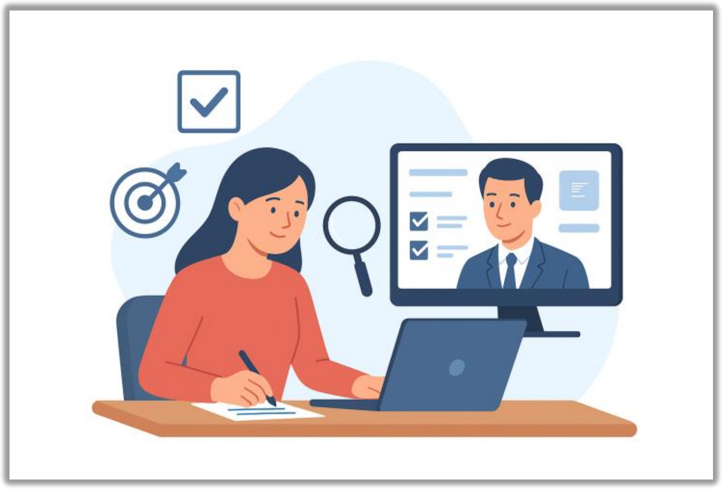
Invertis University, Bareilly [UP]

(2025–26)

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

ONLINE EXAMINATION & SMART PROCTORING SYSTEM



JAVA PROGRAMMING LAB (BCS553)

Submitted To:	**Submitted By:**
Mr. Nishant Thakur	Team Leader : Rohit Agarwal
Technical Trainer	Total Members: 10

CERTIFICATE

This is to certify that the project report titled **“ONLINE EXAMINATION & SMART PROCTORING SYSTEM”** is a bona fide record of the original work carried out by **Rohit Agarwal (TEAM LEADER)**, and the students of My Team, Bachelor of Technology (Computer Science and Engineering).

This project has been completed in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering, during the academic year 2025–2026.

The work presented in this project contributes to the development of a secure, automated, and intelligent online **desktop examination platform**. The system integrates **Java Swing** for the GUI, **JDBC with MySQL** for the database, and **automated proctoring** to ensure fairness and authenticity in remote assessments.

It provides a complete end-to-end solution—from exam creation and management to smart monitoring and result analysis—addressing the modern needs of digital education systems. The project validates a complete workflow, encompassing user authentication, management of exams and questions, automated evaluation, and proctoring using the **Webcam-Capture library** and **window focus detection** techniques.

Mr. Nishant Kumar Thakur
(Project Guide)
Department of Computer Science and Engineering
Invertis University, Bareilly

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to our **Faculty Advisor, Mr. Nishant Kumar Thakur**, for his continuous guidance, constant encouragement, and invaluable insights throughout the development of our project **“Online Examination & Smart Proctoring System.”**
His mentorship, technical expertise, and constructive feedback have been instrumental in enhancing the quality and success of this work.

We are also deeply thankful to the **Department of Computer Science and Engineering, Invertis University**, for providing us with the necessary infrastructure, academic support, and a conducive learning environment to carry out this project effectively.

We would also like to extend our appreciation to our **friends and classmates** for their collaboration, helpful discussions, and motivation, which inspired us to continuously improve our system and documentation.

Finally, we express our deepest gratitude to our **families** for their constant encouragement, patience, and emotional support throughout the duration of this project. Their belief in us has been our strongest motivation to complete this work successfully.

ABSTRACT

The Online Examination & Smart Proctoring System is a comprehensive **desktop application** (Java Swing) developed to digitalize and secure the examination process. It automates every phase of conducting exams—starting from exam creation and question bank management to evaluation and result generation—while ensuring integrity through automated monitoring.

The system enables administrators to manage exams, questions, and results efficiently, and allows students to take tests remotely under controlled and monitored conditions. The Smart Proctoring module utilizes the **Webcam-Capture library** to display a live feed and detects suspicious activities such as **window focus loss (tab switching)**, thereby upholding the fairness of the examination.

Developed using **Core Java** with the **Swing library** for the GUI, **JDBC** for database connectivity, and **MySQL** for data management, the system provides a robust 2-Tier (Client-Server) architecture. This project aims to minimize human intervention, reduce administrative workload, and provide a transparent, efficient, and intelligent platform for conducting online examinations.

TEAM MEMBERS

S. No	Student Name	Student ID
1	Rohit Agarwal (Leader)	BCS2023113
2	Trisha Bhardwaj	BCS2023148
3	Vanshika Uppal	BCS2023143
4	Jahnvi Malik	BCS2023139
5	Mohd. Saqlain Hussain	BCS2023126
6	Shubham Raghav	BCS2023125
7	Yash Sahai	BCS2023135
8	Somya Kumari	BCS2023114
9	Ashraf Ansari	BCS2023137
10	Harshit Kumar Gautam	BCS2023141

TABLE OF CONTENT

CHAPTER 1: INTRODUCTION

1.1. Project Background and Motivation

Traditional pen-and-paper examinations are a cornerstone of academia, but they are burdened with significant logistical challenges. These include high costs associated with printing and paper, extensive human resources for invigilation, and a time-consuming manual evaluation process. Furthermore, in an era of remote learning, conducting secure assessments from different locations is a major hurdle.

This project is motivated by the need for a modern, automated, and cost-effective desktop solution. We leverage the power of Java and its robust Swing library to create a system that can not only automate the entire exam lifecycle but also ensure its integrity.

1.2. Problem Statement

The problem is to design and implement an end-to-end "smart" desktop system that can automatically manage and conduct secure online examinations. This system must be accurate, reliable, and accessible to both administrators and students. It must handle exam creation, secure user authentication, timed test-taking, random question shuffling, instant auto-grading, and a reliable proctoring mechanism to prevent cheating.

1.3. Objectives

Based on the problem statement, our team defined the following objectives:

1. **To design and develop** a secure, 2-tier desktop application using Java Swing for the frontend and a MySQL database for the backend.
2. **To implement** a robust role-based access control system (Admin, Student) with secure password hashing (SHA-256).
3. **To create** a full CRUD (Create, Read, Update, Delete) module for administrators to manage exams and their corresponding question banks.

4. **To build** a time-bound exam engine for students using **Multithreading** (javax.swing.Timer) that supports random question fetching and auto-submission when the timer ends.
5. **To integrate** a "Smart Proctoring" module using the **Webcam-Capture** library and **WindowFocusListener**.
6. **To ensure** that any student switching tabs (losing window focus) is automatically submitted for cheating with a score of 0.
7. **To provide** instant and automated result generation for students upon submission and a consolidated result view for admins.

1.4. Project Scope

The scope of this project is to build a fully functional Proof-of-Concept (POC) end-to-end system. It covers the complete application lifecycle:

1. **Database:** Designing a normalized MySQL schema.
2. **Application:** Building the Java Swing application with all modules (Admin, Student, Exam Engine).
3. **Security:** Implementing password hashing and real-time proctoring logic.
4. **Deployment:** The system runs as a standalone Java application (.jar) connecting to a central MySQL server (currently localhost).

CHAPTER 2: LITERATURE REVIEW & METHODOLOGY

2.1. Traditional Examination Methods

Traditional examination methods, reliant on pen and paper, are well-established but suffer from major inefficiencies.It requires physical presence, large-scale printing, manual distribution of papers, and is highly susceptible to human error during grading. This method is not scalable, environmentally unfriendly, and lacks any form of modern security or data analysis.

2.2. Modern Web-Based Examination Systems

Modern solutions often use web-based (3-Tier) architectures (like JSP, Servlets, PHP, or Node.js). However, they introduce new security challenges, such as browser-based cheating, network instability, and complex server-side setup (like Apache Tomcat servers).

2.3. Our Chosen Methodology (Java Swing & 2-Tier Architecture)

For this project, we selected a **2-Tier (Client-Server) Architecture** using **Java Swing**.

- **Why Swing?** Swing is a powerful and mature GUI toolkit that is built directly into Java (JDK). This eliminates the need for external web servers, browsers, or complex setups. It provides us with direct, low-level control over the user's environment, which is *essential* for effective proctoring.
- **Why 2-Tier?** Our architecture is simple and robust. The **Client** (our Java Swing application) contains all the presentation (GUI) and business logic. It communicates directly with the **Server** (the MySQL database) via a JDBC connection.
- **Proctoring Advantage:** By using a desktop application, our proctoring is more reliable. We can directly access the webcam and, most importantly, monitor the **Window Focus** (WindowFocusListener).

CHAPTER 3: SYSTEM & DATABASE DESIGN

3.1. System Architecture

The system is built on a 2-Tier Client-Server model.

1. **Tier 1 (Client):** The Java Swing application (.jar file). This is the "smart client" that runs on the user's machine. It handles all UI rendering, event handling (button clicks, focus loss), and business logic.
2. **Tier 2 (Server):** The MySQL Database Server. This server's only job is to store, retrieve, and update data as requested by the client via JDBC.

All logic, including password validation, timer countdowns, and score calculation, happens on the client side, making the application fast and responsive.

3.2. Database Schema Design (ER Diagram)

We designed a relational database schema with 4 primary tables:

- **users**: Stores user information, including user_id (Primary Key), username, password_hash, and role ('admin' or 'student').
- **exams**: Stores information about each exam. The **exam_code (Primary Key)** is a manual text-based code (e.g., "MATH101") set by the admin.
- **questions**: Stores all questions. Each question is linked to an exam via exam_code (Foreign Key).
- **results**: Stores student scores. It links users (via user_id) and exams (via exam_code) and records the final score.

[Insert ER Diagram image here]

3.3. Application Workflow (Use Case Diagram)

The system has two main actors with distinct workflows:

- **Admin:**
    1. Logs in.
    2. Navigates to "Manage Exams" to create a new exam (e.g., "MATH101").
    3. Navigates to "Manage Questions" to add questions to "MATH101".
    4. Navigates to "View Results" to see student scores.
- **Student:**
    1. (First time) Clicks "Register" to create an account.
    2. Logs in.
    3. Sees a list of available exams ("MATH101").
    4. Selects the exam and a webcam.
    5. Clicks "Start Exam".
    6. The ExamWindow and ProctorWindow open.
    7. **Scenario A (Normal):** Student finishes the exam, clicks "Submit".
    8. **Scenario B (Time Up):** Timer reaches 00:00, exam auto-submits.
    9. **Scenario C (Cheating):** Student clicks outside the window, exam auto-submits with score 0.
    10. Student sees their result popup and is returned to the dashboard.

[Insert Use Case Diagram image here]

3.4. Security Implementation (Password Hashing)

Passwords are **never** stored in plain text. We use a one-way cryptographic hash function (SHA-256) for security.

1. **Registration:**
    - String plainPassword = "pass123";
    - String hashedPassword = SecurityUtil.hashPassword(plainPassword);
    - The hashedPassword (e.g., "6b27be...") is saved in the database.
2. **Login:**
    - User enters "pass123".
    - The code creates a hash of this input: String inputHash = SecurityUtil.hashPassword("pass123");
    - The code compares the inputHash with the storedHash from the database.
    - if (inputHash.equals(storedHash)) { // Login successful }

CHAPTER 4: MODULE IMPLEMENTATION

4.1. Module 1: Admin Panel (CRUD Operations)

This module consists of three main screens:

- **AdminDashboard.java**: The main menu with three buttons (Manage Exams, Manage Questions, View Results).
- **ManageExams.java**: Provides full CRUD functionality for exams.
    - **Create:** A form to add a new exam with a manual exam_code, exam_name, and duration.
    - **Read:** Uses a JTable to display all existing exams from the database.
    - **Delete:** A button to delete a selected exam. This action uses a database transaction to also delete all related data from questions and results to maintain integrity.
- **ViewResults.java**: Uses a JTable and a complex **SQL JOIN** query (SELECT ... FROM results r JOIN users u ... JOIN exams e ...) to display a user-friendly table of results, showing student names and exam names instead of just IDs.

4.2. Module 2: Student Panel (Registration & Exam Selection)

- **RegistrationScreen.java**: A simple form that collects a new student's name, username, and password. It calls SecurityUtil.hashPassword() before sending the data to the users table.
- **StudentDashboard.java**:
    - Displays a "Welcome" message with the student's name.
    - Fetches and displays all available exams from the exams table in a JTable.
    - Features a "Start Exam" button that initiates the proctoring setup.

4.3. Module 3: Exam Engine (Timer & Auto-Submit)

This is the core of the student experience, managed by ExamWindow.java.

- **Random Questions:** On load, it fetches all questions for the selected exam_code using ORDER BY RAND() in the SQL query. This ensures every student gets a different question order.
- **Timer (Multithreading):** We use a javax.swing.Timer (which is thread-safe for Swing).
    - swingTimer = new Timer(1000, ...): This triggers every 1000ms (1 second).
    - Inside its actionPerformed method, we decrement timeRemaining.
    - if (timeRemaining <= 0) { submitExam(false); }: When the time runs out, it calls the auto-submit function.
- **Auto-Grading:** When submitExam() runs, it loops through the questions list, compares getSelectedAnswer() with getCorrectAnswer(), and calculates the final score, saving it to the results table.

4.4. Module 4: Smart Proctoring (Webcam & Focus Detection)

This module runs alongside the Exam Engine.

- **Webcam Selection:** The StudentDashboard uses Webcam.getWebcams() to show the user a popup, allowing them to select their webcam index (0, 1, 2).
- **ProctorWindow.java**: This class takes the selected webcamIndex and uses WebcamPanel to display the live feed in a small, non-focusable (setFocusableWindowState(false)) window.
- **ExamWindow.java (Focus Detection):**
    - ExamWindow implements WindowFocusListener.

- In the constructor, frame.addWindowFocusListener(this); is registered.
- public void windowLostFocus(WindowEvent e): This function is triggered only when the user clicks outside the exam window.
- Inside this method, we call submitExam(true);, which submits the exam with a score of 0 and displays a cheating message.
- In submitExam(), frame.removeWindowFocusListener(this); is called to prevent a false cheating detection when the result popup appears.

CHAPTER 5: SYSTEM DEPLOYMENT & KEY TECHNOLOGIES

(This chapter explains the points above in greater detail)

5.1. Core Technology Stack

- **Java (JDK 17):** Core programming language.
- **Java Swing:** GUI framework (JFrame, JPanel, JButton, JTable, etc.).
- **MySQL:** Relational database for data storage.
- **Webcam-Capture:** Third-party Java library (JAR) to access webcam hardware.

5.2. Database Connectivity (JDBC)

All database communication happens via JDBC (Java Database Connectivity).

- **DatabaseUtil.java**: A helper class was created that provides a static getConnection() method.
- **try-with-resources**: We used the try-with-resources block (try (Connection conn = ...)) for PreparedStatement and Connection objects. This automatically closes the connection and prevents memory leaks.

5.3. Multithreading for Timer

javax.swing.Timer was used for the timer. It is better than java.util.Timer because it runs on Swing's **Event Dispatch Thread (EDT)**. This means we don't need to use SwingUtilities.invokeLater() to update the timerLabel.setText(...), which keeps the code cleaner.

5.4. Event Handling (ActionListeners & WindowFocusListener)

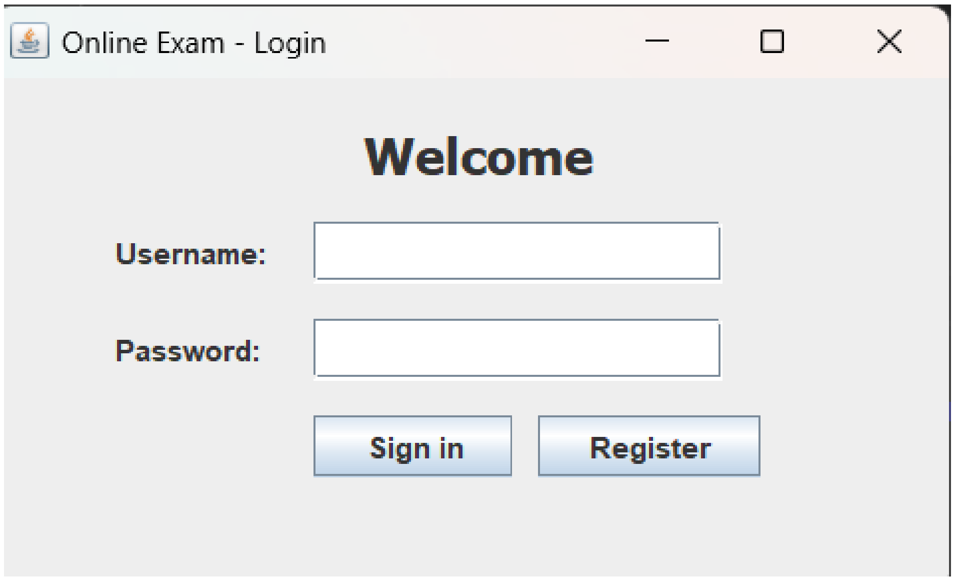
The project's entire flow is event-driven.

- **ActionListener**: An ActionListener is attached to every button (Login, Submit, Add Exam). It "listens" for the "click" event and executes the corresponding logic via a Lambda expression (e -> { ... }).
- **WindowFocusListener**: This is attached to the ExamWindow's frame and "listens" for the "focus lost" event, which is our primary cheating detection mechanism.

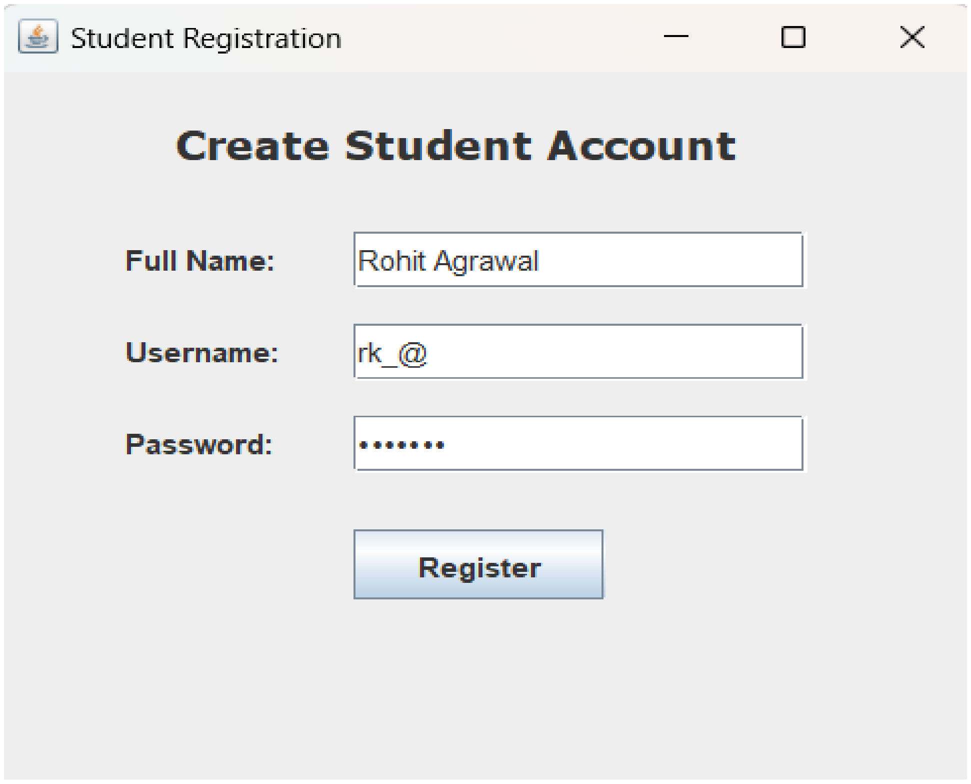
CHAPTER 6: RESULTS & PERFORMANCE ANALYSIS

6.1. System Output

- **Login & Registration:**
  - **Description:** Secure login and registration module. Passwords are hashed using SHA-256.

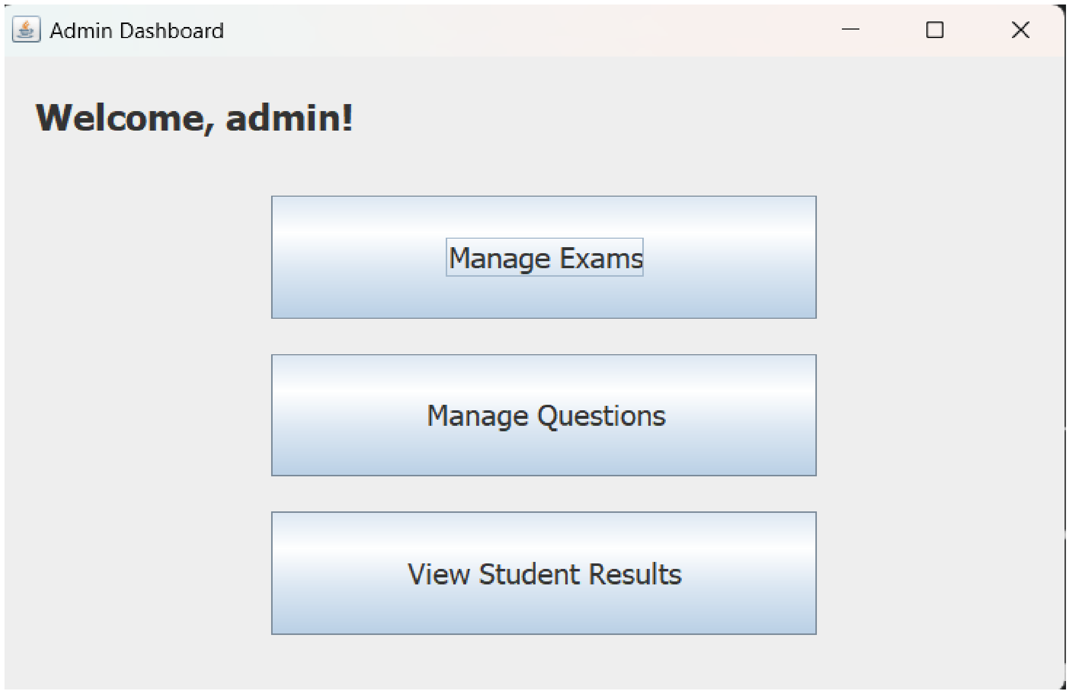


- **Admin Dashboard & Modules:**

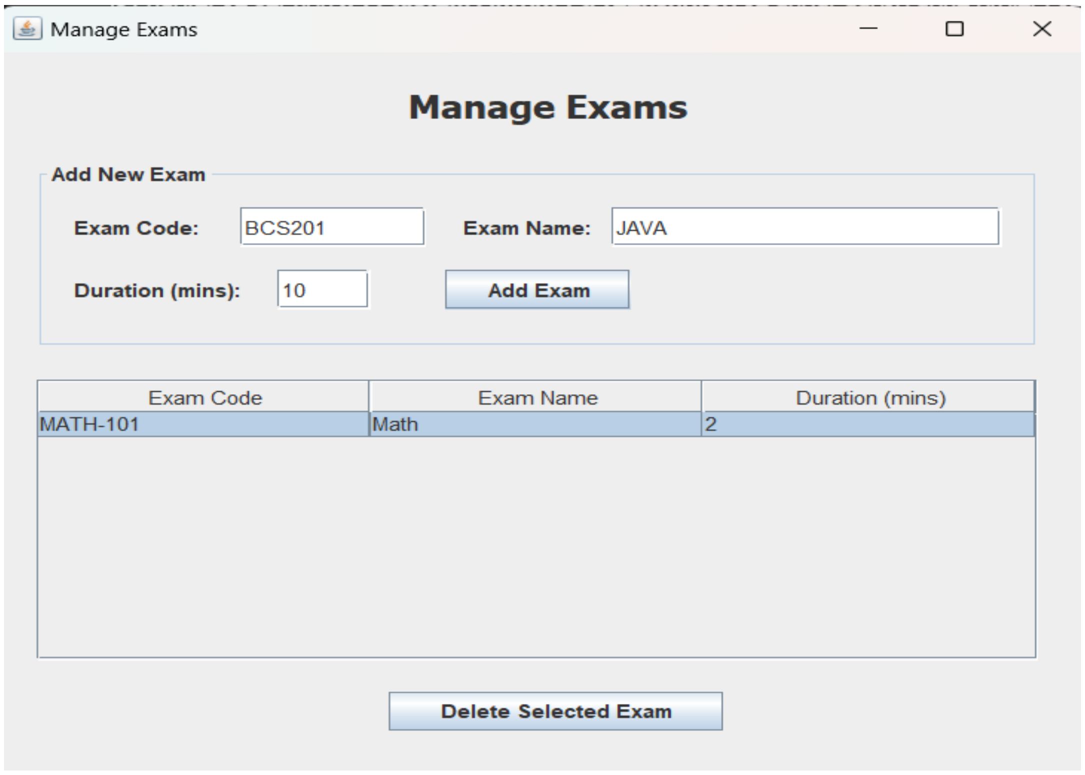


Full Name: Rohit Agrawal
Username: rk_@
Password: •••••••

- **Description:** The complete Admin CRUD workflow, including setting manual exam codes and viewing results.



Welcome, admin!



Manage Exams

Exam Code: BCS201

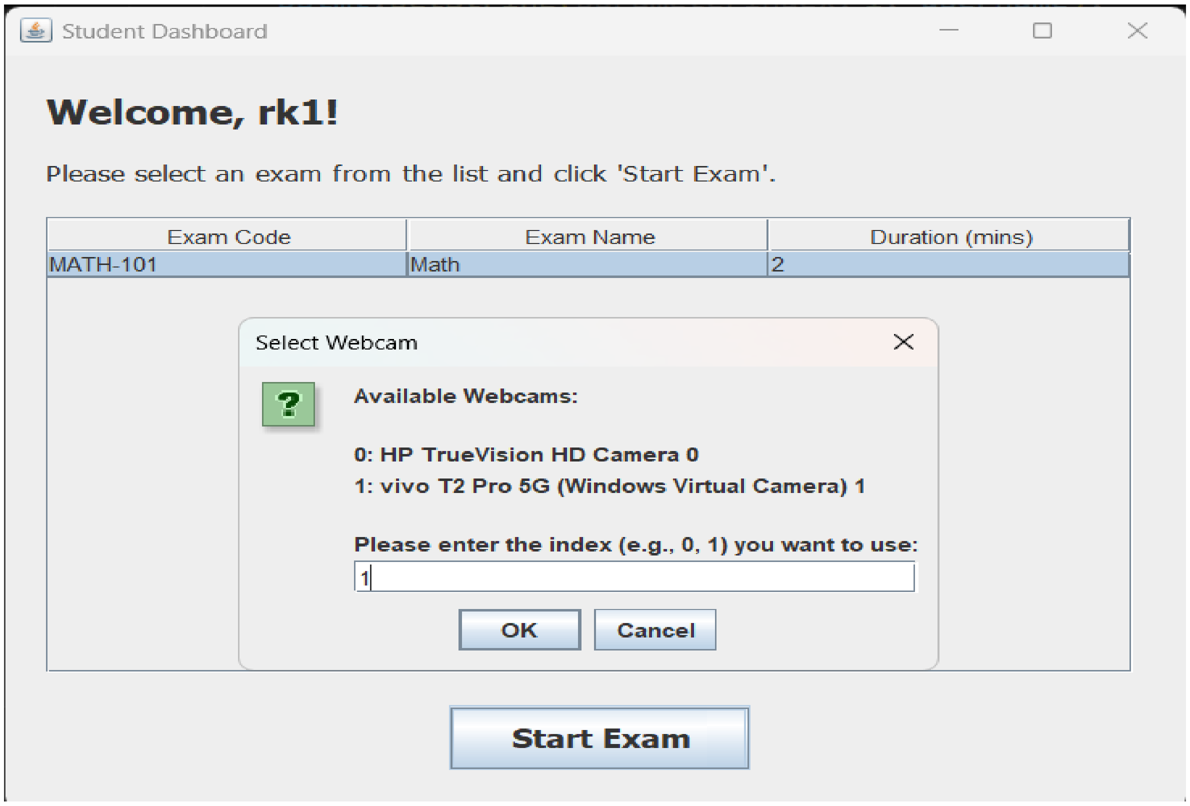
Exam Name: JAVA

Duration (mins): 10

Exam Code	Exam Name	Duration (mins)
MATH-101	Math	2

- **Student Exam Flow:**

- **Description:** The student's dashboard where they select an exam and choose a webcam for proctoring.



Welcome, rk1!

Please select an exam from the list and click 'Start Exam'.

Exam Code	Exam Name	Duration (mins)
MATH-101	Math	2

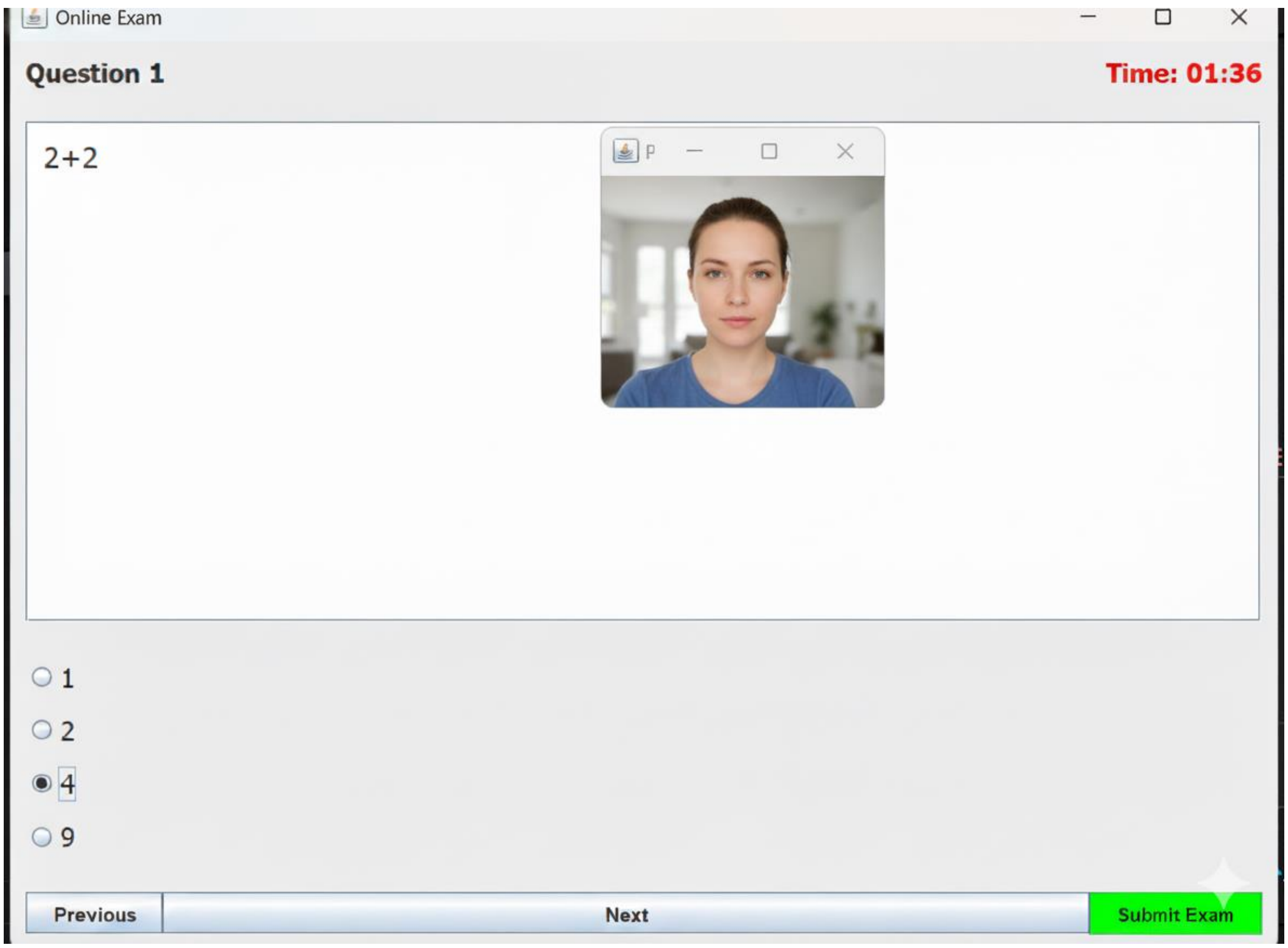
Available Webcams:

0: HP TrueVision HD Camera 0
1: vivo T2 Pro 5G (Windows Virtual Camera) 1

Please enter the index (e.g., 0, 1) you want to use:
1

- **Proctoring in Action:**
  - **Description:** The live exam environment. The timer is running, the question is displayed, and webcam monitoring is active.

- **Cheating Detection & Results:**
  - **Description:** The two final outcomes: a normal submission and a cheating-detected auto-submission.



Question 1

Time: 01:36

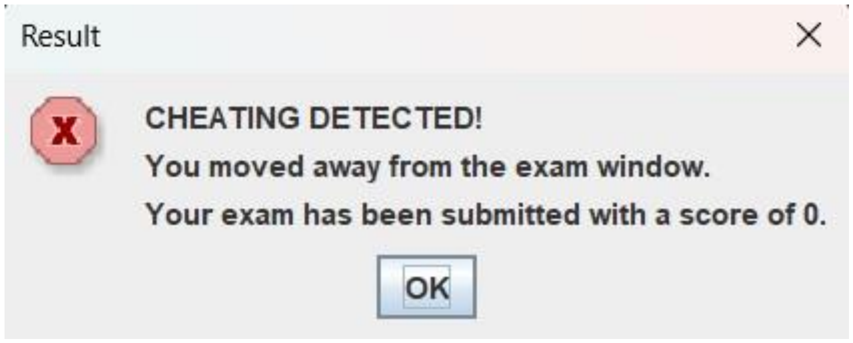
2+2

( ) 1

( ) 2

(●) 4

( ) 9



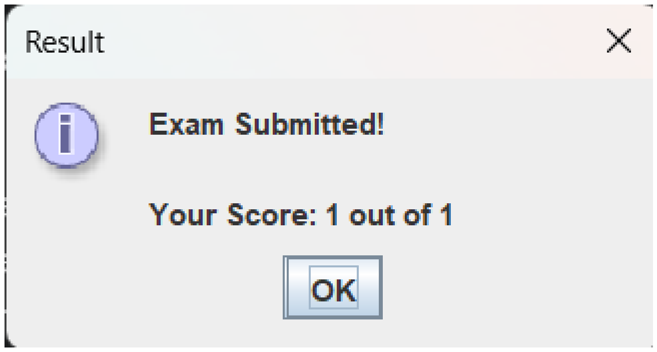
Result

CHEATING DETECTED!
You moved away from the exam window.
Your exam has been submitted with a score of 0.

6.2. Performance & Reliability

- **Performance:** The application is lightweight because Swing uses native components. Database operations (INSERT, SELECT) are optimized using PreparedStatement, which is fast and prevents SQL Injection.

- **Reliability:** The use of try-with-resources makes database connections reliable. javax.swing.Timer makes multithreading safe for the UI. The cheating detection logic triggers instantly on windowLostFocus, making it very reliable.



Exam Submitted!

Your Score: 1 out of 1

CHAPTER 7: CHALLENGES & FUTURE SCOPE

7.1. Challenges Faced

- **Password Hashing Bug:** Initially, there was a bug in the password hashing logic (an extra space was being added), which caused the 'admin' login to fail. This was solved by creating a FixAdminPassword.java utility.
- **Database Constraints:** Changing the exam_id from an INT (Auto-Increment) to a VARCHAR (Manual Code) was a major challenge. It required modifying the entire database schema (3 tables) and approximately 5 Java files.
- **Proctoring Logic:**
  - **False Cheating Alert (On Start):** The webcam window (ProctorWindow) would take focus on launch, causing a false alert. This was fixed with window.setFocusableWindowState(false);.
  - **False Cheating Alert (On Submit):** The focus was lost when the result popup appeared. This was fixed by adding frame.removeWindowFocusListener(this); inside the submitExam() function.

7.2. Future Scope and Enhancements

- **Advanced AI Proctoring:** Replace the webcam-capture library with **OpenCV** to perform real-time face detection. This would allow us to detect "Multiple Faces", "No Face", or "Mobile Phone" usage.
- **Live Admin Monitoring:** Currently, the webcam feed is only visible to the student. In the future, **Socket Programming** could be used to stream this live feed to the AdminDashboard.
- **Cloud Database:** Change the localhost URL in DatabaseUtil.java to an Amazon RDS or Azure SQL Database URL, so the application can be used from anywhere.

CHAPTER 8: CONCLUSION

The Online Examination & Smart Proctoring System successfully implements a reliable and secure digital examination process as a desktop application. It eliminates manual examination limitations while ensuring integrity and automation.

Using **Java Swing**, **JDBC**, **MySQL**, and **automated proctoring** (webcam feed and window focus detection), this project provides a strong foundation for modern, secure assessment systems. The project fulfills all core objectives: secure role-based login, automated exam management, a time-bound exam engine, and effective cheating detection.

This project validates that a robust, secure, and intelligent examination system can be built without the complexities of web servers, providing a strong alternative for institutions needing a reliable internal testing platform.

CHAPTER 9: REFERENCES

1. **Oracle Java Documentation – The Swing Tutorial** (https://docs.oracle.com/javase/tutorial/uiswing/)
2. **Oracle Java Documentation – JDBC API Reference** (httpsD://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/)
3. **MySQL Developer Guide** (https://dev.mysql.com/doc/)
4. **Webcam-Capture Library (Sarxos)** (https://github.com/sarxos/webcam-capture)
5. **Baeldung – "Java SHA-256 Hashing"** (https://www.baeldung.com/sha-256-hashing-java)
6. **Java AWT WindowFocusListener Documentation** (https://docs.oracle.com/javase/7/docs/api/java/awt/event/WindowFocusListener.html)
7. **Jenkov, Jakob. "Java Multithreading - javax.swing.Timer"** (http://tutorials.jenkov.com/java-multithreading/javax-swing-timer.html)