PERSONAL INFORMATION SYSTEM

A MINI PROJECT REPORT

SUBMITTED BY

Aaditya Partha Sarathy 230701001

Aakash.V 230701002

In partial fulfillment for the award of the degree

BACHELOR OF ENGINEERING

IN COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE(AUTONOMOUS)
THANDALAM

CHENNAI-602105

2024- 2025

BONAFIDE CERTIFICATE

Certified that this project report“**PERSONAL INFORMATION SYSTEM**” is the bonafide work of “Aaditya Partha Sarathy (230701001), Aakash V(230701002) ”

who carried out the project work under my supervision.

Submitted for the Practical Examination held on 23.11.2024_____

ABSTRACT

The Personal Information System is a console-based application built using linked lists in C. It allows for efficient management of user records, including the creation, updating, and deletion of profiles. Each profile contains a unique user ID alongside personal details like name and contact information. Users can easily add or update their details, ensuring accurate and current records.

Changes to user profiles are reflected dynamically, while the system also provides the capability to display all stored information. The use of linked lists ensures efficient data handling, enabling flexible insertion, deletion, and retrieval of user records with minimal overhead.

The system's flexibility makes it well-suited for managing diverse types of personal data in both small and large datasets. By leveraging linked lists, the application minimizes memory usage and offers dynamic scalability, making it adaptable to varying data requirements. This approach ensures quick data access and modification, contributing to effective record management and providing users with a seamless experience when managing their personal information.

TABLE OF CONTENTS

1.INTRODUCTION

A Personal Information System (PIS) is designed to collect, manage, store, and utilize individual data for both personal and professional purposes. This system organizes a wide range of personal information, such as contact details, appointments, tasks, and personal records, to ensure easy access and streamlined management. By automating and integrating data, PIS allows users to optimize their day-to-day activities, track progress, and make informed decisions based on their own data. It supports data privacy and security while offering a personalized approach to handling everyday tasks and records. Whether used by individuals, professionals, or organizations, a Personal Information System is key to improving efficiency, organization, and productivity in managing personal data.

REQUIREMENTS

2.1 Software Requirement Specifications

Operating System Front End Back End Server Documentation : Windows 11

Frontend Software: Java

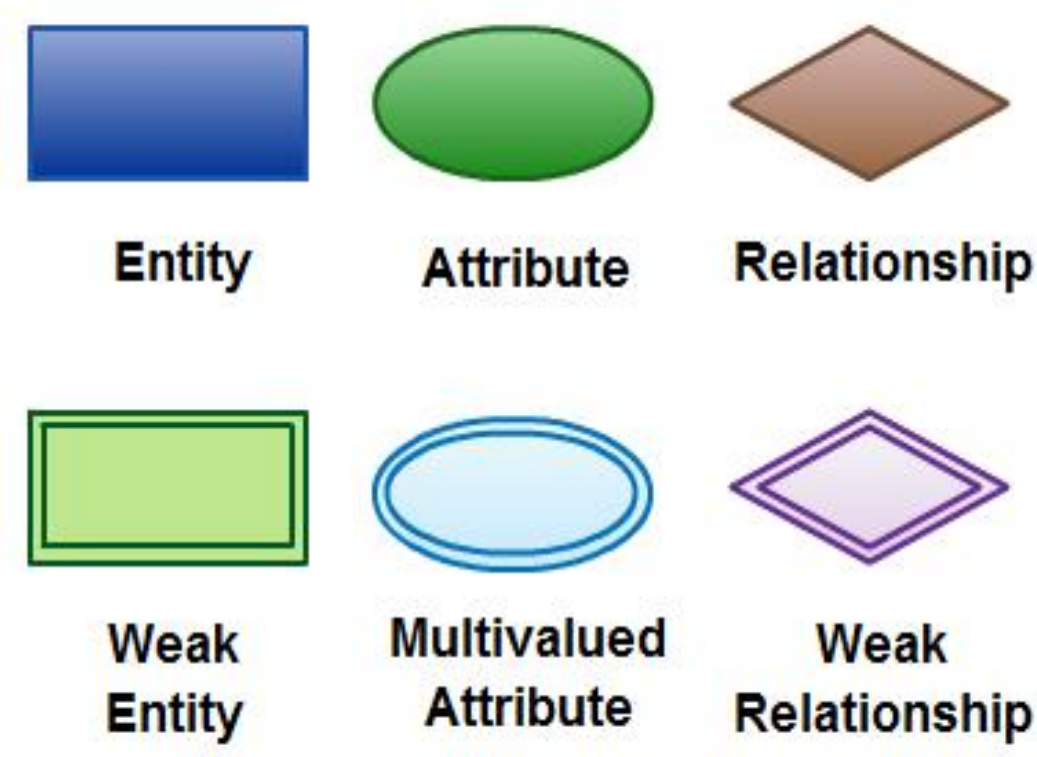
Backend Software: MySQL

2.2 Hardware Requirement Specifications

Computer Processor Core i3 Processor Speed 2.3 GHz Processor Hard Disk 400 GB or more RAM Min 2GB

ENTITY RELATIONSHIP DIAGRAM

An entity-relationship (ER) diagram is a specialized graphic that illustrates the interrelationships between entities in a database. ER diagrams often use symbols to represent three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes. If the application is primarily a database application, the entity-relationship approach can be used effectively for modeling some parts of the problem. The main focus in ER modeling is the Data Items in the system and the relationship between them. It aims to create conceptual scheme for the Data from the user's perspective. The model thus created is independent of any database model. The ER models are frequently represented as ER diagram. Here we present the ER diagram of the above mentioned project.



SCHEMA DIAGRAM

A database schema is the skeleton structure that represents the logical view of the entire database. A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.

A database schema can be divided broadly into two categories −

- Physical Database Schema − This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
- Logical Database Schema − This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

IMPLEMENTATION

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.sql.*;

public class PersonalInfoSystem extends JFrame {
    // Database connection parameters
    private static final String DB_URL = "jdbc:mysql://localhost:3306/aaditya";
    private static final String DB_USER = "root"; // Replace with your MySQL username
    private static final String DB_PASSWORD = "Farfromweak@120705"; // Replace with your MySQL password

    // GUI Components
    private JTextField idField, nameField, emailField, phoneField, addressField;
    private JTable table;
    private DefaultTableModel tableModel;

    public PersonalInfoSystem() {
        // Frame setup
        setTitle("Personal Information System");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
setSize(800, 600);

setLayout(new BorderLayout());


// Top Panel: Form for input

JPanel formPanel = new JPanel(new GridLayout(5, 2, 10, 10));

formPanel.setBorder(BorderFactory.createTitledBorder("Personal Info"));


formPanel.add(new JLabel("ID (Auto-generated):"));

idField = new JTextField();

idField.setEditable(false);

formPanel.add(idField);


formPanel.add(new JLabel("Name:"));

nameField = new JTextField();

formPanel.add(nameField);


formPanel.add(new JLabel("Email:"));

emailField = new JTextField();

formPanel.add(emailField);


formPanel.add(new JLabel("Phone:"));
```

```
        phoneField = new JTextField();
        formPanel.add(phoneField);

        formPanel.add(new JLabel("Address:"));
        addressField = new JTextField();
        formPanel.add(addressField);

        add(formPanel, BorderLayout.NORTH);

        // Center Panel: Table to display records
        tableModel = new DefaultTableModel(new String[]{"ID", "Name", "Email", "Phone",
"Address"}, 0);
        table = new JTable(tableModel);
        JScrollPane tableScrollPane = new JScrollPane(table);
        table.setFillsViewportHeight(true);
        add(tableScrollPane, BorderLayout.CENTER);

        // Bottom Panel: Buttons for operations
        JPanel buttonPanel = new JPanel(new GridLayout(1, 4, 10, 10));
        JButton addButton = new JButton("Add");
        JButton updateButton = new JButton("Update");
        JButton deleteButton = new JButton("Delete");
```

```
    JButton refreshButton = new JButton("Refresh");

    buttonPanel.add(addButton);
    buttonPanel.add(updateButton);
    buttonPanel.add(deleteButton);
    buttonPanel.add(refreshButton);

    add(buttonPanel, BorderLayout.SOUTH);

    // Add action listeners for buttons
    addButton.addActionListener(this::addRecord);
    updateButton.addActionListener(this::updateRecord);
    deleteButton.addActionListener(this::deleteRecord);
    refreshButton.addActionListener(e -> fetchRecords());

    // Load initial data
    initializeDatabase();
    fetchRecords();
}

// Initialize database (Create table if not exists)
```

```
    private void initializeDatabase() {
        try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
             Statement stmt = conn.createStatement()) {
            String createTableSQL = """
                    CREATE TABLE IF NOT EXISTS PersonalInfo (
                        id INT AUTO_INCREMENT PRIMARY KEY,
                        name VARCHAR(100) NOT NULL,
                        email VARCHAR(100) NOT NULL UNIQUE,
                        phone VARCHAR(15),
                        address TEXT
                    );
                    """;
            stmt.execute(createTableSQL);
        } catch (SQLException e) {
            showErrorDialog("Error initializing database: " + e.getMessage());
        }
    }

    // Fetch all records from the database and populate the table
    private void fetchRecords() {
```

```
        try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery("SELECT * FROM PersonalInfo")) {
            tableModel.setRowCount(0); // Clear existing rows
            while (rs.next()) {
                tableModel.addRow(new Object[]{
                        rs.getInt("id"),
                        rs.getString("name"),
                        rs.getString("email"),
                        rs.getString("phone"),
                        rs.getString("address")
                });
            }
        } catch (SQLException e) {
            showErrorDialog("Error fetching records: " + e.getMessage());
        }
    }

    // Add a new record to the database
    private void addRecord(ActionEvent e) {
        String name = nameField.getText();
```

```
        String email = emailField.getText();
        String phone = phoneField.getText();
        String address = addressField.getText();

        if (name.isEmpty() || email.isEmpty()) {
            showErrorDialog("Name and Email are required!");
            return;
        }

        try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD)) {
            String insertSQL = "INSERT INTO PersonalInfo (name, email, phone, address)
VALUES (?, ?, ?, ?)";
            PreparedStatement stmt = conn.prepareStatement(insertSQL);
            stmt.setString(1, name);
            stmt.setString(2, email);
            stmt.setString(3, phone);
            stmt.setString(4, address);
            stmt.executeUpdate();
            showInfoDialog("Record added successfully!");
            fetchRecords();
        } catch (SQLException ex) {
```

```
            showErrorDialog("Error adding record: " + ex.getMessage());
        }
    }

    // Update a selected record in the database
    private void updateRecord(ActionEvent e) {
        int selectedRow = table.getSelectedRow();
        if (selectedRow == -1) {
            showErrorDialog("Please select a record to update!");
            return;
        }

        int id = (int) tableModel.getValueAt(selectedRow, 0);
        String name = nameField.getText();
        String email = emailField.getText();
        String phone = phoneField.getText();
        String address = addressField.getText();

        if (name.isEmpty() || email.isEmpty()) {
            showErrorDialog("Name and Email are required!");
            return;
```

```
        }

        try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD)) {
            String updateSQL = "UPDATE PersonalInfo SET name = ?, email = ?, phone = ?,
address = ? WHERE id = ?";
            PreparedStatement stmt = conn.prepareStatement(updateSQL);
            stmt.setString(1, name);
            stmt.setString(2, email);
            stmt.setString(3, phone);
            stmt.setString(4, address);
            stmt.setInt(5, id);
            stmt.executeUpdate();
            showInfoDialog("Record updated successfully!");
            fetchRecords();
        } catch (SQLException ex) {
            showErrorDialog("Error updating record: " + ex.getMessage());
        }
    }

    // Delete a selected record from the database
    private void deleteRecord(ActionEvent e) {
```

```
        int selectedRow = table.getSelectedRow();
        if (selectedRow == -1) {
            showErrorDialog("Please select a record to delete!");
            return;
        }

        int id = (int) tableModel.getValueAt(selectedRow, 0);

        try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD)) {
            String deleteSQL = "DELETE FROM PersonalInfo WHERE id = ?";
            PreparedStatement stmt = conn.prepareStatement(deleteSQL);
            stmt.setInt(1, id);
            stmt.executeUpdate();
            showInfoDialog("Record deleted successfully!");
            fetchRecords();
        } catch (SQLException ex) {
            showErrorDialog("Error deleting record: " + ex.getMessage());
        }
    }

    // Utility method to show error messages
```

```
    private void showErrorDialog(String message) {
        JOptionPane.showMessageDialog(this, message, "Error",
JOptionPane.ERROR_MESSAGE);
    }

    // Utility method to show information messages
    private void showInfoDialog(String message) {
        JOptionPane.showMessageDialog(this, message, "Info",
JOptionPane.INFORMATION_MESSAGE);
    }

    // Main method
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            PersonalInfoSystem app = new PersonalInfoSystem();
            app.setVisible(true);
        });
    }
}
```

SNAPSHOT

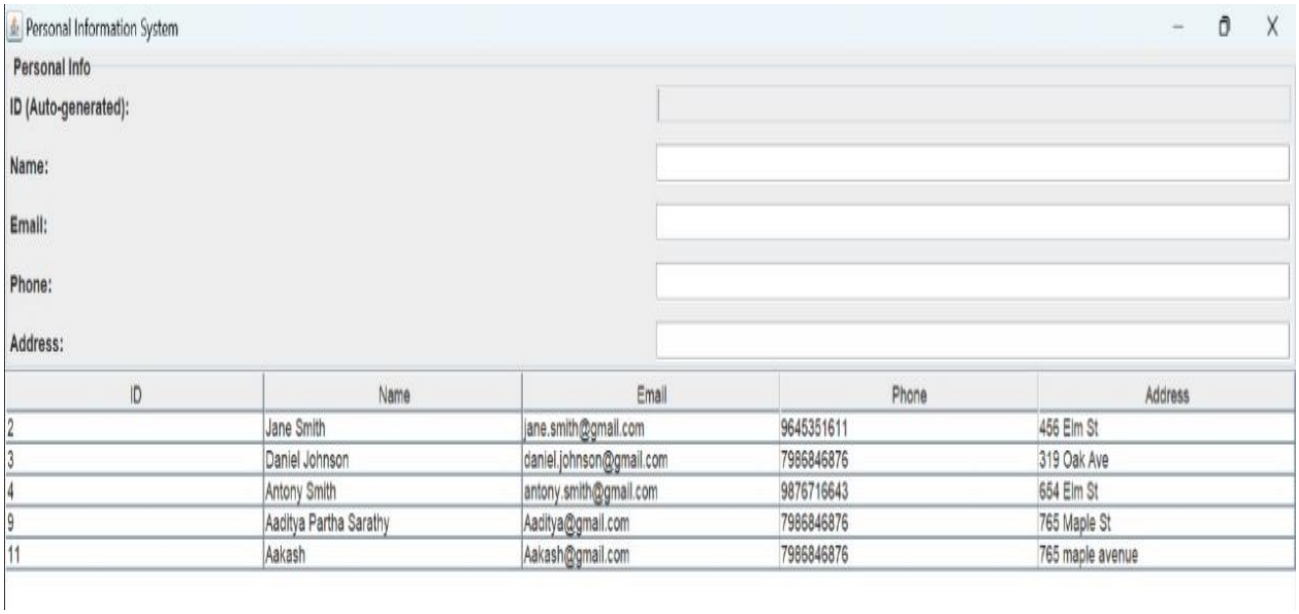


Figure 1:Home Page

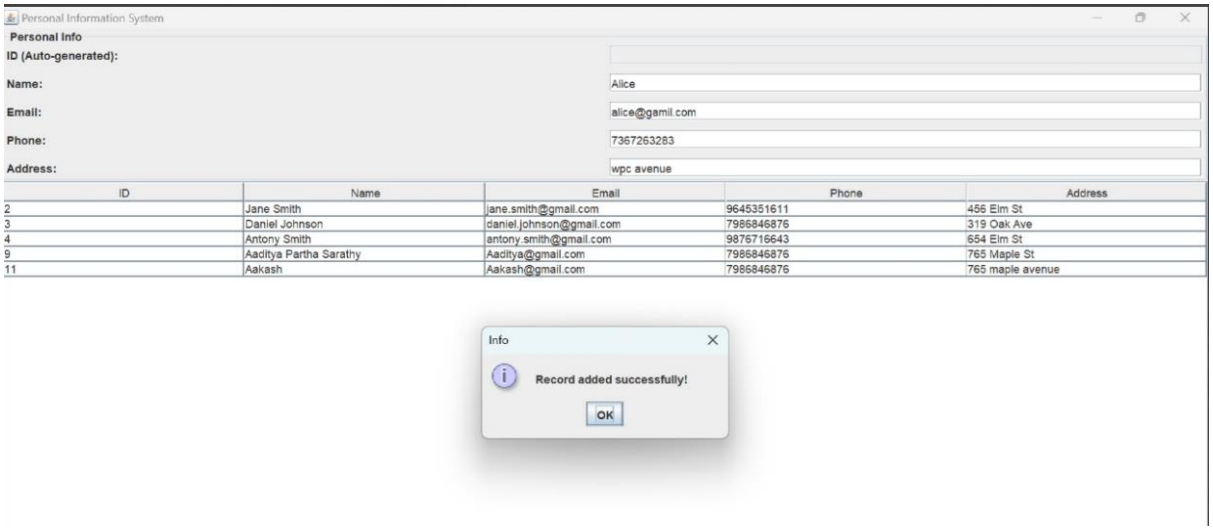


Figure 2:Addion Operation

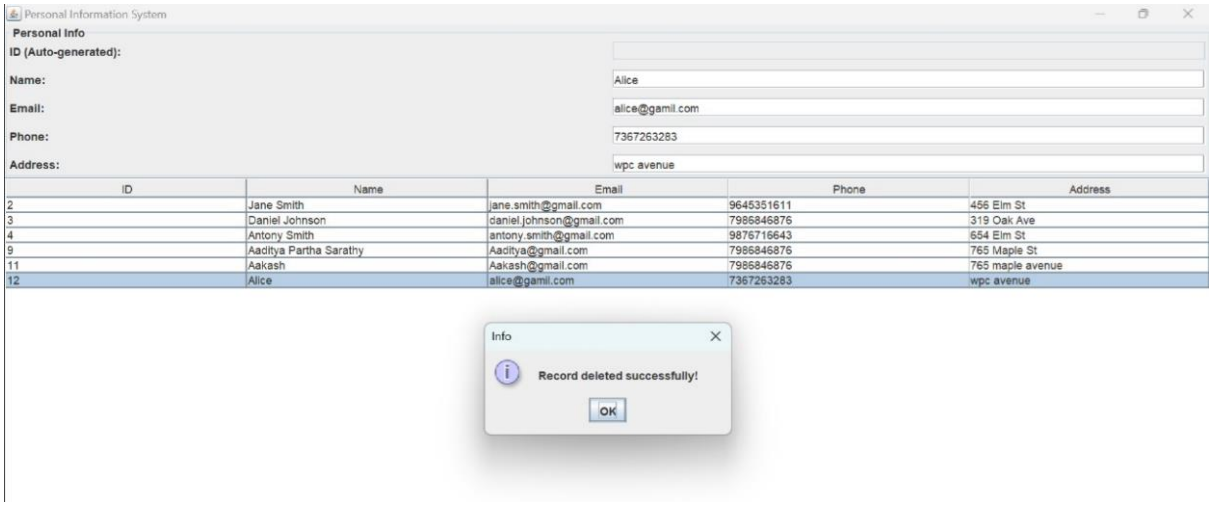


Figure 3:Deletion operation

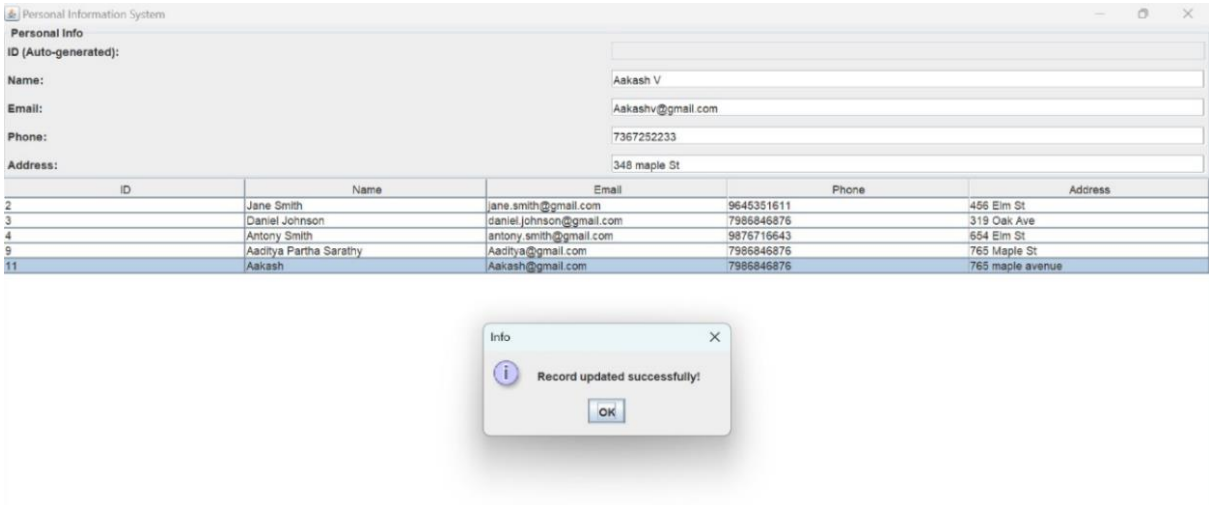


Figure 4:Update operation