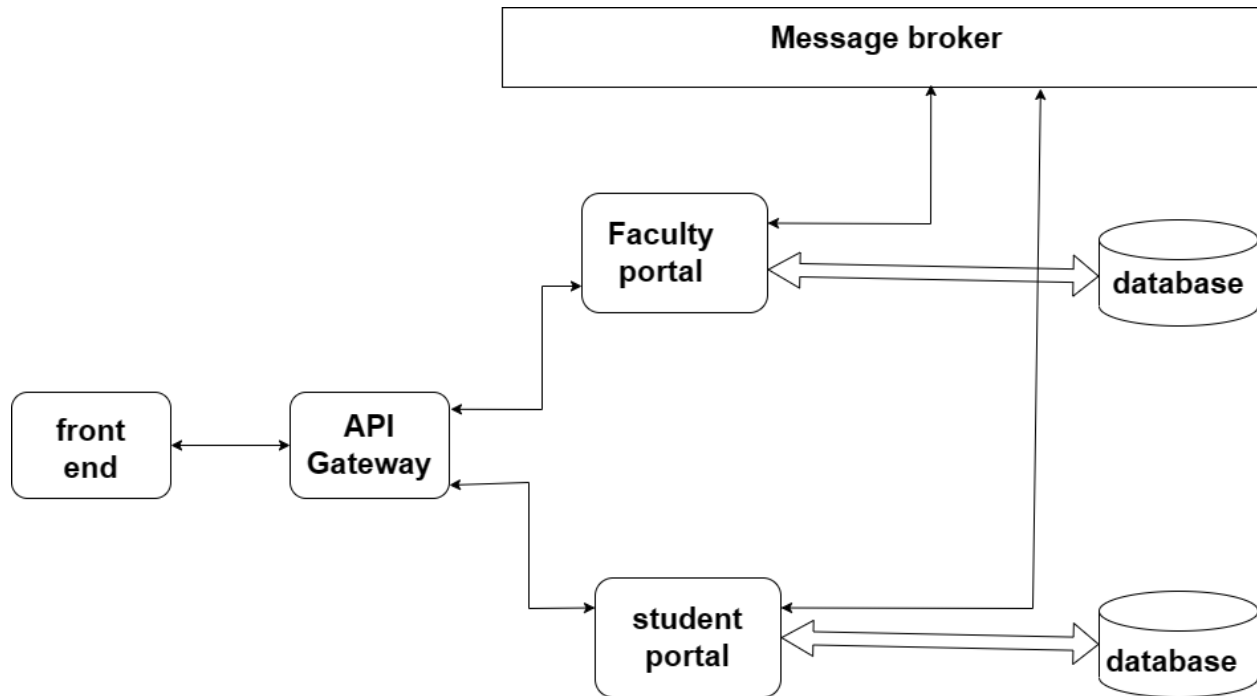


Figure 3.5 API illustration diagram.

### 3.2.4 Faculty microservice

The faculty microservice has two main modules which are the admin and faculty modules. The two modules have sub modules that perform different functions. The main task of the faculty module is to create and grade exam, while the admin module performs the duty of delivering the exam to the students.

#### 1) Create Exam

This module is made up of five methods; namely the create exam, edit questions, delete questions, add more questions and publish exam methods. Once the exam has been created edit, add and delete operations can be performed but once published; no other operations can be carried out. The goal of this module is ultimately to publish an exam.

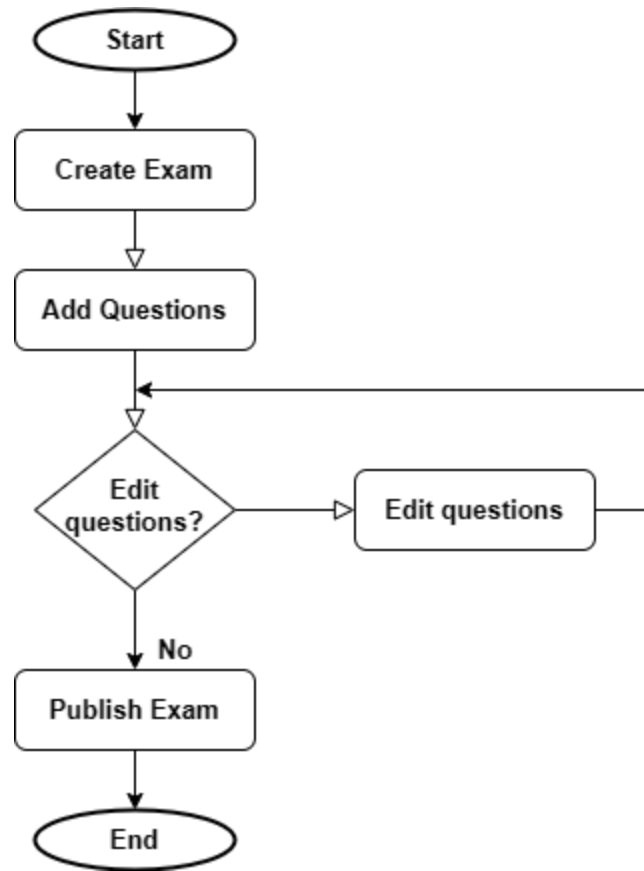


Figure 3.6 Create exam flow chat diagram.

**Below is a snippet of the publish exam source code:**

```
public void publish(String exam_code,int user_id) {  
    ExamSchedule schedule=new ExamSchedule();  
    schedule.setExam_code(exam_code);  
    schedule.setStatus(1);  
    schedule.setUser_id(user_id);  
    teacherRepo.updateExamSchedule(schedule);  
}
```

The method accepts two parameters; a unique string that represents the exam code and the teacher's identification number. An exam schedule object is created and its attributes are assigned values. The schedule object's exam code attribute is assigned the parameter exam code; it's status attribute is set to 1 representing a published status., and the object user id attribute is set to the teacher's id. The teacher repository update exam schedule method is then called and the exam schedule object is passed as a parameter. The repository method then inserts the schedule information into the exam schedule table in the database; hence the exam is published.

**Below is a picture of the running result:**

The screenshot displays a web interface for publishing an exam. It features three question entries, each with a text box for the question, a weight input field, and a 'Delete' button. Below the questions are buttons for 'Edit', 'Publish Exam', and 'Add More Questions'. At the bottom, there are two dropdown menus for selecting the subject and section.

Question Text	Weight	Action
Explain Multidimensional Arrays.	5	Delete
How Can You Use Linked Lists and Arrays? Give Examples.	5	Delete
Do Linked Lists Have Any Advantages Over Arrays?	5	Delete

**Edit**

Data Structures and Algorithms ▾ Section2 ▾

**Publish Exam**

**Add More Questions**

Figure 3.7 Publish exam running result

## 2) Schedule Exam

Once the exam status has been changed to published, the faculty admin can set the date for when the exam is going to take place. After the admin sets the date; a post response via the Spring REST template is sent to the student portal, containing up to date information regarding the exam. Additionally once the time comes for the exam, the faculty admin can change the status of the exam to active and this data is sent to the student portal as well. Once the exam is over the faculty admin has to change the exam status to inactive and the corresponding change is made on the student portal.

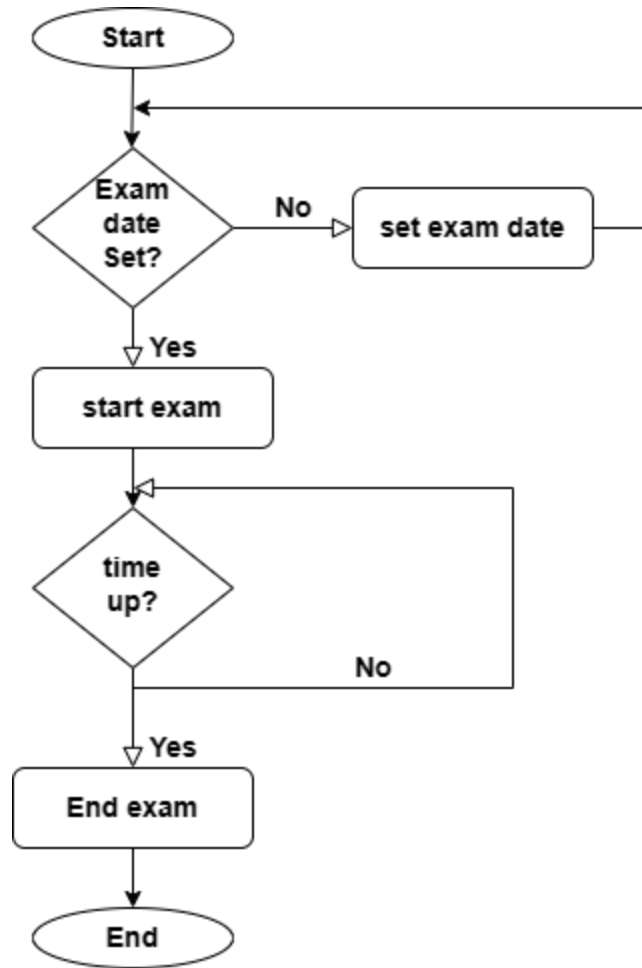


Figure 3.8 Schedule exam flow chat diagram.

**Below is a snippet of the start exam controller method:**

```

@RequireLogin
@RequireRole
@RequestMapping("/faculty/admin/StartExam")
public String startExam(@RequestParam("id")int id,ModelMap modelMap) {
    Sections section= adminRepo.findSectionById(id);
    section.setTime(null);
    request.changeSectionStatus(section);
    service.startExam(id);
    return showAllSections(modelMap);
}

```

Every method in the controller has a Required login and Required role annotation; they make sure the user is logged in and is an admin respectively(Required role is only used for admin

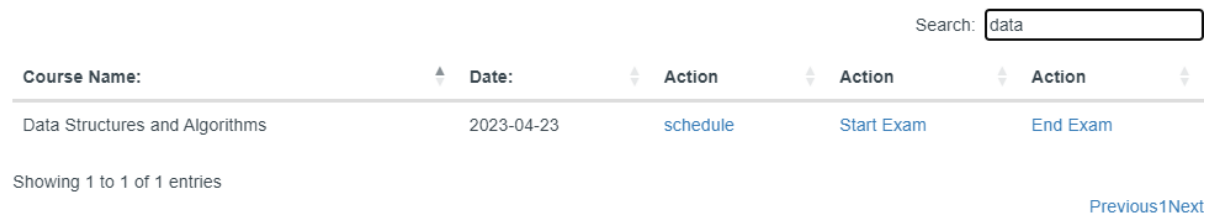
users). The start exam admin controller method has a string return type, that returns a string url for the show all sections page. It does this by calling another controller method called show all sections. The start exam method takes an integer id which holds the value of the section id and model map to transfer data to the view layer, as parameters.

A section object is created and is assigned its value when an admin repository method; find section by id is called, where the id value is passed as a parameter. The section time attribute is then set to null.

A request is then made to the student microservice and the section object is sent to the student microservice where a corresponding REST method will be triggered to enter the section info into the student database. After this process has been executed successfully then a service method called start exam is called that takes id as a parameter. This method uses the given id to determine which section in the section table in the database needs to be set to active.

In short the exam status is changed to active in the student microservice then the faculty microservice follows suite.

**Below is a picture of the running result:**



Search: <input type="text" value="data"/>				
Course Name:	Date:	Action	Action	Action
Data Structures and Algorithms	2023-04-23	<a href="#">schedule</a>	<a href="#">Start Exam</a>	<a href="#">End Exam</a>
Showing 1 to 1 of 1 entries				
<a href="#">Previous</a> 1 <a href="#">Next</a>				

Figure 3.9 schedule exam running result

### 3) Grade Exam

A request for student answers based on the exam code is made to the student portal microservice. The response is a list of all students who took the exam. Once the response from the student microservice has been obtained successfully, the faculty controller queries the faculty database for exam questions. Then the answers and questions are showed together, for the teacher to assign scores.

Once all the scores have been assigned successfully, a post response containing the students' scores is sent to the student microservice.

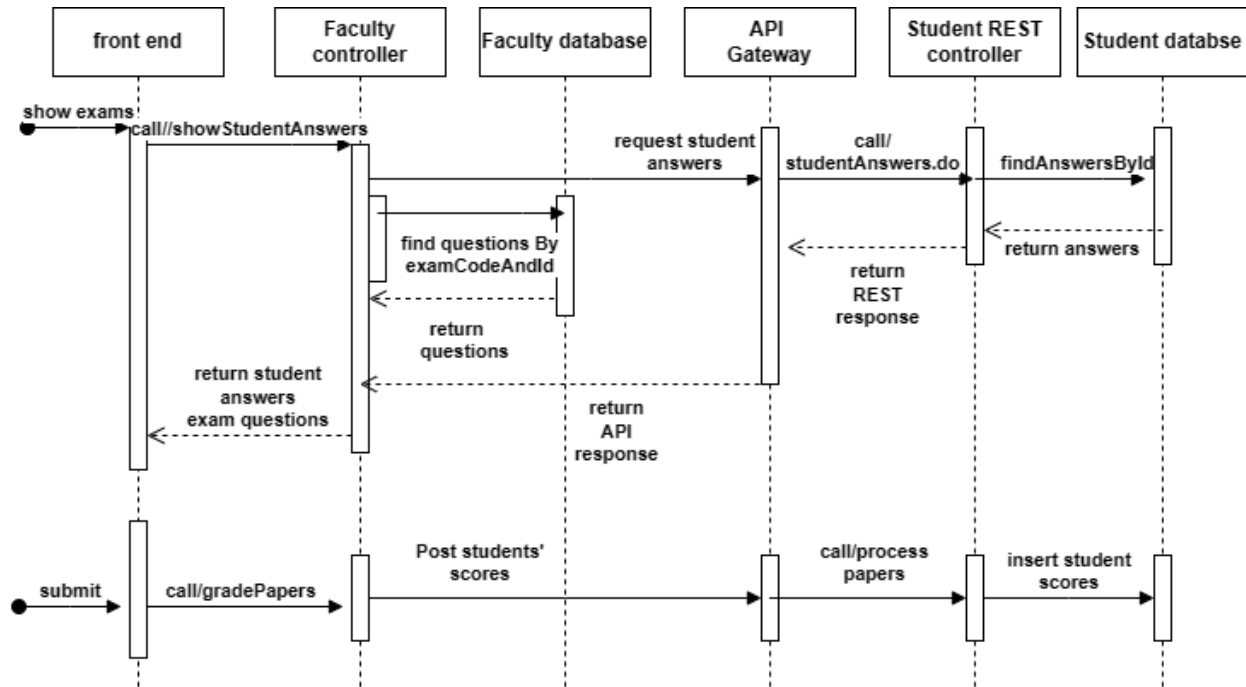


Figure 3.10 Grade exam sequence diagram.

Below is a snippet of the faculty show student answers controller method:

```

@RequireLogin
@RequestMapping("/faculty/user/showStudentAnswers")
public String showStudentAnswers(@RequestParam("user_id")int
user_id,@RequestParam("exam_code")String exam_code,ModelMap modelMap, HttpSession session) {
    int teacherUser_id=(int)session.getAttribute("user_id");
    AnswerBankList info= request.studentAnswers(user_id, exam_code);
    List<QuestionBanks>questions=
teacherRepo.findQuestionsByExam_codeAndTeacher_id(exam_code,teacherUser_id);
    if(questions.isEmpty()) {
        return "403";
    }
    else {
        modelMap.addAttribute("questions", questions);
        modelMap.addAttribute("info", info.getList());
        modelMap.addAttribute("avatar", (String)session.getAttribute("avatar"));
        modelMap.addAttribute("email", (String)session.getAttribute("email"));
    }
}
  
```

```
return "/faculty/StudentAnswers";}}
```

The faculty controller method show student answers returns a string value which is the url string for the student answers page. The method has the following parameters: a string called exam code representing the unique exam code, an integer called user id holding the student's id, a model map object that will be used to store data that will be shown in the view layer, a session object that will be used to retrieve and save important data.

An integer teacher id is initialized, its value is obtained when the session object get attribute method is called. Then An answer bank object is created and its value is generated when the request to the student microservice retrieves student answers based on the user id(student id) and exam code. After this a question bank object Is created; getting its value when a teacher repository method, find question by exam code and teacher id, is called.

If the question bank is empty then the user is directed to a 403 error page, meaning the exam doesn't exist or they don't have any write permission on the specified exam.

Otherwise they are redirected to the student answers page where they can view the questions and answers. However; before this happens one more thing has to happen that is, the model object will be used to store data to be shown in the view layer. The questions, answers and other things will be added to the model map object to be shown in the student answers page.



**Below is a snippet of the faculty graded papers controller method:**

```
@RequireLogin
@RequestMapping("/faculty/user/gradedPapers")
public String gradedPapers(@RequestParam("points") List<Integer>
points, @RequestParam("exam_code") String exam_code, @RequestParam("user_id") int user_id,
ModelMap modelMap, HttpSession session) {
    int teacherUser_id=(int)session.getAttribute("user_id");
    List<String> answers =teacherRepo.findAnswersByTeacher_idAndExamCode(teacherUser_id,(String)
session.getAttribute("section1"));
    int total = teacherRepo.findScheduleByExam_code(exam_code).getTotal();
        GradedPapers paper = new GradedPapers();
        paper.setAnswers(answers);
        paper.setExam_code(exam_code);
        paper.setUser_id(user_id);
        paper.setMaximumPoints(total);
        paper.setPoints(points);
    paper.setCourse_name((String)session.getAttribute("course_name"));
    paper.setSection1((String)session.getAttribute("section1"));
    paper.setSection2((String)session.getAttribute("section2"));
    paper.setSection3((String) session.getAttribute("section3"));
        request.gradedPapers(paper);
        return showStudentInfo(exam_code,modelMap,session);
}
```

The grade papers faculty controller method takes the following parameters and returns a string value(url for the show student info page): a list called points carrying all the scores that were assigned to the student, a string called exam code representing the unique exam code, an integer called user id holding the student's id, a model map object that will be used to store data that will be shown in the view layer, a session object that will be used to retrieve and save important data. A variable called teacher id is initialized and gets its value when a session get attribute method is called. A list called answers(stores multiple choice answers ) is then created and obtains its value when a teacher repository method called find answers by teacher id and exam code, is called. The total score of the exam is obtained when the repository method find exam schedule by exam code is called.

A graded paper object is then created and its attributes are awarded their values accordingly.

Then a request is made to the student microservice and the grade paper object is sent to the student microservice where a corresponding REST controller method will process and then insert the data in the database.

**Below is a picture of the running result:**

Sort the following list {5,59,4,3,0} using bubble sort.[5]

{5,4,3,0,59}  
{4,3,0,5,59}  
{3,0,4,5,59}

Points To be awarded: 5

Explain Multidimensional Arrays.[5]

I have no idea

Points To be awarded:

Figure 3.11 grade exam running result

### 3.2.5 Student microservice

The student microservice has two main modules which are the admin and student modules. The two modules have sub modules that accomplish a number of diverse duties. The chief task of the student module is to allow students to take the exam and show them their grades, while the admin module has the job of assigning students' their credits.

#### 1) Exam process

The student controller requests for exam questions from the faculty microservice, using exam code as a parameter. This request is made three times because there are three exam sections. The answers for each section are saved in the student database.

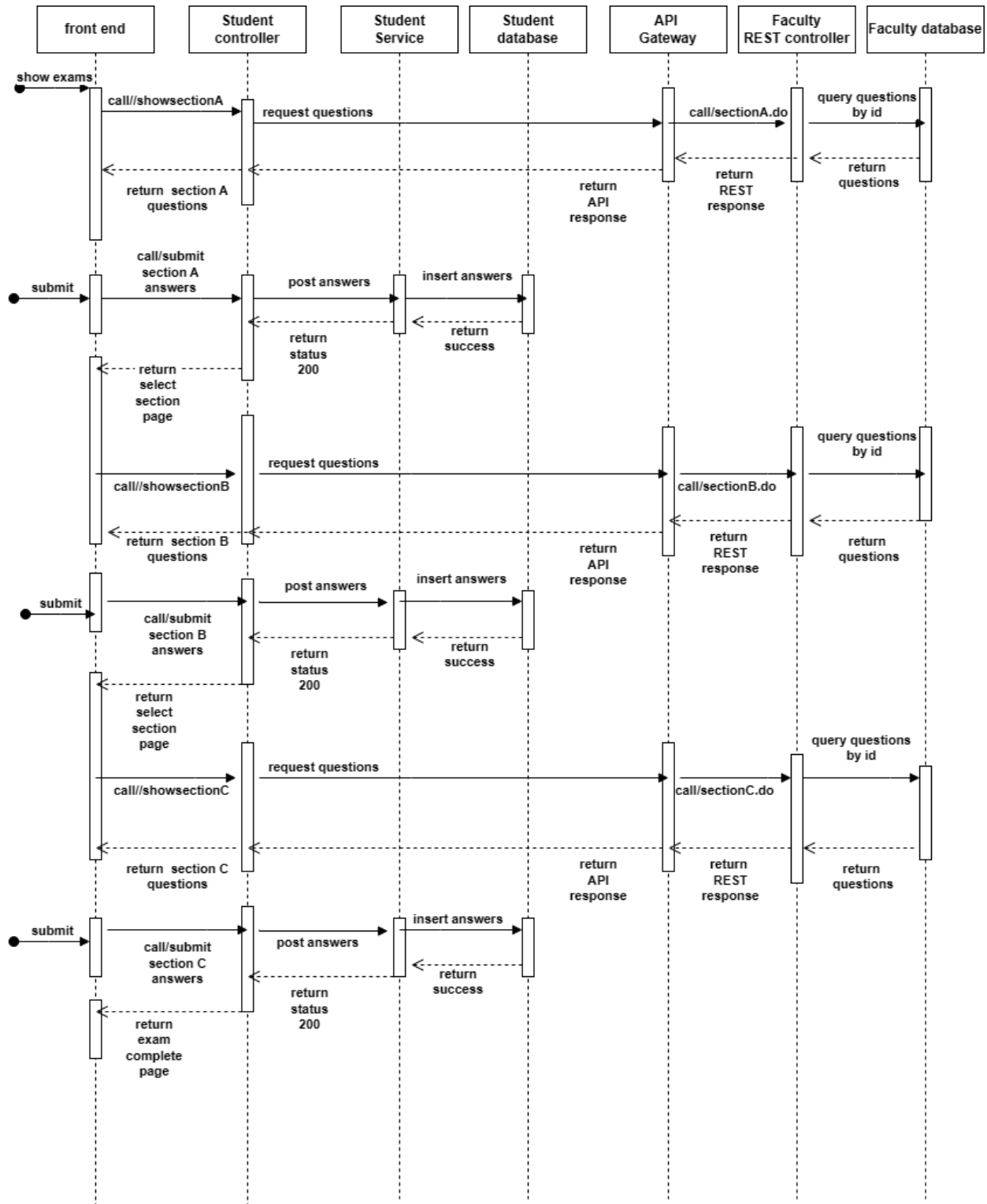


Figure 3.12 Student exam sequence diagram.

**Below is a snippet of the show section A student controller:**

```
@RequireLogin
@RequestMapping("/student/user/showSectionA")
public String showSectionA(ModelMap modelMap, @RequestParam("exam_code")String
exam_code) {
    SpecialQuestionList questions=request.showSectionA(exam_code);
    modelMap.addAttribute("questions", questions.getSpecialList());
    return "/student/SectionA";
}
```

The student controller method show section A returns a string and takes the following parameters: a string called exam code which holds the unique exam identification code and a model map object that helps transfer data to the view layer.

A request is made to the faculty microservice; all multiple choice type questions are retrieved according to the exam code which is passed as a parameter. Then this data is bound to the model map object so that it can be seen in the section A page; Thereafter, the student is directed to the section A page.

**Below is a snippet of the insert answers methods:**

```
@Transactional
@Override
public int insertSingleChoiceAnswers(AnswerBankPojo pojo) {
    int counter=0;
    AnswerBank answers=new AnswerBank();
    int size=pojo.getAnswers().size();
    for(int i=0;i<size;i++) {
        answers.setAnswers(pojo.getAnswers().get(i));
        answers.setExam_code(pojo.getExam_code().get(i));
        answers.setPoints(0);
        answers.setUser_id(pojo.getUser_id());
        counter=studentRepo.insertIntoAnswerBank(answers);
    }
    ExamScores scores=new ExamScores();
    scores.setExam_code(answers.getExam_code());
    scores.setStatus(0);
}
```

```

        scores.setUser_id(answers.getUser_id());
        studentRepo.insertScoreInfo(scores);
        return counter;
    }

```

This method uses the Transaction annotation which helps in managing transactions and JDBC connections. The insert single choice service method returns an integer and only takes one parameter; the answer pojo object, this object holds multiple lists of very important data. It holds lists of all the student's answers, user id and exam code.

All the lists in the pojo object are traversed and inserted into the database recursively. Every time an insertion is completed successfully, it is noted and this value is what the method returns. The user id and exam code are then inserted into the exam scores table; for later operations that will be carried out, such the insertion of the student's scores.

**Below is a picture of the running result:**

29:47

shinigami@gmail.com Log Out

14	For a binary search algorithm to work, it is necessary that the array (list) must be	<input type="checkbox"/> sorted	<input type="checkbox"/> unsorted	<input type="checkbox"/> in a heap	<input type="checkbox"/> 0
15	What data structure is used for breadth first traversal of a graph?	<input type="checkbox"/> list	<input type="checkbox"/> queue	<input type="checkbox"/> none of the above	<input type="checkbox"/> 0
16	Which of the following asymptotic notation is the worst among all?	<input type="checkbox"/> $\Theta(n+9378)$	<input type="checkbox"/> $2\Theta(n)$	<input type="checkbox"/> $\Theta(n^3)$	<input type="checkbox"/> 0

Figure 3.13 Exam running result.

## 2) Credit Assignment

After the exam has ended and the student answers have been assessed by the teacher in the faculty microservice; the student admin requests for the student id of all the students that participated in the exam. The student admin then selects on the individual student and the credits are assigned once the final grade is calculated.

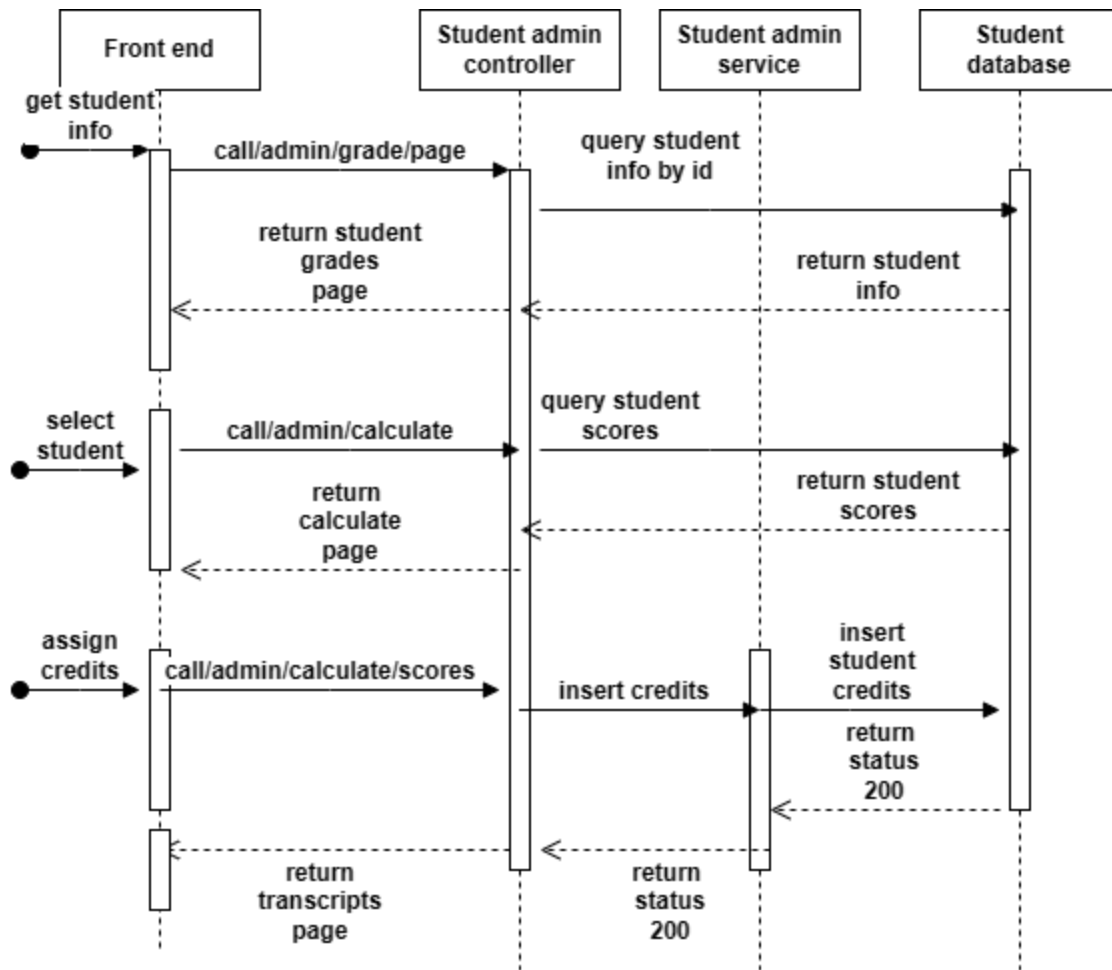


Figure 3.14 Credit Assignment sequence diagram.

**Below is a snippet of the student admin credit assignment method:**

```

@Override
public int jellyFish(int user_id, String section1, String section2, String section3, String
course_name) {
    Grades grade=new Grades();
    UserInfo info=adminRepo.findUserInfoByUserId(user_id);
    Courses course=adminRepo.findCourseByName(course_name);
    Departments dept=adminRepo.findDeptById(info.getDept_id());
    int Section1Score= adminRepo.findTotalByUserIdAndExamCode(user_id, section1);

```

```
        int Section2Score= adminRepo.findTotalByUserIdAndExamCode(user_id, section2);
        int Section3Score= adminRepo.findTotalByUserIdAndExamCode(user_id, section3);
        int max1= request.total(section1).getTotal();
        int max2=request.total(section2).getTotal();
        int max3=request.total(section3).getTotal();
        String result=
Vanillalemonaide(Section1Score,Section2Score,Section3Score,max1,max2,max3);
        grade.setGrade(result);
        grade.setDept_name(dept.getDept_name());
        grade.setCredits(course.getCredits());
        grade.setCourses(course_name);
        grade.setFirst_name(info.getFirst_name());
        grade.setLast_name(info.getLast_name());
        grade.setYear(info.getYear());
        int i=appleSacue(grade);
        crixCross(i,user_id,section1,section2,section3);
        return i;
    }
```

The faculty service method jellyfish returns an integer whose value shows whether the method executed correctly or not. It has the following parameter: An integer called user id which represents the student's id, three strings called section1 section2 and section3 holding the exam codes for their corresponding sections and a string called course name that has the course's name. The total scores obtained by the student for each section are obtained and then assigned to variables; called section1 score, section2 score and section3 score respectively. Then a request is made to the student micorservice in order to get the total score for each section, the variables are names as follows; max1, max2 and max3. Using these values another service method called vanilla lemonade gets these values and calculates the final grade of the student.

The a grade object is created and all it attributes are assigned their necessary values. This object is then passed as a parameter when the apple sauce method is called. This method checks if the student grade related information exist, depending on the outcome it either updates or inserts the grade information. It returns an integer value called I, this value is what the entire method will return, signaling whether the operation was successful or not.

Finally a method called `crix cross` is called; it gets the student Id and all the sections exam codes and changes the student's paper status to graded in the database.

**Below is a picture of the running result:**

Yoyo

Rodriguez

Section Info:					
Section: CSA	Score: 8	Section: CSB	Score: 6	Section: CSC	Score: 15
Course Name:		Section1:	Section2:	Section3:	
Data Structures and Algorithms		CSA	CSB	CSC	<a href="#">Assign Credits</a>

Figure 3.15 credit assignment running result.



## **Chapter4 Result Analysis and Measurements**

### **4.1 Test Purpose**

To ensure every component of the system is working the way it should. To search for any possible bottlenecks that may appear.

### **4.2 Test Environment**

Processor: intel(R)core (TM) i3-5250U CPU @1.8GHz

RAM:8GB

Operating system: windows 10

Browser:Chrome

### **4.3 Main Contents of Tests**

The tests will be carried out to check if the main functions of the system run smoothly. A system test will be carried out on the grading algorithm. A manual test will be used on the following: publish exam, schedule exam, attend exam and assign credits.

### **4.4 White box testing methods**

The basic path coverage method will be used to test the inner workings of one of the most important algorithms of the entire application. The path coverage will be used to test the grade assigning algorithm. The path coverage method was used because it is executes enough test cases to cover all paths in the program.

Procedure (VAR score1 , score2,score3,max1,max2,max3 , grade : REAL);

BEGIN

1.Read finalScore=score1+score2+score3

2.Read finalMax=max1+max2+max3;

3.Read computedScore=(finalScore/finalMax)\*100;

4.IF ( computedScore>=95)

5. THEN grade="A+" ;

6. IF ( computedScore>=90)

7. THEN grade="A" ;

8. IF ( computedScore>=85)

9. THEN grade="A-" ;

10.IF ( computedScore>=80)

11. THEN grade="B+" ;

12. IF ( computedScore>=75)

13. THEN grade="B" ;

14. IF ( computedScore>=70)

15. THEN grade="B-" ;

16. IF ( computedScore>=65)

17. THEN grade="C+" ;

18. IF ( computedScore>=60)

19. THEN grade="C" ;

20. IF ( computedScore>=55)

21. THEN grade="C-" ;

22.IF ( computedScore>=50)

23. THEN grade="D+" ;

24. IF ( computedScore>=45)

25. THEN grade="D" ;

26. IF ( computedScore>=40)

27. THEN grade="D-" ;

28. ELSE

29. THEN grade="F" ;

30.END;

Given the input ;score1,score2,score3,max1,max2 and max3, a single output denoting the final grade is expected. The executable paths are as follows:

P1=[1,2,3,4,5,30]

P2=[1,2,3,4,6,7,30]

P3=[1,2,3,4,6,8,9,30]

P4=[1,2,3,4,6,8,10,11,30]

P5=[1,2,3,4,6,8,10,12,13,30]

P7=[1,2,3,4,6,8,10,12,14,16,17,30]

P8=[1,2,3,4,6,8,10,12,14,16,18,19,30]

P9=[1,2,3,4,6,8,10,12,14,16,18,20,21,30]

P10=[1,2,3,4,6,8,10,12,14,16,18,20,22,23,30]

P11=[1,2,3,4,6,8,10,12,14,16,18,20,22,24,25,30]

P12=[1,2,3,4,6,8,10,12,14,16,18,20,22,24,26,27,30]

P13=[1,2,3,4,6,8,10,12,14,16,18,20,22,24,26,28,29,30]

The expected output for each path is as follows:[P1=A+] [P2=A] [P3=A-] [P4=B+] [P5=B] [P6=B-] [P7=C+] [P8=C] [P9=C-] [P10=D+] [P11=D] [P12=D-] [P13=F].

## 4.5 Black box testing methods

The decision table will be used to examine the publish exam functionality. The exam is published once all the modifications are done, once published the exam cannot be edited. The decision table was used because it provides complete coverage of test cases which help reduce the rework on writing test cases and scenarios.

Table 4.1 Publish exam decision table.

Condition								
Exam created	no	yes	yes	yes	no	no	no	yes
Edit exam	no	no	yes	yes	yes	no	yes	no
Delete exam	no	no	no	yes	yes	yes	no	yes
Action								
Publish exam	X	✓	X	X	X	X	X	X

Once the exam has been published we use the decision table to test the set exam schedule procedure. Once the exam date is set the admin can start and end the exam at the appropriate time.

Table 4.2 Exam schedule decision table.

Condition				
Exam date set	No	Yes	Yes	No
Time Up	No	No	Yes	Yes
Action				
Set exam date	✓	X	X	✓
Start exam	X	✓	X	X
End exam	X	X	✓	X

## 4.6 Test cases

A test case table will be used to explore the student's exam procedure. The student takes the exam on a section by section basis, meaning only after he is done with one section can he progress to the next section.

Table 4.3 Attend exam use case table.

Test case Id	Test case objective	Prerequisite	Steps	Input data	Expected outcome	Actual outcome	Status
Tc_01	Display scheduled exam	N/A	1)Select attend exam in the student portal	N/A	Show Select section page is displayed	Show Select section page is displayed	PASS
Tc_02	Select the first section of the exam	N/A	1)Select section A in the select section page	N/A	Show section A page	Show section A page	PASS
Tc_03	Submit section A answers	1)All fields are required to be filled	1)Answer all the questions in section	1)Section A answers	1)submit section A answers 2)Show Select section page	1)submit section A answers 2)Show Select section page	PASS
Tc_04	Select second section of the exam	N/A	1)select section B in the select	N/A	Show section B page	Show section B page	PASS

			section page				
Tc_05	Submit section B answers	1)All fields are required to be filled	1)Answer all the questions in section	1)Section B answers	1)submit section B answers 2)Show Select section page	1)submit section B answers 2)Show Select section page	PASS
Tc_06	Select third section of the exam	N/A	1)select section C in the select section page	N/A	Show section C page	Show section C page	PASS
Tc_07	Submit section C answers	1)All fields are required to be filled	1)Answer all the questions in section	1)Section C answers	1)submit section C answers 2)Show exam completed page	1)submit section B answers 2)Show exam completed page	PASS

A second test case table will be used to analyse the credit assignment procedure. Once all the credits have been assigned the admin is redirected to a page where he can view all the transcripts.

Table 4.4 Assign credits use case table.

Test case Id	Test case objective	Prerequisite	Steps	Input data	Expected outcome	Actual outcome	Status
Tc_01	1)Get names of all students that completed exam	N/A	1)click on assign credits button in admin portal	N/A	1)Show all students that completed exam	1)Show all students that completed exam	PASS
Tc_02	1)Get individual student information	N/A	1)click on one student in list	N/A	1)Show students score for every section	1)Show students score for every section	PASS
Tc_03	1)calculate final grade of student 2)Assign credits to student	N/A	1)Click on assign credit button	N/A	1)Assign credits and final grade 2)show student transcripts	1)Assign credits and final grade 2)show student transcripts	PASS

## Chapter5 Conclusion

### 5.1 Conclusion

This application's main task is to make the exam process easier for both teachers and students alike. It lets faculty members create exams; edit them accordingly and then later publish them. Afterwards the faculty admin sets the exam date. When the exam date approaches the students then attend the exam; subsequently the faculty member can then grade the exams. Once this is done the students can then view their transcripts only after the student admin assigns them their credits. One of the biggest problems that kept coming up was finding a way to facilitate efficient communication between different microservices; because each microservice has its own database. The safeguarding of data to be sent between services was the number one priority. Measures had to be put in place, in case something went wrong such a connection exception or any other type of exception. Hence the implementation of a vigorous API gateway.

There are a few improvements that can be made to optimize the system even further; such as braking down the admin functionality into its own microservcie thereby making the application even more decoupled. In conclusion the online exam distributed system using microservices is an auspicious tactic that can alter the way exam are operated. The use of microservices provides flexibility, ease of maintenance and scalability.

Furthermore the system guarantees the integrity and security of the exam procedure, as well as offering a suave and well-organized user experience. The online distributed exam system using microservices has the latent ability to provide dependable and effective solution for conducting exams in this modern era. The sustained fostering and amendment of this system can yield for an ameliorate and more equitable education system.



## **5.2 Acknowledgements**

The research in this paper was a result of countless consultations; multiple corrections that led to complete and well researched findings. This of course wouldn't have been possible if my supervisor hadn't prodded me in the right research direction. Giving guidance whenever necessary, and for that I am eternally grateful.

## References

- [1]Fleischmann, A. (1994). Introduction to Distributed Systems and Distributed Software. In: Distributed Systems. Springer, Berlin, Heidelberg. [https://doi.org/10.1007/978-3-642-78612-9\\_1](https://doi.org/10.1007/978-3-642-78612-9_1)
- [2]Familiar, B. (2015). Microservice Architecture. In: Microservices, IoT, and Azure. Apress, Berkeley, CA. [https://doi.org/10.1007/978-1-4842-1275-2\\_3](https://doi.org/10.1007/978-1-4842-1275-2_3)
- [3]Mankovski, S. (2018). Loose Coupling. In: Liu, L., Özsu, M.T. (eds) Encyclopedia of Database Systems. Springer, New York, NY. [https://doi.org/10.1007/978-1-4614-8265-9\\_1187](https://doi.org/10.1007/978-1-4614-8265-9_1187)
- [4]Christudas, B. (2019). MySQL. In: Practical Microservices Architectural Patterns. Apress, Berkeley, CA. [https://doi.org/10.1007/978-1-4842-4501-9\\_27](https://doi.org/10.1007/978-1-4842-4501-9_27)
- [5]Katamreddy, S.P.R., Upadhyayula, S.S. (2023). Getting Started with Spring Boot. In: Beginning Spring Boot 3. Apress, Berkeley, CA. [https://doi.org/10.1007/978-1-4842-8792-7\\_2](https://doi.org/10.1007/978-1-4842-8792-7_2)
- [6]Katamreddy, S.P.R., Upadhyayula, S.S. (2023). Working with MyBatis. In: Beginning Spring Boot 3. Apress, Berkeley, CA. [https://doi.org/10.1007/978-1-4842-8792-7\\_6](https://doi.org/10.1007/978-1-4842-8792-7_6)
- [7]Spielman, S., Kunnumpurath, M., Ellis, N., Weaver, J.L. (2004). Using the JSTL. In: Pro J2EE 1.4: From Professional to Expert. Apress, Berkeley, CA. [https://doi.org/10.1007/978-1-4302-0756-6\\_4](https://doi.org/10.1007/978-1-4302-0756-6_4)
- [8]Taylor, I.J., Harrison, A.B. (2009). Scalability. In: Taylor, I.J., Harrison, A.B. (eds) From P2P and Grids to Services on the Web. Computer Communications and Networks. Springer, London. [https://doi.org/10.1007/978-1-84800-123-7\\_11](https://doi.org/10.1007/978-1-84800-123-7_11)
- [9]Müller, F., Dazer, M., Bertsche, B. (2020). Maintainability. In: Vajna, S. (eds) Integrated Design Engineering. Springer, Cham. [https://doi.org/10.1007/978-3-030-19357-7\\_11](https://doi.org/10.1007/978-3-030-19357-7_11)
- [10]Gentile, M., Melovitz-Vasan, C., Huff, S. *et al.* The Utilization of ExamSoft®-iPad® Technology in Administering and Grading Anatomy Practical Examinations. *Med.Sci.Educ.* **29**, 831–840 (2019). <https://doi.org/10.1007/s40670-019-00750-0>
- [11]Topuz, A.C., Saka, E., Fatsa, Ö.F. *et al.* Emerging trends of online assessment systems in the emergency remote teaching period. *Smart Learn. Environ.* **9**, 17 (2022). <https://doi.org/10.1186/s40561-022-00199-6>
- [12]Dhall, C. (2018). Polyglot Persistence. In: Scalability Patterns. Apress, Berkeley, CA. [https://doi.org/10.1007/978-1-4842-1073-4\\_6](https://doi.org/10.1007/978-1-4842-1073-4_6)
- [13] Gutierrez, F. (2016). Security with Spring Boot. In: Pro Spring Boot. Apress, Berkeley, CA. [https://doi.org/10.1007/978-1-4842-1431-2\\_9](https://doi.org/10.1007/978-1-4842-1431-2_9)

- [14] Vijayakumar, T. (2018). API Gateways. In: Practical API Architecture and Development with Azure and AWS. Apress, Berkeley, CA. [https://doi.org/10.1007/978-1-4842-3555-3\\_4](https://doi.org/10.1007/978-1-4842-3555-3_4)
- [15] Varanasi, B., Bartkov, M. (2022). RESTful Spring. In: Spring REST. Apress, Berkeley, CA. [https://doi.org/10.1007/978-1-4842-7477-4\\_3](https://doi.org/10.1007/978-1-4842-7477-4_3)

