

RAJALAKSHMI ENGINEERING COLLEGE

[AUTONOMOUS]

RAJALAKSHMI NAGAR, THANDALAM – 602 105



RAJALAKSHMI
ENGINEERING COLLEGE

CS23332 DATABASE MANAGEMENT SYSTEMS
Laboratory Record Note Book

EMPLOYEE REGISTRATION

A MINI-PROJECT BY:

Atchaya S (230701045) & Akshaya Sri S (230701024)

*in partial fulfillment of the award of the degree OF BACHELOR
OF ENGINEERING IN
COMPUTER SCIENCE AND ENGINEERING.*

**RAJALAKSHMI ENGINEERING COLLEGE
[AUTONOMOUS]**

RAJALAKSHMI NAGAR, THANDALAM – 602 105

BONAFIDE CERTIFICATE

Academic Year : 2024-2025. Semester: THIRD sem Branch : CSE

Register No.

230701045 & 230701024

**Certified that this is the bonafide record of work done by
ATCHAYA S(230701045) and AKSHAYA SRI S(230701024) the above
student in the CS23332 – Database Management Systems during
the year 2024-2025.**

Signature of Faculty in-charge

Submitted for the Practical Examination held on

Internal Examiner

External Examiner

ABSTRACT

This project on Employee Management System is primarily concerned with management of employee records and performing a multitude of operations like adding, updating, searching and deleting employee details. The objective of the project is to design a software solution that is cost-effective and easy to use with the possibilities of automating most of the activities usually performed in employee management carnation, managing employee information in bulks, making reports out of that information and most importantly making sure proper data is inputted and edited. The system will keep up with the user by processing employee record logically to the extent that the user can view as well as edit and control such information as the employee's name, salary, and mobile phone number with ease. MySQL DBMS is used for the purpose of data storage and retrieval. The system we have developed is efficient in managing employee records and supports the following functionalities:

Adding New Employee: The functionality permits users to place employee records for a new employee into the system.

Updating Employee Data: Users are able to update the particulars of other employees by providing the respective employee id.

Searching Employee: The details of an employee are fetched from the database using their unique assigned id.

Deleting Employee: An employee's record can be erased from the entire system by authorized users with the belief they will not use the id for any other purpose than that it has been intended for.

Languages Used: Java Core (for programming the functionality)

Concept Used: Swing (for creating the graphical user interface)

IDE Used: IntelliJ IDEA

Database Used: MySQL (for storing employee records and managing data)

CONTENTS

CHAPTERS	PAGE NO
Chapter 1	Introduction
	1.1 Problem Definition
	1.2 Need
Chapter 2	Requirements
	2.1 Software Requirement Specifications
	2.2 Hardware Requirement Specifications
Chapter 3	Entity Relationship Diagram
	3.1 Entity relationship diagram
Chapter 4	Schema Diagram
	4.1 Schema diagram
Chapter 5	Implementation
	5.1 Frontend Implementation
	5.2 Backend Implementation
	5.3 Frontend and Backend Interactions
Chapter 6	Snapshots
	Conclusion
	14-20
	21

CHAPTER 1

INTRODUCTION

The Employee Management System is a very sophisticated software tool built to support core activities related to employee management. There are many tasks such as maintaining employee records or handling their salary, monitoring their performances, and processing payroll, which become even more difficult with the growth of an organization. Performing these processes manually can be challenging as they present room for errors, inefficiencies, and delays that eventually lower productivity as well as employee morale. The system solves the problem by integrating important tasks in one system. It provides HR personnel and managers access to information on employees at any given time to ensure data is kept up to date while errors are kept at minimum levels thanks to the automated data entry and its validation. The Employee Management System helps not only the reduction of the operational costs and speeds up processes within the organization but also provides useful information regarding

employees' efficiency and their wages for the benefit of coming up with informed decisions and growth plans. By replacing manual and paper-based structure with computerized systems in operation, the system enables organizations with its modern, lean structure, to provide better customer service and even decrease administrative efforts which in turn increases productivity making the system a very important resource in management of twenty first century human resources.

1.1 Problem Definition:

The Employee Management System addresses the problems faced in handling employee records and payroll in a hard copy format. Manual systems are prone to mistakes and inefficiency in the management of employee records and their remuneration. The system employs a mySql data for safe and structured storage of information, a Java Core frontend (Swing) ensures the two are in sync at any given time. Any modifications or edits done in the frontend are immediately reflected in the backend, thus improving the management of employees data..

1.2 Need

Any organization has to manage employee records efficiently so as to minimize mistakes and inefficiencies. This system will allow management of employee's data, performance as well as payroll in real time and in order to provide accurate information with little or no administration work. The performance of the Employee Management System will optimize data and record handling, ensuring that the HR personnel spend less time and energy in maintaining employee records, thus enhancing productivity in the organization. In most cases, making manual changes into employee records is very long and full of errors. Rather, the system would update the HR system with details of the employee's status, eliminating the need for HR staff to maintain filing systems therefore they would have more time to focus on producing results. This system will generate management information system (MIS) reports for performance evaluation and payroll data, which supports management actions and improves human resource planning.

The main advantages include:

Statistics and analytics of employee record-keeping, performance, and payroll on HR management are available immediately which helps in making informed decisions concerning HR operations.

Data is processed at **higher speed levels**, meaning that all updates and information requests are done fast and with precision.

There is **reliability** on the employee details and less chances of human error on the records and payroll systems.

The system is **dependable** in that it will guarantee that the information will be maintained and will always be current.

This system comes with a **flexible approach** suitable for the growth of the organization in a way that it can accommodate a growing workforce and increasing HR requirements.

CHAPTER 2 REQUIREMENTS 2.1 Software Requirement Specifications

- **Operating System:** Windows 10
- **Frontend Software:** Java (IntelliJ IDEA, JDK 8)
- **Backend Software:** MySQL
- **Server:** Localhost (for development purposes)
- **Documentation:** System Documentation (user manual, technical documentation)

2.2 Hardware Requirement Specifications

- **Computer Processor:** Core i5
- **Processor Speed:** 2.3 GHz
- **Hard Disk:** 400 GB or more
- **RAM:** Minimum 4GB

CHAPTER 3 ENTITY RELATIONSHIP DIAGRAM

An Entity-Relationship (ER) diagram is a graphic depiction of the various relationships that exist in a database with the use of boxes for entities, diamonds for relationships, and ovals for attributes. For instance, the ER diagram in the Employee Management System illustrates data items such as employee information (name, salary, mobile number) and how they are related to each other (e.g. employees and their payroll information). The purpose of the diagram is to facilitate the design of a conceptual data model which is agnostic of any particular database management system by providing an image of how data will flow in the system and more importantly how every bit of data will be paired and connected in due course.

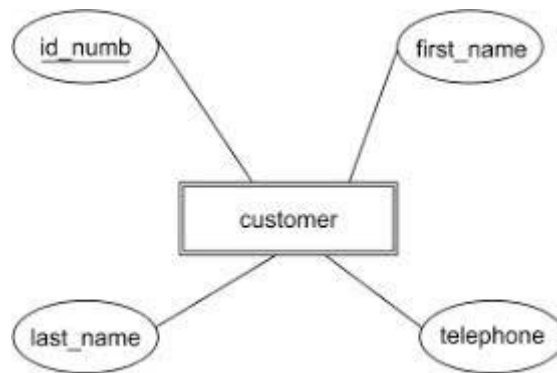


Fig 3.1 Entity relationship diagram

CHAPTER 4 SCHEMA DIAGRAM

4.1 Schema Diagram

The skeletal structure which shows the logical representation of the whole database is termed as a database schema. Equally, a database schema outlines its components and how they relate to each other. It provides an explanatory overview of the database which can be explained using schema diagrams. It describes the structure and interrelation of the data stored within the database. Anaformalizes every stipulation that must be observed concerning that data.

As mentioned above, a database schema outlines its components and how they relate to each other. Also, it contains a descriptive part of the database which can be represented through schema

diagrams. The tools for statistics and turning data teachable are the schema designers dealing with purposes of programmer comprehension of the database and supposing its usefulness».

A database schema can broadly be classified into two types:

Physical Database Schema: This schema deals with the actual data storage. It defines how the data is stored in the secondary storage such as files and indices. It is concerned with the physical implementation of the concepts like the way the data is organized and the techniques used to store them.

Logical Database Schema: This schema carries the logical aspects of the database, and its tables, views, and the relationships that govern them to ensure data integrity. For example, in the case of an Employee Management System, it would consist of tables for the employees, their wages data and how these two entities are related including constraints to ensure the integrity of the data.

CHAPTER 5

IMPLEMENTATION

5.1 Frontend (GUI) Implementation

Main Frame Setup:

```
public static void main(String[] args) {  
  
    JFrame frame = new JFrame("Employee");  
  
    frame.setContentPane(new Employee().Main);  
  
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)  
  
    ; frame.pack(); frame.setVisible(true);  
  
}
```

- **Description:** Initializes the main frame of the application, setting up the content pane (`Main JPanel`), and displays the GUI.

Frontend Input Fields:

- **Text Fields:**
 - `txtName`: Employee's name.
 - `txtMobile`: Employee's mobile number.
 - `txtSalary`: Employee's salary.
 - `txtid`: Employee ID for search, update, or delete.
- **Buttons:**
 - `saveButton`: To save a new employee record.
 - `updateButton`: To update an existing employee record.

- **deleteButton**: To delete an employee record.
- **searchButton**: To search an employee record by ID.
- **JTable**:
 - **table1**: Displays employee data in a table format.

Event Listeners (Button Actions):

- **saveButton**: Adds a new record to the database when clicked.
- **updateButton**: Updates an existing record in the database when clicked.
- **deleteButton**: Deletes an employee record from the database when clicked.
- **searchButton**: Searches for an employee record based on the employee ID entered in the **txtid** field.

5.2 Backend (Database Operations) Implementation

1.Database Connection:

```
public void connect() {
    try {
        con =
        DriverManager.getConnection("jdbc:mysql://localhost:3306/rdcompany?useSSL=false&serverTimezone=UTC", "root", "");

        System.out.println("Connection Successful");
    } catch (SQLException ex) {
        System.err.println("SQL Error: " + ex.getMessage());
        ex.printStackTrace();
    }
}
```

Description: Establishes a connection to the MySQL database (rdcompany) using JDBC.

2.Table Data Loading (Fetch and Display): void table_load()

```
{ try { pst = con.prepareStatement("select * from
employee"); ResultSet rs = pst.executeQuery();
table1.setModel(DbUtils.resultSetToTableModel(rs));
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

```
}
```

Description: Fetches all employee records from the `employee` table in the database and populates the `JTable` with the data.

5.3 Frontend and Backend Interactions (Event Handlers and CRUD Operations)

1. Save New Employee Record (Create):

```
saveButton.addActionListener(new ActionListener() {  
  
    @Override public void  
  
    actionPerformed(ActionEvent e) { String  
  
    empname, salary, mobile;  
  
  
  
    empname = txtName.getText();  
  
    salary = txtSalary.getText();  
  
    mobile = txtMobile.getText();  
  
  
  
    try { pst = con.prepareStatement("insert into employee(empname,salary,mobile)  
        values(?,?,?)"); pst.setString(1, empname); pst.setString(2, salary); pst.setString(3, mobile);  
  
        pst.executeUpdate();  
  
        JOptionPane.showMessageDialog(null, "Record  
        Added"); table_load(); txtName.setText("");  
        txtSalary.setText(""); txtMobile.setText("");  
        txtName.requestFocus();  
    } catch (SQLException e1) {  
        e1.printStackTrace();  
    }  
    }  
});
```

Description: This action listens for the "Save" button click. It collects the data entered in the input fields and inserts it into the `employee` table.

2.Search Employee by ID (Read):

```
searchButton.addActionListener(new ActionListener() {
    @Override public void

    actionPerformed(ActionEvent e) { try {

        String empid = txtid.getText();

        pst = con.prepareStatement("select empname, salary, mobile from employee where
        id=?"); pst.setString(1, empid); ResultSet rs = pst.executeQuery();

        if (rs.next() == true) {

            String empname = rs.getString(1);

            String emsalary = rs.getString(2);

            String emmobile = rs.getString(3);

            txtName.setText(empname);

            txtSalary.setText(emsalary);

            txtMobile.setText(emmobile);

        } else {

            txtName.setText("");

            txtSalary.setText("");

            txtMobile.setText("");

            JOptionPane.showMess
            ageDialog(null, "Invalid
            Employee No");

        }
    }
}
```

```

    } catch (SQLException ex) {

        ex.printStackTrace();

    }

}

});

```

Description: This function allows searching for an employee by their ID and populating the text fields with their data.

3.Update Employee Record (Update):

```

updateButton.addActionListener(new ActionListener() {

    @Override public void

    actionPerformed(ActionEvent e) { String

    empid, empname, salary, mobile;

    empname = txtName.getText();

    salary = txtSalary.getText();

    mobile = txtMobile.getText();

    empid = txtid.getText();

    try {

        pst = con.prepareStatement("update employee set empname=?, salary=?, mobile=? where
id=?"); pst.setString(1,

        empname); pst.setString(2,

        salary); pst.setString(3,

        mobile); pst.setString(4,

        empid);

        pst.executeUpdate();
    }
}
});

```

```

JOptionPane.showMessageDialog(null, "Record
Updated"); table_load(); txtName.setText("");
txtSalary.setText(""); txtMobile.setText("");
txtName.requestFocus();
} catch (SQLException e1) {
    e1.printStackTrace();
}
}
});

```

Description: This function listens for the "Update" button click event, updates the employee record in the database based on the entered ID, and reflects the changes in the UI.

4.Delete Employee Record (Delete):

```

deleteButton.addActionListener(new ActionListener() {
    @Override public void
    actionPerformed(ActionEvent e) { String
    empid; empid = txtid.getText();

    try { pst = con.prepareStatement("delete from employee where
    id=?"); pst.setString(1, empid); pst.executeUpdate();
    JOptionPane.showMessageDialog(null, "Record Deleted");
    table_load();
    txtName.setText("");
    txtSalary.setText("");
    txtMobile.setText("");
    txtName.requestFocus();

```

```
        } catch (SQLException e1) {  
            e1.printStackTrace();  
        }  
    }  
}  
});
```

Description: This function listens for the "Delete" button click event, deletes the employee record by ID, and updates the displayed data in the JTable.

Summary of Classifications

1. **Frontend:**
 - **GUI Setup:** Initializes and displays the GUI.
 - **Input Fields:** For user interaction and data entry.
 - **Buttons:** For triggering actions like saving, updating, deleting, and searching.
 - **JTable:** Displays employee records.
2. **Backend:**
 - **Database Connection:** Establishes connection to the MySQL database.
 - **Table Data Handling:** Loads and updates the employee data.
 - **CRUD Operations:** Create, Read, Update, and Delete operations are performed on the employee data.
3. **Frontend and Backend Interactions:**
 - **Save, Search, Update, and Delete** employee records with appropriate database queries and table updates.

CHAPTER 6

SNAPSHOTS

Employee

k

Employee Registration

Employee Name

Salary

Mobile

Save

Update

Delete

Search

Id	Empname	Salary	Mobile
----	---------	--------	--------

Fig 6.1 Overview

Employee

k

Employee Registration

Employee Name

Tom Holland

Salary

78000

Mobile

987655504

Save

Update

Delete

Search

Id	Empname	Salary	Mobile
1	Alice	50000	987654322
2	John Doe	45000	976543210
3	Jane Smith	52000	876543210
4	Michael Johnson	60000	900076548
5	Emily Davis	47500	778899669
6	Daniel Brown	55000	897856453
7	Sophia Wilson	62000	987765434
8	James Taylor	50000	987655542
9	Tom Holland	70000	982254321
10		80000	876665551
11		72000	899700066

Message

i

Record Added

OK

Fig 6.2 Record Insertion

Employee

k

Employee Registration

Employee Name

Daniel Brown

Salary

55000

Mobile

897856453

Save

Update

Delete

Search

Id	Empname	Salary	Mobile
1	Alice	50000	987654322
2	John Doe	45000	976543210
3	Jane Smith	52000	876543210
4	Michael Johnson	60000	900076548
5	Emily Davis	47500	778899669
6	Daniel Brown	55000	897856453
7	Sophia Wilson	62000	987765434
8	James Taylor	50000	987655542
9	Olivia Martinez	70000	982254321
10	David Lee	80000	876665551
11	Isabella Clark	72000	899700066
12	Tom Holland	78000	987655504

6

Fig 6.3 Search Records

Employee

Employee Registration

Id	Empname	Salary	Mobile
1	Alice	50000	987654322
2	John Doe	45000	976543210
3	Jane Smith	52000	876543210
4	Michael Johnson	60000	900076548
5	Emily Davis	47500	778899669
6	Daniel Brown	55000	897856453
7	Sophia Wilson	62000	987765434
8	James Taylor	50000	987655542
		70000	982254321
		80000	876665551
		72000	899700066
		78000	987655504

Employee Name

Salary

Mobile

Message


 **Record Updated**

Fig 6.4 Record Updation

Employee

k

Employee Registration

Employee Name

David Lee

Salary

80000

Mobile

876665551

Save

Update

Delete

Search

10

Id	Empname	Salary	Mobile
1	Alice	50000	987654322
2	John Doe	45000	976543210
3	Jane Smith	52000	876543210
4	Michael Johnson	60000	900076548
5	Emily Davis	90000	778899669
6	Daniel Brown	55000	897856453
7	Sophia Wilson	62000	987765434
8	James Taylor	50000	987655542
		70000	982254321
		80000	876665551
		72000	899700066
		78000	987655504

Message

i

Record Deleted

OK

Fig 6.5 Record Deletion

Employee

—

□

×

k

Employee Registration

Employee Name

Salary

Mobile

Save

Update

Delete

Search

Id	Empname	Salary	Mobile
1	Alice	50000	987654322
2	John Doe	45000	976543210
3	Jane Smith	52000	876543210
4	Michael Johnson	60000	900076548
5	Emily Davis	90000	778899669
6	Daniel Brown	55000	897856453
7	Sophia Wilson	62000	987765434
8	James Taylor	50000	987655542
9	Olivia Martinez	70000	982254321
11	Isabella Clark	72000	899700066
12	Tom Holland	78000	987655504

Fig 6.6 Overall Structure

REFERENCES

1. <https://www.javatpoint.com/java-tutorial>
2. <https://www.wikipedia.org/>
3. <https://www.w3schools.com/sql/>
4. SQL | Codecademy