OBE IMPLEMENTATION:UNIVERSITY SETTING

by

Team Name: SteveRogers
Module Name: Courses

Darvini[AP22110010167]
Sindhu[AP22110010180]
Anjana[AP22110010181]
Vidhathri[AP22110010196]
Manasa[AP22110010606]

A report for the CS307:Mobile Application Development using JAVA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRM UNIVERSITY AP::AMARAVATI

INDEX

Introduction **2**
Project Modules: 3
Architecture Diagram **3**
Module Description **4**
Programming Details naming conventions to be used: 5
Table details:Courses 5
Source Code **5**
Screen Shots **6**
Conclusion **7**

Introduction

Our University (herewith considered as SRM-AP) is going to implement OBE(Outcome Based Education) in their university and you are assigned in the project to develop a CURD(Create,Update,Retrieve and Delete) windows and mobile application using JAVA programming and Android studio for the same.

Project Modules:

Various Modules available in the project are

1.Blooms Level setting

2.Program Level Objective Setting

3.University

4.Schools

5.Department

6.Programs

7.Courses

8.Course objective setting

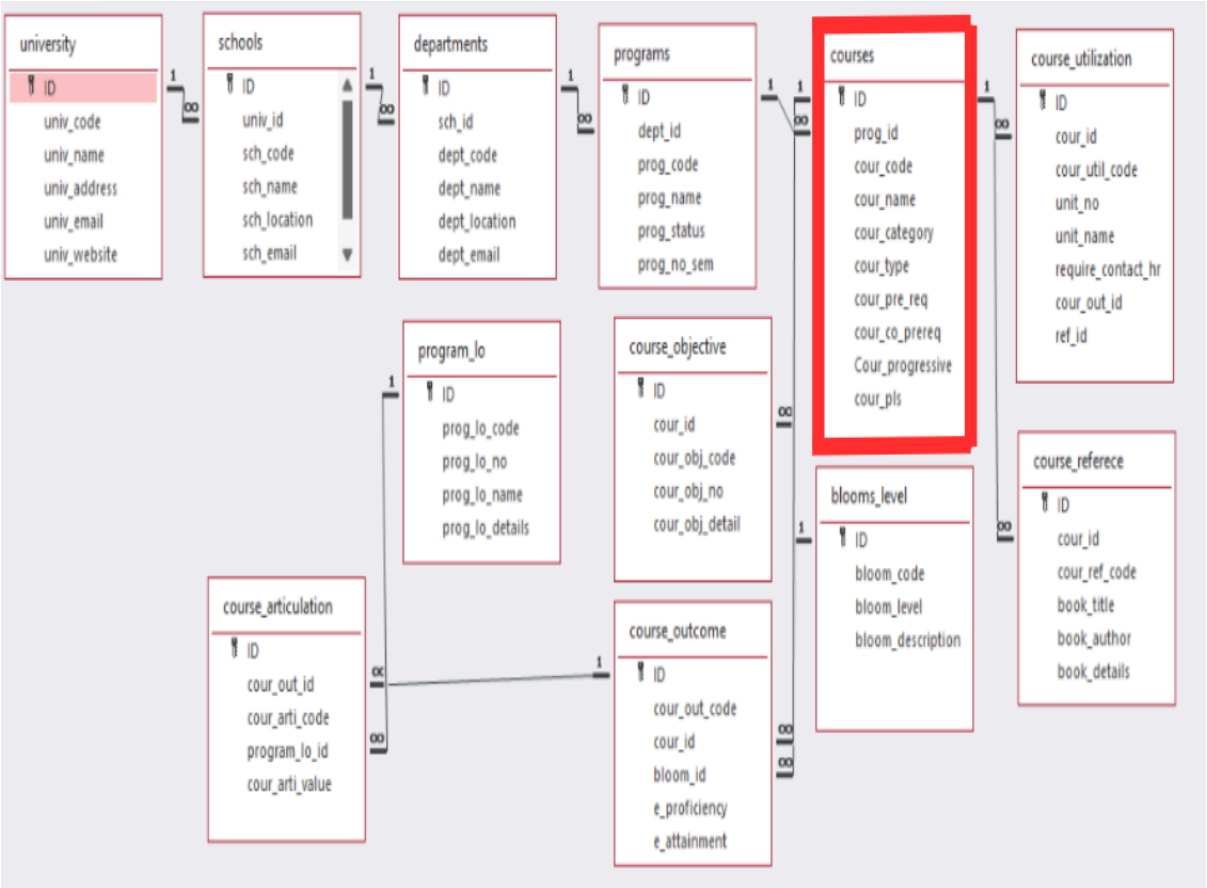
9.Course Outcome Setting

10.Course Articulation matrix Setting

11.Course Utilization Setting

12.Course Reference Setting.

Architecture Diagram



Module Description

Module Name: Courses
Module Description:

This module is used to create,Update,Retrieve,Delete(hereafter known as CURD) details of the module and storing the details in the database table(eg.MySQL).

Programming Details naming conventions to be used:

- **class name/activity name:** SteveRogers_courses
- **Function/method name**
  - **Create:** AP22110010167_courses_create
  - **Update:** AP22110010180_courses_update
  - **Retrieve:** AP22110010181_courses_retrive
  - **Delete:** AP22110010606_courses_delete

Table details:[Courses]

Field Name	Data type
id	Integer
programId	Integer
code	String
name	String
category	String
type	String
preReq	boolean or String
coReq	boolean or String
progressive	boolean or String
pls	boolean or String

Source Code

LoginForm.java:

```
package com.mycompany.sqliteproject;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class LoginForm extends JFrame {
    private JTextField usernameField;
    private JPasswordField passwordField;

    public LoginForm() {
        setTitle("Login");
        setSize(350, 180);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(5, 5, 5, 5);

        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.anchor = GridBagConstraints.EAST;
        add(new JLabel("Username:"), gbc);

        gbc.gridx = 1;
        gbc.anchor = GridBagConstraints.WEST;
        usernameField = new JTextField(15);
```

```
        add(usernameField, gbc);

        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.anchor = GridBagConstraints.EAST;
        add(new JLabel("Password:"), gbc);

        gbc.gridx = 1;
        gbc.anchor = GridBagConstraints.WEST;
        passwordField = new JPasswordField(15);
        add(passwordField, gbc);

        JPanel buttonPanel = new JPanel();
        JButton loginButton = new JButton("Login");
        JButton resetButton = new JButton("Reset");
loginButton.addActionListener(e -> authenticateUser());
        resetButton.addActionListener(e -> {
            usernameField.setText("");
            passwordField.setText("");
        });

        buttonPanel.add(loginButton);
        buttonPanel.add(resetButton);
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.gridwidth = 2;
        gbc.anchor = GridBagConstraints.CENTER;
        add(buttonPanel, gbc);

        setVisible(true);
    }

    private void authenticateUser() {
        String username = usernameField.getText().trim();
        String password = new String(passwordField.getPassword()).trim();

        if (username.isEmpty() || password.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please enter both username and
password.");
            return;
        }

        if (isValidUser(username, password)) {
            JOptionPane.showMessageDialog(this, "Login Successful!");
            dispose();
            new CoursesGUI();
        } else {
            JOptionPane.showMessageDialog(this, " Invalid Username or Password.");
        }
    }

    private boolean isValidUser(String username, String password) {
        boolean isValid = false;
        String url = "jdbc:sqlite:C:/Users/gvidh/Desktop/Apps/mydatabase.db";
        String query = "SELECT password FROM users WHERE username = ?";

        try (Connection conn = DriverManager.getConnection(url);
```

```
         PreparedStatement pstmt = conn.prepareStatement(query)) {
        pstmt.setString(1, username);
        ResultSet rs = pstmt.executeQuery();

        if (rs.next()) {
            String storedPassword = rs.getString("password");
            System.out.println("Stored Password: " + storedPassword);
            System.out.println("Entered Password: " + password);

            isValid = storedPassword.equals(password);
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, " Database Error: " + e.getMessage());
    }
    return isValid;
  }

  public static void main(String[] args) {
    new LoginForm();
  }
}
```

SteveRogers_courses.java:

```
package com.mycompany.sqliteproject;
import java.sql.*;
import java.util.Scanner;
import java.util.List;
import java.util.ArrayList;
public class SteveRogers_courses {
   private static final String DB_URL =
"jdbc:sqlite:C:/Users/gvidh/Desktop/Apps/mydatabase.db";
   private static Connection conn = null;
   public static void connect() {
      try {
         if (conn == null || conn.isClosed()) {
            conn = DriverManager.getConnection(DB_URL);
            System.out.println("Connected to SQLite database.");
         }
      } catch (SQLException e) {
         System.out.println("Connection failed: " + e.getMessage());
      }
   }
   public static void closeConnection() {
      try {
         if (conn != null && !conn.isClosed()) {
            conn.close();
            System.out.println("Database connection closed.");
         }
      } catch (SQLException e) {
         System.out.println("Error closing connection: " + e.getMessage());
      }
   }
```

```
    public static void AP22110010167_courses_create(int progId, String code, String
name, String category, String type, String preReq, String coPrereq, String
progressive, String pls) {
        String sql = "INSERT INTO courses (prog_id, cour_code, cour_name,
cour_category, cour_type, cour_pre_req, cour_co_prereq, cour_progressive,
cour_pls) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, progId);
            pstmt.setString(2, code);
            pstmt.setString(3, name);
            pstmt.setString(4, category);
            pstmt.setString(5, type);
            pstmt.setString(6, preReq);
            pstmt.setString(7, coPrereq);
            pstmt.setString(8, progressive);
            pstmt.setString(9, pls);
            pstmt.executeUpdate();
            System.out.println("Course added successfully.");
        } catch (SQLException e) {
            System.out.println("Error inserting course: " + e.getMessage());
        }
    }
    public static void AP22110010180_courses_retrieve() {
        String sql = "SELECT * FROM courses";
        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            boolean hasData = false;
            while (rs.next()) {
                hasData = true;
                System.out.println("\nCourse ID: " + rs.getInt("ID") +
                        "\nProgram ID: " + rs.getInt("prog_id") +
                        "\nCode: " + rs.getString("cour_code") +
                        "\nName: " + rs.getString("cour_name") +
                        "\nCategory: " + rs.getString("cour_category") +
                        "\nType: " + rs.getString("cour_type") +
                        "\nPre-req: " + rs.getString("cour_pre_req") +
                        "\nCo-prereq: " + rs.getString("cour_co_prereq") +
                        "\nProgressive: " + rs.getString("cour_progressive") +
                        "\nPLS: " + rs.getString("cour_pls") + "\n-------------------");
            }
            if (!hasData) {
                System.out.println("No courses found.");
            }
        } catch (SQLException e) {
            System.out.println("Error retrieving courses: " + e.getMessage());
        }
    }
    public static void AP22110010181_courses_update(int id, Integer progId, String
code, String name, String category,
                                String type, String preReq, String coReq, String progressive,
String pls) {
        StringBuilder sql = new StringBuilder("UPDATE courses SET ");
        List<Object> values = new ArrayList<>();
```

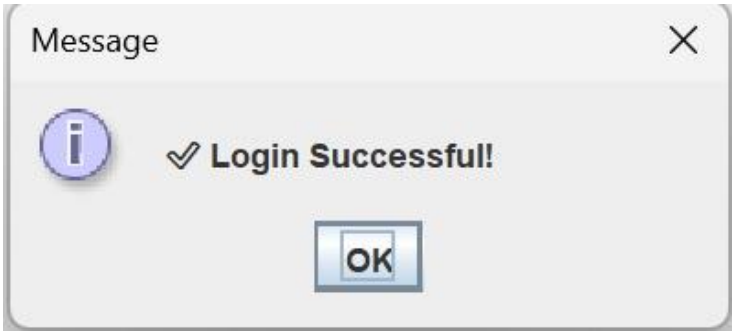
```
        if (progId != null) {
            sql.append("prog_id = ?, ");
            values.add(progId);
        }
        if (code != null && !code.isEmpty()) {
            sql.append("cour_code = ?, ");
            values.add(code);
        }
        if (name != null && !name.isEmpty()) {
            sql.append("cour_name = ?, ");
            values.add(name);
        }
        if (category != null && !category.isEmpty()) {
            sql.append("cour_category = ?, ");
            values.add(category);
        }
        if (type != null && !type.isEmpty()) {
            sql.append("cour_type = ?, ");
            values.add(type);
        }
        if (preReq != null && !preReq.isEmpty()) {
            sql.append("cour_pre_req = ?, ");
            values.add(preReq);
        }
        if (coReq != null && !coReq.isEmpty()) {
            sql.append("cour_co_prereq = ?, ");
            values.add(coReq);
        }
        if (progressive != null && !progressive.isEmpty()) {
            sql.append("cour_progressive = ?, ");
            values.add(progressive);
        }
        if (pls != null && !pls.isEmpty()) {
            sql.append("cour_pls = ?, ");
            values.add(pls);
        }
        if (values.isEmpty()) {
            System.out.println("No fields provided for update.");
            return;
        }
        sql.setLength(sql.length() - 2);
        sql.append(" WHERE ID = ?");
        values.add(id);
        try(Connection conn =
DriverManager.getConnection("jdbc:sqlite:C:/Users/gvidh/Desktop/Apps/mydatabase
.db");
            PreparedStatement pstmt = conn.prepareStatement(sql.toString())) {
            for (int i = 0; i < values.size(); i++) {
                pstmt.setObject(i + 1, values.get(i));
            }
            int updated = pstmt.executeUpdate();
            System.out.println(updated > 0 ? "Updated successfully!" : "No such course
with ID " + id);
```

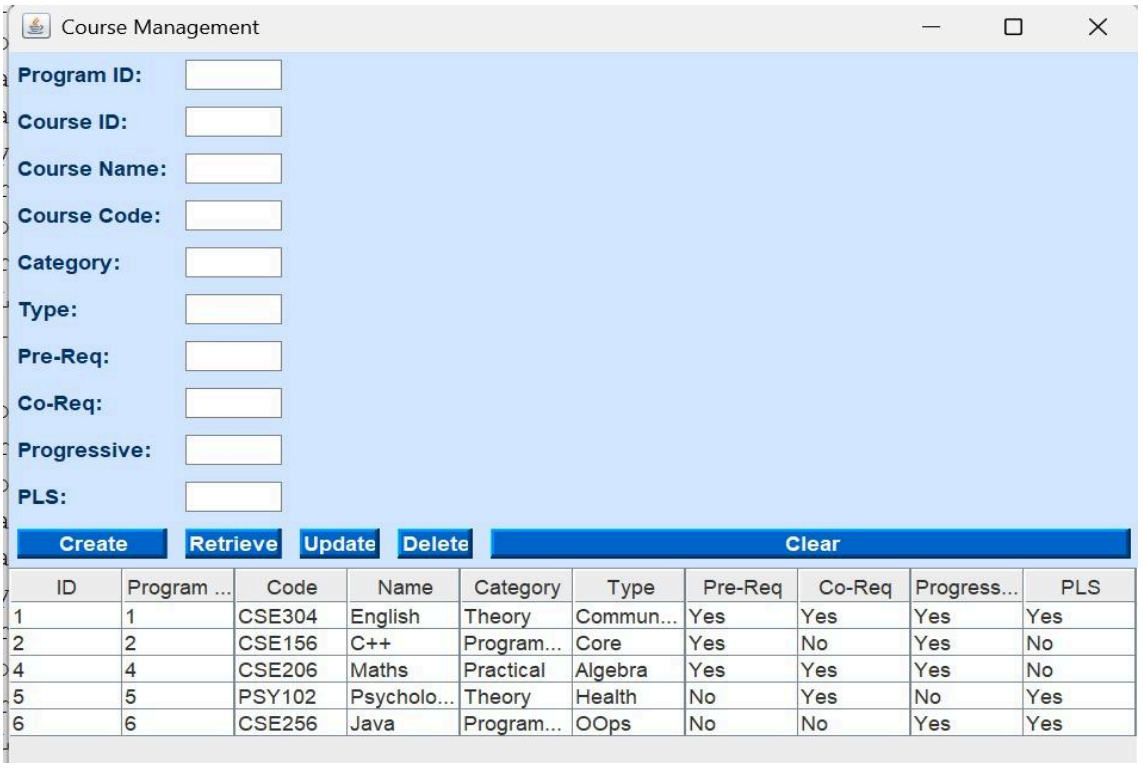
```
        } catch (SQLException e) {
            System.out.println("SQL Error: " + e.getMessage());
        }
    }
    public static void AP22110010606_courses_delete(int id) {
        String sql = "DELETE FROM courses WHERE ID = ?";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, id);
            int rowsDeleted = pstmt.executeUpdate();
            if (rowsDeleted > 0) {
                System.out.println("Course deleted successfully.");
            } else {
                System.out.println("Course not found.");
            }
        } catch (SQLException e) {
            System.out.println("Error deleting course: " + e.getMessage());
        }
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        connect();
        while (true) {
            System.out.println("\nCRUD Operations: 1. Create  2. Retrieve  3. Update  4.
Delete  5. Exit");
            System.out.print("Enter choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine();
            switch (choice) {
                case 1:
                    System.out.print("Enter Program ID: ");
                    int progId = scanner.nextInt();
                    scanner.nextLine();
                    System.out.print("Enter Course Code: ");
                    String code = scanner.nextLine();
                    System.out.print("Enter Course Name: ");
                    String name = scanner.nextLine();
                    System.out.print("Enter Course Category: ");
                    String category = scanner.nextLine();
                    System.out.print("Enter Course Type: ");
                    String type = scanner.nextLine();
                    System.out.print("Enter Course Pre-Req: ");
                    String preReq = scanner.nextLine();
                    System.out.print("Enter Course Co-Prereq: ");
                    String coPrereq = scanner.nextLine();
                    System.out.print("Enter Course Progressive: ");
                    String progressive = scanner.nextLine();
                    System.out.print("Enter Course PLS: ");
                    String pls = scanner.nextLine();
                    AP22110010167_courses_create(progId, code, name, category, type,
preReq, coPrereq, progressive, pls);
                    break;
                case 2:
                    AP22110010180_courses_retrieve();
```

```
                break;
            case 3:
                System.out.print("Enter Course ID to update: ");
                int updateId = scanner.nextInt();
                scanner.nextLine();
                String fetchSql = "SELECT * FROM courses WHERE ID = ?";
                try (PreparedStatement fetchStmt = conn.prepareStatement(fetchSql)) {
                    fetchStmt.setInt(1, updateId);
                    ResultSet rs = fetchStmt.executeQuery();
                    if(rs.next()){
                        int currProgId = rs.getInt("prog_id");
                        String currCode = rs.getString("cour_code");
                        String currName = rs.getString("cour_name");
                        String currCategory = rs.getString("cour_category");
                        String currType = rs.getString("cour_type");
                        String currPreReq = rs.getString("cour_pre_req");
                        String currCoPrereq = rs.getString("cour_co_prereq");
                        String currProgressive = rs.getString("cour_progressive");
                        String currPls = rs.getString("cour_pls");
                        System.out.print("Enter new Program ID (or -1 to keep current): ");
                        int newProgId = scanner.nextInt();
                        scanner.nextLine();
                        System.out.print("Enter new Course Code (leave blank to keep
current): ");
                        String newCode = scanner.nextLine();
                        if (newCode.isEmpty()) newCode = currCode;
                        System.out.print("Enter new Course Name (leave blank to keep
current): ");
                        String newName = scanner.nextLine();
                        if (newName.isEmpty()) newName = currName;
                        System.out.print("Enter new Course Category (leave blank to keep
current): ");
                        String newCategory = scanner.nextLine();
                        if (newCategory.isEmpty()) newCategory = currCategory;
                        System.out.print("Enter new Course Type (leave blank to keep
current): ");
                        String newType = scanner.nextLine();
                        if (newType.isEmpty()) newType = currType;
                        System.out.print("Enter new Course Pre-Req (leave blank to keep
current): ");
                        String newPreReq = scanner.nextLine();
                        if (newPreReq.isEmpty()) newPreReq = currPreReq;
                        System.out.print("Enter new Course Co-Prereq (leave blank to
keep current): ");
                        String newCoPrereq = scanner.nextLine();
                        if (newCoPrereq.isEmpty()) newCoPrereq = currCoPrereq;
                        System.out.print("Enter new Course Progressive (leave blank to
keep current): ");
                        String newProgressive = scanner.nextLine();
                        if (newProgressive.isEmpty()) newProgressive = currProgressive;
                        System.out.print("Enter new Course PLS (leave blank to keep
current): ");
                        String newPls = scanner.nextLine();
```

```
                    if (newPls.isEmpty()) newPls = currPls;
                    if (newProgId == -1) newProgId = currProgId;
                    AP22110010181_courses_update(updateId, newProgId, newCode,
newName, newCategory, newType, newPreReq, newCoPrereq,
newProgressive, newPls);
                } else {
                    System.out.println("Course not found with ID: " + updateId);
                }
            } catch (SQLException e) {
                System.out.println("Error fetching course for update: " +
e.getMessage());
            }
            break;
        case 4:
            System.out.print("Enter Course ID to delete: ");
            int deleteId = scanner.nextInt();
            AP22110010606_courses_delete(deleteId);
            break;
        case 5:
            closeConnection();
            System.out.println("Exiting...");
            return;
        default:
            System.out.println("Invalid choice, try again.");
    }
  }
}
}
```

Screen Shots





ID	Program ...	Code	Name	Category	Type	Pre-Req	Co-Req	Progress...	PLS
1	1	CSE304	English	Theory	Commun...	Yes	Yes	Yes	Yes
2	2	CSE156	C++	Program...	Core	Yes	No	Yes	No
4	4	CSE206	Maths	Practical	Algebra	Yes	Yes	Yes	No
5	5	PSY102	Psycholo...	Theory	Health	No	Yes	No	Yes
6	6	CSE256	Java	Program...	OOps	No	No	Yes	Yes

Conclusion

The Courses Module of the Outcome-Based Education (OBE) Implementation System has been successfully developed using Java Swing for the graphical user interface and SQLite for the backend database. This module allows users to perform essential CRUD (Create, Retrieve, Update, Delete) operations on course-related data through an intuitive and responsive GUI. The inclusion of a JTable provides a dynamic and interactive view of the stored course information, enhancing data visibility and user experience.

Key course details such as Program ID, Course Code, Course Name, Category, Type, Prerequisites, Corequisites, Progressive nature, and PLS have been seamlessly integrated and are displayed in real-time. This ensures that administrators can efficiently manage academic courses in alignment with OBE standards.

Overall, the project lays a solid foundation for building a comprehensive and scalable education management system. It improves academic planning and transparency while simplifying course handling for faculty and administrative staff.