Personal Information System

A MINI-PROJECT BY:

S T Kaif Rehman 230701134

B R Gowtham 230701524

I*n partial fulfillment of the award of the degree*

OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

AN AUTONOMOUS INSTITUTE CHENNAI NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project report "**Personal Information System"** is a Bonafide work of "**S T Kaif Rehman (230701134)& B R Gowtham 230701524" .**

Submitted for the Practical Examination held on _____

Signature

Ms Dharshini B S

Assistant Professor,

Computer Science and Engineering,

Rajalakshmi Engineering College (Autonomous),

Thandalam, Chennai-602 105

Internal Examiner External Examiner

ACKNOWLEDGEMENT

I would like to extend my sincere gratitude to everyone who has contributed to the success in the incoming successful completion of this mini project.

First , I am deeply thankful to my Professor Ms Dharshini B S my project advisor, for their invaluable guidance, and insightful feedback, and continuous support throughout the duration of this mini project. Their expertise and encouragement have been instrumental in shaping my research and bringing this mini project to completion.

I would also like to express my appreciation to the faculty and staff of the Computer Science and Engineering Department at Rajalakshmi Engineering College for providing the necessary resources and learning environment.

We express our sincere thanks to Dr.P. Kumar, M.E., Ph.D., Professor and Head of the Department Computer Science and Engineering for his guidance and encouragement throughout the project work.My heartfelt thanks go to my peers and friends for their collaboration,constructive criticism, and moral support

Thank you all for your contributions, to the success of this project.

Personal Information System.

Abstract : This is done to achieve the safety and ensuring the storage of a Person's data in the database. Data can be inserted.deleted and displayed to showcase the extravaganda of this database

Front End Tools: Overview of the technology stack (Java, Swing, JDBC).

Back End Tools : Using Oracle Database

Table of Contents

Table of contents involve

CONTENTS :

Introduction

The Personal Information System (PIS) is designed to store, manage, and retrieve personal information for individuals within a database structure, incorporating essential details such as contact information, job history, educational background, and emergency contacts. This system is crucial for organizations and institutions that need to keep detailed and organized records of their employees, clients, or other stakeholders.

PIS provides a secure and structured environment to handle personal data efficiently. By utilizing Java for front-end interaction and Oracle SQL for database management, the system achieves both user-friendliness and robust back-end support. Through a graphical user interface (GUI) built using Java Swing, users can seamlessly insert, update, view, and delete personal records. Additionally, by integrating JDBC (Java Database Connectivity), this system ensures reliable communication between the Java application and the Oracle database, enabling smooth data transactions and management.

System Overview

The Personal Information System (PIS) is a comprehensive application designed to handle and manage individual records, combining data integrity, accessibility, and user interaction. The system uses a **three-tier architecture** that includes:

> **User Interface (UI) Layer**: Built using Java Swing, the UI layer enables user interaction through a graphical interface, allowing users to perform operations such as adding, updating, deleting, and viewing records. The UI presents a straightforward and organized set of tabs and fields where users can input and retrieve data.

Application Logic Layer: The core logic of the system, implemented in Java, manages the operations of inserting, retrieving, and deleting data in the database. It serves as the middle layer between the UI and the database, ensuring that all requests are processed correctly and that data is consistently stored and retrieved. Java's event-driven approach handles user actions on the interface, and exceptions are managed to ensure robustness.

Database Layer: The Oracle SQL database forms the system's backend, where data is securely stored in well-defined tables. The database schema consists of several interrelated tables, including **Person, Address, Contact, Job, Education,** and **EmergencyContact**. Relationships among these tables are maintained through foreign keys, ensuring referential integrity across records. SQL triggers and stored procedures are used to automate specific tasks and manage data entry efficiently.

1. **Data Management**: PIS allows users to add, edit, and delete personal information, encompassing a wide range of details such as names, contact information, job roles, educational **qualifications, and emergency contact data.**
2. **Data Integrity**: By using primary and foreign keys, the database ensures that all relationships between tables remain consistent. For instance, a contact cannot exist without an associated person record.
3. **Automation**: SQL triggers automate tasks such as generating unique IDs for records, while stored

procedures enable batch operations, allowing multiple records to be added with a single command.

4. **Front End Tools**:Overview of the technology stack (Java, Swing, JDBC).
5. **Back End Tools :** Using Oracle Database
6. **Database Design**
7. ER Diagram (Entity-Relationship Diagram).Table structures and schema with SQL CREATE statements (including Person, Address, Contact, Job, Education, EmergencyContact tables).
8. Explanation of each table's role and relationships.

Implementation of SQL Code

Detailed SQL schema with code for tables, triggers, and procedures.

Example SQL code, with explanations for each key SQL construct.

Explanation of the Swing GUI, highlighting the Insert, Display, and Delete tabs.

Java code integration with JDBC for database operations.

Explanation of PIS class structure and important methods.

ER Diagram:

The following ER diagram represents the relationships and attributes in the Personal Information System database, including tables like Person, Address, Contact, Job, Education, and Emergency Contact. Each table is interconnected to provide a cohesive data management structure for personal information.

ER diagrams helps in maintain contiguous and well known appearance of the schema for our entities and attributes

TABLES INVOLVED:

==========

| Person Table |

==========

| PersonID | FirstName | LastName | DOB | Gender | MaritalStatus |

| 1 | John | Doe | 1990-01-01 | M | Single |

| 2 | Jane | Smith | 1992-02-15 | F | Married |

==========

==========

| Address Table |

==========

| AddressID | PersonID | Street | City | State | ZipCode | Country |

| 1 | 1 | 123 Main St | Anytown | Anystate | 12345 | USA |

| 2 | 2 | 456 Oak Rd | Othertown| Otherstate| 67890 | USA |

==========

| Contact Table |

==========

| ContactID | PersonID | PhoneNumber | Email | ContactType | CreatedDate | UpdatedDate |

| 1 | 1 | 123-456-7890| john.doe@email.com | Home | 2024-01-01 | 2024-01-01 |

| 2 | 2 | 987-654-3210| jane.smith@email.com| Work | 2024-02-01 | 2024-02-02 |

==========

==========

| Job Table |

==========

| JobID | PersonID | JobTitle | Department | StartDate | EndDate | Salary |

| 1 | 1 | Developer | IT | 2020-01-01 | NULL | 80000.00 |

| 2 | 2 | Manager | HR | 2019-03-01 | 2023-12-31 | 90000.00 |

==========

==========

| Education Table |

==========

| EducationID | PersonID | Institution | Degree | GraduationYear | Major | GPA |

| 1 | 1 | Tech Univ | B.Sc. | 2012 | Computer Science | 3.8 |

| 2 | 2 | State College| M.A. | 2015 | Psychology | 3.9 |

| Emergency Contact Table |

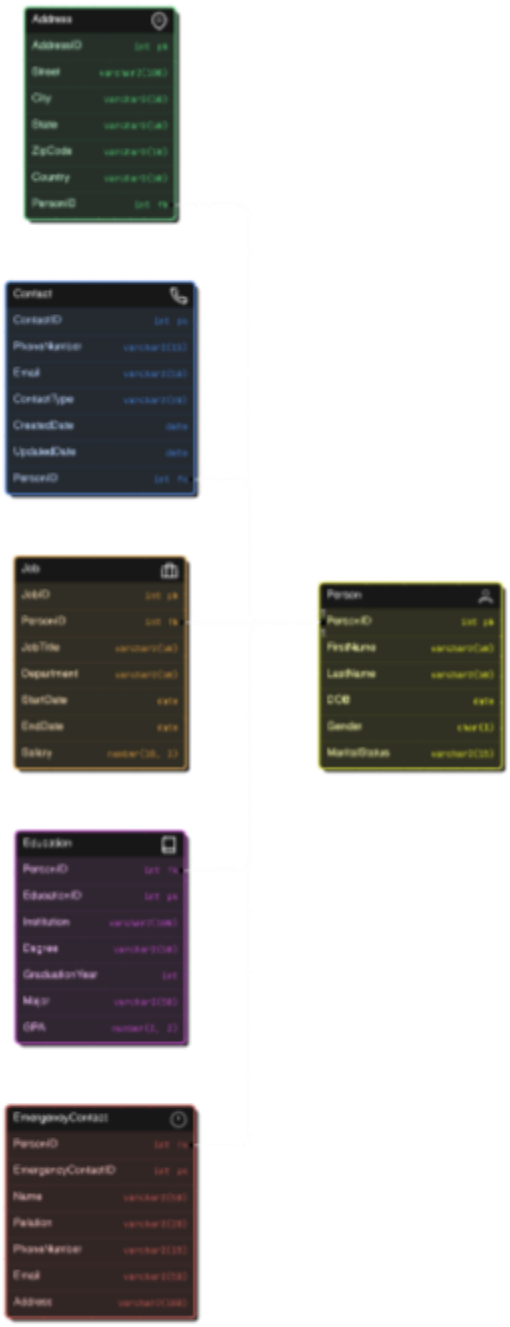
==========

| EmergencyContactID | PersonID | Name | Relation | PhoneNumber | Email | Address |

| 1 | 1 | Alice Doe | Sister | 123-111-2222 | alice.doe@email.com | 123 Main St, Anytown |

| 2 | 2 | Bob Smith | Father | 987-222-3333 | bob.smith@email.com | 456 Oak Rd, Othertown|

ER DIAGRAM FOR PERSONAL INFORMATION SYSTEM



SQL DATABASE IMPLEMENTATION:

```
CREATE TABLE Person (
    PersonID INT PRIMARY KEY,
    FirstName VARCHAR2(50),
    LastName VARCHAR2(50),
    DOB DATE,
    Gender CHAR(1),
    MaritalStatus VARCHAR2(15)
);

CREATE TABLE Address (
    AddressID INT PRIMARY KEY,
    PersonID INT REFERENCES Person(PersonID),
    Street VARCHAR2(100),
    City VARCHAR2(50),
    State VARCHAR2(50),
    ZipCode VARCHAR2(10),
    Country VARCHAR2(50)
);

CREATE TABLE Contact (
    ContactID INT PRIMARY KEY,
    PersonID INT REFERENCES Person(PersonID),
    PhoneNumber VARCHAR2(15),
    Email VARCHAR2(50),
    ContactType VARCHAR2(20),
    CreatedDate DATE,
    UpdatedDate DATE
);

CREATE TABLE Job (
    JobID INT PRIMARY KEY,
    PersonID INT REFERENCES Person(PersonID),
    JobTitle VARCHAR2(50),
    Department VARCHAR2(50),
    StartDate DATE,
    EndDate DATE,
    Salary NUMBER(10, 2)
);

CREATE TABLE Education (
    EducationID INT PRIMARY KEY,
    PersonID INT REFERENCES Person(PersonID),
    Institution VARCHAR2(100),
    Degree VARCHAR2(50),
    GraduationYear INT,
    Major VARCHAR2(50),GPA NUMBER(3, 2));
```

```
CREATE TABLE EmergencyContact (
    EmergencyContactID INT PRIMARY KEY,
    PersonID INT REFERENCES Person(PersonID),
    Name VARCHAR2(50),
    Relation VARCHAR2(20),
    PhoneNumber VARCHAR2(15),
    Email VARCHAR2(50),
    Address VARCHAR2(100)
);
```

SQL CODE FOR TRIGGER AND PROCEDURE :

```
CREATE OR REPLACE TRIGGER trg_auto_person_id
BEFORE INSERT ON Person
FOR EACH ROW
BEGIN
    IF :NEW.PersonID IS NULL THEN
        :NEW.PersonID := person_seq.NEXTVAL;
    END IF;
END;
/

CREATE OR REPLACE PROCEDURE AddPersonDetails (
    p_FirstName VARCHAR2, p_LastName VARCHAR2, p_DOB DATE, p_Gender
CHAR, p_MaritalStatus VARCHAR2,
    p_Street VARCHAR2, p_City VARCHAR2, p_State VARCHAR2, p_ZipCode
VARCHAR2, p_Country VARCHAR2,
    p_PhoneNumber VARCHAR2, p_Email VARCHAR2, p_ContactType VARCHAR2,
    p_JobTitle VARCHAR2, p_Department VARCHAR2, p_StartDate DATE,
p_EndDate DATE, p_Salary NUMBER,
    p_Institution VARCHAR2, p_Degree VARCHAR2, p_GraduationYear INT,
p_Major VARCHAR2, p_GPA NUMBER,
    p_EmergencyName VARCHAR2, p_EmergencyRelation VARCHAR2,
p_EmergencyPhone VARCHAR2, p_EmergencyEmail VARCHAR2,
p_EmergencyAddress VARCHAR2
) AS
    v_PersonID INT; -- Declare variable to hold the PersonID
BEGIN
    -- Insert into Person table
    INSERT INTO Person (FirstName, LastName, DOB, Gender,
MaritalStatus)
    VALUES (p_FirstName, p_LastName, p_DOB, p_Gender, p_MaritalStatus)
    RETURNING PersonID INTO v_PersonID;
```

```
    -- Insert into Address table
    INSERT INTO Address (PersonID, Street, City, State, ZipCode,
Country)
    VALUES (v_PersonID, p_Street, p_City, p_State, p_ZipCode,
p_Country);

    -- Insert into Contact table
    INSERT INTO Contact (PersonID, PhoneNumber, Email, ContactType,
CreatedDate)
    VALUES (v_PersonID, p_PhoneNumber, p_Email, p_ContactType,
SYSDATE);

    -- Insert into Job table
    INSERT INTO Job (PersonID, JobTitle, Department, StartDate,
EndDate, Salary)
    VALUES (v_PersonID, p_JobTitle, p_Department, p_StartDate,
p_EndDate, p_Salary);

    -- Insert into Education table
    INSERT INTO Education (PersonID, Institution, Degree,
GraduationYear, Major, GPA)
    VALUES (v_PersonID, p_Institution, p_Degree, p_GraduationYear,
p_Major, p_GPA);

    -- Insert into EmergencyContact table
    INSERT INTO EmergencyContact (PersonID, Name, Relation,
PhoneNumber, Email, Address)
    VALUES (v_PersonID, p_EmergencyName, p_EmergencyRelation,
p_EmergencyPhone, p_EmergencyEmail, p_EmergencyAddress);

    COMMIT;
END;
/

================================================================
```

JAVA CODE IMPLEMENTATION USING JDBC SWING

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class PIS {
    private Connection connection;

    public PIS() {
        // Initialize Database Connection
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            connection = DriverManager.getConnection(
                "jdbc:oracle:thin:@127.0.0.1:1521:xe",
                "230701134KAIF",
                "Kaif@college1"
            );
            System.out.println("Database connected successfully!");
        } catch (Exception e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, "Database connection
failed!");
            System.exit(0);
        }

        // Setup GUI
        JFrame frame = new JFrame("Personal Information System");
        frame.setSize(600, 400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // GUI Setup Code Continues...
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(PIS::new);
    }
}
```

NORMALIZATION:

1st Normal Form (1NF) - Atomic Values

In **1NF,** each column must contain **atomic values** (i.e., no repeating groups or lists in a single column). All rows should have a unique identifier (primary key).

```
==============================
|       Person Table          |
==============================
| PersonID | FirstName | LastName | DOB       | Gender | MaritalStatus
|
-------------------------------------------------------------------------
| 1        | John      | Doe      | 1990-01-01 | M      | Single
|
| 2        | Jane      | Smith    | 1992-02-15 | F      | Married
|
==============================
```

- **Explanation:** The Person table is in **1NF** because each column has atomic values (no lists), and there is a unique PersonID for each record.

2nd Normal Form (2NF) - No Partial Dependencies

In **2NF,** the table must be in **1NF,** and all **non-key attributes** must be fully functionally dependent on the entire **primary key.**

```
==============================

|       Address Table        |

==============================

| AddressID | PersonID | Street       | City     | State    | ZipCode |
Country    |

-------------------------------------------------------------------------
-----------------

| 1         | 1        | 123 Main St  | Anytown  | Anystate | 12345   |
USA        |

| 2         | 2        | 456 Oak Rd   | Othertown| Otherstate| 67890
| USA        |

==============================
```

- **Explanation:** The Address table is in **2NF** because AddressID is the primary key, and all other attributes like Street, City, State, etc., depend fully on this key, not just part of it. PersonID is a foreign key here and refers to Person.

3rd Normal Form (3NF) - No Transitive Dependencies

In **3NF**, the table must be in **2NF**, and there should be **no transitive dependencies** (i.e., non-key attributes should not depend on other non-key attributes).

```
==============================

|       Contact Table        |

==============================

| ContactID | PersonID | PhoneNumber | Email             | ContactType
| CreatedDate | UpdatedDate |

-------------------------------------------------------------------------
------------------------------

| 1         | 1        | 123-456-7890| john.doe@email.com | Home
| 2024-01-01  | 2024-01-01  |

| 2         | 2        | 987-654-3210| jane.smith@email.com| Work
| 2024-02-01  | 2024-02-02  |

==============================
```

- **Explanation:** The Contact table is in **3NF** because it is in **2NF**, and there are no transitive dependencies. For example, Email depends directly on ContactID, not indirectly via PersonID.

Boyce-Codd Normal Form (BCNF) - Superkey Dependency

In **BCNF**, a table must be in **3NF**, and for every functional dependency, the left-hand side must be a **superkey** (a set of attributes that uniquely identifies a record).

markdown

Copy code

```
==============================

|        Job Table           |

==============================

| JobID | PersonID | JobTitle   | Department | StartDate  | EndDate
| Salary   |

-------------------------------------------------------------------------
-----------------

| 1     | 1        | Developer  | IT         | 2020-01-01 | NULL
| 80000.00 |

| 2     | 2        | Manager    | HR         | 2019-03-01 | 2023-12-31
| 90000.00 |

==============================
```

- **Explanation:** The Job table is in **BCNF** because every functional dependency is based on a **superkey** (JobID is the superkey). There are no dependencies where a non-superkey determines another attribute.

4th Normal Form (4NF) - No Multi-Valued Dependencies

In **4NF**, the table must be in **BCNF**, and there should be **no multi-valued dependencies** (i.e., a record should not have two or more independent multi-valued attributes).

```
==============================

|      Education Table       |

==============================

| EducationID | PersonID | Institution  | Degree   | GraduationYear |
Major    | GPA  |

-------------------------------------------------------------------------
---------------------------

| 1           | 1        | Tech Univ    | B.Sc.    | 2012          |
Computer Science | 3.8 |

| 2           | 2        | State College| M.A.     | 2015          |
Psychology | 3.9 |

==============================
```

- **Explanation:** The Education table is in **4NF** because there are no multi-valued dependencies. For example, Degree and Major are not multiple values for the same record — they are distinct, and each non-key column depends on the EducationID.

5th Normal Form (5NF) - No Join Dependencies

In **5NF**, the table must be in **4NF**, and **no join dependencies** exist. The table should not be decomposed into smaller tables without losing information.

markdown

Copy code

```
==============================

|  Emergency Contact Table   |

==============================

| EmergencyContactID | PersonID | Name       | Relation   | PhoneNumber
| Email                | Address        |

-------------------------------------------------------------------------
-----------------------------------

| 1                  | 1        | Alice Doe  | Sister     | 123-111-2222
| alice.doe@email.com   | 123 Main St, Anytown |

| 2                  | 2        | Bob Smith  | Father     | 987-222-3333
| bob.smith@email.com   | 456 Oak Rd, Othertown|
```

- **Explanation:** The Emergency Contact table is in **5NF** because it cannot be further decomposed without losing meaningful relationships between EmergencyContactID, PersonID, Name, PhoneNumber, Email, and Address.

Summary of Normal Forms:

- **1NF:** Ensures atomic values in columns.
- **2NF:** Ensures full dependency on the primary key.
- **3NF:** Eliminates transitive dependencies.
- **BCNF:** Ensures that all dependencies are on superkeys.
- **4NF:** Removes multi-valued dependencies.
- **5NF:** Eliminates join dependencies, ensuring no further decomposition without loss of data.

CONCLUSION :

The Personal Information System developed using Java (JDBC, Swing) and SQL provides a robust and user-friendly interface for managing individual records with critical personal details...
Past records and Future Records along with Present Records are concisely and secured safely in the database.

The **Personal Information System** (PIS) is a robust and efficient database application designed to streamline the management of personal data. By integrating a Java front-end with JDBC and Oracle Express 23C as the back-end database, the system demonstrates a practical implementation of modern database connectivity and software development principles.

This project successfully achieves the following:

1. **Centralized Data Management**:
   It consolidates personal information, including contact details, addresses, job history, educational background, and emergency contacts, into an organized

and interconnected database. This ensures data consistency and reduces redundancy.

2. **User-Friendly Interface**:
   The Java Swing GUI offers an intuitive and interactive interface, allowing users to easily manage personal records without requiring technical expertise. This makes the application accessible to a wider audience.

3. **Scalability**:
   The system is designed with scalability in mind. New tables or attributes can be added seamlessly, and the current schema supports one-to-many relationships, which makes the system adaptable to future extensions.

4. **Data Integrity and Security**:
   The use of triggers, stored procedures, and relational constraints ensures the integrity of the data. The system also enforces referential integrity through foreign key constraints, reducing the risk of orphaned records.

5. **Automated Data Handling**:
   The use of a trigger (trg_update_contact) automates updates for related records, minimizing manual errors. Additionally, the stored procedure (AddPersonAddressContact) simplifies the process of inserting new entries across multiple tables.

6. **Real-World Applicability**:
   PIS can be implemented in various real-world scenarios, such as employee management systems, student information systems, and customer relationship management platforms, where structured personal data is critical.

7. **Learning Outcomes**:
   This project enhanced skills in:

   - Designing ER diagrams and normalizing databases.
   - Writing SQL queries, stored procedures, and triggers.
   - Implementing Java-JDBC for database connectivity.
   - Developing an intuitive graphical user interface with Java Swing.

APPLICATIONS:

-Human Resource Management

- Educational Institutions

- Healthcare Systems

- Government Agencies

- Customer Relationship Management (CRM)

- Personal Portfolio Management

1. **Adding Authentication**: Implementing a login system for secure access to personal records.
2. **Enhancing GUI**: Upgrading the Swing interface to JavaFX for a more modern look.
3. **Data Analytics**: Including reporting features to analyze personal data trends.
4. **Integration with Cloud**: Hosting the database on a cloud platform for better scalability and remote access.
5. **Multi-Language Support**: Enabling support for multiple languages to make the application more inclusive.

==========