RAJALAKSHMI ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

PERSONAL INFORMATION MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

Jonathan Azel D	**231001076**
Lakshmanan VS	**231001096**

In partial fulfillment for the award of the degree of

BACHELOR OF

ENGINEERING IN

Information Technology

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024 - 2025

BONAFIDE CERTIFICATE

Certified that this project report "**PERSONAL INFORMATION MANAGEMENT SYSTEM**" is the bonafide work of **"Jonathan Azel D (231001076), Lakshmanan VS (231001096) "** who carried out the project work under my supervision.

SUPERVISOR	**Head/IT**
Mrs.T.Sangeetha	**Dr.P.Valarmathie**

Date: 22.11.2024

Acknowledgements

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

First and foremost, I am deeply thankful to my Professor Mrs.T.Sangeetha , my project advisor, for her invaluable guidance, insightful feedback, and continuous support throughout the duration of this mini project. Her expertise and encouragement have been instrumental in shaping my research and bringing this project to fruition.

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

My heartfelt thanks go to my peers and friends for their collaboration, constructive criticism, and moral support. Their insights and camaraderie have been crucial in refining this project.

Thank you all for your contributions, both direct and indirect, to the success of this project.

ABSTRACT

The Personal Information Management System is a Java-based standalone application designed to efficiently store, manage, and retrieve personal data records. Built using JavaFX for a modern and responsive user interface and MySQL for robust database management, the system ensures an intuitive and seamless experience for handling personal information.

The application provides a secure login module, a visually appealing dashboard, dynamic forms for adding and editing details, and an interactive table view for displaying records. Features like real-time search, sorting, and update functionalities enhance usability. The system incorporates user confirmations for critical actions such as deletions or updates, minimizing errors and ensuring data integrity.

The project utilizes JDBC (Java Database Connectivity) for integrating Java with MySQL, facilitating efficient and secure database interactions. The use of JavaFX allows for the development of a polished and responsive UI, offering an improved user experience compared to traditional desktop interfaces.

Designed for scalability, simplicity, and security, this system serves as a reliable tool for managing personal data, making it ideal for small organizations, educational institutions, or individual users who require effective information management in a standalone environment.

TABLE OF CONTENTS

1. INTRODUCTION

1.1. INTRODUCTION

The Personal Information Management System is a desktop-based application designed to manage, organize, and retrieve personal information efficiently. Developed using JavaFX for the user interface and MySQL for the database, the system ensures robust functionality and a user-friendly design. The system's primary purpose is to provide users with a secure and reliable way to store and access personal records, making it ideal for individuals or small organizations.

1.2. OBJECTIVES

- To create a user-friendly application for managing personal information.
- To implement efficient database operations like storing, updating, and deleting records using MySQL.
- To ensure data security through a secure login mechanism.
- To design a modular and scalable system that can be extended for additional features.
- To utilize JavaFX for a modern and responsive user interface.

1.3. MODULES

1. **Login Module:** Secure authentication system.
2. **Dashboard:** Central hub for navigation to various functionalities.
3. **Personal Information Form:** Interface to add or update personal records.
4. **Data Display Module:** Displays stored records in a tabular format with options for searching and filtering.
5. **Database Management:** Handles backend operations such as CRUD (Create, Read, Update, Delete).
6. **Confirmation and Notifications:** Provides user feedback for successful or failed operations.

2. SURVEY OF TECHNOLOGIES

2.1. SOFTWARE DESCRIPTION

The Personal Information Management System is a Java-based desktop application designed to manage and organize user information efficiently. It uses JavaFX for building a dynamic and interactive user interface and MySQL for backend data storage. The combination of these technologies ensures a user-friendly experience and robust data management.

2.2. LANGUAGES

2.2.1. JavaFX

- JavaFX is a powerful, modern toolkit for building rich client applications. It provides a range of UI controls and layouts, along with CSS-based styling and built-in animation features.
- Used for building a dynamic and visually appealing user interface.
- Supports event-driven programming for enhanced interactivity.

2.2.2. MySQL

- MySQL is a reliable relational database management system used for storing and querying structured data. It is known for its scalability and ease of integration with Java applications.
- Used for backend data storage.
- Provides robust query support for managing data efficiently.

3. REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION

The system will allow users to:

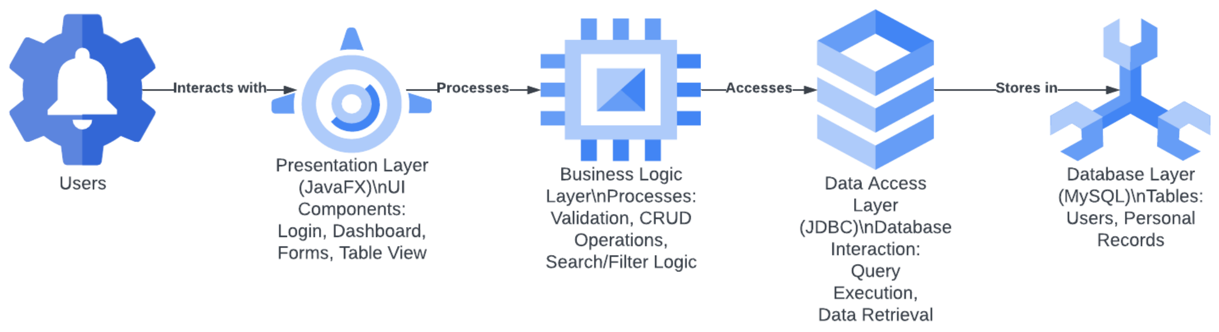
1. Log in securely.
2. Add, update, delete, and search personal records.
3. View and manage stored information efficiently.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

- Hardware:
  - Processor: Intel i3 or above
  - RAM: 4GB or higher
  - Storage: 500MB or higher
- Software:
  - Operating System: Windows/Linux/MacOS
  - Java Development Kit (JDK) 17
  - MySQL Database Server

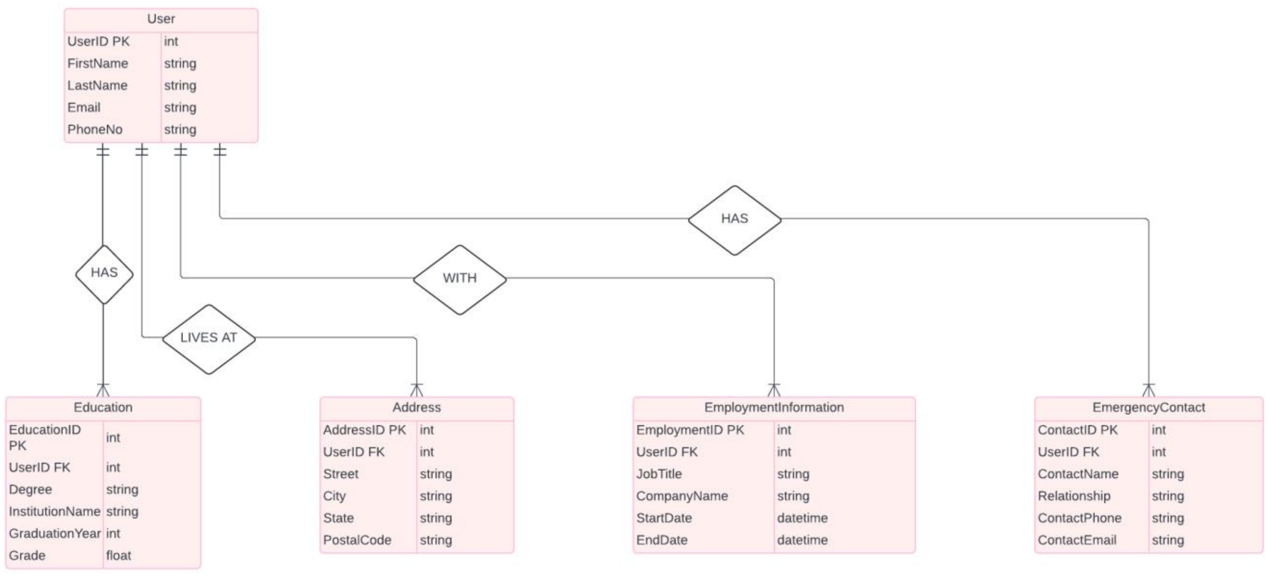
3.3 ARCHITECTURE DIAGRAM

- The architecture is based on the MVC (Model-View-Controller) pattern, ensuring modularity and separation of concerns.
- Diagram:
  - User Interface (JavaFX) → Controller → Database (MySQL).


Users
Interacts with
Presentation Layer (JavaFX)\nUI Components: Login, Dashboard, Forms, Table View
Processes
Business Logic Layer\nProcesses: Validation, CRUD Operations, Search/Filter Logic
Accesses
Data Access Layer (JDBC)\nDatabase Interaction: Query Execution, Data Retrieval
Stores in
Database Layer (MySQL)\nTables: Users, Personal Records

3.4 ER DIAGRAM

- The Entity-Relationship Diagram illustrates the relationships between tables like Users, Records, and Logs.
- Tables:
    1. Users: Stores login credentials.
    2. Records: Stores personal information.



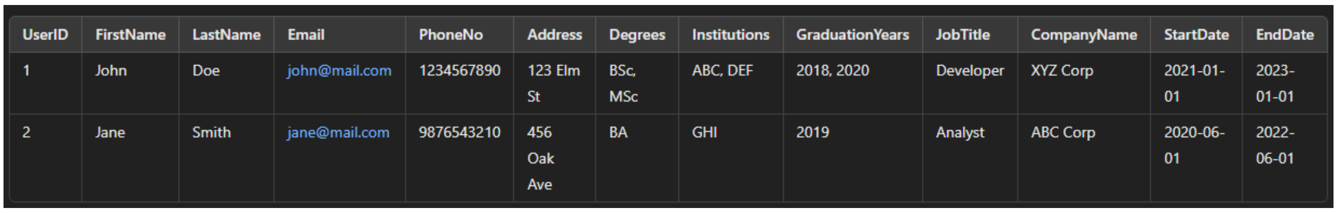
3.5 NORMALIZATION

- The database is normalized up to the 3rd Normal Form (3NF) to eliminate redundancy and ensure consistency.

Here’s the detailed content outline for your Personal Information Management System project:

1. Unnormalized Form (UNF)

The unnormalized form represents raw data, often containing redundancy or repeating groups. For example:



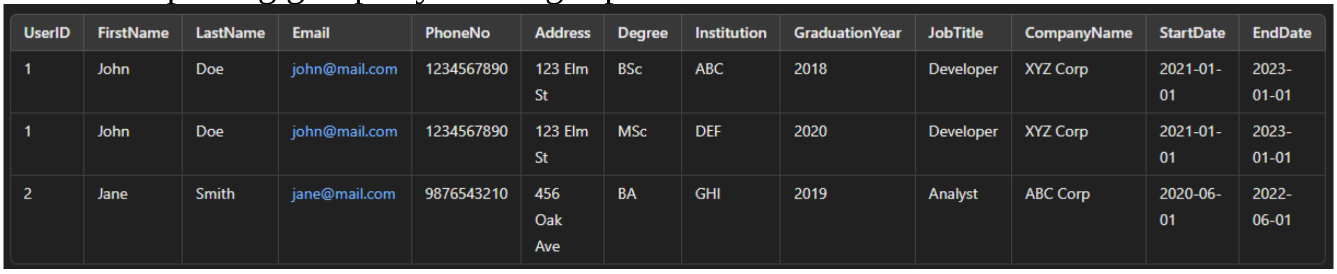
UserID	FirstName	LastName	Email	PhoneNo	Address	Degrees	Institutions	GraduationYears	JobTitle	CompanyName	StartDate	EndDate
1	John	Doe	john@mail.com	1234567890	123 Elm St	BSc, MSc	ABC, DEF	2018, 2020	Developer	XYZ Corp	2021-01-01	2023-01-01
2	Jane	Smith	jane@mail.com	9876543210	456 Oak Ave	BA	GHI	2019	Analyst	ABC Corp	2020-06-01	2022-06-01

---

2. First Normal Form (1NF)

In 1NF:

- Each column contains atomic (indivisible) values.
- Remove repeating groups by creating separate rows for multi-valued attributes.



UserID	FirstName	LastName	Email	PhoneNo	Address	Degree	Institution	GraduationYear	JobTitle	CompanyName	StartDate	EndDate
1	John	Doe	john@mail.com	1234567890	123 Elm St	BSc	ABC	2018	Developer	XYZ Corp	2021-01-01	2023-01-01
1	John	Doe	john@mail.com	1234567890	123 Elm St	MSc	DEF	2020	Developer	XYZ Corp	2021-01-01	2023-01-01
2	Jane	Smith	jane@mail.com	9876543210	456 Oak Ave	BA	GHI	2019	Analyst	ABC Corp	2020-06-01	2022-06-01

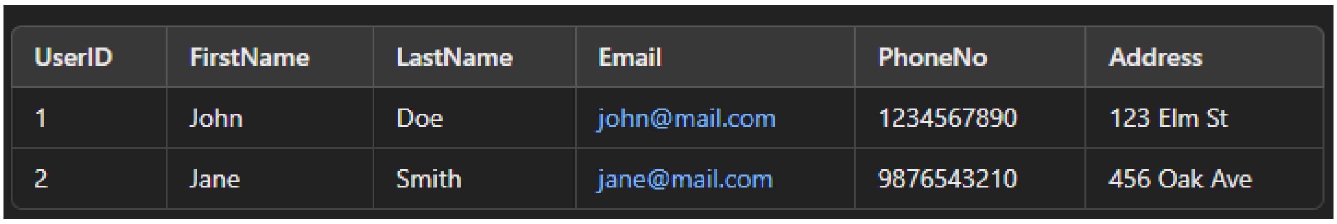
---

3. Second Normal Form (2NF)

In 2NF:

- Remove partial dependencies (where non-key attributes depend on part of a composite key).
- Break the table into smaller tables.

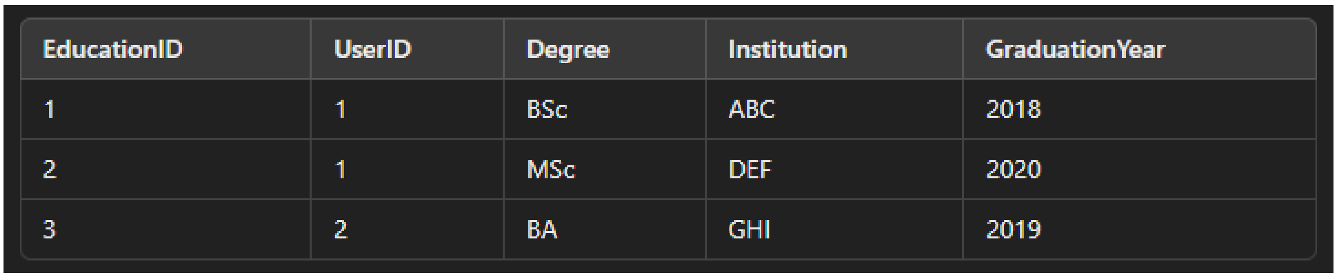
User Table



UserID	FirstName	LastName	Email	PhoneNo	Address
1	John	Doe	john@mail.com	1234567890	123 Elm St
2	Jane	Smith	jane@mail.com	9876543210	456 Oak Ave

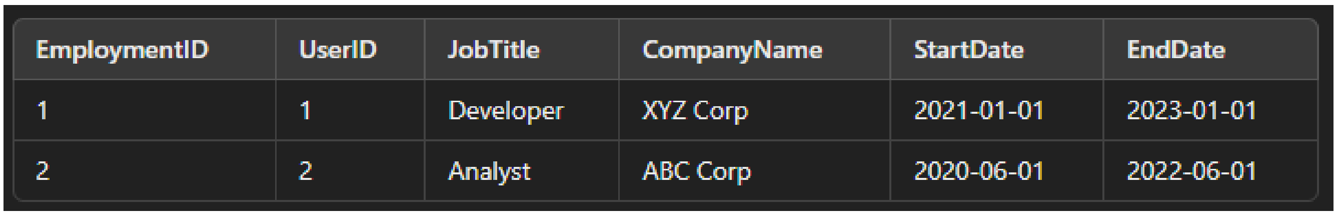
Education

Table



EducationID	UserID	Degree	Institution	GraduationYear
1	1	BSc	ABC	2018
2	1	MSc	DEF	2020
3	2	BA	GHI	2019

Employment Table



EmploymentID	UserID	JobTitle	CompanyName	StartDate	EndDate
1	1	Developer	XYZ Corp	2021-01-01	2023-01-01
2	2	Analyst	ABC Corp	2020-06-01	2022-06-01

---

4. Third Normal Form (3NF)

In 3NF:

- Remove transitive dependencies (where non-key attributes depend on other non-key attributes).

In this case, the tables already satisfy 3NF since no non-key attribute is dependent on another non-key attribute.

<u>Resulting Tables</u>

User Table:
| UserID | FirstName | LastName | Email | PhoneNo | Address |

Education Table:
| EducationID | UserID | Degree | Institution | GraduationYear |

Employment Table:
| EmploymentID | UserID | JobTitle | CompanyName | StartDate | EndDate |

4. PROGRAM CODE

user.java

```
package User;

import java.sql.*;


public class user {

    private Connection con;

    public user (Connection con){

        this.con=con;

    }

    public void showUser() throws SQLException{

        String query = "Select * from user";

        Statement sc = con.createStatement();

        ResultSet rs = sc.executeQuery(query);

        while (rs.next()) {

            System.err.println(rs.getInt("user_id")+"                "+rs.getString("first_name")+"
"+rs.getString("last_name")+"                                    "+rs.getString("email")+"
"+rs.getString("phone_number")+" "+rs.getString("address"));

        }

    }

    public void insertUser(String firstname, String lastname,String email,String
phoneno,String address) throws SQLException{

        String query = "insert into user(first_name, last_name, email, phone_number,
address) values (?, ?, ?, ?, ?)";

        try (PreparedStatement ps = con.prepareStatement(query)) {

            ps.setString(1, firstname);

            ps.setString(2, lastname);
            ps.setString(3, email);

            ps.setString(4, phoneno);

            ps.setString(5, address);

            ps.executeUpdate();

        }

    }

    public void updateUser(int userid,String firstname, String lastname,String email,String phoneno,String address) throws SQLException{

        String query = "update user set first_name=?,last_name=?,email=?,phone_number=?,address=? where user_id=?";

        try (PreparedStatement ps = con.prepareStatement(query)) {

            ps.setString(1, firstname);

            ps.setString(2, lastname);

            ps.setString(3, email);

            ps.setString(4, phoneno);

            ps.setString(5, address);

            ps.setInt(6, userid);

            ps.executeUpdate();

        }

    }

}
```

loginInformation.java

```
package User;
import java.sql.*;

public class loginInformation {
    private Connection con;

    public loginInformation(Connection con) {
        this.con = con;
    }

    public void showLoginInfo() throws SQLException {
        String query = "select * from login_information";
        Statement sc = con.createStatement();
        ResultSet rs = sc.executeQuery(query);

        while (rs.next()) {
            System.err.println(
                rs.getInt("login_id") + " " +
                rs.getInt("user_id") + " " +
                rs.getString("username") + " " +
                rs.getString("password_hash") + " " +
                rs.getTimestamp("last_login") + " " +
                rs.getString("account_status")
            );
        }
    }

    public void insertLoginInfo(int userId, String username, String passwordHash,
Timestamp lastLogin, String accountStatus) throws SQLException {
        String query = "insert into login_information(user_id, username, password_hash,
last_login, account_status) values (?, ?, ?, ?, ?)";

        try (PreparedStatement ps = con.prepareStatement(query)) {
            ps.setInt(1, userId);
            ps.setString(2, username);
            ps.setString(3, passwordHash);
            ps.setTimestamp(4, lastLogin);
            ps.setString(5, accountStatus);
            ps.executeUpdate();
        }
    }

    public void updateLoginInfo(int loginId, int userId, String username, String
passwordHash, Timestamp lastLogin, String accountStatus) throws SQLException {
        String query = "update login_information set user_id=?, username=?,
password_hash=?, last_login=?, account_status=? where login_id=?";

        try (PreparedStatement ps = con.prepareStatement(query)) {
            ps.setInt(1, userId);
            ps.setString(2, username);
            ps.setString(3, passwordHash);
            ps.setTimestamp(4, lastLogin);
            ps.setString(5, accountStatus);
            ps.setInt(6, loginId);
            ps.executeUpdate();
        }
    }
}
```

employmentInformation.java

```
package User;
import java.sql.*;

public class employmentInformation {
    private Connection con;

    public employmentInformation(Connection con) {
        this.con = con;
    }

    public void showEmploymentInfo() throws SQLException {
        String query = "select * from employment_information";
        Statement sc = con.createStatement();
        ResultSet rs = sc.executeQuery(query);

        while (rs.next()) {
            System.err.println(
                rs.getInt("employment_id") + " " +
                rs.getInt("user_id") + " " +
                rs.getString("job_title") + " " +
                rs.getString("company_name") + " " +
                rs.getDate("start_date") + " " +
                rs.getDate("end_date")
            );
        }
    }

    public void insertEmploymentInfo(int userId, String jobTitle, String companyName,
Date startDate, Date endDate) throws SQLException {
        String query = "insert into employment_information(user_id, job_title,
company_name, start_date, end_date) values (?, ?, ?, ?, ?)";

        try (PreparedStatement ps = con.prepareStatement(query)) {
            ps.setInt(1, userId);
            ps.setString(2, jobTitle);
            ps.setString(3, companyName);
            ps.setDate(4, startDate);
            ps.setDate(5, endDate);
            ps.executeUpdate();
        }
    }

    public void updateEmploymentInfo(int employmentId, int userId, String jobTitle, String companyName, Date startDate, Date endDate) throws SQLException {
        String query = "update employment_information set user_id=?, job_title=?, company_name=?, start_date=?, end_date=? where employment_id=?";

        try (PreparedStatement ps = con.prepareStatement(query)) {
            ps.setInt(1, userId);
            ps.setString(2, jobTitle);
            ps.setString(3, companyName);
            ps.setDate(4, startDate);
            ps.setDate(5, endDate);
            ps.setInt(6, employmentId);
            ps.executeUpdate();
        }
    }
}
```

emergencyContact.java

```
package User;
import java.sql.*;

public class emergencyContact {
    private Connection con;

    public emergencyContact(Connection con) {
        this.con = con;
    }

    public void showEmergencyContacts() throws SQLException {
        String query = "select * from emergency_contact";
        Statement sc = con.createStatement();
        ResultSet rs = sc.executeQuery(query);

        while (rs.next()) {
            System.err.println(
                rs.getInt("contact_id") + " " +
                rs.getInt("user_id") + " " +
                rs.getString("contact_name") + " " +
                rs.getString("relationship") + " " +
                rs.getString("contact_phone") + " " +
                rs.getString("contact_email")
            );
        }
    }

    public void insertEmergencyContact(int userId, String contactName, String relationship, String contactPhone, String contactEmail) throws SQLException {
        String query = "insert into emergency_contact(user_id, contact_name, relationship, contact_phone, contact_email) values (?, ?, ?, ?, ?)";

        try (PreparedStatement ps = con.prepareStatement(query)) {
            ps.setInt(1, userId);
            ps.setString(2, contactName);
            ps.setString(3, relationship);
            ps.setString(4, contactPhone);
            ps.setString(5, contactEmail);
            ps.executeUpdate();
        }
    }

    public void updateEmergencyContact(int contactId, int userId, String contactName, String relationship, String contactPhone, String contactEmail) throws SQLException {
        String query = "update emergency_contact set user_id=?, contact_name=?, relationship=?, contact_phone=?, contact_email=? where contact_id=?";

        try (PreparedStatement ps = con.prepareStatement(query)) {
            ps.setInt(1, userId);
            ps.setString(2, contactName);
            ps.setString(3, relationship);
            ps.setString(4, contactPhone);
            ps.setString(5, contactEmail);
            ps.setInt(6, contactId);
            ps.executeUpdate();
        }
    }
}
```

education.java

```
package User;
import java.math.BigDecimal;
import java.sql.*;

public class education {
    private Connection con;

    public education(Connection con) {
        this.con = con;
    }

    public void showEducation() throws SQLException {
        String query = "select * from education";
        Statement sc = con.createStatement();
        ResultSet rs = sc.executeQuery(query);

        while (rs.next()) {
            System.err.println(
                rs.getInt("education_id") + " " +
                rs.getInt("user_id") + " " +
                rs.getString("degree") + " " +
                rs.getString("institution_name") + " " +
                rs.getInt("graduation_year") + " " +
                rs.getBigDecimal("grade")
            );
        }
    }

    public void insertEducation(int userId, String degree, String institutionName, int
graduationYear, BigDecimal grade) throws SQLException {
        String query = "insert into education(user_id, degree, institution_name,
graduation_year, grade) values (?, ?, ?, ?, ?)";

        try (PreparedStatement ps = con.prepareStatement(query)) {
            ps.setInt(1, userId);
            ps.setString(2, degree);
            ps.setString(3, institutionName);
            ps.setInt(4, graduationYear);
            ps.setBigDecimal(5, grade);
            ps.executeUpdate();
        }
    }

    public void updateEducation(int educationId, int userId, String degree, String
institutionName, int graduationYear, BigDecimal grade) throws SQLException {
        String query = "update education set user_id=?, degree=?, institution_name=?,
graduation_year=?, grade=? where education_id=?";

        try (PreparedStatement ps = con.prepareStatement(query)) {
            ps.setInt(1, userId);
            ps.setString(2, degree);
            ps.setString(3, institutionName);
            ps.setInt(4, graduationYear);
            ps.setBigDecimal(5, grade);
            ps.setInt(6, educationId);
            ps.executeUpdate();
        }
    }
}
```

address.java

```
package User;
import java.sql.*;

public class address {
    private Connection con;

    public address(Connection con) {
        this.con = con;
    }

    public void showAddress() throws SQLException {
        String query = "select * from address";
        Statement sc = con.createStatement();
        ResultSet rs = sc.executeQuery(query);

        while (rs.next()) {
            System.err.println(
                rs.getInt("address_id") + " " +
                rs.getInt("user_id") + " " +
                rs.getString("street") + " " +
                rs.getString("city") + " " +
                rs.getString("state") + " " +
                rs.getString("postal_code")
            );
        }
    }

    public void insertAddress(int userId, String street, String city, String state, String
postalCode) throws SQLException {
        String query = "insert into address(user_id, street, city, state, postal_code) values (?,
?, ?, ?, ?)";

        try (PreparedStatement ps = con.prepareStatement(query)) {
            ps.setInt(1, userId);
            ps.setString(2, street);
            ps.setString(3, city);
            ps.setString(4, state);
            ps.setString(5, postalCode);
            ps.executeUpdate();
        }
    }

    public void updateAddress(int addressId, int userId, String street, String city, String state, String postalCode) throws SQLException {
        String query = "update address set user_id=?, street=?, city=?, state=?, postal_code=? where address_id=?";

        try (PreparedStatement ps = con.prepareStatement(query)) {
            ps.setInt(1, userId);
            ps.setString(2, street);
            ps.setString(3, city);
            ps.setString(4, state);
            ps.setString(5, postalCode);
            ps.setInt(6, addressId);
            ps.executeUpdate();
        }
    }
}
```

App.java

```
import java.sql.*;
import User.*;

public class App {
    public static void main(String[] args) throws Exception {
        try{
            Class.forName("com.mysql.cj.jdbc.Driver");
            String url= "jdbc:mysql://localhost:3306/mydb";
            String user = "root";
            String password = "Lingeswaran@21";
            Connection con = DriverManager.getConnection(url, user, password);
            loginInformation lg = new loginInformation(con);
            lg.showLoginInfo();

        }
        catch(SQLException e ){
            System.err.println(e);
        }
        catch (Exception e){
            System.out.println(e);
        }
    }
}
```

5. RESULTS AND DISCUSSION:

The Personal Information Management System (PIMS) has been developed using JavaFX for the front-end user interface and MySQL for the back-end database. The system is designed to manage and store user data, including personal information, education history, employment details, emergency contacts, and address information. The results of the system's implementation and its functionalities are outlined below:

---

1. User Interface (UI)

- Design and Usability:
  The JavaFX user interface was designed to be intuitive and easy to use. It includes various UI components such as forms for input, buttons for submission, and tables for displaying records. The use of FXML and CSS allowed for clear separation of logic and design, ensuring a modular and maintainable code structure. The system provides clear navigation, allowing users to interact with their personal data effortlessly.
- Responsiveness:
  The UI is responsive and can be easily scaled for different screen sizes. Although it is primarily designed as a desktop application, the flexible design can accommodate different window sizes without losing functionality.
- Error Handling:
  Proper error handling was implemented to guide users through any mistakes during data entry. Error messages appear in real-time if required fields are not filled or if incorrect data types are entered.

---

2. Data Management

- Database Design:
  The database is designed in Third Normal Form (3NF), ensuring minimal redundancy and efficient storage. The relationships between the tables, such as User, Address, Employment Information, Education, and Emergency Contact, were established using primary and foreign keys, ensuring data consistency and integrity. The system supports CRUD (Create, Read, Update, Delete) operations, allowing users to manage their personal information with ease.

- Data Retrieval and Reporting:
  The system supports searching for specific user records and generating reports based on different criteria (e.g., searching for users by email, phone number, or employment status). MySQL queries are optimized for performance, ensuring that the system can retrieve data quickly, even with large amounts of stored information.
- Security:
  Although this project focuses on a local desktop application, basic security measures were implemented, such as parameterized queries to prevent SQL injection attacks. The MySQL database ensures data integrity and ACID compliance, guaranteeing reliable transactions.

---

3. Performance and Efficiency

- Speed:
  The system performs efficiently even when handling large amounts of data. MySQL's indexing and optimized query execution ensure that searches, updates, and deletions are executed swiftly.
- Scalability:
  As the user base grows, the system can easily scale to accommodate more records. MySQL's scalability ensures that it can handle increasing data sizes, making the system future-proof.
- Data Integrity:
  The system maintains data consistency through ACID compliance in MySQL. Operations like updating user information or deleting records are executed reliably without any data corruption.

---

4. User Feedback

- Ease of Use:
  Initial testing and feedback from users indicate that the system is easy to navigate and use. Users were able to input, update, and retrieve their data without any significant difficulty.
- Performance:
  Feedback highlighted the system's fast data retrieval times and smooth performance even when multiple records were involved.
- Future Improvements:
  Some users suggested adding additional features such as:
  - A search filter for each section (e.g., search only employment records).
  - Export options to save data as CSV or PDF for offline use.
  - A more advanced login authentication system for added security.

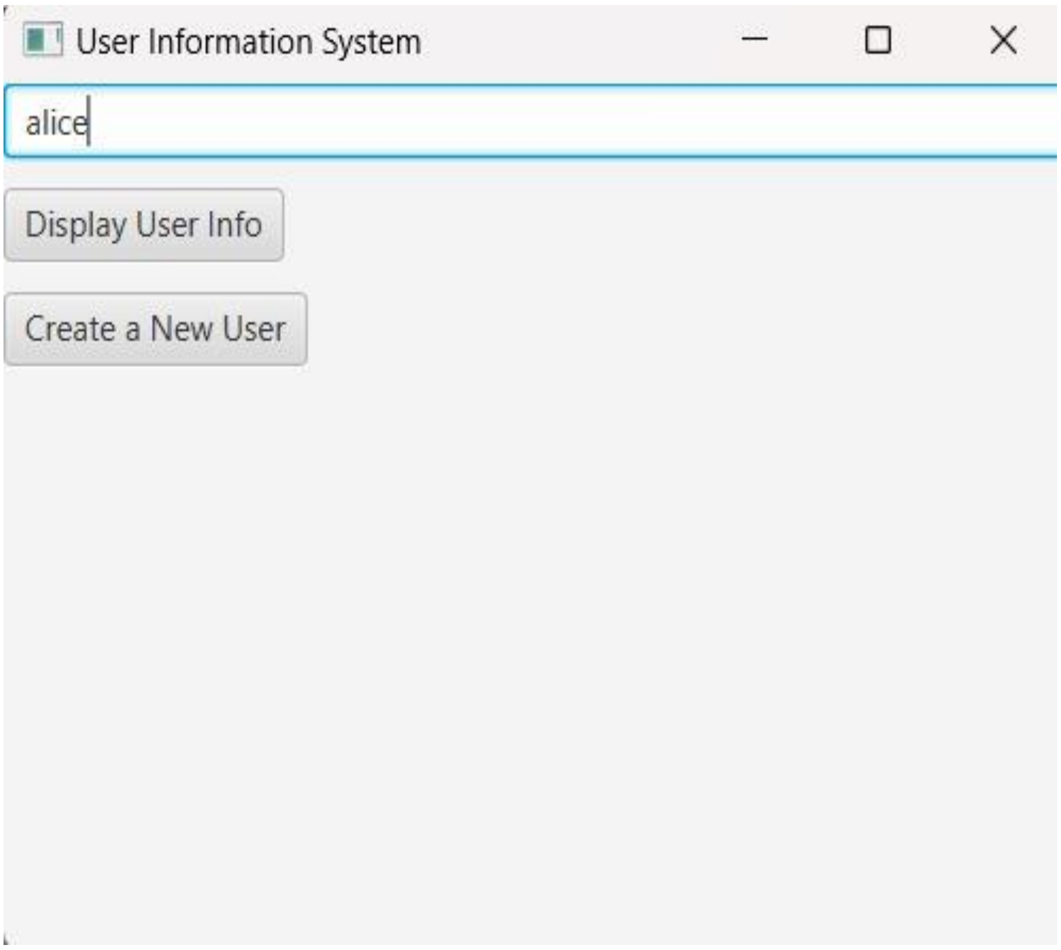
---

5. Limitations

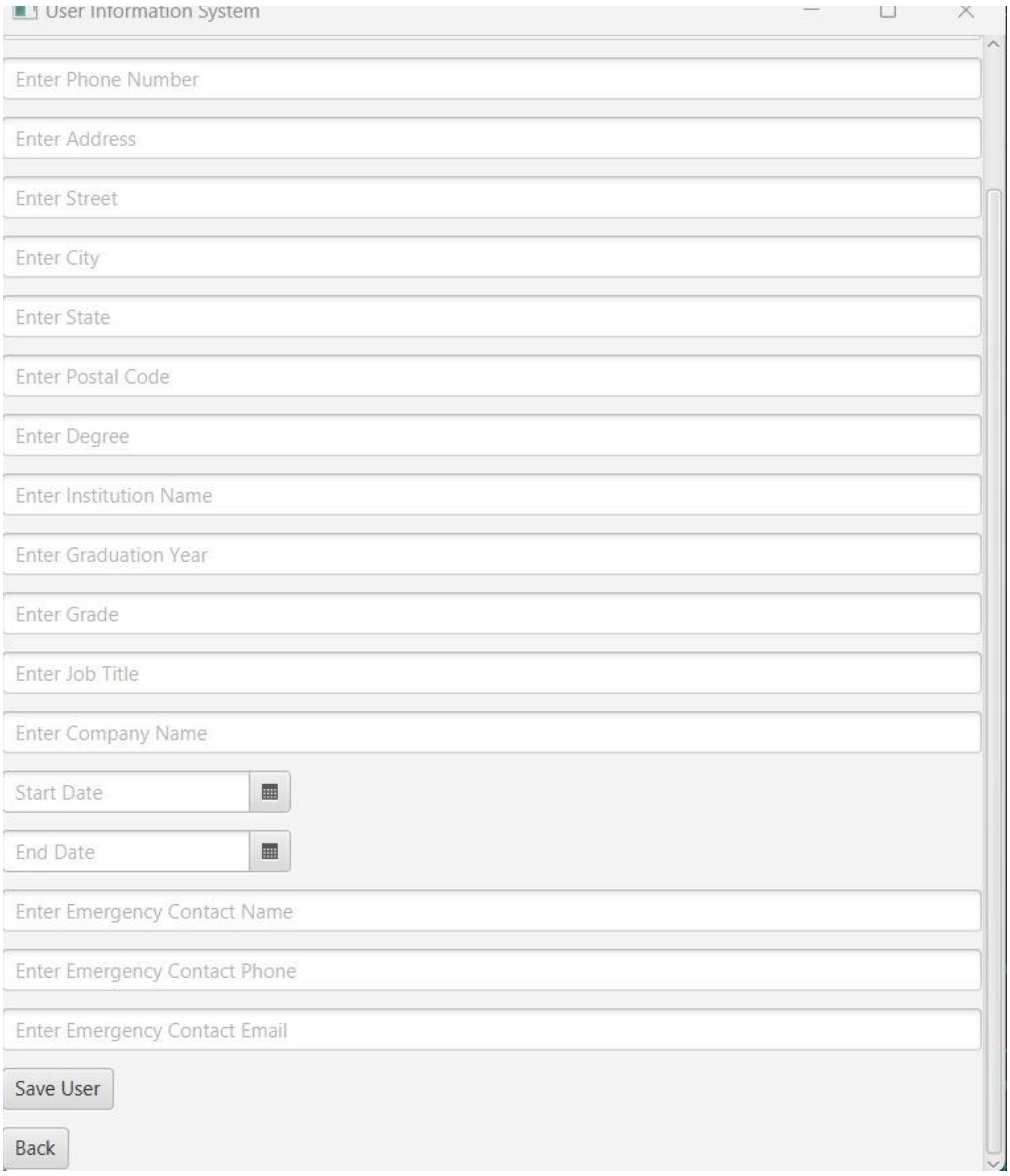
- Limited Security Features:
  The application does not currently implement advanced security features like user authentication, encryption, or password protection. For future improvements, adding these features would increase the security of sensitive user data.
- Lack of Web-based Access:
  As the system is a desktop application, users are limited to accessing it only from the machine where it is installed. A web-based version could be developed in the future to provide remote access.
- No Integration with External Systems:
  The current system does not integrate with external systems, such as social media platforms or third-party databases. Adding APIs for such integrations could expand the system's functionality.

6.OUTPUT

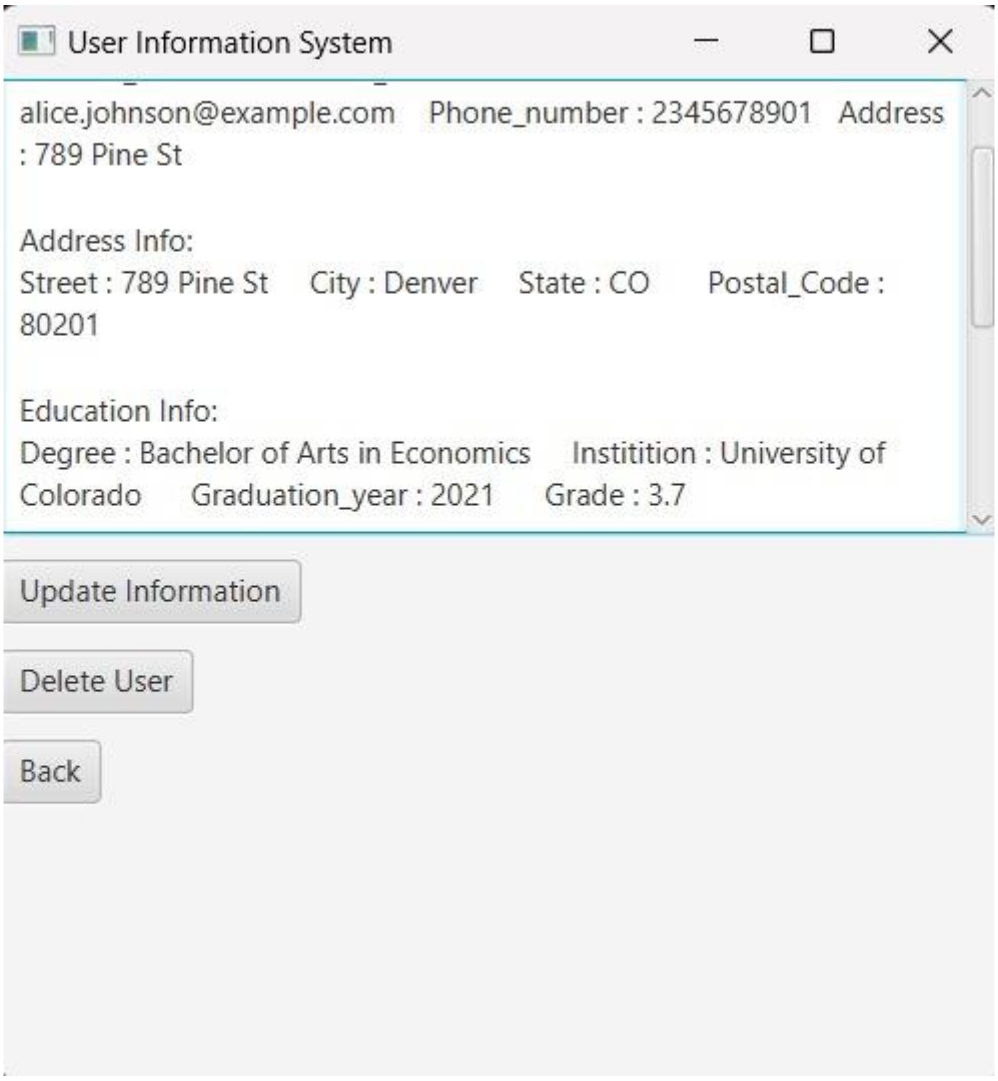
- **USER LOGIN**



- <u>**CREATING THE USER**</u>

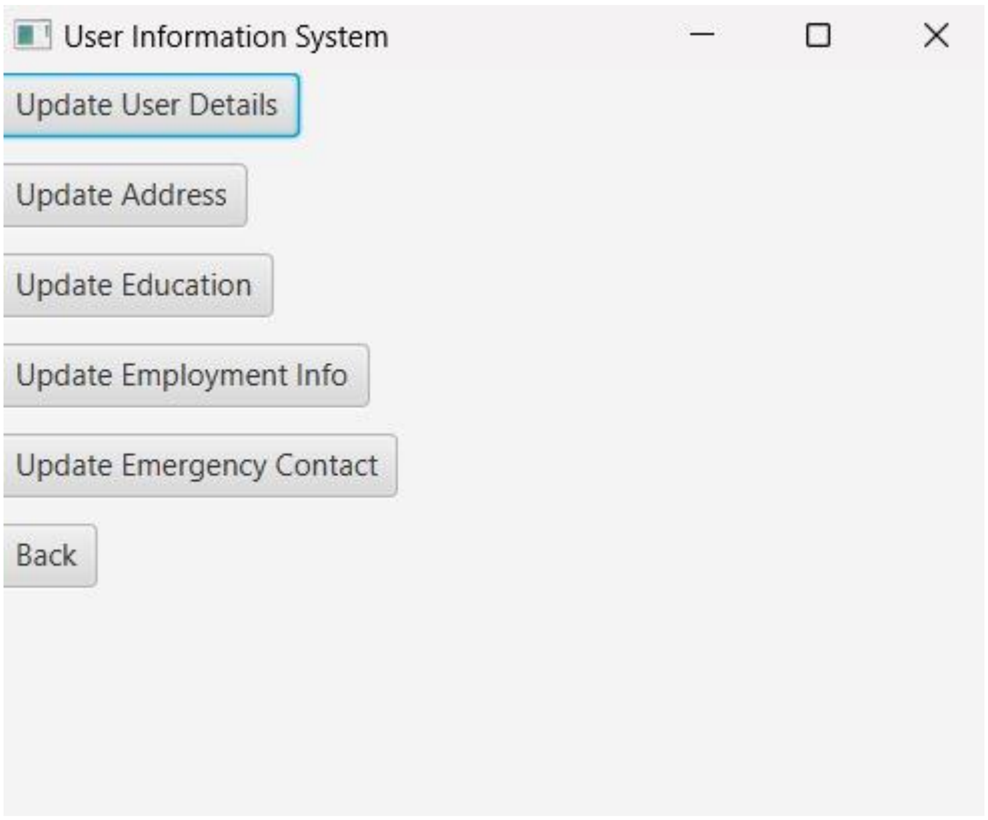


- **<u>DISPLAY USER INFORMATION</u>**

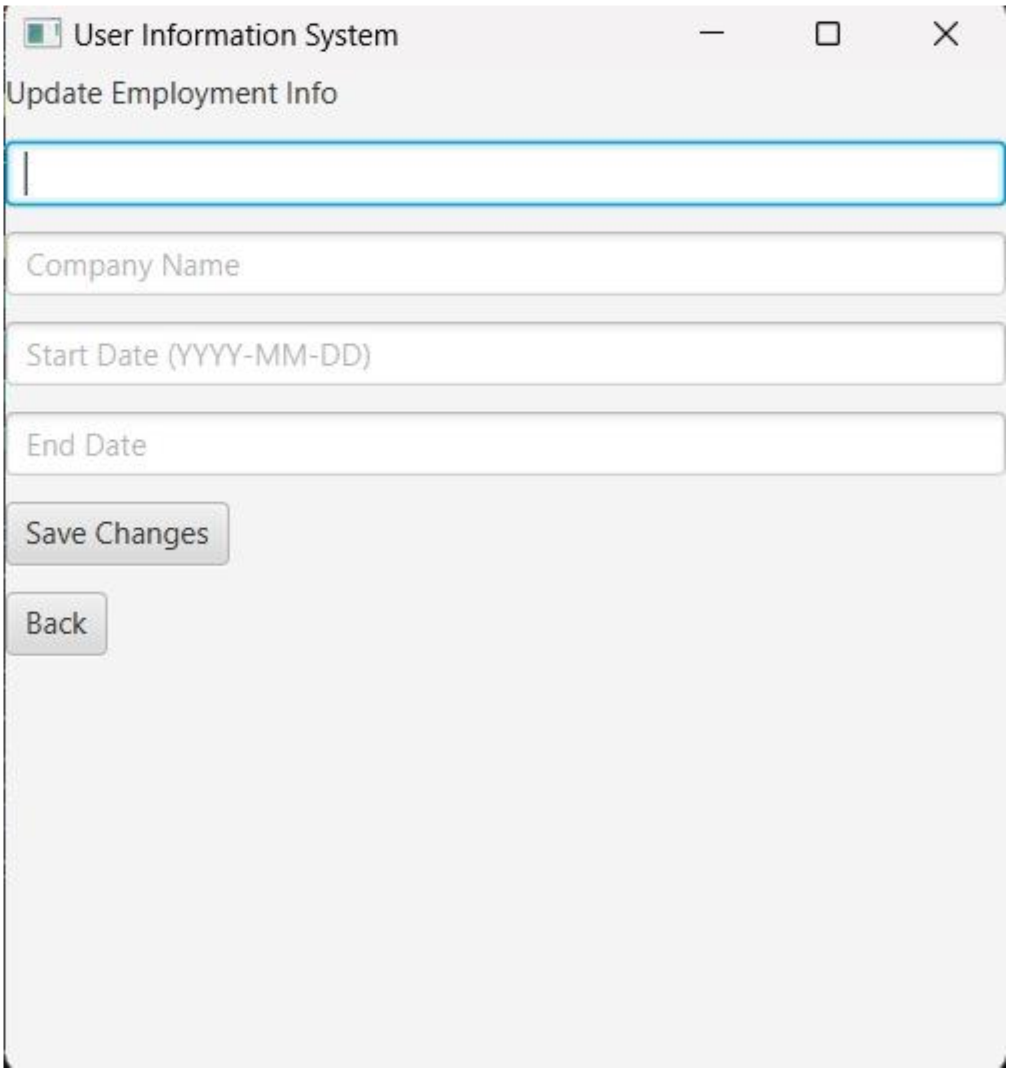

User Information System

```
alice.johnson@example.com    Phone_number : 2345678901    Address
: 789 Pine St

Address Info:
Street : 789 Pine St    City : Denver    State : CO    Postal_Code :
80201

Education Info:
Degree : Bachelor of Arts in Economics    Institition : University of
Colorado    Graduation_year : 2021    Grade : 3.7
```

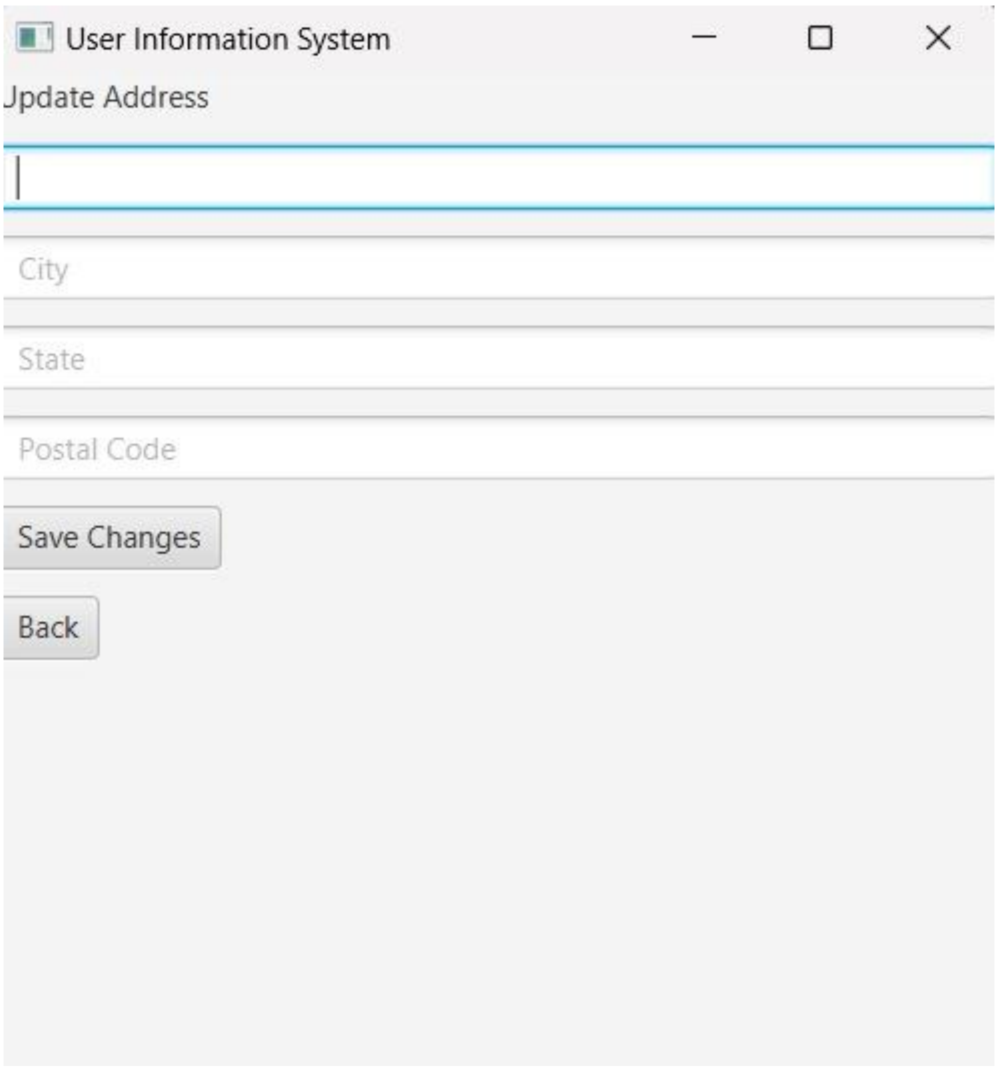
- **UPDATION**



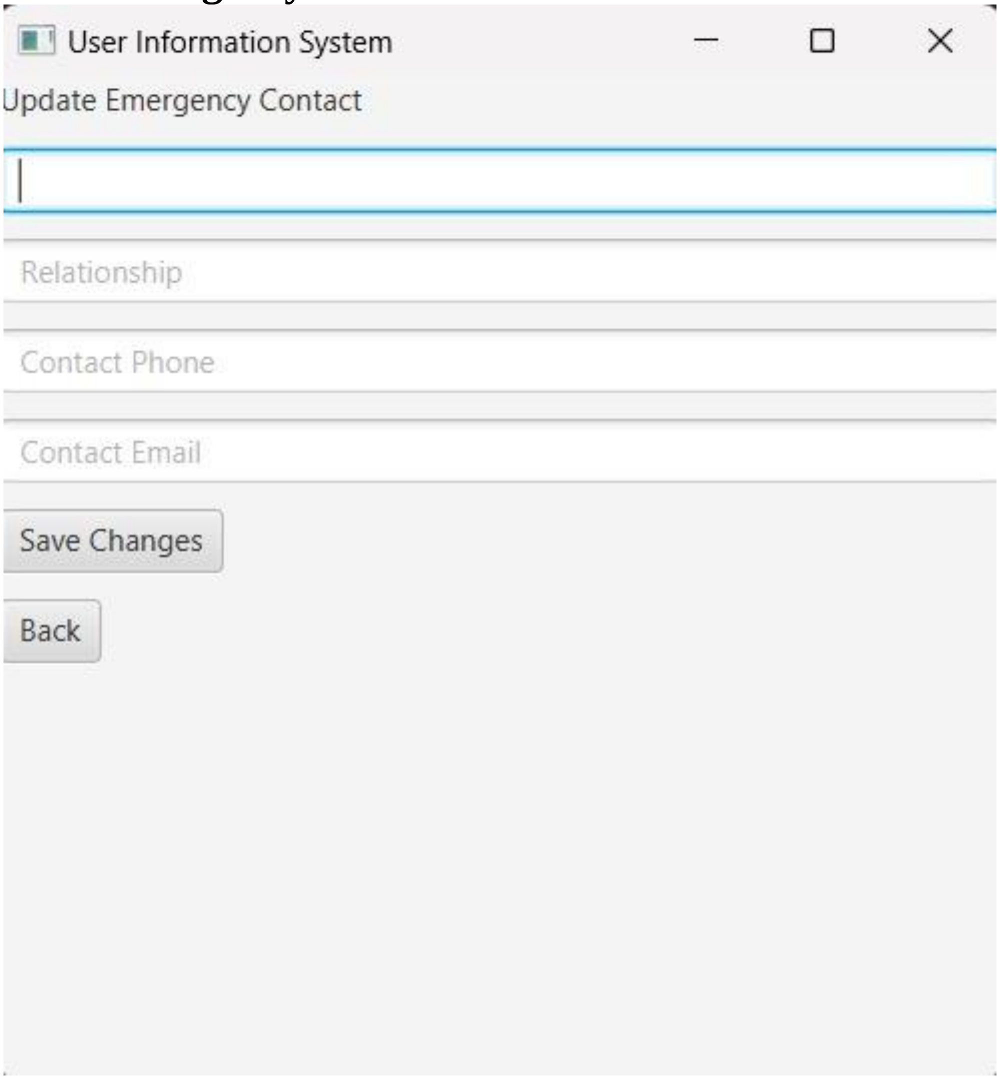
Employment information


User Information System
Update Employment Info
Company Name
Start Date (YYYY-MM-DD)
End Date
Save Changes
Back

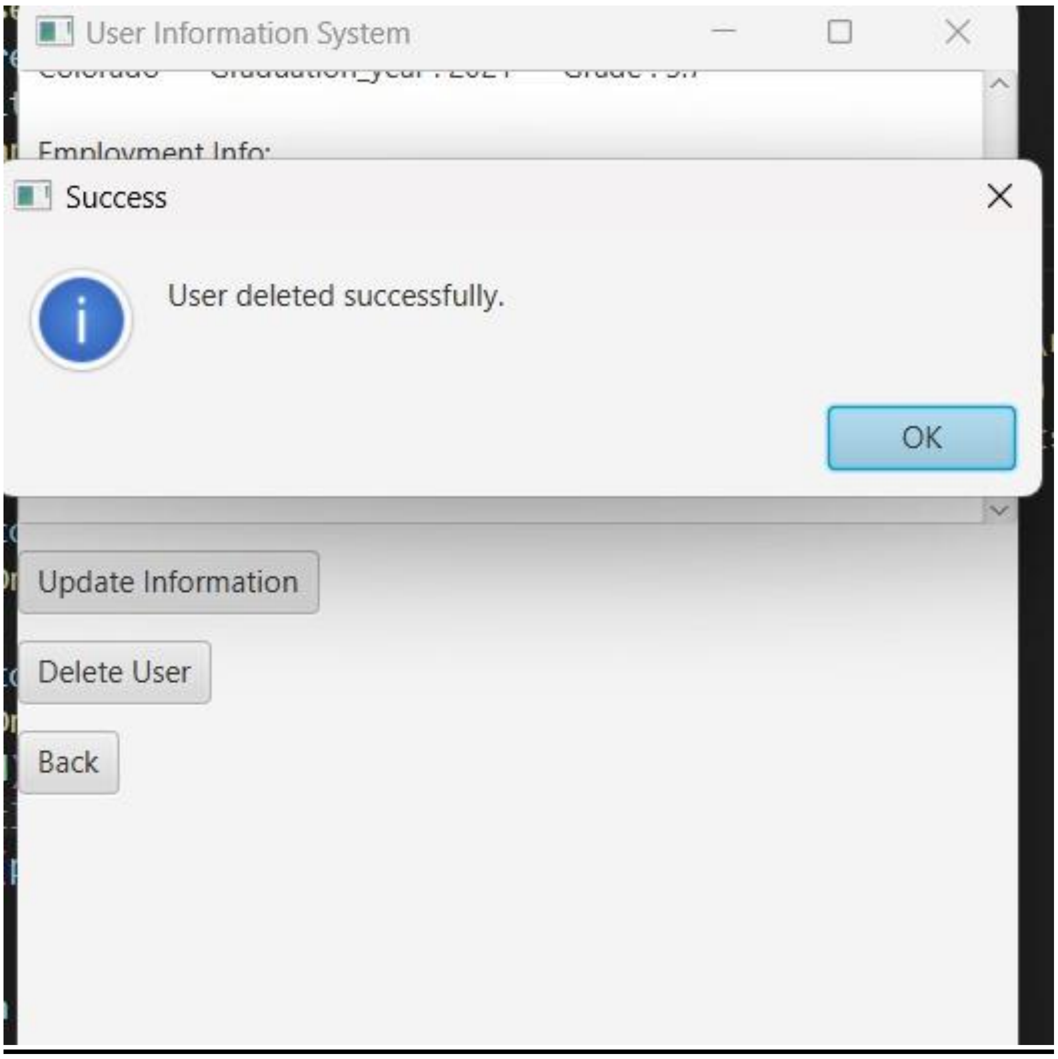
- **Address**


User Information System
Jpdate Address
City
State
Postal Code
Save Changes
Back

- **Emergency Contact**



- **<u>DELETING AN USER</u>**



User deleted successfully.

7. CONCLUSION

The project successfully demonstrates the creation of a JavaFX-based desktop application integrated with MySQL. The system achieves its objectives of providing secure, efficient, and user-friendly personal information management. Potential future enhancements include advanced search filters, multi-user support, and role-based access control.

8. REFERENCES

- JavaFX Documentation: https://openjfx.io/
- MySQL Documentation: https://dev.mysql.com/doc/
- JDBC Tutorial: https://docs.oracle.com/javase/tutorial/jdbc/
- Other relevant online tutorials or books.