



Ex. No. :

Date :

### MINI-PROJECT

## PROJECT DESCRIPTION:

The Student Record Management System is a desktop application developed using Java (with Swing components for GUI) and a relational database (SQLPlus) for data storage. Its primary function is to centralize and manage key academic and administrative information for students.

## FUNCTIONAL OVERVIEW:

The system provides Create, Read, Update, and Delete (CRUD) operations across five dedicated tables, categorized into four modules:

1. **Student Core Records:** Manages basic student data (Name, Register Number, DOB, Address).
2. **Academic History:** Stores prior educational details (11th/12th marks, family names).
3. **Student Attendance:** Tracks subject-wise attendance, separated for Year I and Year II.
4. **Student Achievements:** Records extra-curricular awards and accomplishments.

## DATABASE SCHEMA:

Table Name	Attributes (Columns)
STUDENT_RECORDS	Sno (Auto-increment), <b>REG_NO</b> (Primary Key), S_NAME, S_YEAR, DOB, HOSTELLER, PHONE_NO, ADDRESS
STUDENT_HISTORY	Sno (Auto-increment), <b>REG_NO</b> (Foreign Key), S_NAME, FATHER_NAME, MOTHER_NAME, GR_11_GROUP, GR_11_SCHOOL, GR_11_MARKS, GR_12_SCHOOL, GR_12_MARKS
STUDENT_ACHIEVEMENTS	Sno (Auto-increment), <b>REG_NO</b> (Foreign Key), S_NAME, A_YEAR, PRIZE, EVENT_NAME, EVENT_DATE, FEEDBACK, TECH_NONTECH
STUDENT_ATTENDANCE_01	Sno (Auto-increment), <b>REG_NO</b> (Foreign Key), S_NAME, AI, DBMS, DM, NDSA, OOPS, SYS_DISC, CLUB, AI_LAB, NDSA_LAB, OOP_LAB, DBMS_LAB, ESS, MENTORING, TOTAL_ATTENDANCE
STUDENT_ATTENDANCE_02	Sno (Auto-increment), <b>REG_NO</b> (Foreign Key), S_NAME, MATHS, CHEMISTRY, PYTHON, BEEE, MENTORING, TAMIL, ENGLISH, LIFESKILLS, EPL_LAB, PYTHON_LAB, CHEMISTRY_LAB, TOTAL_ATTENDANCE



## Database Implementation Script (DDL):

### 1. User Access and Privileges:

```
GRANT CREATE TABLE, DROP ANY TABLE, CREATE SEQUENCE,  
CREATE TRIGGER, UNLIMITED TABLESPACE TO AD7013;
```

```
GRANT DBA TO AD7013;
```

```
EXEC SETUP_STUDENT_SCHEMA;
```

### 2. Schema Definition (DDL Script):

```
CREATE OR REPLACE PROCEDURE SETUP_STUDENT_SCHEMA  
IS  
BEGIN  
    BEGIN  
        EXECUTE IMMEDIATE 'DROP TABLE STUDENT_ACHIEVEMENTS CASCADE  
CONSTRAINTS';  
        EXECUTE IMMEDIATE 'DROP TABLE STUDENT_ATTENDANCE_02 CASCADE  
CONSTRAINTS';  
        EXECUTE IMMEDIATE 'DROP TABLE STUDENT_ATTENDANCE_01 CASCADE  
CONSTRAINTS';  
        EXECUTE IMMEDIATE 'DROP TABLE STUDENT_HISTORY CASCADE  
CONSTRAINTS';  
        EXECUTE IMMEDIATE 'DROP TABLE STUDENT_RECORDS CASCADE  
CONSTRAINTS';  
  
        EXECUTE IMMEDIATE 'DROP SEQUENCE sr_seq';  
        EXECUTE IMMEDIATE 'DROP SEQUENCE sh_seq';  
        EXECUTE IMMEDIATE 'DROP SEQUENCE sa01_seq';  
        EXECUTE IMMEDIATE 'DROP SEQUENCE sa02_seq';  
        EXECUTE IMMEDIATE 'DROP SEQUENCE sach_seq';  
    EXCEPTION WHEN OTHERS THEN  
        NULL; -- Ignore errors if objects don't exist  
    END;  
  
    EXECUTE IMMEDIATE 'CREATE SEQUENCE sr_seq START WITH 1 INCREMENT BY 1';  
    EXECUTE IMMEDIATE 'CREATE SEQUENCE sh_seq START WITH 1 INCREMENT BY 1';  
    EXECUTE IMMEDIATE 'CREATE SEQUENCE sa01_seq START WITH 1 INCREMENT BY  
1';  
    EXECUTE IMMEDIATE 'CREATE SEQUENCE sa02_seq START WITH 1 INCREMENT BY  
1';  
    EXECUTE IMMEDIATE 'CREATE SEQUENCE sach_seq START WITH 1 INCREMENT BY  
1';
```



**-- STUDENT\_RECORDS Table (Core Student Data)**

```
EXECUTE IMMEDIATE 'CREATE TABLE STUDENT_RECORDS (  
    SNO NUMBER(10) PRIMARY KEY,  
    REG_NO VARCHAR2(12) NOT NULL UNIQUE,  
    S_NAME VARCHAR2(100) NOT NULL,  
    S_YEAR CHAR(1) NOT NULL,  
    DOB DATE NOT NULL,  
    HOSTELLER CHAR(1) NOT NULL,  
    PHONE_NO NUMBER(10) NOT NULL UNIQUE,  
    ADDRESS VARCHAR2(250) NOT NULL,  
    CONSTRAINT chk_s_year CHECK (S_YEAR IN ("1", "2", "3", "4")),  
    CONSTRAINT chk_hosteller CHECK (HOSTELLER IN ("Y", "N"))  
)';  
EXECUTE IMMEDIATE 'CREATE OR REPLACE TRIGGER sr_trigger  
BEFORE INSERT ON STUDENT_RECORDS FOR EACH ROW  
BEGIN SELECT sr_seq.NEXTVAL INTO :NEW.SNO FROM dual; END;';
```

**-- STUDENT\_HISTORY Table (Prior Academic/Family Data)**

```
EXECUTE IMMEDIATE 'CREATE TABLE STUDENT_HISTORY (  
    SNO NUMBER(10) PRIMARY KEY,  
    REG_NO VARCHAR2(12) NOT NULL,  
    S_NAME VARCHAR2(100) NOT NULL,  
    FATHER_NAME VARCHAR2(100) NOT NULL,  
    MOTHER_NAME VARCHAR2(100) NOT NULL,  
    GR_11_GROUP VARCHAR2(50),  
    GR_11_SCHOOL VARCHAR2(100),  
    GR_11_MARKS NUMBER(5, 2) CHECK (GR_11_MARKS <= 100),  
    GR_12_SCHOOL VARCHAR2(100),  
    GR_12_MARKS NUMBER(5, 2) CHECK (GR_12_MARKS <= 100),  
    CONSTRAINT fk_history_regno FOREIGN KEY (REG_NO) REFERENCES  
STUDENT_RECORDS (REG_NO)  
)';  
EXECUTE IMMEDIATE 'CREATE OR REPLACE TRIGGER sh_trigger  
BEFORE INSERT ON STUDENT_HISTORY FOR EACH ROW  
BEGIN SELECT sh_seq.NEXTVAL INTO :NEW.SNO FROM dual; END;';
```

**-- STUDENT\_ATTENDANCE\_01 Table (Year 1 Attendance)**

```
EXECUTE IMMEDIATE 'CREATE TABLE STUDENT_ATTENDANCE_01 (  
    SNO NUMBER(10) PRIMARY KEY, REG_NO VARCHAR2(12) NOT NULL UNIQUE,  
    S_NAME VARCHAR2(100) NOT NULL,  
    AI NUMBER(5, 2) CHECK (AI <= 100), DBMS NUMBER(5, 2) CHECK (DBMS <= 100),  
    DM NUMBER(5, 2) CHECK (DM <= 100), NDSA NUMBER(5, 2) CHECK (NDSA <= 100),  
    OOPS NUMBER(5, 2) CHECK (OOPS <= 100), SYS_DISC NUMBER(5, 2) CHECK  
(SYS_DISC <= 100), CLUB NUMBER(5, 2) CHECK (CLUB <= 100),  
    AI_LAB NUMBER(5, 2) CHECK (AI_LAB <= 100), NDSA_LAB NUMBER(5, 2) CHECK  
(NDSA_LAB <= 100), OOP_LAB NUMBER(5, 2) CHECK (OOP_LAB <= 100),  
    DBMS_LAB NUMBER(5, 2) CHECK (DBMS_LAB <= 100), ESS NUMBER(5, 2) CHECK  
(ESS <= 100), MENTORING NUMBER(5, 2) CHECK (MENTORING <= 100),  
    TOTAL_ATTENDANCE NUMBER(5, 2) NOT NULL CHECK (TOTAL_ATTENDANCE  
<= 100),
```



```
CONSTRAINT fk_att01_regno FOREIGN KEY (REG_NO) REFERENCES  
STUDENT_RECORDS (REG_NO) );
```

```
EXECUTE IMMEDIATE 'CREATE OR REPLACE TRIGGER sa01_trigger BEFORE INSERT  
ON STUDENT_ATTENDANCE_01 FOR EACH ROW BEGIN SELECT sa01_seq.NEXTVAL  
INTO :NEW.SNO FROM dual; END;';
```

#### -- STUDENT\_ATTENDANCE\_02 Table (Year 2 Attendance)

```
EXECUTE IMMEDIATE 'CREATE TABLE STUDENT_ATTENDANCE_02 (  
SNO NUMBER(10) PRIMARY KEY, REG_NO VARCHAR2(12) NOT NULL UNIQUE,  
S_NAME VARCHAR2(100) NOT NULL,  
MATHS NUMBER(5, 2) CHECK (MATHS <= 100), CHEMISTRY NUMBER(5, 2) CHECK  
(CHEMISTRY <= 100), PYTHON NUMBER(5, 2) CHECK (PYTHON <= 100),  
BEEE NUMBER(5, 2) CHECK (BEEE <= 100), MENTORING NUMBER(5, 2) CHECK  
(MENTORING <= 100), TAMIL NUMBER(5, 2) CHECK (TAMIL <= 100),  
ENGLISH NUMBER(5, 2) CHECK (ENGLISH <= 100), LIFESKILLS NUMBER(5, 2)  
CHECK (LIFESKILLS <= 100), EPL_LAB NUMBER(5, 2) CHECK (EPL_LAB <= 100),  
PYTHON_LAB NUMBER(5, 2) CHECK (PYTHON_LAB <= 100), CHEMISTRY_LAB  
NUMBER(5, 2) CHECK (CHEMISTRY_LAB <= 100),  
TOTAL_ATTENDANCE NUMBER(5, 2) NOT NULL CHECK (TOTAL_ATTENDANCE  
<= 100),  
CONSTRAINT fk_att02_regno FOREIGN KEY (REG_NO) REFERENCES  
STUDENT_RECORDS (REG_NO)  
);  
EXECUTE IMMEDIATE 'CREATE OR REPLACE TRIGGER sa02_trigger BEFORE INSERT  
ON STUDENT_ATTENDANCE_02 FOR EACH ROW BEGIN SELECT sa02_seq.NEXTVAL  
INTO :NEW.SNO FROM dual; END;';
```

#### -- STUDENT\_ACHIEVEMENTS Table (Extracurricular Data)

```
EXECUTE IMMEDIATE 'CREATE TABLE STUDENT_ACHIEVEMENTS (  
SNO NUMBER(10) PRIMARY KEY, REG_NO VARCHAR2(12) NOT NULL, S_NAME  
VARCHAR2(100) NOT NULL,  
A_YEAR VARCHAR2(10) NOT NULL, PRIZE VARCHAR2(50), EVENT_NAME  
VARCHAR2(100) NOT NULL,  
EVENT_DATE DATE NOT NULL, FEEDBACK VARCHAR2(150), TECH_NONTECH  
CHAR(1) NOT NULL,  
CONSTRAINT chk_tnt CHECK (TECH_NONTECH IN ("Y", "N")),  
CONSTRAINT fk_ach_regno FOREIGN KEY (REG_NO) REFERENCES  
STUDENT_RECORDS (REG_NO)  
);  
EXECUTE IMMEDIATE 'CREATE OR REPLACE TRIGGER sach_trigger BEFORE INSERT  
ON STUDENT_ACHIEVEMENTS FOR EACH ROW BEGIN SELECT sach_seq.NEXTVAL  
INTO :NEW.SNO FROM dual; END;';
```

```
END;  
/
```

```
-- Execution command to run the schema setup  
EXEC SETUP_STUDENT_SCHEMA;
```



### 3. Test Data Population:

```
CREATE OR REPLACE PROCEDURE LOAD_TEST_DATA
IS
BEGIN
```

```
EXECUTE IMMEDIATE 'TRUNCATE TABLE STUDENT_ACHIEVEMENTS';
EXECUTE IMMEDIATE 'TRUNCATE TABLE STUDENT_ATTENDANCE_02';
EXECUTE IMMEDIATE 'TRUNCATE TABLE STUDENT_ATTENDANCE_01';
EXECUTE IMMEDIATE 'TRUNCATE TABLE STUDENT_HISTORY';
EXECUTE IMMEDIATE 'TRUNCATE TABLE STUDENT_RECORDS';
```

#### -- A. STUDENT\_RECORDS

```
INSERT INTO STUDENT_RECORDS (REG_NO, S_NAME, S_YEAR, DOB, HOSTELLER,
PHONE_NO, ADDRESS) VALUES ('311124243001', 'Arjun Kumar', '1', TO_DATE('2006-05-15',
'YYYY-MM-DD'), 'Y', 9876543210, 'Hostel Block A, Chennai');
```

```
INSERT INTO STUDENT_RECORDS (REG_NO, S_NAME, S_YEAR, DOB, HOSTELLER,
PHONE_NO, ADDRESS) VALUES ('311124243002', 'Priya Sharma', '1', TO_DATE('2005-11-22',
'YYYY-MM-DD'), 'N', 9988776655, '12/A Gandhipuram, Coimbatore');
```

-- ... (23 more STUDENT\_RECORDS inserts) ...

#### -- B. STUDENT\_HISTORY

```
INSERT INTO STUDENT_HISTORY (REG_NO, S_NAME, FATHER_NAME,
MOTHER_NAME, GR_11_GROUP, GR_11_SCHOOL, GR_11_MARKS, GR_12_SCHOOL,
GR_12_MARKS) VALUES ('311124243001', 'Arjun Kumar', 'Rajesh Kumar', 'Meena Kumar',
'Maths-Biology', 'HSS School 1', 92.5, 'HSS School 1', 94.0);
```

```
INSERT INTO STUDENT_HISTORY (REG_NO, S_NAME, FATHER_NAME,
MOTHER_NAME, GR_11_GROUP, GR_11_SCHOOL, GR_11_MARKS, GR_12_SCHOOL,
GR_12_MARKS) VALUES ('311124243002', 'Priya Sharma', 'Anil Sharma', 'Geeta Sharma',
'Computer Science', 'Ideal School', 88.0, 'Ideal School', 91.5);
```

-- ... (23 more STUDENT\_HISTORY inserts) ...

#### -- C. STUDENT\_ATTENDANCE\_01 (Year 1)

```
INSERT INTO STUDENT_ATTENDANCE_01 (REG_NO, S_NAME, AI, DBMS, DM, NDSA,
OOPS, SYS_DISC, CLUB, AI_LAB, NDSA_LAB, OOP_LAB, DBMS_LAB, ESS, MENTORING,
TOTAL_ATTENDANCE) VALUES
```

```
('311124243001', 'Arjun Kumar', 75.0, 88.5, 90.0, 78.0, 85.5, 92.0, 70.0, 80.0, 82.0, 79.5, 91.0,
95.0, 80.0, 84.0);
```

-- ... (4 more STUDENT\_ATTENDANCE\_01 inserts) ...

#### -- D. STUDENT\_ATTENDANCE\_02 (Year 2)

```
INSERT INTO STUDENT_ATTENDANCE_02 (REG_NO, S_NAME, MATHS, CHEMISTRY,
PYTHON, BEEE, MENTORING, TAMIL, ENGLISH, LIFESKILLS, EPL_LAB, PYTHON_LAB,
CHEMISTRY_LAB, TOTAL_ATTENDANCE) VALUES
```

```
('311125243006', 'Meena Krishnan', 85.0, 88.0, 92.0, 80.0, 85.0, 95.0, 90.0, 88.0, 93.0, 90.0, 86.0,
88.8);
```

-- ... (4 more STUDENT\_ATTENDANCE\_02 inserts) ...

#### -- E. STUDENT\_ACHIEVEMENTS

```
INSERT INTO STUDENT_ACHIEVEMENTS (REG_NO, S_NAME, A_YEAR, PRIZE,
EVENT_NAME, EVENT_DATE, FEEDBACK, TECH_NONTECH) VALUES
```

```
('311124243001', 'Arjun Kumar', '2024-25', 'First Place', 'AI Hackathon', TO_DATE('2025-01-20',
'YYYY-MM-DD'), 'Developed a functional chatbot model.', 'Y');
```

```
INSERT INTO STUDENT_ACHIEVEMENTS (REG_NO, S_NAME, A_YEAR, PRIZE,
EVENT_NAME, EVENT_DATE, FEEDBACK, TECH_NONTECH) VALUES
```



```
('311124243002', 'Priya Sharma', '2024-25', 'Second Place', 'Essay Writing Contest',  
TO_DATE('2025-02-15', 'YYYY-MM-DD'), 'Excellent descriptive writing skills.', 'N');  
-- ... (23 more STUDENT_ACHIEVEMENTS inserts) ...
```

```
COMMIT;
```

```
END;
```

```
/
```

```
-- Execution command to load the test data
```

```
EXEC LOAD_TEST_DATA;
```

## Application Source Code:

### 1. DBUtil.java (Database Utility):

```
import java.sql.*;  
import java.util.ArrayList;  
import java.util.List;  
  
public class DBUtil {  
  
    // --- CONFIGURATION ---  
    private static final String DB_URL = "jdbc:oracle:thin:@192.168.1.18:1521:orcl";  
    private static final String DB_USER = "ad7013";  
    private static final String DB_PASSWORD = "licet";  
  
    public static Connection createDbConnection() {  
        Connection con = null;  
        try {  
            // Load the Oracle Driver  
            Class.forName("oracle.jdbc.driver.OracleDriver");  
            con = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);  
            con.setAutoCommit(false); // Enable manual transaction control  
            System.out.println("Database Connection Successful.");  
        } catch (ClassNotFoundException e) {  
            System.err.println("JDBC Driver not found: " + e.getMessage());  
        } catch (SQLException e) {  
            System.err.println("Database Connection Failed: " + e.getMessage());  
        }  
        return con;  
    }  
  
    public static void closeDbConnection(Connection con) {  
        if (con != null) {  
            try {  
                con.close();  
                System.out.println("Database Connection Closed.");  
            } catch (SQLException e) {  
                System.err.println("Error closing connection: " + e.getMessage());  
            }  
        }  
    }  
}
```



```
}  
}  
}  
  
/**  
 * Executes a SELECT query.  
 * @return A List of String arrays (data) and a String array (column names).  
 */  
public static Object[] fetchAllData(Connection con, String tableName) {  
    String query = "SELECT * FROM " + tableName;  
    List<String[]> data = new ArrayList<>();  
    String[] columns = null;  
  
    try (Statement stmt = con.createStatement();  
         ResultSet rs = stmt.executeQuery(query)) {  
  
        ResultSetMetaData rsmd = rs.getMetaData();  
        int columnCount = rsmd.getColumnCount();  
        columns = new String[columnCount];  
  
        // Get column names  
        for (int i = 1; i <= columnCount; i++) {  
            columns[i - 1] = rsmd.getColumnName(i);  
        }  
  
        // Get data rows  
        while (rs.next()) {  
            String[] row = new String[columnCount];  
            for (int i = 1; i <= columnCount; i++) {  
                // Get data as String for simple GUI display  
                row[i - 1] = rs.getString(i);  
            }  
            data.add(row);  
        }  
  
    } catch (SQLException e) {  
        System.err.println("Error fetching data from " + tableName + ": " + e.getMessage());  
        return new Object[]{new ArrayList<String[]>(), null}; // Return empty on error  
    }  
  
    return new Object[]{data, columns};  
}  
  
/**  
 * Executes INSERT, UPDATE, or DELETE query.  
 * @param params The parameters for the PreparedStatement.  
 * @return The number of rows affected, or -1 on error.  
 */  
public static int executeUpdate(Connection con, String query, Object[] params) {  
    try (PreparedStatement pstmt = con.prepareStatement(query)) {  
        for (int i = 0; i < params.length; i++) {  
            pstmt.setObject(i + 1, params[i]);  
        }  
    }  
}
```





```
    }
    int rows = pstmt.executeUpdate();
    con.commit(); // Commit the transaction on success
    return rows;
} catch (SQLException e) {
    System.err.println("Error executing update query: " + e.getMessage());
    try {
        if (con != null) con.rollback(); // Rollback on error
    } catch (SQLException rollbackEx) {
        System.err.println("Rollback failed: " + rollbackEx.getMessage());
    }
    return -1;
}
}

/**
 * Executes a SELECT query with parameters to fetch a single record.
 */
public static Object[] fetchRecord(Connection con, String tableName, String regNo) {
    String query = "SELECT * FROM " + tableName + " WHERE REG_NO = ?";
    String[] recordData = null;
    String[] columns = null;

    try (PreparedStatement pstmt = con.prepareStatement(query)) {
        pstmt.setString(1, regNo);
        try (ResultSet rs = pstmt.executeQuery()) {
            if (rs.next()) {
                ResultSetMetaData rsmd = rs.getMetaData();
                int columnCount = rsmd.getColumnCount();
                columns = new String[columnCount];
                recordData = new String[columnCount];

                for (int i = 1; i <= columnCount; i++) {
                    columns[i - 1] = rsmd.getColumnName(i);
                    recordData[i - 1] = rs.getString(i);
                }
            }
        }
    } catch (SQLException e) {
        System.err.println("Error fetching single record: " + e.getMessage());
    }
    return new Object[] {recordData, columns};
}
```





## 2. java (Reusable GUI Component):

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionListener;
import java.util.HashMap;
import java.util.Map;

public class MenuPanel extends JPanel {
    private final Map<String, JButton> buttons = new HashMap<>();

    public MenuPanel(String title) {
        setLayout(new BorderLayout(this, BorderLayout.Y_AXIS));
        setBorder(BorderFactory.createEmptyBorder(50, 50, 50, 50));

        JLabel titleLabel = new JLabel(title);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 20));
        titleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
        add(titleLabel);
        add(Box.createRigidArea(new Dimension(0, 30)));

        String[] options = {"VIEW", "ADD RECORD", "EDIT RECORD", "BACK"};

        for (String text : options) {
            JButton button = new JButton(text);
            button.setFont(new Font("Arial", Font.PLAIN, 16));
            button.setPreferredSize(new Dimension(200, 40));
            button.setMaximumSize(new Dimension(300, 40));
            button.setAlignmentX(Component.CENTER_ALIGNMENT);

            buttons.put(text, button);
            add(button);
            add(Box.createRigidArea(new Dimension(0, 15)));
        }
    }

    public void setButtonAction(String buttonText, ActionListener listener) {
        JButton button = buttons.get(buttonText);
        if (button != null) {
            // Remove previous listeners and add the new one
            for (ActionListener al : button.getActionListeners()) {
                button.removeActionListener(al);
            }
            button.addActionListener(listener);
        }
    }
}
```

## 3. StudentRecordManager.java (Main Application):

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
```



```
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.sql.Connection;
import java.text.SimpleDateFormat;
import javax.swing.table.DefaultTableModel;
import javax.swing.border.EmptyBorder;

public class StudentRecordManager extends JFrame {

    private final CardLayout cardLayout = new CardLayout();
    private final JPanel mainPanel = new JPanel(cardLayout);
    private final Connection dbConn;

    // Table definitions for dynamic form generation
    private final Map<String, Object[]> tableFields = new HashMap<>();

    public StudentRecordManager() {
        super("AI & DS STUDENT RECORD MANAGER");

        // 1. Database Connection
        dbConn = DBUtil.createDbConnection();
        if (dbConn == null) {
            JOptionPane.showMessageDialog(this, "Failed to connect to the database. Application will exit.", "DB
Error", JOptionPane.ERROR_MESSAGE);
            System.exit(1);
        }

        // 2. Frame Setup
        setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
        setSize(800, 600);
        setResizable(false); // Do not allow maximize
        setLocationRelativeTo(null);
        add(mainPanel);

        // 3. Initialize Fields Map (used by RecordFormPanel)
        initializeTableFields();

        // 4. Create Panels (GUI components)
        createOpeningInterface();
        createAllMenuFrames();

        // 5. Initial Display
        cardLayout.show(mainPanel, "OpeningInterface");
        setVisible(true);

        // 6. Window Close Handler (for Case 1.4 equivalent - exit)
        addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                DBUtil.closeDbConnection(dbConn);
                dispose();
            }
        });
    }

    // --- Helper Methods ---

    private void initializeTableFields() {
        // [COL_NAME, Label Text, Type (text, int, float, date, option), [options_list], isEditableInEditMode]
        tableFields.put("STUDENT_RECORDS", new Object[][] {
            {"REG_NO", "Register Number:", "text", null, false},
            {"S_NAME", "Name:", "text", null, true},
        }
    }
}
```



```
{
    {"S_YEAR", "Year (1-4):", "option", new String[]{"1", "2", "3", "4"}, true},
    {"DOB", "DOB (YYYY-MM-DD):", "date", null, true},
    {"HOSTELLER", "Hosteller (Y/N):", "option", new String[]{"Y", "N"}, true},
    {"PHONE_NO", "Phone Number:", "int", null, true},
    {"ADDRESS", "Address (Hint: Enter 'Hostel' if Y):", "text", null, true}
});

tableFields.put("STUDENT_HISTORY", new Object[][] {
    {"REG_NO", "Register Number:", "text", null, false},
    {"S_NAME", "Name:", "text", null, true},
    {"FATHER_NAME", "Father Name:", "text", null, true},
    {"MOTHER_NAME", "Mother Name:", "text", null, true},
    {"GR_11_GROUP", "11th Group Studied:", "text", null, true},
    {"GR_11_SCHOOL", "11th School:", "text", null, true},
    {"GR_11_MARKS", "11th Marks (%)ate", null, true},
    {"GR_12_SCHOOL", "12th School:", "text", null, true},
    {"GR_12_MARKS", "12th Marks (%)ate", null, true}
});

// ATTENDANCE tables have dynamic fields (handled in getFormFieldsForAttendance)
// ACHIEVEMENTS
tableFields.put("STUDENT_ACHIEVEMENTS", new Object[][] {
    {"REG_NO", "Register Number:", "text", null, false},
    {"S_NAME", "Name:", "text", null, true},
    {"A_YEAR", "Academic Year:", "text", null, true},
    {"PRIZE", "Prize:", "text", null, true},
    {"EVENT_NAME", "Event Name:", "text", null, true},
    {"EVENT_DATE", "Event Date (YYYY-MM-DD):", "date", null, true},
    {"FEEDBACK", "Feedback (Max 150 chars):", "text", null, true},
    {"TECH_NONTECH", "Tech/Non-Tech (Y/N):", "option", new String[]{"Y", "N"}, true}
});
}

// --- GUI Creation Methods ---

private void createOpeningInterface() {
    JPanel panel = new JPanel();
    panel.setLayout(new BorderLayout(panel, BorderLayout.Y_AXIS));
    panel.setBorder(new EmptyBorder(80, 0, 0, 0)); // Top padding

    JLabel title = new JLabel("AI & DS STUDENT RECORD MANAGER");
    title.setFont(new Font("Arial", Font.BOLD, 24));
    title.setAlignmentX(Component.CENTER_ALIGNMENT);
    panel.add(title);
    panel.add(Box.createRigidArea(new Dimension(0, 50)));

    String[][] options = {
        {"View Students records", "RecordsMenu"},
        {"View student history", "HistoryMenu"},
        {"View Student Attendance", "AttendanceYearSelect"},
        {"View student Achievements", "AchievementsMenu"}
    };

    for (String[] option : options) {
        JButton button = new JButton(option[0]);
        button.setFont(new Font("Arial", Font.PLAIN, 16));
        button.setPreferredSize(new Dimension(300, 40));
        button.setMaximumSize(new Dimension(300, 40));
        button.setAlignmentX(Component.CENTER_ALIGNMENT);
        button.addActionListener(e -> showFrame(option[1]));
        panel.add(button);
        panel.add(Box.createRigidArea(new Dimension(0, 15)));
    }
}
```



```
mainPanel.add(panel, "OpeningInterface");
}

private void createAllMenuFrames() {
    // Case 1: STUDENT_RECORDS Menu
    createMenuFrame("RecordsMenu", "STUDENT RECORDS", "STUDENT_RECORDS",
"RecordsMenu");
    // Case 2: STUDENT_HISTORY Menu
    createMenuFrame("HistoryMenu", "STUDENT HISTORY", "STUDENT_HISTORY", "HistoryMenu");
    // Case 4: STUDENT_ACHIEVEMENTS Menu
    createMenuFrame("AchievementsMenu", "STUDENT ACHIEVEMENTS",
"STUDENT_ACHIEVEMENTS", "AchievementsMenu");

    // Case 3: Attendance Year Selection
    createAttendanceYearSelection();

    // Case 3.1: Attendance Year I Menu
    createMenuFrame("Year1Menu", "ATTENDANCE - YEAR I", "STUDENT_ATTENDANCE_01",
"Year1Menu", "AttendanceYearSelect");
    // Case 3.2: Attendance Year II Menu
    createMenuFrame("Year2Menu", "ATTENDANCE - YEAR II", "STUDENT_ATTENDANCE_02",
"Year2Menu", "AttendanceYearSelect");
}

private void createMenuFrame(String frameKey, String title, String tableName, String menuFrameKey) {
    // Default back button goes to Opening Interface
    createMenuFrame(frameKey, title, tableName, menuFrameKey, "OpeningInterface");
}

private void createMenuFrame(String frameKey, String title, String tableName, String menuFrameKey, String
backFrame) {
    MenuPanel panel = new MenuPanel(title);

    // Case X.1: VIEW
    panel.setButtonAction("VIEW", e -> showDataView(tableName, title, menuFrameKey));
    // Case X.2: ADD RECORD
    panel.setButtonAction("ADD RECORD", e -> showRecordForm("ADD", tableName, menuFrameKey,
null));
    // Case X.3: EDIT RECORD
    panel.setButtonAction("EDIT RECORD", e -> showEditPrompt(tableName, menuFrameKey));
    // Case X.4: BACK
    panel.setButtonAction("BACK", e -> showFrame(backFrame));

    mainPanel.add(panel, frameKey);
}

private void createAttendanceYearSelection() {
    JPanel panel = new JPanel();
    panel.setLayout(new BorderLayout(panel, BorderLayout.Y_AXIS));
    panel.setBorder(new EmptyBorder(80, 0, 0, 0));

    JLabel title = new JLabel("VIEW STUDENT ATTENDANCE");
    title.setFont(new Font("Arial", Font.BOLD, 24));
    title.setAlignmentX(Component.CENTER_ALIGNMENT);
    panel.add(title);
    panel.add(Box.createRigidArea(new Dimension(0, 50)));

    String[][] options = {
        {"Year I", "Year1Menu"},
        {"Year II", "Year2Menu"}
    };

    for (String[] option : options) {
        JButton button = new JButton(option[0]);
```



```
button.setFont(new Font("Arial", Font.PLAIN, 16));
button.setPreferredSize(new Dimension(300, 40));
button.setMaximumSize(new Dimension(300, 40));
button.setAlignmentX(Component.CENTER_ALIGNMENT);
button.addActionListener(e -> showFrame(option[1]));
panel.add(button);
panel.add(Box.createRigidArea(new Dimension(0, 15)));
}

JButton backButton = new JButton("BACK");
backButton.setFont(new Font("Arial", Font.PLAIN, 16));
backButton.setPreferredSize(new Dimension(300, 40));
backButton.setMaximumSize(new Dimension(300, 40));
backButton.setAlignmentX(Component.CENTER_ALIGNMENT);
backButton.addActionListener(e -> showFrame("OpeningInterface"));
panel.add(Box.createRigidArea(new Dimension(0, 30)));
panel.add(backButton);

mainPanel.add(panel, "AttendanceYearSelect");
}

// --- Core Navigation and Display ---

private void showFrame(String name) {
    cardLayout.show(mainPanel, name);
}

private void showDataView(String tableName, String title, String backFrame) {
    String frameKey = tableName + "DataView";
    if (mainPanel.getComponent(frameKey) == null) {
        DataViewPanel panel = new DataViewPanel(this, tableName, title, backFrame);
        mainPanel.add(panel, frameKey);
    }
    ((DataViewPanel) mainPanel.getComponent(frameKey)).updateView(); // Update data before showing
    showFrame(frameKey);
}

private void showRecordForm(String mode, String tableName, String backFrame, String[] existingData) {
    String frameKey = tableName + "Form";
    RecordFormPanel formPanel;

    // Reuse the form panel if it exists
    if (mainPanel.getComponent(frameKey) == null) {
        formPanel = new RecordFormPanel(this, tableName, backFrame);
        mainPanel.add(formPanel, frameKey);
    } else {
        formPanel = (RecordFormPanel) mainPanel.getComponent(frameKey);
        formPanel.setBackFrame(backFrame);
    }

    formPanel.loadForm(mode, existingData);
    showFrame(frameKey);
}

private void showEditPrompt(String tableName, String backFrame) {
    String regNo = JOptionPane.showInputDialog(this, "Enter Register Number for " + tableName + ":", "Edit
Record", JOptionPane.QUESTION_MESSAGE);

    if (regNo != null && !regNo.trim().isEmpty()) {
        Object[] result = DBUtil.fetchRecord(dbConn, tableName, regNo.trim());
        String[] existingData = (String[]) result[0];

        if (existingData != null) {
            showRecordForm("EDIT", tableName, backFrame, existingData);
        }
    }
}
```



```
        } else {
            JOptionPane.showMessageDialog(this, "No record found for Register Number: " + regNo, "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }
}

// --- Data View Panel Implementation (Case X.1) ---

class DataViewPanel extends JPanel {
    private final StudentRecordManager controller;
    private String tableName;
    private String titleText;
    private String backFrame;
    private final JTable table;
    private final JLabel titleLabel;

    public DataViewPanel(StudentRecordManager controller, String tableName, String title, String
backFrame) {
        this.controller = controller;
        this.tableName = tableName;
        this.titleText = title;
        this.backFrame = backFrame;

        setLayout(new BorderLayout());

        titleLabel = new JLabel("VIEW: " + titleText, SwingConstants.CENTER);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 20));
        add(titleLabel, BorderLayout.NORTH);

        table = new JTable();
        JScrollPane scrollPane = new JScrollPane(table);
        add(scrollPane, BorderLayout.CENTER);

        JButton backButton = new JButton("BACK");
        backButton.setFont(new Font("Arial", Font.PLAIN, 16));
        backButton.addActionListener(e -> controller.showFrame(this.backFrame));
        add(backButton, BorderLayout.SOUTH);
    }

    public void updateView() {
        titleLabel.setText("VIEW: " + titleText);

        Object[] result = DBUtil.fetchAllData(controller.dbConn, tableName);
        List<String[]> data = (List<String[]>) result[0];
        String[] columns = (String[]) result[1];

        if (columns == null || data.isEmpty()) {
            titleLabel.setText("VIEW: " + titleText + " (No Records Found)");
            table.setModel(new DefaultTableModel()); // Clear table
            return;
        }

        // Convert List<String[]> to Object[][] for DefaultTableModel
        Object[][] dataArray = new Object[data.size()][columns.length];
        for (int i = 0; i < data.size(); i++) {
            dataArray[i] = data.get(i);
        }

        DefaultTableModel model = new DefaultTableModel(dataArray, columns);
        table.setModel(model);
    }
}
```



// --- Record Form Panel Implementation (Case X.2, X.3) ---

```
class RecordFormPanel extends JPanel {
    private final StudentRecordManager controller;
    private String tableName;
    private String backFrame;
    private String mode; // ADD or EDIT
    private String existingRegNo;
    private final Map<String, JComponent> inputFields = new HashMap<>();
    private final JLabel titleLabel;
    private final JPanel formPanel; // The panel holding all the labels and fields
    private final JButton confirmButton;

    public RecordFormPanel(StudentRecordManager controller, String tableName, String backFrame) {
        this.controller = controller;
        this.tableName = tableName;
        this.backFrame = backFrame;
        this.setLayout(new BorderLayout());

        titleLabel = new JLabel("", SwingConstants.CENTER);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 20));
        this.add(titleLabel, BorderLayout.NORTH);

        formPanel = new JPanel(new GridBagLayout());
        JScrollPane scrollPane = new JScrollPane(formPanel);
        scrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
        this.add(scrollPane, BorderLayout.CENTER);

        JPanel buttonPanel = new JPanel();
        confirmButton = new JButton("CONFIRM");
        confirmButton.addActionListener(e -> confirmRecord());
        buttonPanel.add(confirmButton);

        JButton backButton = new JButton("BACK");
        backButton.addActionListener(e -> controller.showFrame(this.backFrame));
        buttonPanel.add(backButton);
        this.add(buttonPanel, BorderLayout.SOUTH);
    }

    public void setBackFrame(String backFrame) {
        this.backFrame = backFrame;
    }

    private Object[][] getFormFieldsForAttendance(String tableName) {
        String[] subjects01 = {"AI", "DBMS", "DM", "NDSA", "OOPS", "SYS_DISC", "CLUB", "AI_LAB",
            "NDSA_LAB", "OOP_LAB", "DBMS_LAB", "ESS", "MENTORING"};
        String[] subjects02 = {"MATHS", "CHEMISTRY", "PYTHON", "BEEE", "MENTORING", "TAMIL",
            "ENGLISH", "LIFESKILLS", "EPL_LAB", "PYTHON_LAB", "CHEMISTRY_LAB"};
        String[] subjects = tableName.equals("STUDENT_ATTENDANCE_01") ? subjects01 : subjects02;

        int initialSize = 2 + subjects.length + 1; // REG_NO, S_NAME, subjects, TOTAL
        Object[][] fields = new Object[initialSize][5];

        fields[0] = new Object[]{"REG_NO", "Register Number:", "text", null, false};
        fields[1] = new Object[]{"S_NAME", "Name:", "text", null, true};

        for (int i = 0; i < subjects.length; i++) {
            fields[i + 2] = new Object[]{subjects[i], subjects[i].replace('_', ' ').toUpperCase() + " Attendance (%):",
                "float", null, true};
        }

        fields[fields.length - 1] = new Object[]{"TOTAL_ATTENDANCE", "Total Attendance (%):", "float",
            null, true};
        return fields;
    }
}
```





```
}

public void loadForm(String mode, String[] existingData) {
    this.mode = mode;
    this.existingRegNo = (mode.equals("EDIT") && existingData.length > 0) ? existingData[0] : null;

    titleLabel.setText(mode + " RECORD - " + tableName.replace('_', ' ').toUpperCase());
    confirmButton.setText(mode.toUpperCase());

    // Clear previous form fields
    formPanel.removeAll();
    inputFields.clear();

    Object[][] fieldDefs;
    if (tableName.startsWith("STUDENT_ATTENDANCE")) {
        fieldDefs = getFormFieldsForAttendance(tableName);
    } else {
        fieldDefs = tableFields.get(tableName);
    }

    if (fieldDefs == null) {
        JOptionPane.showMessageDialog(this, "Configuration for " + tableName + " not found.", "Error",
        JOptionPane.ERROR_MESSAGE);
        return;
    }

    GridBagConstraints gbc = new GridBagConstraints();
    gbc.insets = new Insets(5, 10, 5, 10);
    gbc.anchor = GridBagConstraints.WEST;
    gbc.fill = GridBagConstraints.HORIZONTAL;

    for (int i = 0; i < fieldDefs.length; i++) {
        String colName = (String) fieldDefs[i][0];
        String labelText = (String) fieldDefs[i][1];
        String type = (String) fieldDefs[i][2];
        String[] options = (String[]) fieldDefs[i][3];
        boolean isEditable = (boolean) fieldDefs[i][4];

        JLabel label = new JLabel(labelText);
        gbc.gridx = 0;
        gbc.gridy = i;
        formPanel.add(label, gbc);

        JComponent inputComponent;

        if (type.equals("option") && options != null) {
            JComboBox<String> comboBox = new JComboBox<>(options);
            inputComponent = comboBox;
        } else {
            JTextField textField = new JTextField(20);
            inputComponent = textField;
            if (labelText.contains("Hint:")) {
                labelText = labelText.substring(0, labelText.indexOf("Hint:"));
            }
        }

        // Set initial value in EDIT mode
        if (mode.equals("EDIT") && existingData != null && existingData.length > i) {
            if (inputComponent instanceof JComboBox) {
                ((JComboBox<String>) inputComponent).setSelectedItem(existingData[i]);
            } else if (inputComponent instanceof JTextField) {
                ((JTextField) inputComponent).setText(existingData[i]);
            }
        }
    }
}
```



```
// Set editability
if (mode.equals("EDIT") && !isEditable) {
    inputComponent.setEnabled(false);
}

gbc.gridx = 1;
gbc.weightx = 1.0;
formPanel.add(inputComponent, gbc);
inputFields.put(colName, inputComponent);

// Add hint label
if (labelText.contains("Hint:")) {
    JLabel hintLabel = new JLabel(labelText.substring(labelText.indexOf("Hint:") + 5).trim());
    hintLabel.setForeground(Color.GRAY);
    hintLabel.setFont(new Font("Arial", Font.ITALIC, 10));
    gbc.gridx = 2;
    gbc.weightx = 0.0;
    formPanel.add(hintLabel, gbc);
}
gbc.weightx = 0.0;
}

formPanel.revalidate();
formPanel.repaint();
}

private Map<String, String> collectData() {
    Map<String, String> data = new HashMap<>();
    for (Map.Entry<String, JComponent> entry : inputFields.entrySet()) {
        String colName = entry.getKey();
        JComponent comp = entry.getValue();
        String value = "";

        if (comp instanceof JTextField) {
            value = ((JTextField) comp).getText().trim();
        } else if (comp instanceof JComboBox) {
            Object selectedItem = ((JComboBox) comp).getSelectedItem();
            if (selectedItem != null) {
                value = selectedItem.toString().trim();
            }
        }
        data.put(colName, value);
    }
    return data;
}

private boolean validateInput(Map<String, String> data) {
    // This is a minimal implementation, proper validation requires referencing tableFields again
    // and mapping them to the collected data.
    // Example:
    String errors = "";

    for (Object[] fieldDef : (tableName.startsWith("STUDENT_ATTENDANCE") ?
        getFormFieldsForAttendance(tableName) : tableFields.get(tableName))) {
        String colName = (String) fieldDef[0];
        String label = (String) fieldDef[1];
        String type = (String) fieldDef[2];
        String value = data.get(colName);

        if (value == null || value.isEmpty()) {
            if (colName.equals("S_NAME") || colName.equals("REG_NO") ||
                colName.equals("PHONE_NO")) { // Mandatory fields
                errors += label + " is required.\n";
            }
        }
    }
}
```



```
    }
    continue;
}

try {
    if (type.equals("int")) {
        long val = Long.parseLong(value);
        if (colName.equals("PHONE_NO") && (val < 1000000000L || val > 9999999999L)) {
            errors += "Phone Number must be 10 digits.\n";
        }
    } else if (type.equals("float")) {
        float val = Float.parseFloat(value);
        if (val < 0 || val > 100) {
            errors += label + " must be between 0 and 100.\n";
        }
    } else if (type.equals("date")) {
        new SimpleDateFormat("yyyy-MM-dd").parse(value);
    }
} catch (Exception ex) {
    errors += "Invalid format for " + label + ". Expected " + type + ".\n";
}
}

if (errors.isEmpty() && tableName.equals("STUDENT_RECORDS")) {
    // REG_NO business logic (Year 1: 3111242430, Year 2: 3111252430)
    String regNo = data.get("REG_NO");
    String year = data.get("S_YEAR");
    if (year != null && regNo != null) {
        String prefix = null;
        if (year.equals("1")) prefix = "3111242430";
        else if (year.equals("2")) prefix = "3111252430";

        if (prefix != null && !regNo.startsWith(prefix)) {
            errors += "Reg No for Year " + year + " must start with " + prefix + ".\n";
        }
    }
}

if (!errors.isEmpty()) {
    JOptionPane.showMessageDialog(controller, errors, "Input Error",
JOptionPane.ERROR_MESSAGE);
    return false;
}
return true;
}

private void confirmRecord() {
    Map<String, String> data = collectData();
    if (!validateInput(data)) return;

    String query;
    List<Object> params = new ArrayList<>();
    int rowsAffected = -1;

    // Get the field definitions again to maintain order
    Object[][] fieldDefs = tableName.startsWith("STUDENT_ATTENDANCE") ?
getFormFieldsForAttendance(tableName) : tableFields.get(tableName);

    if (mode.equals("ADD")) {
        // INSERT QUERY
        List<String> colNames = new ArrayList<>();
        String placeholders = "";
```



```
for (Object[] def : fieldDefs) {
    String col = (String) def[0];
    colNames.add(col);
    params.add(data.get(col));
}

placeholders = String.join(" ", java.util.Collections.nCopies(colNames.size(), "?"));
query = String.format("INSERT INTO %s (%s) VALUES (%s)", tableName, String.join(" ",
colNames), placeholders);

// Add Unique Check for STUDENT_RECORDS (optional, but good practice)
if(tableName.equals("STUDENT_RECORDS")) {
    Object[] check = DBUtil.fetchRecord(dbConn, tableName, data.get("REG_NO"));
    if(check[0] != null) {
        JOptionPane.showMessageDialog(controller, "Register Number already exists.", "Error",
JOptionPane.ERROR_MESSAGE);
        return;
    }
}

} else { // EDIT MODE
    // UPDATE QUERY
    List<String> setClauses = new ArrayList<>();

    for (Object[] def : fieldDefs) {
        String col = (String) def[0];
        boolean isEditable = (boolean) def[4];

        if (isEditable) {
            setClauses.add(col + " = ?");
            params.add(data.get(col));
        }
    }

    // Add WHERE clause parameter (existingRegNo)
    params.add(existingRegNo);

    query = String.format("UPDATE %s SET %s WHERE REG_NO = ?", tableName, String.join(" ",
setClauses));
}

rowsAffected = DBUtil.executeUpdate(controller.dbConn, query, params.toArray());

if (rowsAffected > 0) {
    JOptionPane.showMessageDialog(controller, "Record " + mode + "ED successfully!", "Success",
JOptionPane.INFORMATION_MESSAGE);
    controller.showFrame(backFrame);
} else {
    JOptionPane.showMessageDialog(controller, "Failed to " + mode + " the record.", "DB Error",
JOptionPane.ERROR_MESSAGE);
}
}

}

public static void main(String[] args) {
    // Run the GUI creation on the Event Dispatch Thread (standard Java Swing practice)
    SwingUtilities.invokeLater(StudentRecordManager::new);
}
}
```



## Output :

AI & DS STUDENT RECORD MANAGER

View Students records

View student history

View Student Attendance

View student Achievements

MAIN MENU

AI & DS STUDENT RECORD MANAGER

STUDENT RECORDS

VIEW

ADD RECORD

EDIT RECORD

BACK

OPTION TAB

VIEW: Student Records

Sno	Reg No	S Name	S Year	Deb	Hosteller	Phone No	Addr
1	311124243001	Gautam Singh	1	2005-11-27	Y	9166742299	Door 21, St
2	311125243002	Shreya Pillai	2	2004-05-14	Y	9363874040	Door 24, St
3	311124243003	Lavanya Singh	1	2006-08-05	Y	9158917859	Door 32, St
4	311125243004	Lavanya Menon	2	2004-10-23	Y	9826496751	Door 94, St
5	311124243005	Suresh Kumar	1	2004-06-28	N	9634074877	Door 44, St
6	311125243006	Rohan Reddy	2	2004-04-10	Y	9413074788	Door 63, St
7	311124243007	Anjali Iyer	1	2003-05-12	N	9326203399	Door 41, St
8	311125243008	Meera Menon	2	2003-06-07	N	9952381366	Door 14, St
9	311124243009	Rohan Jain	1	2006-06-02	Y	9837973902	Door 34, St
10	311125243010	Vikram Pillai	2	2003-06-21	N	9466402936	Door 25, St
11	311124243011	Suresh Kumar	1	2006-02-28	N	9396228621	Door 65, St
12	311125243012	Priya Jain	2	2005-04-27	N	9337047364	Door 10, St
13	311124243013	Gautam Iyer	1	2005-03-03	N	9794475254	Door 36, St
14	311125243014	Vikram Reddy	2	2004-12-12	N	9411548887	Door 30, St
15	311124243015	Snaha Menon	1	2005-10-19	N	9730147940	Door 30, St
16	311125243016	Karthik Reddy	2	2005-04-30	Y	9191617187	Door 88, St
17	311124243017	Vivek Singh	1	2006-12-19	N	9448065216	Door 54, St
18	311125243018	Asha Menon	2	2004-07-26	Y	9715096462	Door 19, St
19	311124243019	Arun Nair	1	2003-06-04	N	9746114896	Door 51, St
20	311125243020	Deepa Pillai	2	2004-10-07	N	9311832763	Door 58, St
21	311124243021	Priya Kumar	1	2006-12-04	N	9812081293	Door 62, St

BACK

STUDENT RECORDS

VIEW: Student History

Sno	Reg No	S Name	Father Name	Mother Name	Gr 11 Group	Gr 11 School	Gr 11 M.
1	311124243001	Gautam Singh	Mr. Gautam S.	Mrs. Singh B.	Maths-CS	School A4	82.8
2	311125243002	Shreya Pillai	Mr. Shreya S.	Mrs. Pillai B.	Biology-Maths	School A5	87.1
3	311124243003	Lavanya Singh	Mr. Lavanya S.	Mrs. Singh B.	Biology-Maths	School A3	82.3
4	311125243004	Lavanya Menon	Mr. Lavanya S.	Mrs. Menon B.	Biology-Maths	School A2	82.5
5	311124243005	Suresh Kumar	Mr. Suresh S.	Mrs. Kumar B.	Biology-Maths	School A3	81.2
6	311125243006	Rohan Reddy	Mr. Rohan S.	Mrs. Reddy B.	Maths-CS	School A2	94.6
7	311124243007	Anjali Iyer	Mr. Anjali S.	Mrs. Iyer B.	Commerce	School A2	84.1
8	311125243008	Meera Menon	Mr. Meera S.	Mrs. Menon B.	Biology-Maths	School A5	81.2
9	311124243009	Rohan Jain	Mr. Rohan S.	Mrs. Jain B.	Biology-Maths	School A1	90.5
10	311125243010	Vikram Pillai	Mr. Vikram S.	Mrs. Pillai B.	Maths-CS	School A4	91.5
11	311124243011	Suresh Kumar	Mr. Suresh S.	Mrs. Kumar B.	Biology-Maths	School A4	92.9
12	311125243012	Priya Jain	Mr. Priya S.	Mrs. Jain B.	Commerce	School A2	86.8
13	311124243013	Gautam Iyer	Mr. Gautam S.	Mrs. Iyer B.	Maths-CS	School A2	86.1
14	311125243014	Vikram Reddy	Mr. Vikram S.	Mrs. Reddy B.	Maths-CS	School A2	80.9
15	311124243015	Snaha Menon	Mr. Snaha S.	Mrs. Menon B.	Biology-Maths	School A4	82.6
16	311125243016	Karthik Reddy	Mr. Karthik S.	Mrs. Reddy B.	Commerce	School A2	89.3
17	311124243017	Vivek Singh	Mr. Vivek S.	Mrs. Singh B.	Biology-Maths	School A4	84.3
18	311125243018	Asha Menon	Mr. Asha S.	Mrs. Menon B.	Maths-CS	School A2	86.4
19	311124243019	Arun Nair	Mr. Arun S.	Mrs. Nair B.	Biology-Maths	School A5	81.4
20	311125243020	Deepa Pillai	Mr. Deepa S.	Mrs. Pillai B.	Commerce	School A4	91.0
21	311124243021	Priya Kumar	Mr. Priya S.	Mrs. Kumar B.	Biology-Maths	School A3	86.4

BACK

STUDENT HISTORY



**LOYOLA-ICAM COLLEGE OF ENGINEERING AND TECHNOLOGY (LICET)**  
(Autonomous)  
Loyola Campus, Nungambakkam, Chennai –600034

Page No:

AI & DS STUDENT RECORD MANAGER

VIEW: Attendance - Year I

Sno	Reg No	S Name	AI	DBMS	DM	NDSS	OOPS
1	311124243001	Gautam Singh	89.4	75.7	78.0	96.3	78.4
2	311124243003	Lavanya Menon	85.9	84.2	84.7	88.0	90.6
3	311124243005	Suresh Kumar	91.8	93.9	95.4	83.9	91.7
4	311124243007	Arjuni Iyer	86.0	86.3	84.1	83.1	96.5
5	311124243009	Rohan Jain	82.8	80.1	97.8	93.9	93.9
6	311124243011	Suresh Kumar	93.1	81.3	78.7	78.7	84.9
7	311124243013	Gautam Iyer	91.7	85.0	86.4	85.9	89.7
8	311124243015	Sneha Menon	85.5	89.5	85.6	95.2	79.7
9	311124243017	Vivek Singh	88.5	97.8	92.8	84.2	90.2
10	311124243019	Arjun Nair	82.5	78.5	87.5	78.6	81.3
11	311124243021	Priya Kumar	97.0	91.1	78.2	84.2	88.3
12	311124243023	Nandini Nair	88.7	94.5	95.7	89.4	96.2
13	311124243025	Arjun Sharma	85.8	80.5	93.8	88.1	94.0
14	311124243027	Asha Singh	96.5	83.8	80.6	84.2	95.1
15	311124243029	Lavanya Singh	94.3	88.0	82.0	78.2	89.9
16	311124243031	Tanya Menon	90.4	84.4	90.3	85.5	93.7
17	311124243033	Rahul Gupta	87.1	85.8	87.0	82.7	81.3
18	311124243035	Nandini Menon	92.5	82.9	87.2	83.4	89.0
19	311124243037	Rahul Nair	83.0	76.6	88.9	90.6	97.9
20	311124243039	Harish Sharma	87.9	91.8	95.1	82.4	78.8
21	311124243041	Asha Menon	86.2	86.2	80.6	95.2	76.8

BACK

## ATTENDANCE - I

AI & DS STUDENT RECORD MANAGER

VIEW: Attendance - Year II

Sno	Reg No	S Name	Maths	Chemistry	Python	Beer	Mentor
1	311125243002	Shreya Pillai	81.2	88.0	90.8	95.7	87.8
2	311125243004	Lavanya Menon	84.3	82.6	83.5	82.2	91.7
3	311125243006	Rohan Reddy	94.9	91.2	97.5	95.7	95.5
4	311125243008	Meera Menon	91.8	91.3	89.4	90.9	91.8
5	311125243010	Vikram Pillai	93.8	82.2	83.2	95.1	78.9
6	311125243012	Priya Jain	85.9	94.0	94.8	85.0	79.6
7	311125243014	Vikram Reddy	76.0	94.4	95.6	89.3	86.5
8	311125243016	Karthik Reddy	86.3	82.1	95.7	84.6	97.9
9	311125243018	Aisha Menon	97.3	81.7	88.5	84.7	87.6
10	311125243020	Deepa Pillai	92.3	95.8	96.8	86.7	78.1
11	311125243022	Aarav Pillai	82.4	91.6	93.0	85.5	85.8
12	311125243024	Vikram Reddy	91.3	93.1	90.7	82.6	80.2
13	311125243026	Priya Nair	84.0	90.6	76.1	86.7	94.5
14	311125243028	Tanya Gupta	94.8	85.8	77.7	79.9	77.6
15	311125243030	Meera Singh	92.8	89.4	80.7	87.9	90.1
16	311125243032	Rohan Sharma	81.3	93.9	86.3	77.5	91.4
17	311125243034	Asha Kumar	78.5	85.8	78.0	90.8	79.4
18	311125243036	Sneha Reddy	86.0	96.6	83.2	97.9	76.5
19	311125243038	Tanya Singh	95.7	96.0	84.4	90.1	93.3
20	311125243040	Deepa Gupta	87.3	92.1	86.3	96.9	95.7
21	311125243042	Aarav Reddy	96.4	81.7	96.9	92.6	77.1

BACK

## ATTENDANCE - II

AI & DS STUDENT RECORD MANAGER

VIEW: Student Achievements

Sno	Reg No	S Name	A Year	Prize	Event Name	Event Date	Feedback
1	311124243001	Gautam Singh	2023-24	First Place	Hackathon	2025-02-18	Excellent
2	311125243002	Shreya Pillai	2023-24	Second Place	Data Science Quiz	2025-01-01	Excellent
3	311124243003	Lavanya Singh	2023-24	Participation	Hackathon	2025-02-13	Excellent
4	311125243004	Lavanya Menon	2024-25	First Place	Hackathon	2025-01-25	Excellent
5	311124243005	Suresh Kumar	2023-24	Second Place	Hackathon	2024-11-08	Excellent
6	311125243006	Rohan Reddy	2024-25	Second Place	Data Science Quiz	2025-03-28	Excellent
7	311124243007	Arjuni Iyer	2024-25	Participation	Coding Marathon	2025-06-03	Excellent
8	311125243008	Meera Menon	2023-24	Participation	Debate Competiti	2024-12-10	Excellent
9	311124243009	Rohan Jain	2023-24	Participation	Debate Competiti	2025-04-23	Excellent
10	311125243010	Vikram Pillai	2024-25	Participation	Debate Competiti	2025-10-24	Excellent
11	311124243011	Suresh Kumar	2024-25	First Place	Debate Competiti	2025-05-08	Excellent
12	311124243012	Priya Jain	2024-25	Participation	Data Science Quiz	2025-02-16	Excellent
13	311124243013	Gautam Iyer	2024-25	First Place	Hackathon	2025-07-23	Excellent
14	311125243014	Vikram Reddy	2024-25	Participation	Coding Marathon	2025-10-06	Excellent
15	311124243015	Sneha Menon	2024-25	Second Place	Hackathon	2025-01-14	Excellent
16	311125243016	Karthik Reddy	2024-25	First Place	Hackathon	2025-09-18	Excellent
17	311124243017	Vivek Singh	2023-24	Participation	Hackathon	2025-10-14	Excellent
18	311125243018	Aisha Menon	2023-24	Second Place	Debate Competiti	2025-09-03	Excellent
19	311124243019	Arjun Nair	2023-24	Participation	Coding Marathon	2025-04-28	Excellent
20	311125243020	Deepa Pillai	2024-25	First Place	Coding Marathon	2025-06-27	Excellent
21	311124243021	Priya Kumar	2024-25	Second Place	Hackathon	2025-01-08	Excellent

BACK

## STUDENT ACHIEVEMENTS

AI & DS STUDENT RECORD MANAGER

ADD RECORD - Student Achievements

Register Number:

Name:

Academic Year:

Prize:  Optional

Event Name:

Event Date (YYYY-MM-DD):

Feedback (Max 150 char):  Optional

Tech/Non-Tech Event:  Y/N

CONFIRM BACK

## ADDING NEW RECORD



**ADD RECORD - Student Achievements**

Register Number: 311124243030  
Name: John Febin  
Academic Year: 2  
Prize: First Prize  
Event Name: All India Chess Tournament  
Event Date (YYYY-MM-DD): 2025-07-01  
Feedback (Max 150 char): Excellent tournament  
Tech/Non-Tech Event: N

CONFIRM BACK

**ADDING NEW RECORD**

**EDIT RECORD - Student Records**

Name: Nandini Jain  
Year (1, 2, 3, or 4): 1  
Register Number: 311124243001  
DOB (YYYY-MM-DD): 2004-02-05  
Hostel: Y  
Phone Number (10 digits): 9381974674  
Address: Door 37, Street 1, City Red

CONFIRM BACK

**EDITING RECORDS**

## Conclusion :

The Student Record Management System successfully provided a robust Java desktop application with a SQLPlus database backend. The project demonstrated proficiency in JDBC connectivity and database design by implementing a relational schema with Primary/Foreign Keys and Triggers for automatic indexing. It met all functional goals, ensuring efficient, centralized CRUD management of core student, history, attendance, and achievement records.