

EMPLOYEE MANAGEMENT SYSTEM

CS23333 – Object Oriented Programming using Java Project Report

Submitted by

AADITHYA. B -231001002

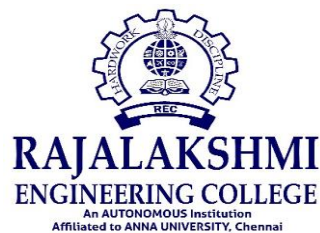
ADITHYA. B -231001008

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY



DEPARTMENT OF INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE

NOVEMBER-2024

BONAFIDE CERTIFICATE

Certified that this project titled “Employee Management System” is the bonafide work of “**AADITHYA(231001002), ADITHYA(231001008)** ” who carried out the project work under my supervision.

SIGNATURE

Dr.P.Valarmathie

HEAD OF THE DEPARTMENT

Department of Information Technology
Rajalakshmi Engineering College

SIGNATURE

Mrs.Usha S

COURSE INCHARGE
Assistant Professor(S.G)

Department of Information Technology
Rajalakshmi Engineering College

This project is submitted for CS23333 – Object Oriented Programming using Java held on _____

INTERNAL EXAMINAR

EXTERNAL EXAMINAR

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	List of Figures	4
	List of Tables	5
1	1.1 Abstract	6
	1.1 Introduction	6
	1.3 Purpose	6
	1.4 Scope of Project	7
	1.5 Software Requirement Specification	7
2	System flow Diagrams	12
	2.1 Use Case Diagram	12
	2.2 Entity-relationship Diagrams	13
	2.3 Data Flow Diagram	13
3	Module Description	14
4	4.1 Design	15
	4.2 Database Design	18
	4.3 Code	19
5	Conclusion	30
6	Reference	30

LIST OF FIGURES

Figure Numbers	Figure Captions	Pg.no
2.1	Use Case Diagram	12
2.2	Entity Relation diagram	13
2.3	Date Flow Diagram	13
4.1.1	Login Page	15
4.1.2	Main Page	15
4.1.3	Add Employee Page	16
4.1.4	Delete Selection	16
4.1.5	View Employee Page	17
4.1.6	Update Employee	17

LIST OF Table

Figure Numbers	Figure Captions	Pg.no
4.2.1	Login Table	18
4.2.2	Employee Table	18

1.1 Abstract

The primary objective of the JDBC-powered **Employee Management System** is to provide a reliable and efficient platform for managing employee information within an organization. This system leverages Java's database connectivity capabilities to seamlessly add, update, delete, and retrieve employee records, ensuring streamlined operations and effective data management. By utilizing JDBC transactions, the system guarantees data integrity and consistency, establishing a secure connection with a relational database for real-time updates on employee details. The intuitive user interface, developed using Java Swing and AWT, enhances user experience, while the scalability of the application makes it suitable for organizations of varying sizes.

1.2 Introduction

The **Employee Management System** is a comprehensive solution aimed at transforming traditional employee data management processes. By automating tasks such as adding, updating, and retrieving records, the system reduces time and operational costs. Built using Java Swing for an intuitive interface and powered by JDBC for secure database connectivity with MySQL, it ensures real-time updates and accurate record-keeping. Only authorized personnel can access and manage the data, ensuring security and reliability. This system enhances the efficiency and effectiveness of HR operations, making it an indispensable tool for modern organizations.

1.3 Purpose

The purpose of this project is to develop a reliable and efficient system for managing employee data within organizations. The system aims to:

- Streamline HR processes by automating tasks such as adding, updating, and deleting employee records.
- Provide a user-friendly interface for efficient data entry and retrieval.
- Ensure data consistency and accuracy through secure database connectivity using JDBC.

- Reduce manual errors and operational costs by modernizing traditional employee management workflows.

1.4 Scope of the Project:

The Employee Management System automates HR processes to efficiently manage employee data. Built using Java with JDBC for secure database connectivity and MySQL as the backend, the system streamlines tasks such as adding, updating, deleting, and viewing employee records. It ensures secure access through authentication, allowing only authorized personnel to handle sensitive data. With real-time updates, data validation, and accurate record-keeping, the system enhances efficiency and reliability. Featuring a user-friendly Java Swing interface, it is scalable for organizations of various sizes, offering detailed reporting and improved decision-making, modernizing traditional employee management workflows.

1.5 Software Requirement Specification

Introduction

The Employee Management System is a Java-based solution designed to automate and streamline employee data management. It enables efficient record handling, secure access, real-time updates, and accurate reporting, improving overall HR processes and operational efficiency within organizations.

Product Scope

The Employee Management System streamlines HR tasks by automating employee record management. Built with Java and MySQL, it ensures secure, real-time updates and data validation. The system offers scalable, user-friendly features for managing employee data, generating reports, and improving decision-making.

References and Acknowledgement

[1] <https://www.javatpoint.com/java-awt>

[2] <https://www.javatpoint.com/java-swing>

Overall Description

The Employee Management System is a Java-based solution that automates HR processes, securely manages employee records using MySQL, and provides real-time updates. It features a user-friendly interface, ensuring efficient record management, reporting, and improved decision-making.

Product Perspective

The system is designed to be compatible with the Microsoft Windows Operating System. The front end is developed using Java AWT and SWING, while the backend leverages the MySQL server for efficient data management.

Product Functionality

- a) Admin Login: Enables existing administrators to log in securely.
- b) Add Employee: Facilitates the addition of new Employee details.
- c) View Employee: Allows users to view and update existing Employee information.
- d) Delete Employee: Permits the removal of existing Employee from the system.
- e) Update Employee: Allows users to view and modify existing Employee.
- f) Remove Admin: Provides the option to delete specific administrator accounts.
- g) Print Details: Allows the option to print the details of all the employees.
- h) Search Employee: Facilitates the search operation for the employees.

User and Characteristics

Qualification: Users should have at least basic educational qualifications, such as matriculation, and be comfortable with English.

Experience: Familiarity with the university registration process is advantageous. Technical

Experience: Users are expected to have elementary knowledge of computers for optimal system interaction.

Operating Environment

Hardware Requirements

- Processor: Intel i3 or higher (or equivalent AMD processor)
- Operating System: Windows 8,10, 11
- Processor Speed: 2.0 GHz
- RAM: 4GB
- Hard Disk: 500GB

Software Requirements

- Database: MySQL
- Frontend: Java (SWING, AWT)
- Technology: Java (JDBC)

Constraints

- System access is limited to authorized personnel for security purposes.
- The system is designed to handle a moderate volume of employee data; scalability for very large datasets may require optimization.

User Interface

The Employee Management System provides user-friendly, menu driven interfaces for

- a) Add Employee: Create new employee with details.
- b) View Employee: View the employee details
- c) Delete Employee: Delete the Employee .
- d) Update Employee: Allows users to view and modify existing Employee.
- e) Remove Admin: Provides the option to delete specific administrator accounts.
- f) Print Details: Print the details of all the employees.
- g) Search Employee: Search the employee.

Hardware Interface

- Screen resolution of at least 640 x 480 or above.
- Compatible with any version of Windows 8, 10, 11.

Software Interface

- a) MS-Windows Operating System
- b) Java AWT and SWING for designing the front end
- c) MySQL for the backend
- d) Platform: Java Language
- e) Integrated Development Environment (IDE): IntelliJ IDEA

Functional Requirements

- User Authentication and Access Control:
 - Admins must log in using a username and password (passwords are masked for security).
 - Only authorized administrators can access the system.
 - Successful login verification with the database is required for access.

Employee Management

- Admins can add, update, delete, and view employee records.
- The system supports searching for specific employees based on various parameters.
- Admins can print detailed reports of employee information.

Admin Management

- Admins can manage other administrator accounts, including adding, updating, or deleting admin records.
- Only authorized users can modify admin accounts.

Database Interaction

- The system communicates with the MySQL database via JDBC for CRUD operations on employee and admin data.
- Ensures data consistency, security, and integrity while handling employee and admin records.

Non-functional Requirements

1)Performance

- The system must efficiently handle employee record operations, ensuring a response time of less than 2 seconds for adding, updating, or deleting records.
- Any critical system failures, such as issues with data retrieval or updates, must be addressed immediately to maintain smooth functionality.

2)Reliability

- The system should be highly reliable, ensuring no data loss or corruption during operations.
- *In case of any failure or downtime, immediate corrective actions must be taken to restore normal functionality.*

3)Availability

- The system should always be available and responsive, with minimal downtime.
- All user requests, including searching, viewing, and managing employee data, should be processed within 2 seconds to ensure a seamless user experience.

4)Security

- A secure authentication system must be implemented to prevent unauthorized access to the system.
- Employee data, including personal and sensitive information, must be securely stored and protected from unauthorized access.
- The system should use encryption and other security protocols to safeguard data integrity and confidentiality.

5)Maintainability

- Design and documentation for system maintenance, including regular updates and database management, should be available for future modifications.
- Administrative access must be provided for performing system maintenance, ensuring its long-term stability and adaptability.

2.SYSTEM FLOW DIAGRAMS

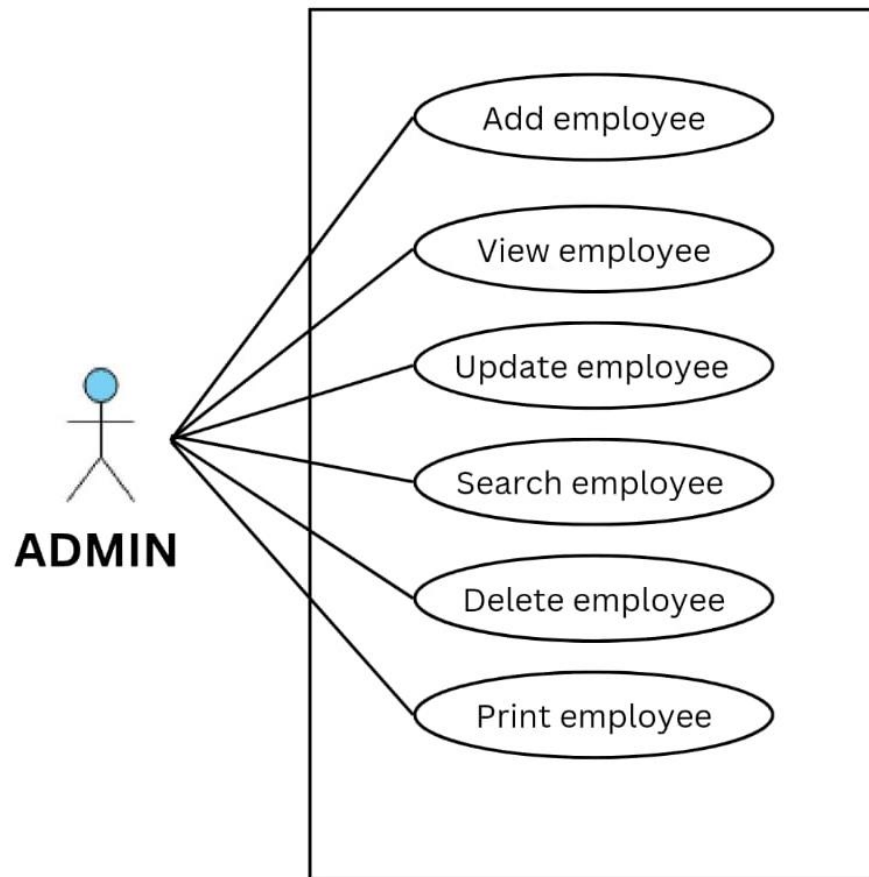


Figure 2.1 Use Case Diagrams

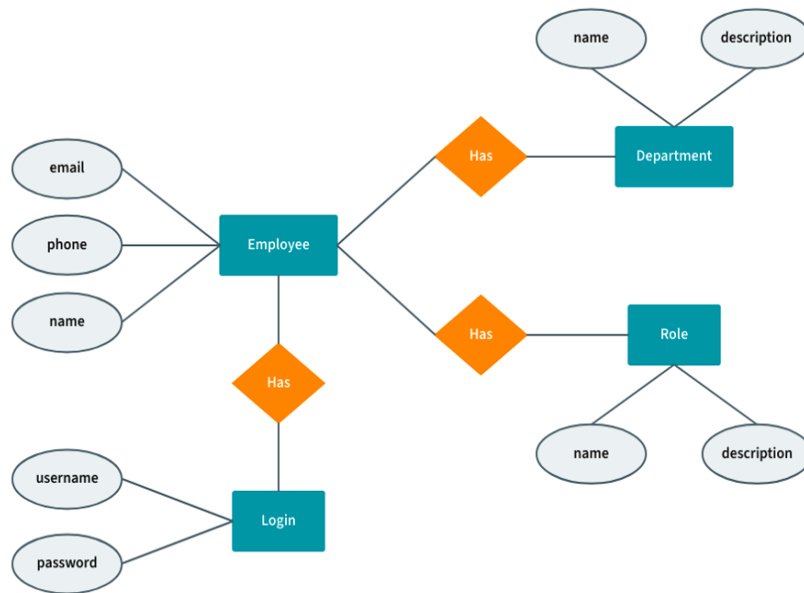


Figure 2.2 Entity Relationship Diagram

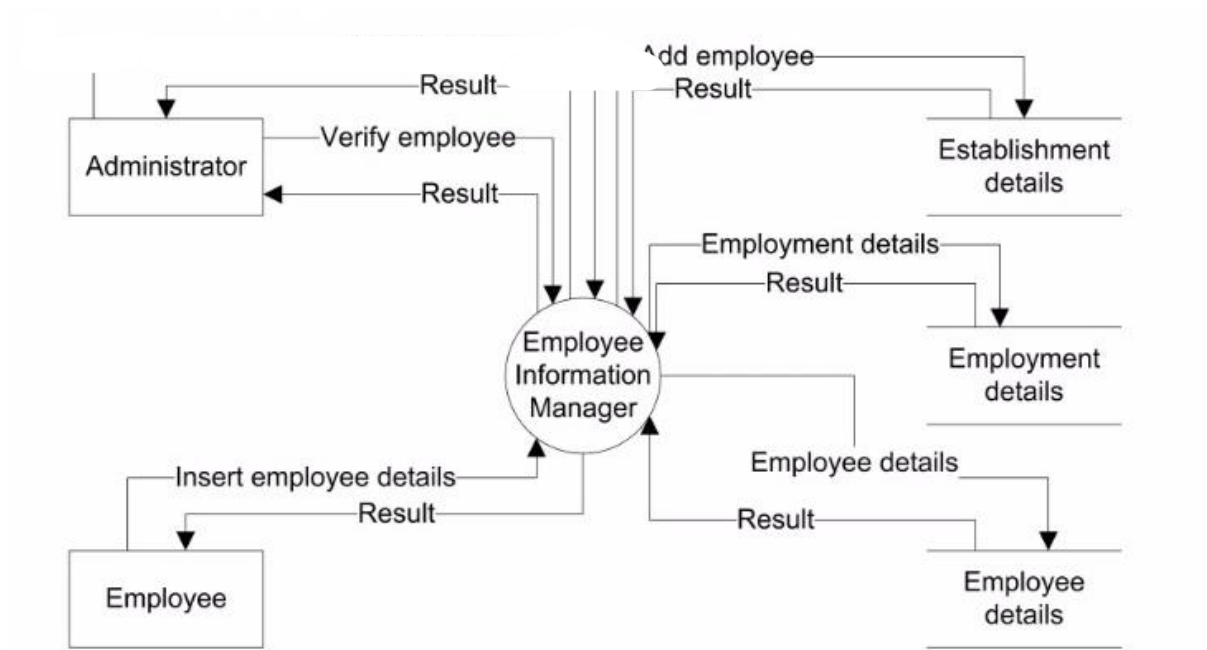
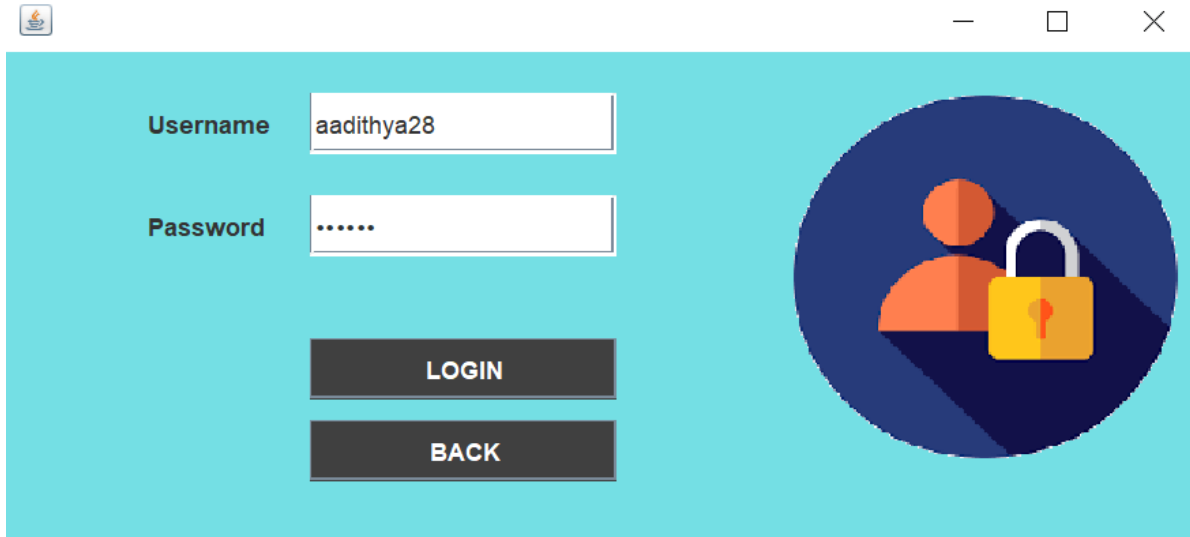


Figure 2.3 Data-flow diagram

3. MODULE DESCRIPTION

1. Admin Login: Provides a secure login interface for existing admins to authenticate and access the system.
2. Add Employee: Allows the admin to create and store new employee records by entering their details, such as name, ID, and designation.
3. View Employee: Enables the admin to view detailed information of individual employees, such as their personal details and job-related data.
4. Delete Employee: Gives the admin the ability to remove an employee record from the system permanently.
5. Update Employee: Lets the admin view and update an employee's details, including modifications to their personal and job information.
6. Remove Admin: Allows the deletion of a specific administrator account, managing system access and privileges for admins.
7. Print Details: Provides the option to generate and print a comprehensive list of all employees in the system.
8. Search Employee: Allows the admin to search for employees by specific criteria, such as name or employee ID, to quickly find their records.

4.1 Design



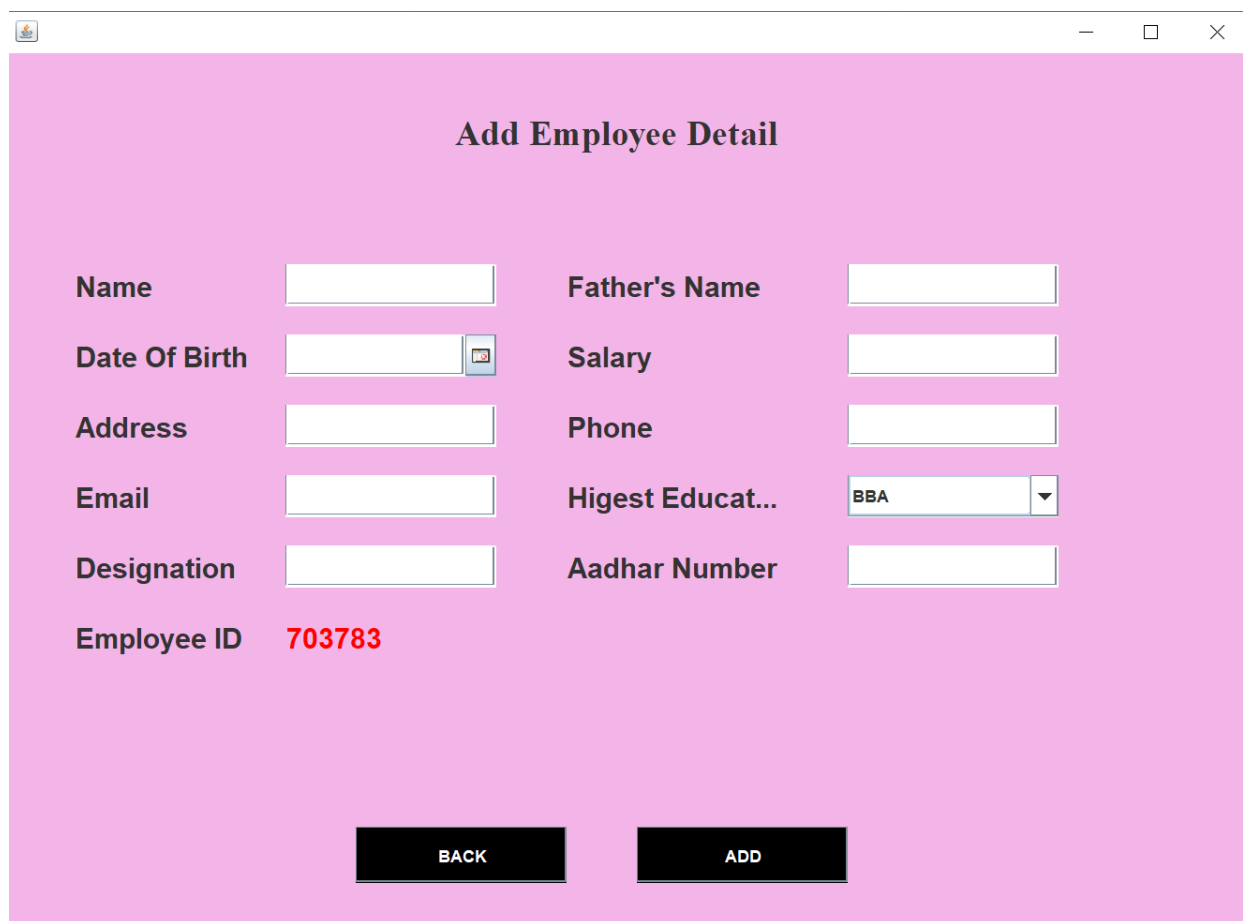
A screenshot of a login page within a window. The window has a standard title bar with minimize, maximize, and close buttons. The login page has a light blue background. On the left, there are two input fields: 'Username' with the text 'aadithya28' and 'Password' with masked characters '.....'. Below these fields are two dark grey buttons labeled 'LOGIN' and 'BACK'. On the right side, there is a large circular graphic with a dark blue background, featuring an orange silhouette of a person and a yellow padlock.

Figure 4.1.1 Login Page



A screenshot of the main page of an 'Employee Management System'. The window has a title bar with minimize, maximize, and close buttons, and an 'Exit' button in the top right corner. The page has a dark blue background. At the top left, the title 'Employee Management System' is displayed in large white text. Below the title is a large illustration of several stylized people in various colors (grey, purple, yellow, blue, green, orange) working together to assemble a large, interconnected gear system. On the right side of the page, there are three dark grey buttons stacked vertically, labeled 'Add Employee', 'Remove Employee', and 'View Employee'.

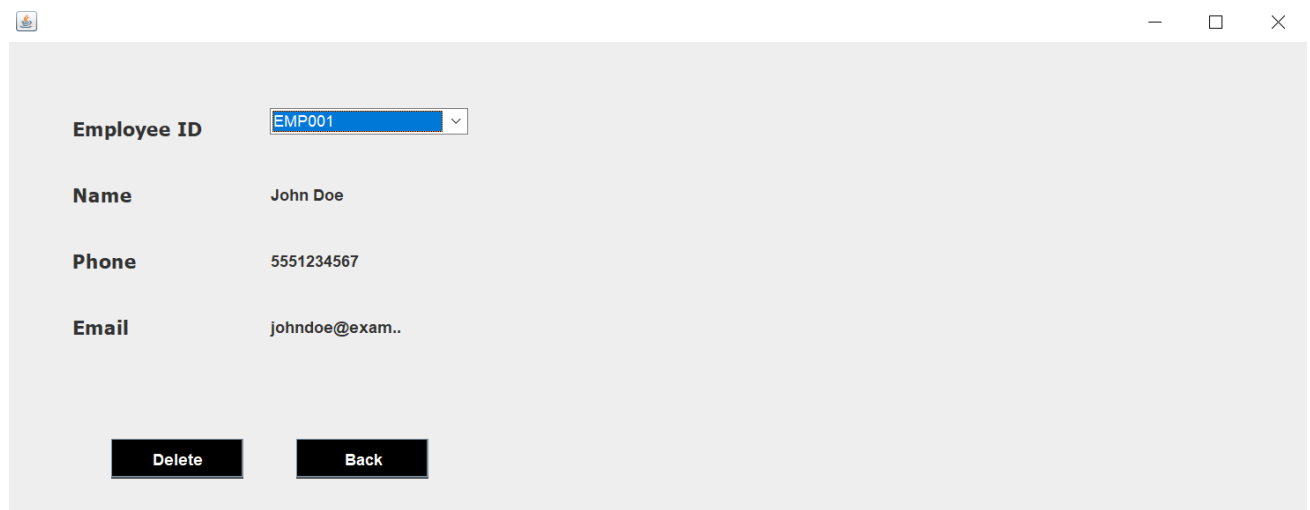
Figure 4.1.2 Main page



A screenshot of a web application window titled "Add Employee Detail". The window has a pink background. It contains several input fields for employee information, arranged in two columns. The fields are: Name, Date Of Birth (with a calendar icon), Address, Email, Designation, Employee ID (displayed as 703783 in red), Father's Name, Salary, Phone, Highest Education (a dropdown menu showing BBA), and Aadhar Number. At the bottom, there are two black buttons labeled "BACK" and "ADD".

Field	Value
Name	
Date Of Birth	
Address	
Email	
Designation	
Employee ID	703783
Father's Name	
Salary	
Phone	
Highest Education	BBA
Aadhar Number	


Figure 4.1.3 Add employee page



A screenshot of a web application window titled "Delete Selection". The window has a light gray background. It displays the details of an employee selected for deletion. The fields are: Employee ID (a dropdown menu showing EMP001), Name (John Doe), Phone (5551234567), and Email (johndoe@exam..). At the bottom, there are two black buttons labeled "Delete" and "Back".

Field	Value
Employee ID	EMP001
Name	John Doe
Phone	5551234567
Email	johndoe@exam..


Figure 4.1.4 Delete Selection


— □ ×

Search by employee id

name	fname	dob	salary	address	phone	email	education	designation	aadhar	empID
							BBA			489796
							BBA			64054
John Doe	Robert Doe	1990-05-15	50000	123 Elm Str...	5551234567	johndoe@e...	Bachelor's D...	Software En...	1234-5678-...	EMP001
John Doe	Robert Doe	1990-05-15	50000	123 Elm Str...	5551234567	johndoe@e...	Bachelor's D...	Software En...	1234-5678-...	EMP001
John Doe	Robert Doe	1990-05-15	50000	123 Elm Str...	5551234567	johndoe@e...	Bachelor's D...	Software En...	1234-5678-...	EMP001
sadhg	asd	8 Nov 2024	asd	asdaas	ads	asd	B.Tech	asd	asdasdasasd	172481

Figure 4.1.5 View Employee page


— □ ×

Add Employee Detail

Name
Date Of Birth
Address
Email
Designation
Employee ID

Father's Name
Salary
Phone
Higest Educat...
Aadhar Number

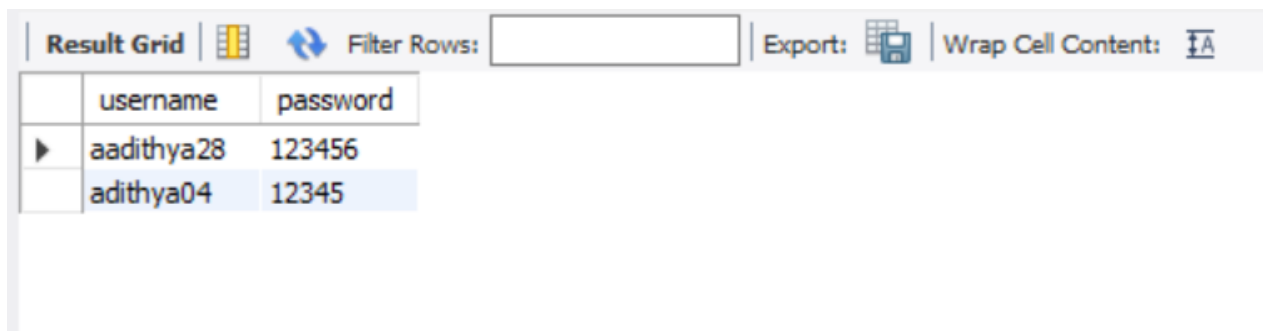
Figure 4.1.6 Update Employee

4.2 Database Design

The data in the system has to be stored and retrieved from database. Designing the database is part of system design. Data elements and data structures to be stored have been identified at analysis stage. They are structured and put together to design the data storage and retrieval system.

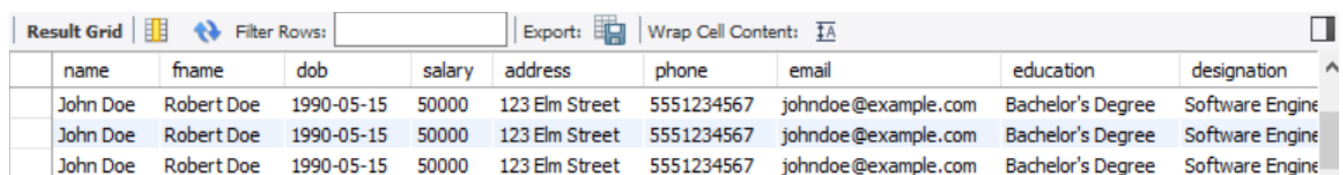
A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make database access easy, quick, inexpensive and flexible for the user. Relationships are established between the data items and unnecessary data items are removed. Normalization is done to get an internal consistency of data and to have minimum redundancy and maximum stability.

Employee Management system which contains 2 MySQL tables



	username	password
▶	aadithya28	123456
	adithya04	12345

Table 4.2.1 Login Table



	name	fname	dob	salary	address	phone	email	education	designation
	John Doe	Robert Doe	1990-05-15	50000	123 Elm Street	5551234567	johndoe@example.com	Bachelor's Degree	Software Engineer
	John Doe	Robert Doe	1990-05-15	50000	123 Elm Street	5551234567	johndoe@example.com	Bachelor's Degree	Software Engineer
	John Doe	Robert Doe	1990-05-15	50000	123 Elm Street	5551234567	johndoe@example.com	Bachelor's Degree	Software Engineer

Table 4.2.2 Employee Table

4.3 IMPLEMENTATIONS (CODE)

Splash.java:

```
package employee.management.system;
```

```
import javax.swing.*;
```

```
import java.awt.*;
```

```
public class Splash extends JFrame{
```

```
    Splash(){
```

```
        ImageIcon i1= new ImageIcon(ClassLoader.getResource("icons/front.png"));
```

```
        Image i2=i1.getImage().getScaledInstance(1170,620, Image.SCALE_DEFAULT);
```

```
        ImageIcon i3=new ImageIcon(i2);
```

```
        JLabel image = new JLabel(i3);
```

```
        image.setBounds(0,0,1170,620);
```

```
        add(image);
```

```
        setSize(1170,620);
```

```
        setLayout(null);
```

```
        setVisible(true);
```

```
        setLocation(50,50);
```

```
        try{
```

```
            Thread.sleep(2500);
```

```
            new Login();
```

```
            setVisible(false);
```

```
        }catch(Exception e){
```

```
            e.printStackTrace();
```

```
        }
```

```
    }
```

```

    public static void main(String[] args){
        new Splash();

    }
}

```

Login.java:

```

package employee.management.system;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.ResultSet;

```

```

public class Login extends JFrame implements ActionListener {

```

```

    JTextField tusername ;
    JPasswordField tpassword;
    JButton login,back;

```

```

    Login(){

```

```

        JLabel username = new JLabel("Username");
        username.setBounds(70,20,100,30);
        add(username);

```

```

        JLabel password = new JLabel("Password");
        password.setBounds(70,70,100,30);
        add(password);

```

```

        tusername = new JTextField();
        tusername.setBounds(150,20,150,30);

```

```

add(tusername);

tpassword = new JPasswordField();
tpassword.setBounds(150,70,150,30);
add(tpassword);

login = new JButton("LOGIN");
login.setBounds(150,140,150,30);
login.setBackground(Color.DARK_GRAY);
login.setForeground(Color.WHITE);
login.addActionListener(this);
add(login);

back = new JButton("BACK");
back.setBounds(150,180,150,30);
back.setBackground(Color.DARK_GRAY);
back.setForeground(Color.WHITE);
back.addActionListener(this);
add(back);

ImageIcon i1= new ImageIcon(ClassLoader.getResource("icons/login.png"));
Image i2=i1.getImage().getScaledInstance(600,300, Image.SCALE_DEFAULT);
ImageIcon i3=new ImageIcon(i2);
JLabel image = new JLabel(i3);
image.setBounds(0,0,600,300);
add(image);

setSize(600,280);
setLayout(null);
setVisible(true);
setLocation(300,170);

}

```

```

@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == login){
        try{
            String username = tusername.getText();
            String password = tpassword.getText();

            Conn conn=new Conn();

            String query = "Select * from login where username = '"+username+"' and
password = '"+password+"'";
            ResultSet resultSet = conn.statement.executeQuery(query);
            if (resultSet.next()) {
                setVisible(false);
                new Main_class();

            }else {
                JOptionPane.showMessageDialog(null,"Invalid Username or Password");
            }

        }catch(Exception e2){
            e2.printStackTrace();

        }

    } else if (e.getSource() == back) {
        System.exit(90);
    }
}

```

```

    public static void main(String[] args){
        new Login();
    }

}

Conn.java:

package employee.management.system;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class Conn {

    Connection connection;
    Statement statement;
    public Conn(){
        try{
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/employee_management","root","
123456");
            statement = connection.createStatement();

        }catch(Exception e ){
            e.printStackTrace();
        }
    }

}

```

```

view_employee.java:

package employee.management.system;


import net.proteanit.sql.DbUtils;


import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.ResultSet;


public class View_Employee extends JFrame implements ActionListener {


    JTable table;
    Choice choiceEMP;
    JButton searchbtn, print, update, back;
    View_Employee(){

        getContentPane().setBackground(new Color(116,123,182));
        JLabel search = new JLabel("Search by employee id");
        search.setBounds(20,20,150,20);
        add(search);


        choiceEMP = new Choice();
        choiceEMP.setBounds(180,20,150,20);
        add(choiceEMP);


        try{

            Conn c = new Conn();
            ResultSet resultSet = c.statement.executeQuery("select * from employee");
            while (resultSet.next()){
                choiceEMP.add(resultSet.getString("empId"));
            }
        }
    }
}

```



```

    }
} catch (Exception e){
    e.printStackTrace();
}

table = new JTable();
try{
    Conn c= new Conn();
    ResultSet resultSet = c.statement.executeQuery("select * from employee");
    table.setModel(DbUtils.resultSetToTableModel(resultSet));
} catch (Exception e){
    e.printStackTrace();
}

JScrollPane jp = new JScrollPane(table);
jp.setBounds(0,100,900,600);
add(jp);

searchbtn = new JButton("Search");
searchbtn.setBounds(20,70,80,20);
searchbtn.addActionListener(this);
add(searchbtn);

print = new JButton("Print");
print.setBounds(120,70,80,20);
print.addActionListener(this);
add(print);

update = new JButton("Update");
update.setBounds(220,70,80,20);
update.addActionListener(this);
add(update);

back = new JButton("Back");

```

```

back.setBounds(320,70,80,20);
back.addActionListener(this);
add(back);

```

```

setSize(900,600);
setLayout(null);
setLocation(150,50);
setVisible(true);
}

```

```

@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == searchbtn){
        String query = "select * from employee where empId =
"+choiceEMP.getSelectedItemAt()+"";
        try {
            Conn c = new Conn();
            ResultSet resultSet = c.statement.executeQuery(query);
            table.setModel(DbUtils.resultSetToTableModel(resultSet));
        } catch (Exception E){
            E.printStackTrace();
        }
    } else if (e.getSource() == print) {
        try {
            table.print();
        } catch (Exception E){
            E.printStackTrace();
        }
    } else if (e.getSource() == update){
        setVisible(false);
        new UpdateEmployee(choiceEMP.getSelectedItemAt());
    } else {
        setVisible(false);
    }
}

```

```

        new Main_class();
    }
}

public static void main(String[] args) {
    new View_Employee();
}
}

```

Main_class.java:

```

package employee.management.system;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Main_class extends JFrame {
    Main_class(){

        getContentPane().setBackground(new Color(116,123,182));

        JLabel heading = new JLabel("Employee Management System");
        heading.setBounds(40, 15, 700, 80);
        heading.setFont(new Font("Raleway", Font.BOLD, 45));
        heading.setForeground(Color.BLACK); // Set text color to black
        add(heading);

        JButton add = new JButton("Add Employee");
        add.setBounds(650,200,200,50);
        add.setForeground(Color.WHITE);
    }
}

```

```

add.setBackground(Color.BLACK);
add.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        new AddEmployee();
        setVisible(false);
    }
});
add(add);

```

```

JButton remove = new JButton("Remove Employee");
remove.setBounds(650,260,200,50);
remove.setForeground(Color.WHITE);
remove.setBackground(Color.BLACK);
remove.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        new RemoveEmployee();
        setVisible(false);
    }
});
add(remove);

```

```

JButton view = new JButton("View Employee");
view.setBounds(650,320,200,50);
view.setForeground(Color.WHITE);
view.setBackground(Color.BLACK);
view.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        new View_Employee();
        setVisible(false);
    }
});
add(view);

```

```

    }
});
add(view);

JButton exit = new JButton("Exit");
exit.setBounds(1000,10,100,30);
exit.setForeground(Color.WHITE);
exit.setBackground(Color.BLACK);
exit.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        JOptionPane.showMessageDialog(null,"Logged Out Successfully..!");
        System.exit(90);

    }
});
add(exit);

setSize(1170,620);
setLocation(50,50);
setLayout(null);
setVisible(true);
}

public static void main(String[] args){
    new Main_class();
}
}

```

5.CONCLUSION

The Movie Reservation System project, executed under experienced guidance, embodies a meticulous approach to design and implementation. With a focus on user-friendly functionalities like adding/viewing movies and reservation management, the system ensures a seamless movie-going experience. Robust security measures, particularly in the "Remove Admin" module, highlight a commitment to data integrity and system protection. The project stands as a well-rounded solution, meeting current requirements while allowing for future adaptability and enhancements.

6. REFERENCES

- [1] Reference link for java-awt at javatpoint : <https://www.javatpoint.com/java-awt>
- [2] Reference link for java-swing at javatpoint : <https://www.javatpoint.com/java-swing>