OBE IMPLEMENTATION : COURSES

By

G.Sai Chandhan [AP22110010343]

E.Likhith [AP22110010386]

N.Mahimanth [AP22110010329]

K.Nehar [AP22110010381]

K.Deepika [AP22110010423]

A report for CES307:Mobile Application Development using JAVA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRM UNIVERSITY AP::AMARAVATI

INDEX

Introduction

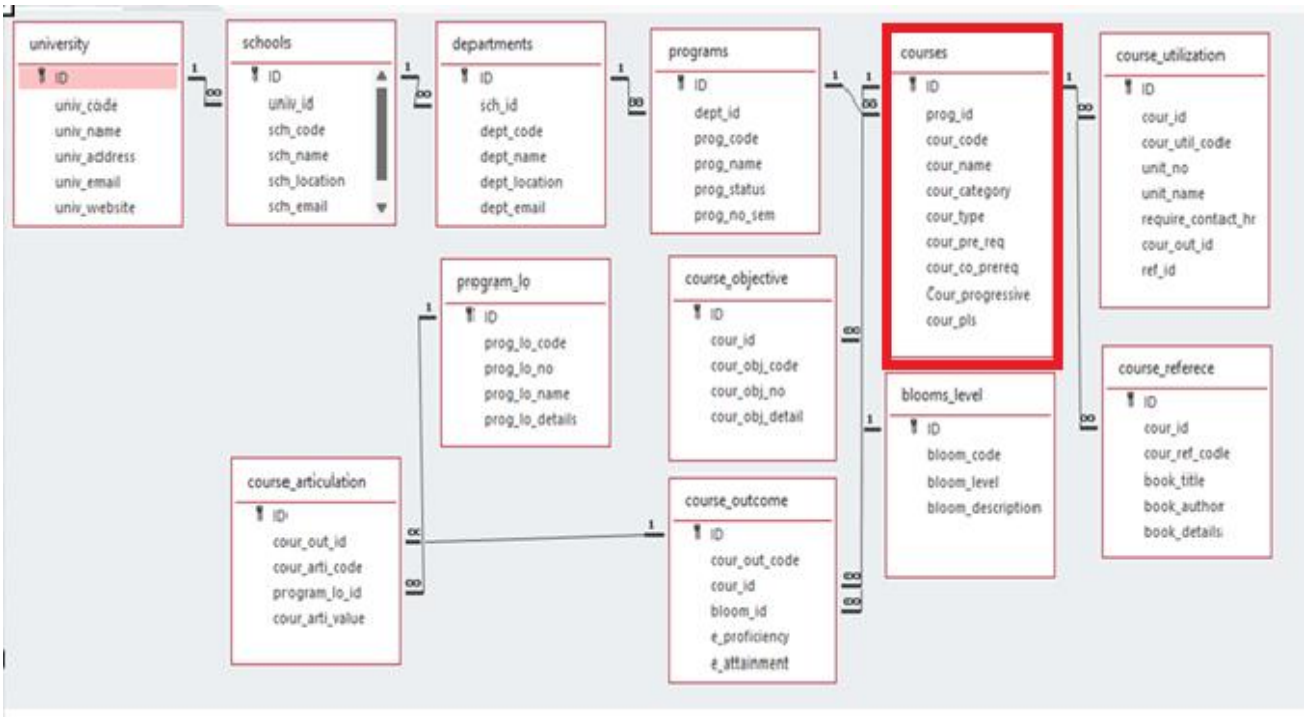
Our University (herewith considered as SRM-AP) is going to implement OBE(Outcome Based Education) in their university and you are assigned in the project to develop a CURD(Create,Update,Retrieve and Delete) windows and mobile application using JAVA programming and Android studio for the same.

Modules in the project:

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 (in SQLite).

Programming Details naming conventions to be used:

- **class name/activity name:**JavaBeans_courses
- **Function/method name**
  - **Create:** JavaBeans_courses_create
  - **Update:** JavaBeans_courses_update
  - **Retrieve:** JavaBeans_courses _retrive
  - **Delete:** JavaBeans_courses _delete

Table Details:

Field Name	Datatype
id	String
program_id	String
code	String
name	String
category	String
type	String
prereq	String
coreq	String
progression	String
course_plan	String

Source Code:

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.util.ArrayList;

public class JavaBeans_courses {
    public static void main(String[] args) {
        try (Connection con = DriverManager.getConnection("jdbc:sqlite:courses.db")) {
            con.createStatement().execute("CREATE TABLE IF NOT EXISTS users (username
TEXT PRIMARY KEY, password TEXT)");
            con.createStatement().execute("""
                CREATE TABLE IF NOT EXISTS courses (
                    id TEXT,
                    program_id TEXT,
                    code TEXT PRIMARY KEY,
                    name TEXT,
                    category TEXT,
                    type TEXT,
                    prereq TEXT,
                    coreq TEXT,
                    progression TEXT,
                    course_plan TEXT
                )
            """);
        } catch (Exception e) {
            e.printStackTrace();
        }
        new ModernLogin();
    }
}

class ModernLogin extends Frame {
    TextField user, pass;
    Label status;

    ModernLogin() {
        setTitle("Login");
        setSize(300, 200);
        setLayout(null);
        setBackground(Color.WHITE);

        Label title = new Label("Course Login", Label.CENTER);
```

```
title.setFont(new Font("Segoe UI", Font.BOLD, 15));
title.setBounds(70, 30, 160, 20);
add(title);

Label userLabel = new Label("Username:");
Label passLabel = new Label("Password:");
userLabel.setBounds(40, 60, 70, 20);
passLabel.setBounds(40, 90, 70, 20);
add(userLabel);
add(passLabel);

user = new TextField();
pass = new TextField();
pass.setEchoChar('*');
user.setBounds(120, 60, 120, 22);
pass.setBounds(120, 90, 120, 22);
add(user);
add(pass);

Button login = new Button("Login");
Button register = new Button("Register");
login.setBounds(40, 130, 80, 25);
register.setBounds(160, 130, 80, 25);
add(login);
add(register);

status = new Label("", Label.CENTER);
status.setBounds(40, 160, 200, 20);
status.setForeground(Color.RED);
add(status);

login.addActionListener(e -> {
    if (CourseOperations.authenticate(user.getText(), pass.getText())) {
        status.setText("Login Successful");
        dispose();
        new CourseDashboard(user.getText());
    } else {
        status.setText("Invalid Credentials");
    }
});

register.addActionListener(e -> {
    CourseOperations.register(user.getText(), pass.getText());
    status.setText("User Registered");
```

```
        });

        setVisible(true);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                dispose();
            }
        });
    }
}

class CourseDashboard extends Frame implements ActionListener {
    TextField[] fields = new TextField[10];
    TextArea display;
    String[] labels = {"ID", "Program ID", "Code", "Name", "Category", "Type", "Pre-req", "Co-req",
"Progression", "Course plan"};
    String originalCode = "";
    String loggedInUser;

    CourseDashboard(String username) {
        this.loggedInUser = username;

        setTitle("Course Dashboard");
        setSize(780, 600);
        setLayout(null);
        setBackground(Color.WHITE);

        Label header = new Label("Course Management", Label.CENTER);
        header.setFont(new Font("Segoe UI", Font.BOLD, 20));
        header.setBounds(20, 40, 740, 30);
        add(header);

        Label userInfo = new Label("Logged in as: " + username);
        userInfo.setBounds(40, 70, 300, 20);
        add(userInfo);

        Panel formPanel = new Panel();
        formPanel.setLayout(new GridLayout(10, 2, 8, 6));
        formPanel.setBounds(40, 100, 700, 280);
        formPanel.setBackground(Color.LIGHT_GRAY);
        for (int i = 0; i < labels.length; i++) {
            Label l = new Label(labels[i]);
            TextField tf = new TextField();
            fields[i] = tf;
```

```
            formPanel.add(l);
            formPanel.add(tf);
        }
        add(formPanel);

        Panel buttonPanel = new Panel();
        buttonPanel.setLayout(new GridLayout(1, 7, 10, 0));
        buttonPanel.setBounds(40, 390, 700, 30);
        String[] btns = {"Create", "Update", "Delete", "View All", "Retrive", "Get by Program",
"Reset"};
        for (String btn : btns) {
            Button b = new Button(btn);
            b.addActionListener(this);
            buttonPanel.add(b);
        }
        add(buttonPanel);

        display = new TextArea();
        display.setBounds(40, 430, 700, 120);
        display.setEditable(false);
        display.setBackground(new Color(245, 245, 245));
        display.setFont(new Font("Monospaced", Font.PLAIN, 12));
        add(display);

        setVisible(true);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                dispose();
            }
        });
    }

    public void actionPerformed(ActionEvent e) {
        String cmd = e.getActionCommand();
        String[] data = new String[10];
        for (int i = 0; i < 10; i++) {
            data[i] = fields[i].getText().trim();  // Trim input
            System.out.println("Field " + i + ": " + data[i]);  // Debug log
        }

        switch (cmd) {
            case "Create" -> {
                CourseOperations.JavaBeans_courses_create(data);
                display.setText("Course created!");
```

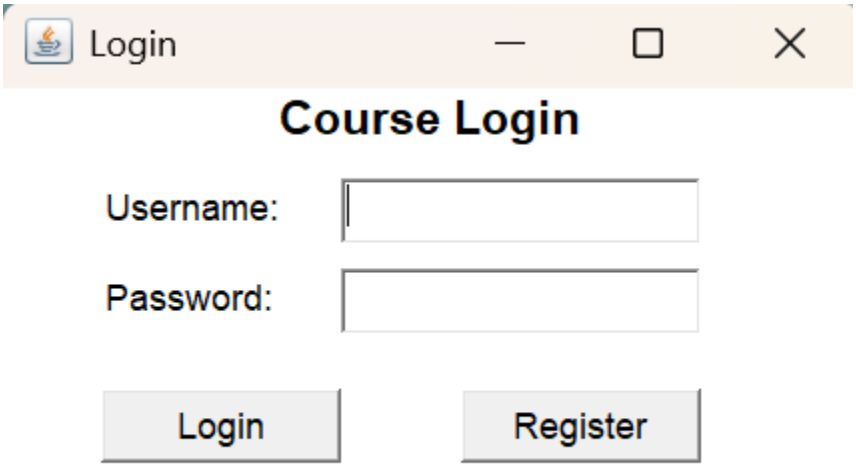
```
}
case "Update" -> {
    CourseOperations.JavaBeans_courses_update(data, originalCode);
    display.setText("Course updated!");
}
case "Delete" -> {
    CourseOperations.JavaBeans_courses_delete(data[2]);
    display.setText(" Course deleted: " + data[2]);
}
case "View All" -> {
    var all = CourseOperations.viewAll();
    display.setText("");
    for (String[] row : all) display.append(String.join(" | ", row) + "\n");
}
case "Retrive" -> {
    if (!data[0].isEmpty()) {
        var row = CourseOperations.retrieveById(data[0]);
        if (row != null) {
            for (int i = 0; i < 10; i++) fields[i].setText(row[i]);
            originalCode = row[2];
            display.setText(" Retrieved by ID");
        } else {
            display.setText(" Not found by ID.");
        }
    } else if (!data[1].isEmpty()) {
        var rows = CourseOperations.retrieveByProgramId(data[1]);
        if (!rows.isEmpty()) {
            String[] row = rows.get(0);
            for (int i = 0; i < 10; i++) fields[i].setText(row[i]);
            originalCode = row[2];
            display.setText(" Courses for Program ID:\n");
            for (String[] r : rows) display.append(String.join(" | ", r) + "\n");
        } else {
            display.setText(" No courses for Program ID.");
        }
    } else {
        display.setText("Please enter ID or Program ID");
    }
}
case "Get by Program" -> {
    var rows = CourseOperations.retrieveByProgramId(data[1]);
    display.setText("");
    for (String[] row : rows) display.append(String.join(" | ", row) + "\n");
}
```

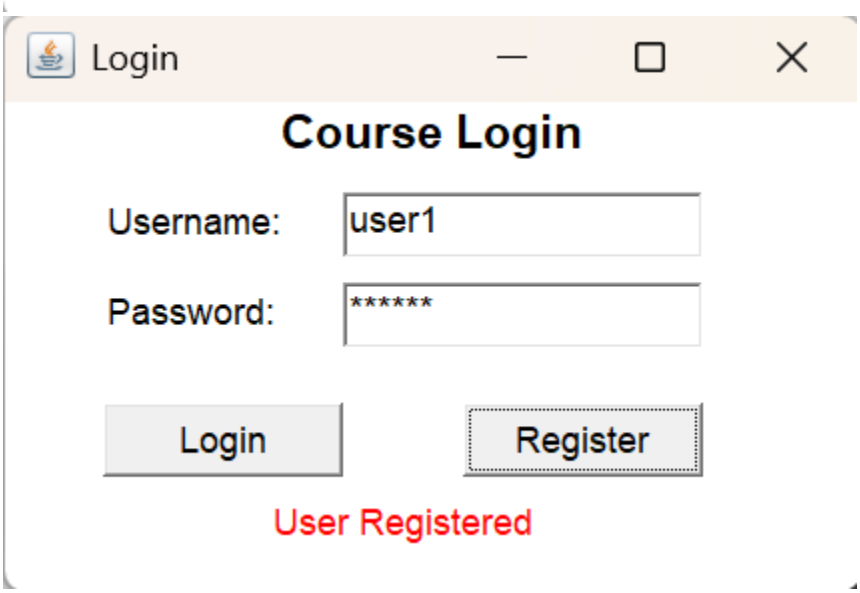
```
            case "Reset" -> {
                for (TextField field : fields) field.setText("");
                display.setText("Reset");
                originalCode = "";
            }
        }
    }
}

class CourseOperations {
    static void register(String u, String p) {
        try (Connection con = DriverManager.getConnection("jdbc:sqlite:courses.db")) {
            PreparedStatement ps = con.prepareStatement("INSERT OR IGNORE INTO users
VALUES (?, ?)");
            ps.setString(1, u);
            ps.setString(2, p);
            ps.execute();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    static boolean authenticate(String u, String p) {
        try (Connection con = DriverManager.getConnection("jdbc:sqlite:courses.db")) {
            PreparedStatement ps = con.prepareStatement("SELECT * FROM users WHERE
username=? AND password=?");
            ps.setString(1, u);
            ps.setString(2, p);
            return ps.executeQuery().next();
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }

    static void JavaBeans_courses_create(String[] d) {
        try (Connection con = DriverManager.getConnection("jdbc:sqlite:courses.db")) {
            PreparedStatement ps = con.prepareStatement("INSERT OR REPLACE INTO courses
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
            for (int i = 0; i < 10; i++) ps.setString(i + 1, d[i]);
            ps.execute();
        } catch (Exception e) {
            e.printStackTrace();
        }
```

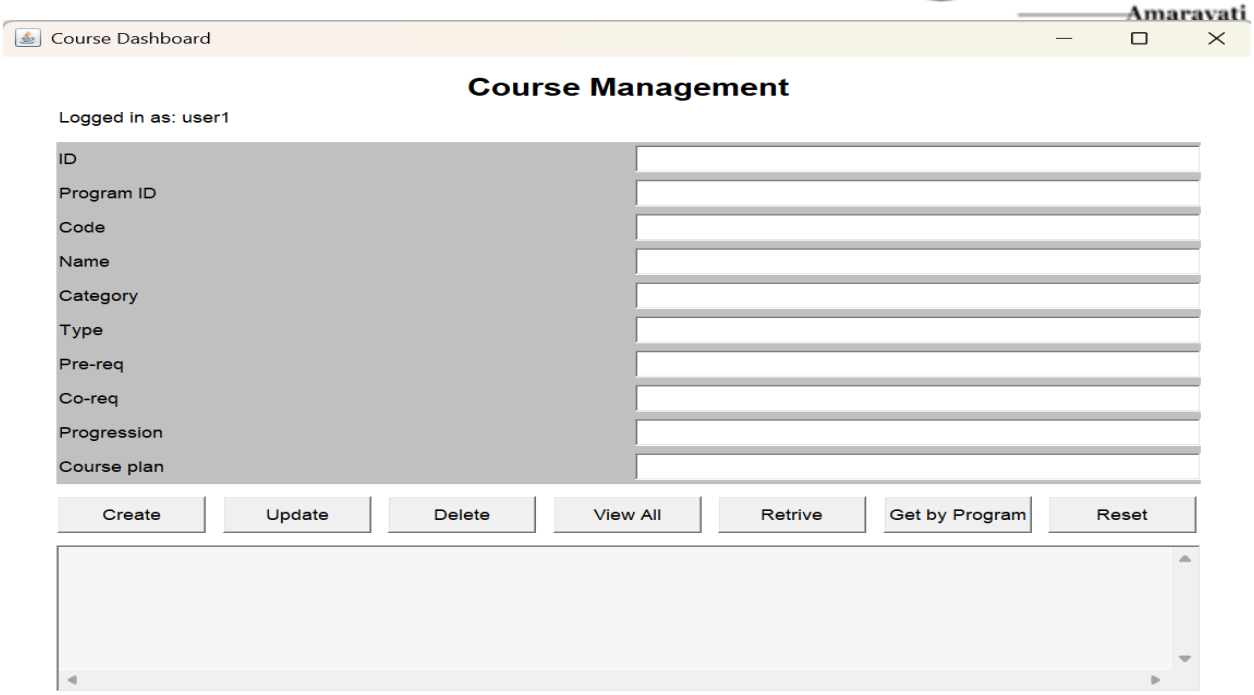
```
    }

    static void JavaBeans_courses_update(String[] d, String oldCode) {
        try (Connection con = DriverManager.getConnection("jdbc:sqlite:courses.db")) {
            PreparedStatement ps = con.prepareStatement(
                "UPDATE courses SET id=?, program_id=?, code=?, name=?, category=?, type=?,
prereq=?, coreq=?, progression=?, course_plan=? WHERE code=?"
            );
            for (int i = 0; i < 10; i++) ps.setString(i + 1, d[i]);
            ps.setString(11, oldCode);
            ps.execute();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    static void JavaBeans_courses_delete(String code) {
        try (Connection con = DriverManager.getConnection("jdbc:sqlite:courses.db")) {
            PreparedStatement ps = con.prepareStatement("DELETE FROM courses WHERE
code=?");
            ps.setString(1, code);
            ps.execute();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    static ArrayList<String[]> viewAll() {
        ArrayList<String[]> list = new ArrayList<>();
        try (Connection con = DriverManager.getConnection("jdbc:sqlite:courses.db")) {
            ResultSet rs = con.createStatement().executeQuery("SELECT * FROM courses");
            while (rs.next()) {
                String[] row = new String[10];
                for (int i = 0; i < 10; i++) row[i] = rs.getString(i + 1);
                list.add(row);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return list;
    }

    static String[] retrieveById(String id) {
        try (Connection con = DriverManager.getConnection("jdbc:sqlite:courses.db")) {
```

```
            PreparedStatement ps = con.prepareStatement("SELECT * FROM courses WHERE
id=?");
            ps.setString(1, id);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                String[] row = new String[10];
                for (int i = 0; i < 10; i++) row[i] = rs.getString(i + 1);
                return row;
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    static ArrayList<String[]> retrieveByProgramId(String programId) {
        ArrayList<String[]> list = new ArrayList<>();
        try (Connection con = DriverManager.getConnection("jdbc:sqlite:courses.db")) {
            PreparedStatement ps = con.prepareStatement("SELECT * FROM courses WHERE
program_id=?");
            ps.setString(1, programId);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                String[] row = new String[10];
                for (int i = 0; i < 10; i++) row[i] = rs.getString(i + 1);
                list.add(row);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return list;
    }
}
```

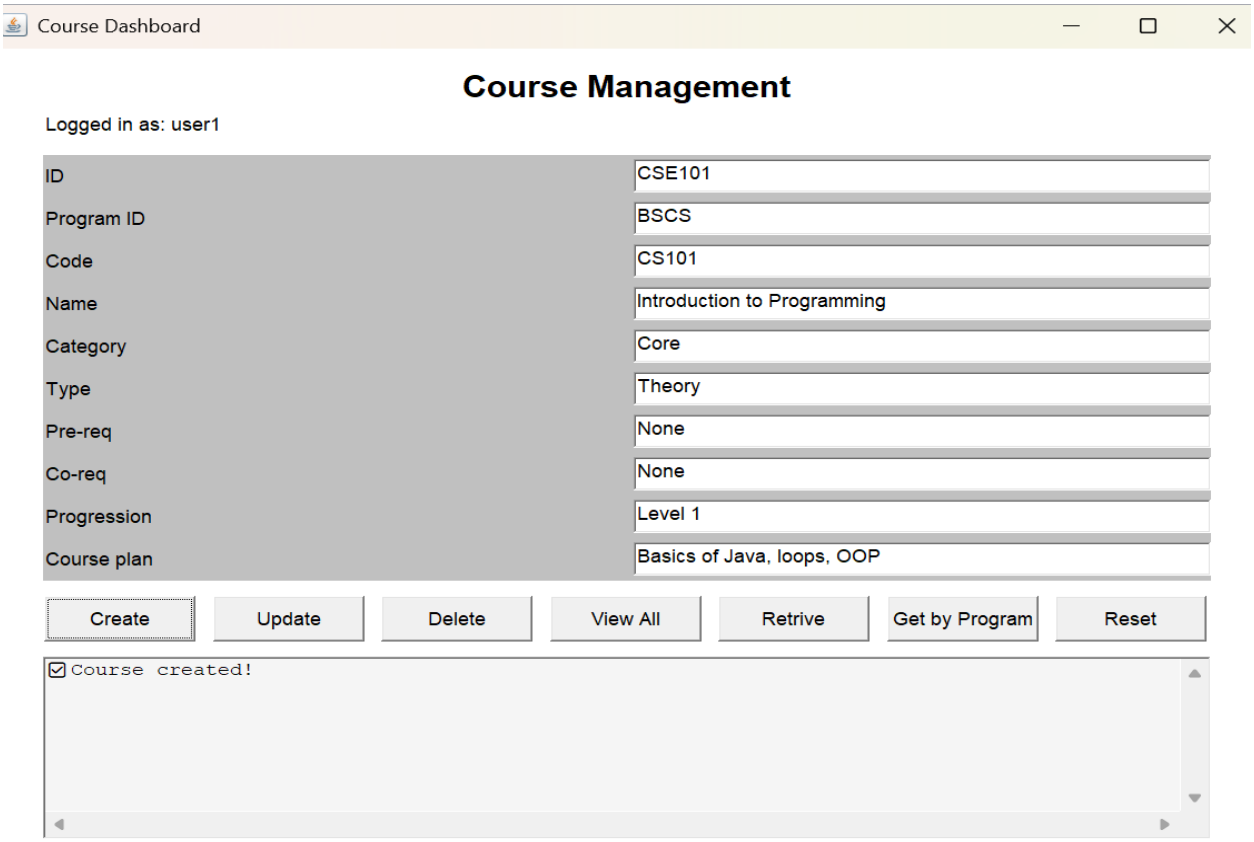
Screen Shots:


Login
Course Login
Username:
Password:
Login
Register

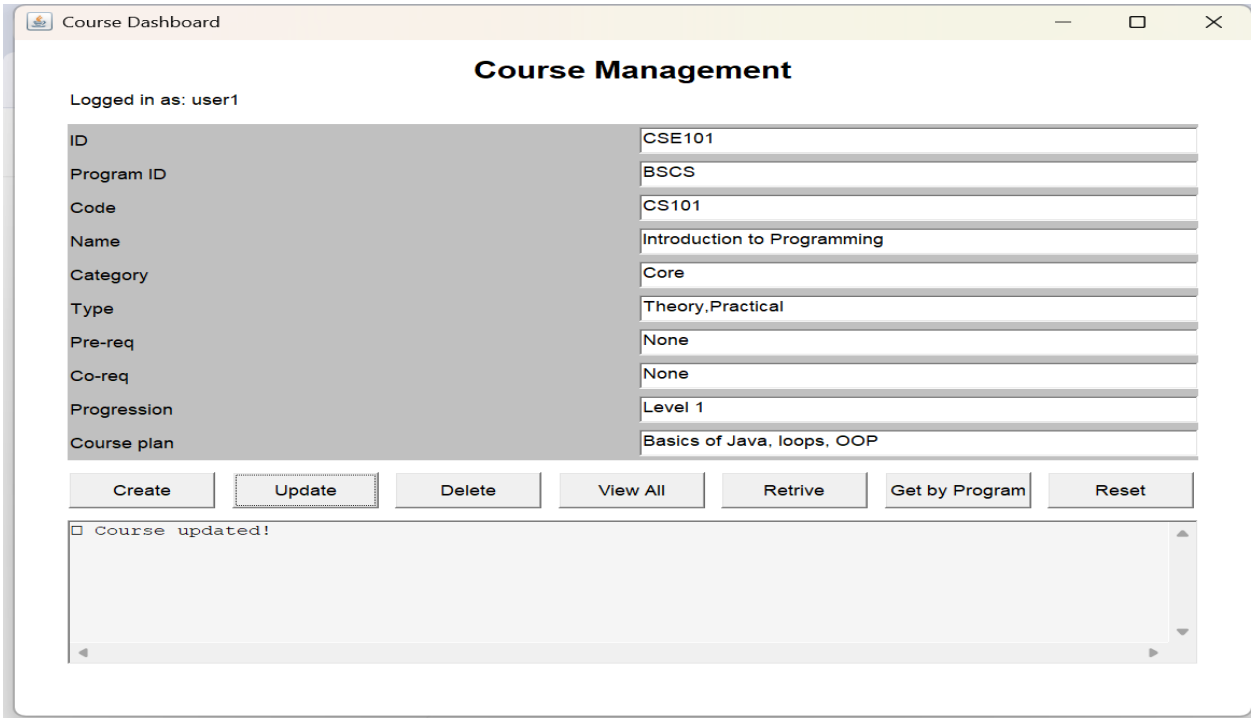

Login
Course Login
Username:
user1
Password:
******
Login
Register
User Registered

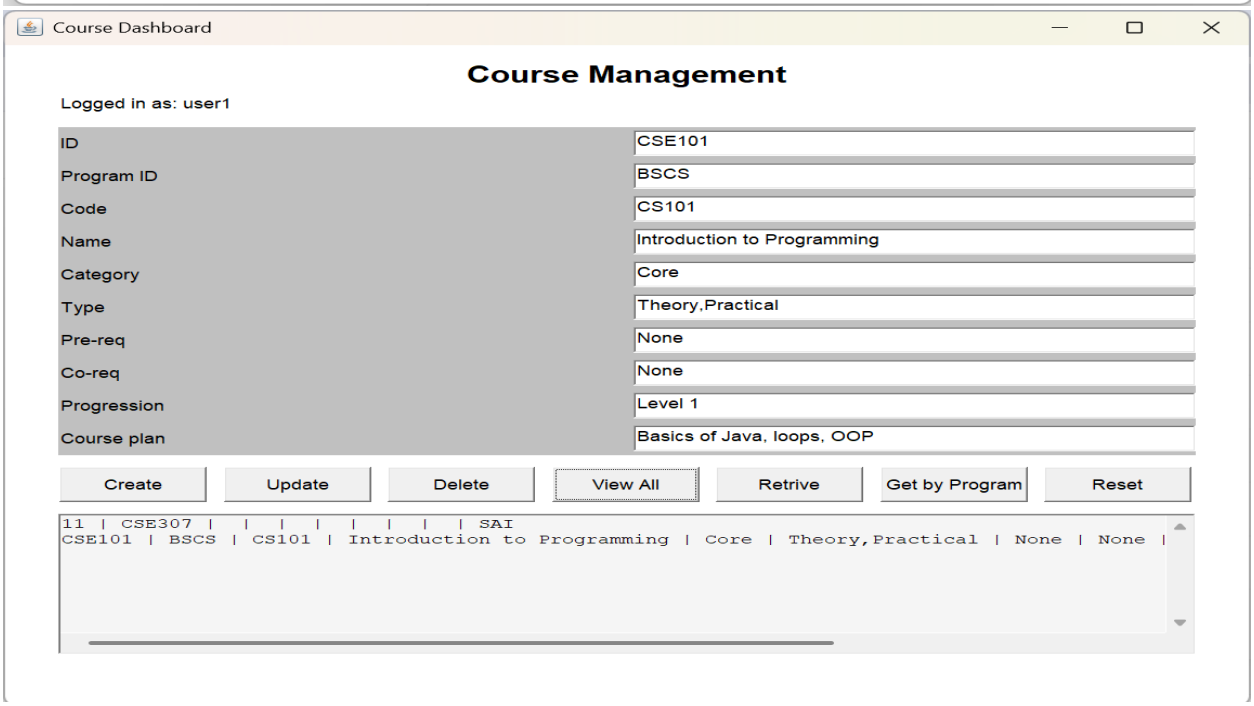

Course Dashboard
Course Management
Logged in as: user1
ID
Program ID
Code
Name
Category
Type
Pre-req
Co-req
Progression
Course plan
Create
Update
Delete
View All
Retrive
Get by Program
Reset

CREATE:

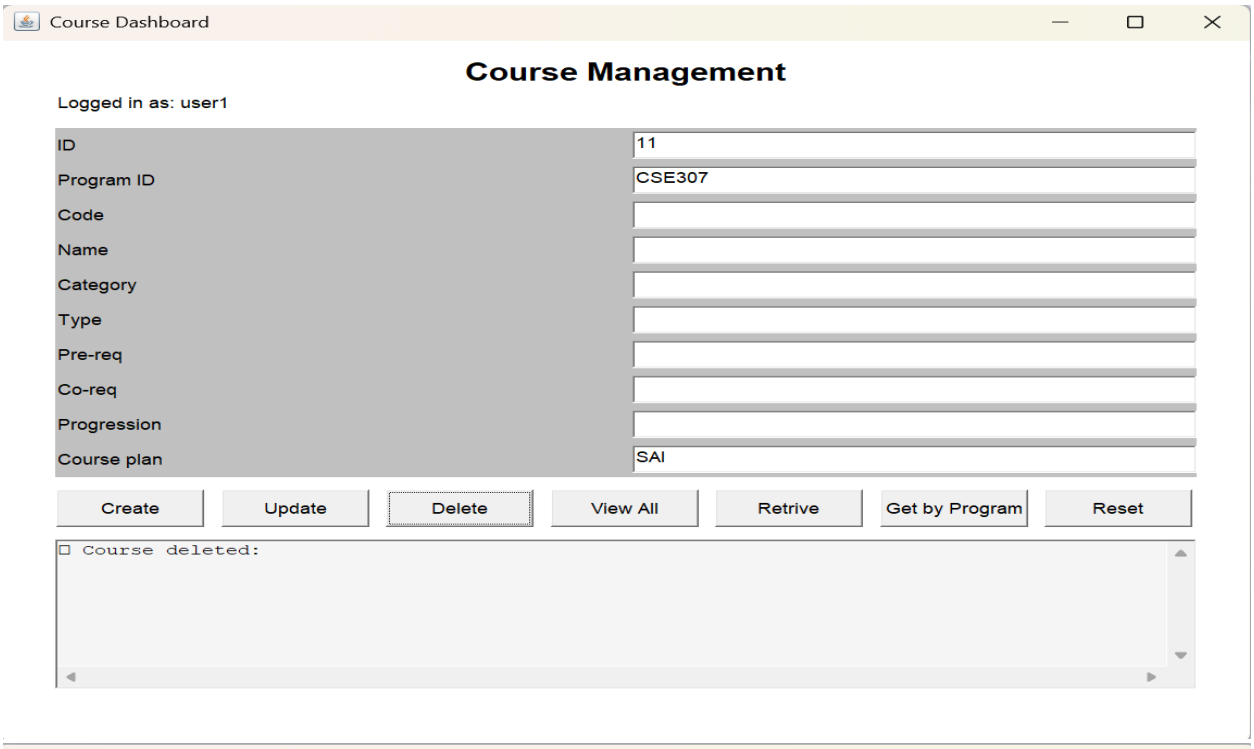

Course Dashboard
Course Management
Logged in as: user1
ID
CSE101
Program ID
BSCS
Code
CS101
Name
Introduction to Programming
Category
Core
Type
Theory
Pre-req
None
Co-req
None
Progression
Level 1
Course plan
Basics of Java, loops, OOP
Create
Update
Delete
View All
Retrive
Get by Program
Reset
Course created!

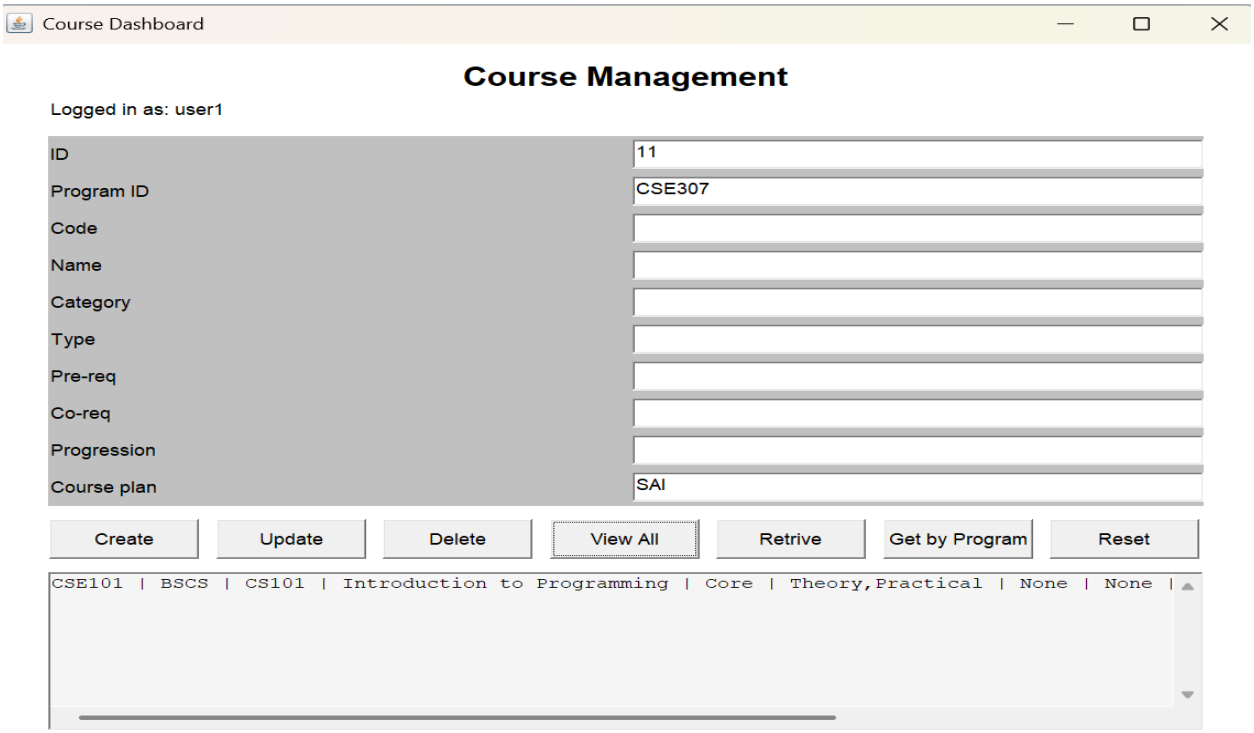
UPDATE:


Course Dashboard
Course Management
Logged in as: user1
ID
CSE101
Program ID
BSCS
Code
CS101
Name
Introduction to Programming
Category
Core
Type
Theory,Practical
Pre-req
None
Co-req
None
Progression
Level 1
Course plan
Basics of Java, loops, OOP
Create
Update
Delete
View All
Retrive
Get by Program
Reset
Course updated!

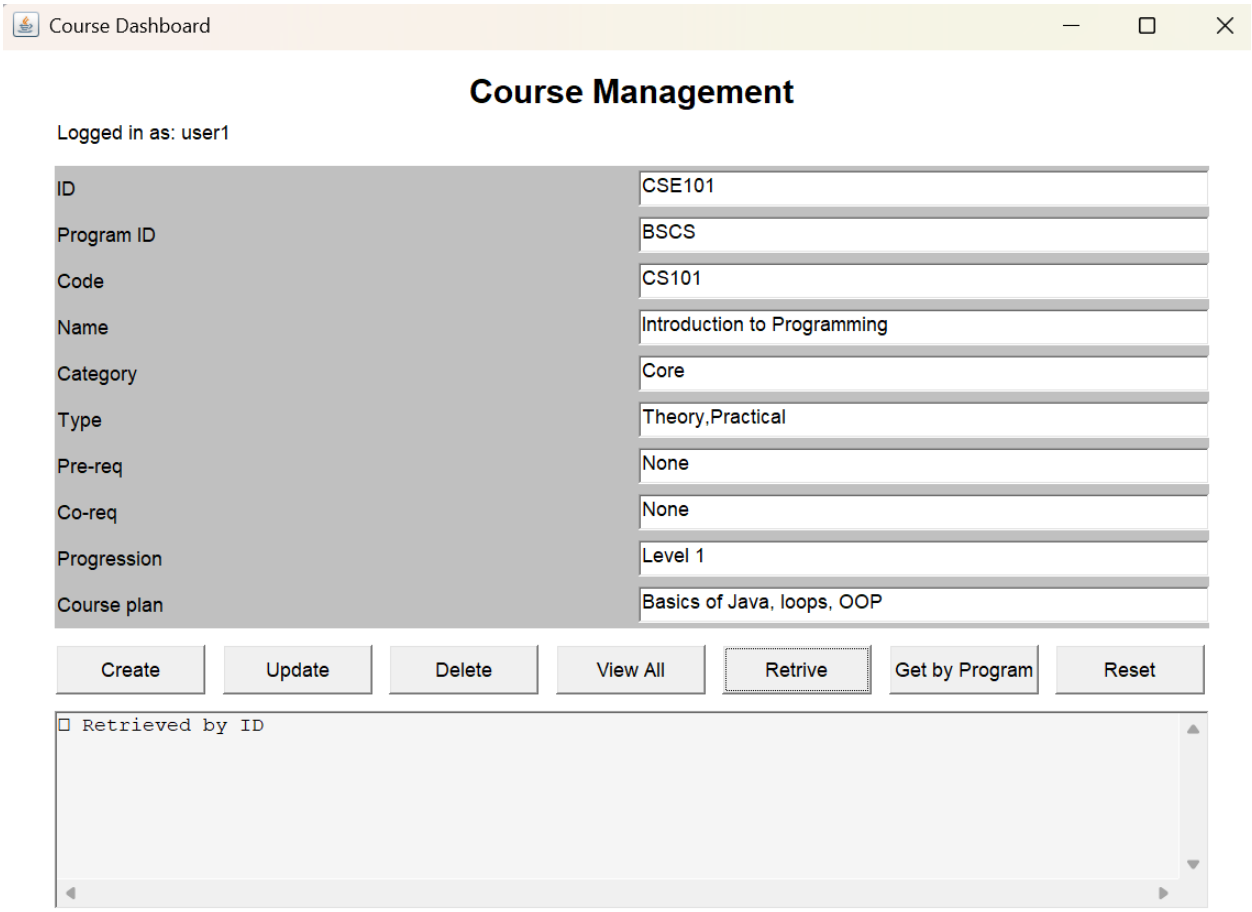

Course Dashboard
Course Management
Logged in as: user1
ID
CSE101
Program ID
BSCS
Code
CS101
Name
Introduction to Programming
Category
Core
Type
Theory,Practical
Pre-req
None
Co-req
None
Progression
Level 1
Course plan
Basics of Java, loops, OOP
Create
Update
Delete
View All
Retrive
Get by Program
Reset
11 | CSE307 | | | | | | | | SAI
CSE101 | BSCS | CS101 | Introduction to Programming | Core | Theory,Practical | None | None |

DELETE:


Course Dashboard
Course Management
Logged in as: user1
ID
11
Program ID
CSE307
Code
Name
Category
Type
Pre-req
Co-req
Progression
Course plan
SAI
Create
Update
Delete
View All
Retrive
Get by Program
Reset
Course deleted:


Course Dashboard
Course Management
Logged in as: user1
ID
11
Program ID
CSE307
Code
Name
Category
Type
Pre-req
Co-req
Progression
Course plan
SAI
Create
Update
Delete
View All
Retrive
Get by Program
Reset
CSE101 | BSCS | CS101 | Introduction to Programming | Core | Theory,Practical | None | None |

RETRIVE:


Course Dashboard
Course Management
Logged in as: user1
ID
CSE101
Program ID
BSCS
Code
CS101
Name
Introduction to Programming
Category
Core
Type
Theory,Practical
Pre-req
None
Co-req
None
Progression
Level 1
Course plan
Basics of Java, loops, OOP
Create
Update
Delete
View All
Retrive
Get by Program
Reset
Retrieved by ID

CONCLUSION:

This project effectively demonstrates the implementation of a full-fledged Java AWT-based GUI application integrated with SQLite for managing course data. It supports complete CRUD (Create, Read, Update, Delete) operations, allowing users to register, log in, and manage course records such as course codes, program IDs, prerequisites, and more through an intuitive and interactive interface.The structure combines clean form-based input with functional logic, using Java's Frame, TextField, and Button components. The integration of JDBC with SQLite ensures reliable data persistence, while features like login authentication, course retrieval by ID or Program ID, and form reset add usability and depth.

The separation of GUI and database logic via the CourseOperations class keeps the project modular and maintainable. Future enhancements can include features such as:Input validation, Enhanced UI with Swing or JavaFX,User roles (e.g., admin vs student).Overall, this project lays a solid foundation for educational course management systems in Java.