A

PROJECT REPORT ON

"EXAMINATION SYSTM"

Submitted to

UNITED SCHOOL OF BUSINESS MANAGEMENT

In Partial Fulfillment of Post Graduate Degree in **Master in Computer Application (MCA)**

Submitted By

SASWAT RANJAN MOHANTY

Regd. No- **2105274095**

Under

BIJU PATNAIK UNIVERSITY OF TECHNOLOGY, ODISHA

Annexure-I

CERTIFICATION OF COMPLETION

This is to certify that, Mr. Saswat Ranjan Mohanty bearing Regd. No 2105274095 is a bona-fide student studying in 4th Semester Master in Computer Application (MCA) program of Biju Patnaik University of Technology, Odisha for the year 2021-23.

As per the university curriculum, he has successfully completed the major project title as "Examination System" under the guidance of Prof. Subhasis Mohapatra.

GUIDED BY:

Prof. Subhasis Mohaptra
Assistant Professor

Program Coordinator

External Examiner

CERTIFICATION OF COMPLETION

This is to certify that, Mr. Saswat Ranjan Mohanty bearing Regd. No 2105274095 is a bona-fide student studying in 4th Semester Master in Computer Application (MCA) program of Biju Patnaik University of Technology, Odisha for the year 2021-23.

As per the university curriculum, he has successfully completed the major project title as "Examination System" under the guidance of (External Guide/Trainer Name).

GUIDED BY:

Mr.Subhasish Mohapatra

(External Guide/Trainer Name)
Designation with Company name and Seal

Annexure-II

DECLARATION

I undersigned hereby declare that, the project report entitled "Examination System" is developed and submitted by me for the partial fulfillment of the Master of Computer Application (MCA) under the guidance of Prof. Subhasis Mohaptra, Assistant Professor, Dept. of MCA

All the material obtained for this project is exclusively based on information collected and research done during the system study and have not from part of any other reports previously.

Saswat Ranjan Mohanty Dt.

Regd No:- 2105274095

CERTIFICATION OF COMPLETION

This is to certify that, Mr. Saswat Ranjan Mohanty bearing Regd. No 2105274095 is a bona-fide student studying in 4th Semester Master in Computer Application (MCA) program of Biju Patnaik University of Technology, Odisha for the year 2021-23.

As per the university curriculum, he has successfully completed the major project title as "Examination System" under the guidance of (External Guide/Trainer Name).

GUIDED BY:

Mr.Subhasish Mohapatra

(External Guide/Trainer Name)
Designation with Company name and Seal

CERTIFICATION OF COMPLETION

This is to certify that, Mr. Saswat Ranjan Mohanty bearing Regd. No 2105274095 is a bona-fide student studying in 4th Semester Master in Computer Application (MCA) program of Biju Patnaik University of Technology, Odisha for the year 2021-23.

As per the university curriculum, he has successfully completed the major project title as "Examination System" under the guidance of (External Guide/Trainer Name).

GUIDED BY:

Mr.Subhasish Mohapatra

(External Guide/Trainer Name)
Designation with Company name and Seal

Annexure-II

ACKNOWLEDGEMENT

It gives me immense pleasure to express my deepest sense of gratitude and sincere thanks to my highly respected and esteemed guide Prof. Subhasis Mohapatra, Dept. of MCA, USBM for his valuable guidance, encouragement and help for completing this work. His useful suggestions for this whole work and co-operative behavior are sincerely acknowledged.

I would like to express my sincere thanks to Dr. Sanjib Kumar Das, Principal, USBM for giving me this opportunity to undertake this project.

I also wish to express my gratitude to Prof. S.E.Reddy, Program Coordinator, MCA for his kind hearted support. I am also grateful to my faculty members for their constant support and guidance.

I also wish to express my indebtedness to my parents as well as my family member whose blessings and support always helped me to face the challenges ahead.

At the end I would like to express my sincere thanks to all my friends and others who helped me directly or indirectly during this project work.

Place: Bhubaneswar

Date:

Student Name: - Saswat Ranjan Mohanty

Enrolment no: - 2105274095

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION :

1.1 Background

The Examination System project is a console application developed to manage and automate the process of conducting examinations.

The system provides functionalities for creating and managing exam questionnaires, allowing students to take exams, and generating result reports. It is designed to simplify the examination process, reduce administrative efforts, and provide accurate and efficient evaluation of student performance.

1.2 Objectives

The objective of this program is to create an Examination System that allows students to take an exam consisting of multiple-choice questions and stores their exam results in a database.

The program also provides an admin interface to add questions to the exam and view the marks obtained by students.

1.3 Purpose, Scope, and Applicability :

The main objectives of the Examination System project are as follows:

1. Develop a console application to facilitate the examination process.
2. Create and manage exam questionnaires.
3. Allow students to take exams and provide instant feedback.

4. Automate the evaluation of exam answers and generate result reports.
5. Ensure data security and integrity.

1.4 Achievements :

This program allows for the creation of an interactive examination system with several functionalities, which can achieve the following:

1. **Conduct Exams:** The program enables students to take exams consisting of multiple-choice questions. Students can answer the questions within a specified time duration, and the program automatically calculates their obtained marks.

2. **Automated Grading:** The program automatically grades the exams submitted by students. It compares the student's selected answers with the correct answers and calculates the obtained marks for each question and the overall exam.

3. **Student Registration:** The program provides a student signup functionality, allowing new students to create accounts to access the exam system.

4. **Student Authentication:** The program allows students to log in with their registered credentials, ensuring secure access to the exam system.

5. **Admin Privileges:** The program provides admin login functionality, enabling authorized individuals to manage the examination system. Admins can add new questions to the exam, view student marks, and perform other administrative tasks.

6. **Database Integration:** The program integrates with a MySQL database to store student information, exam questions, and exam results. This ensures data persistence and facilitates data management.

7. **Question Bank Management:** Admins can add new questions to the exam, expanding the question bank. This ensures variety and randomization in exams for different students.

8. **Reporting:** The program allows admins to view and analyze student marks, providing insights into student performance and overall exam results.

9. **Time Limit Management:** The program includes a timer function to limit the duration of the exam, automatically submitting the exam when the time is up.

10. **Flexibility:** The program is modular and easily extendable. It can be further enhanced to include additional features like multiple exam types, various question formats, and more complex grading algorithms.

Overall, this program provides an efficient and automated platform for conducting exams, reducing manual grading efforts, and facilitating data management. It benefits both students and administrators, making the examination process more organized and streamlined.

CHAPTER 2: SURVEY OF TECHNOLOGIES :

1. Java: The core programming language used for implementing the project. Java is known for its portability, object-oriented approach, and extensive standard libraries, making it suitable for building robust applications.
2. JDBC (Java Database Connectivity): JDBC is used to establish a connection to the MySQL database and perform database operations like inserting, updating, and retrieving data. It allows seamless integration between Java and the relational database management system.
3. MySQL: The chosen database management system for storing exam questions, options, student data, and exam results. MySQL is a widely used open-source RDBMS known for its reliability, performance, and ease of use.
4. ExecutorService and Executors: These classes from the java.util.concurrent package are used to manage the timer functionality for the exam. The ExecutorService is used to schedule and execute tasks asynchronously.
5. Multi-threading: Multi-threading is utilized to handle the exam timer. It enables the program to perform multiple tasks concurrently, ensuring accurate timing and responsiveness during the exam.
6. Scanner: The Scanner class from java.util package is used to read user input from the console. It enables interaction with users and allows them to enter their exam answers and other relevant information.
7. SQL (Structured Query Language): SQL is used to create, retrieve, update, and delete data in the MySQL database. SQL queries are employed to insert student records, exam questions, options, and results.
8. PreparedStatement: PreparedStatements are used for parameterized SQL queries to prevent SQL injection attacks and enhance the security of database operations.

9. Exception Handling: The project employs exception handling mechanisms to catch and handle runtime errors gracefully, providing informative error messages to users.
10. Object-Oriented Programming (OOP): The project follows OOP principles, including encapsulation, inheritance, and polymorphism, to create reusable and modular code.
11. ArrayList: The ArrayList class from the java.util package is used to store exam questions, options, and other collections of objects dynamically.
12. InputMismatchException: The InputMismatchException class from the java.util package is used to handle exceptions caused by incorrect user input during the exam.
13. Statement and PreparedStatement: These classes from the java.sql package are used to execute SQL statements and parameterized queries against the database.
14. try-with-resources: The try-with-resources statement is used to automatically close resources like database connections, statements, and result sets, enhancing code readability and reducing resource leaks.

CHAPTER 3: REQUIREMENTS AND ANALYSIS :

3.1 Problem Definition :

1. **Student Functionality:**
    - Students can log in or sign up to the system.
    - Once logged in, students can take an exam, which includes a collection of multiple-choice questions.
    - Each question has a predefined set of options, and students can select one option as their answer.
    - The exam is timed, and students must complete it within a specified duration.
    - After submitting the exam, students can view their obtained marks and the total marks.
2. **Administrative Functionality:**
    - Admins can log in to the system using predefined credentials.
    - Admins have the authority to add new questions to the question bank.
    - Each question can have multiple options, and the admin can specify which option is correct.
    - Admins can view all student exam results, including student names and obtained marks.
3. **Database Management:**
    - The system uses a MySQL database to store question details, options, student information, and exam results.
    - Database operations include retrieving questions, storing student information, and recording exam results.

High-Level Workflow:

1. **Admin Login:**

- An admin can log in using predefined admin credentials.
- Once logged in, the admin can access administrative functionalities.

2. **Student Login/Signup:**
    - A student can either log in with existing credentials or sign up as a new student.
    - For sign up, the student provides a unique username and password.
3. **Exam Taking:**
    - After login, the student can start the exam.
    - The system presents each question with its options to the student.
    - The student selects one option for each question or skips the question.
    - The exam is timed, and once the time limit is reached, the exam is automatically submitted.
4. **Exam Submission:**
    - After the student completes the exam or the time is up, the exam is submitted.
    - The system calculates the obtained marks based on the selected options.
    - The obtained marks and total marks are stored in the database along with the student's name.
5. **Admin Functions:**
    - The admin can add new questions to the question bank.
    - Each question can have multiple options, and the correct option is specified by the admin.
    - The admin can view all student exam results, including names and obtained marks.

Constraints and Scope:

- The system is designed for small-scale usage and may not be optimized for high concurrency.
- It uses a single-threaded executor for the timer, which may not be the most efficient approach for handling multiple exams concurrently.
- The database schema and codebase are suitable for managing a limited number of questions and students.
- The security measures in this implementation are basic and may not be sufficient for production use. For production, additional security measures should be implemented, such as password hashing and encryption of sensitive data.

Overall, the Examination System aims to provide a basic platform for conducting exams and managing student results, suitable for small-scale usage in educational institutions or training programs.

3.2 Requirements Specification :

Functional Requirements:

1. **Student Functionality:**
    - The system shall allow students to log in with existing credentials.
    - The system shall allow students to sign up as new users with a unique username and password.
    - Students shall be able to start the exam, which consists of multiple-choice questions with predefined options.
    - Each question shall have a specified number of marks.
    - The system shall present questions one by one to the student and display the available options.
    - Students shall select one option for each question or skip the question if needed.
    - The system shall automatically submit the exam when the time limit is reached.

- After exam submission, the system shall calculate and display the obtained marks.

2. **Admin Functionality:**
   - The system shall allow admins to log in using predefined admin credentials.
   - Admins shall have the authority to add new questions to the question bank.
   - Each question added by admins shall have multiple options, and the correct option should be specified.
   - Admins shall be able to view all student exam results, including names and obtained marks.
3. **Database Management:**
   - The system shall use a database to store question details, options, student information, and exam results.
   - The database shall store questions along with their options and marks.
   - Student information, including usernames and passwords, shall be securely stored in the database.
   - The database shall record exam results, including student names and obtained marks.

Non-Functional Requirements:

1. **Security:**
   - The system shall ensure the secure storage of student information and credentials.
   - Passwords shall be hashed and stored securely in the database.
   - Only authorized users (students and admins) shall be allowed to access the system.
2. **User Interface:**
   - The user interface shall be intuitive and user-friendly for both students and admins.
   - Clear instructions and messages shall be provided during the exam to guide students.

3. **Performance:**
    - The system shall handle multiple concurrent exam sessions without significant performance degradation.
    - The exam timer and submission process shall be efficient and responsive.
4. **Scalability:**
    - The database and system design shall be scalable to accommodate a growing number of questions and students.
5. **Error Handling:**
    - The system shall handle errors and exceptions gracefully and display meaningful error messages to users.
    - Proper validation shall be applied to user inputs to prevent invalid data entry.
6. **Compatibility:**
    - The system shall be compatible with common web browsers and operating systems.
7. **Data Backup and Recovery:**
    - Regular data backups shall be performed to ensure data integrity and quick recovery in case of any system failure.

Constraints and Scope:

- The system is intended for small-scale usage and may not be suitable for high-demand scenarios with thousands of concurrent users.
- It is assumed that a local MySQL database is available for use.
- Security measures will be basic in this implementation and should be enhanced for production usage.

3.3 Planning and Scheduling :

Planning and scheduling are critical aspects of any software development project to ensure successful and timely completion. Here's an outline of the planning and scheduling process for the Examination System project:

1. Project Scope and Objectives:

- Define the scope of the project, including the features and functionalities to be implemented.
- Set clear objectives, such as creating a user-friendly examination platform for students and efficient management tools for admins.

2. Requirement Gathering:

- Gather detailed requirements through discussions with stakeholders (educators, administrators, and students).
- Document functional and non-functional requirements to serve as a reference throughout the development process.

3. Technology Stack Selection:

- Decide on the appropriate technology stack for development, including the programming language (Java in this case), database (MySQL), and any relevant frameworks or libraries.

4. High-Level Design:

- Create a high-level design of the system architecture, database schema, and user interfaces.
- Define the interactions between different modules, including the flow of data and information.

5. Task Breakdown and Estimation:

- Break down the development tasks into smaller units, such as creating classes for Option, Question, and Exam, setting up database connections, and implementing the user interfaces.
- Estimate the effort required for each task, considering factors like complexity and dependencies.

6. Development Iterations:

- Plan the development process in iterations (sprints) to facilitate incremental development and testing.
- Prioritize the features to be developed in each iteration based on their importance and dependencies.

7. Implementation:

- Start implementing the code based on the high-level design and task breakdown.
- Collaborate with team members (if applicable) to ensure smooth progress and code quality.

8. Testing and Quality Assurance:

- Conduct unit testing for individual modules to check their functionality and identify bugs early in the development process.
- Perform integration testing to verify that different modules work together as expected.
- Conduct user acceptance testing (UAT) with real users to gather feedback and address any issues.

9. Database Setup and Data Migration:

- Set up the MySQL database and define the required tables and relationships.

- Migrate sample data or use scripts to populate the database with test data for testing purposes.

10. User Documentation:

- Prepare user documentation and guides for students and admins, explaining how to use the system effectively.

11. Deployment and Production Readiness:

- Deploy the application on a testing environment for final rounds of testing and validation.
- Ensure that all necessary security measures are in place, such as encrypted passwords and secure database connections.
- Conduct load testing to assess the system's performance under simulated production conditions.

12. Launch and Support:

- After successful testing and validation, deploy the Examination System to the production environment.
- Monitor the system closely during the initial days to address any unforeseen issues.
- Provide ongoing support and maintenance, including bug fixes and updates as needed.

13. Timeline and Milestones:

- Create a project timeline with milestones to track progress and ensure timely completion.
- Regularly review the progress against milestones and adjust the schedule if necessary.

3.4 Software and Hardware Requirements :

SOFTWARE REQUIREMENTS

The Examination System project is developed using the following technologies:

- Programming Language: Java
- Database: MySQL
- Database Connector: JDBC (Java Database Connectivity)

HARDWARE REQUIREMENTS:

Processor:- Pentium IV or higher, (PIV-300GHz recommended)

Primary Memory:- 512 MB or More

HDD :- Min 1 GB hard free drive space

3.5 Preliminary Product Description :

The Examination System is a Java-based software application designed to facilitate the conduction of exams in an educational setting. The system provides a user-friendly interface for both students and administrators, enabling students to take exams and administrators to manage the exam process efficiently. The primary objective of the system is to automate the exam-taking and grading process, reducing manual efforts and ensuring accurate and timely results.

Key Features:

1. **Student Registration and Login:** The system allows students to create accounts by registering with their username and password. Once registered, students can log in using their credentials to access the exam interface.

2. **Exam Management:** The system allows administrators to add multiple-choice questions to the exam. Each question can have multiple options, and administrators can mark the correct answer for each question.

3. **Timer-Based Exam:** The system incorporates a timer function to control the duration of the exam. Students are allowed to answer questions within the specified time limit, after which the exam is automatically submitted.

4. **Automated Grading:** Upon exam submission, the system automatically grades the exam based on the selected answers. It compares the student's answers with the correct answers stored in the database to calculate the obtained marks.

5. **Database Integration:** The system integrates with a MySQL database to store student information, exam questions, and exam results. This ensures data persistence and easy management of student records.

6. **Admin Privileges:** Administrators have exclusive access to certain functionalities, including question bank management and viewing student marks. Admins can log in using their credentials to access these privileges.

7. **Student Marks Reporting:** The system allows administrators to view and analyze student marks. They can see the obtained marks of each student and the total marks scored in each exam.

Usage:

1. **Student Perspective:**
    - Students can sign up for a new account by providing a unique username and password.
    - After successful registration, students can log in to the system using their credentials.
    - Once logged in, students can access the exam interface and start the exam within the specified time duration.
    - Students answer multiple-choice questions, and the system automatically calculates their obtained marks upon exam submission.
2. **Admin Perspective:**
    - Administrators log in using their unique admin credentials.
    - Admins can add new multiple-choice questions to the exam, enhancing the question bank for future exams.
    - They have access to view and analyze student marks, providing valuable insights into student performance.

Future Enhancements:

The preliminary version of the Examination System lays the foundation for a robust exam management platform. Future enhancements may include:

- Support for different question types, such as true/false, fill in the blanks, and essay questions.

- Customizable exam durations and options for setting different time limits for different exams.
- Improved reporting and data visualization for a more comprehensive analysis of student performance.
- Integration with other authentication mechanisms, such as OAuth, to enhance security and user convenience.

Note: This preliminary product description outlines the core features and functionalities of the Examination System. Further development and refinement will be carried out to meet specific requirements and enhance the overall user experience.

3.6 Conceptual Models :

1. **Use Case Diagram:** The Use Case Diagram presents the interactions between different actors (students and admins) and the main functionalities of the Examination System.

    - Actors:
        - Student: Represents a student who can sign up, log in, and take exams.
        - Admin: Represents an administrator who can log in, manage questions, and view student marks.

    - Use Cases:
        - Student Signup: Allows a student to create a new account by providing a unique username and password.

- Student Login: Enables a student to log in using their registered credentials.
- Take Exam: Allows a student to take the exam, answer multiple-choice questions, and submit the exam.
- Admin Login: Enables an admin to log in using their unique admin credentials.
- Add Questions: Allows an admin to add multiple-choice questions to the exam.
- View Student Marks: Allows an admin to view and analyze student marks.

2. **Class Diagram:** The Class Diagram illustrates the classes and their relationships in the Examination System.

   - Classes:
     - Option: Represents an option for a multiple-choice question, containing the text of the option and a flag indicating whether it is the correct answer.
     - Question: Represents a single question with its options and associated marks.
     - Exam: Represents a collection of questions that make up the exam.
     - Student: Represents a student with attributes like username and password.
     - Admin: Represents an administrator with attributes like username and password.

- Relationships:
    - Student and Exam: Association representing the student taking the exam.
    - Admin and Question: Association representing the admin adding questions to the exam.
    - Exam and Question: Aggregation representing the questions included in the exam.
    - Question and Option: Aggregation representing the options available for each question.

These conceptual models provide a visual representation of the Examination System's structure, behavior, and interactions. They aid in understanding the system's architecture and flow of information between different components.

CHAPTER 4: SYSTEM DESIGN :

4.1 Basic Modules :

1. **Option Class:**
    - Description: Represents an option for a multiple-choice question, containing the text of the option and a flag indicating whether it is the correct answer.
    - Attributes:
        - **text**: String - The text of the option.
        - **isCorrect**: boolean - Indicates whether the option is the correct answer.
2. **Question Class:**

    - Description: Represents a single question with its options and associated marks.
    - Attributes:
        - **id**: int - Unique identifier for the question.
        - **text**: String - The text of the question.
        - **options**: List<Option> - List of options associated with the question.
        - **marks**: int - Marks assigned to the question.
        - **obtainedMarks**: int - Marks obtained by the student for this question (-1 if not answered).

3. **Exam Class:**

    - Description: Represents a collection of questions that make up an exam.
    - Attributes:
        - **questions**: List<Question> - List of questions included in the exam.
        - **studentName**: String - Name of the student taking the exam.
        - **totalMarks**: int - Total marks for the exam.

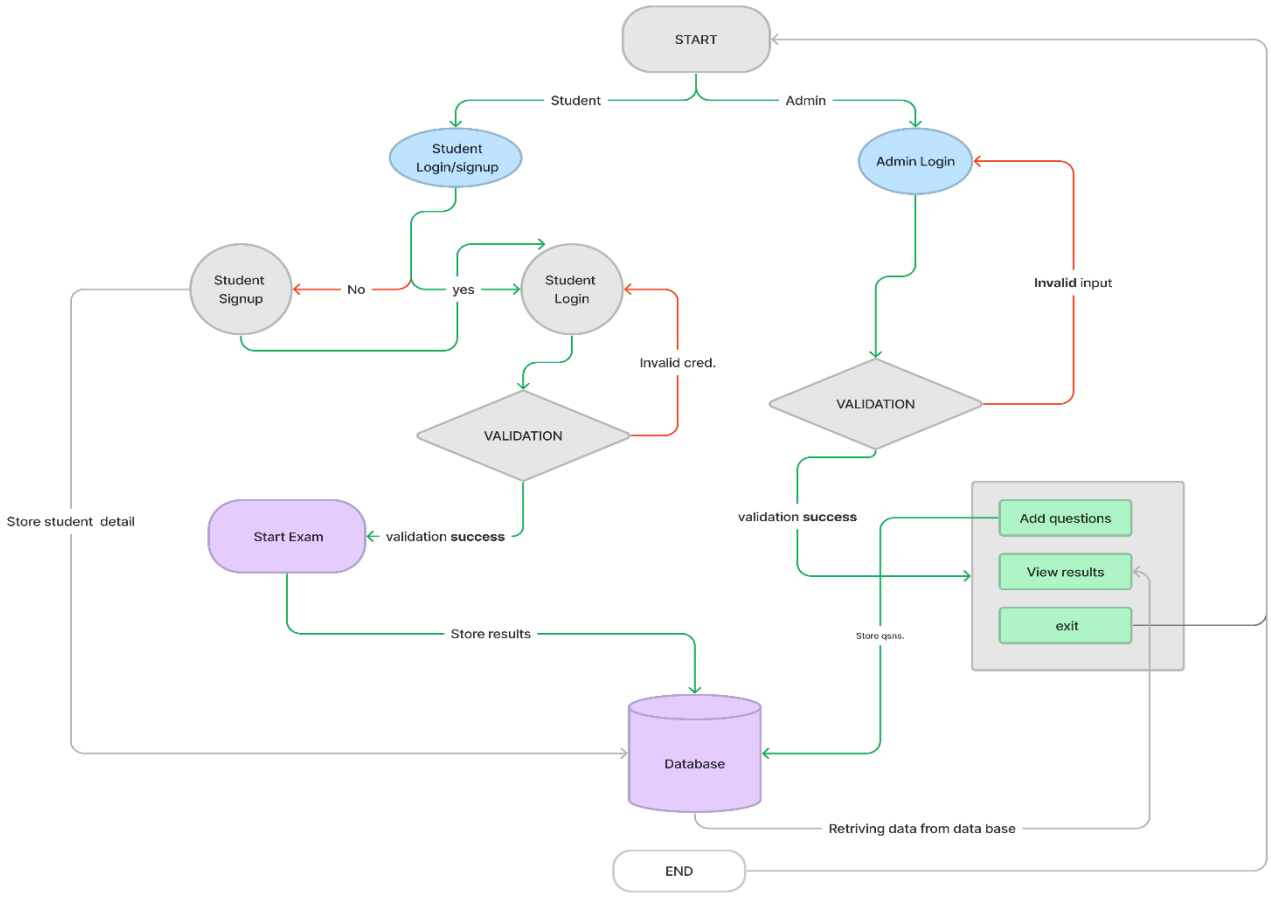
- **obtainedMarks**: int - Total marks obtained by the student in the exam.

- Methods:
  - **addQuestion(Question question)**: Adds a question to the exam.
  - **startExam(int duration)**: Starts the exam for the student and prompts for answers.
  - **submitExam()**: Submits the exam and stores the obtained marks in the database.
  - **displayAllStudentMarks()**: Displays all student marks from the database.

4. **ExaminationSystem Class:**
   - Description: The main class for the Examination System.
   - Methods:
     - **main(String[] args)**: Contains the main logic for user interaction and navigation in the system.
     - **getValidatedInput(String prompt, int min, int max)**: Validates user input within a specified range.
     -

These modules define the core functionalities of the Examination System. The Option, Question, and Exam classes handle question creation, management, and exam taking, while the ExaminationSystem class manages user interactions and the main flow of the program.

4.2. DFD, UML Diagrams:



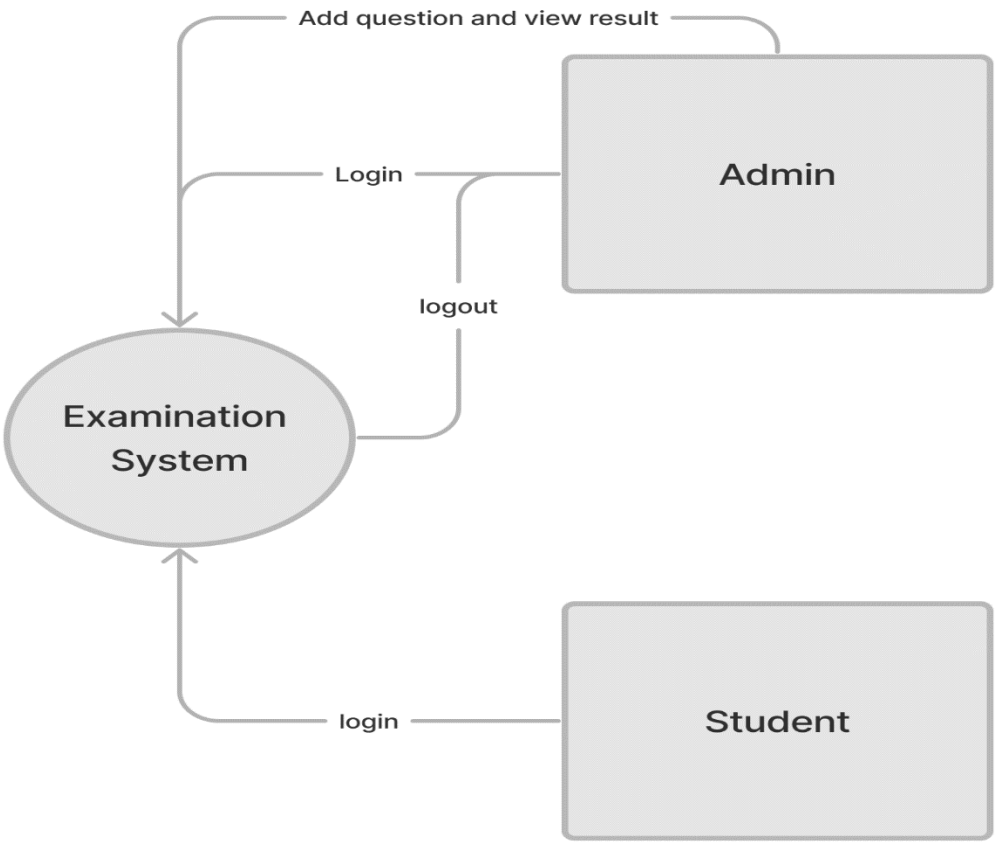
System Diagram

Data Flow Diagram:

Data flow diagram depicts the way data is processed by a system, the inputs and outputs with a special focus on the information flow and storage.

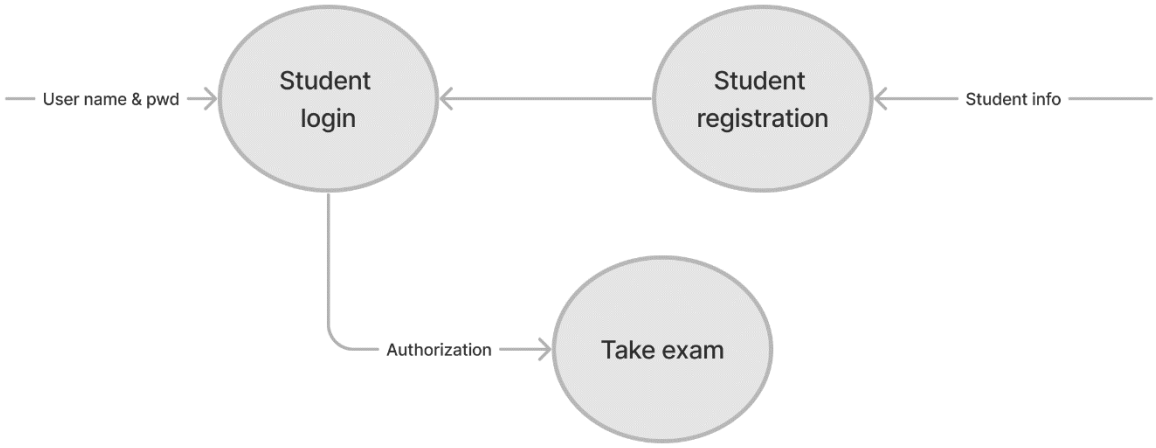
Level – 0 DFD:

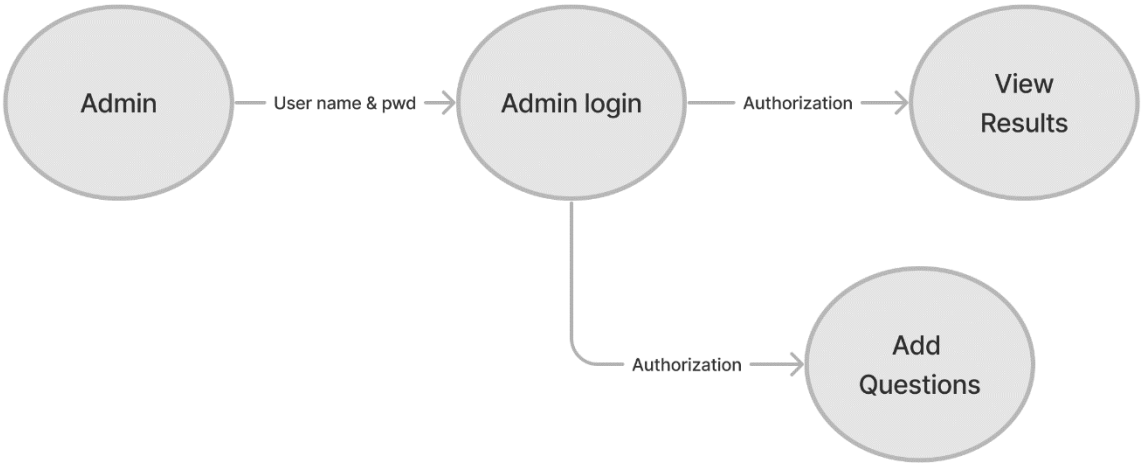
Level - 0 DFD is also known as context diagram. Its shows an abstract view of main processes .



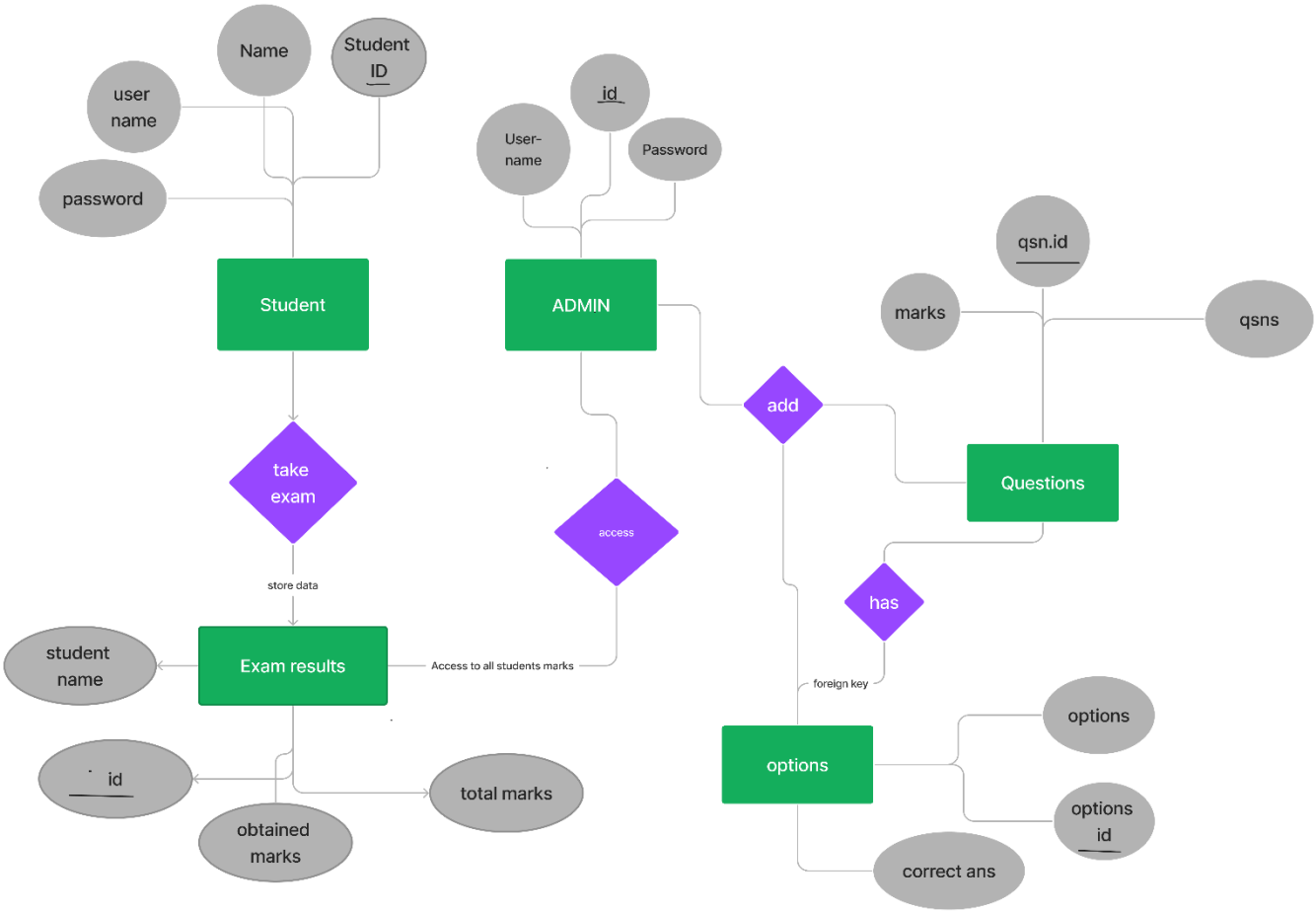
Level – 1 DFD:

Level 1 DFD enumerates all of the major sub processes that comprise the entire system. It also identifies the data store of the student record, which contains all of the student's data, which contains several master data used throughout the system.





4.3. E-R Diagram:



CHAPTER 5: IMPLEMENTATION AND TESTING :

5.1 Implementation Approaches :

1. **Designing the Database Schema:**
    - Define the tables for storing information about students, questions, options, and exam results.
    - Create a relational database to store and manage this data.

2. **Implementing the Database Connectivity:**
    - Use JDBC (Java Database Connectivity) to establish a connection with the database.
    - Implement methods to perform CRUD (Create, Read, Update, Delete) operations on the database, such as adding questions, options, and retrieving student marks.

3. **Creating Classes for Option and Question:**
    - Implement the Option and Question classes to represent the data model for the questions and options.
    - Include methods for adding options to questions and calculating obtained marks for each question.

4. **Implementing the Exam Class:**
    - Create the Exam class to manage a collection of questions and the exam-taking process.
    - Implement methods to start the exam, display questions and options, and record student answers.

5. **User Interface:**
    - Design a user interface to interact with the program.
    - Implement menus for admin and student login/signup, exam start, and displaying student marks.
    - Handle user input and navigate through different functionalities.

6. **Admin Functionality:**
    - Allow admin users to log in with their credentials.
    - Implement admin functionalities, such as adding questions to the database and displaying all student marks.

7. **Student Functionality:**
    - Implement student login and signup functionalities.
    - Allow students to start the exam, view questions, select options, and submit the exam.
    - Record student answers and calculate the obtained marks.

8. **Timer Functionality:**
    - Implement a timer for the exam to limit the duration of the exam.
    - Automatically submit the exam when the time is up.

9. **Error Handling and Input Validation:**
    - Include error handling mechanisms to catch and handle exceptions gracefully.
    - Validate user input to ensure correct data entry.

10.Testing:

- Conduct extensive testing to ensure the proper functioning of the system.
- Test different scenarios, such as various question types, multiple students taking exams simultaneously, and handling edge cases.

11.Documentation:

- Provide clear and concise documentation for the code, including comments, method descriptions, and overall project documentation.

12.Security Considerations:

- Implement security measures to protect sensitive data, such as encrypting passwords and preventing SQL injection attacks.

13.Deployment:

- Package the project for deployment, making it accessible to users.

These implementation approaches will help build a robust and user-friendly Examination System.

5.2 Coding Details and Code Efficiency :

Coding Details: The given project is an Examination System implemented in Java, using MySQL as the database for storing questions, options, student information, and exam results. Let's discuss the coding details and how various functionalities are implemented:

1. **Database Connectivity:**
    - The **getConnection()** method establishes a connection to the MySQL database using JDBC.
    - The database credentials (URL, username, password) are hardcoded in the **getConnection()** method.

2. **Option and Question Classes:**
    - The **Option** class represents an option for a question, with text and a boolean flag indicating if it's correct.
    - The **Question** class represents a single question, containing an ID, text, marks, and a list of options.
    - The **Question** class includes a method to add options to a question.
3. **Exam Class:**

    The **Exam** class manages a collection of questions and the exam-taking process.
    - It includes methods for starting the exam, displaying questions and options, and recording student answers.

- The **startExam()** method uses a timer to limit the exam duration and automatically submits the exam when the time is up.
- The **submitExam()** method calculates the obtained marks and stores the results in the database.

4. **Admin Functionality:**
    - Admin login is checked against the **admins** table in the database.
    - Admins can add questions and options to the database using the **addQuestions()** method.
5. **Student Functionality:**
    - Students can log in or sign up.
    - Student login is checked against the **students** table in the database.
    - After login, students can start the exam, view questions, select options, and submit the exam.
6. **User Interface:**
    - The user interface is implemented through console interactions using the **Scanner** class for input.
    - Menus are presented to users for various functionalities (admin login, student login/signup, exam start, etc.).

7. **Error Handling and Input Validation:**
    - The program includes input validation for user inputs to ensure correct data entry.
    - Try-catch blocks are used to handle exceptions gracefully.

Code Efficiency: Efficiency in software can be assessed in several ways, such as performance, maintainability, readability, and resource utilization. Here are some aspects related to code efficiency in this project:

1. **Performance:** The code handles basic functionalities and database operations effectively. However, it lacks optimization for large-scale deployments. For better performance, additional techniques like connection pooling and query optimization can be implemented.

2. **Maintainability and Readability:** The code demonstrates good modularity by separating functionalities into classes and methods. However, it could benefit from more comments and documentation to enhance code readability and ease of maintenance.
3. **Database Handling:** The code directly includes database credentials, which is not recommended for production use. A more secure approach would be to externalize the credentials and use environment variables or configuration files.

4. **Input Validation:** The code implements input validation for user inputs, preventing erroneous data from being processed.
5. **Scalability:** The current implementation is suitable for small-scale usage. For larger systems, the database schema and codebase might need further optimizations to handle higher user loads efficiently.
6. **Error Handling:** The code includes basic error handling with try-catch blocks. However, it could be improved to handle specific exceptions and provide informative error messages to users.
7. **Timer Implementation:** The timer for the exam uses a single-threaded executor. While it works for small-scale exams, it might not be the most efficient approach for larger exams with multiple concurrent users.

5.3 Testing Approach :

Testing is a crucial part of the software development process to ensure that the Examination System works as expected, is reliable, and meets the specified requirements. Here's an outline of the testing approach for the project:

1. Unit Testing:

- Conduct unit testing for individual components (classes) to verify their functionality in isolation.
- Test methods within the Option, Question, and Exam classes to ensure they produce the correct output and handle edge cases.
- Use testing frameworks like JUnit to automate unit tests and streamline the testing process.

2. Integration Testing:

- Perform integration testing to validate the interactions between different modules, such as Question and Option classes, and the Exam class.
- Test the data flow between classes and the database to ensure data integrity and accuracy.

3. User Interface (UI) Testing:

- Validate the user interfaces for both the student and admin sides.
- Test all user interactions, buttons, forms, and input validations to ensure a smooth user experience.
- Check for responsive design and compatibility across different devices and screen sizes.

4. Functional Testing:

- Conduct functional testing to ensure that all specified features and functionalities work correctly.

- Test various exam scenarios, including answering questions, skipping questions, and submitting the exam.
- Verify that obtained marks are calculated accurately based on the selected options.

5. Security Testing:

- Perform security testing to identify vulnerabilities and ensure the system is protected against common security threats.
- Test for SQL injection, cross-site scripting (XSS), and other potential security risks.
- Validate user authentication and authorization mechanisms to prevent unauthorized access.

6. Performance Testing:

- Conduct performance testing to assess the system's response time and scalability.
- Test the system with multiple concurrent users to ensure it can handle the expected load during peak times.
- Identify and address any performance bottlenecks.

7. Usability Testing:

- Conduct usability testing with real users to gather feedback on the system's ease of use and user-friendliness.
- Incorporate user feedback to make improvements to the user interface and overall user experience.

8. End-to-End Testing:

- Perform end-to-end testing to validate the complete flow of the examination process, from student login to exam submission and result storage.
- Ensure that the system works seamlessly without any interruptions or errors.

9. Regression Testing:

- Implement regression testing to verify that new changes and updates do not introduce new issues or impact existing functionalities.
- Rerun existing tests after code changes to confirm that all previously working features remain intact.

10. Data Integrity Testing:

- Test data storage and retrieval to ensure that data is correctly saved to and retrieved from the database.
- Check for data consistency and accuracy.

11. Compatibility Testing:

- Test the system on different web browsers, operating systems, and devices to ensure compatibility and a consistent user experience.

12. Error Handling and Exception Testing:

- Verify that the system handles errors and exceptions gracefully.
- Test scenarios where incorrect data is provided or unexpected inputs are entered.

13. User Acceptance Testing (UAT):

- Involve real users (students and admins) to perform user acceptance testing to ensure that the system meets their needs and expectations.
- Address any feedback received during UAT.

14. Continuous Testing:

- Implement continuous integration and continuous testing practices to automatically run tests with each code commit.
- Use automated testing tools to accelerate the testing process and catch issues early.

15. Test Documentation:

- Document all test cases, expected outcomes, and actual results for reference and future regression testing.

5.3.1Unit Testing :

Unit testing is a crucial aspect of software development that involves testing individual units or components of code in isolation to ensure they function correctly. In the context of the Examination System project, unit testing is performed to verify the correctness of individual classes and methods. The goal is to identify any bugs or issues early in the development process, making it easier to maintain and enhance the codebase.

Unit Testing Approach:

For unit testing the Examination System project, we use the JUnit testing framework, one of the most popular testing libraries for Java applications. JUnit provides a simple and efficient way to write test cases and perform assertions to check whether the actual output matches the expected output.

Setting up JUnit:

To use JUnit for testing, you need to add the JUnit dependency to the project. This can be done using build tools like Maven or Gradle. Here's an example of how to add JUnit as a dependency in Maven:

Writing Unit Tests:

Unit tests should be written for critical classes and methods in the project. It's important to test various scenarios, including boundary cases, edge cases, and common use cases. For the Examination System project, we can create test classes for individual classes like **Question**, **Exam**, and **ExaminationSystem**. Each test class should test the methods of the corresponding class.

Running Unit Tests:

To run the unit tests, you can use your IDE's testing capabilities or use build tools like Maven or Gradle to execute the tests. After running the tests, you will get feedback on whether the code passes all the test cases or if any tests fail, helping you identify and fix issues early in the development process.

5.4 Modifications and Improvements :

1. **Admin Authentication and Security Enhancement:**
    - Implement a secure authentication mechanism for admin login, such as using hashed passwords and salting to store admin credentials securely in the database.
    - Add roles and permissions to limit access to specific admin features based on their roles (e.g., super admin, regular admin).
2. **Student Registration and Authentication:**
    - Allow students to register and create accounts with unique usernames and passwords.
    - Implement a secure authentication mechanism for student login.
3. **Randomizing Question Order:**
    - Randomize the order of questions presented to students in the exam to prevent cheating and make each exam unique.
4. **Enhanced Question Types:**
    - Add support for various question types, such as multiple-choice with multiple correct answers, true/false, fill in the blanks, etc.

5. **Question Categories and Tags:**
    - Allow admins to categorize and tag questions to better organize them and enable students to filter questions based on categories during exams.
6. **Feedback Mechanism:**
    - Implement a feedback mechanism for students to provide feedback on the exam difficulty, question quality, and overall user experience.
7. **Time Extensions and Grace Periods:**
    - Allow admins to grant time extensions or define grace periods for specific students based on their needs or circumstances.
8. **Graceful Error Handling:**
    - Implement proper error handling throughout the application to provide meaningful error messages and prevent crashes.
9. **Data Validation and Sanitization:**
    - Implement data validation and sanitization to prevent common security vulnerabilities like SQL injection and cross-site scripting (XSS) attacks.
10. **User-Friendly UI Enhancements:**
    - Improve the user interface (UI) to make it more intuitive and visually appealing for both admins and students.
11. **Database Backup and Restore:**
    - Implement a backup and restore mechanism for the database to safeguard data in case of any data loss or corruption.
12. **Performance Optimization:**
    - Analyze and optimize the code to improve performance, especially for large-scale usage.

13. **Multilingual Support:**
    - Add support for multiple languages to cater to users from different regions.
14. **Accessibility Improvements:**
    - Ensure the application is accessible to users with disabilities by following accessibility guidelines.
15. **Logging and Monitoring:**
    - Implement logging and monitoring to track application usage, identify potential issues, and monitor system health.
16. **Automated Testing:**
    - Set up automated testing using tools like JUnit and integration testing to ensure the code remains reliable during future updates and changes.
17. **Version Control:**
    - Use version control systems like Git to track changes and collaborate efficiently with team members.
18. **Documentation:**
    - Maintain comprehensive documentation for the codebase, API endpoints, and other relevant information to help future developers understand and work on the project.

SAMPLE RESULT :

Start with admin menu:

```
ExaminationSystem
================================================
================================================
=== Welcome to the Examination System! ===
************************************************
Are you a student or an admin? (student/admin): admin
================================
=== Admin Login ===
Enter admin username: saswat
Enter admin password: saswat
Invalid admin credentials. Please try again.
Enter admin username: saswat1234
Enter admin password: saswat1234
Admin login successful.

====== Admin Menu =======
1. Add Questions
2. Display Student Marks
3. Logout
Enter your choice:
```

Admin menu functionality:

1.Add questions:

```
****************************************
Are you a student or an admin? (student/admin): admin
================================
=== Admin Login ===
Enter admin username: saswat1234
Enter admin password: saswat1234
Admin login successful.

====== Admin Menu =======
1. Add Questions
2. Display Student Marks
3. Logout
Enter your choice: 1
========================
=== Add Questions ===
Enter the question text: what is react
Enter the marks for this question: 10
Enter option text (or 'exit' to finish adding options): framework
Is this option correct? (true/false): false
Enter option text (or 'exit' to finish adding options): package
Is this option correct? (true/false): true
Enter option text (or 'exit' to finish adding options): tag
Is this option correct? (true/false): false
Enter option text (or 'exit' to finish adding options): webpack
Is this option correct? (true/false): false
Enter option text (or 'exit' to finish adding options): exit
Question added successfully!
====== Admin Menu =======
1. Add Questions
2. Display Student Marks
3. Logout
Enter your choice:
```

2.Display result:

```
==================================
Student ID: 71
Student Name: sam
Obtained Marks: 40
Total Marks: 230
==================================
==================================
Student ID: 75
Student Name: raj
Obtained Marks: 160
Total Marks: 230
==================================
==================================
Student ID: 76
Student Name: roj
Obtained Marks: 30
Total Marks: 230
==================================
==================================
Student ID: 79
Student Name: ran
Obtained Marks: 10
Total Marks: 230
==================================
====== Admin Menu =======
1. Add Questions
2. Display Student Marks
3. Logout
Enter your choice:
```

Student menu:

```
==================================================
==================================================
=== Welcome to the Examination System! ===
**************************************************
Are you a student or an admin? (student/admin): student
=== Student Login/Signup ===
Do you have a student account? (y/n): n
================================
=== Student Signup ===
Create a new student username: yuji
Create a new student password: yuji
Student signup successful. Please proceed with student login
=== Student Login ===
Enter student username: yuji
Enter student password: yuji
Student login successful.

Enter your name: yuji

Question 1: what is the capital of india?
1) NEW_DELHI
2) HYDRABAD
3) PANIPATH
4) BBSR
Enter your answer (1-4), or enter 0 to skip:
```

```
4) j
Enter your answer (1-4), or enter 0 to skip: 1

Question 22: who yu are?
1) a
2) b
3) c
4) d
Enter your answer (1-4), or enter 0 to skip: 1

Question 23: who is the founder of java?
1) van rossum
2) james gosling
3) brendan eich
4) no one
Enter your answer (1-4), or enter 0 to skip: 1
Exam submitted
You Can Exit now
**************************
```

Source Code

```
import java.sql.*;
import java.util.*;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;
// import java.util.List;
// import java.util.ArrayList;
// import java.util.Scanner;
// import java.util.InputMismatchException;

// Option class represents an option for a question
class Option {
    private String text;
    private boolean isCorrect;

    public Option(String text, boolean isCorrect) {
        this.text = text;
        this.isCorrect = isCorrect;
    }

    public String getText() {
        return text;
    }

    public boolean isCorrect() {
        return isCorrect;
    }
}

// Question class represents a single question with its options
class Question {
    private int id;
    private String text;
    private List<Option> options;
    private int marks;
    private int obtainedMarks = -1;

    public Question(int id, String text, int marks) {
        this.id = id;
        this.text = text;
        this.marks = marks;
        this.options = new ArrayList<>();
    }

    public void addOption(String text, boolean isCorrect) {
        Option option = new Option(text, isCorrect);
        options.add(option);
    }
```

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

    public String getText() {
        return text;
    }

    public int getMarks() {
        return marks;
    }

    public List<Option> getOptions() {
        return options;
    }

    public int getObtainedMarks() {
        return obtainedMarks;
    }

    public void setObtainedMarks(int obtainedMarks) {
        this.obtainedMarks = obtainedMarks;
    }
}

// Exam class represents a collection of questions
class Exam {
    private List<Question> questions = new ArrayList<>();
    private String studentName;
    private int totalMarks;
    private int obtainedMarks;

    public void addQuestion(Question question) {
        questions.add(question);
    }

    private static Connection getConnection() throws SQLException {
        String url = "jdbc:mysql://localhost:3306/examination";
        String username = "root";
        String password = "Saswat@0602";

        return DriverManager.getConnection(url, username, password);
    }

    // Method to start the exam and prompt for answers
    public void startExam(int duration) {
        Scanner sc = new Scanner(System.in);
        ExecutorService executorService = Executors.newSingleThreadExecutor();

        System.out.print("Enter your name: ");
        studentName = sc.nextLine();
```

```
        // USING THREAD FOR TIMER FUNCTIONS
        Runnable timerTask = () -> {
            try {
                TimeUnit.MINUTES.sleep(duration);
                System.out.println("\nTime's up! Your exam will be automatically
submitted.");
                submitExam();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        };
        executorService.submit(timerTask);

        for (int i = 0; i < questions.size(); i++) {
            Question question = questions.get(i);
            System.out.println("\nQuestion " + (i + 1) + ": " + question.getText());
            List<Option> options = question.getOptions();
            for (int j = 0; j < options.size(); j++) {
                Option option = options.get(j);
                System.out.println((j + 1) + ") " + option.getText());
            }
            int answer = getValidatedInput("Enter your answer (1-" + options.size()
+ "), or enter 0 to skip: ", 0,
                    options.size());
            if (answer == 0) {
                continue; // Skip the question
            }
            question.setObtainedMarks(options.get(answer - 1).isCorrect() ?
question.getMarks() : 0);
        }
//The shutdown() method will allow previously submitted tasks to execute before
terminating, while the  shutdownNow() method prevents waiting tasks from starting
and attempts to stop currently executing tasks.
        executorService.shutdown();
        submitExam();
    }

    private int getValidatedInput(String message, int min, int max) {
        Scanner sc = new Scanner(System.in);
        int answer;
        while (true) {
            System.out.print(message);
            try {
                answer = sc.nextInt();
                if (answer >= min && answer <= max) {
                    break;
                } else {
                    System.out.println("Invalid input! Please enter a valid
option.");
                }
            } catch (InputMismatchException e) {
```

```
                System.out.println("Invalid input! Please enter a valid option.");
                sc.nextLine(); // Clear the input buffer
            }
        }
        return answer;
    }

    //give unique id to student or retrive student using id

    private int retrieveOrAssignStudentId() {
        int studentId = -1; // Default value for student ID

        // Check if the student already exists in the database
        try (Connection connection = getConnection()) {
            String selectQuery = "SELECT student_id FROM students WHERE username =
?";
            PreparedStatement selectStmt = connection.prepareStatement(selectQuery);
            selectStmt.setString(1, studentName);
            ResultSet resultSet = selectStmt.executeQuery();

            if (resultSet.next()) {
                // Student exists, retrieve the student ID
                studentId = resultSet.getInt("student_id");
            } else {
                // Student doesn't exist, assign a new student ID
                String insertQuery = "INSERT INTO students (student_name) VALUES
(?)";
                PreparedStatement insertStmt =
connection.prepareStatement(insertQuery,
                        Statement.RETURN_GENERATED_KEYS);
                insertStmt.setString(1, studentName);
                insertStmt.executeUpdate();

                ResultSet generatedKeys = insertStmt.getGeneratedKeys();
                if (generatedKeys.next()) {
                    studentId = generatedKeys.getInt(1);
                }

            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return studentId;
    }

    private void submitExam() {
        // Calculate total marks and obtained marks
        totalMarks = getTotalMarks();
        obtainedMarks = getObtainedMarks();

        // Store the exam results in the database
```

```
        try (Connection connection = getConnection()) {

            // Retrieve or assign studentId before inserting the exam results
            int studentId = retrieveOrAssignStudentId();

            // Insert the exam results into the database (exam_results )


            String insertMarksQuery = "INSERT INTO exam_results (student_id,
student_name, obtained_marks, total_marks) VALUES (?, ?, ?, ?)";

            PreparedStatement insertMarksStmt =
connection.prepareStatement(insertMarksQuery);
            insertMarksStmt.setInt(1, studentId);
            insertMarksStmt.setString(2, studentName);
            insertMarksStmt.setInt(3, obtainedMarks);
            insertMarksStmt.setInt(4, totalMarks);
            insertMarksStmt.executeUpdate();

        } catch (SQLException ex) {
            Exception suppressedException = new Exception("Suppressed Exception");
            ex.addSuppressed(suppressedException);
            ex.printStackTrace();

        }

        // Display exam results
        System.out.println("Exam submitted");
        System.out.println("You Can Exit now");
        System.out.println("************************");
        // System.out.println("\nTotal marks: " + totalMarks);
        // System.out.println("Obtained marks: " + obtainedMarks);
        // System.out.println("Percentage: " + ((double) obtainedMarks / totalMarks)
*
        // 100 + "%");
    }

    public int getTotalMarks() {
        int total = 0;
        for (Question question : questions) {
            total += question.getMarks();
        }
        return total;
    }

    public int getObtainedMarks() {
        int total = 0;
        for (Question question : questions) {
            if (question.getObtainedMarks() != -1) {
                total += question.getObtainedMarks();
            }
        }
```

```
        return total;
    }
}

// ExaminationSystem class contains the main method to start the program

public class ExaminationSystem {
    public static void main(String[] args) throws SQLException {
        String url = "jdbc:mysql://localhost:3306/examination";
        String username = "root";
        String password = "Saswat@0602";
        Scanner sc = new Scanner(System.in);
        System.out.println("==================================================");
        System.out.println("==================================================");
        System.out.println("=== Welcome to the Examination System! ===");
        System.out.println("******************************************");

        System.out.print("Are you a student or an admin? (student/admin): ");
        String userType = sc.nextLine();

        if (userType.equalsIgnoreCase("student")) {
            // Student login or signup
            System.out.println("=== Student Login/Signup ===");
            System.out.print("Do you have a student account? (y/n): ");
            String haveAccount = sc.nextLine();

            if (haveAccount.equalsIgnoreCase("y")) {
                // Student login
                boolean studentLoggedIn = false;
                while (!studentLoggedIn) {
                    System.out.println("=== Student Login ===");
                    System.out.print("Enter student username: ");
                    String studentUsername = sc.nextLine();
                    System.out.print("Enter student password: ");
                    String studentPassword = sc.nextLine();

                    int studentId = studentLogin(studentUsername, studentPassword);

                    if (studentId != -1) {
                        studentLoggedIn = true;
                        System.out.println("Student login successful.\n");

                        Exam exam = new Exam();

                        // Retrieve questions and options from the database
                        try (Connection connection = getConnection()) {
                            // Retrieve questions
                            String selectQuestionsQuery = "SELECT * FROM questions";
                            PreparedStatement selectQuestionsStmt =
connection.prepareStatement(selectQuestionsQuery);
                            ResultSet questionResultSet =
selectQuestionsStmt.executeQuery();
```

```
                            while (questionResultSet.next()) {
                                int questionId = questionResultSet.getInt("id");
                                String questionText =
questionResultSet.getString("text");
                                int marks = questionResultSet.getInt("marks");
                                Question question = new Question(questionId,
questionText, marks);

                                // Retrieve options for each question
                                String selectOptionsQuery = "SELECT * FROM options
WHERE question_id = ?";
                                PreparedStatement selectOptionsStmt =
connection.prepareStatement(selectOptionsQuery);
                                selectOptionsStmt.setInt(1, questionId);
                                ResultSet optionResultSet =
selectOptionsStmt.executeQuery();
                                //check the correct answer
                                while (optionResultSet.next()) {
                                    String optionText =
optionResultSet.getString("text");
                                    boolean isCorrect =
optionResultSet.getBoolean("is_correct");
                                    question.addOption(optionText, isCorrect);
                                }
                                exam.addQuestion(question);
                            }
                        } catch (SQLException e) {
                            e.printStackTrace();
                        }

                        // Start the exam with a duration of 5 minutes
                        exam.startExam(5);
                    } else {
                        System.out.println("Invalid student credentials. Please try
again.");
                    }
                }
            } else if (haveAccount.equalsIgnoreCase("n")) {
                // Student signup
                System.out.println("==================================");
                System.out.println("=== Student Signup ===");
                System.out.print("Create a new student username: ");
                String studentUsername = sc.nextLine();

                // Check if student username is empty
                while (studentUsername.isEmpty()) {
                    System.out.println("Username cannot be empty. Please try
again.");
                    System.out.print("Create a new student username: ");
                    studentUsername = sc.nextLine();
                }
```

```
                System.out.print("Create a new student password: ");
                String studentPassword = sc.nextLine();

                // Check if student password is empty
                while (studentPassword.isEmpty()) {
                    System.out.println("Password cannot be empty. Please try
again.");
                    System.out.print("Create a new student password: ");
                    studentPassword = sc.nextLine();
                }

                if (studentSignup(studentUsername, studentPassword)) {
                    System.out.println("Student signup successful. Please proceed
with student login.");

                    // Student login
                    System.out.println("=== Student Login ===");
                    System.out.print("Enter student username: ");
                    String loginUsername = sc.nextLine();
                    System.out.print("Enter student password: ");
                    String loginPassword = sc.nextLine();

                    int studentId = studentLogin(loginUsername, loginPassword);

                    if (studentId != -1) {
                        System.out.println("Student login successful.\n");

                        // start the exam after successfull login

                        Exam exam = new Exam();

                        // Retrieve questions and options from the database
                        try (Connection connection = getConnection()) {
                            // Retrieve questions
                            String selectQuestionsQuery = "SELECT * FROM questions";
                            PreparedStatement selectQuestionsStmt =
connection.prepareStatement(selectQuestionsQuery);
                            ResultSet questionResultSet =
selectQuestionsStmt.executeQuery();
                            while (questionResultSet.next()) {
                                int questionId = questionResultSet.getInt("id");
                                String questionText =
questionResultSet.getString("text");
                                int marks = questionResultSet.getInt("marks");
                                Question question = new Question(questionId,
questionText, marks);

                                // Retrieve options for each question
                                String selectOptionsQuery = "SELECT * FROM options
WHERE question_id = ?";
                                PreparedStatement selectOptionsStmt =
connection.prepareStatement(selectOptionsQuery);
```

```
                                selectOptionsStmt.setInt(1, questionId);
                                ResultSet optionResultSet =
selectOptionsStmt.executeQuery();
                                while (optionResultSet.next()) {
                                    String optionText =
optionResultSet.getString("text");
                                    boolean isCorrect =
optionResultSet.getBoolean("is_correct");
                                    question.addOption(optionText, isCorrect);
                                }

                                exam.addQuestion(question);
                            }
                        } catch (SQLException e) {
                            e.printStackTrace();
                        }

                        // Start the exam with a duration of 5 minutes
                        exam.startExam(5);

                        // ...

                    } else {
                        System.out.println("Invalid student credentials. Exiting the
program.");
                    }
                } else {
                    System.out.println("Student signup failed. Exiting the
program.");
                }
            }

        }

        else if (userType.equalsIgnoreCase("admin")) {
            // Admin login

            System.out.println("==================================");

            System.out.println("=== Admin Login ===");

            boolean adminLoggedIn = false;
            while (!adminLoggedIn) {
                System.out.print("Enter admin username: ");
                String adminUsername = sc.nextLine();
                System.out.print("Enter admin password: ");
                String adminPassword = sc.nextLine();

                if (adminLogin(adminUsername, adminPassword)) {
                    adminLoggedIn = true;
```

```
                    Connection connection = DriverManager.getConnection(url,
username, password);
                    System.out.println("Admin login successful.\n");

                    while (true) {
                        System.out.println("====== Admin Menu =======");
                        System.out.println("1. Add Questions");
                        System.out.println("2. Display Student Marks");
                        System.out.println("3. Logout");
                        System.out.print("Enter your choice: ");

                        int choice = sc.nextInt();
                        System.out.println("==========================");
                        sc.nextLine(); // Clear the input buffer

                        switch (choice) {
                            case 1:
                                addQuestions(connection);
                                break;
                            case 2:
                                displayAllStudentMarks();
                                break;
                            case 3:
                                System.out.println("Admin logged out. Exiting the
program.");
                                System.out.println("*********************************
******");

                                return;
                            default:
                                System.out.println("Invalid choice! Please enter a
valid option.");
                        }
                    }
                } else {
                    System.out.println("Invalid admin credentials. Please try
again.");
                }
            }
        } else {
            System.out.println("Invalid input. Exiting the program.");
        }

    }

    private static Connection getConnection() throws SQLException {
        return
DriverManager.getConnection("jdbc:mysql://localhost:3306/examination", "root",
"Saswat@0602");
    }
//logic for retireve student login like user name and password from data base
    private static int studentLogin(String username, String password) {
```

```
        try (Connection connection = getConnection()) {
            String selectStudentQuery = "SELECT id FROM students WHERE username = ?
AND password = ?";
            PreparedStatement selectStudentStmt =
connection.prepareStatement(selectStudentQuery);
            selectStudentStmt.setString(1, username);
            selectStudentStmt.setString(2, password);
            ResultSet studentResultSet = selectStudentStmt.executeQuery();

            if (studentResultSet.next()) {
                return studentResultSet.getInt("id");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return -1;
    }
//logic for store student login like user name and password from data base

    private static boolean studentSignup(String username, String password) {
        try (Connection connection = getConnection()) {
            String insertStudentQuery = "INSERT INTO students (username, password)
VALUES (?, ?)";
            PreparedStatement insertStudentStmt =
connection.prepareStatement(insertStudentQuery);
            insertStudentStmt.setString(1, username);
            insertStudentStmt.setString(2, password);
            int rowsAffected = insertStudentStmt.executeUpdate();

            return rowsAffected > 0;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
    }
//logic for match admin crednetials from data base

    private static boolean adminLogin(String username, String password) {
        try (Connection connection = getConnection()) {
            String selectAdminQuery = "SELECT * FROM admins WHERE username = ? AND
password = ?";
            PreparedStatement selectAdminStmt =
connection.prepareStatement(selectAdminQuery);
            selectAdminStmt.setString(1, username);
            selectAdminStmt.setString(2, password);
            ResultSet adminResultSet = selectAdminStmt.executeQuery();

            return adminResultSet.next();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
```

```
    }

// addd question and mark to question table and option and correct option
in  optaion table data base by admin

private static void addQuestions(Connection connection) {
    try {
        Scanner sc = new Scanner(System.in);
        System.out.println("=== Add Questions ===");

        System.out.print("Enter the question text: ");
        String questionText = sc.nextLine();

        int marks;
        while (true) {
            System.out.print("Enter the marks for this question: ");
            String marksInput = sc.nextLine();
            try {
                marks = Integer.parseInt(marksInput);
                break; // If the input is a valid integer, exit the loop
            } catch (NumberFormatException e) {
                System.out.println("Invalid input. Please enter a valid number for
marks.");
            }
        }

        // Insert the question into the database
        String insertQuestionQuery = "INSERT INTO questions (text, marks) VALUES (?,
?)";
        PreparedStatement insertQuestionStmt =
connection.prepareStatement(insertQuestionQuery,
                Statement.RETURN_GENERATED_KEYS);
        insertQuestionStmt.setString(1, questionText);
        insertQuestionStmt.setInt(2, marks);
        insertQuestionStmt.executeUpdate();

        ResultSet generatedKeys = insertQuestionStmt.getGeneratedKeys();
        int questionId;
        if (generatedKeys.next()) {
            questionId = generatedKeys.getInt(1);

            // Add options for the question
            while (true) {
                System.out.print("Enter option text (or 'exit' to finish adding
options): ");
                String optionText = sc.nextLine();
                if (optionText.equalsIgnoreCase("exit")) {
                    break;
                }

                boolean isCorrect;
                while (true) {
```

```
                    System.out.print("Is this option correct? (true/false): ");
                    String isCorrectInput = sc.nextLine();
                    if (isCorrectInput.equalsIgnoreCase("true") ||
isCorrectInput.equalsIgnoreCase("false")) {
                        isCorrect = Boolean.parseBoolean(isCorrectInput);
                        break;
                    } else {
                        System.out.println("Invalid input. Please enter 'true' or
'false' for option correctness.");
                    }
                }

                // Insert the option into the database
                String insertOptionQuery = "INSERT INTO options (question_id, text,
is_correct) VALUES (?, ?, ?)";
                PreparedStatement insertOptionStmt =
connection.prepareStatement(insertOptionQuery);
                insertOptionStmt.setInt(1, questionId);
                insertOptionStmt.setString(2, optionText);
                insertOptionStmt.setBoolean(3, isCorrect);
                insertOptionStmt.executeUpdate();
            }

            System.out.println("Question added successfully!");
        } else {
            System.out.println("Failed to add the question.");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}


    private static void displayAllStudentMarks() {
        try (Connection connection = getConnection()) {
            String selectMarksQuery = "SELECT * FROM exam_results";
            PreparedStatement selectMarksStmt =
connection.prepareStatement(selectMarksQuery);
            ResultSet marksResultSet = selectMarksStmt.executeQuery();
            System.out.println("==================================");
            System.out.println("======== Student Marks ============");
            System.out.println("==================================");

            while (marksResultSet.next()) {
                int studentId = marksResultSet.getInt("student_id");
                String studentName = marksResultSet.getString("student_name");
                int obtainedMarks = marksResultSet.getInt("obtained_marks");
                int totalMarks = marksResultSet.getInt("total_marks");
                System.out.println("==================================");
                System.out.println("Student ID: " + studentId);
                System.out.println("Student Name: " + studentName);
                System.out.println("Obtained Marks: " + obtainedMarks);
```

```
                System.out.println("Total Marks: " + totalMarks);
                System.out.println("==================================");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

CHAPTER 6: RESULTS AND DISCUSSION :

6.1 Test Reports :

1. **Test Summary:** An overview of the testing activities conducted, including the types of tests performed (e.g., unit testing, integration testing, system testing, etc.), the number of test cases executed, and the pass/fail status of each test category.
2. **Test Coverage:** A detailed breakdown of the code coverage achieved during unit testing. This may include the percentage of lines of code covered, branches tested, and overall test coverage metrics.
3. **Defect Summary:** A summary of defects or issues found during testing, including their severity, priority, and status (open, closed, fixed, etc.).
4. **Test Results:** Detailed information on individual test cases, including their descriptions, input data, expected outcomes, and actual outcomes. This section highlights any discrepancies between the expected and actual results.
5. **Test Environment:** Details about the testing environment, including hardware specifications, software versions, configurations, and any dependencies used during testing.
6. **Test Execution Logs:** Logs or records of the test execution process, including timestamps, actions performed, and test case statuses.

7. **Test Pass/Fail Analysis:** An analysis of test case pass rates and failure patterns to identify common issues or areas of improvement.
8. **Traceability Matrix:** A matrix that maps each requirement to the corresponding test cases to ensure all requirements are adequately tested.
9. **Testing Progress:** A visual representation of the testing progress, such as burndown charts or graphs, to show the number of test cases executed over time.
10. **Issues and Risks:** A section dedicated to discussing any unresolved issues, potential risks, and their impact on the project.
11. **Recommendations:** Suggestions for improvements based on the testing results and lessons learned during the testing phase.

6.2 User Documentation :

Table of Contents

1. System Overview

The Examination System is designed to facilitate online exams for students and allow administrators to manage the exam questions and view student marks. The system is easy to use and ensures a secure and efficient examination process.

2. Student Features

a. Student Login

If you are a student, follow these steps to log in to the system:

1. Enter "2" when prompted for your choice.
2. Enter your username and password when prompted.

b. Starting an Exam

Once logged in as a student, you can start an exam by following these steps:

1. Enter "1" when prompted for your choice in the student menu.
2. The exam will start, and you will be presented with each question one by one.
3. Read the question and its options carefully.
4. Enter the number corresponding to your answer (1, 2, 3, etc.) or "0" to skip the question.
5. Once the exam duration is over or you finish all the questions, the system will automatically submit your exam.

3. Administrator Features

a. Administrator Login

If you are an administrator, follow these steps to log in to the system:

1. Enter "1" when prompted for your choice.
2. Enter your admin username and password when prompted.

b. Adding Questions

Once logged in as an administrator, you can add questions to the exam by following these steps:

1. Enter "1" when prompted for your choice in the admin menu.
2. Enter the question text and marks for the new question.
3. Add options for the question one by one, indicating which options are correct.
4. Repeat the process to add more questions if needed.

c. Displaying Student Marks

As an administrator, you can view the marks obtained by students in the exam:

1. Enter "2" when prompted for your choice in the admin menu.
2. The system will display the student names and their corresponding obtained marks.

4. Exiting the System

To exit the system, follow these steps:

1. Enter "3" when prompted for your choice in the main menu.
2. The system will display a farewell message, and you will be logged out.

CHAPTER 7: CONCLUSIONS :

7.1 Conclusion :

- In conclusion, the Examination System is a basic implementation of an online exam management system that allows students to take exams and administrators to manage questions and view student marks. The system provides a user-friendly interface for conducting exams, storing student data, and displaying exam results. However, it also has several limitations and areas for improvement.
- The project successfully achieves its primary objective of facilitating online exams and automating the grading process. It demonstrates the use of Java programming, JDBC for database connectivity, and multi-threading for timer functionality. The code structure is organized into separate classes for questions, options, exams, and database operations, enhancing maintainability.
- Despite its achievements, the system has limitations, such as limited question types, security concerns, scalability issues, and the absence of exam proctoring. It lacks features like question randomization, exam review, and comprehensive result analysis, which are essential for a robust examination system.
- To make the system more efficient and secure, it requires enhancements such as advanced security measures, support for various question types, scalability improvements, and exam proctoring features to prevent cheating. Additionally, the user interface and reporting capabilities could be enhanced to provide better insights into student performance.
- In the future, the project could be extended to include features like user management, a broader range of question types, real-time analytics, and

support for multiple exams with varied durations. Implementing a more robust database system and enhancing data backup and recovery mechanisms would ensure data integrity and availability.

7.2 Limitations of the System :

1. **Limited Question Types**: The system currently supports only multiple-choice questions. It does not handle other question types such as true/false, short answer, or essay-type questions.
2. **Security Concerns**: The system lacks advanced security features, making it susceptible to potential security breaches or cheating. It does not employ techniques to prevent unauthorized access or protect against various forms of cheating during exams.
3. **Scalability**: The system's design may not be optimized for handling a large number of concurrent users or a significant volume of exams. As a result, performance issues may arise in scenarios with high traffic.
4. **Limited User Management**: While the system distinguishes between students and administrators, it lacks extensive user management capabilities. For instance, there is no option for students to create accounts, and administrators have limited control over user roles.
5. **No Question Randomization**: The system does not randomize the order of questions or answer options in the exams. This omission could lead to a bias in the exam process, as students taking the same test might see questions in the same sequence.

6. **Absence of Exam Proctoring**: The system does not incorporate any proctoring features, such as live monitoring or facial recognition, to ensure the integrity of online exams. This could allow students to collaborate or refer to external materials during the exam.
7. **Lack of Exam Review**: After completing an exam, students do not have the option to review their answers and the correct solutions. This feature could be helpful for learning and understanding mistakes.
8. **Dependency on Local Database**: The system relies on a local database for storing exam data. This setup might not be suitable for large-scale deployments or for situations where data backup and accessibility are critical.

7.3 Future Scope of the Project :

1. **Admin Authentication and Security Enhancement:**
   - Implement a secure authentication mechanism for admin login, such as using hashed passwords and salting to store admin credentials securely in the database.
   - Add roles and permissions to limit access to specific admin features based on their roles (e.g., super admin, regular admin).
2. **Student Registration and Authentication:**
   - Allow students to register and create accounts with unique usernames and passwords.
   - Implement a secure authentication mechanism for student login.
3. **Randomizing Question Order:**

   Randomize the order of questions presented to students in the exam to prevent cheating and make each exam unique.

4. **Enhanced Question Types:**

   Add support for various question types, such as multiple-choice with multiple correct answers, true/false, fill in the blanks, etc.

5. **Question Categories and Tags:**

   Allow admins to categorize and tag questions to better organize them and enable students to filter questions based on categories during exams.

6. **Feedback Mechanism:**

   Implement a feedback mechanism for students to provide feedback on the exam difficulty, question quality, and overall user experience.

7. **Time Extensions and Grace Periods:**

   Allow admins to grant time extensions or define grace periods for specific students based on their needs or circumstances.

8. **Graceful Error Handling:**

   Implement proper error handling throughout the application to provide meaningful error messages and prevent crashes.

9. **Data Validation and Sanitization:**

   Implement data validation and sanitization to prevent common security vulnerabilities like SQL injection and cross-site scripting (XSS) attacks.

10. **User-Friendly UI Enhancements:**

    Improve the user interface (UI) to make it more intuitive and visually appealing for both admins and students.

11. **Database Backup and Restore:**
    Implement a backup and restore mechanism for the database to safeguard data in case of any data loss or corruption.
12. **Performance Optimization:**
    Analyze and optimize the code to improve performance, especially for large-scale usage.
13. **Multilingual Support:**
    Add support for multiple languages to cater to users from different regions.
14. **Accessibility Improvements:**
    Ensure the application is accessible to users with disabilities by following accessibility guidelines.
15. **Logging and Monitoring:**
    Implement logging and monitoring to track application usage, identify potential issues, and monitor system health.
16. **Automated Testing:**
    Set up automated testing using tools like JUnit and integration testing to ensure the code remains reliable during future updates and changes.
17. **Version Control:**
    Use version control systems like Git to track changes and collaborate efficiently with team members.
18. **Documentation:**
    Maintain comprehensive documentation for the codebase, API endpoints, and other relevant information to help future developers understand and work on the project.