

# **EXAM PORTAL**

## **Submitted By:**

**Sidharth Singh (20BCB0014)**- [sidharth.singh2020@vitstudent.ac.in](mailto:sidharth.singh2020@vitstudent.ac.in)

**Deepanshu Upadhyay (20BCB0147)**- [deepanshu.upadhyay2020@vitstudent.ac.in](mailto:deepanshu.upadhyay2020@vitstudent.ac.in)

**Md Danish Anwar (20BCE0106)**- [mddanish.anwar2020@vitstudent.ac.in](mailto:mddanish.anwar2020@vitstudent.ac.in)

**Mohammad Samme (20BCB0046)**- [mohammad.samme2020@vitstudent.ac.in](mailto:mohammad.samme2020@vitstudent.ac.in)

## **Project Description:**

The Exam Portal is a web-based application developed using Spring Boot as the backend framework, AngularJS as the frontend framework, and MySQL as the database management system. It aims to provide a user-friendly platform for conducting and managing online exams and assessments.

## **Key Features:**

- **User Authentication and Authorization:** The Exam Portal allows users to register, login, and manage their accounts. User authentication and authorization mechanisms ensure secure access to the system. User information is stored in the MySQL database.
- **Exam Management:** The system provides functionality for administrators to create and manage exams. They can define the exam structure, set question types (multiple choice, descriptive, etc.), specify time limits, and manage exam categories. Exam data, including exam details and structure, is stored in the MySQL database.
- **Question Bank:** The Exam Portal includes a question bank where administrators can add, edit, and delete questions. Questions can be categorized based on

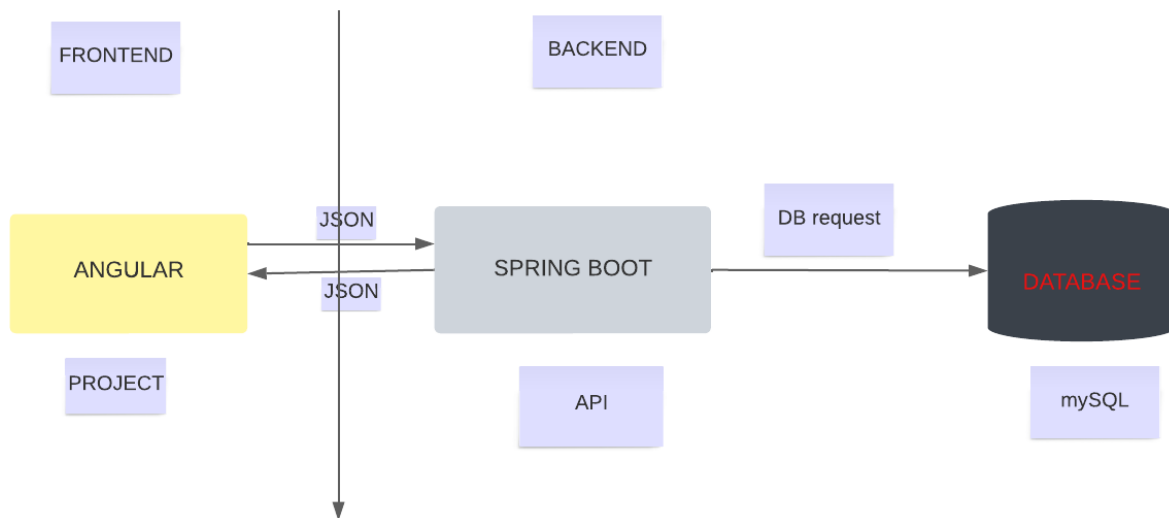
subjects or topics to facilitate easy organization and retrieval. Question data is stored in the MySQL database.

- **Exam Enrollment:** Students can enroll in available exams based on their course or subject of interest. The system tracks enrolled students and manages exam schedules. Enrollment data and exam schedules are stored in the MySQL database.
- **Exam Taking:** Students can attempt exams online within the specified time limits. The system presents questions based on the exam structure and allows students to submit their answers. Various question types are supported, such as multiple choice, descriptive, and more. Student answers and exam progress are stored in the MySQL database.
- **Results and Analysis:** Once an exam is completed, the Exam Portal automatically evaluates the student's answers and generates results. Students can view their scores, correct and incorrect answers, and receive feedback. Exam result data is stored in the MySQL database. Administrators can access detailed exam reports and analyze performance trends using data retrieved from the database.
- **Notifications and Reminders:** The system sends notifications and reminders to students about upcoming exams, enrollment deadlines, and other important updates. Notification details and student preferences are stored in the MySQL database.
- **User Management:** Administrators have the ability to manage user accounts, including adding new users, modifying user details, and assigning roles and permissions. User account information is stored in the MySQL database.
- **Responsive and Intuitive UI:** The frontend interface developed using AngularJS provides a responsive and user-friendly experience, enabling seamless navigation and interaction. Data retrieved from the MySQL database is presented to users through the UI.
- **Security and Data Persistence:** The Exam Portal ensures the security of user data and system integrity. Data stored in the MySQL database is encrypted and protected. The MySQL database provides data persistence and reliability.

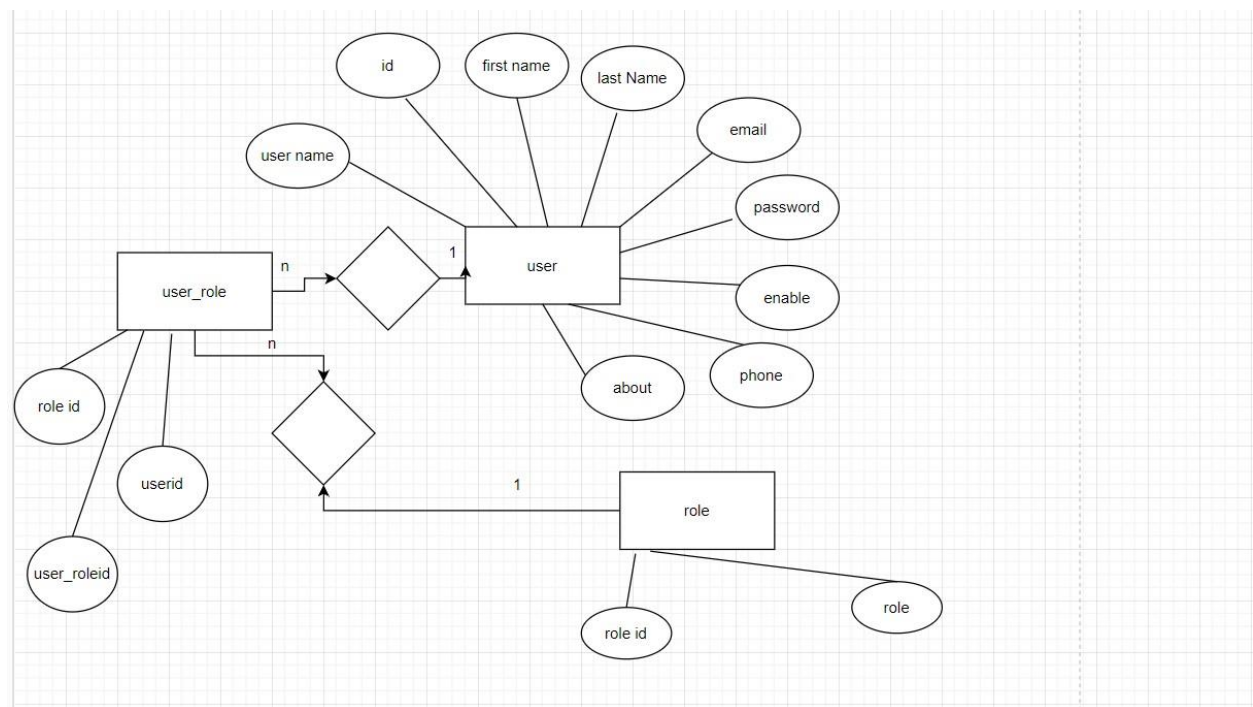
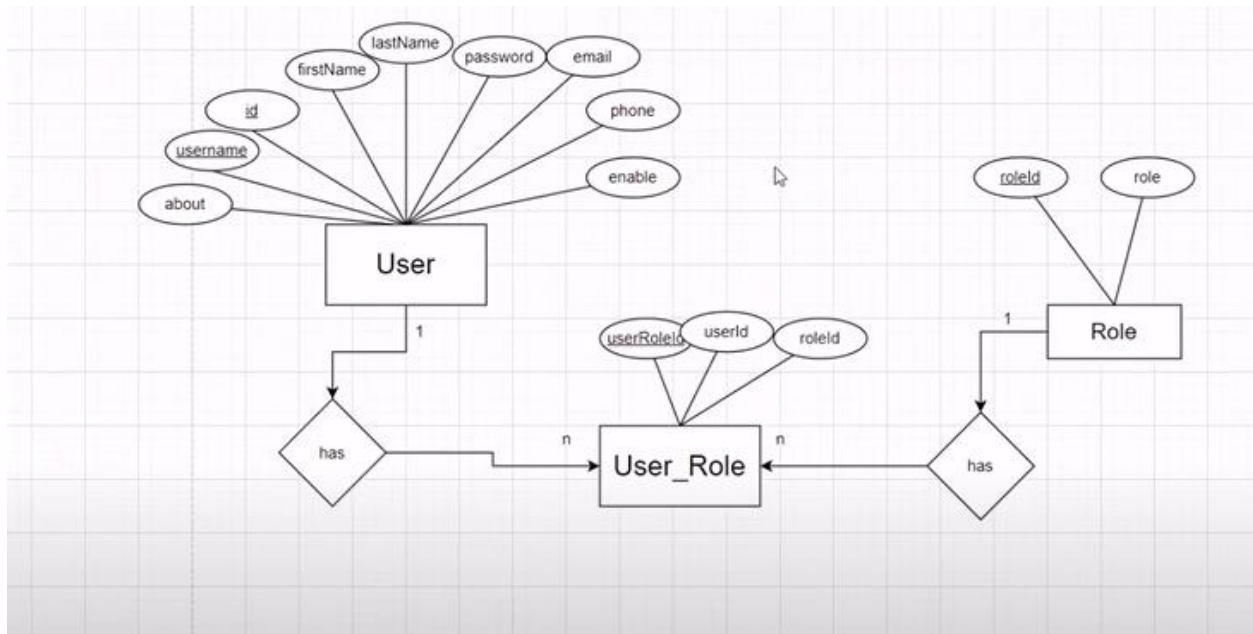
The Exam Portal combines the power of Spring Boot's robust backend capabilities, AngularJS's dynamic frontend, and the reliability of MySQL as the database management system. It offers an intuitive user interface, advanced features for exam

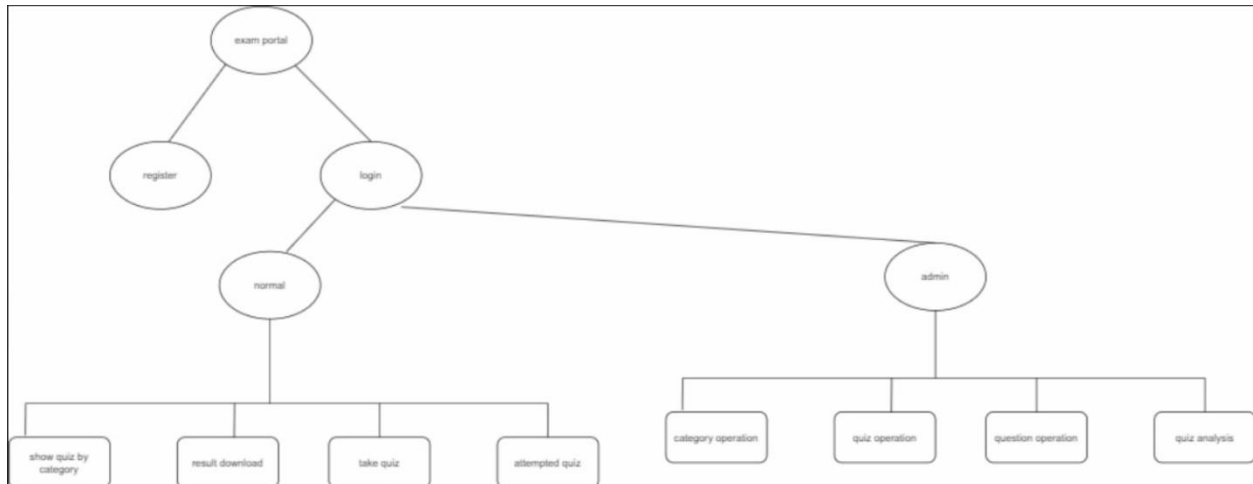
creation and management, and detailed result analysis. This project aims to streamline the exam administration process, improve accessibility for students, and provide valuable insights to administrators for effective evaluation and decision-making.

### **Project Architecture:**



### **Project Structure:**





### **Pre-requisites :**

Here are the prerequisites for a Java Spring Boot project with MySQL and MongoDB, along with relevant links for download and installation:

**1-Java Development Kit (JDK):**JDK is required to compile and run Java applications, providing the necessary tools and libraries. Download and install the latest JDK version from Oracle's website.

- Download JDK: <https://www.oracle.com/java/technologies/javase-jdk11-downloads.html>

**2-Integrated Development Environment (IDE):** An IDE offers a comprehensive development environment for writing, debugging, and managing code. IntelliJ IDEA, Eclipse, or Visual Studio Code are popular choices for Java development.

- IntelliJ IDEA: <https://www.jetbrains.com/idea/download/>

- Eclipse: <https://www.eclipse.org/downloads/>

- Visual Studio Code: <https://code.visualstudio.com/download>

**3-Spring Boot:** Spring Boot simplifies Java application development by providing predefined configurations, automatic dependency management, and a streamlined

development experience. Use Spring Initializr or Spring Tools for your IDE to create a Spring Boot project.

- Spring Initializr (Online): <https://start.spring.io/>
- Spring Tools 4 for Eclipse: <https://spring.io/tools>
- Spring Tools for Visual Studio Code: Install via Extensions in Visual Studio Code

**4-MySQL Database:** MySQL is a popular relational database management system. Install MySQL Community Server and optionally MySQL Workbench, a graphical tool for managing MySQL databases.

- MySQL Community Server: <https://dev.mysql.com/downloads/installer/>
- MySQL Workbench: <https://dev.mysql.com/downloads/workbench/>

**5-MySQL Connector/J:** MySQL Connector/J is the official JDBC driver for connecting Java applications to MySQL databases. Include this dependency in your project to enable connectivity and interaction with MySQL.

- Maven:
- Add the following dependency to your project's pom.xml:

xml

```
<dependency>
```

```
    <groupId>mysql</groupId>
```

```
    <artifactId>mysql-connector-java</artifactId>
```

```
    <version>8.0.27</version>
```

```
</dependency>
```

- Maven Repository: <https://mvnrepository.com/artifact/mysql/mysql-connector-java>

- **Gradle:**

- Add the following dependency to your project's build.gradle:

implementation 'mysql:mysql-connector-java:8.0.27'

- Gradle Repository: <https://search.maven.org/artifact/mysql/mysql-connector-java/8.0.27/jar>

## **Role Based Access:**

### **1. Admin Role:**

- **Responsibilities:** The admin role has full control and administrative privileges over the Exam Portal system.
- **Permissions:**
  - **Manage Exams:** Admins can create, edit, and delete exams, including setting exam structures, time limits, and question types.
  - **Manage Questions:** Admins can add, edit, and delete questions in the question bank, categorize them, and manage question types.
  - **Manage Users:** Admins can create, edit, and delete user accounts, as well as manage their roles and permissions within the Exam Portal system.
  - **Generate Reports:** Admins have access to generate reports and analytics on exam performance, student progress, and other system statistics.

### **2. Instructor role:**

- **Responsibilities:** Instructors are responsible for managing and conducting exams within the Exam Portal.
- **Permissions:**
  - **Create Exams:** Instructors can create exams, set up question structures, time limits, and other exam-related configurations.
  - **Manage Enrollments:** Instructors can manage student enrollments for their specific exams, including adding and removing students.
  - **Grade Exams:** Instructors can evaluate student answers, provide feedback, and assign grades for exams they are responsible for.
  - **View Reports:** Instructors can access reports and analytics on student performance, exam statistics, and other relevant data.

### **3. Student Role:**

- **Responsibilities:** Students are the users who participate in exams and assessments within the Exam Portal.

- **Permissions:**

- **Enroll in Exams:** Students can view available exams and enroll themselves in exams based on their courses or subjects of interest.
- **Take Exams:** Students can access and attempt exams within the specified time limits, providing answers and submitting their responses.
- **View Results:** Students can view their exam results, including scores, correct and incorrect answers, and feedback provided by instructors.
- **View Personal Profile:** Students can view and update their profile information, such as contact details or academic information.

These roles allow for proper segregation of responsibilities and permissions within the Exam Portal system, ensuring that administrators, instructors, and students have appropriate access and control based on their roles and responsibilities.

### **Project Flow:**

- Frontend Development
- Backend Development
- Integration
- Containerization of the Application
- Deployment to Kubernetes Cluster

- **Frontend Development:**

- 1. UI/UX Design:**

- Create wireframes and mockups to visualize the user interface (UI).
- Design an intuitive and user-friendly UI/UX for seamless navigation.
- Ensure responsiveness and compatibility across different devices and screen sizes.
- Incorporate branding elements and maintain a consistent visual design.

- 2. User Registration and Authentication:**

- Develop a user registration form to capture user details.
- Implement authentication mechanisms for secure user login and session management.
- Provide password reset and account recovery options.

- 3. Dashboard and Navigation:**

- Design a dashboard that provides an overview of relevant information and actions.



- Implement a navigation menu for easy access to different sections of the portal.
- Incorporate breadcrumbs or a hierarchical menu for navigating through the portal.

#### **4. Exam Creation and Management:**

- Develop a form for instructors to create exams, including setting exam details, duration, and question types.
- Enable instructors to manage exam questions, add/edit/delete questions, and assign marks.
- Implement validation checks to ensure proper formatting and accuracy of entered data.

#### **5. Exam Taking:**

- Design an intuitive interface for students to view and select available exams.
- Create exam pages to display questions and allow students to provide answers.
- Implement timers and progress indicators to track the remaining time and completion status.
- Validate and store student answers during the exam.

#### **6. Result Display:**

- Design a visually appealing layout to display exam results.
- Develop functionality to calculate and display student scores and grades.
- Provide a detailed breakdown of marks obtained in each section or question.
- Enable students to view their past exam results and performance history.

#### **7. User Profile and Settings:**

- Implement a user profile page for users to view and update their personal information.
- Allow users to customize their preferences, such as notification settings and language preferences.
- Provide options to change passwords and update account settings.

#### **8. Error Handling and Feedback:**

- Implement error handling mechanisms to display meaningful error messages to users.
- Provide feedback messages and notifications to inform users of successful actions or updates.
- Perform form validations and display appropriate error messages for invalid inputs.

#### **9. Integration with Backend:**

- Integrate the frontend with the backend APIs using HTTP requests.
- Handle API responses and update the UI accordingly.
- Implement proper error handling for failed API requests.

#### **10. Testing and Optimization:**

- Conduct comprehensive testing of the frontend components and functionality.
- Ensure cross-browser compatibility and responsiveness on different devices.
- Optimize the frontend code and assets for faster loading and improved performance.

### **• Backend development**

#### **1. Database Design:**

- Design the database schema using MySQL to store exam-related data.
- Define tables for users, exams, questions, answers, and results.
- Establish appropriate relationships between tables (e.g., one-to-many, many-to-many).

#### **2. API Development:**

- Implement RESTful APIs using Spring Boot to handle client requests and responses.
- Create endpoints for user authentication, registration, and profile management.
- Develop APIs for exam creation, management, and retrieval of exam details.
- Implement endpoints for submitting exam answers and retrieving results.

#### **3. User Authentication and Authorization:**

- Develop authentication mechanisms, such as JWT (JSON Web Tokens) or session-based authentication.
- Implement user registration and login functionality.
- Set up authorization rules to control access based on user roles (e.g., administrators, instructors, students).

#### **4. Exam Management:**

- Create API endpoints for creating, updating, and deleting exams.
- Implement logic to handle exam scheduling, duration, and access permissions.
- Develop endpoints for adding, editing, and deleting exam questions.
- Incorporate validation checks to ensure data integrity.

#### **5. Result Calculation and Management:**

- Develop algorithms for calculating exam results based on student answers.
- Create API endpoints to store and retrieve student results.
- Implement logic for grade calculation and score aggregation.
- Provide APIs for generating result reports or exporting data.

#### **6. Data Validation and Security:**

- Implement validation checks for incoming data to ensure data integrity.
- Apply input sanitization and validation techniques to prevent security vulnerabilities.
- Implement appropriate security measures, such as encryption and hashing for sensitive data.

#### **7. Integration with Frontend:**

- Expose APIs and endpoints required by the frontend application.
- Handle requests from the frontend and process them in the backend.
- Validate and sanitize user inputs received from the frontend.
- Send appropriate responses to the frontend based on the requested data or actions.

#### **8. Error Handling and Logging:**

- Implement error handling mechanisms to provide meaningful error messages.
- Log errors and exceptions for debugging and troubleshooting purposes.
- Handle exceptions gracefully to prevent system crashes or data corruption.

#### **9. Testing and Quality Assurance:**

- Perform unit testing for individual backend components and functions.
- Conduct integration testing to ensure proper communication between frontend and backend.
- Validate the accuracy of result calculation algorithms.
- Perform security testing and vulnerability assessments.

## **10. Deployment and Performance Optimization:**

- Set up a production environment to host the backend application.
- Deploy the Spring Boot application to a server or cloud platform.
- Monitor performance metrics and optimize database queries for efficiency.
- Implement caching mechanisms to improve response times.

### **• Integration**

#### **1. API Definition:**

- Define the API endpoints in the backend that the frontend needs to communicate with.
- Document the API specifications, including request formats, response formats, and authentication requirements.

#### **2. API Integration:**

- In the frontend codebase, make HTTP requests to the backend APIs using libraries like Axios or Angular's HttpClient.
- Set up appropriate headers for authentication, such as including tokens or session information in the requests.
- Handle the responses received from the backend APIs, including processing the data or handling errors.

#### **3. Data Exchange:**

- Determine the required data to be exchanged between the frontend and backend.
- Send data from the frontend to the backend through API requests, including user inputs, exam details, and student answers.
- Receive data from the backend in response to API requests, such as exam results, user information, or exam details.

#### **4. Error Handling:**

- Implement error handling mechanisms in the frontend to handle API call failures or errors returned by the backend.
- Display appropriate error messages or notifications to the users in case of API failures.
- Handle potential scenarios where the frontend and backend have different data validation rules or error codes.

## **5. Testing and Debugging:**

- Test the integration between the frontend and backend to ensure the proper flow of data and functionality.
- Debug any issues or errors that arise during the integration process, such as incorrect API calls or mismatched data formats.
- Use debugging tools and logs to identify and resolve integration-related issues.

## **6. Security Considerations:**

- Ensure that the data exchanged between the frontend and backend is securely transmitted over HTTPS.
- Implement proper authentication and authorization mechanisms to prevent unauthorized access to backend resources.
- Validate and sanitize user inputs on both the frontend and backend to prevent security vulnerabilities.

## **7. Continuous Integration and Deployment:**

- Automate the integration process as part of the continuous integration (CI) pipeline.
- Ensure that the frontend and backend components are deployed and updated together to maintain compatibility and consistency.
- Test the integrated system thoroughly after each deployment to verify the functionality and data exchange.

## **● Containerization of the Application**

### **1. Containerization Platform Selection:**

- Choose a containerization platform such as Docker or Kubernetes based on your project requirements and infrastructure.

### **2. Containerization Setup:**

- Set up the containerization platform and ensure it is properly installed and configured on the development and deployment environments.

### **3. Dockerization:**

- Create a Dockerfile that defines the necessary steps to build a Docker image of your application.
- Specify the base image, dependencies, and instructions to install and configure your backend application.
- Include any additional setup steps such as copying configuration files and running database migrations.
- Build the Docker image using the Dockerfile, which packages your application along with its dependencies into a container.

#### **4. Containerization of Frontend:**

- Create a separate Dockerfile for the frontend application.
- Define the base image, dependencies, and instructions to build and package the frontend code into a container.
- Specify the command to run the frontend server or application within the container.

#### **5. Container Orchestration:**

- If required, set up a container orchestration platform like Kubernetes to manage and deploy multiple containers.
- Define the container configurations, such as resource limits and scaling rules, in the deployment manifests.

#### **6. Container Deployment:**

- Push the built Docker images to a container registry or repository, such as Docker Hub or a private registry.
- Pull the Docker images onto the deployment environment or use a container orchestration platform to automatically deploy the containers.
- Configure the networking and environment variables required for the containers to run correctly.

#### **7. Monitoring and Scaling:**

- Set up monitoring tools to monitor the health and performance of the containers.
- Configure automatic scaling rules to scale up or down based on resource utilization or incoming traffic.

#### **8. Continuous Integration and Deployment:**

- Integrate containerization into your CI/CD pipeline, ensuring that the Docker images are built and pushed to the registry automatically.
- Automate the deployment of containers to staging or production environments as part of the CI/CD process.

Containerization provides benefits such as easy deployment, scalability, and portability of the application across different environments. It helps in isolating dependencies, streamlining deployment processes, and ensuring consistent application behavior.

- **Deployment to Kubernetes Cluster**

- 1. Set up the Kubernetes Cluster:**

- Set up a Kubernetes cluster on your preferred infrastructure, such as a cloud provider or on-premises environment.
- Configure the cluster nodes, networking, and storage resources.

- 2. Containerize the Application:**

- Containerize your backend and frontend applications using Docker, as described earlier.
- Build Docker images of your applications and push them to a container registry accessible by your Kubernetes cluster.

- 3. Define Kubernetes Deployment Manifests:**

- Create Kubernetes deployment manifests (YAML files) that define the desired state of your application deployments.
- Specify the containers to be deployed, their images, resource requirements, and any necessary environment variables or configuration.

- 4. Define Kubernetes Service Manifests:**

- Create Kubernetes service manifests to define the services that expose your application to external clients.
- Specify the service type (e.g., LoadBalancer, NodePort), ports, and target containers.

- 5. Deploy the Application:**

- Use the Kubernetes command-line tool (kubectl) or a deployment management tool to apply the deployment and service manifests to the cluster.
- The Kubernetes scheduler will create the specified number of container instances, distribute them across the cluster nodes, and manage their lifecycle.

- 6. Monitor and Scale:**

- Utilize Kubernetes monitoring tools or integrated metrics to monitor the health and performance of your application deployments.

- Configure scaling rules or use Kubernetes' built-in scaling features to scale the application pods based on resource utilization or incoming traffic.

## **7. Manage Updates and Rollbacks:**

- Use Kubernetes deployment features such as rolling updates or canary deployments to manage updates to your application without downtime.
- Perform version updates or configuration changes by updating the deployment manifests and applying them to the cluster.
- If needed, roll back to a previous version by reverting the deployment manifests.

w

## **8. Secure the Deployment:**

- Implement appropriate security measures such as network policies, access controls, and secrets management to secure your Kubernetes cluster.
- Regularly update Kubernetes and container images to leverage security patches and updates.

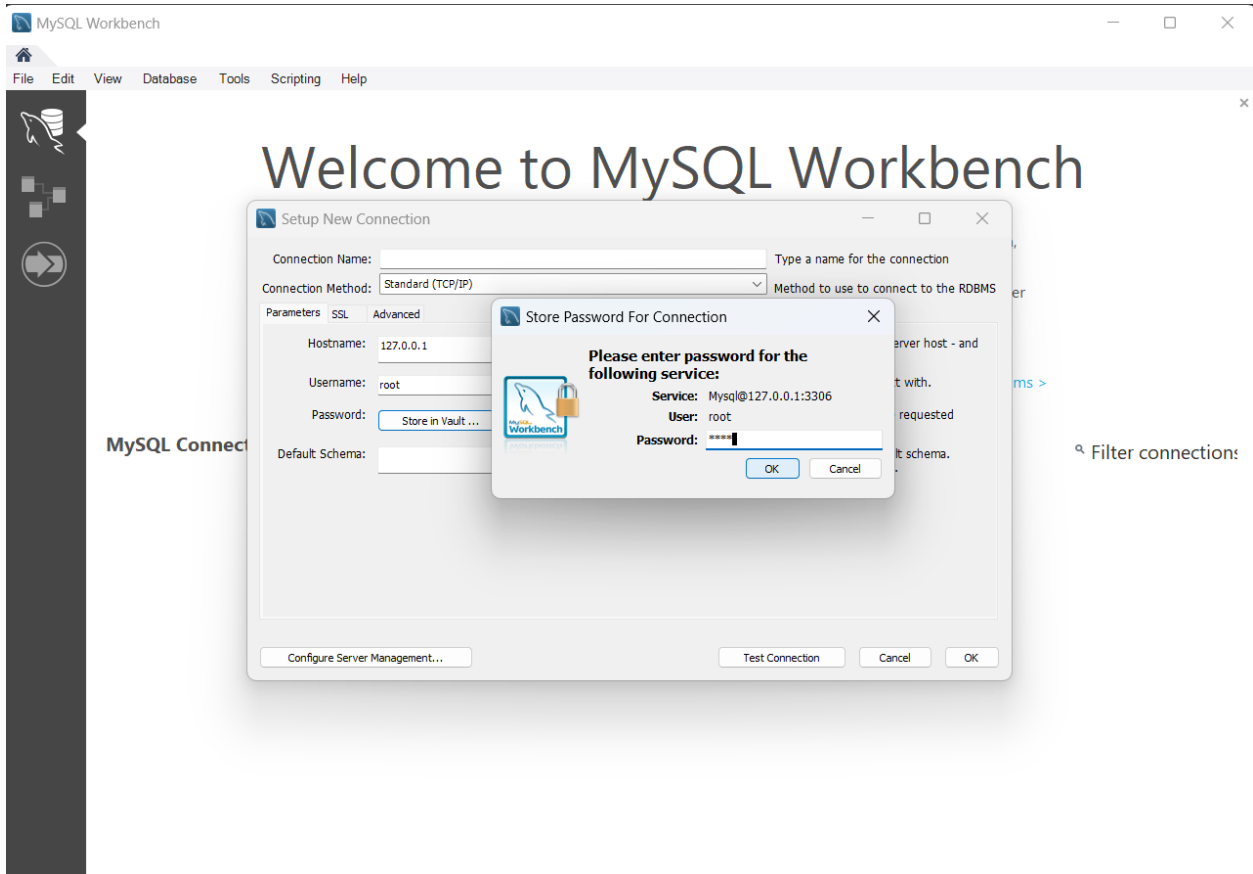
## **9. Continuous Deployment and Automation:**

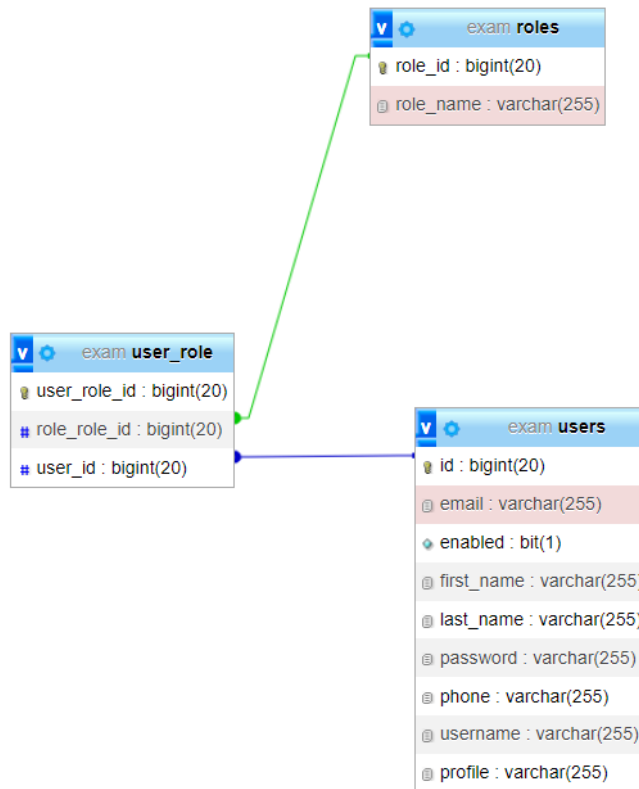
- Integrate your deployment process with your CI/CD pipeline to automate the deployment of new versions or changes to the Kubernetes cluster.
- Use deployment automation tools or scripts to streamline the process and ensure consistency.

Deploying your application to a Kubernetes cluster provides benefits such as scalability, resilience, and simplified management of your application deployments. Kubernetes handles the distribution, scaling, and monitoring of your containers, allowing for efficient resource utilization and easy management of your application in a production environment.

## **MySQL setup**








Examportal

localhost:4200/login

Pariskha

Home Login Register



Login Here !!

Username \*

Password \*

Login Reset

er/quiz/ongoing/10

Logout durgesh3

## Java Basics Online Examination

Programming Category

Number of questions **6** Max Marks : **100**

Quiz will automatically subr when timer reaches to 0 : 0

**11 min : 48 se**

**What is Class?**

☐ Logical Entity
 ☐ Physical Entity
 ☐ Nothing
 ☐ All

**What is object?**

☐ Real World Entity
 ☐ Logical Entity
 ☐ All
 ☐ None

**What is Package?**

☐ Logical Container
 ☐ Thread
 ☐ Process
 ☐ All

**In java control statements break, continue, return, try-catch-finally and assert belongs to?**

☐ Selection statements
 ☐ Loop Statements
 ☐ Transfer statements
 ☐ Pause Statement

Examportal

localhost:4200/user/attempted

Pariskha Logout durgesh3

Action

**Quiz Attempts**

Categories

☐ All
 ☐ PROGRAMMING
 ☐ GENERAL KNOWLEDGE

### You have attempted the following quiz

#### Java Basics

Attempted Date 4/20/2021, 4:21:35 PM

When we consider a Java program, it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what do class, object, methods, and instance variables mean.

**Marks Obtained : 50**
**Max. Marks : 100**

**Questions Attempted: 5**
**Correct Answers : 3**

#### Python Programming

Attempted Date 4/20/2021, 11:22:01 AM

this is a python programming quiz. this contains questions related to python.

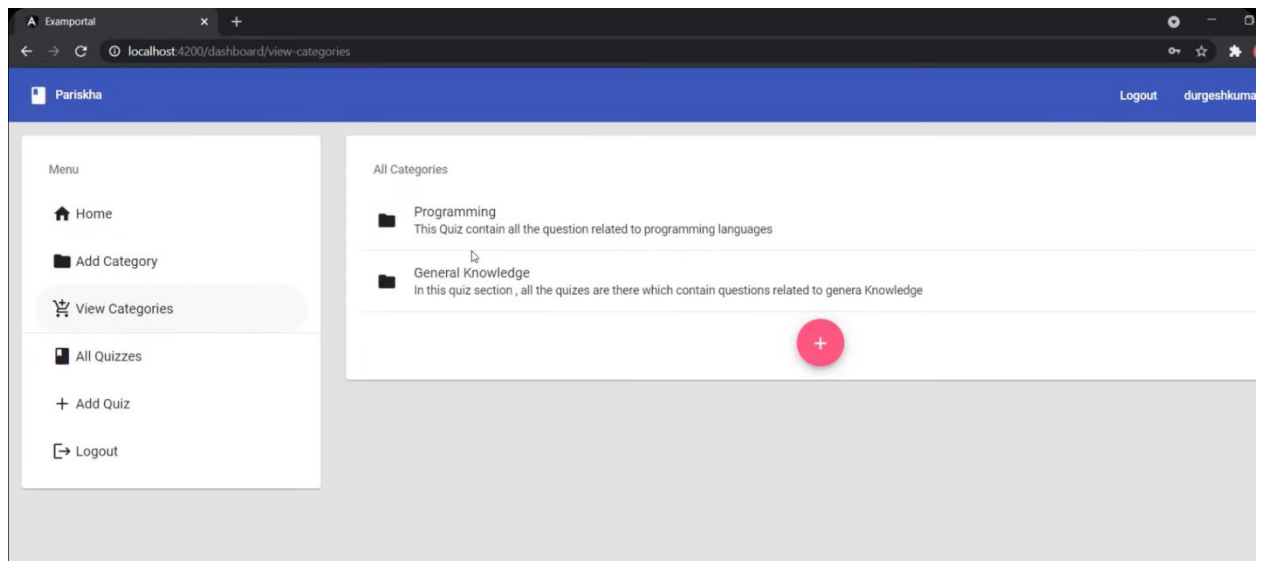
**Marks Obtained : 0**
**Max. Marks : 200**

**Questions Attempted: 2**
**Correct Answers : 0**

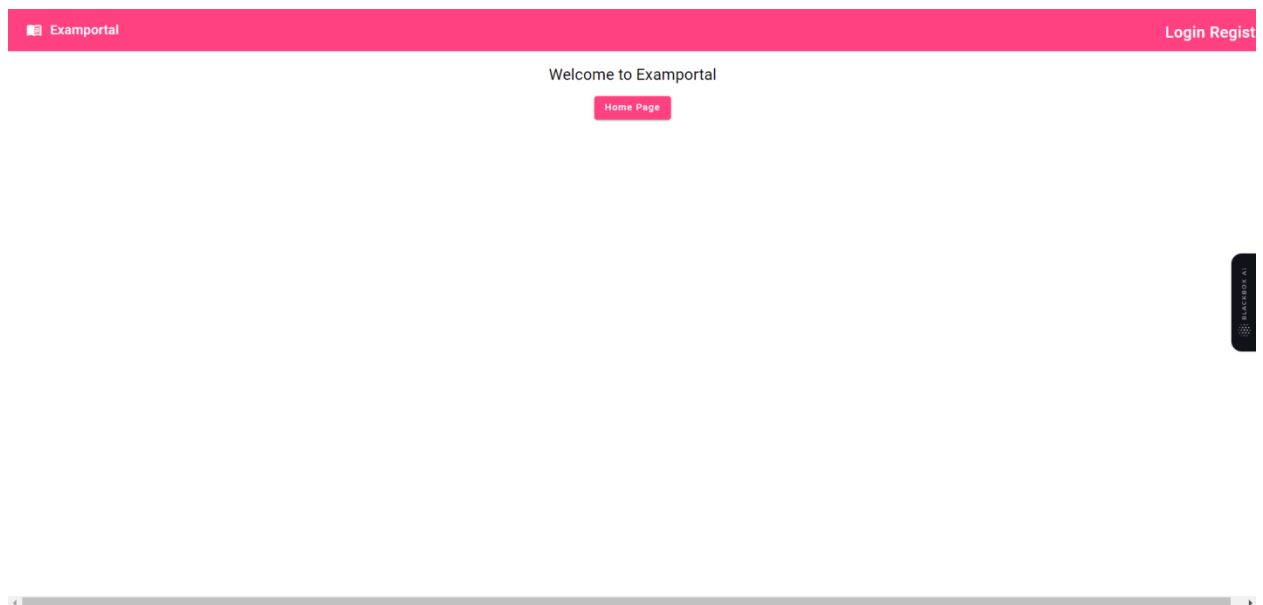
#### Java Basics

Attempted Date 4/1/2021, 9:15:58 PM

When we consider a Java prooram. it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what do




## Landing Page:



## New User Registration Page

localhost:4200/signup

Examportal Login Register



NATURE LOGO

Register Here !!

User Name\*

Username must be unique !!

User Password\*

First Name\*

Last Name\*


Email Address\*

Phone Number\*

Register Clear


localhost:4200/signup

Examportal Login Register



NATURE LOGO

Register Here !!



Successfully done !!

User id is 21

OK

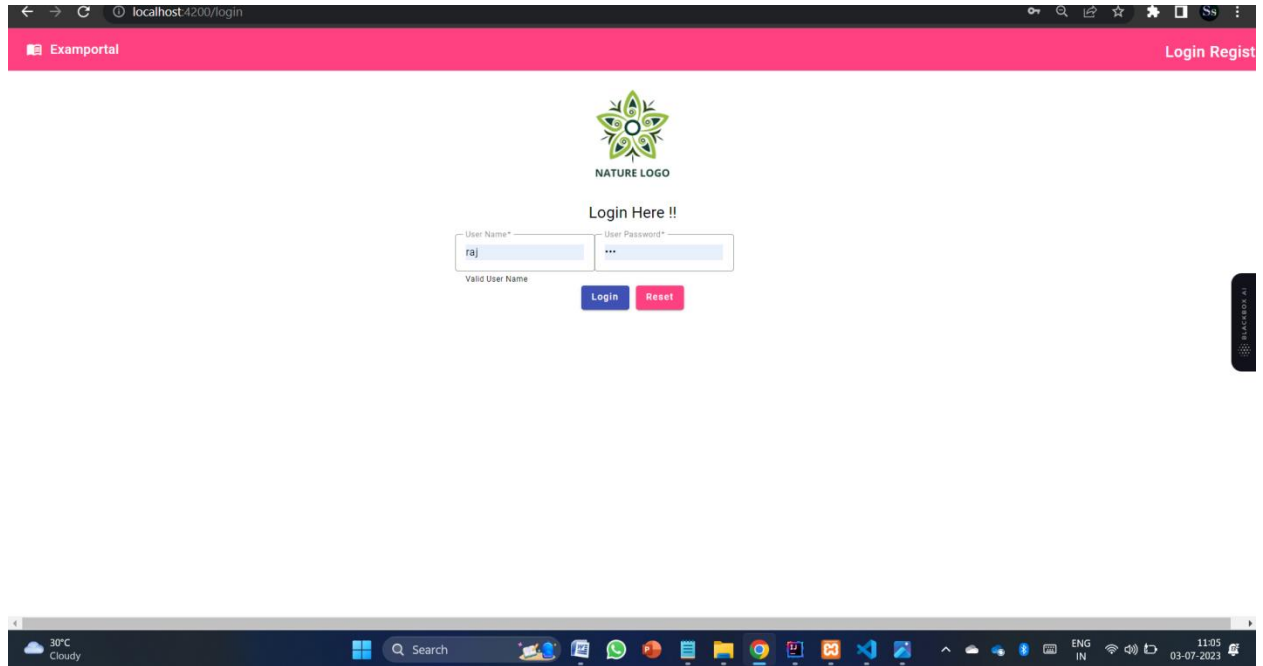
duqub@mailinator.com

Phone Number\*

812

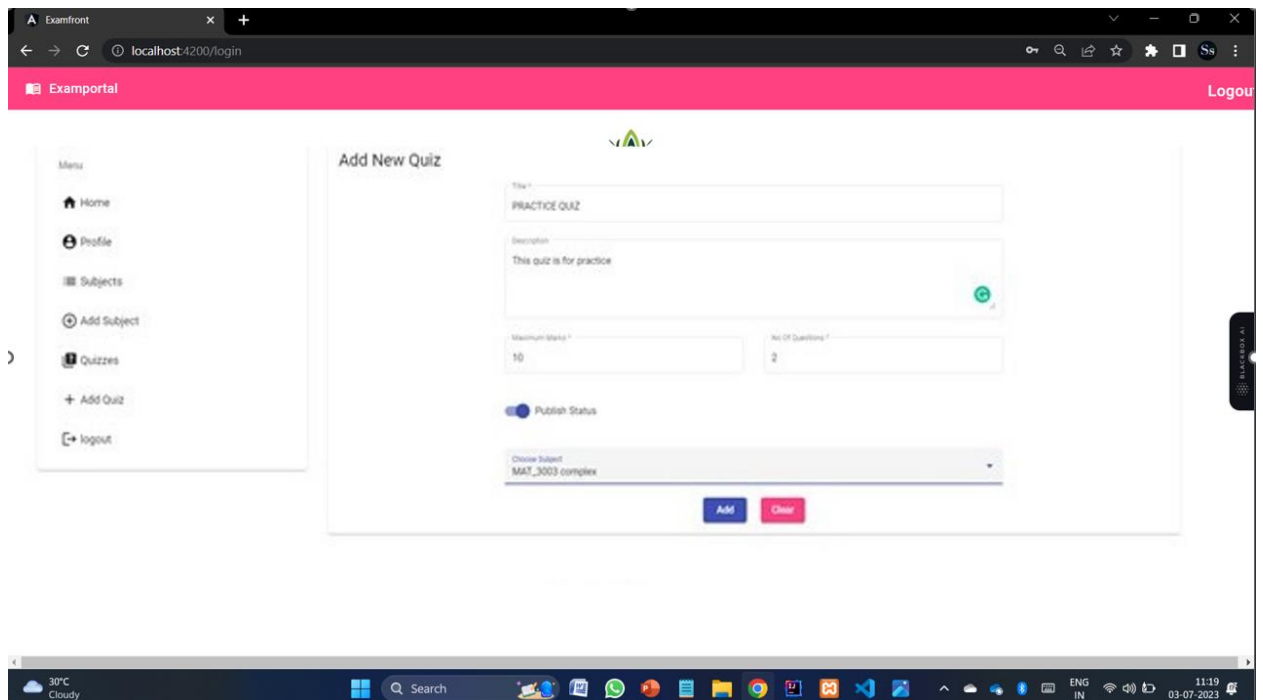
Register Clear

## Login Page

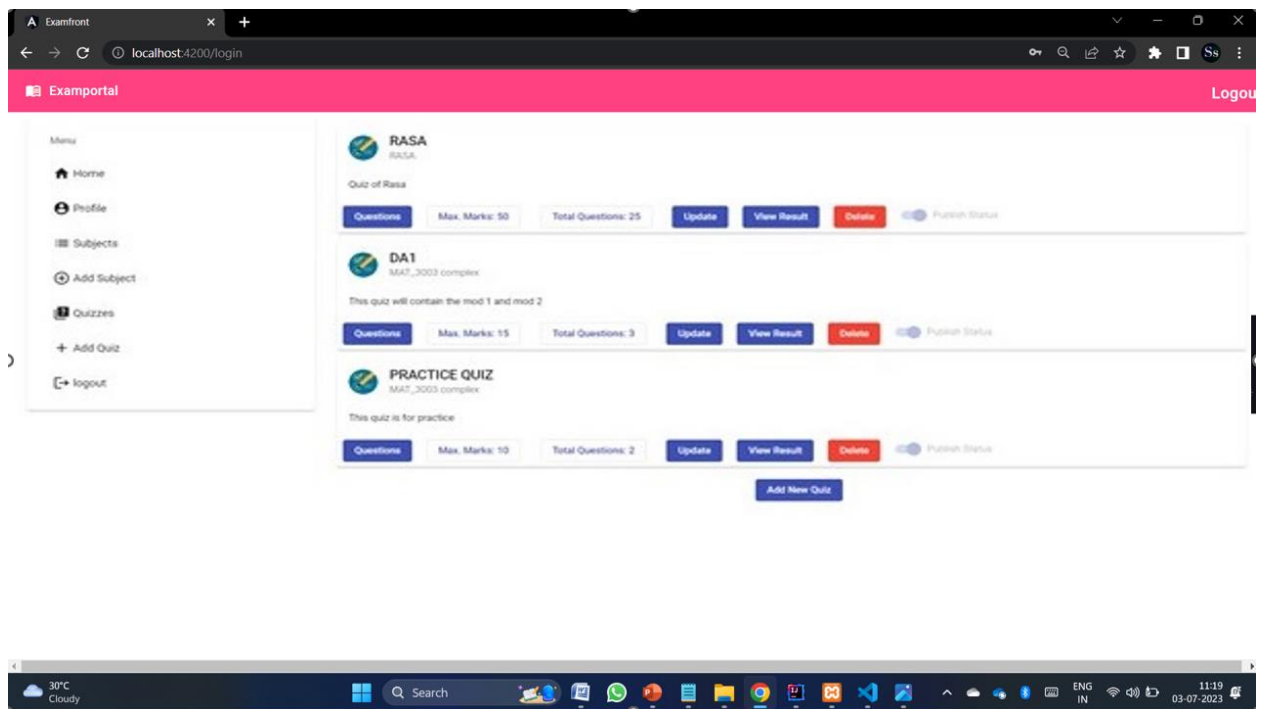
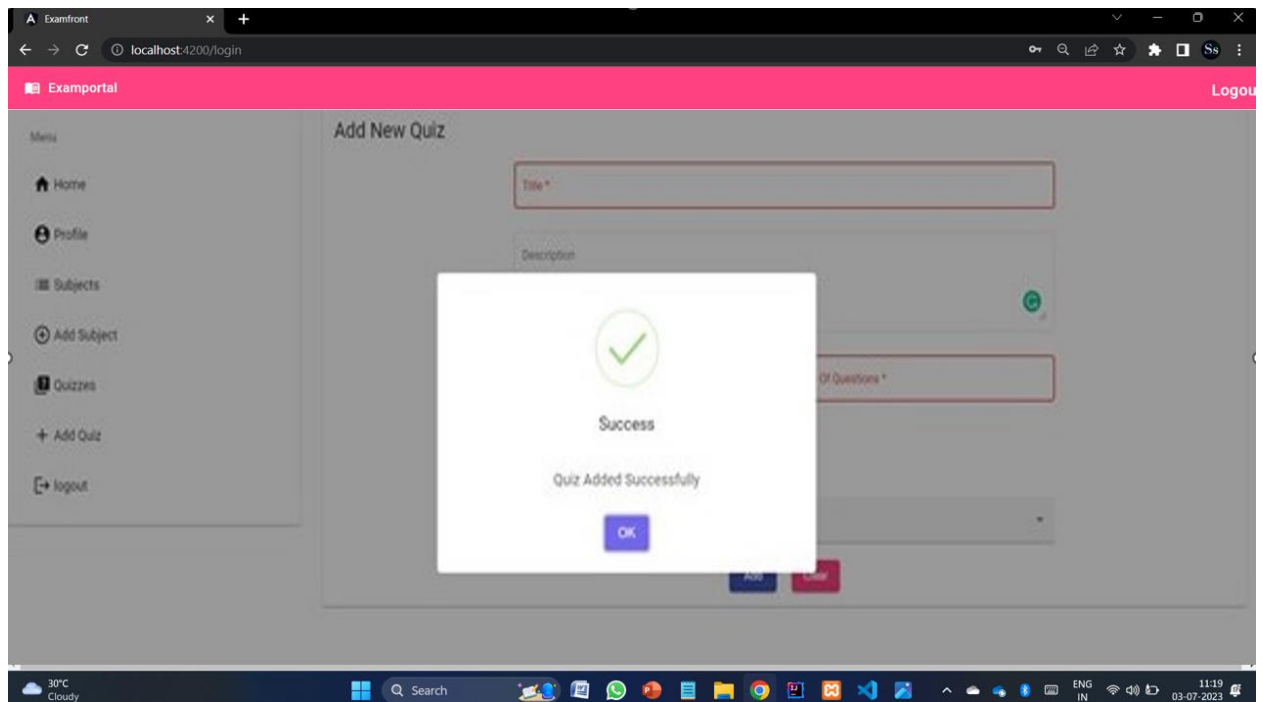


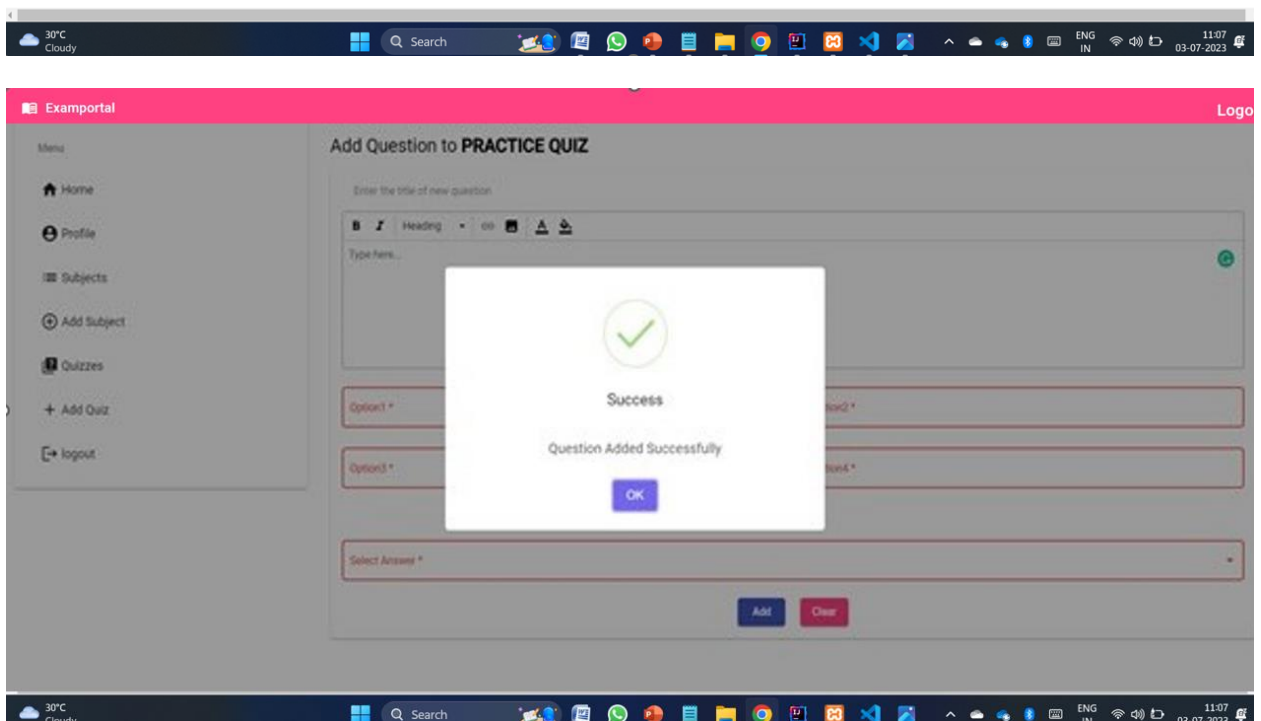
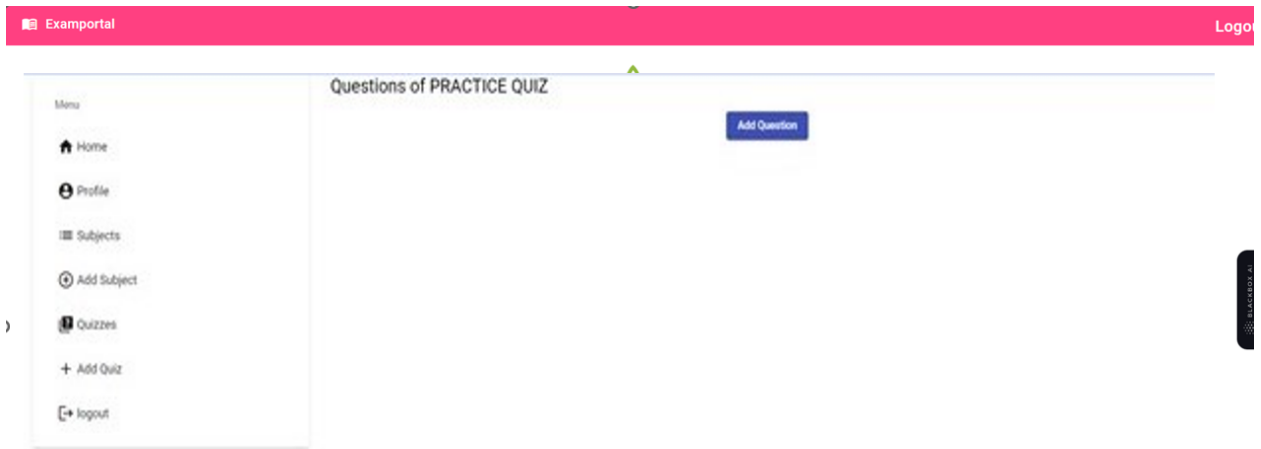
The screenshot shows a web browser window with the URL `localhost:4200/login`. The page has a pink header bar with the text "Examportal" on the left and "Login Regist" on the right. Below the header, there is a green logo with the text "NATURE LOGO" underneath it. The main heading is "Login Here !!". There are two input fields: "User Name\*" with the value "fbj" and "User Password\*" with three dots. Below the "User Name\*" field, it says "Valid User Name". There are two buttons: "Login" (blue) and "Reset" (pink). On the right side of the page, there is a vertical black bar with the text "SHACEDBY AI". The Windows taskbar at the bottom shows the date and time as 11:05 on 03-07-2023, and the weather as 30°C Cloudy.

## Admin Page



The screenshot shows a web browser window with the URL `localhost:4200/login`. The page has a pink header bar with the text "Examportal" on the left and "Logout" on the right. Below the header, there is a green logo with the text "NATURE LOGO" underneath it. The main heading is "Add New Quiz". There are two input fields: "Title\*" with the value "PRACTICE QUIZ" and "Description" with the value "This quiz is for practice". There are two input fields: "Maximum Mark\*" with the value "10" and "No Of Questions\*" with the value "2". There is a "Publish Status" section with a blue radio button selected. There is a "Choose Subject" dropdown menu with the value "MAT\_3003 complex". There are two buttons: "Add" (blue) and "Clear" (pink). On the left side of the page, there is a vertical black bar with the text "SHACEDBY AI". The Windows taskbar at the bottom shows the date and time as 11:19 on 03-07-2023, and the weather as 30°C Cloudy.







## Databases

```
mysql> use examportal;
Database changed
mysql> show tables;
+-----+
| Tables_in_examportal |
+-----+
| category               |
| hibernate_sequence     |
| question               |
| quiz                   |
| result                 |
| role                   |
| user                   |
| user_role              |
+-----+
8 rows in set (0.06 sec)
```

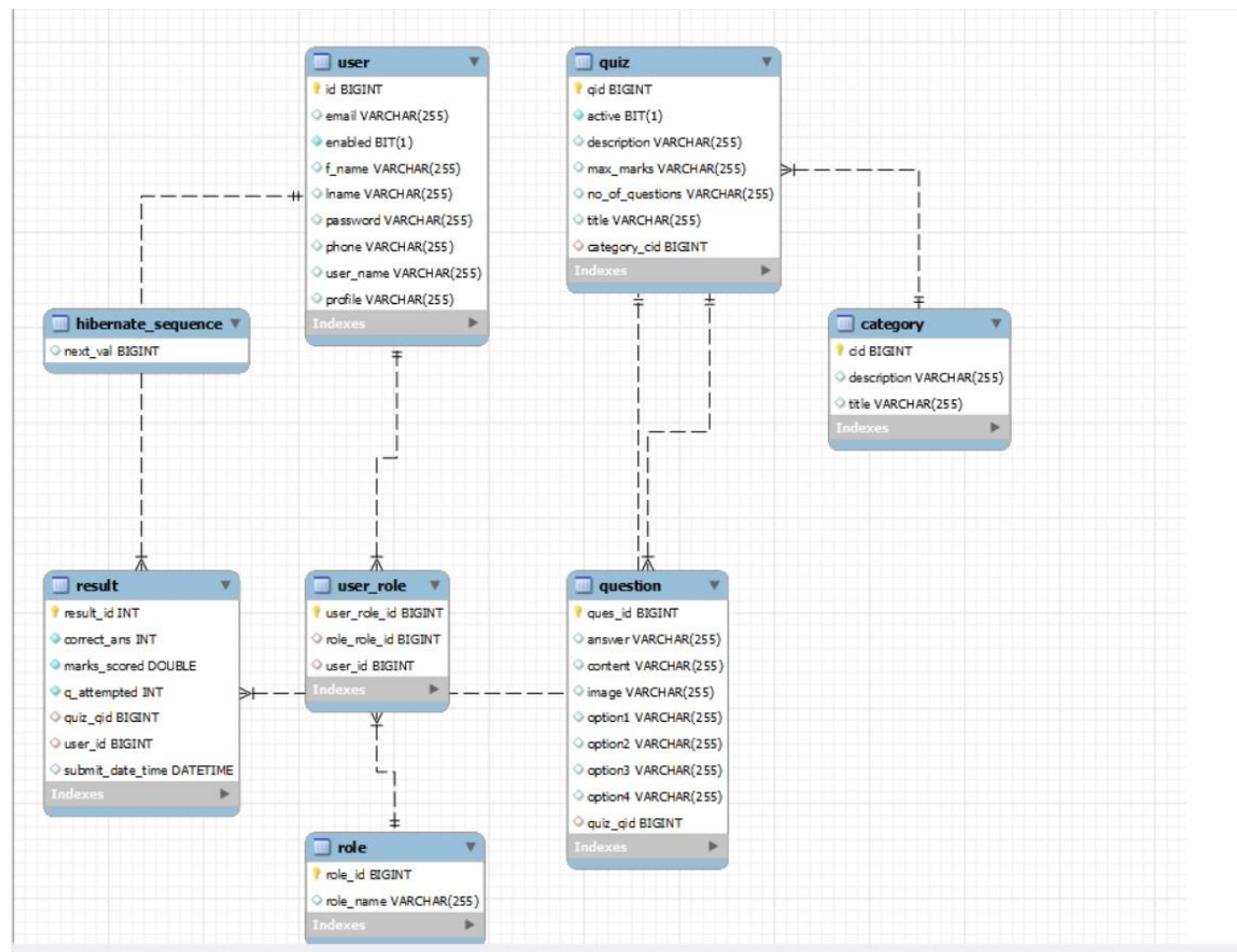
```
mysql> desc user;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id         | bigint        | NO   | PRI | NULL    |       |
| email      | varchar(255)  | YES  |     | NULL    |       |
| enabled    | bit(1)        | NO   |     | NULL    |       |
| f_name     | varchar(255)  | YES  |     | NULL    |       |
| lname     | varchar(255)  | YES  |     | NULL    |       |
| password   | varchar(255)  | YES  |     | NULL    |       |
| phone      | varchar(255)  | YES  |     | NULL    |       |
| user_name  | varchar(255)  | YES  |     | NULL    |       |
| profile    | varchar(255)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.03 sec)
```

```
mysql> desc role;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| role_id    | bigint        | NO   | PRI | NULL    |       |
| role_name  | varchar(255)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql> desc quiz;
```

Field	Type	Null	Key	Default	Extra
qid	bigint	NO	PRI	NULL	
active	bit(1)	NO		NULL	
description	varchar(255)	YES		NULL	
max_marks	varchar(255)	YES		NULL	
no_of_questions	varchar(255)	YES		NULL	
title	varchar(255)	YES		NULL	
category_cid	bigint	YES	MUL	NULL	

```
7 rows in set (0.01 sec)
```



## Instructions

- 1) Don't try to Switch tab
- 2) If you leave the site your quiz will automatically submitted
- 3) Don't Refresh
- 4) The quiz will automatically submitted once the timer reach to 00:00
- 5) Scroll down to see the questions

On Going Quiz **PRACTICE QUIZ**Q1)  $\sin \theta$  is where  $\sin \theta$  is

- ☐ east ☐ north  
☐ west ☐ south

Q2)  $\sin \theta$  is where  $\sin \theta$  is

- ☐ 4 ☐ 6  
☐ 5 ☐ 7

[Submit Quiz](#)

## Progress

Quiz Will Automatically submitted once time get over

Time Remaining:

3 min : 57 sec



## Quiz Result

Name :: Vaibhav Agarwal

Quiz on PRACTICE QUIZ

Completed in 0 min : 14 sec

Marks Got:: 0

Correct Answer::0

Questions Attempted::0

[Go To Home](#)