Online Course Registration System

A MINI-PROJECT REPORT

Submitted by

NIVEDHITHA K 240701371

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project **"ONLINE TUITION MANAGEMENT SYSTEM "** is the Bonafide work of **"NIVEDHITHA K"** who carried out the project work under my supervision.

SIGNATURE

Mrs Deepa

ASSISTANT PROFESSOR

Dept. of Computer Science and Eng,

Rajalakshmi Engineering College

Chennai

This mini project report is submitted for the viva voce examination to be held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

This project introduces the **Online Tuition Management System**, a desktop application specifically engineered for the administrative oversight of a tuition center. Utilizing **Java Swing** for a responsive user interface, the system achieves robust data persistence and integrity through **Java Database Connectivity (JDBC)** integration with a **MySQL database**. The application manages core tuition entities: Students, Instructors, and Courses (which represent tuition classes/subjects). A key feature is the financial management component, which includes dedicated **Enrollment** and **Payment** tracking. The architecture employs a critical **MySQL trigger** to automatically update a student's enrollment status to 'Paid' immediately upon successful payment insertion, guaranteeing real-time financial synchronization and administrative efficiency. The system provides a reliable, scalable foundation for managing student lifecycles and fee collection within a modern tuition environment.

.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable chairman **MR. S. MEGANATHAN** and the chairperson DR. **M. THANGAM MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by our Head of The Department **Dr. E.M. MALATHY** and our Deputy Head of The Department **Dr. J. MANORANJINI for** being ever supporting force during our project work

We also extend our sincere and hearty thanks to our internal guide **Mrs.Sathiyavathi S,** for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

Nivedhitha K

2. Nithyashree K

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ABSTRACT	**3**
1	**INTRODUCTION**	**7**
1.1	INTRODUCTION	
1.2	SCOPE OF THE WORK	
1.3	PROBLEM STATEMENT	
1.4	AIM AND OBJECTIVES OF THE PROJECT	
2	**SYSTEM SPECIFICATIONS**	**9**
2.1	HARDWARE SPECIFICATIONS	
2.2	SOFTWARE SPECIFICATIONS	
3	**MODULE DESCRIPTION**	**10**
4	**CODING**	**12**
5	**SCREENSHOTS**	**15**
6	**CONCLUSION AND FUTUR ENHANCEMENT**	**19**
7	**REFERENCES**	**20**

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The **Online Tuition Management System (OTMS)** is a desktop application designed to streamline the administrative, enrollment, and financial processes within a modern tuition or coaching center. Developed using **Java Swing** for the graphical user interface (GUI) and **Java Database Connectivity (JDBC)** for data persistence, the system provides a centralized platform for managing core academic and operational entities. The primary goal of OTMS is to transition from manual, error-prone record-keeping to an efficient, automated digital system, ensuring accuracy in student records, faculty management, class allocation, and, critically, fee collection and tracking.

1.2 SCOPE OF THE WORK

The scope of the Online Tuition Management System is strictly defined by the functionalities and architectural components implemented in this development phase.

Module	Implemented Functionality	Status
Data Creation (CRUD: C)	**Registration Forms** for **Students** and **Instructors**, enabling input validation (e.g., ensuring required fields are non-empty) and database insertion.	**Complete**
Data Retrieval (CRUD: R)	Dynamic **List Views** (JTable) for Students and Instructors, which refresh data from the MySQL backend via the refreshData() method upon request.	**Complete**
Core Entities	Database and GUI support for managing **Student**, **Instructor**, and **Course** records.	**Complete**

Module	Implemented Functionality	Status
Enrollment	Linking a Student to a Course (Enrollment table) and setting the initial payment status to **'Pending'**.	**Partial**
Payment Automation	Implementation of the **AutoUpdateEnrollmentStatus MySQL Trigger** to automatically change the Enrollment status from 'Pending' to **'Paid'** when a payment record is inserted. This completes the core financial workflow.	**Complete**

1.3 PROBLEM STATEMENT

Here is the **Problem Statement** for your **Online Tuition Management System** project, focusing on the pain points that justify the development of your Java Swing and MySQL solution.

Background

Traditional methods for managing tuition centers, such as relying on spreadsheets, paper ledgers, or fragmented software tools, lead to significant operational inefficiencies and data synchronization challenges. While the core business is academic instruction, the administrative overhead associated with student intake, class management, and fee tracking often consumes excessive staff time.

The Problem

The current administrative processes suffer from the following critical issues:

1. **Inefficient Record Management:** The lack of a centralized digital system makes the creation, storage, and retrieval of **Student** and **Instructor** records cumbersome, increasing the potential for data entry errors, such as duplicate entries or inconsistent contact information.

2. **Financial Discrepancy and Lack of Transparency:** Manual tracking of tuition fees creates delays and errors in identifying a student's payment status. There is a lack of **real-time synchronization** between payment receipt and a student's official enrollment status, making it difficult for administrators to quickly assess who has paid their fees.

3. **Absence of Automation:** Administrative staff must manually update enrollment status after recording a payment, a redundant step that is prone to human error and slows down the enrollment process.

4. **Security Vulnerability (Current State):** The current application model lacks a **User Authentication** mechanism (Login Page) and provides **direct administrative access** to all data and features upon launch. This poses a significant security risk, as there is no system for **Role-Based Access Control (RBAC)** to differentiate privileges between administrators, faculty, and students.

1.3 AIM OF THE PROJECT

The primary aim of the Online Tuition Management System (OTMS) is to develop a stable, centralized, and semi-automated desktop application using Java Swing for the user interface and MySQL JDBC for data persistence, specifically to streamline the administrative and financial lifecycle of a tuition center.The core focus is on achieving data integrity and operational efficiency by centralizing student, instructor, and course records, and utilizing database-level automation (MySQL Trigger) to ensure the instantaneous and accurate synchronization of enrollment status upon fee payment

1.4

.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Component	Detailed Specification
Processor	2.0 GHz Dual-Core or higher (Quad-Core recommended for the server host)
RAM	4 GB DDR3/DDR4 Minimum (8 GB recommended for stable performance)
Storage	100 MB free disk space for the application (SSD preferred for faster database access)
Network	Stable Internet or LAN connection (Essential for JDBC connectivity to the MySQL server)

2.2 SOFTWARE SPECIFICATIONS (Updated for MySQL)

COMPONENT	DETAILED SPECIFICATION	RATIONALE
PROGRAMMING LANGUAGE	JAVA SE DEVELOPMENT KIT (JDK) 8 OR HIGHER	THE CORE LANGUAGE REQUIRED TO COMPILE AND EXECUTE THE APPLICATION LOGIC AND GUI.
GUI FRAMEWORK	JAVA SWING	THE PRIMARY LIBRARY USED FOR CONSTRUCTING THE DESKTOP

Component	Detailed Specification	Rationale
		application's user interface, including forms and tables.
Database Server	MySQL Server (Version 8.0 or newer)	The centralized relational database management system used for persistent data storage and executing server-side logic (Triggers).
Database Driver	MySQL Connector/J (JDBC Driver)	The mandatory Java library that enables the application to connect to and communicate with the MySQL server.
Operating System	Windows 10/11, macOS, or Linux Distribution	Required to host the Java Runtime Environment (JRE) and execute the compiled application.
Development IDE	IntelliJ IDEA, Eclipse, or NetBeans	An Integrated Development Environment is necessary for

COMPONENT	DETAILED SPECIFICATION	RATIONALE
		EFFICIENT CODING, COMPILATION, AND DEBUGGING.
DATABASE TOOL	MYSQL WORKBENCH OR EQUIVALENT (OPTIONAL)	RECOMMENDED FOR SCHEMA DESIGN, EXECUTION OF DDL/DML STATEMENTS, AND VIEWING/MANAGING DATABASE TABLES AND TRIGGERS.

CHAPTER 3

MODULE DESCRIPTION

The application is structured into four primary modules: Data Model, Data Access Layer (DAO), Database Utility, and User Interfaces (UI).

3.1 DATA MODEL MODULE

This module defines the structure of the application's data objects, mirroring the tables in your MySQL database. These classes, often called Plain Old Java Objects (POJOs) or Entities, are responsible solely for holding data**.**

3.2 DATA ACCESS LAYER (DAO) MODULE

This module, often implemented using the Data Access Object (DAO) pattern, is the application's persistence interface. It contains all the JDBC code and logic required to perform CRUD (Create, Read, Update, Delete) operations on the database. It acts as the *only* gateway between the business logic (your Swing application) and the database (MySQL).

3.3 DATABASE UTILITY MODULE

- **DBConnection.java**: A singleton class responsible for managing the SQLite connection.
    - **Core Function:** Provides a static method get Connection () to obtain a live database connection.
    - **Initialization:** Contains initialize Database (), which runs on application start to create the products and cart items tables and insert initial sample data.

3.4 USER INTERFACE(UI) MODULE

The UI Module for the Online Tuition Management System is built using Java Swing and comprises three primary control points that manage the user experience, interaction, and system navigation.

1. Welcome Frame (System Entry Point)

- Function: This frame acts as the initial system entry point, typically the RegistrationApp constructor and the createHomePanel().
- Initialization: Its first responsibility is to initialize the database (by calling initDatabase()), ensuring all tables and the critical MySQL Trigger are set up before any interaction occurs.
- Navigation: In a complete security model (future scope), this frame directs users to the appropriate access point. In the current iteration, the main JMenuBar immediately grants access to the administrative functions, allowing the user to bypass the non-implemented login/customer separation.

2. Main Administration Frame (CardLayout View)

- Function: This is the primary control panel, managed by the RegistrationApp's JFrame using CardLayout. It serves as the single, centralized point for all administrative tasks.
- Navigation Control: The MenuBar is the central navigation mechanism, allowing administrators to seamlessly switch between different functional panels (cards) without opening new windows.

Chapter 4

CODING

DBCONNECTION CODE:

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.sql.*;
import java.util.Vector;

public class RegistrationApp extends JFrame {

    // --- Database Configuration ---
    // Make sure your MySQL server is running and the database 'course_registration' exists.
    private static final String DB_URL = "jdbc:mysql://localhost:3306/course_registration";  2 u
    private static final String USER = "root";  2 usages
    private static final String PASS = "KumaranSujatha@1974"; // Your specified password  2 usage

    // GUI Components
    private JPanel mainPanel;  23 usages
    private CardLayout cardLayout;  13 usages

    // References to refreshable panels
    private ListStudentsPanel listStudentsPanel;  3 usages
    private ListInstructorsPanel listInstructorsPanel;  3 usages
    private ListCoursesPanel listCoursesPanel;  3 usages
    private ListEnrollmentsPanel listEnrollmentsPanel; // NEW  4 usages
    private RecordPaymentPanel recordPaymentPanel;     // NEW  3 usages
```

WELCOMEFRAME CODE:

```
public class RegistrationApp extends JFrame {                    ⚠36 ⚠1 ✓3
    private JMenuBar createMenuBar() {  1 usage
        });
        instructorMenu.add(addInstructorItem);
        instructorMenu.add(listInstructorItem);

        // --- Course/Enrollment Menu (UPDATED) ---
        JMenu courseMenu = new JMenu( s: "Courses & Enroll");
        JMenuItem addCourseItem = new JMenuItem( text: "Add Course");
        JMenuItem listCoursesItem = new JMenuItem( text: "List Courses");
        JMenuItem enrollStudentItem = new JMenuItem( text: "Enroll Student");
        JMenuItem listEnrollmentsItem = new JMenuItem( text: "List Enrollments"); // NEW
        JMenuItem recordPaymentItem = new JMenuItem( text: "Record Payment");     // NEW


        // Action listeners for features
        addCourseItem.addActionListener( ActionEvent e -> cardLayout.show(mainPanel,  name: "AddCourse"));
        listCoursesItem.addActionListener( ActionEvent e -> {
            listCoursesPanel.refreshData();
            cardLayout.show(mainPanel,  name: "ListCourses");
        });
        enrollStudentItem.addActionListener( ActionEvent e -> cardLayout.show(mainPanel,  name: "EnrollStudent"));

        // --- NEW MENU ACTIONS ---
        listEnrollmentsItem.addActionListener( ActionEvent e -> {
            listEnrollmentsPanel.refreshData();
            cardLayout.show(mainPanel,  name: "ListEnrollments");
        });
        recordPaymentItem.addActionListener( ActionEvent e -> {
            recordPaymentPanel.refreshEnrollmentData(); // Refresh dropdown before showing
            cardLayout.show(mainPanel,  name: "RecordPayment");
        });
```

```
courseMenu.add(addCourseItem);
courseMenu.add(listCoursesItem);
courseMenu.addSeparator();
courseMenu.add(enrollStudentItem);
courseMenu.add(listEnrollmentsItem);
courseMenu.add(recordPaymentItem);

// Add menus to the bar
menuBar.add(fileMenu);
menuBar.add(studentMenu);
menuBar.add(instructorMenu);
menuBar.add(courseMenu);
```

CHAPTER 5

SCREEN SHOTS

Fig 5.1 WELCOME PAGE

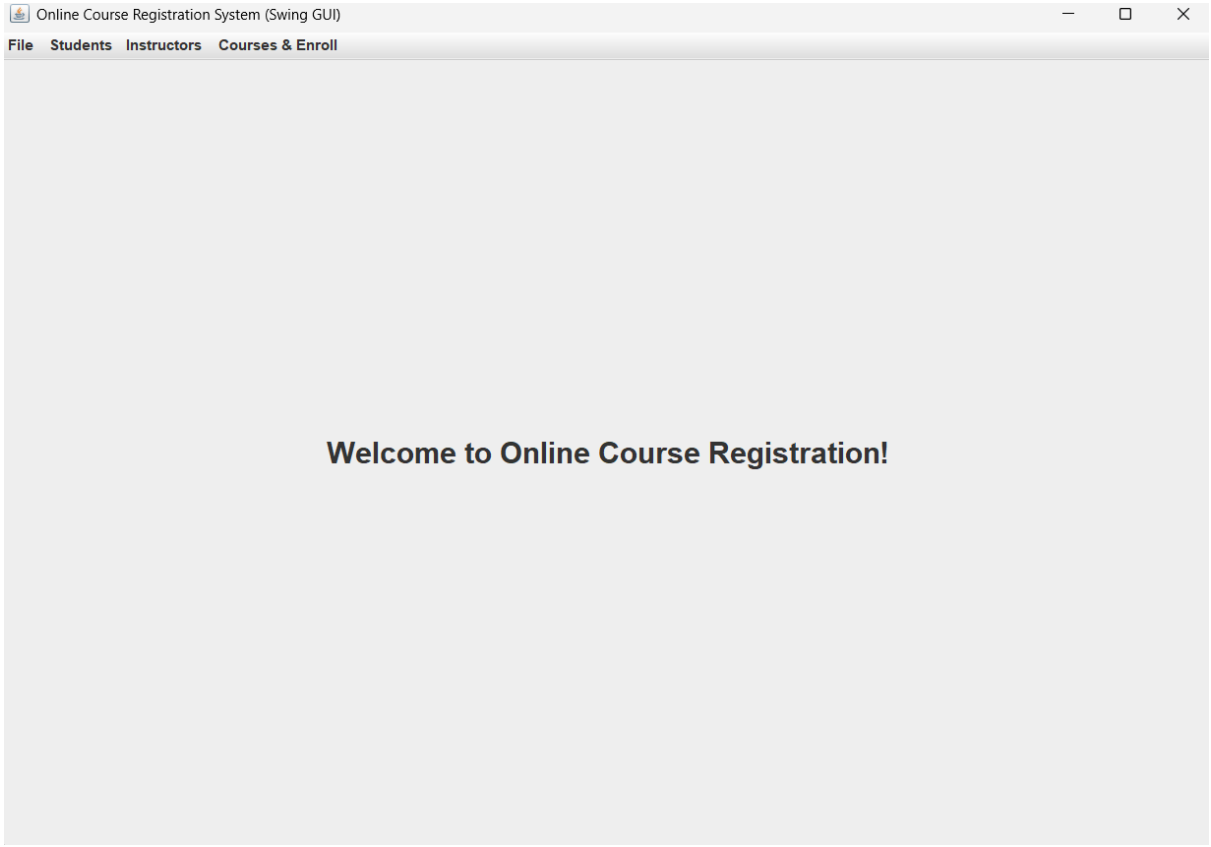


Fig 5.2 Student Login

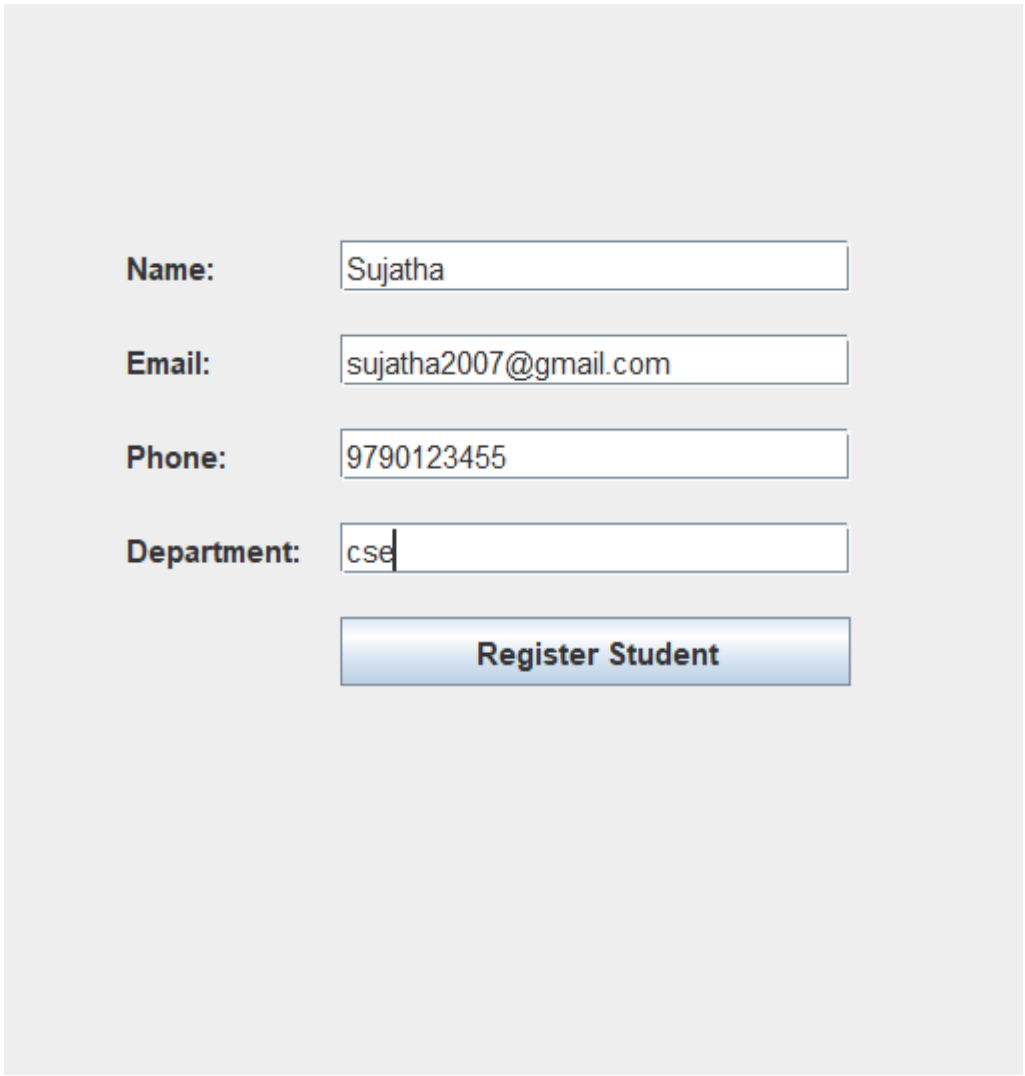

Name:
Sujatha
Email:
sujatha2007@gmail.com
Phone:
9790123455
Department:
cse
Register Student

Fig 5.3 Student Registered Successfully

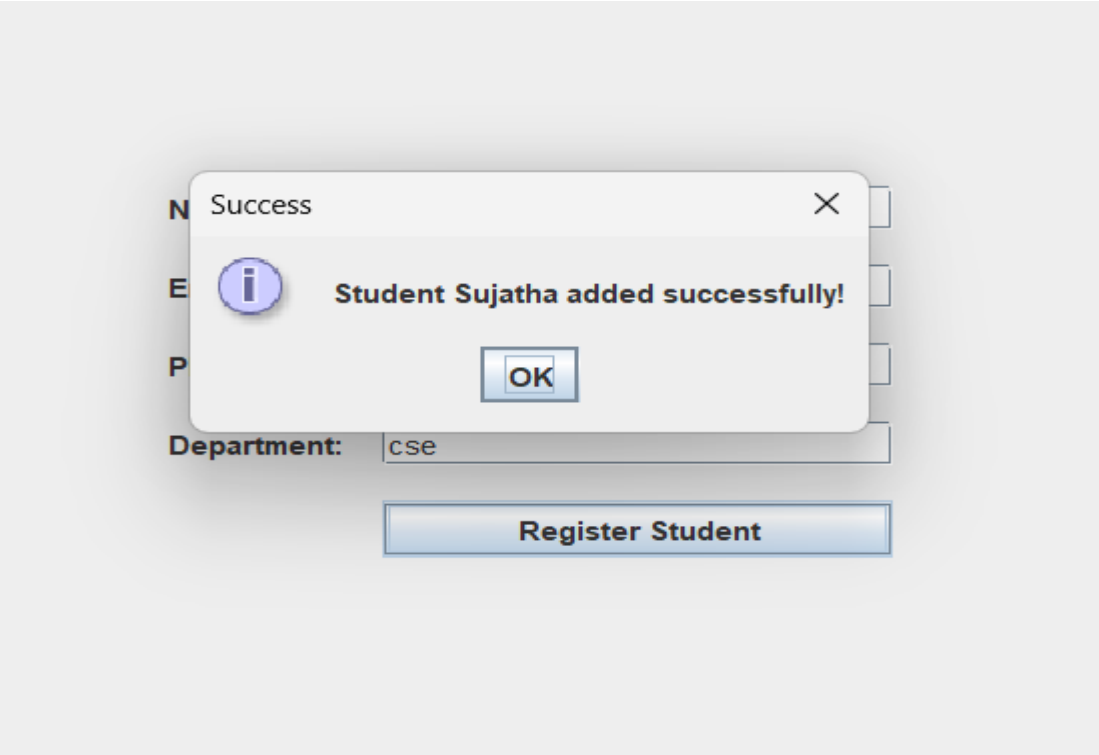


Fig 5.4 Instructor Login

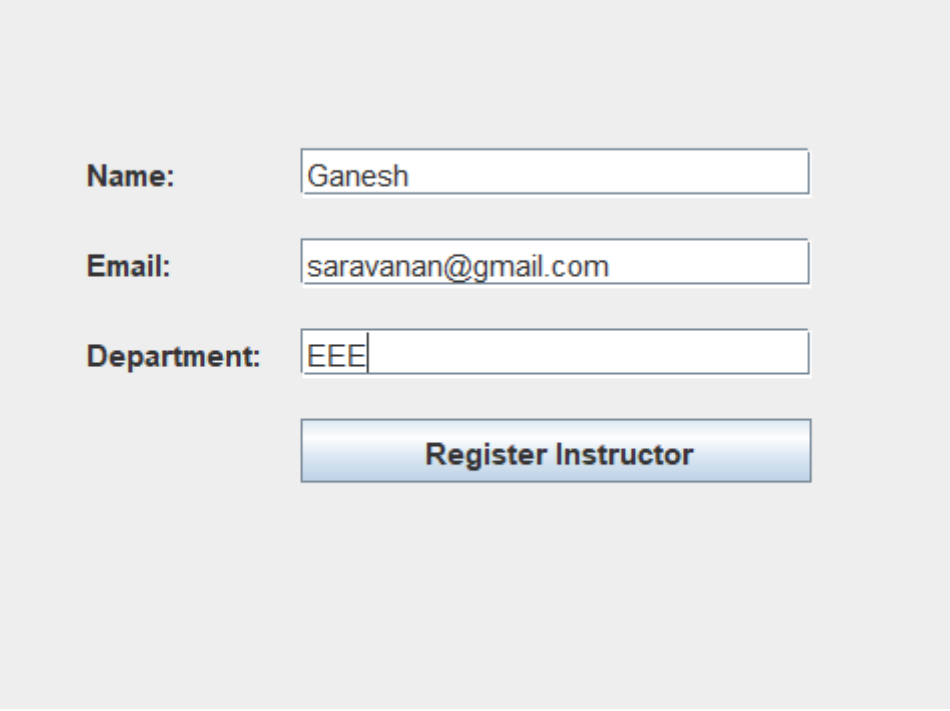


Fig 5.5 Instructor Registered Successfully

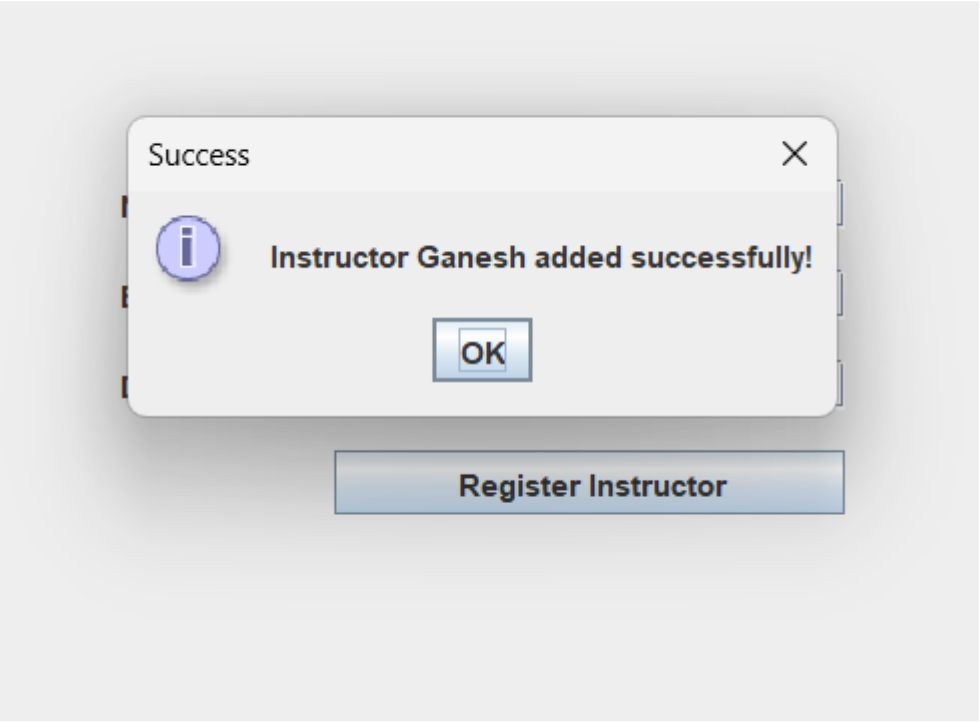


Fig 5.5 Course Registration

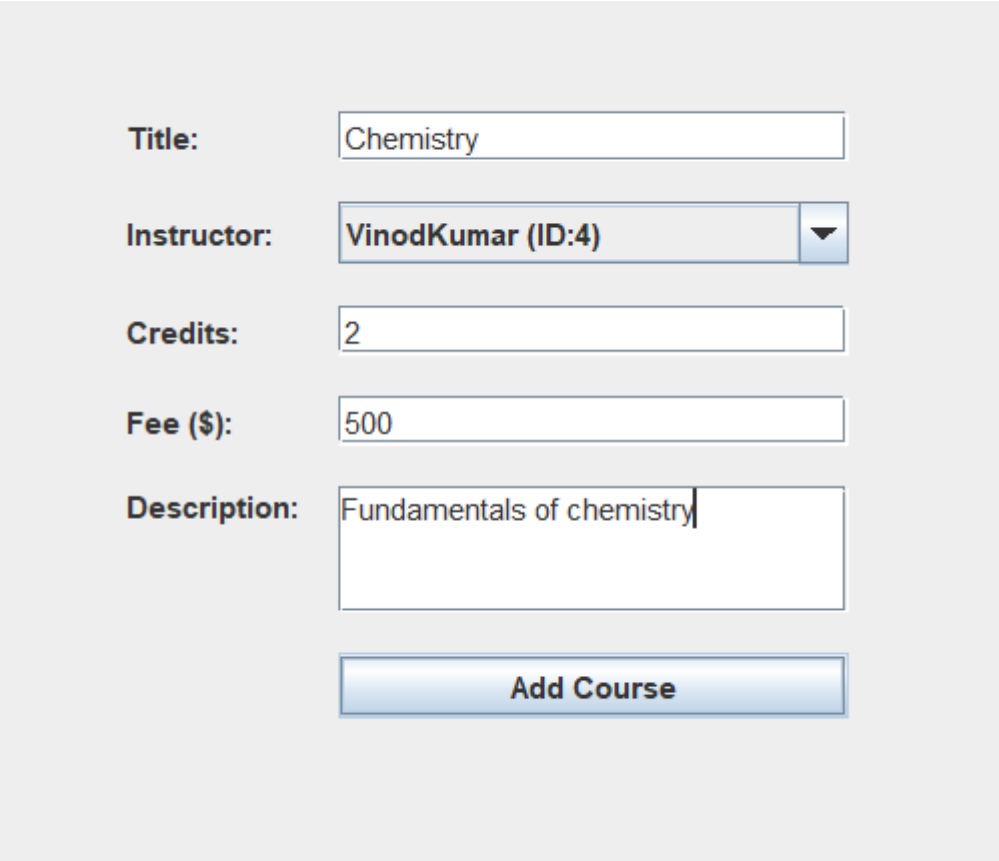


Fig 5.6 Course Registrated succesfully

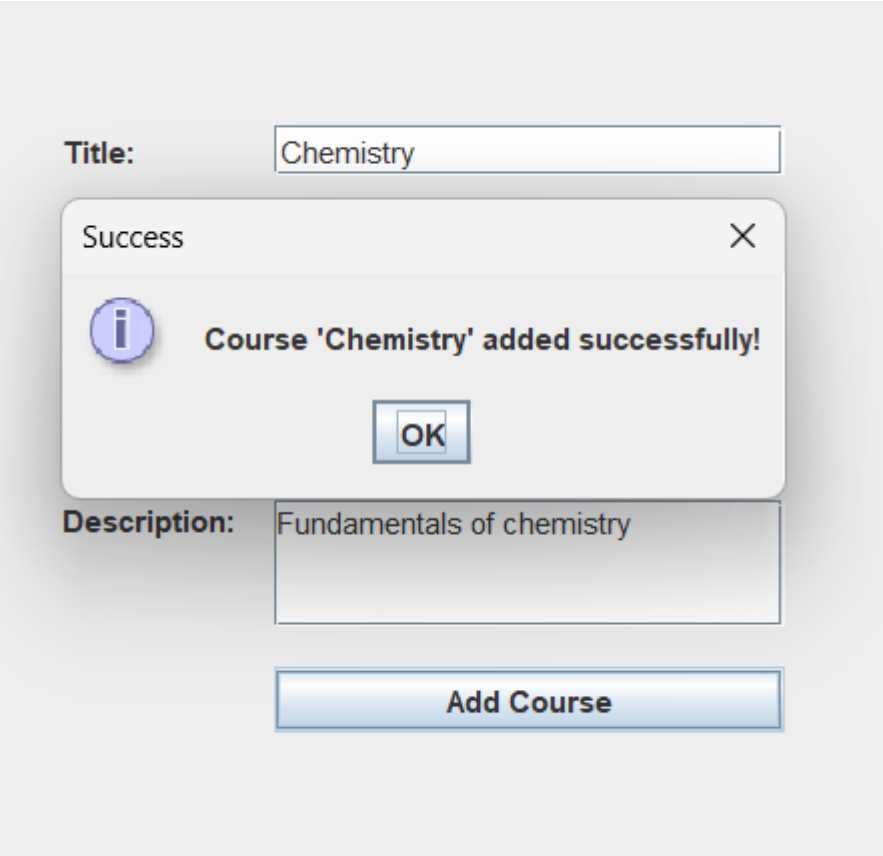
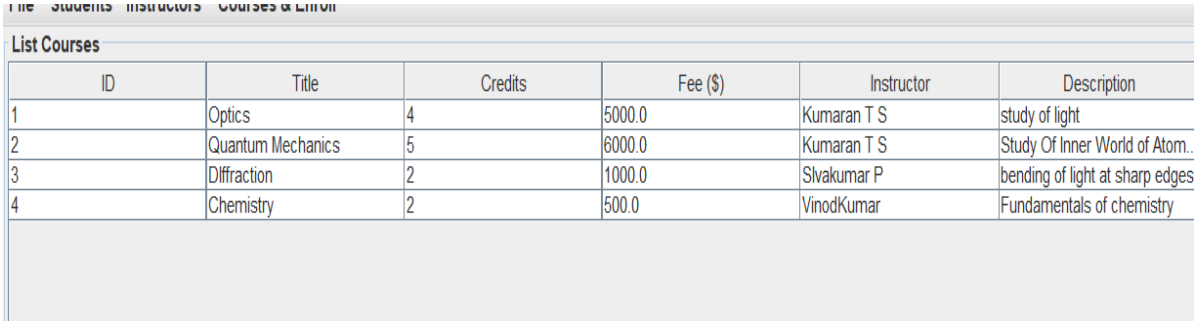


Fig 5.7 List of Courses



ID	Title	Credits	Fee ($)	Instructor	Description
1	Optics	4	5000.0	Kumaran T S	study of light
2	Quantum Mechanics	5	6000.0	Kumaran T S	Study Of Inner World of Atom..
3	Diffraction	2	1000.0	Sivakumar P	bending of light at sharp edges
4	Chemistry	2	500.0	VinodKumar	Fundamentals of chemistry

Fig 5.7 Course Enrollment

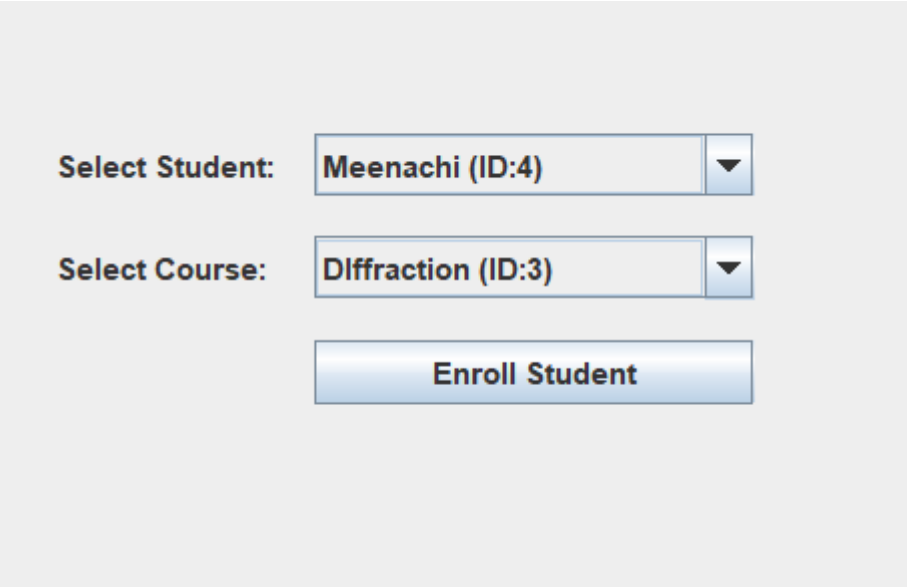


Fig 5.8 Course Enrolled Successfully

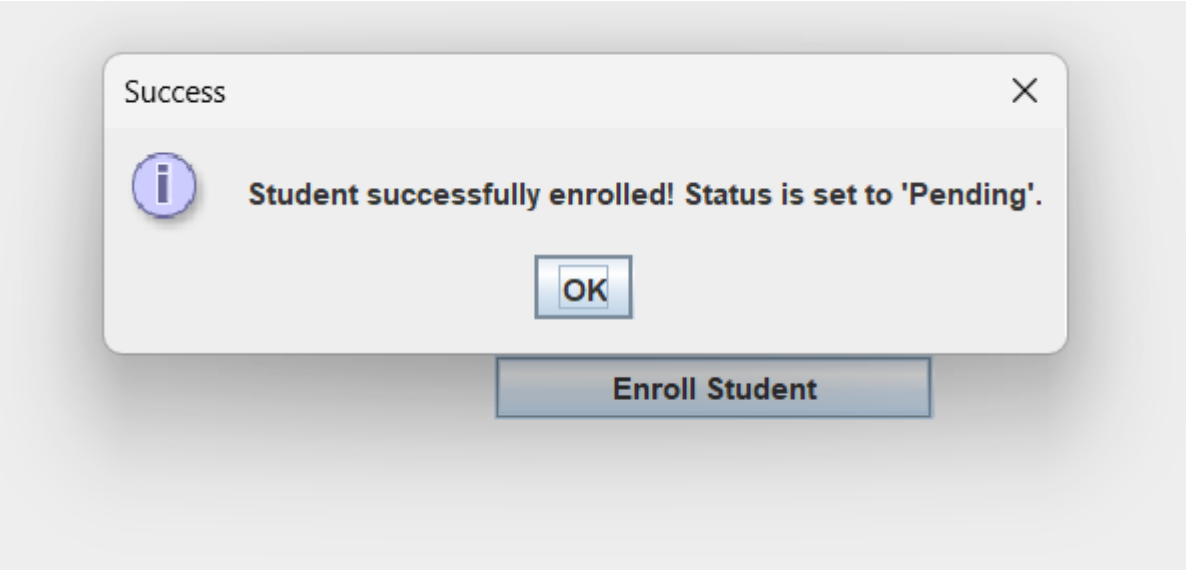


Fig 5.9 Status Set as Pending

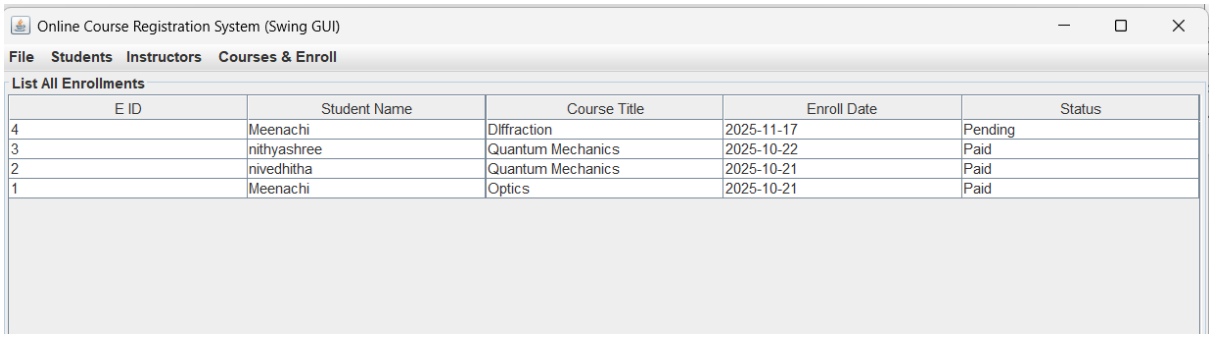


Fig 5.10 Paying Fees

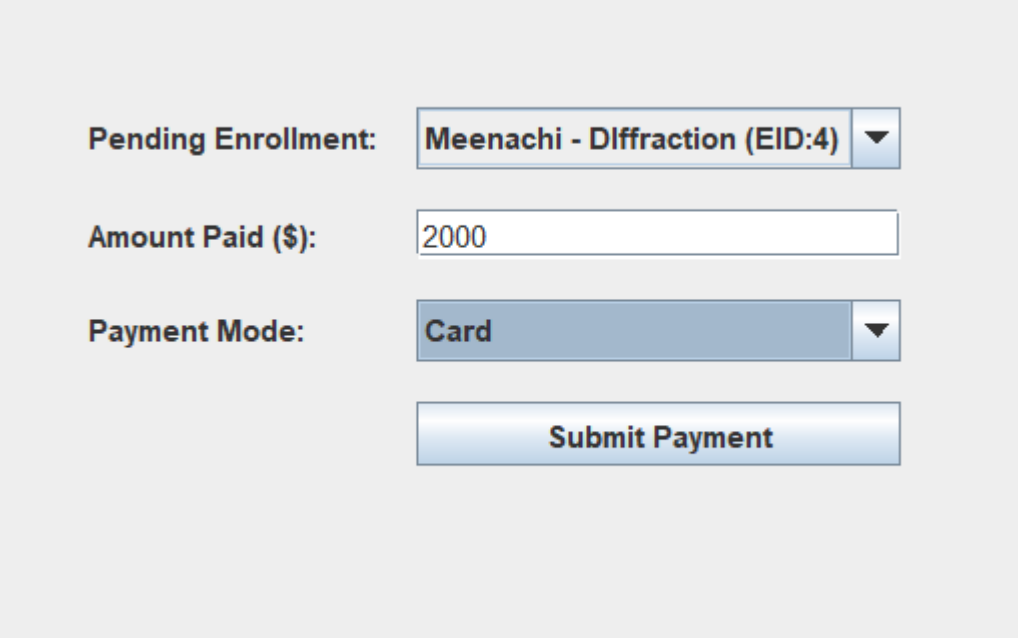
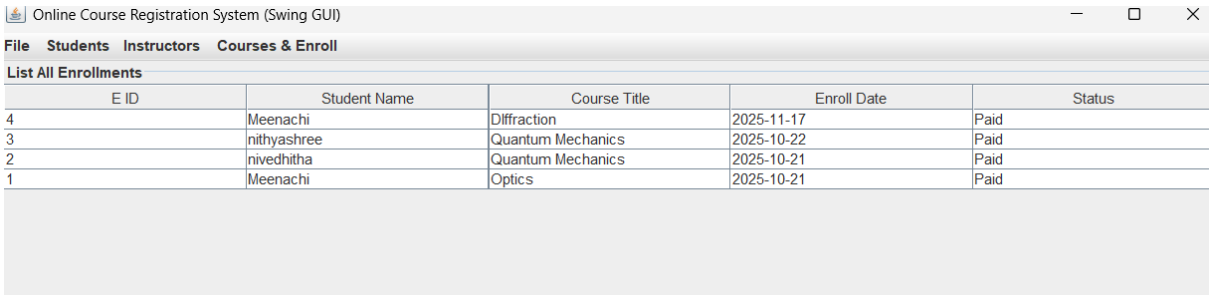


Fig 5.11 Paid Successfully



CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION

The **Online Tuition Management System (OTMS)** successfully achieves its primary aim: establishing a centralized, functional desktop administrative platform using **Java Swing** and **MySQL JDBC**. The project successfully integrates the necessary persistence layer, defining the core database schema for managing students, instructors, and tuition classes. Critically, the system demonstrates the efficacy of **database-level automation** through the use of a **MySQL Trigger**, ensuring immediate and accurate synchronization of a student's enrollment status upon fee payment. This architecture successfully addresses the key problem of manual financial tracking and provides a reliable, robust foundation for a modern tuition management application.

While the current scope focused on the essential **Create and Read (C/R)** operations and core financial workflow, the system is structurally sound and prepared for expansion into a production-ready environment.

FUTURE ENHANCEMENT

The following enhancements are recommended to transition the OTMS from its current functional prototype state to a secure, complete, and enterprise-ready application.

1. Security and Access Control (Highest Priority)

User Authentication: Implement a dedicated **Login Frame** to secure the system's entry point, addressing the current vulnerability of direct administrative access.

Role-Based Access Control (RBAC): Introduce user roles (e.g., Administrator, Instructor, Student) to restrict feature access based on permissions (e.g., Instructors can only view their assigned classes; Students can only view their own enrollment/payment status).

2. Complete CRUD Operations

- **Update and Delete Functionality (CRUD: U & D):** Integrate dedicated buttons and logic into the List View Panels (ListStudentsPanel, etc.) to enable administrators to modify or permanently remove existing records.

3. Expanded Functional Modules

- **Course Management Completion:** Finalize the module to allow full CRUD operations for courses and improve the interface for assigning instructors to courses.
- **Student/Instructor Profile Views:** Develop detailed individual views that display all related records (e.g., clicking a Student name opens a profile showing all courses enrolled and a full payment history).

4. Reporting and Analytics

- **Financial Reports:** Implement advanced queries and panels to generate analytical reports

REFERRENCES

A. Technical Documentation (APIs and Tools)

1. **Java Swing Components & GUI:**

   - Official Oracle Documentation for Swing. *Retrieved from:* https://docs.oracle.com/javase/8/docs/api/javax/swing/package-summary.htm

2. **Java Database Connectivity (JDBC) API:**

   - Oracle JDBC Basics Documentation. *Retrieved from:* https://docs.oracle.com/javase/tutorial/jdbc/index.html

3. **MySQL Server & Database Management:**

   - MySQL 8.0 Reference Manual. *Retrieved from:* https://dev.mysql.com/doc/refman/8.0/en/

4. **MySQL Connector/J (JDBC Driver):**

- Official MySQL Connector/J Documentation. *Retrieved from:* https://dev.mysql.com/doc/connector-j/8.0/en/

B. Architectural and Design Patterns

5. **Data Access Object (DAO) Pattern Implementation:**

   - Oracle Technology Network. *Data Access Object (DAO) Pattern and Best Practices. Retrieved from:* https://www.oracle.com/java/technologies/data-access-object.html

   - *Rationale:* This reference is essential for justifying the choice of the DAO pattern for separating your application logic from the JDBC implementation.

6. **Software Architecture: Separation of Concerns:**

   - *Reference:* Search results for *Separation of Concerns in Database-Oriented Architectures. Retrieved from:* https://scholar.google.com/scholar?q=Separation+of+Concerns+in+Database-Oriented+Architectures

- *Rationale:* This supports the architectural decision to isolate the UI (Swing), the Business Logic (Core Java), and the Persistence (DAO/JDBC).

C. Specialized Implementation Topics

7. **Database Triggers for Data Integrity:**

   Reference: Search results for *MySQL Trigger Implementation for Status Synchronization. Retrieved from:* https://dev.mysql.com/doc/refman/8.0/en/triggers.html

   Rationale: This source is necessary to reference the specialized technique used for the automatic enrollment status update (AutoUpdateEnrollmentStatus).

8. **JDBC Performance and Resource Management:**

   Reference: Search results for *JDBC Performance Analysis and Resource Management (Try-with-Resources). Retrieved from:* https://scholar.google.com/scholar?q=JDBC+Performance+Analysis+MySQL+Java

- *Rationale:* This validates the use of optimized practices like PreparedStatement and the Try-with-Resources construct.