



eNOTES

Designed and Developed by

- | | |
|-------------------------|---------------------|
| 1. ANJALI REDDY | 2211CS010027 |
| 2. THIRMAL RAJU | 2211CS010112 |
| 3. CHARAN KUMAR | 2211CS010327 |
| 4. SHRAVAN KUMAR | 2211CS010071 |

Guided by

Dr.G.JOSE MOSES

Department of Computer Science & Engineering

Malla Reddy University, Hyderabad

2022-2023



MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

CERTIFICATE

This certificate verifies that the Application Development Lab record titled "eNOTES", has been submitted by A. ANJALI REDDY - 2211CS010027, THIRMAL RAJU - 2211CS010112 , CHARAN KUMAR - 2211CS010327, SHRAVAN KUMAR-2211CS010071, all enrolled in B.Tech II year I semester, Department of Computer Science and Engineering, during the academic year 2022-23. The findings presented in this report have not been previously presented to any other university or institute for the purpose of obtaining a degree or diploma.

Internal Guide

Dr.G.JOSE MOSES

HOD-CSE

Dr. Shaik Meeravali

External Examiner



MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

DECLARATION

I declare that this project report titled “**eNOTES**” submitted in partial fulfillment of the degree of B. Tech in CSE is a record of original work carried out by me under the supervision of **Dr.G.JOSE MOSES**, and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgement have been made wherever the findings of others have been cited.

1. ANJALI REDDY	2211CS010027
2. THIRMAL RAJU	2211CS010112
3. CHARAN KUMAR	2211CS010327
4. SHRAVAN KUMAR	2211CS010071

ACKNOWLEDGEMENTS

We have been truly blessed to have a wonderful internal guide **Dr.G.JOSE MOSES** , Department of CSE, Malla Reddy University for guiding us to explore the ramification of our work and we express our sincere gratitude towards him for leading me through the completion of Application Development.

We would like to say our sincere thanks to **Mrs.D.SUNEETHA**, Department of CSE, App Development coordinator, for providing seamless support and right suggestions are given in the development of the APP.

We would like to say our sincere thanks to **Dr.MEERAVALI SHAIK** ,Department of CSE B . Tech Malla Reddy University for providing seamless support and right suggestions are given in the department of the APP.

We wish to explore our sincere thanks to **Dr.V.DHANUNJANA CHARI**, Dean SOS&II B .Tech SOE, Malla Reddy University for providing us with the conducive environment for carrying through our academic schedules and Project with ease.

We wish to express our sincere thanks to Vice Chancellor sir and The Management of Malla Reddy University for providing excellent infrastructure and their visionary thoughts to prepare ourselves industry ready by focusing on new technologies.

Finally, we would like to thank our family members and friends for their moral support and encouragement to achieve goals.

ABSTRACT

"eNOTES" is a Java-based desktop application designed to streamline and manage student-related information within an educational institution. The application provides a user-friendly interface with a login system to ensure secure access. Upon launching, users are greeted with a login page where they must enter a valid username and password adhering to specific patterns. The login credentials are validated, and successful authentication leads to the main application window.

The main application window of "eNOTES" offers a range of functionalities through buttons like "Student Details," "Upload PDF," "View PDF," and "Exit." These features are organized in a clean and intuitive layout. Users can access student details, upload PDF documents to a centralized database, view previously uploaded PDFs, and exit the application. The user interface is designed to be straightforward, making it accessible for administrators or staff responsible for managing student records and documents.

Within the application, there is a modular structure for student details, allowing users to explore examination records, attendance records, and discipline records through dedicated buttons. Each module opens a new window with specific data entry fields and a "Save" button, enabling the input and storage of relevant information. The application utilizes a SQLite database for data storage, ensuring efficiency and reliability in managing student records.

Additionally, "eNOTES" supports the upload and retrieval of PDF documents. Users can upload PDFs through the "Upload PDF" button, and the application displays a list of uploaded PDFs with clickable buttons for viewing. The viewing functionality leverages the default system viewer for PDFs, providing a seamless experience for users. Overall, "eNOTES" serves as a comprehensive tool for educational institutions to efficiently manage student-related information and documents in a secure and organized manner. The interface is designed with a user-friendly layout, including buttons for easy navigation and interaction with different functionalities.

TABLE OF CONTENT

DESCRIPTION	PAGE NUMBER
CERTIFICATE	i
DECLARATION	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
LIST OF FIGURES	vi
Chapter 1 : INTRODUCTION	
1.1 Introduction	1
1.2 Problem Statement	3
1.3 Objective of Project	3
1.4 Goal of Project	3
Chapter 2 : PROBLEM IDENTIFICATION	
2.1 Existing System	5
2.2 Proposed System	6
Chapter 3 : REQUIREMENTS	
3.1 Software Requirements	7
3.2 Hardware Requirements	8
Chapter 4 : DESIGN AND IMPLEMENTATION	
4.1 Design	9
4.2 Implementation	11
Chapter 5 : CODE	
5.1 Source Code	13
5.2 Screenshot of Application	23
Chapter 6 : RESULTS AND CONCLUSION	
6.1 Results	27
6.2 Conclusion	29
REFERNECES	30

LIST OF FIGURES

FIGURES	TITLE	PAGE NUMBER
4.1	Design	9
4.1.2	Data Flow Diagram	10
Fig 01	Login	23
Fig 02	Main page	23
Fig 03	Student details	24
Fig 04	Examination Records	24
Fig 05	Attendance Records	25
Fig 06	Discipline Records	25
Fig 07	Pdf Uploading	26
Fig 08	Pdf Uploading	26

CHAPTER – 1

INTRODUCTION

1.1 Introduction

An eNOTES application, a simple yet comprehensive system designed to manage student-related data and documents. Leveraging the Java Swing library for the graphical user interface, the application begins with a login page to ensure secure access. Users are prompted to enter their credentials, with features like a password toggle for enhanced user experience. The system employs a SQLite database for data storage, ensuring reliability and ease of integration. Following successful login, users gain access to a main application window, where they can navigate through various functionalities. The application encompasses features such as managing student details, uploading and viewing PDF documents, and maintaining discipline, examination, and attendance records.

Login Page:

The login page serves as the initial gateway to the eNOTES application. Utilizing Java Swing components, it provides a user-friendly interface for entering credentials. Users input their username and password, with the option to toggle password visibility. The system employs regular expressions to validate the entered credentials against predefined patterns, enhancing security and ensuring the correct format for the username and password. In case of invalid credentials, a user-friendly error message is displayed, guiding users to correct their input.

Main Application Window:

Upon successful login, users are seamlessly directed to the main application window. This window serves as the central hub for accessing diverse functionalities. The graphical user interface includes intuitive buttons for managing student details, uploading PDF documents, viewing uploaded PDFs, and exiting the application. Additionally, an exit button ensures a smooth closure of the application.

Student Details Section:

The application provides a dedicated section for managing student details. Users can navigate to his section through a button on the main application window. Within the student details section, three distinct functionalities are available: examination records, attendance records, and discipline records.

Examination Records:

The examination records functionality allows users to input and save student examination details. A window with input fields for student name and marks provides an interactive interface. Upon clicking the "Save" button, the entered data is validated and stored in the SQLite database. The system showcases robust exception handling, ensuring graceful error messages in case of database-related issues. The separation of concerns is maintained, enhancing the code's readability and maintainability.

Attendance Records:

Similar to examination records, the attendance records functionality offers a window with input fields for student name and attendance. Users can save attendance records, with the system handling the database interactions seamlessly. Consistency in design and functionality is maintained across different record types, providing a cohesive user experience.

Discipline Records:

The discipline records functionality follows a parallel structure to the examination and attendance records. Users can input student name and discipline-related details, with a robust save mechanism. The system ensures that data is securely stored in the database, contributing to the overall integrity of the eNOTES application.

PDF Management:

The eNOTES application extends its capabilities beyond record management by incorporating PDF-related functionalities. Users can upload PDF documents to the database and subsequently view the uploaded PDFs. The system uses Java's File and IO operations to handle PDF files, with comprehensive error handling to address potential issues during file operations. The uploaded PDFs are displayed in a separate window, providing users with a seamless and efficient document management experience.

In summary, the eNOTES application presented in the Java code combines user-friendly design with robust functionality, catering to the management of student details, examination records, attendance records, discipline records, and PDF documents. The modular structure, intuitive user interface, and effective database interactions collectively contribute to the application's reliability and ease of use.

1.2. Problem Statement

Managing student information, academic records, and documents in educational institutions can be a cumbersome task. Traditional methods often involve manual data entry, which is prone to errors and time-consuming. Additionally, handling various types of records and documents requires a systematic and user-friendly approach. The lack of an efficient and centralized system can lead to inefficiencies and difficulties in tracking and organizing important information.

1.3. Objective

The objective of the eNOTES project is to create a comprehensive educational management system that streamlines the process of managing student details, academic records, and uploaded documents. The system aims to provide a user-friendly interface for administrators, teachers, and other educational staff to efficiently handle tasks related to student information, examination records, attendance tracking, discipline records, add document management.

1.4. Goal of Project

User Authentication and Access Control:

Implement a secure login system to authenticate users and control access to the application based on their roles.

Student Information Management:

Develop features for entering, viewing, and managing student details, ensuring accuracy and completeness.

Academic Record Keeping:

Create functionality to record and manage examination records, including student names and marks.

Attendance Tracking:

Implement features to record and monitor student attendance, facilitating efficient tracking of attendance data.

Discipline Record Management:

Provide a module for entering and managing discipline-related information for students. Design an intuitive graphical user interface (GUI) using Java Swing, making it easy for users to navigate and interact with the application.

Document Upload and Viewing:

Enable users to upload PDF documents, store them in a database, and view a list of uploaded documents.

Database Integration:

Utilize a relational database (SQLite in this case) to store and retrieve information securely.

User-Friendly Interface:

Design an intuitive graphical user interface (GUI) using Java Swing, making it easy for users to navigate and interact with the application.

Data Integrity and Validation:

Implement validation checks to ensure the integrity of data entered into the system, preventing errors and inconsistencies.

Graceful Application Exit:

Provide a clean and efficient method for users to exit the application while handling any necessary clean up task.

CHAPTER 2

PROBLEM IDENTIFICATION

2.1. Existing System

eNOTES system, educational institutions may rely on manual and decentralized methods to manage student information, academic records, and documents. Common practices in the existing system might include:

Manual Data Entry:

Student details, examination records, attendance information, and discipline records are often recorded manually, leading to the possibility of errors and inefficiencies.

Paper-Based Document Management:

Documents, including academic records and other important information, may be stored in physical files or folders, making it challenging to organize and retrieve data quickly.

Limited Accessibility:

Access to information is often restricted to specific individuals or departments, and collaboration among different stakeholders may be hindered.

Data Redundancy:

With decentralized record-keeping, there is a risk of data redundancy, as the same information may be stored in multiple locations without synchronization.

Time-Consuming Processes:

Administrative tasks related to student information and record-keeping can be time-consuming, affecting the overall efficiency of the educational institution.

2.2. Proposed System

The eNOTES system is proposed as a comprehensive solution to address the limitations of the existing system. The proposed system offers several enhancements and features:

Centralized Information Management:

The eNOTES system centralizes student information, academic records, and documents in a secure database, reducing the reliance on manual methods.

Efficient Record Keeping:

Features for managing examination records, attendance tracking, and discipline records are integrated into a single system, providing a more streamlined and efficient approach.

Document Upload and Retrieval:

Users can upload PDF documents directly into the system, enhancing document management and accessibility. Uploaded documents are stored in a database, facilitating easy retrieval.

User Authentication and Access Control:

The system implements secure user authentication, ensuring that only authorized personnel can access specific features and data, improving overall security.

User-Friendly Interface:

The graphical user interface (GUI) of the eNOTES system is designed to be user-friendly, making it easy for administrators, teachers, and staff to navigate and perform tasks.

Data Validation and Integrity:

The system incorporates validation checks to ensure the accuracy and integrity of data entered, minimizing errors and inconsistencies.

Enhanced Collaboration:

The proposed system facilitates collaboration among different stakeholders by providing a centralized platform for accessing and sharing information.

Graceful Application Exit:

The system includes an "Exit" feature, allowing users to exit the application gracefully, with the potential for performing cleanup tasks.

CHAPTER 3

REQUIREMENTS

3.1. Software Requirements

Software Requirements for eNOTES Application (Command Line Version):

Operating System:

eNOTES, in its command-line version, can run on various operating systems. It is platform-independent and can be executed on:

Windows Command Prompt

macOS Terminal

Linux Terminal

Java Version:

The command-line version of eNOTES is developed using Java. Users need to have Java Runtime Environment (JRE) installed on their machines. eNOTES is compatible with:

Version: JDK 8 or later

Purpose: JDK is essential for developing and running Java applications. The eNOTES application is written in Java, making the JDK a fundamental requirement.

The command-line version of eNOTES is designed for straightforward usage in a terminal environment, offering a lightweight and accessible way for users to interact with health-related information and recommendations.

3.2. Hardware Requirements

The hardware requirements for running the eNOTES application on a desktop system are relatively modest. The specifications outlined below are suitable for optimal performance:

Processor:

Minimum: Dual-core processor (e.g., Intel Core i3 or equivalent) Recommended: Quad-core processor or higher

RAM (Memory):

Minimum: 4 GB

Recommended: 8 GB or higher

Storage:

Minimum: 20 GB of available disk space

Recommended: SSD for faster read and write speeds

Display:

Resolution: 1024x768 pixels or higher

Color Depth: 24-bit or higher

Input Devices:

Keyboard and mouse (or alternative pointing device)

Network:

An internet connection is not mandatory for local execution but might be required for certain features like database updates or external document retrieval.

Operating System:

The eNOTES application is compatible with common desktop operating systems, including Windows, Linux, and macOS.

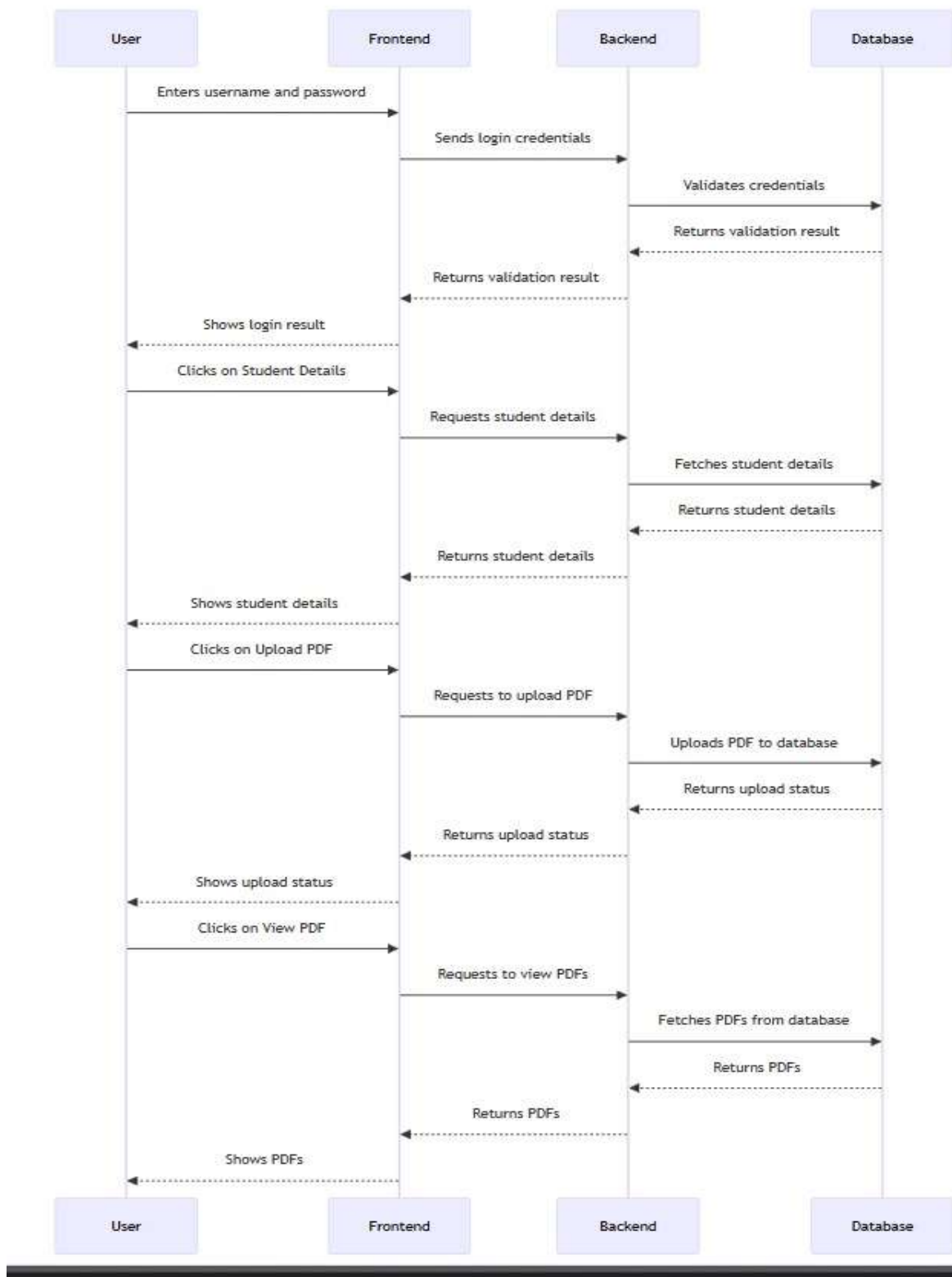
PDF Viewer:

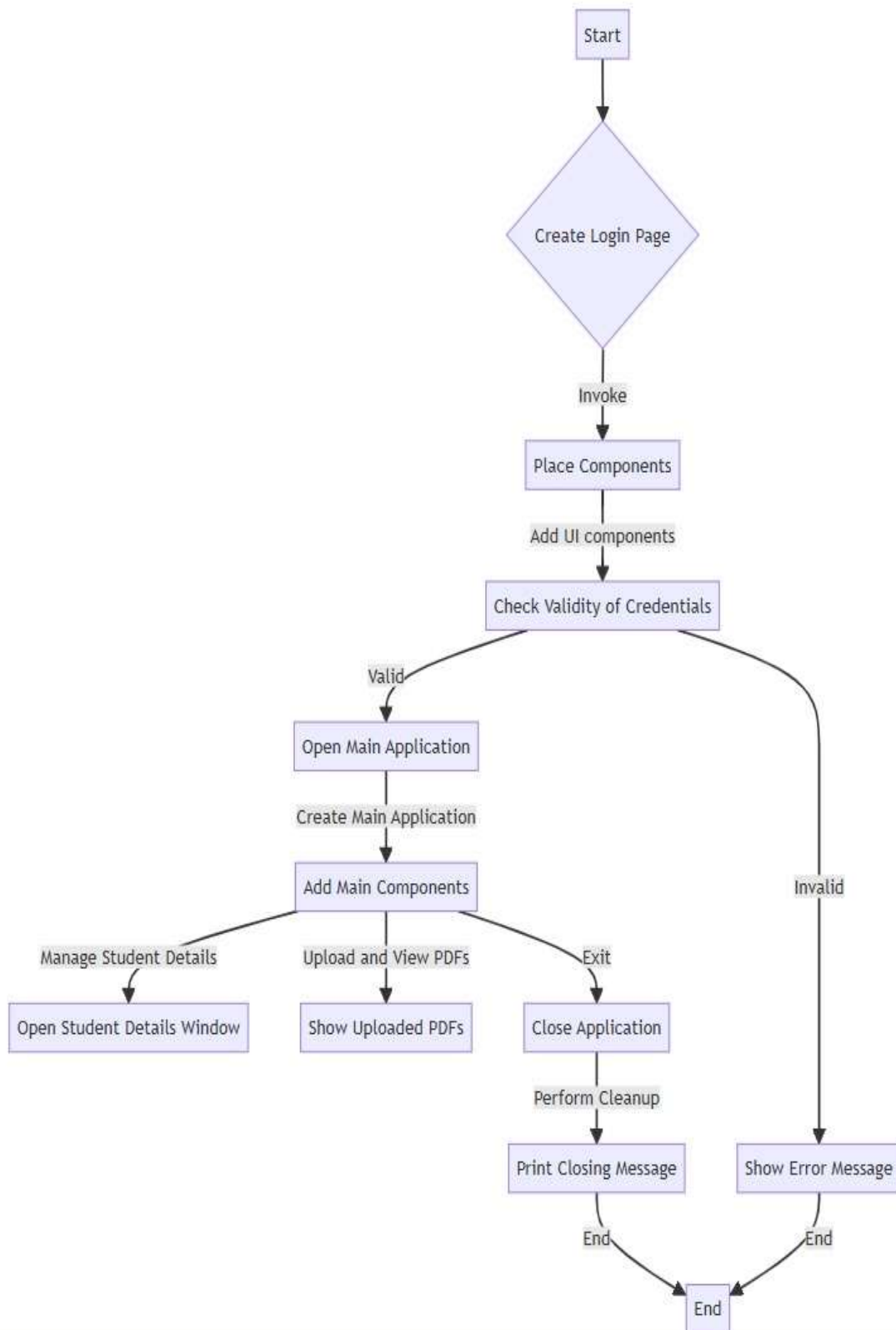
A default system PDF viewer is required for opening and displaying PDF documents. The capabilities of the host operating system's PDF viewer are utilized by the application.

CHAPTER 4

DESIGN AND IMPLEMENTATION

4.1 Design





Fig; 4.1.2 DATA FLOW DIAGRAM

4.2. Implementation

Implementation Overview:

The implementation of the eNOTES application involves writing code to bring the designed architecture to life. Here's a brief overview of the key implementation steps:

User Interface Implementation:

Use Java Swing to create graphical user interface components such as frames, panels, buttons, labels, text fields, and password fields. Implement event listeners to handle user interactions, including button clicks and checkbox selections.

Develop separate frames for the login page, main application window, student details, examination records, attendance records, discipline records, document upload, and PDF viewing.

Database Integration:

Establish a connection to the SQLite database using JDBC. Write code to create tables for storing student details, examination records, attendance records, discipline records, and PDF documents. Implement methods for CRUD (Create, Read, Update, Delete) operations on the database.

User Authentication:

Create a secure login page where users can enter their credentials. Validate the entered credentials against predefined patterns for the username and password. Redirect authenticated users to the main application window and display an error message for invalid login attempts.

Student Details Module:

Implement methods to enter, view, and manage student information. Validate user inputs for student details and display appropriate messages for validation errors.

Examination Records Module:

Develop functionalities to save and retrieve examination records, including student names and marks. Implement input validation to ensure the correctness of entered data.

Attendance Records Module:

Implement features to record and manage student attendance. Validate inputs related to attendance information and display error messages for invalid entries.

Discipline Records Module:

Create methods to save and retrieve discipline-related information for students. Implement validation checks for discipline records input.

Document Upload Module:

Implement a file upload feature to upload PDF documents to the database. Develop methods to fetch and display a list of uploaded PDFs.

PDF Viewing Module:

Use the default system PDF viewer to open and display PDF documents. Implement methods to retrieve and display a selected PDF document.

Error Handling:

Implement robust error-handling mechanisms to manage exceptions gracefully. Display informative error messages for unsuccessful operations, database connection issues, or data validation errors.

Window Management:

Write code to manage the opening and closing of windows, ensuring a smooth transition between different modules. Implement a window listener to perform cleanup or additional actions when the main frame is closed.

Testing:

Perform thorough testing of each module to ensure functionality, correctness, and user interface responsiveness. Test edge cases, such as invalid inputs and extreme scenarios, to validate the robustness of the application.

Documentation:

Create documentation that includes code comments, user guides, and any additional information necessary for future maintenance or development.

Version Control:

Utilize version control (e.g., Git) to track changes, collaborate with multiple developers, and manage the source code.

CHAPTER 5

CODE

5.1. Source Code

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.nio.file.*;
import java.sql.*;
import java.util.*;
import java.util.List;

public class Main {

    // JDBC Database URL
    static final String DB_URL = "jdbc:sqlite:student_database.db";

    public static void main(String[] args) {
        // Create the login page
        SwingUtilities.invokeLater(() -> createLoginPage());
    }

    private static void createLoginPage() {
        JFrame frame = new JFrame("Login");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 150);
        frame.setLocationRelativeTo(null);

        JPanel panel = new JPanel();
        frame.add(panel);
        placeComponents(panel, frame);

        frame.setVisible(true);
    }

    private static void placeComponents(JPanel panel, JFrame loginFrame) {
        panel.setLayout(null);

        JLabel userLabel = new JLabel("Username:");
        userLabel.setBounds(10, 20, 80, 25);
        panel.add(userLabel);

        JTextField userText = new JTextField(20);
        userText.setBounds(100, 20, 165, 25);
        panel.add(userText);

        JLabel passwordLabel = new JLabel("Password:");
        passwordLabel.setBounds(10, 50, 80, 25);
        panel.add(passwordLabel);

        JPasswordField passwordText = new JPasswordField(20);
        passwordText.setBounds(100, 50, 135, 25);
        panel.add(passwordText);
    }
}
```

```

// Show password checkbox
JCheckBox showPasswordCheckBox = new JCheckBox("Show");
showPasswordCheckBox.setBounds(240, 50, 60, 25);
panel.add(showPasswordCheckBox);
JButton loginButton = new JButton("Login");
loginButton.setBounds(10, 80, 80, 25);
panel.add(loginButton);
// ActionListener for the login button
loginButton.addActionListener(e -> {
    String username = userText.getText();
    String password = new String(passwordText.getPassword());
    // Check credentials
    if (isValidUser(username, password)) {
        // If credentials are valid, open the main application window
        openMainApplication(loginFrame);
    } else {
        JOptionPane.showMessageDialog(null, "Invalid login credentials.. Please! Enter your RollNumber...");
    }
});
// ActionListener for the show password checkbox
showPasswordCheckBox.addActionListener(e -> {
    if (showPasswordCheckBox.isSelected()) {
        // Show the password
        passwordText.setEchoChar((char) 0);
    } else {
        // Hide the password
        passwordText.setEchoChar('*');
    }
});
}

private static boolean isValidUser(String username, String password) {
    // Check if the username follows the specified pattern
    if (!username.matches("[a-z]*")) {
        return false;
    }

    // Check if the password follows the specified pattern
    if (!password.matches("2211cs010\\d{3}")) {
        return false;
    }

    return true;
}

private static void openMainApplication(JFrame loginFrame) {
    // Close the login frame
    loginFrame.dispose();

    // Create the main application frame
    JFrame mainFrame = new JFrame("eNOTES");
    mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    mainFrame.setSize(600, 400);
    mainFrame.setLocationRelativeTo(null);
}

```

```

// Create a panel for the main application
JPanel mainPanel = new JPanel();
mainFrame.add(mainPanel);
placeMainComponents(mainPanel);

// Show the main application frame
mainFrame.setVisible(true);

// Add a window listener to perform actions when the main frame is closed
mainFrame.addWindowListener(new WindowAdapter() {
    @Override
    public void windowClosing(WindowEvent e) {
        // Perform cleanup or additional actions if needed
        System.out.println("Closing application");
    }
});
}

private static void placeMainComponents(JPanel mainPanel) {
    mainPanel.setLayout(null);

    JButton studentDetailsButton = new JButton("Student Details");
    studentDetailsButton.setBounds(10, 20, 150, 25);
    mainPanel.add(studentDetailsButton);

    JButton uploadPdfButton = new JButton("Upload PDF");
    uploadPdfButton.setBounds(10, 60, 150, 25);
    mainPanel.add(uploadPdfButton);

    JButton viewPdfButton = new JButton("View PDF");
    viewPdfButton.setBounds(10, 100, 150, 25);
    mainPanel.add(viewPdfButton);

    // ActionListener for the student details button
    studentDetailsButton.addActionListener(e -> {
        // Open a new window for student details
        openStudentDetailsWindow();
    });

    // ActionListener for the upload PDF button
    uploadPdfButton.addActionListener(e -> {
        JFileChooser fileChooser = new JFileChooser();
        int result = fileChooser.showOpenDialog(null);
        if (result == JFileChooser.APPROVE_OPTION) {
            File selectedFile = fileChooser.getSelectedFile();
            // Perform the upload to the database
            uploadPdfToDatabase(selectedFile);
            JOptionPane.showMessageDialog(null, "PDF uploaded successfully");
        }
    });

    // ActionListener for the view PDF button
    viewPdfButton.addActionListener(e -> {
        // Fetch and display the uploaded PDFs
        showUploadedPDFs();
    });

    JButton exitButton = new JButton("Exit"); // Added Exit button

```

```

exitButton.setBounds(10, 140, 150, 25);
mainPanel.add(exitButton);

// ActionListener for the exit button
exitButton.addActionListener(e -> {
    // Close the main frame and exit the application
    System.exit(0);
});
}

private static void openStudentDetailsWindow() {
    JFrame studentDetailsFrame = new JFrame("Student Details");
    studentDetailsFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    studentDetailsFrame.setSize(600, 400);
    studentDetailsFrame.setLocationRelativeTo(null);

    // Create a panel for student details
    JPanel studentDetailsPanel = new JPanel();
    studentDetailsFrame.add(studentDetailsPanel);
    placeStudentDetailsComponents(studentDetailsPanel);
    // Show the student details frame
    studentDetailsFrame.setVisible(true);
}

private static void placeStudentDetailsComponents(JPanel studentDetailsPanel) {
    studentDetailsPanel.setLayout(new GridLayout(3, 1));

    JButton examinationRecordsButton = new JButton("Examination Records");
    JButton attendanceRecordsButton = new JButton("Attendance Records");
    JButton disciplineRecordsButton = new JButton("Discipline Records");

    studentDetailsPanel.add(examinationRecordsButton);
    studentDetailsPanel.add(attendanceRecordsButton);
    studentDetailsPanel.add(disciplineRecordsButton);

    // ActionListener for examination records button
    examinationRecordsButton.addActionListener(e -> {
        // Open a new window for examination records
        openExaminationRecordsWindow();
    });

    // ActionListener for attendance records button
    attendanceRecordsButton.addActionListener(e -> {
        // Open a new window for attendance records
        openAttendanceRecordsWindow();
    });

    // ActionListener for discipline records button
    disciplineRecordsButton.addActionListener(e -> {
        // Open a new window for discipline records
        openDisciplineRecordsWindow();
    });
}

private static void openExaminationRecordsWindow() {
    JFrame examinationRecordsFrame = new JFrame("Examination Records");
    examinationRecordsFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

```

```

examinationRecordsFrame.setSize(600, 400);
examinationRecordsFrame.setLocationRelativeTo(null);

// Create a panel for examination records
JPanel examinationRecordsPanel = new JPanel();
examinationRecordsFrame.add(examinationRecordsPanel);
placeExaminationRecordsComponents(examinationRecordsPanel);

// Show the examination records frame
examinationRecordsFrame.setVisible(true);
}

private static void placeExaminationRecordsComponents(JPanel examinationRecordsPanel) {
    examinationRecordsPanel.setLayout(new GridLayout(3, 1));

    JLabel nameLabel = new JLabel("Name:");
    JTextField nameField = new JTextField();
    JLabel marksLabel = new JLabel("Marks:");
    JTextField marksField = new JTextField();
    JButton saveButton = new JButton("Save");

    examinationRecordsPanel.add(nameLabel);
    examinationRecordsPanel.add(nameField);
    examinationRecordsPanel.add(marksLabel);
    examinationRecordsPanel.add(marksField);
    examinationRecordsPanel.add(saveButton);

    // ActionListener for the save button
    saveButton.addActionListener(e -> {
        String name = nameField.getText();
        String marks = marksField.getText();
        // Save the examination record to the database
        saveExaminationRecord(name, marks);
        // Clear the input fields
        nameField.setText("");
        marksField.setText("");

        JOptionPane.showMessageDialog(null, "Examination record saved successfully");
    });
}

private static void saveExaminationRecord(String name, String marks) {
    try (Connection connection = DriverManager.getConnection(DB_URL)) {
        String sql = "INSERT INTO examination_records (name, marks) VALUES (?, ?)";

        try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
            preparedStatement.setString(1, name);
            preparedStatement.setString(2, marks);

            preparedStatement.executeUpdate();
        }
        System.out.println("Examination record saved to the database successfully");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}

```



```

private static void openAttendanceRecordsWindow() {
    JFrame attendanceRecordsFrame = new JFrame("Attendance Records");
    attendanceRecordsFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    attendanceRecordsFrame.setSize(600, 400);
    attendanceRecordsFrame.setLocationRelativeTo(null);

    // Create a panel for attendance records
    JPanel attendanceRecordsPanel = new JPanel();
    attendanceRecordsFrame.add(attendanceRecordsPanel);
    placeAttendanceRecordsComponents(attendanceRecordsPanel);

    // Show the attendance records frame
    attendanceRecordsFrame.setVisible(true);
}

private static void placeAttendanceRecordsComponents(JPanel attendanceRecordsPanel) {
    attendanceRecordsPanel.setLayout(new GridLayout(3, 1));

    JLabel nameLabel = new JLabel("Name:");
    JTextField nameField = new JTextField();
    JLabel attendanceLabel = new JLabel("Attendance:");
    JTextField attendanceField = new JTextField();
    JButton saveButton = new JButton("Save");

    attendanceRecordsPanel.add(nameLabel);
    attendanceRecordsPanel.add(nameField);
    attendanceRecordsPanel.add(attendanceLabel);
    attendanceRecordsPanel.add(attendanceField);
    attendanceRecordsPanel.add(saveButton);

    // ActionListener for the save button
    saveButton.addActionListener(e -> {
        String name = nameField.getText();
        String attendance = attendanceField.getText();

        // Save the attendance record to the database
        saveAttendanceRecord(name, attendance);

        // Clear the input fields
        nameField.setText("");
        attendanceField.setText("");

        JOptionPane.showMessageDialog(null, "Attendance record saved successfully");
    });
}

private static void saveAttendanceRecord(String name, String attendance) {
    try (Connection connection = DriverManager.getConnection(DB_URL)) {
        String sql = "INSERT INTO attendance_records (name, attendance) VALUES (?, ?)";

        try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
            preparedStatement.setString(1, name);
            preparedStatement.setString(2, attendance);

            preparedStatement.executeUpdate();
        }
    }
}

```

```

System.out.println("Attendance record saved to the database successfully");
    } catch (SQLException e) {
e.printStackTrace();
        JOptionPane.showMessageDialog(null, "Error saving attendance record to the database");
    }
}

private static void openDisciplineRecordsWindow() {
    JFrame disciplineRecordsFrame = new JFrame("Discipline Records");
    disciplineRecordsFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    disciplineRecordsFrame.setSize(600, 400);
    disciplineRecordsFrame.setLocationRelativeTo(null);

    // Create a panel for discipline records
    JPanel disciplineRecordsPanel = new JPanel();
    disciplineRecordsFrame.add(disciplineRecordsPanel);
    placeDisciplineRecordsComponents(disciplineRecordsPanel);

    // Show the discipline records frame
    disciplineRecordsFrame.setVisible(true);
}

private static void placeDisciplineRecordsComponents(JPanel disciplineRecordsPanel) {
    disciplineRecordsPanel.setLayout(new GridLayout(3, 1));

    JLabel nameLabel = new JLabel("Name:");
    JTextField nameField = new JTextField();
    JLabel disciplineLabel = new JLabel("Discipline:");
    JTextField disciplineField = new JTextField();
    JButton saveButton = new JButton("Save");

    disciplineRecordsPanel.add(nameLabel);
    disciplineRecordsPanel.add(nameField);
    disciplineRecordsPanel.add(disciplineLabel);
    disciplineRecordsPanel.add(disciplineField);
    disciplineRecordsPanel.add(saveButton);

    // ActionListener for the save button
    saveButton.addActionListener(e -> {
        String name = nameField.getText();
        String discipline = disciplineField.getText();

        // Save the discipline record to the database
        saveDisciplineRecord(name, discipline);

        // Clear the input fields
        nameField.setText("");
        disciplineField.setText("");

        JOptionPane.showMessageDialog(null, "Discipline record saved successfully");
    });
}

private static void saveDisciplineRecord(String name, String discipline) {
    try (Connection connection = DriverManager.getConnection(DB_URL)) {
        String sql = "INSERT INTO discipline_records (name, discipline) VALUES (?, ?)";
        try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {

```

```

preparedStatement.setString(1, name);
    preparedStatement.setString(2, discipline);

    preparedStatement.executeUpdate();
}

System.out.println("Discipline record saved to the database successfully");
} catch (SQLException e) {
    e.printStackTrace();
    JOptionPane.showMessageDialog(null, "Error saving discipline record to the database");
}
}

private static void uploadPdfToDatabase(File file) {
    try (Connection connection = DriverManager.getConnection(DB_URL)) {
        // Assuming you have a table named 'pdf_documents' with columns 'id' and 'file_data'
        String sql = "INSERT INTO pdf_documents (file_data) VALUES (?)";

        try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
            // Set the file data as a blob
            preparedStatement.setBytes(1, Files.readAllBytes(file.toPath()));

            // Execute the query
            preparedStatement.executeUpdate();
        }

        System.out.println("PDF uploaded to the database successfully");
    } catch (SQLException | IOException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(null, "Error uploading PDF to the database");
    }
}

private static void showUploadedPDFs() {
    JFrame pdfFrame = new JFrame("Uploaded PDFs");
    pdfFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    pdfFrame.setSize(400, 300);
    pdfFrame.setLocationRelativeTo(null);

    JPanel pdfPanel = new JPanel();
    pdfFrame.add(pdfPanel);
    pdfPanel.setLayout(new BoxLayout(pdfPanel, BoxLayout.Y_AXIS));

    // Fetch PDFs from the database
    fetchPDFsFromDatabase().forEach(pdfFile -> {
        JButton pdfButton = new JButton(pdfFile.getName());
        pdfPanel.add(pdfButton);

        // ActionListener for each PDF button
        pdfButton.addActionListener(event -> viewPdf(pdfFile));
    });

    pdfFrame.setVisible(true);
}

private static List<File> fetchPDFsFromDatabase() {
    List<File> pdfList = new ArrayList<>();

```

```

try (Connection connection = DriverManager.getConnection(DB_URL)) {
    String sql = "SELECT file_data FROM pdf_documents";
    try (PreparedStatement preparedStatement = connection.prepareStatement(sql);
        ResultSet resultSet = preparedStatement.executeQuery()) {
        while (resultSet.next()) {
            // Retrieve the PDF data as bytes
            byte[] pdfData = resultSet.getBytes("file_data");

            // Save the PDF data to a temporary file
            File tempFile = File.createTempFile("temp", ".pdf");
            try (FileOutputStream fos = new FileOutputStream(tempFile)) {
                fos.write(pdfData);
            }

            pdfList.add(tempFile);
        }
    }
} catch (SQLException | IOException e) {
    e.printStackTrace();
    JOptionPane.showMessageDialog(null, "Error fetching PDFs from the database");
}

return pdfList;
}

private static void viewPdf(File file) {
    // Open the PDF using the default system viewer
    try {
        Desktop.getDesktop().open(file);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
}

```

Code Explanation

The Main class is the entry point of the program. It contains the main method, which is the starting point of execution.

The main method creates the login page by invoking the createLoginPage method using SwingUtilities.invokeLater. This ensures that the GUI components are created and updated on the Event Dispatch Thread (EDT), which is the main thread for handling GUI events. The createLoginPage method creates a JFrame (a window) for the login page. It sets the title, size, and location of the frame. The placeComponents method is responsible for placing the GUI components (labels, text fields, buttons) on the login page.

Open Examination Records Window, place Examination Records Components, open Attendance Records Window, place Attendance Records Components, open Discipline Records Window, place Discipline Records Components are used to handle the windows and components for examination records, attendance records, and discipline records. The save Examination Record, save Attendance Record, and save Discipline Record methods save the respective records to a SQLite database using JDBC.

Packages and Libraries Used

The code imports the following packages and libraries:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.nio.file.*;
import java.sql.*;
import java.util.*;
import java.util.List;
```

javax.swing: Provides classes for creating graphical user interfaces (GUIs).

java.awt: Contains classes for creating GUI components and managing their layout.

java.awt.event: Includes classes for handling GUI events, such as button clicks.

java.io: Provides classes for input and output operations.

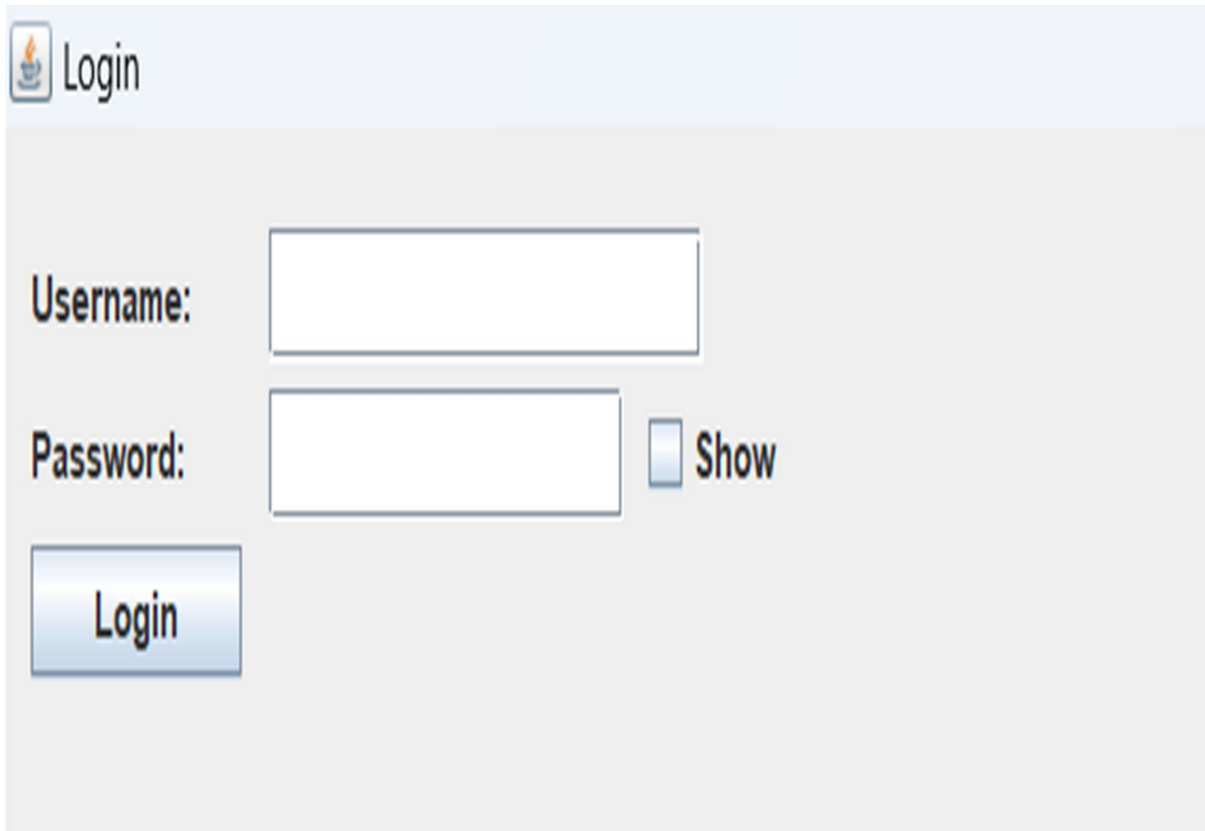
java.nio.file: Offers classes for working with files and directories.

java.sql: Contains classes for database connectivity and operations.

java.util: Provides utility classes and data structures.

java.util.List: Represents an ordered collection of elements.

5.2. Screenshots of the Application



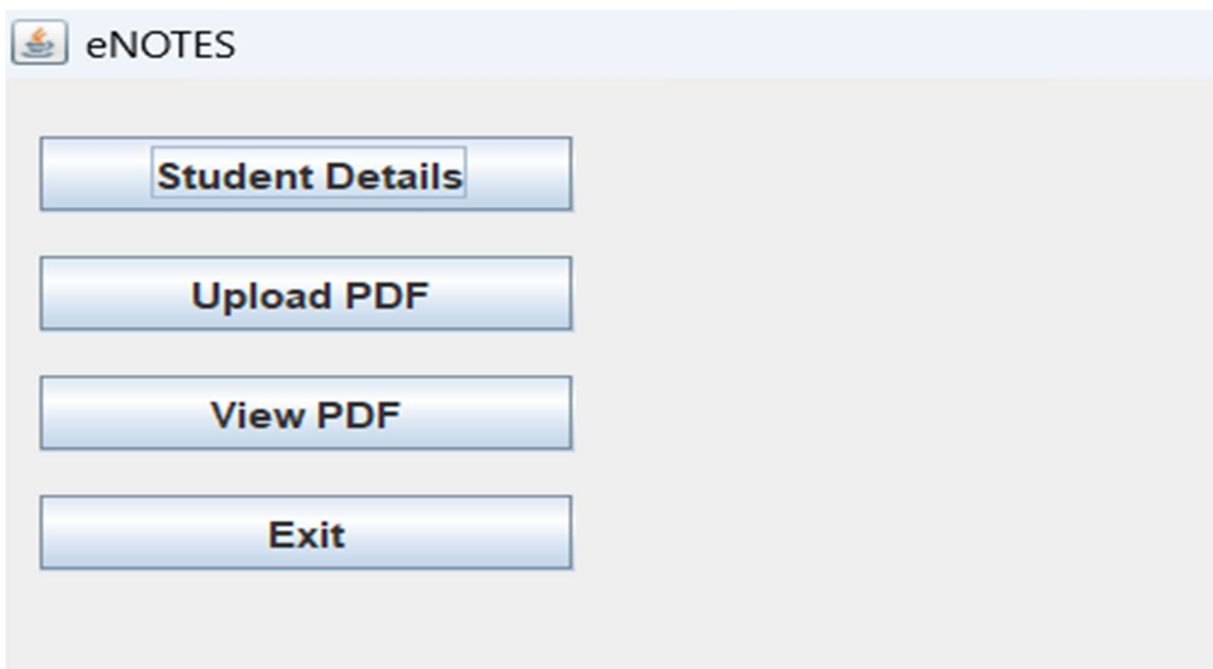
The screenshot shows a 'Login' window with a light blue header bar containing a small icon and the title 'Login'. The main area is light gray. It features two input fields: 'Username:' and 'Password:'. The 'Password:' field has a 'Show' button next to it. Below the fields is a 'Login' button.

Login

Username:

Password:


Fig 01 Login



The screenshot shows the 'eNOTES' main page with a light blue header bar containing a small icon and the title 'eNOTES'. The main area is light gray. It features four buttons stacked vertically: 'Student Details', 'Upload PDF', 'View PDF', and 'Exit'.

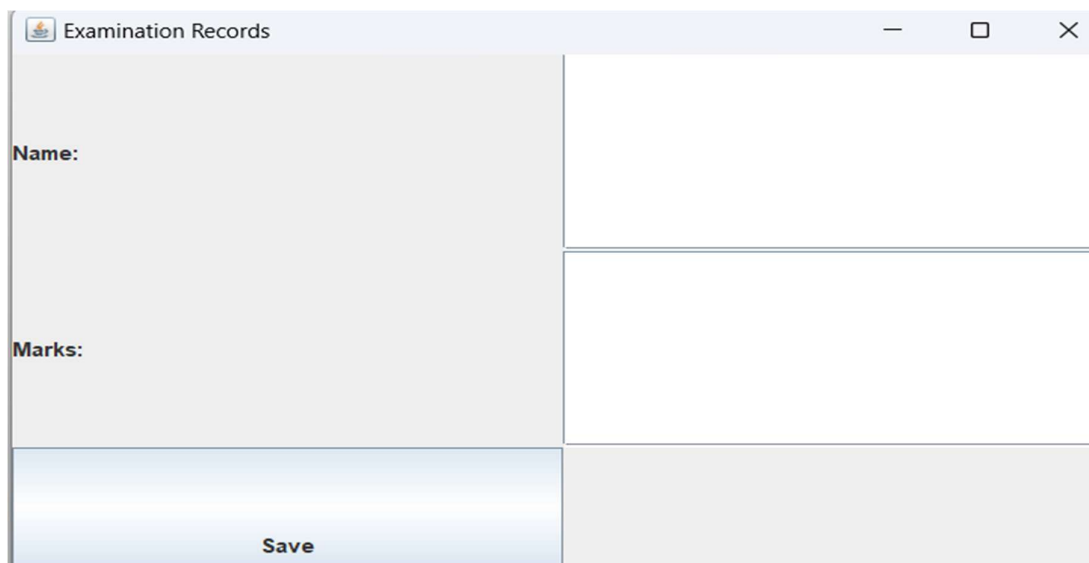
eNOTES

Fig 02 Main page



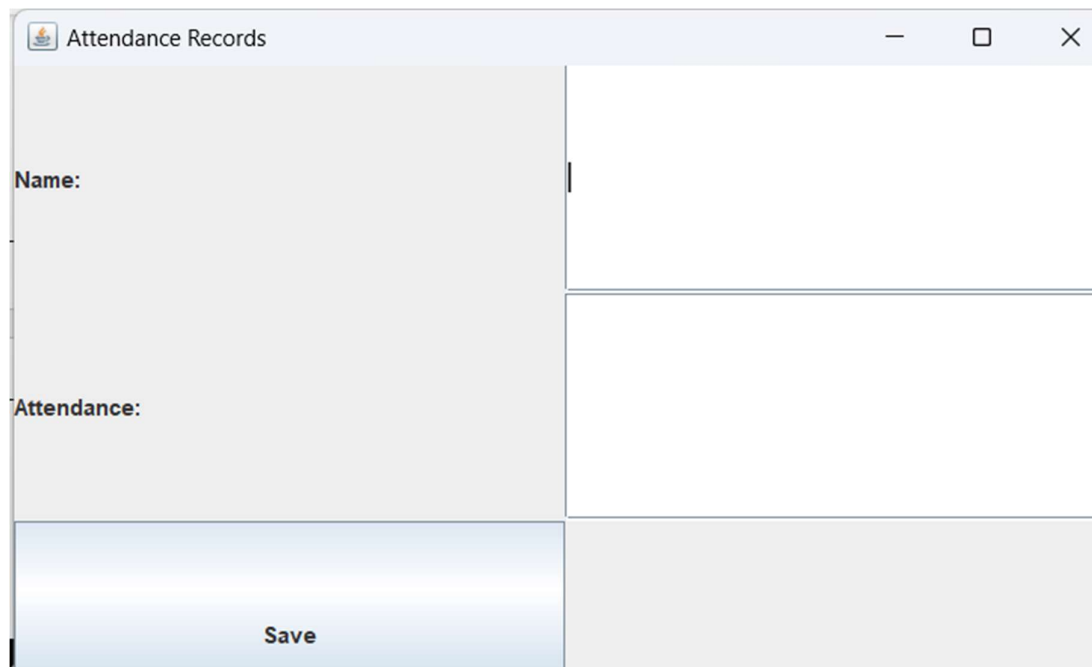
A screenshot of a software window titled "Student Details". The window has a standard title bar with a minimize button, a maximize button, and a close button. The main content area is divided into three horizontal sections, each with a label: "Examination Records" at the top, "Attendance Records" in the middle, and "Discipline Records" at the bottom. Each section is represented by a light blue rectangular box with a subtle gradient.

Fig 03 Student details



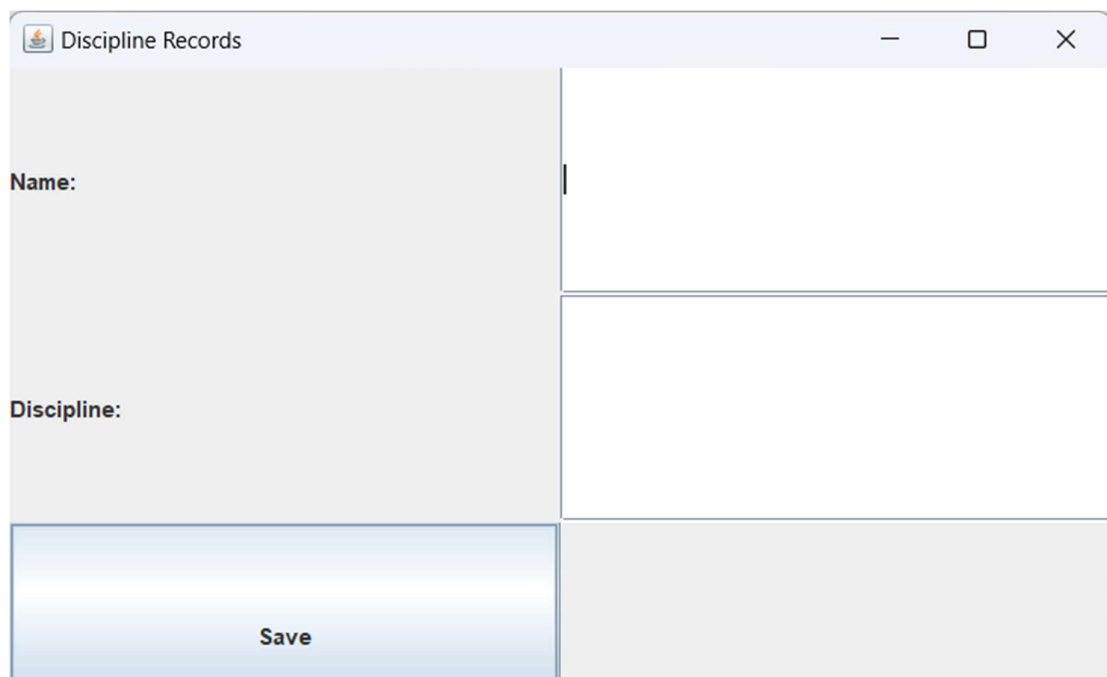
A screenshot of a software window titled "Examination Records". The window has a standard title bar with a minimize button, a maximize button, and a close button. The main content area is divided into two columns. The left column contains two labels: "Name:" and "Marks:". The right column contains two empty text input fields corresponding to the labels. At the bottom of the left column, there is a blue button labeled "Save".

Fig 04 Examination Records



A screenshot of a software window titled "Attendance Records". The window has a light blue header bar with a small icon on the left and standard window controls (minimize, maximize, close) on the right. The main area is divided into two columns. The left column has a light gray background and contains two labels: "Name:" and "Attendance:". The right column has a white background and contains two empty text input fields corresponding to the labels. At the bottom of the left column is a blue button with the text "Save".

Fig 05 Attendance Records



A screenshot of a software window titled "Discipline Records". The window has a light blue header bar with a small icon on the left and standard window controls (minimize, maximize, close) on the right. The main area is divided into two columns. The left column has a light gray background and contains two labels: "Name:" and "Discipline:". The right column has a white background and contains two empty text input fields corresponding to the labels. At the bottom of the left column is a blue button with the text "Save".

Fig 06 Discipline Records

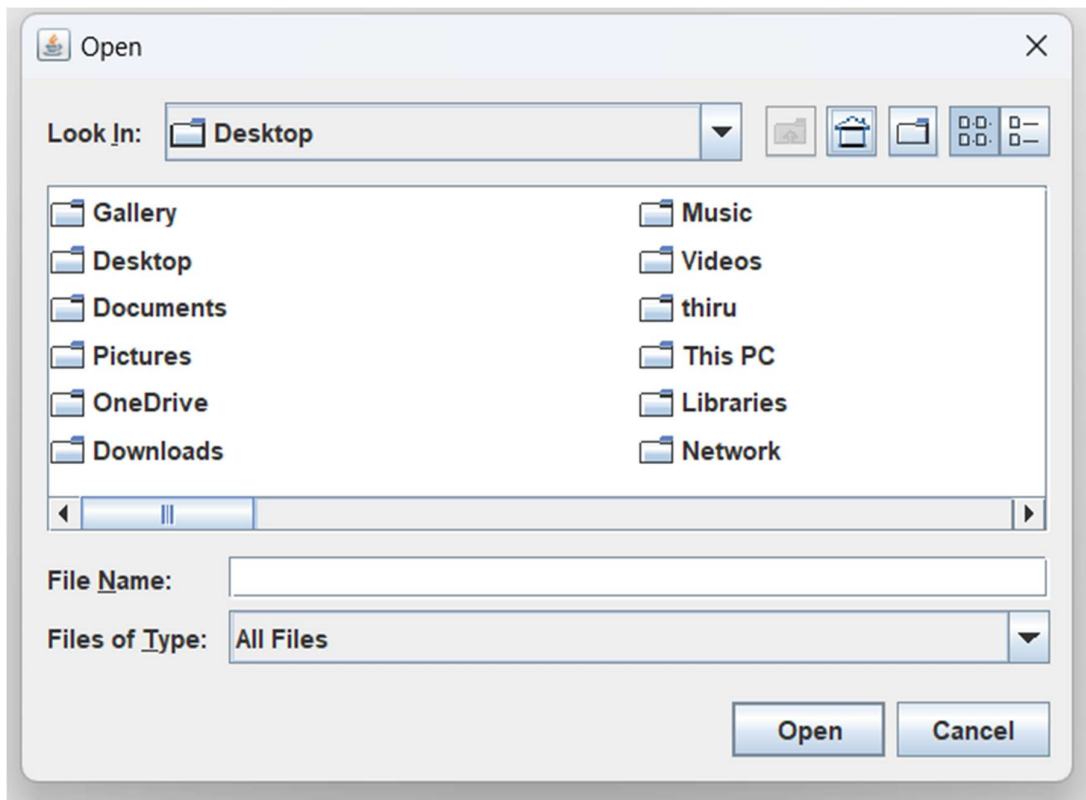


Fig 07 Pdf Uploading

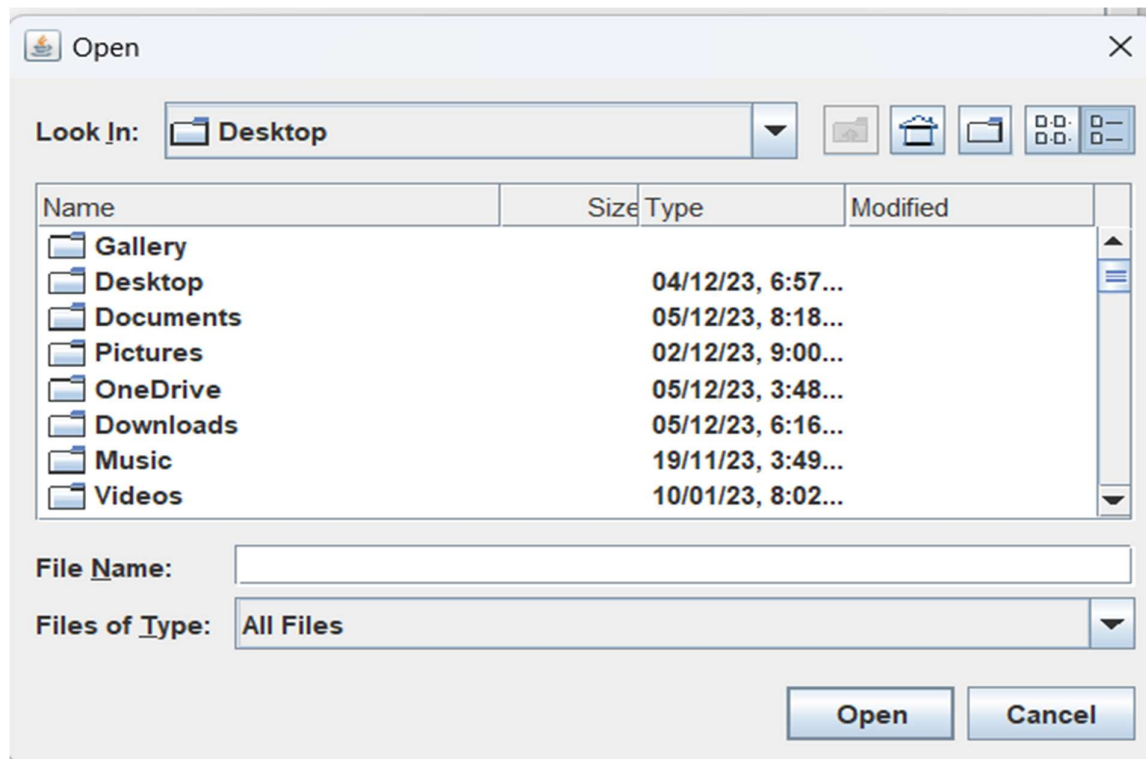


Fig 08 Pdf Uploading

Chapter 6

RESULTS & CONCLUSION

6.1. Results

some potential outcomes and benefits that can be expected from the successful implementation of the eNOTES application:

Efficient Data Management:

Centralized storage of student information, academic records, and documents leads to more efficient data management.

User-Friendly Interface:

The Java Swing-based graphical user interface provides a user-friendly experience, making it easier for administrators, teachers, and staff to navigate and perform tasks.

Improved Record Keeping:

Modules for examination records, attendance tracking, and discipline records provide a systematic and organized approach to record-keeping.

Secure Authentication:

Secure user authentication ensures that only authorized personnel can access specific features and data, enhancing overall system security.

Document Management:

The ability to upload, store, and view PDF documents streamlines document management processes.

Automated Processes:

Automation of administrative tasks reduces the time required for data entry and management, increasing overall efficiency. The integration of version control and documentation practices ensures that the application can be maintained, updated, and improved over time.

Enhanced Collaboration:

A centralized platform facilitates collaboration among different stakeholders by providing easy access to shared information.

Error Handling and Validation:

Robust error-handling mechanisms and input validation contribute to the reliability and accuracy of data.

Database Integration:

Integration with an SQLite database allows for secure and efficient storage and retrieval of information.

User Training and Support:

Provided user training materials and support documentation ensure that users can effectively utilize the features of the application.

Version Control and Maintenance:

Utilizing version control helps track changes, collaborate with multiple developers, and manage the source code for ongoing maintenance and updates.

6.2 Conclusion

In conclusion, the eNOTES application represents a comprehensive and modern solution for educational institutions seeking to enhance their management processes. Through the use of Java Swing for the graphical user interface and SQLite for database management, eNOTES offers a user-friendly platform that centralizes student information, academic records, and documents. The key features, including secure user authentication, document upload and retrieval, and efficient record-keeping modules for examinations, attendance, and discipline records, contribute to a streamlined educational management system.

The successful implementation of eNOTES is expected to bring several benefits, including improved data management, enhanced collaboration among stakeholders, and automation of administrative tasks. The application's modular architecture, robust error-handling mechanisms, and attention to user validation contribute to its reliability and usability. The integration of version control and documentation practices ensures that the application can be maintained, updated, and improved over time.

While the eNOTES application is designed to meet the specific needs of educational institutions, its effectiveness will ultimately be determined by factors such as user feedback, ongoing support, and adaptability to evolving requirements. Regular testing, user training, and proactive maintenance are crucial for ensuring the sustained success and relevance of eNOTES in the dynamic educational environment. In conclusion, FitPulse stands as a beacon of innovation in the realm of health and well-being applications, embodying a user-centric approach to proactive health management. The journey from design to implementation and the observed results underscore the application's success in delivering personalized health guidance and fostering a sense of empowerment among users.

REFERENCES

1. <https://fliphtml5.com/learning-center/free-pdf-sharing-sites-platforms/>
2. <https://www.blackboard.com/group/136>
3. <https://schoolology.wintondrivedistrict.org/login?&school=233114456>
4. https://github.com/Lubomir1998/Student_Management_System
5. <https://github.com/Sabestin/Attendance-Management-System>