

INTERNSHIP REPORT

ON

JAVA PROGRAMING

(Project Employee Management System)

Submitted in partial fulfilment towards the award of degree in

B.TECH in Computer Science and Engineering

SESSION 2024-25



**NITRA TECHNICAL CAMPUS,
GHAZIABAD**

(college Code – 802)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY:

NAME: PRABHAT MAURYA

ROLL NO: 2308020100060

BRANCH: CSE

SEM/YEAR: III/II YEAR

INDEX

S No.	CONTENT	PAGE
1	ABOUT ORGANIZATION	3
2	ACKNOWLEDGEMENT	4
3	CERTIFICATION	5
4	DECLARATION	6
5	INTRODUCTION	7
6	INSTALLATION	9
7	KEY FEATURES	12
8	JAVA GUI	14
9	MYSQL	17
10	PROJECT CODE	18
11	PROJECT VIEW	38
12	CONCLUSION	39
13	REFERENCES	40

ABOUT ORGANIZATION

(NPTEL)

The National Programme on Technology Enhanced Learning (NPTEL) is an initiative by India's top IITs (Indian Institutes of Technology) and IISc (Indian Institute of Science). It provides online courses, certification, and learning resources in engineering, science, and humanities. NPTEL aims to improve the quality of education and make learning accessible globally. It offers video lectures, assignments, and exams to help students and professionals enhance their knowledge and skills.

ACKNOWLEDGEMENT

It is my pleasure to be indebted to various people, who directly contributed in the development of this work and who influenced my thinking , behavior , and acts during the course of study.

I am also thankful to my friends those give me support to helpful to complete this project and solve my all difficulties.

I am taking this opportunity to express out gratefulness to the Management , Teachers and staff.

(Sign of Student)

PRABHAT MAURYA

2308020100060

CSE 2nd Year

CERTIFICATION



Elite

NPTEL Online Certification

(Funded by the MoE, Govt. of India)



This certificate is awarded to

PRABHAT MAURYA

for successfully completing the course

Programming in Java

with a consolidated score of **64** %

Online Assignments	22.85/25	Proctored Exam	41.51/75
--------------------	----------	----------------	----------

Total number of candidates certified in this course: **14693**

Jan-Apr 2024

(12 week course)

Prof. Haimanti Banerji
Coordinator, NPTEL
IIT Kharagpur



Indian Institute of Technology Kharagpur



Roll No: NPTEL24CS43S954304307

To verify the certificate



No. of credits recommended: 3 or 4

DECLARATION

I, **PRABHAT MAURYA (2308020100060)** hereby declare that the presented report of online course titled “**JAVA PROGRAMING**” is uniquely prepared by me after completion of **12 WEEKS** on the platform of **NPTEL**.

I also confirm that the report is only prepared for my academic requirement , not for any other purpose. It might not be used with the interest of the opposite party of the any organization.

(Sign of Student)

PRABHAT MAURYA

2308020100060

CSE 2nd Year

INTRODUCTION

Provide a brief description of the project and its purpose. What problem does it solve? What functionality does it offer?

Example: This project is a **Task Manager** application written in Java that allows users to create, update, and delete tasks. The application offers features for scheduling tasks, assigning priorities, and marking tasks as completed.

What is Java?

- ❖ Java is a high-level, object-oriented programming language.
- ❖ Developed by James Gosling at Sun Microsystems (now owned by Oracle).
- ❖ Designed for portability across platforms (Write Once, Run Anywhere).
- ❖ Used for building a wide range of applications (web, mobile, enterprise).

Features of Java

- ❖ **Platform Independence:** Java programs can run on any system with a JVM (Java Virtual Machine).
- ❖ **Object-Oriented:** Everything in Java is treated as an object.
- ❖ **Multithreading:** Supports multiple threads for efficient execution.
- ❖ **Rich API:** Extensive libraries for developers to build applications.
- ❖ **Automatic Garbage Collection:** Memory management is handled automatically.

Basic Java Syntax

❖ **Class Definition:**

```
public class MyFirstProgram {  
    public static void main(String[] args) {  
        System.out.println("Hello, Java!");  
    }  
}
```

}

❖ **Main Method:** The entry point for any Java program.

Statements and Semicolons: Each statement in Java ends with a semicolon

List all the technologies used in the project.

❖ **Programming Language:** Java 17 (or any version you are using)

❖ **Libraries/Frameworks:**

- JavaFX for GUI (if applicable)
- JDBC for database connectivity (if applicable)
- JUnit for testing

❖ **Database:** MySQL / PostgreSQL / SQLite (or none if no database)

❖ **Build Tool:** Maven / Gradle (or none)

INSTALLATION

This section should include steps on how to set up the project locally.

Prerequisites:

- Install **JDK 17** or above
- Install **Maven** or **Gradle** (if using a build tool)
- **IDE** (e.g., IntelliJ IDEA, Eclipse, etc.)

Installation Steps:

1. Clone the repository from GitHub:
`git clone https://github.com/yourusername/projectname.git`
2. Navigate to the project directory: `cd projectname`
3. If using Maven or Gradle, run the following to build the project:
 - **Maven:** `mvn clean install`
 - **Gradle:** `gradle build`
4. If the project uses a database, configure your database connection in the `application.properties` (or similar configuration file).
5. To run the project:
 - For command-line: `java -jar target/your-jar-file.jar`
 - Or through IDE by running the `Main.java` class.

Provide detailed explanations of key sections of the code, algorithms, or methods that are central to the application.

Example:

- **Main.java:** This class contains the `main()` method that initializes the application, loads the user interface, and sets up event listeners.
- **Task.java:** This class represents a task object and contains fields such as `taskName`, `description`, `priority`, and methods to get and set these values.

```
public class Task {  
  
    private String taskName;
```

```
private String description;
```

```
private String priority;
```

```
public Task(String taskName, String description, String priority) {
```

```
    this.taskName = taskName;
```

```
    this.description = description;
```

```
    this.priority = priority;
```

```
}
```

```
// Getters and Setters
```

```
public String getTaskName() {
```

```
    return taskName;
```

```
}
```

```
public void setTaskName(String taskName) {
```

```
    this.taskName = taskName;
```

```
}
```

```
public String getDescription() {
```

```
    return description;
```

```
}
```

```
public void setDescription(String description) {  
    this.description = description;  
}  
  
public String getPriority() {  
    return priority;  
}  
  
public void setPriority(String priority) {  
    this.priority = priority;  
}  
}
```

KEY FEATURES

This section lists and explains the main features of the project.

Example:

- **Task Management:** Users can create, update, and delete tasks.
- **Prioritization:** Each task can be assigned a priority (low, medium, high).
- **Search:** The application allows searching for tasks by title or priority.
- **Database Integration:** The application stores task data in an SQLite database.

Data Types in Java

❖ Primitive Data Types:

int, float, char, boolean, double, etc.

❖ Non-Primitive Data Types:

- Arrays, Strings, Classes, Interfaces.

Control Structures in Java

❖ Conditionals:

- ❖ if, else if, else, switch.

❖ Loops:

- ❖ for, while, do-while.

❖ Break and Continue: Used to control loop flow.

Object-Oriented Concepts in Java

❖ Classes and Objects: Blueprint for creating objects.

❖ Encapsulation: Wrapping data and methods in a class.

❖ Inheritance: Ability to derive new classes from existing ones.

❖ Polymorphism: Objects can take many forms (method overloading and overriding).

❖ Abstraction: Hiding complexity and exposing only essential features.

Java Libraries

- ❖ **Library** in Java refers to a collection of pre-written classes and methods, packaged together for easy reuse.
- ❖ **Example:**
- ❖ **Purpose:** Libraries save time and effort by providing common functionalities like data structures, networking, or utilities.
 - ❖ The Java Standard Library includes essential classes like **`java.util.ArrayList`**, **`java.lang.String`**.

Java Use Cases

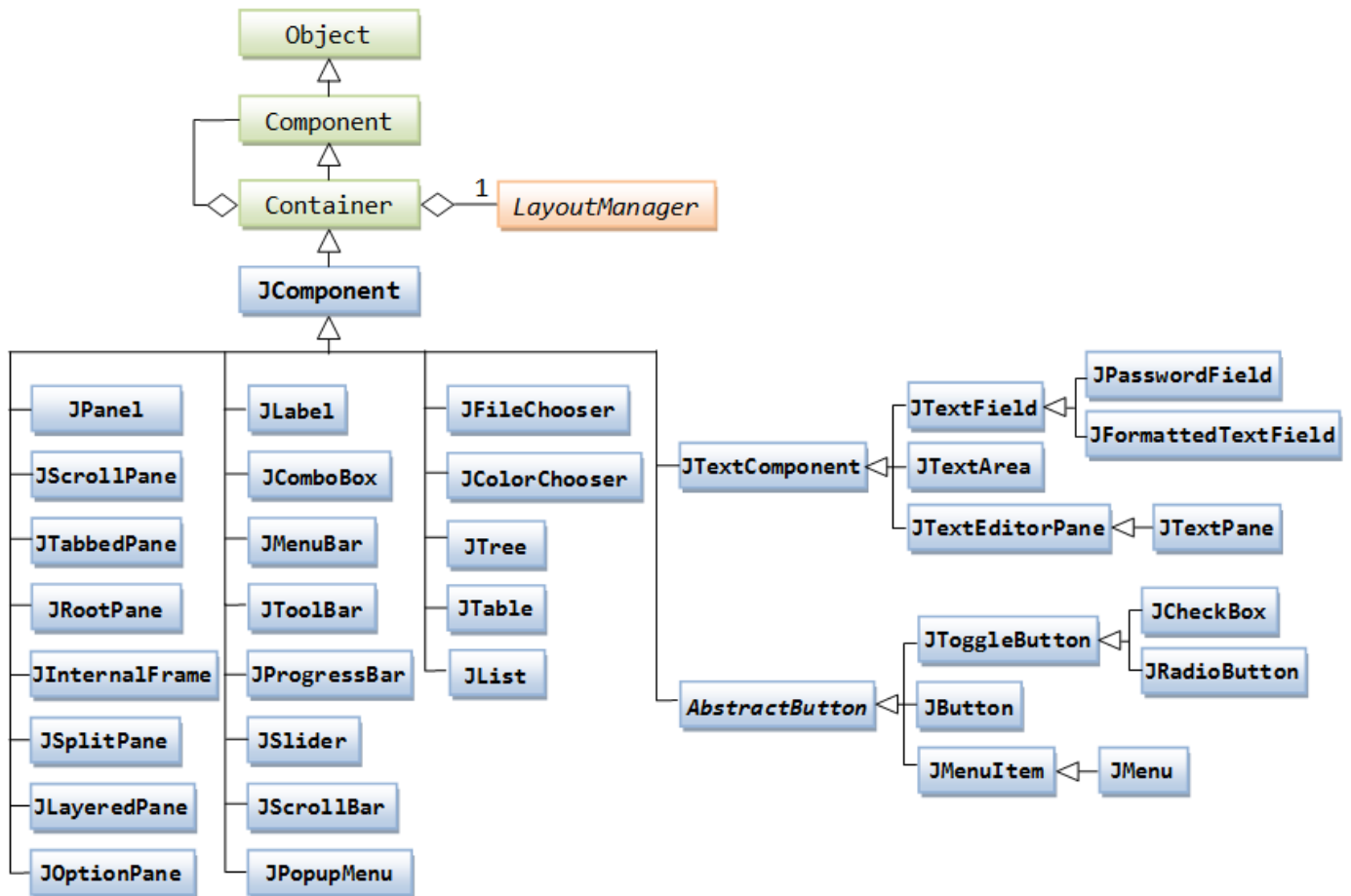
- ❖ **Web Development:** Building dynamic websites (using frameworks like Spring).
- ❖ **Mobile Development:** Android applications.
- ❖ **Enterprise Applications:** Large-scale business solutions (using Java EE).
- ❖ **Big Data:** Used in big data technologies like Apache Hadoop.
- ❖ **Games & GUI:** JavaFX for graphics and game development.

JAVA GUI

Introduction to Java GUI

- ❖ **Java GUI** allows developers to create interactive, graphical user interfaces for Java applications.
- ❖ Java provides several libraries to build GUI applications, with **Swing** and **JavaFX** being the most popular.





Key Components of Java GUI

- ❖ **Swing:** A part of the Java Foundation Classes (JFC), Swing is used for creating lightweight and platform-independent GUI applications.
- ❖ **JavaFX:** A newer alternative to Swing for building rich, modern UIs.

Supports advanced features like animations, 2D/3D graphics, and media

- ❖ **AWT (Abstract Window Toolkit):** Basic GUI components; platform-dependent..

```
import javax.swing.*;
public class MyGUI {
    public static void main(String[] args) {
        JFrame frame = new JFrame("My First GUI");
        JButton button = new JButton("Click Me!");
        frame.add(button);
        frame.setSize(300, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```



Use Cases:

- ❖ Desktop applications (e.g., calculators, games).
- ❖ IDEs and tools (e.g., NetBeans, IntelliJ IDEA).

Advantages:

- ❖ Platform-independent.
- ❖ Rich libraries for easy development.

MYSQL



What is MySQL?

- ❖ Open-source **Relational Database Management System** (RDBMS).
- ❖ Developed by Oracle Corporation.
- ❖ Uses **Structured Query Language (SQL)** for database operations.

Basic SQL Operations:

❖ connection

```
mysql -u root -p
```

❖ create database

```
CREATE DATABASE my_database;
```

❖ create table

```
CREATE TABLE users ( id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), email VARCHAR(100));
```

❖ Insert data

```
INSERT INTO users (name, email) VALUES ('Name', 'joky@example.com');
```

❖ Query data

```
SELECT * FROM users;
```

Applications:

- ❖ Web applications (e.g., WordPress, Joomla).
- ❖ eCommerce platforms.
- ❖ Analytics and reporting tools.

PROJECT CODE

MYSQL DATABASE CODE:

```
create database empolyeeData;  
  
CREATE TABLE employee (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(100),  
    gender VARCHAR(10),  
    dob DATE,  
    marital_status VARCHAR(20),  
    email VARCHAR(100),  
    mobile VARCHAR(15),  
    department VARCHAR(50),  
    salary DECIMAL(10, 2)  
);
```

JAVA CODE:

```
import java.awt.*;  
import java.sql.*;  
import java.awt.event.*;  
import java.text.SimpleDateFormat;  
import javax.swing.*;  
import javax.swing.table.DefaultTableCellRenderer;  
import javax.swing.table.DefaultTableModel;  
  
// Conection  
  
public class Employee {  
    private static Connection getConnection() throws SQLException {  
        String url = "jdbc:mysql://localhost:3306/empolyeeData";  
        String user = "root";
```

```
String password = "Prabhat@1234#";  
return DriverManager.getConnection(url, user, password);  
}
```

```
public static int ide;
```

```
public static void main(String[] args) {
```

```
    JFrame f = new JFrame("Employee Data Entry");
```

```
    JLabel ltag = new JLabel("Employee Data Entry");
```

```
    ltag.setBounds(90, 20, 300, 50);
```

```
    ltag.setFont(new Font("Arial", Font.BOLD, 24));
```

```
    f.add(ltag);
```

```
    JLabel lname = new JLabel("Name:");
```

```
    JLabel lemail = new JLabel("Email:");
```

```
    JLabel lnumber = new JLabel("Phone Number:");
```

```
    lname.setBounds(20, 80, 80, 20);
```

```
    lemail.setBounds(20, 200, 80, 20);
```

```
    lnumber.setBounds(20, 240, 210, 20);
```

```
    JTextField tname = new JTextField();
```

```
    JTextField temail = new JTextField();
```

```
    JTextField tnumber = new JTextField(Integer.BYTES);
```

```
    tname.setBounds(130, 80, 300, 35);
```

```
    temail.setBounds(130, 200, 300, 35);
```

```
    tnumber.setBounds(130, 240, 300, 35);
```

```
    // bob
```

```
    JLabel ldob = new JLabel("Enter DOB:");
```

```
JLabel ldob2 = new JLabel("(yyyy-MM-dd)");
```

```
f.add(ldob2);
```

```
ldob.setBounds(20, 120, 210, 15);
```

```
ldob2.setBounds(20, 140, 210, 15);
```

```
JFormattedTextField tdob = new JFormattedTextField(new SimpleDateFormat("yyyy-MM-dd"));
```

```
tdob.setColumns(10);
```

```
tdob.setBounds(130, 120, 300, 35);
```

```
// gender
```

```
JLabel lgender = new JLabel("Gender:");
```

```
lgender.setBounds(20, 160, 210, 20);
```

```
f.add(lgender);
```

```
JRadioButton male = new JRadioButton("Male");
```

```
JRadioButton female = new JRadioButton("Female");
```

```
male.setBounds(130, 160, 150, 35);
```

```
female.setBounds(280, 160, 150, 35);
```

```
ButtonGroup bgender = new ButtonGroup();
```

```
bgender.add(male);
```

```
bgender.add(female);
```

```
f.add(male);
```

```
f.add(female);
```

```
// marital
```

```
JLabel lmarital = new JLabel("Marital Status:");
```

```
lmarital.setBounds(20, 280, 210, 20);
```

```
f.add(lmarital);
```

```
JRadioButton married = new JRadioButton("Married");
```

```

JRadioButton unmarried = new JRadioButton("Unmarried");
married.setBounds(130, 280, 150, 35);
unmarried.setBounds(280, 280, 150, 35);

ButtonGroup bmarital = new ButtonGroup();
bmarital.add(married);
bmarital.add(unmarried);
f.add(married);
f.add(unmarried);

// Department
String Dep[] = { "IT Department", "HR Department", "Accountent Department", "Mnagement
Department", "Other" };
JComboBox jdepartment = new JComboBox(Dep);
JLabel ldepartment = new JLabel("Select Department:");
ldepartment.setBounds(20, 320, 210, 20);
jdepartment.setBounds(130, 320, 300, 35);
f.add(ldepartment);
f.add(jdepartment);

// Salary
JLabel lsalary = new JLabel("Salary:");
lsalary.setBounds(20, 360, 210, 20);
JTextField tsalary = new JTextField();
tsalary.setBounds(130, 360, 300, 35);
f.add(lsalary);
f.add(tsalary);

// Button save
JButton save = new JButton("Save");
save.setBounds(200, 400, 100, 35);
f.add(save);

```

```
// Search
JButton search = new JButton("Search");
search.setBounds(320, 400, 100, 35);
f.add(search);

// Clear
JButton clear = new JButton("Clear");
clear.setBounds(200, 440, 100, 35);
f.add(clear);

// add
JButton add = new JButton("Add+");
add.setBounds(80, 400, 100, 35);

f.add(add);

// updata
JButton update = new JButton("Update");
update.setBounds(200, 400, 100, 35);
update.setVisible(false);
f.add(update);

JButton show = new JButton();
show.setVisible(false);
f.add(show);

// Profile Panel
JPanel profilePanel = new JPanel();
profilePanel.setBounds(500, 30, 600, 420);
profilePanel.setLayout(null);
profilePanel.setBorder(BorderFactory.createTitledBorder("Profile Details"));

// Profile Icon
JLabel profileIcon = new JLabel();
```

```

profileIcon.setBounds(290, 15, 50, 50);
profileIcon.setIcon(new ImageIcon(new ImageIcon("pro.png").getImage()
    .getScaledInstance(50, 50, Image.SCALE_SMOOTH)));
profilePanel.add(profileIcon);

// profile logo
JLabel profileIcon2 = new JLabel();
profileIcon2.setBounds(30, 20, 100, 100);
profileIcon2.setIcon(new ImageIcon(new ImageIcon("logo.png").getImage()
    .getScaledInstance(100, 100, Image.SCALE_SMOOTH)));
profilePanel.add(profileIcon2);

// neme
JLabel pname = new JLabel("Prabhat Maurya");
pname.setBounds(260, 60, 400, 35);
pname.setFont(new Font("Arial", Font.TYPE1_FONT, 25));
pname.setForeground(Color.gray);
profilePanel.add(pname);

// Button edit
JButton edit = new JButton("Edit");
edit.setBounds(470, 370, 70, 25);
profilePanel.add(edit);
edit.setForeground(Color.white);
edit.setBackground(Color.lightGray);

// Radiobuttonstatus
JRadioButton bactive = new JRadioButton("Active");
JRadioButton binactive = new JRadioButton("Inactive");
bactive.setBounds(400, 370, 90, 25);
binactive.setBounds(500, 370, 90, 25);
ButtonGroup bstatus = new ButtonGroup();
bactive.setForeground(Color.green);
binactive.setForeground(Color.red);

```

```
bstatus.add(bactive);
bstatus.add(binactive);
profilePanel.add(bactive);
profilePanel.add(binactive);
bactive.setVisible(false);
binactive.setVisible(false);
```

```
// Labels for Details
```

```
JLabel nameLabel = new JLabel("Name: ");
JLabel statusLabel = new JLabel("Active");
```

```
JLabel idLabel = new JLabel("Employee ID: ");
JLabel genderLabel = new JLabel("Gender: ");
JLabel dobLabel = new JLabel("DOB: ");
JLabel maritalLabel = new JLabel("Marital Status: ");
JLabel emailLabel = new JLabel("Email: ");
JLabel mobileLabel = new JLabel("Mobile: ");
JLabel departmentLabel = new JLabel("Department: ");
JLabel salaryLabel = new JLabel("Salary: ");
```

```
// Positioning Labels
```

```
statusLabel.setBounds(280, 80, 400, 35);

int y = 140;

int labelHeight = 25;

nameLabel.setBounds(20, y, 400, labelHeight);
idLabel.setBounds(20, y += 30, 400, labelHeight);

genderLabel.setBounds(20, y += 30, 400, labelHeight);
dobLabel.setBounds(20, y += 30, 400, labelHeight);
maritalLabel.setBounds(20, y += 30, 400, labelHeight);
```



```

emailLabel.setBounds(20, y += 30, 400, labelHeight);
mobileLabel.setBounds(20, y += 30, 400, labelHeight);
departmentLabel.setBounds(20, y += 30, 400, labelHeight);
salaryLabel.setBounds(20, y += 30, 400, labelHeight);

// Add Labels to Panel
profilePanel.add(nameLabel);
profilePanel.add(idLabel);
profilePanel.add(genderLabel);
profilePanel.add(dobLabel);
profilePanel.add(maritalLabel);
profilePanel.add(emailLabel);
profilePanel.add(mobileLabel);
profilePanel.add(departmentLabel);
profilePanel.add(salaryLabel);
profilePanel.add(statusLabel);

profilePanel.setVisible(false);

// Employee Table
String[] columnNames = { "ID", "Name", "Gender", "DOB", "Email", "Phone", "Department", "Salary",
"Status" };

DefaultTableModel tableModel = new DefaultTableModel(columnNames, 0);
JTable employeeTable = new JTable(tableModel);
JScrollPane tableScroll = new JScrollPane(employeeTable);
tableScroll.setBounds(20, 500, 1200, 170);
applyStatusColumnRenderer(employeeTable, 8);

f.add(tableScroll);

f.add(profilePanel);

```

```

refreshTable(tableModel);

f.add(lname);
f.add(ldob);
f.add(lnumber);
f.add(lemail);
f.add(tname);
f.add(tnumber);
f.add(tdob);
f.add(temail);

f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

f.setSize(1500, 900);

f.setLayout(null);

f.setVisible(true);

// Backed code

save.addActionListener(e -> {
    try (Connection conn = getConnection()) {
        String query = "INSERT INTO employee (name, gender, dob, marital_status, email, mobile,
department, salary) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

        PreparedStatement ps = conn.prepareStatement(query);

        ps.setString(1, tname.getText());
        ps.setString(2, male.isSelected() ? "Male" : "Female");
        ps.setDate(3, Date.valueOf(tdob.getText()));
        ps.setString(4, married.isSelected() ? "Married" : "Unmarried");
        ps.setString(5, temail.getText());
        ps.setString(6, tnumber.getText());
        ps.setString(7, (String) jdepartment.getSelectedItem());
        ps.setDouble(8, Double.parseDouble(tsalary.getText()));
    }
});

```

```

ps.executeUpdate();

JOptionPane.showMessageDialog(f, "Employee Saved!");

} catch (Exception ex) {
    JOptionPane.showMessageDialog(f, "Error: " + ex.getMessage());
}

// show
try {
    Connection conn = getConnection();

    String query = "SELECT * FROM employee\n" +
        "ORDER BY id DESC\n" +
        "LIMIT 1;";

    PreparedStatement ps = conn.prepareStatement(query);

    ResultSet rs = ps.executeQuery();

    // profile show part
    if (rs.next()) {
        // Update UI components

        ide = rs.getInt("id");

        String status = rs.getString("status");

        if (status.equals("Active")) {
            statusLabel.setForeground(Color.GREEN);
        } else {
            statusLabel.setForeground(Color.RED);
        }

        statusLabel.setText(status);

        nameLabel.setText("Name: " + rs.getString("name"));

        pname.setText(rs.getString("name"));

        idLabel.setText("Employee ID: " + rs.getInt("id"));
    }
}

```

```

        genderLabel.setText("Gender: " + rs.getString("gender"));
        dobLabel.setText("DOB: " + rs.getDate("dob").toString());
        maritalLabel.setText("Marital Status: " + rs.getString("marital_status"));
        emailLabel.setText("Email: " + rs.getString("email"));
        mobileLabel.setText("Mobile: " + rs.getString("mobile"));
        departmentLabel.setText("Department: " + rs.getString("department"));
        salaryLabel.setText("Salary: " + rs.getDouble("salary"));
        profilePanel.setVisible(true);
        clear.doClick();
        refreshTable(tableModel);

    } else {
        JOptionPane.showMessageDialog(f, "Empolyee Not found");

    }
    conn.close();
} catch (SQLException e1) {
    e1.printStackTrace();
    JOptionPane.showMessageDialog(f, "Database error: " + e1.getMessage());
}

});

clear.addActionListener(e -> {
    tname.setText("");
    temail.setText("");
    tnumber.setText("");
    tdob.setText("");
    bgender.clearSelection();
    bmarital.clearSelection();
    jdepartment.setSelectedIndex(0);

```

```

        tsalary.setText("");
    });

search.addActionListener(e -> {
    String id = JOptionPane.showInputDialog(f, "Enter Employee ID:");
    try (Connection conn = getConnection()) {
        String query = "SELECT * FROM employee WHERE id = ?";
        PreparedStatement ps = conn.prepareStatement(query);
        ps.setInt(1, Integer.parseInt(id));
        ResultSet rs = ps.executeQuery();

        if (rs.next()) {
            // Update UI components
            ide = rs.getInt("id");
            String status = rs.getString("status");
            if (status.equals("Active")) {
                statusLabel.setForeground(Color.GREEN);
            } else {
                statusLabel.setForeground(Color.RED);
            }

            statusLabel.setText(status);
            nameLabel.setText("Name: " + rs.getString("name"));
            pname.setText(rs.getString("name"));
            idLabel.setText("Employee ID: " + rs.getInt("id"));
            genderLabel.setText("Gender: " + rs.getString("gender"));
            dobLabel.setText("DOB: " + rs.getDate("dob").toString());
            maritalLabel.setText("Marital Status: " + rs.getString("marital_status"));
            emailLabel.setText("Email: " + rs.getString("email"));
            mobileLabel.setText("Mobile: " + rs.getString("mobile"));
            departmentLabel.setText("Department: " + rs.getString("department"));
        }
    }
});

```

```

        salaryLabel.setText("Salary: " + rs.getDouble("salary"));
        profilePanel.setVisible(true);

    } else {
        JOptionPane.showMessageDialog(f, "Empolyee Not found");

    }

    conn.close();
} catch (SQLException e1) {
    e1.printStackTrace();
    JOptionPane.showMessageDialog(f, "Database error: " + e1.getMessage());
}
});

update.addActionListener(e -> {

    try (Connection conn = getConnection()) {

        String query = "UPDATE employee SET name = ?, gender = ?, dob = ?, marital_status = ?, email = ?,
mobile = ?, department = ?, salary = ?, status = ? WHERE id = ?";
        PreparedStatement ps = conn.prepareStatement(query);

        ps.setString(1, tname.getText());
        ps.setString(2, male.isSelected() ? "Male" : "Female");
        ps.setDate(3, Date.valueOf(tdob.getText()));
        ps.setString(4, married.isSelected() ? "Married" : "Unmarried");
        ps.setString(5, temail.getText());
        ps.setString(6, tnumber.getText());
        ps.setString(7, (String) jdepartment.getSelectedItem());
        ps.setDouble(8, Double.parseDouble(tsalary.getText()));
        ps.setString(9, bactive.isSelected() ? "Active" : "Inactive");
    }
});

```

```

        ps.setInt(10, ide);

        ps.executeUpdate();
        JOptionPane.showMessageDialog(f, "Employee Updated!");
        update.setVisible(false);
        save.setVisible(true);
        show.doClick();
        bactive.setVisible(false);
        binactive.setVisible(false);
        edit.setVisible(true);
        clear.doClick();
        refreshTable(tableModel);

    } catch (Exception ex) {
        JOptionPane.showMessageDialog(f, "Error: " + ex.getMessage());
    }
});

// Add button
add.addActionListener(e -> {
    profilePanel.setVisible(false);

    save.setVisible(true);
    update.setVisible(false);
    bactive.setVisible(false);
    binactive.setVisible(false);
    edit.setVisible(true);
    refreshTable(tableModel);

});

```

```

// Edit button
edit.addActionListener(e -> {
    edit.setVisible(false);
    save.setVisible(false);
    update.setVisible(true);
    bactive.setVisible(true);
    binactive.setVisible(true);

    try (Connection conn = getConnection()) {
        String query = "SELECT * FROM employee WHERE id = ?";
        PreparedStatement ps = conn.prepareStatement(query);
        ps.setInt(1, ide);
        ResultSet rs = ps.executeQuery();

        if (rs.next()) {
            tname.setText(rs.getString("name"));
            temail.setText(rs.getString("email"));
            tnumber.setText(rs.getString("mobile"));
            tdob.setText(rs.getDate("dob").toString());
            if ("Male".equals(rs.getString("gender"))) {
                male.setSelected(true);
            } else {
                female.setSelected(true);
            }
            if ("Married".equals(rs.getString("marital_status"))) {
                married.setSelected(true);
            } else {
                unmarried.setSelected(true);
            }
            if ("Active".equals(rs.getString("status"))) {

```



```

        bactive.setSelected(true);
    } else {
        binactive.setSelected(true);
    }

    jdepartment.setSelectedItem(rs.getString("department"));
    tsalary.setText(String.valueOf(rs.getDouble("salary")));
} else {
    JOptionPane.showMessageDialog(f, "Employee not found!");
}
} catch (Exception ex) {
    JOptionPane.showMessageDialog(f, "Error: " + ex.getMessage());
}

});

// hide show button
show.addActionListener(e -> {

    try (Connection conn = getConnection()) {
        String query = "SELECT * FROM employee WHERE id = ?";
        PreparedStatement ps = conn.prepareStatement(query);
        ps.setInt(1, ide);
        ResultSet rs = ps.executeQuery();

        if (rs.next()) {
            // Update UI components
            ide = rs.getInt("id");
            String status = rs.getString("status");
            if (status.equals("Active")) {
                statusLabel.setForeground(Color.GREEN);
            } else {

```

```

        statusLabel.setForeground(Color.RED);

    }

    statusLabel.setText(status);
    nameLabel.setText("Name: " + rs.getString("name"));
    pname.setText(rs.getString("name"));
    idLabel.setText("Employee ID: " + rs.getInt("id"));
    genderLabel.setText("Gender: " + rs.getString("gender"));
    dobLabel.setText("DOB: " + rs.getDate("dob").toString());
    maritalLabel.setText("Marital Status: " + rs.getString("marital_status"));
    emailLabel.setText("Email: " + rs.getString("email"));
    mobileLabel.setText("Mobile: " + rs.getString("mobile"));
    departmentLabel.setText("Department: " + rs.getString("department"));
    salaryLabel.setText("Salary: " + rs.getDouble("salary"));
    profilePanel.setVisible(true);

} else {

    JOptionPane.showMessageDialog(f, "Employee Not found");

}

conn.close();
} catch (SQLException e1) {
    e1.printStackTrace();
    JOptionPane.showMessageDialog(f, "Database error: " + e1.getMessage());
}
});

employeeTable.addMouseListener(new MouseAdapter() {
    public void mousePressed(MouseEvent e) {
        // Get the clicked row and column

```

```

int row = employeeTable.rowAtPoint(e.getPoint());
int column = employeeTable.columnAtPoint(e.getPoint());
if (column != 0) {

    Object idValue = employeeTable.getValueAt(row, 0);
    ide = ((Integer) idValue).intValue();

    show.doClick();

}
}
});
}

```

// Refresh Table

```

private static void refreshTable(DefaultTableModel model) {
    model.setRowCount(0); // Clear existing rows
    try (Connection conn = getConnection()) {
        String query = "SELECT * FROM employee";
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(query);

        // Add rows to the model
        while (rs.next()) {
            Object[] row = {
                rs.getInt("id"),
                rs.getString("name"),
                rs.getString("gender"),
                rs.getDate("dob"),
            }
        }
    }
}

```

```

        rs.getString("email"),
        rs.getString("mobile"),
        rs.getString("department"),
        rs.getDouble("salary"),
        rs.getString("status")
    };
    model.addRow(row);
}
} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, "Error refreshing table: " + ex.getMessage());
}
}

```

```

public static void applyStatusColumnRenderer(JTable table, int columnIndex) {
    DefaultTableCellRenderer statusRenderer = new DefaultTableCellRenderer() {
        @Override
        public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected,
            boolean hasFocus, int row, int column) {
            Component cell = super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row,
column);

            // Apply custom text color based on the status value
            if (value != null && value.toString().equalsIgnoreCase("Active")) {
                cell.setForeground(Color.GREEN);
            } else if (value != null && value.toString().equalsIgnoreCase("Inactive")) {
                cell.setForeground(Color.RED);
            } else {
                cell.setForeground(Color.BLACK);
            }

            return cell;
        }
    };
}

```

```
    }  
};  
  
// Set custom renderer to the specified column  
table.getColumnModel().getColumn(columnIndex).setCellRenderer(statusRenderer);  
}  
  
}
```

PROJECT VIEW

Employee Data Entry

Employee Data Entry

Name:

Enter DOB: (yyyy-MM-dd)

Gender: ☐ Male ☐ Female

Email:


Phone Number:


Marital Status: ☐ Married ☐ Unmarried

Select Department: IT Department

Salary:

Profile Details





Prabhat Maurya
Active

Name: Prabhat Maurya

Employee ID: 4

Gender: Male

DOB: 2004-10-27

Marital Status: Unmarried

Email: Prabhat@

Mobile: 8

Department: IT Department

Salary: 32245.0

ID	Name	Gender	DOB	Email	Phone	Department	Salary	Status
1	Prabhat Maurya	Male	2005-09-27	Prabhat@	8052850331	IT	32245.0	Active
2	Prabhat Maurya	Male	2004-01-05	Prabhatmaurya@323	8052850331	IT Department	125222.0	Inactive
3	Prabhat Maurya	Male	2005-09-27	Prabhat@	8052850331	Accountant Department	32.0	Inactive
	Prabhat Maurya	Male	2004-10-27	Prabhat@	8	IT Department	32245.0	Active
	Abhishek Tiwari	Male	2003-09-23	Abhishek@348922dh	7800527308	HR Department	20000.0	Active
	Krishna Yadav	Male	2004-08-13	Krishna1334809@g	851236566	Accountant Department	101000.0	Active
	Chitransh Maurya	Male	2002-10-05	Chitransh3454@gma	8956426982	HR Department	100000.0	Inactive
	Radha	Female	1998-05-10	Radha4922928@gm	528552089	Other	1000000.0	Inactive
	HElio	Male	2005-10-10	PrjksjsU@84909.com	8554952132	Other	50000.0	Active

CONCLUSION

Provide a final summary of the project and its main features.

Example: The **Task Manager** Java application provides a simple yet powerful way to manage tasks. With features like prioritization, task creation, and an intuitive user interface, the project demonstrates key Java programming concepts such as object-oriented design, database interaction, and testing

REFERENCES

Provide any references or resources used in the development of the project.

Example:

- ❖ Java Documentation: <https://docs.oracle.com/javase/8/docs/>
- ❖ JUnit 5 Documentation: <https://junit.org/junit5/docs/current/user-guide/>
- ❖ SQLite Documentation: <https://www.sqlite.org/docs.html>