A Project Report on

ONLINE EXAMINATION PORTAL

A dissertation submitted to the

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, KAKINADA

In partial fulfillment of the requirement for the award of degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

M.K.RAMA REDDY (216T1A0535)

Under the esteemed Guidance of

Mr. Perneedi Chakradhara Rao, MCA, M.TECH,

Assistant professor

Department Of Computer Science and Engineering

PYDAH COLLEGE OF ENGINEERING

(Approved by AICTE, New Delhi, Affiliated to JNTUK &

Accredited by NAAC with 'A')

Patavala-533461.

April – 2025

CERTIFICATE

This is to certify that the thesis entitled **"ONLINE EXAMINATION PORTAL"**, is being submitted by **M.K.RAMA REDDY** bearing **PINNO : 216T1A05035** respectively in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering in the Pydah College of Engineering, Kakinada, Andhra Pradesh to the Jawaharlal Nehru Technological University, Kakinada is record of bonifide work carried out by them under by guidance of supervision.

The result embodied in this thesis have not been submitted to any other university or institute for the award of any degree or diploma.

Project Guide

External Examiner

Principal

Head of the Department

ACKNOWLEDGEMENT

"Task successful" makes everyone happy. But happiness will be gold without glitter if we didn't state the persons who have supported us to make it a success.

Success will be crowned to people who made it a reality but the people whose constant guidance and encouragement made it possible will be crowned first on the eve of success.

This acknowledgement transcends the reality of formality when we would like to express gratitude and respect to all those people behind the screen who guided, inspired, and helped us for the completion of our project work.

I consider myself lucky enough to get such a good project. This project would be an asset to my academic profile.

I would like to express my thankfulness to my project guide, **MR.P.CHAKRADHAR RAO**, and **MS.M. JYOTHI,** Head of the Department, for their constant motivation and valuable help throughout the project work and also thankful to all the department of CSE staff as they give excellent support to my project.

I express my gratitude to **DR. P.V. SURYA PRAKASH,** Principal of **PYDAH COLLEGE OF ENGINEERING**, for their constant supervision, guidance, and co-operation throughout the project.

I express my gratitude to **DR. M. VEERBHADRA RAO,** Dean of **PYDAH COLLEGE OF ENGINEERING,** for their constant supervision, guidance, and co-operation throughout the project.

I also extend my thanks to team members for their co-operation during my course. Finally, I would like to thank my friends for their co-operation to complete this project.

M.K.RAMA REDDY (216T1A0535)

ABSTRACT

The Online Examination Portal is a web-based application designed to streamline the process of conducting and managing exams. The system allows teachers to create and schedule tests, register students, and monitor attendance. Students can attempt tests only if they have been granted access by the teacher using a unique college code. After entering the code, students are directed to the instructions page, where they can see the scheduled tests. Upon completion, the system automatically evaluates the test and displays the student's score. The portal enhances efficiency, reduces paperwork, and ensures secure and fair test-taking conditions.

The Online Examination Portal is a web-based platform designed to automate and streamline the process of conducting exams in educational institutions. Developed using Java and powered by the Spring Framework, this system provides a robust, modular, and scalable architecture suitable for real-time online assessments.

The development environment is set up using Eclipse IDE, with Chocolatey serving as the package manager to automate the installation of essential tools and libraries such as Java JDK, Maven, and Tomcat. Apache Maven is employed for project management, build automation, and dependency handling, ensuring a clean and maintainable codebase.

The application follows the MVC (Model-View-Controller) architecture facilitated by Spring Boot, which simplifies configuration and accelerates development. The backend data is managed using MySQL, a reliable relational database system, for storing user details, question banks, results, and logs. Apache Tomcat acts as the deployment server, enabling smooth execution and hosting of the web application.

Keywords: Authentication, front-end development, User Privacy, Java, MySQL, User Security, Springboot, JDBC.

CONTENTS

LIST OF FIGURES

LIST OF SCREEN SHOTS

CHAPTER-1

INTRODUCTION

1.1 BACKGROUND

With the rapid advancement and widespread adoption of digital learning platforms, the limitations of traditional examination systems have become increasingly evident. These conventional systems are often plagued by logistical challenges such as the need for physical infrastructure, scheduling constraints, and large-scale coordination efforts. Additionally, manual evaluation processes not only consume significant time and resources but are also prone to human error and subjectivity. Security concerns, including the potential for impersonation, paper leaks, and unauthorized access, further compromise the integrity of such examinations.

To address these issues, the **Online Examination Portal** has been developed as a modern, secure, and highly efficient solution tailored for educational institutions. This digital platform enables educators to seamlessly create, schedule, and manage a variety of assessments ranging from quizzes to full-length examinations. Students can participate in these assessments from approved devices within a controlled virtual environment, which simulates the discipline and fairness of traditional exam settings.

A standout feature of the portal is its integration of a **unique college code system**, which acts as a gatekeeping mechanism. Only verified students associated with a specific institution can register and participate in exams linked to that institution's code. This approach significantly reduces the risk of unauthorized access, impersonation, and academic dishonesty, thereby upholding the integrity and credibility of the examination process.

By combining user-friendly interfaces, real-time monitoring tools, automated evaluation systems, and robust security protocols, the Online Examination Portal not only streamlines the administrative burden for educators but also enhances the overall assessment experience for students. It stands as a reliable and scalable solution for the evolving needs of digital education in the 21st century.

1.2 AIM OF THE PROJECT

The main objective of this project is to develop an online examination portal that simplifies the process of conducting assessments while ensuring security and accessibility. The system aims to:

- Provide a structured platform for test creation and administration.
- Secure student access using a college code system.
- Automate test evaluation and result generation.
- Enable teachers to monitor student performance and attendance.

1.3 Methodology

The development of this system follows the Software Development Life Cycle (SDLC), which includes:

- **Requirement Analysis:** Identifying functional and non-functional requirements.
- **System Design:** Creating architecture diagrams, data flow models, and database structures.
- **Implementation:** Developing the frontend using HTML, CSS, Tailwind CSS, and JavaScript, while using Java Spring Boot for backend processing and MySQL for data storage.
- **Testing:** Conducting unit, integration, and system testing to ensure functionality and security.
- **Deployment:** Deploying the system on a server and making it accessible to users.

1.4 Significance of this Project

- The Online Examination Portal offers several advantages, including:
- **Time Efficiency:** Eliminates manual paperwork and speeds up test administration.
- **Enhanced Security:** Only authorized students can access exams using the college code.
- **Automated Evaluation:** Reduces human error in grading and provides instant results.
- **Improved Accessibility:** Students can take exams from anywhere, making learning more flexible.
- **Performance Tracking:** Teachers can track student progress and identify absentees.

- **Cost-Effective:** Reduces expenses related to printing, physical space, and invigilation.
- **Scalability:** Easily accommodates a growing number of students and institutions.
- **Environmentally Friendly:** Minimizes paper usage, contributing to sustainability efforts.
- **Data Management:** Secure storage and retrieval of exam records, results, and analytics.

1.5 Outline of this Report

This document consists of the following chapters:

Chapter 1: Introduction – Provides an overview of the project, including background, aims, methodology, and significance.

Chapter 2: Literature Survey – Reviews existing examination systems and discusses how this project improves upon them.

Chapter 3: System Analysis – Describes the existing and proposed systems, along with feasibility analysis and requirement specifications.

Chapter 4: System Study – Explains the system architecture, module descriptions, and UML diagrams.

Chapter 5: Software Environment – Lists the technologies used in the project.

Chapter 6: Sample Code – Provides relevant code snippets used in the system.

Chapter 7: System Testing – Details the testing process and results.

Chapter 8: Screenshots – Includes screenshots of the developed system.

Chapter 9: Conclusion – Summarizes the project's outcomes and future enhancements.

Chapter 10: Bibliography – Lists references and sources used in the project.

1.6 Conclusion

The introduction chapter lays the foundation for understanding the need for an online examination portal and the methodology adopted for its development. By addressing security concerns through the college code mechanism and automating evaluation, this system provides a modern approach to conducting online exams. The following chapters provide deeper insights into the system’s design, implementation, and testing phases.

Additionally, the chapter highlights the key challenges faced in traditional examination systems and how the proposed solution effectively overcomes them. It sets the context for understanding the project's objectives, scope, and overall significance in the realm of digital education.

CHAPTER-2

LITERATURE SURVEY

An online examination portal is a digital platform that enables the administration of exams through the internet. These systems have gained prominence due to their flexibility, efficiency, and accessibility. Over the years, several researchers have explored different aspects of online examination systems, including security, scalability, user experience, and the use of emerging technologies.

1. E-exam Systems and Their Development

Ayo et al. (2007) discussed the benefits of e-exam systems in automating the traditional paper-based examination approach. These systems offer real-time results processing, improved reach, and significant administrative cost savings. However, challenges such as system reliability, question randomization, and cheating prevention remain key issues.

2. Security in Online Examinations

Mohammad, Sulaiman, and Shamsuddin (2014) explored the security vulnerabilities in online examination systems and proposed various authentication mechanisms. The study emphasizes the importance of multi-factor authentication, secure socket layers (SSL), and encryption techniques to maintain the integrity of exam data and user identity.

3. User Experience and Interface Design

Jain and Chawla (2015) highlighted that the effectiveness of an online examination system is greatly influenced by its user interface. A well-designed interface can enhance usability, reduce test anxiety, and ensure accessibility for users with different needs. Their study advocates for responsive design, clear navigation, and accessible layouts.

CHAPTER-3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Existing online examination portals have revolutionized the testing landscape by offering efficient and flexible alternatives to traditional paper-based exams. These systems enable real-time result processing, remote access, and streamlined administrative operations, making them popular in educational institutions and professional certification programs. However, they are not without drawbacks. Many platforms still struggle with ensuring robust security measures, which can lead to issues such as unauthorized access, data breaches, and impersonation. Additionally, technical glitches, inconsistent internet connectivity, and system downtimes can disrupt exam schedules and affect fairness. User interface challenges, such as non-intuitive navigation and inadequate support for diverse devices, further impede the overall user experience. Moreover, the heavy reliance on digital infrastructure can exacerbate problems during peak times, particularly in large-scale examinations, where system overload can compromise both performance and reliability.

Disadvantages of Existing Online Examination Portals

- **Security Vulnerabilities:** Susceptible to hacking, unauthorized access, and data breaches, which can compromise exam integrity and user privacy.
- **Technical Glitches and Reliability Issues:** System down times, server crashes, or performance bottlenecks, especially during peak usage, can disrupt the examination process.
- **Dependency on Stable Internet Connectivity:** A reliable internet connection is essential; any disruptions can affect exam continuity and fairness.
- **Scalability Limitations:** Handling a large number of concurrent users can strain the system, leading to delays and performance degradation.
- **Data Privacy Concerns:** Inadequate protection of sensitive information may lead to privacy issues and compliance risks.

3.2 PROPOSED SYSTEM

The Online Examination Portal is a web-based application designed to facilitate seamless examination management for educational institutions. The system provides a structured and secure approach to conducting exams, ensuring accessibility, efficiency, and automation in the examination process. The system comprises a Home Page with a Teacher Module, allowing teachers to log in and access their Dashboard. Within the dashboard, teachers have two primary modules:

1. **Number of Exams Module** – Displays the total exams created and conducted by the teacher.
2. **Exam Creation Module** – Enables teachers to create tests by entering essential test details such as subject, date, duration, and question types.

Once a test is created, the teacher can add users (students) to the test. The system automatically generates login credentials (username and password) for each student. Teachers can then distribute these credentials to students via email.

To enhance security and restrict unauthorized access, students must enter a unique College Code to gain access to scheduled tests. Teachers manage student access, control attendance, and monitor test performance. The system also incorporates automated result processing, ensuring accurate and instant score generation post-exam.

Advantages of the Proposed System

- **Automated Exam Management**: Reduces manual workload by enabling teachers to create, schedule, and manage exams efficiently.
- **Secure Student Authentication**: Implements college code-based access control to ensure that only authorized students can take the test.
- **Automated Credential Generation**: Eliminates manual effort in creating student login credentials by generating passwords automatically.
- **Efficient Attendance Management**: Teachers can monitor student participation and track absentees effectively.

- **Real-Time Result Processing**: Ensures immediate score calculation and report generation, reducing delays in result declaration.
- **User-Friendly Dashboard**: Provides an intuitive interface for teachers to manage exams, students, and test details.
- **Enhanced Communication**: Teachers can distribute login credentials and test details to students via email.
- **Scalability and Performance**: The system can handle multiple tests and users simultaneously without performance degradation.

3.3 FEASIBILITY STUDY

A feasibility study evaluates the practicality of developing and implementing the Project by assessing various aspects such as technical requirements, operational efficiency, cost-effectiveness, legal considerations, and project timeline. This study ensures that the project is both achievable and sustainable in real-world educational institutions.

Three key considerations involved in the feasibility analysis are

- **TECHNICAL FEASIBILITY**
- **ECONOMICAL FEASIBILITY**
- **OPERATIONAL FEASIBILITY**

1. Technical Feasibility

Technical feasibility assesses whether the required technology, software, and hardware resources are available to develop the system effectively. The Online Examination Portal is built using modern and scalable technologies to ensure efficiency and reliability. The frontend is developed using HTML, Tailwind CSS, and JavaScript, providing a user-friendly and responsive interface for seamless interaction. The backend is powered by Java with Spring Boot, which efficiently handles authentication, test creation, and database interactions. These technologies collectively ensure a robust and scalable platform capable of managing online examinations securely and effectively. Database Management: MySQL (to store user credentials, exam details, student responses, and results securely).

2. Economic Feasibility

The Online Examination Portal is a cost-effective solution that significantly reduces expenses associated with manual exam processing, paper-based assessments, and result generation. By eliminating the need for physical infrastructure such as exam halls, printed question papers, and invigilators, institutions can save considerable resources. Additionally, the system operates on cloud-based hosting, ensuring minimal maintenance costs while maintaining high availability and scalability. With automated test scheduling, result processing, and student authentication, the portal also reduces the workload for teachers and administrative staff, allowing them to focus on more critical academic activities and improving overall productivity.

3. Operational Feasibility

The Online Examination Portal simplifies the examination process for students, teachers, and administrators by offering a seamless and efficient digital platform. Teachers can easily create, schedule, and manage tests, reducing the complexities associated with traditional exam administration. Students can securely access their tests using college codes and automatically generated login credentials, ensuring controlled access and preventing unauthorized participation. Additionally, the system features automated result processing, which enhances efficiency by eliminating manual evaluation errors and providing instant feedback. With its intuitive and user-friendly interface, the platform ensures easy adoption for all stakeholders, making online examinations more accessible and efficient.

3.4 REQUIREMENT SPECIFICATION

The Project requires a combination of hardware and software resources to ensure smooth functionality, security, and scalability. The following specifications define the essential components needed for the successful implementation of the system.

Hardware Requirements

To support the system’s processing, storage, and hosting needs, the following hardware components are required:

- **Server**: A dedicated or cloud-based server to handle requests, authentication, and database operations.
- **Database Storage**: Sufficient storage capacity to store user details, exam records, results, and logs securely.
- **Web Hosting**: A reliable hosting solution for deploying the portal and ensuring uninterrupted access for users.
- **System : Inspiron i3, i5 Processor.**
- **Hard Disk : 500 GB.**
- **Input Devices : Keyboard, Mouse**
- **Ram : 4 GB**

Software Requirements

The system is built using a modern technology stack to ensure efficiency, scalability, and security. The key software requirements include:

Backend Development: Java with Spring Boot for handling business logic, authentication, and database operations.

Database Management: MySQL, a relational database system for storing and managing exam-related data securely.

Frontend Development: HTML, CSS, JavaScript, and Tailwind CSS to create a responsive, visually appealing, and user-friendly interface.

- **Operating system : Windows 10.**
- **Coding Language : Java 23.10.9.**
- **Web Framework : Spring-Boot.**

CHAPTER-4

SYSTEM STUDY

4.1 MODULE DESCRIPTION

This section provides a detailed explanation of the different modules within the Project. Each module is designed to ensure a smooth and efficient online examination process.

Home Module – Displays login and registration options for students, teachers, and administrators.

Teacher Module – Allows teachers to create, schedule, and manage tests while monitoring student participation.

Student Module – Enables students to access scheduled tests using a college code and auto-generated credentials.

Authentication Module – Ensures secure login and access management for users.

Exam Management Module – Handles test creation, question management, and automated result processing.

Admin Module – Provides administrative control over user data, test settings, and system configurations.

4.2 SYSTEM ARCHITECTURE

The system follows a three-tier architecture, ensuring scalability, security, and performance:

Presentation Layer (Frontend): Developed using HTML, CSS, JavaScript, and Tailwind CSS, providing a user-friendly interface.

Application Layer (Backend): Built with Java and Spring Boot, managing business logic, authentication, and test operations.

Database Layer: Uses MySQL to store user data, test details, questions, and results securely.

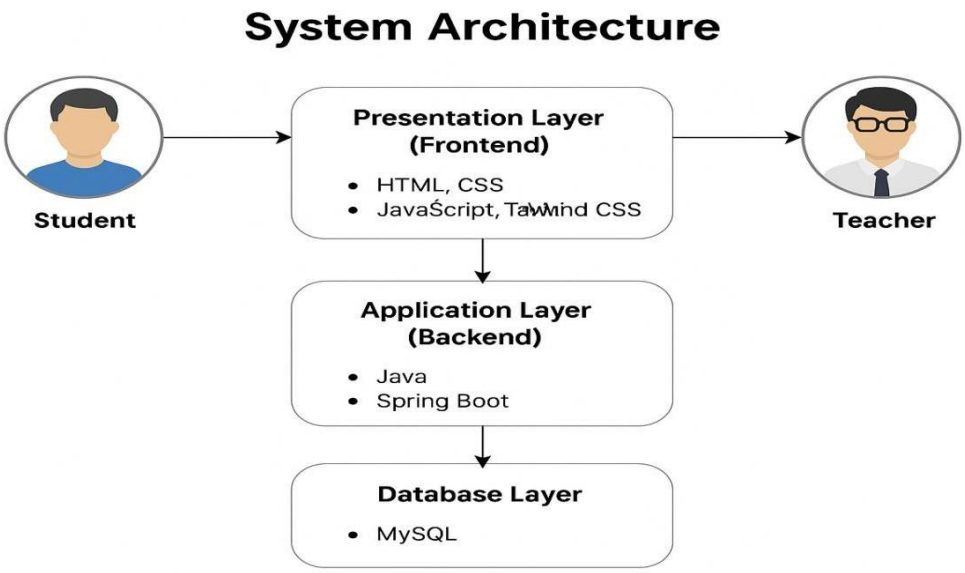

Fig 4.2: Architecture

4.3 DATA FLOW DIAGRAM (DFD)

The **Data Flow Diagram (DFD)** represents the flow of information within the Online Examination Portal. It includes:

Level 0 (Context Diagram): Shows the system's interaction with users (students, teachers, and admins).

Level 1: Displays how data flows between modules such as authentication, test management, and result processing.

Level 2: Provides a detailed breakdown of functionalities like test creation, user authentication, and report generation.

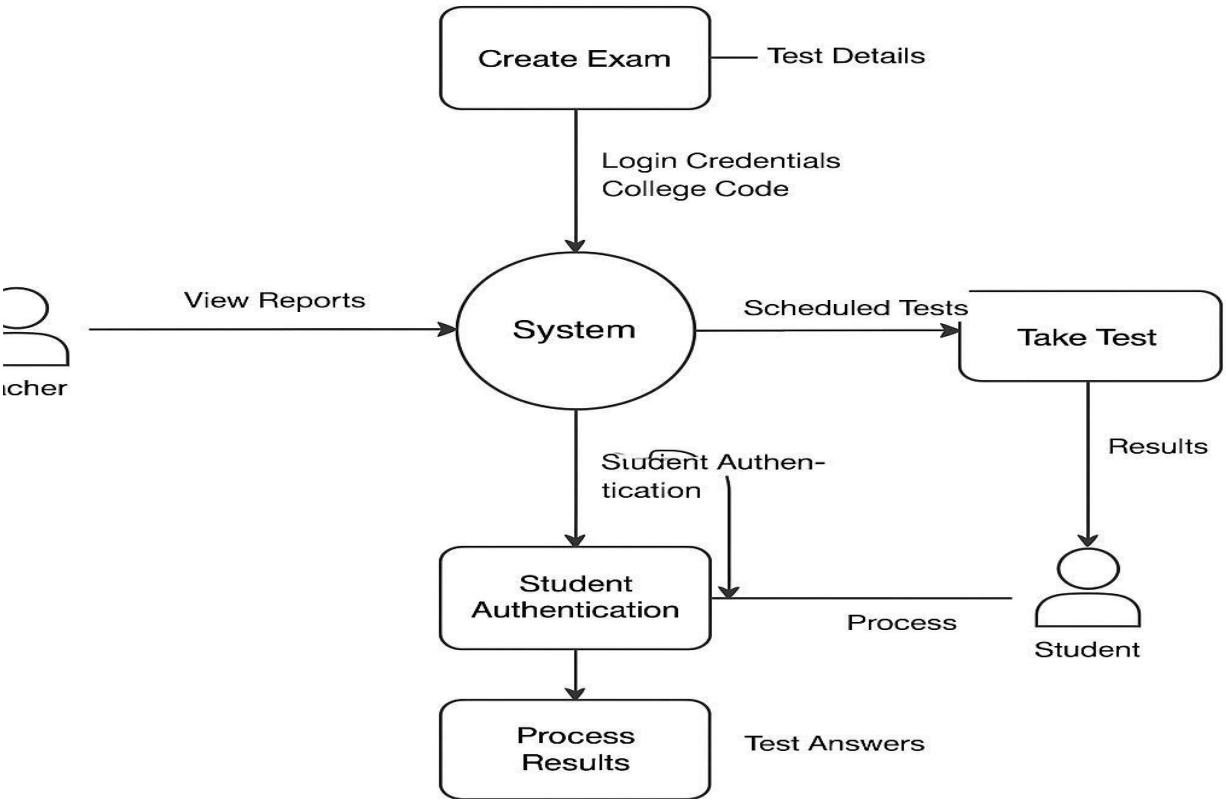

fig 4.3: Data Flow Diagram

4.4 UML DIAGRAMS

Unified Modeling Language (UML) is a standardized modeling language used to visualize, specify, construct, and document the design of a software system. It provides a set of diagrams that help developers, designers, and stakeholders understand the system's structure and behavior before implementation.

4.4.1 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

- This is a Use Case Diagram for a Online Eximanition Portal, illustrating the interactions between Students and Administrators with the system.

  Use Cases:

  1. Student Registration– Students register in the system.
  2. Manage Examination – Admin oversees exam scheduling and details.
  3. Manage Question Bank – Admin manages questions for exams.
  4. System Login – Both students and administrators log in.
  5. Test – Students attempt tests.
  6. Reports– Admin generates and manages reports.

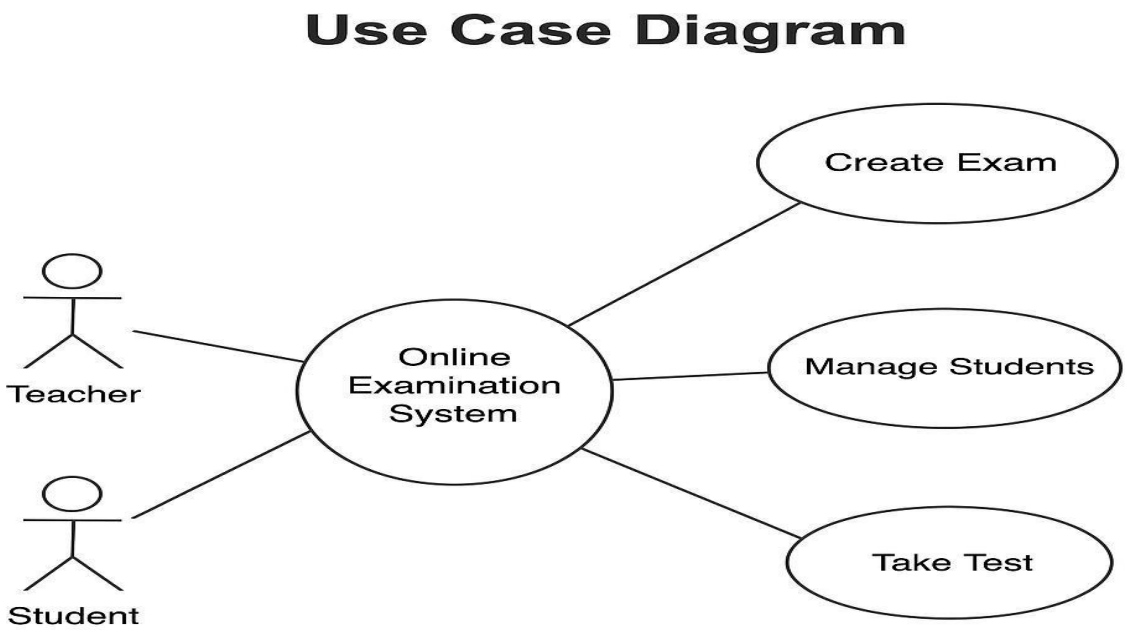

Fig 4.4.1: Use Case Diagram

4.4.2 CLASS DIAGRAM

A Class Diagram is a structural UML diagram that represents the blueprint of a system by showing the relationships between different classes, their attributes, and behaviors (methods). It is mainly used in object-oriented programming (OOP) and database design to define how various entities in a system interact.

- This is an Entity-Relationship Diagram (ERD) for an online examination system. It represents key entities such as Exam, User, Question, Option, Answer, and UserExam and their relationships.

  Exam: Contains details like title, description, start date, and question marks.

  User: Includes users like students and organizers.

  Question & Option: Each question has multiple options.

  Answer: Stores correct answers for validation.

  UserExam: Manages user participation in exams.

  UserAnswer: Stores users' submitted answers.

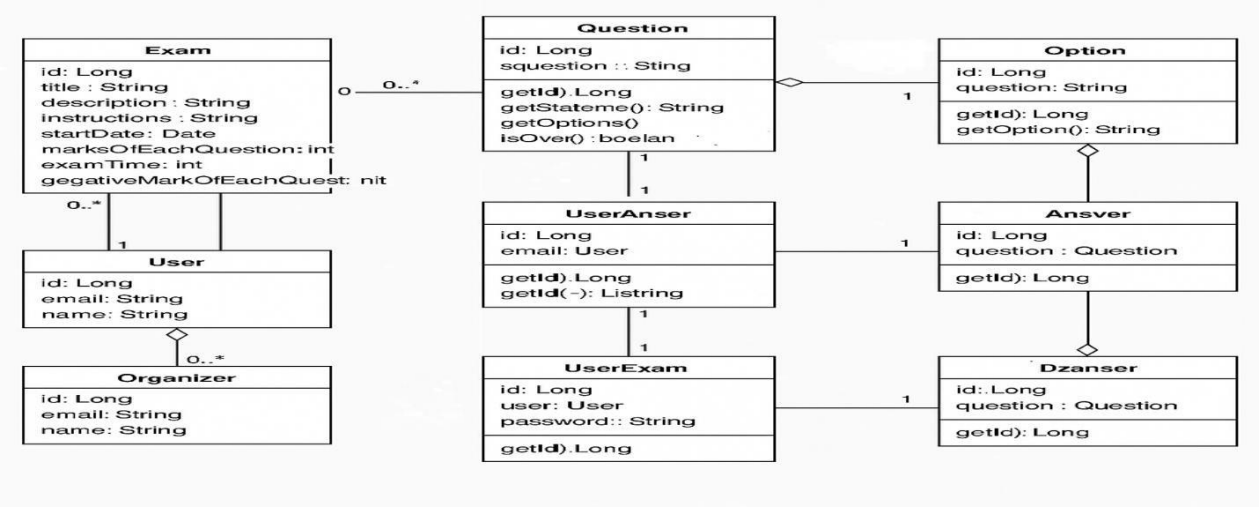

Fig 4.4.2: Class Diagram

4.4.3 SEQUENCE DIAGRAM

A Sequence Diagram is a type of UML interaction diagram that illustrates how objects interact with each other over time. It shows the sequence of messages exchanged between different components in a system from top to bottom, making it easier to understand the flow of operations.

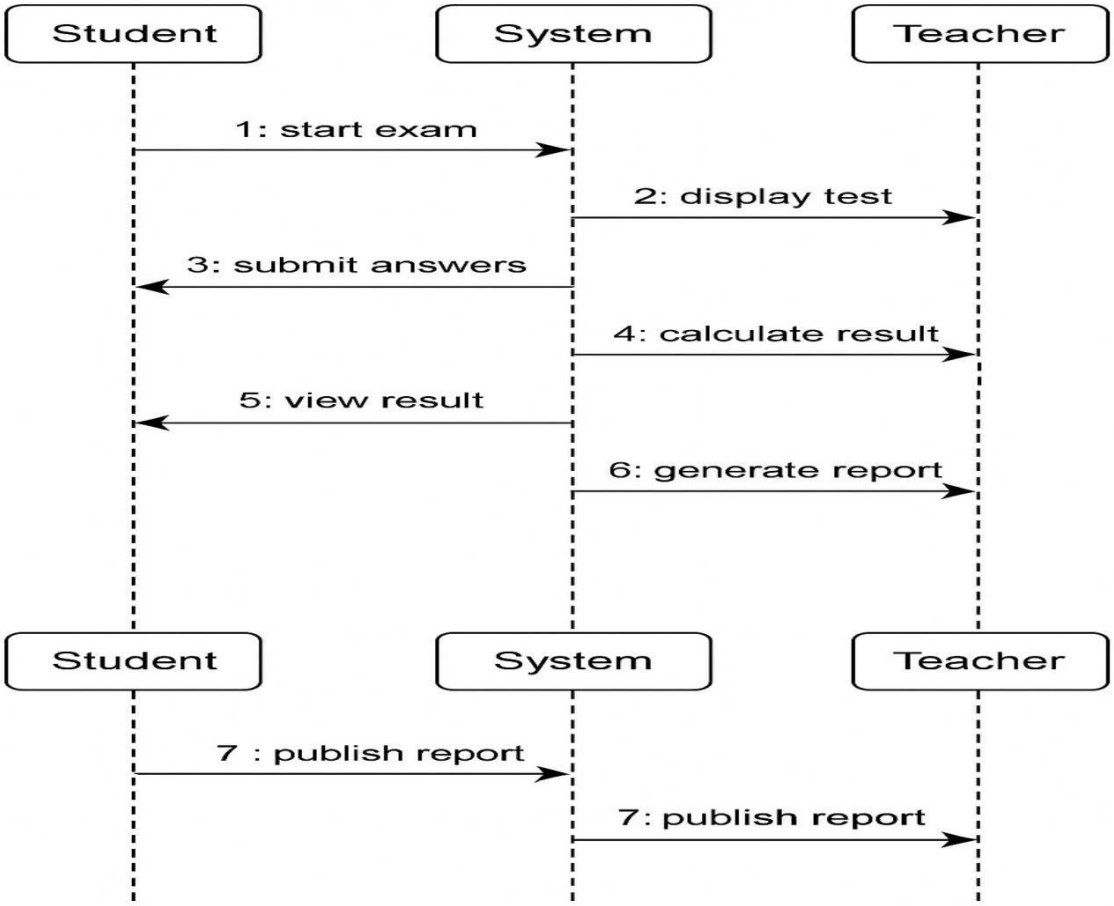

Fig 4.4.3: Sequence Diagram

4.4.4 ACTIVITY DIAGRAM

An Activity Diagram is a behavioral UML diagram that represents the workflow of a system by showing a sequence of activities, decision points, and transitions. It is used to model business processes, system operations, and user interactions in a graphical way.

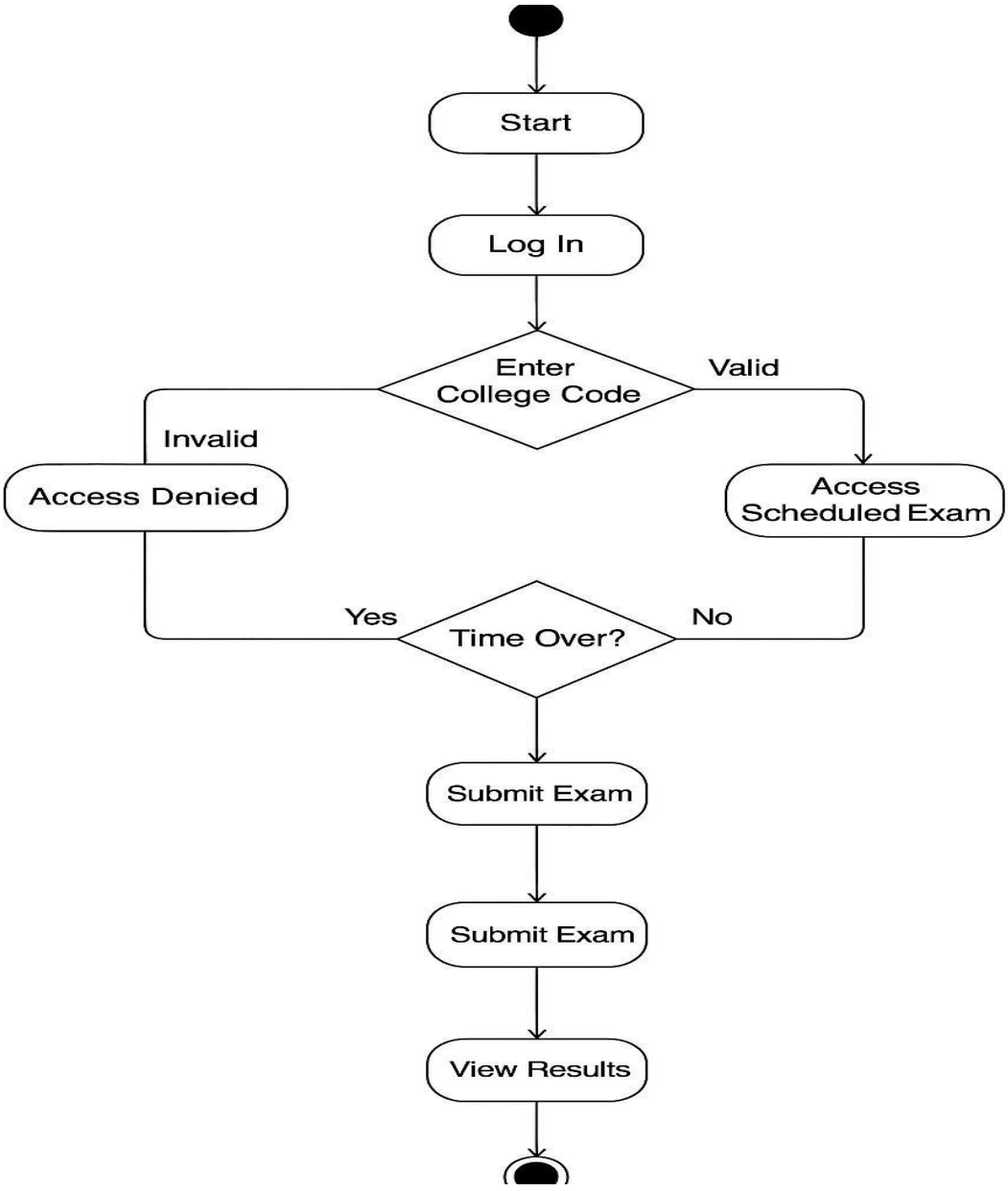

Fig 4.4.4: Activity Diagram

CHAPTER 5

SOFTWARE ENVIRONMENT

5.1 SOFTWARE ENVIRONMENT

The **software environment** defines the necessary platforms, tools, frameworks, and technologies used in the **Online Examination Portal** to ensure smooth development, deployment, and execution. Below is a detailed breakdown of the **software environment** for this project.

1. Operating System

The system is developed to run on **cross-platform environments**, ensuring flexibility and compatibility. The recommended operating systems include:

Windows 10/11 (Preferred for development and deployment)

Linux (Ubuntu, CentOS, Red Hat, etc.) (For hosting on cloud servers)

macOS (For development and testing purposes)

2. Programming Languages & Frameworks

The system is built using modern web development technologies that ensure **scalability, security, and performance**.

Frontend (User Interface Development)

- **HTML5** – Structure of the web pages
- **CSS3 & Tailwind CSS** – Styling for a responsive and modern UI
- **JavaScript** – Enhancing user interaction and validation

Backend (Server-Side Processing & API Development)

- **Java (JDK 17+)** – Core programming language for handling business logic
- **Spring Boot** – Framework for rapid and secure backend development
- **Spring Security** – For authentication and authorization of users
- **Hibernate ORM** – For seamless database interaction with Java

3. Database Management System (DBMS)

To store user credentials, exam details, student results, and logs securely.

- **MySQL 8.0+** – Relational database management system for structured data storage
- **JPA (Java Persistence API)** – For ORM mapping between Java objects and database tables
- **Spring Data JPA** – Simplifies data persistence and database queries

4. Web Server & Deployment Environment

The system is designed for **deployment on local and cloud-based servers**.

- **Apache Tomcat 9+** – Java-based web server for hosting the application
- **Spring Boot Embedded Server (Tomcat)** – To avoid external server configuration.

5. Development Tools & IDEs

To ensure a smooth development experience, the following tools are used:

- **Eclipse IDE / IntelliJ IDEA** – Java development environment
- **Visual Studio Code** – For frontend development and UI design
- **Postman** – For API testing and debugging
- **Maven** – Dependency and project management for Java

Java:

1. Introduction to Java

Java is a **high-level, object-oriented, and platform-independent programming language** developed by **Sun Microsystems** (now owned by Oracle Corporation) in **1995**. It is widely used for building secure, scalable, and platform-independent applications. Java follows the **"Write Once, Run Anywhere" (WORA)** principle, making it a top choice for developers across different industries.

2. Key Features of Java

1. Platform Independence

Java programs are compiled into **bytecode**, which runs on the **Java Virtual Machine (JVM)**. This allows Java applications to run on any operating system that has a JVM installed, making it platform independent.

2. Object-Oriented

Java follows **object-oriented programming (OOP) principles**, such as:

Encapsulation – Hiding implementation details from the user.

Inheritance – Allowing one class to inherit properties from another.

Polymorphism – Enabling multiple implementations using a single interface.

Abstraction – Simplifying complex real-world concepts in programming.

3. Robust and Secure

Java provides strong memory management, exception handling, and security features, such as:

Automatic Garbage Collection – Frees up memory by removing unused objects.

Exception Handling – Prevents runtime errors by handling exceptions properly.

Security Manager – Prevents unauthorized access to Java applications.

4. Multi-threading Support

Java supports **multithreading**, allowing multiple tasks to run concurrently. This improves efficiency and performance for complex applications.

5. Rich API and Libraries

Java provides a vast collection of libraries for handling **data structures, networking, file I/O, concurrency, and security**, making development faster and easier.

6. Scalability and Performance

Java is widely used for **enterprise-level applications** because of its ability to handle large-scale applications efficiently. It supports **distributed computing, microservices, and cloud-based applications**.

3. Java Architecture

1. Java Development Kit (JDK)

Contains **Java Compiler (javac), Java Runtime Environment (JRE), and libraries**.

Used for developing and compiling Java applications.

2. Java Runtime Environment (JRE)

Provides the runtime environment for Java applications.

Includes **JVM, class libraries, and configuration files**.

3. Java Virtual Machine (JVM)

Converts **Java bytecode into machine code** to run on different operating systems.

JVM is platform-dependent, but Java programs are platform-independent.

5.2 SOFTWARE SETUP

1. Java Installation Guide

Step 1: Download and Install Java

Go to the Oracle JDK website

Download **JDK (Java Development Kit)** for your OS.

Install the JDK and set up the environment variables (JAVA_HOME)

Verify installation:

```
java -version
```

2. Eclipse Installation Gide

Step 2: Download the Eclipse Installer

Download Eclipse Installer from http://www.eclipse.org/downloads

Select the package to install

Select your installation folder

Launch Eclipse

3. Spring Boot Setup

Spring Tool Suite (STS) – built on Eclipse

Download STS: https://spring.io/tools

4. Install Chocolatey

1. Open PowerShell as Administrator

- Click **Start**, search for **PowerShell.**
- Right-click and choose **"Run as administrator".**

2. Run the Installation Command

Paste the following command into the PowerShell window and press **Enter**:

```
Set-ExecutionPolicy Bypass -Scope Process -Force; `
[System.Net.ServicePointManager]::SecurityProtocol = `
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
choco install maven -y
```

5. Install with Chocolatey (Optional for Windows users)

If you have Chocolatey installed, you can install Maven with a single command:

bash

```
choco install maven
```

6. MySQL Installation Steps:

Step 1: Open chrome and search for MySQL

Step 2: Browse the official MySQL website.

Step 3: There you will find the Download option

Step 4: Click on MySQL Community (GPL) Downloads.

Step 5: Click on MySQL Installer for Windows

Step 6: Click on the second download link.

Step 7: After the download, open the installer.

Step 8: Now, choose the setup type. Here, select "Custom."

Step 9: The MySQL server, MySQL Workbench, and MySQL shell are ready to install.

Step 10: To download and install the MySQL server, MySQL Workbench, and MySQL shell, click the Execute button.

Step 11: Click Next once the product is ready for configuration.

Step 12: Use a strong password for authentication

Step 13: Set your password and click Next.

Step 14: Open the Windows service settings and click on Next.

Step 15: Apply the configuration and click Execute.

Step 16: Once the configuration is complete, click on Next.

Step 17: Once the installation is complete, this will now launch the MySQL Workbench and the MySQL Shell.

7. How to Runing

1. **In command prompt run this command**

```
mvn spring-boot:run
```

CHAPTER-6

SAMPLE CODE

CONTROLLERS

UserController.java

Repository & Model References

@Autowired UserRepository userRepo;

@Autowired ExamRepository examRepository;

@Autowired UserExamRepository userExamRepository;

@PostMapping("/user/csv/upload")

User Login Logic

Show how the system validates and logs in a user using exam code, email, and password.

```
@PostMapping("{examCode}/login")
public String loginUser(@PathVariable String examCode, @RequestParam String email, @RequestParam String password, Model model, HttpSession session) {
    // Check exam validity and time constraints...
    User user = userRepo.findByEmail(email);
    UserExam userExam = userExamRepository.findUserExamByUser(exam_id, user.getId());
    if (userExam != null && userExam.getPassword().equals(password)) {
        session.setAttribute("user_exam_id", userExam.getId());
        session.setAttribute("exam_id", exam_id);
        userExam.setStatus(1); // Mark Present
        userExamRepository.save(userExam);
```

```
        return "redirect:/" + examCode + "/instruction" }
        return "redirect:/" + examCode + "/login?error=1";}
```

Save User Answer

Show how a user's answer is saved or updated with correctness verification.

```
@PostMapping("{examcode}/submit")
public String saveAnswers(HttpSession session, @PathVariable String examcode, @RequestParam Long question_id, @RequestParam(required = false) Long answer_id) {
    UserExam userExam = userExamRepository.findById(user_id).get();
    Question question = questionRepository.findById(question_id).get();
    Option answer = optionRepository.findById(answer_id).get();
    boolean correct = question.getAnswer().getAnswer().getId().equals(answer.getId());
    UserAnswer userAnswer = new UserAnswer(userExam, question, answer, correct);
    userAnswerRepository.save(userAnswer);
    return "redirect:/" + examcode + "/exam/" + exam.getNextQuestionNo(question_id);
}
```

Exam Submission & Result Calculation

Show how final submission and result calculation works.

```
@GetMapping("{examcode}/final")
public String submitExam(@PathVariable String examCode, HttpSession session) {
    UserExam userExam = userExamRepository.findById(user_id).get();
    userExam.setStatus(2); // Mark as submitted
    userExamRepository.save(userExam);
    session.setAttribute("result", true);
```

```
    return "redirect:/" + examCode + "/result";
}
@GetMapping("/{examcode}/result")
public String showResult(@PathVariable String examCode, HttpSession session, Model model)
{
    int correct = userAnswerRepository.findCorrectAnswersCount(user_id);
    int incorrect = userAnswerRepository.findInCorrectAnswersCount(user_id);
    int score = exam.calculateScore(correct, incorrect);
    model.addAttribute("correctAnswers", correct);
    model.addAttribute("incorrectAnswers", incorrect);
    model.addAttribute("score", score);
    return "user/result";
}
```

QuestionController.java

Repository & Model References

```
@Autowired ExamRepository examRepo;
@Autowired QuestionRepository repo;
@Autowired OptionRepository optionRepo;
@Autowired AnswerRepository answerRepo;
```

Handles CRUD operations for exam questions, including adding, editing, and deleting questions and their corresponding options and answers.

Add Question Functionality

```
@PostMapping("/organiser/question/add")
```

```
public String addQuestion(@RequestParam(name = "question") String question,
                          @RequestParam(name = "option[]") List<String> options,
                          @RequestParam(name="exam_id") Long exam_id,
                          @RequestParam(name = "answer") Integer answer) {
    // Logic to save question, options, and correct answer
}
```

Edit Question Functionality

```
@PostMapping("/organiser/question/edit")
public String editQuestion(@RequestParam(name = "question") String question,
                           @RequestParam(name = "option[]") List<String> options,
                           @RequestParam(name="question_id") Long question_id,
                           @RequestParam(name = "answer") Integer answer) {
    // Logic to update question and associated options/answer
}
```

Delete Question Functionality

```
@GetMapping("/organiser/question/delete")
public String deleteQuestion(@RequestParam(name = "question_id") Long question_id,
                             @RequestParam(name="exam_id") Long exam_id) {
    // Logic to remove question, options, and answer
}
```

OrganiserController.java

Repository & Model References

```
@Autowired OrganiserRepository repo;

@Autowired ExamRepository examRepository;
```

1. Organiser Registration Page

```
@GetMapping("organiser/register")

public String register(Model model) {

    model.addAttribute("organiser", new Organiser());

    return "organiser/register";}
```

2. Handle Organiser Registration Submission

```
@PostMapping("organiser/register")

public String registerPost(Organiser org) {

    BCryptPasswordEncoder encoder = new BCryptPasswordEncoder();

    String encodedPassword = encoder.encode(org.getPassword());

    org.setPassword(encodedPassword);

    repo.save(org);

    return "redirect:/organiser/dashboard";}
```

3. Organiser Login Page

```
@GetMapping("organiser/login")

public String login() {

    Object user = SecurityContextHolder.getContext().getAuthentication().getPrincipal();

    if (user instanceof OrganiserDetails) {
```

```
        return "redirect:/organiser/dashboard";}
    return "organiser/login";}
```

4. Organiser Dashboard

```
@GetMapping("organiser/dashboard")
public String dashboard(Model model) {
    int examcount = 0;
    Object user = SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    if (user instanceof OrganiserDetails) {
        Organiser org = ((OrganiserDetails) user).getOrg();
        examcount = examRepository.findByOrganiserId(org.getId()).size();
    }
    model.addAttribute("examcount", examcount);
    return "organiser/dashboard";
}
```

ExamController.java

Repository & Model References

```
@Autowired private JavaMailSender javaMailSender;
@Autowired ExamRepository repo;
@Autowired UserAnswerRepository userAnswerRepository;
@Autowired UserExamRepository userExamRepository;
```

1. Display All Exams for Logged-in Organiser

```
@GetMapping("/organiser/exams")
```

```
public String showExams(Model model) {
    Object user = SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    if (user instanceof OrganiserDetails) {
        Organiser org = ((OrganiserDetails) user).getOrg();
        model.addAttribute("exams", repo.findByOrganiserId(org.getId()));
        return "organiser/exam/list";
    } else {
        return OrganiserController.LOGIN_ROUTE;
    }
}
```

2. Create New Exam

```
@PostMapping("/organiser/exams/create")
public String createExam(Exam exam) {
    Object user = SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    if (user instanceof OrganiserDetails) {
        exam.setOrganisers(((OrganiserDetails) user).getOrg());
        repo.save(exam);
        return "redirect:/organiser/exams";
    } else {
        return OrganiserController.LOGIN_ROUTE;
    }
}
```

3. Edit Existing Exam

java

CopyEdit

```
@GetMapping("/organiser/exams/edit")
public String editExam(@RequestParam(name = "id") Long exam_id, Model model) {
    Exam oldExam = repo.findById(exam_id).get();
    model.addAttribute("oldExam", oldExam);
    return "organiser/exam/edit";
}
```

4. View Exam Result Summary

java

CopyEdit

```
@GetMapping("/organiser/exams/result")
public String viewResult(@RequestParam(name = "id") Long id, Model model) {
    Exam exam = repo.findById(id).get();
    List<UserExam> examUsers = exam.getUserExam();
    int presentCount = userExamRepository.findPresentUsersCount(exam.getId());
    int adbsetCount = userExamRepository.findAbsentUsersCount(exam.getId());
    HashMap<Long, Integer> correctAnswers = new HashMap<>();
    HashMap<Long, Integer> incorrectAnswers = new HashMap<>();
    HashMap<Long, Integer> score = new HashMap<>();
    for (UserExam userExam : examUsers) {
        int correct = userAnswerRepository.findCorrectAnswersCount(userExam.getId());
```

```
        int incorrect = userAnswerRepository.findInCorrectAnswersCount(userExam.getId());
        correctAnswers.put(userExam.getId(), correct);
        incorrectAnswers.put(userExam.getId(), incorrect);
        score.put(userExam.getId(), exam.calculateScore(correct, incorrect));
    }
    model.addAttribute("exam", exam);
    model.addAttribute("examUsers", examUsers);
    model.addAttribute("correctAnswers", correctAnswers);
    model.addAttribute("incorrectAnswers", incorrectAnswers);
    model.addAttribute("score", score);
    model.addAttribute("presentCount", presentCount);
    model.addAttribute("adbsetCount", adbsetCount);
    return "organiser/result/list";
}
```

App Controller: This controller handles routing to general pages like the homepage, about page, and contact page.

Returns the index.html view when users access the root URL (/).

```
@GetMapping("")
public String index() {  return "index";}
```

Maps the /about URL to the about.html view.

```
    @GetMapping("/about")
    public String about() { return "about";}
```

Maps /contact-us to the contact.html view.

```
@GetMapping("/contact-us")
public String contactUs() {  return "contact";}
```

Models

UserExam.java

1. Entity & Table Declaration

Use this to show how the UserExam entity is mapped to the database:

```
@Entity
@Table(name = "user_exams")
public class UserExam {
   // Fields and relationships}
```

2. Primary Key and Fields

```
@Id
@GeneratedValue(strategy=GenerationType.IDENTITY)
private Long id;
@Column(name = "password")
private String password;
@Column(name = "status", nullable = false)
private int status;
```

3. User Relationship

```
@ManyToOne(targetEntity = User.class)
@JoinColumn(name= "user_id", nullable=false)
private User user;
```

4. Exam Relationship

```
@ManyToOne(targetEntity = Exam.class)
@JoinColumn(name= "exam_id", nullable=false)
private Exam exams;
```

5. Sets up the relationship and initializes exam participation status.

```
public UserExam(User user, Exam exam, String password) {
   this.exams = exam;
   this.user = user;
   this.password = password;
   this.status = 0; // default to "not started"
}
```

UserAnswer.java

Links this entity to the user_answer table in the database using JPA.

```
@Entity
@Table(name = "user_answer")
public class UserAnswer {
   // Fields and mappings }
```

The id uniquely identifies the answer, while answer_status shows if the user's answer is correct (true) or incorrect (false).

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
@Column(name = "answer_status", nullable = false)
```

private Boolean answer_status;

Question & Option Relationships

Mapping to the Question

Links each user answer to both the question and the selected option.

@OneToOne

@JoinColumn(name = "queston_id", nullable = false)

private Question questions;

Mapping to the Chosen Option

@OneToOne

@JoinColumn(name = "option_id", nullable = false)

private Option answer;

Mapping to UserExam

@OneToOne

@JoinColumn(name = "user_id", nullable = false)

private UserExam userexam;

Associates the answer with a specific user's attempt at an exam.

Answer Status Setter

```
public void setAnswerStatus(boolean answerStatus) {
    this.answer_status = answerStatus;
}
```

Useful to record the result of automatic answer evaluation (e.g., marking answers after submission).

User.java

Entity and Table Declaration

Maps this class to the users table in your SQL database using JPA.

```
@Entity
@Table(name = "users")
public class User {
    // Fields and logic
}
```

Primary Key Field

Auto-generates a unique ID for every user entry.

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
```

User Attributes

```
@Column(nullable = false, unique = true, length = 50)
private String email;
@Column(nullable = false, length = 50)
private String name;
```

Constructors

Allows creation of User objects manually or via frameworks like Hibernate.

```
public User(String email, String name) {
    this.email = email;
    this.name = name;
}
```

```
public User() {
    // Default constructor
}
```

Getters and Setters

Standard JavaBean-style methods for field access and modification.

```
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 getName() { return name; }
public void setName(String name) { this.name = name; }
```

Question.java

1. Entity Structure

```
@Entity
@Table(name = "questions")
public class Question {
    // class fields and methods
}
```

Maps the class to the questions table in the database using JPA.

2. Primary Key

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
private Long id;
```

Auto-generates a unique ID for each question.

3. Question Statement

```
@Column(nullable = false, length = 300)
private String statement;
```

Stores the actual text of the question with a character limit.

4. Exam Association

```
@ManyToOne
@JoinColumn(name = "exam_id", nullable = false)
private Exam exams;
```

Many questions belong to one exam.

5. Options Relationship

```
@OneToMany(mappedBy = "questions")
private List<Option> options = new ArrayList<>();
```

Defines a one-to-many relationship between questions and their possible answer options.

6. Correct Answer Reference

```
@OneToOne(mappedBy = "questions")
private Answer answer;
```

Points to the correct answer for the question (optional for multiple-choice type systems).

7. Utility Method

```
public String getBaseEncodedId() {
   return Base64.getEncoder().encode(new byte[]{this.id.byteValue()}).toString();
}
```

Generates a Base64-encoded version of the question ID, useful for obscuring raw IDs in URLs or APIs.

Organiser.java

Entity Declaration

@Entity

```
@Table(name = "organisers")
public class Organiser implements Serializable {
    // class fields and methods
}
```

Purpose: Maps this class to the organisers table in the database using JPA and enables serialization.

2. Primary Key

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
```

Purpose: Auto-generates a unique ID for each organiser.

3. Attributes

```
@Column(nullable = false, unique = true, length = 50)
private String email;
@Column(nullable = false, length = 50)
private String name;
@Column(nullable = false, length = 250)
private String password;
```

Purpose: Stores the organiser's email (must be unique), name, and password.

4. Exam Relationship

```
@OneToMany(mappedBy = "organisers")
private List<Exam> exams = new ArrayList<>();
```

Purpose: Defines a one-to-many relationship — one organiser can create many exams.

5. Getter and Setter Methods

```
public String getEmail() { return email; }
public void setEmail(String email) { this.email = email; }
public String getName() { return name; }
public void setName(String name) { this.name = name; }
public String getPassword() { return password; }
public void setPassword(String password) { this.password = password; }
public List<Exam> getExams() { return exams; }
```

Purpose: Provides access to class fields and allows external updates to data.

Options.java

1. Entity Declaration

```
@Entity
@Table(name = "options")
public class Option {
    // class fields and methods
}
```

Purpose: Defines this class as a persistent entity mapped to the options table.

2. Primary Key

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
```

Purpose: Auto-generates a unique identifier for each option.

3. Relationship with Question

```
@ManyToOne
@JoinColumn(name = "queston_id", nullable = false)
private Question questions;
```

Purpose: Each option is linked to one question, establishing a many-to-one relationship.

4. Option Text

```
@Column(length = 50, name = "statement")
private String option;
```

Purpose: Stores the text of the option (e.g., “A. 42”).

5. Constructors

```
public Option(Question questions, String option) {
   this.questions = questions;
   this.option = option;
}
public Option() {}
```

Purpose: Provides a parameterized constructor for easy instantiation and a default constructor required by JPA.

6. Getter and Setter Methods

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

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

public Question getQuestions() { return questions; }

public void setQuestions(Question questions) { this.questions = questions; }

public String getOption() { return option; }

public void setOption(String option) { this.option = option; }
```

Purpose: Provides controlled access and mutation of fields.

Exam.java

1. Entity & Table Annotation

```
@Entity

@Table(name = "exams")

public class Exam {

    // fields and methods

}
```

Purpose: ***Marks this class as a persistent entity mapped to the exams table.***

2. Primary Key

```
@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;
```

Purpose : Automatically generates a unique ID for each exam.

3. Exam Metadata

```
@Column(nullable = false, length = 50, name = "title")
private String title;
@Column(length = 100, name = "description")
private String description;
@Column(length = 150, name = "instructions")
private String instructions;
```

Purpose :Stores basic information like title, description, and rules.

4. Exam Timing

```
@DateTimeFormat(pattern = "dd/MM/yyyy h:mm a")
@Column(nullable = false, name = "start_date")
private Date startDate;
@Column(nullable = false, length = 5, name = "time")
private int examTime;  // in minutes
```

Purpose : Handles start time and total duration.

5. Scoring Scheme

```
@Column(nullable = false, length = 5, name = "marks")
private int marksOfEachQuestion;
@Column(length = 5, name = "negative_marks", nullable = true)
private int negativeMarkOfEachQuestion;
```

Purpose : Defines marks for correct and incorrect answers.

6. Relationships

@ManyToOne

@JoinColumn(name = "organiser_id", nullable = false)

private Organiser organisers;

@OneToMany(mappedBy = "exams")

private List<Question> questions = new ArrayList<>();

@OneToMany(mappedBy = "exams")

private List<UserExam> userExam;

Purpose: Links the exam to its organiser, list of questions, and participating users.

7. Business Logic Snippets

Generate Exam Code

public String getExamCode() {

// Creates short unique exam code from title

}

8. Exam Status Check

public boolean isStarted() { ... }

public boolean isOver() { ... }

public String getExamStatus() { ... }

public long getRemainingTime() { ... }

Purpose :These methods check whether the exam is running, finished, or yet to begin.

9. Scoring Calculation

public int calculateScore(int correctAnswers, int incorrectAnswer) {

// Calculates score based on correct and wrong answers}

10. Question Navigation

public int getNextQuestionNo(Long question_id) { ... }

public boolean isLastQuestion(Long question_id) { ... }

Purpose: Helps navigate between questions during the exam.

Answers.java

1. Entity & Table Mapping

@Entity

@Table(name = "answers")

public class Answer {

    // fields and methods

}

Purpose : Maps this class to the answers table in the database.

2. Primary Key

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

Purpose :Auto-generated primary key for each answer record.

3. Relationships

Connects to Question

@OneToOne

@JoinColumn(name = "queston_id", nullable = false)

private Question questions;

Connects to Correct Option

```
@OneToOne
@JoinColumn(name = "option_id", nullable = false)
private Option option;
```

Purpose :Each Answer is uniquely linked to one Question and one correct Option.

4. Constructors

```
public Answer() {}
public Answer(Question questions, Option option) {
    this.questions = questions;
    this.option = option;
}
```

Purpose :Allows creation of answer objects with or without initialization.

5. Getter & Setter Methods

```
public Long getId() { return id; }
public void setId(Long id) { this.id = id; }
public Question getQuestions() { return questions; }
public void setQuestions(Question questions) { this.questions = questions; }
public Option getAnswer() { return option; }
public void setAnswer(Option answer) { this.option = answer; }
```

Purpose :Encapsulation for accessing and modifying fields.

DATABASE

```
mysql> use examportal;
Database changed
mysql> show tables;
+---------------------------+
| Tables_in_examportal      |
+---------------------------+
| answers                   |
| exams                     |
| options                   |
| organisers                |
| questions                 |
| spring_session            |
| spring_session_attributes |
| user_answer               |
| user_exams                |
| users                     |
+---------------------------+
10 rows in set (0.06 sec)
```

Fig 6.1 :Database

```
mysql> select * from users;
+----+---------------------------+-----------------+
| id | email                     | name            |
+----+---------------------------+-----------------+
|  1 | laxmi@gmail.com           | Laxmi           |
|  2 | suman95@gmail.com         | Suman Sharma    |
|  3 | sumitjha12@gmail.com      | Sumit Jha       |
|  4 | kavsin22@gmail.com        | Kavya Sinha     |
|  5 | pooja@gmail.com           | Pooja Meena     |
|  6 | kapil43@gmail.com         | Kapil           |
|  7 | meghna0102@gmail.com      | Meghna          |
|  8 | Surendra23199@gmail.com   | Surendra Tomar  |
|  9 | kishangopal@gmail.com     | Kishan Gopal    |
| 10 | dr1999@gmail.com          | Devendra Rane   |
| 11 | tripathi3333@gmail.com    | Somya Tripathi  |
| 12 | safal5499@gmail.com       | Safalta         |
| 13 | kumaripriya@gmail.com     | Priya Kumari    |
| 14 | solanki34@gmail.com       | Pushpa solanki  |
| 15 | jitendra2000@gmail.com    | Jitendra Singh  |
| 16 | karrtikshah1996@gmail.com | Kartikey        |
| 17 | rahul@gmail.com           | rahul           |
| 18 | srivalli.snsd@gmail.com   | srivalli        |
| 19 | h@gmail.com               | hemanth         |
| 20 | t@gmail.com               | thusara         |
| 21 | m@gmail.com               | maneesha        |
| 22 | suma@gamil.com            | Suma            |
+----+---------------------------+-----------------+
22 rows in set (0.03 sec)
```

Fig 6.2 :UsersTable

CHAPTER-7

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub- assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. it is done after the completion of an individual unit before integration. This is structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified, and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box:

Testing Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot "see" into it. The test provides input and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually, and functional tests will be written in detail. **Test objectives**

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing:

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER - 8

SCREENSHORTES

8.1 Student Access

1. This field is for students to enter their unique **exam code** to access their specific exam.

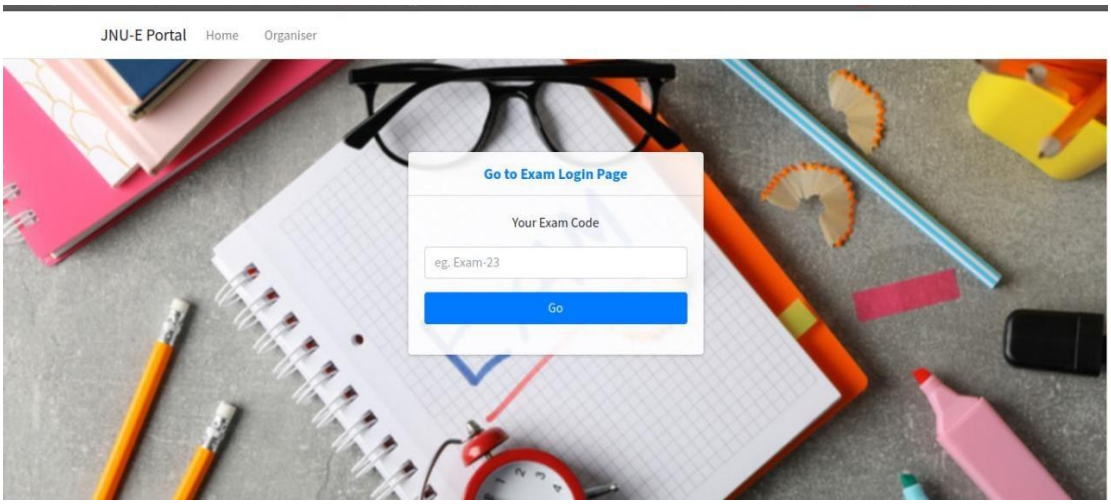

Fig 8.1.1 Exam Code Page

2. First it validates the exam code entered by the student and then redirect to this screen allows a student to login and access the exam by entering the username and password (i.e., automatically generate).

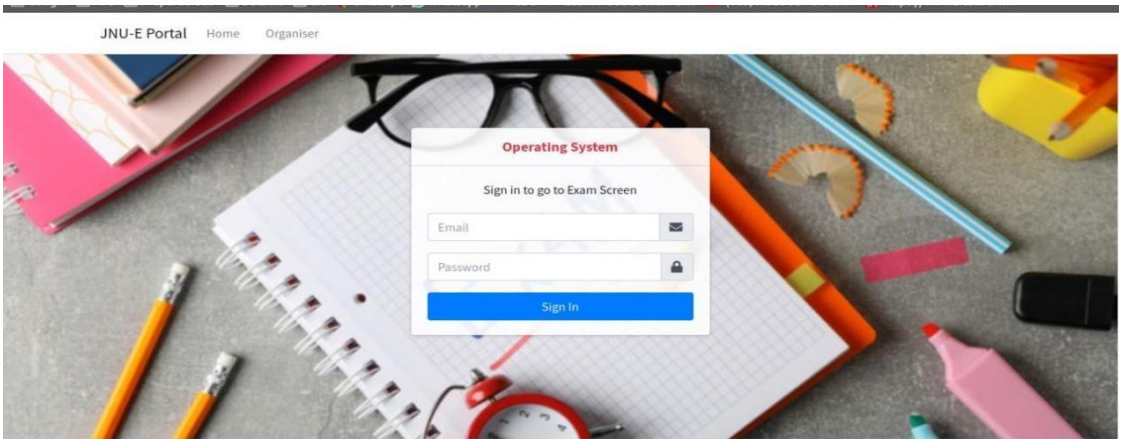

Fig 8.1.2Student Login Page

3. If student's credentials are valid then exam will be started automatically. As per schedule time by the admin. And here student can reset the answer, and if student wants to save the answer by clicking on save and next they will get next question. On left side student can be able to see no: of questions the answered and not answered.

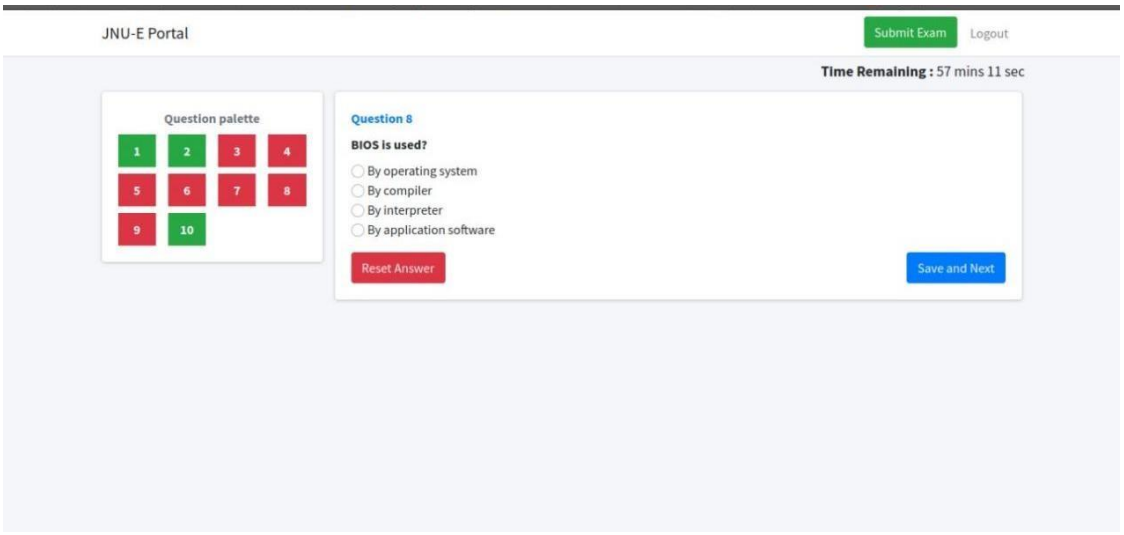

Fig 8.1.3 Exam Page

4. By Clicking on Submit exam, instantly student can be able to get their score.

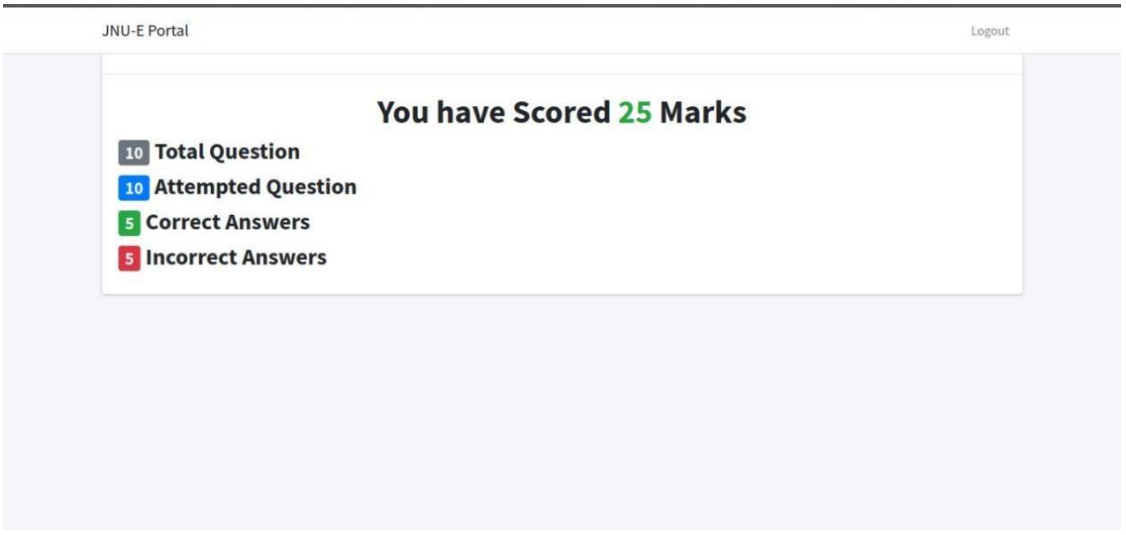

Fig 8.2.4 Score Card

8.2 Admin Access

1. Admin have access to create account and login. By clicking on Register a new membership. Admin can have an account on this portal.

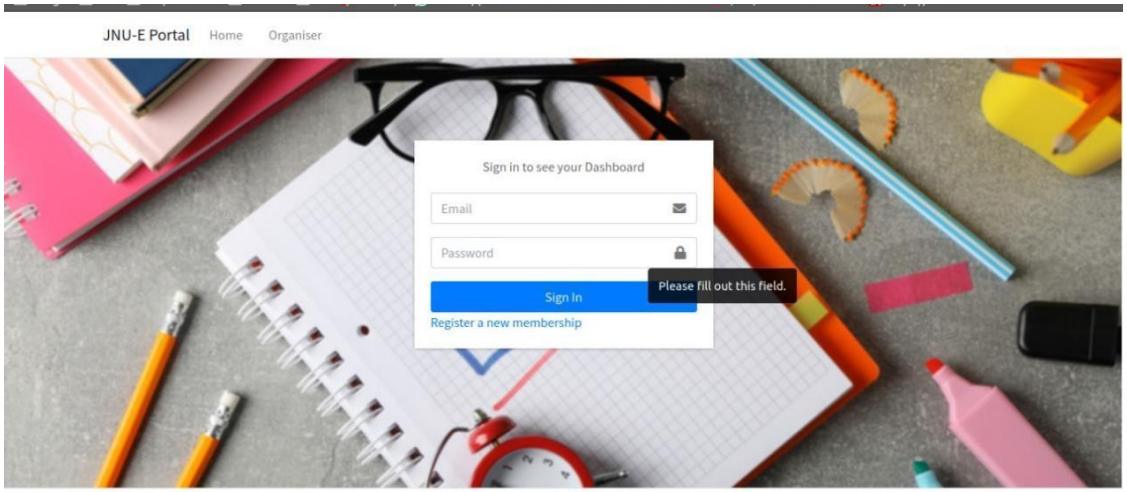

Fig 8.2.1 Admin Login Page

2. Admin has Dashboard after registration and have their own credentials. With those credentials admin can access the portal.

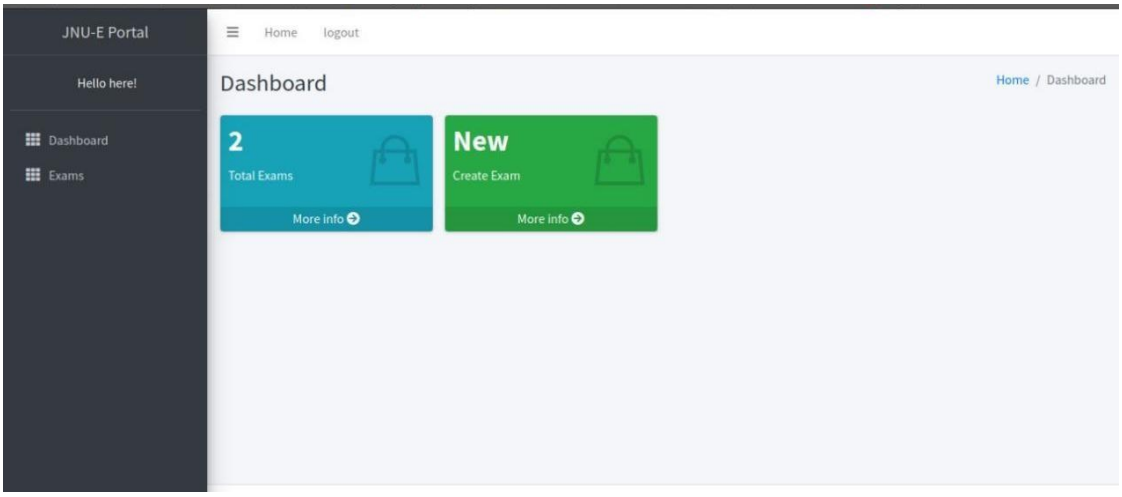

Fig 8.2.2 Dashboard

Here, in Dashboard admin can check exam details, no: of students attempted exam and absentees count, student's exam repots. And can be able to create new exams.

3. Admin can create exam. By their own customized Questions and set the exam time, marks for the questions and the negative makes can also give by the admins. And can be able to add students.

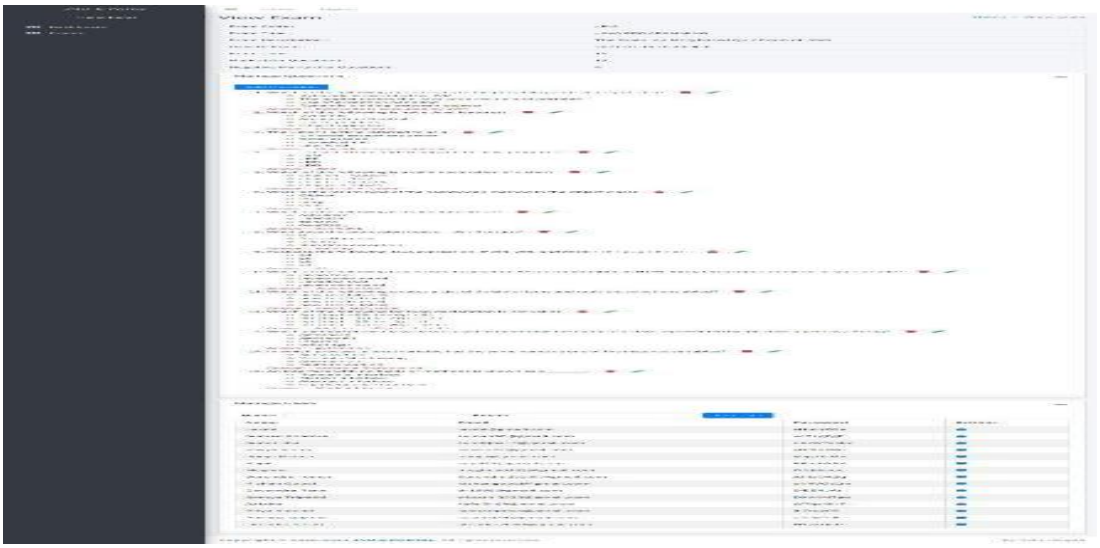

Fig 8.2.3 Exam Creation Page

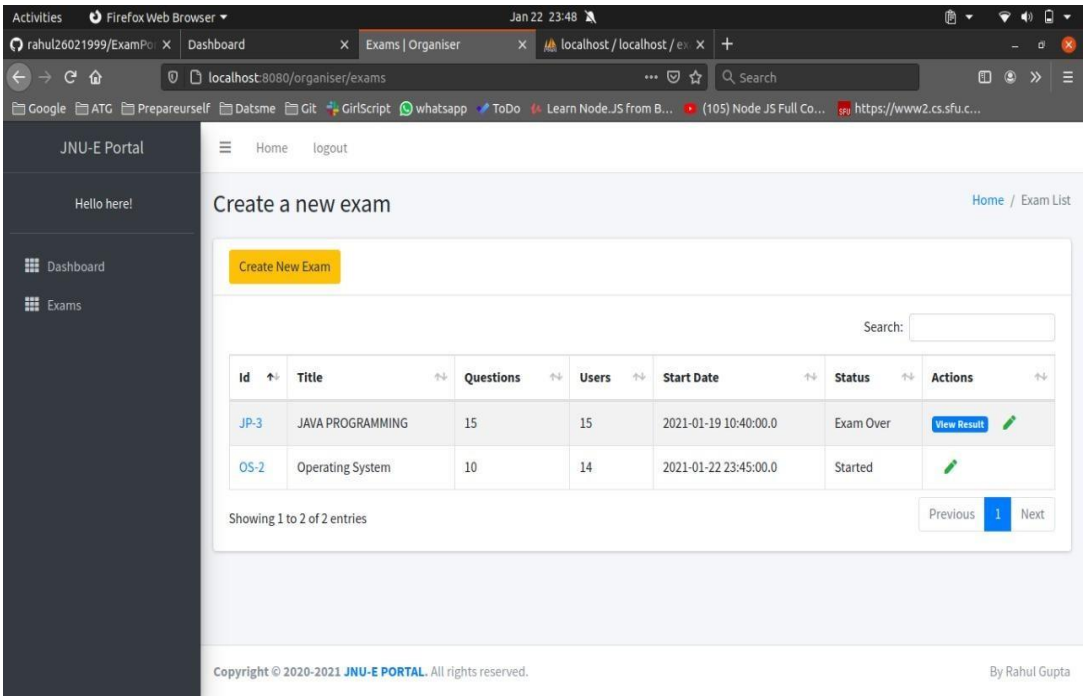

Fig 8.2.4 Available Exams Page

CHAPTER-9

CONCLUSION

The **Online Examination Portal** is a robust, user-friendly platform designed to streamline the examination process for both students and administrators. It provides a **secure, accessible, and efficient environment** to conduct online exams with features such as:

- **Exam Code Entry System**:
- Each exam is associated with a unique code, allowing students to quickly and accurately access their respective tests. This feature ensures simplicity and reduces the chances of students accessing the wrong exam.
- **Student Login Authentication**
- A secure login system verifies each student's identity before granting access to the examination. This not only helps prevent impersonation but also ensures that the right individual takes the right exam, maintaining academic integrity.
- **Centralized Exam Management**
- Administrators can create, schedule, assign, and monitor exams from a centralized dashboard. This feature simplifies the management of multiple exams, tracks student participation, and provides real-time monitoring capabilities for better control and supervision.

By digitizing the traditional exam process, this portal reduces administrative overhead, minimizes paper usage, and enables remote examination capabilities — which is especially valuable in scenarios like distance learning or during global disruptions (e.g., pandemics).

Overall, this portal empowers institutions with a **modern, scalable solution** for managing examinations, while offering students a **convenient and transparent** exam experience.

The Online Examination Portal transforms the traditional examination approach into a modern, digital experience. It offers a secure, efficient, and transparent way to manage and conduct exams, empowering institutions to uphold academic standards while embracing technology.

CHAPTER-10

BIBLIOGRAPHY

1. Singh Dipendra, Vishal Singh Kahlon, Kavyashree N, “Online Examination Portal” Published in International Research Journal of Innovations in Engineering and Technology - IRJIET, Volume 6, Issue 6, pp 202-204, June 2022. Article DOI https://doi.org/10.47001/IRJIET/2022.606028
2. "ONLINE EXAMINATION SYSTEM", International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056, Volume: 0,3Issue: 01| Jan-2016.
3. "Challenges of Online Exam, Performances and problems for Online University Exam", International Research Journal of Engineering and Technology IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 1, No 1, January 2013 ISSN (Print): 1694-0784 | ISSN (Online): 1694-0814.
4. "Online Descriptive Examination and Assessment System", International Journal of Emerging Technology and Advanced Engineering, (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 4, Issue 3, March 2014).
5. The Design and Implementation of On-Line Examination Using Firewall security, IOSR Journal of Computer Engineering (IOSR-JCE) e -ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 16, Issue 6, Ver. V (Nov –Dec. 2014), PP 20-24.
6. "Research and Development of Online Examination System", Proceedings of the 2012 2nd International Conference on Computer and Information Application (ICCIA 2012).
7. Web Based online Secured Exam; B. Persis Urbana Ivy, A. Shalini, A Yamuna, International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 2, Issue 1, Jan - Feb 2012, pp. 943-944.