

# Employee Management System Documentation

## Table of Contents

1. **Project Overview**
  2. **Technology Stack**
  3. **System Architecture**
  4. **Features**
  5. **Pages**
    - Home Page
    - View Page
    - Onboard Page
    - Update Page
  6. **Code Snippets and Explanations**
    - Home Page
    - View Page
    - Onboard Page
    - Update Page
  7. **Screenshots**
  8. **Conclusion**
- 

## 1. Project Overview

The Employee Management System is a desktop application developed using **Java (Swing API, JDBC)** and **MySQL**. The system enables efficient management of employee details, including adding, viewing, editing, and deleting records. This project was developed as a task assigned by **Punit Sir** at **KodNest**.

---

## 2. Technology Stack

- **Programming Language:** Java
  - **Frameworks/Libraries:** Swing API, JDBC
  - **Database:** MySQL
  - **IDE:** IntelliJ IDEA/Eclipse/NetBeans (whichever you used)
- 

## 3. System Architecture

- **Frontend:** Java Swing for the graphical user interface (GUI).
  - **Backend:** JDBC for database connectivity and query execution.
  - **Database:** MySQL for storing and managing employee data.
-

## 4. Features

- **Add Employee:** Add details like name, department, salary, etc.
  - **View Employees:** Display all employee details in a tabular format.
  - **Edit Employee:** Update existing employee records.
  - **Delete Employee:** Remove employee records from the database.
  - **Profile View:** View detailed information about a specific employee.
- 

## 5. Pages

### Home Page

**Description:** The home page serves as the main dashboard, allowing users to navigate to different functionalities like adding, viewing, editing, or deleting employee details.

### View Page

**Description:** This page displays all employees in a tabular format, retrieving data dynamically from the MySQL database.

### Onboard Page

**Description:** This page is used for onboarding new employees by entering their details and saving them in the database.

### Update Page

**Description:** The update page allows users to modify existing employee records by selecting an employee and editing their details.

---

## 6. Code Snippets and Explanations

### Home Page

```
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.FlowLayout;
import javax.swing.JPanel;
import javax.swing.JScrollPane;

import java.awt.Color;
import java.awt.Component;

import javax.swing.JLabel;
import javax.swing.JOptionPane;

import java.awt.Font;
import java.util.ArrayList;

import javax.swing.SwingConstants;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JTable;
import javax.swing.border.LineBorder;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableCellEditor;
import javax.swing.table.TableCellRenderer;

import Backend.EmployeeApp;

import javax.swing.ListSelectionModel;
import javax.swing.AbstractCellEditor;
import javax.swing.DefaultCellEditor;
import javax.swing.ImageIcon;
import javax.swing.event.ActionListener;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import javax.swing.event.ActionEvent;

public class HomePage {

    protected static JFrame frame;
    private JTextField searchTxt;
    private JTable table;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    HomePage window = new HomePage();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the application.
     */
    public HomePage() {
        initialize();
    }

    /**
     * Initialize the contents of the frame.
     */
    private void initialize() {
        frame = new JFrame();
        frame.getContentPane().setForeground(new Color(0, 0, 0));
        frame.getContentPane().setBackground(new Color(255, 255, 255));
        frame.setBounds(100, 100, 1197, 706);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(null);

        JLabel lblNewLabel = new JLabel("EMPLOYEE MANAGEMENT SYSTEM");
        lblNewLabel.setHorizontalAlignment(SwingConstants.CENTER);
        lblNewLabel.setFont(new Font("MS UI Gothic", Font.BOLD, 40));
        lblNewLabel.setForeground(new Color(0, 0, 255));
        lblNewLabel.setBounds(266, 10, 669, 36);
        frame.getContentPane().add(lblNewLabel);

        // SEARCH FUNCTIONALITY
        searchTxt = new JTextField();
        searchTxt.setFont(new Font("Arial", Font.BOLD, 16));
        searchTxt.setText("SEARCH");
        searchTxt.setBounds(41, 101, 238, 36);
        frame.getContentPane().add(searchTxt);
        searchTxt.setColumns(1);

        JButton searchBtn = new JButton("🔍");
        searchBtn.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if (searchTxt.getText().isEmpty() || !searchTxt.getText().matches("[0-9]*")) {
                    new MessageBox().showMessage("Enter a Valid ID");
                    return;
                }
                int id = Integer.parseInt(searchTxt.getText());

                EmployeeApp emp = new EmployeeApp();
                ResultSet res = emp.getEmployee(id);

                tableStructure(res);
            }
        });
        searchBtn.setBounds(283, 101, 51, 38);
        frame.getContentPane().add(searchBtn);

        JButton addBtn = new JButton("NEW +");
        addBtn.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                frame.setVisible(false);
                new OnboardPage(frame);
            }
        });
        addBtn.setFont(new Font("Tahoma", Font.BOLD, 19));
        addBtn.setBackground(new Color(255, 255, 255));
        addBtn.setForeground(new Color(0, 0, 255));
        addBtn.setBounds(1013, 101, 135, 57);
        frame.getContentPane().add(addBtn);
    }
}
```

```

// Create the JTable with data and column names
table = new JTable();
table.setFillViewportHeight(true);
table.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);

EmployeeApp emp = new EmployeeApp();
ResultSet res = emp.getEmployees();
tableStructure(res);

// Styling
table.setBorder(new LineBorder(new Color(0, 0, 0), 1, true));
table.setFont(new Font("Tahoma", Font.PLAIN, 19));
table.setForeground(Color.BLACK);
table.setBackground(Color.WHITE);
table.setRowHeight(40);

// Table header styling
table.getTableHeader().setOpaque(false);
table.getTableHeader().setBackground(new Color(32, 136, 203));
table.getTableHeader().setForeground(Color.WHITE);
table.getTableHeader().setFont(new Font("Castellar", Font.BOLD, 24));
// table.getTableHeader().

// Wrap the JTable in a JScrollPane
JScrollPane scrollPane = new JScrollPane(table);
scrollPane.setBounds(10, 164, 1163, 360);

// Add the JScrollPane to the frame, not the JTable directly
frame.getContentPane().add(scrollPane);

// REFRESH FUNCTIONALITY
JButton refreshBtn = new JButton("");
refreshBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        EmployeeApp emp = new EmployeeApp();
        ResultSet res = emp.getEmployees();
        tableStructure(res);
    }
});

refreshBtn.setBackground(new Color(255, 255, 255));
refreshBtn.setIcon(new ImageIcon(HomePage.class.getResource("/images/icons8-refresh-30.png")));
refreshBtn.setBounds(918, 110, 65, 43);
frame.getContentPane().add(refreshBtn);

JLabel lblNewLabel_1 = new JLabel("@KODNEST | ALL RIGHTS RESERVED");
lblNewLabel_1.setHorizontalAlignment(SwingConstants.CENTER);
lblNewLabel_1.setFont(new Font("Tw Cen MT Condensed Extra Bold", Font.PLAIN, 14));
lblNewLabel_1.setBounds(409, 646, 381, 13);
frame.getContentPane().add(lblNewLabel_1);

JLabel lblNewLabel_2 = new JLabel("EMPLOYEES");
lblNewLabel_2.setFont(new Font("Arial", Font.PLAIN, 16));
lblNewLabel_2.setHorizontalAlignment(SwingConstants.CENTER);
lblNewLabel_2.setBounds(569, 145, 146, 13);
frame.getContentPane().add(lblNewLabel_2);
}

void tableStructure(ResultSet res) {
    // Create Table Model
    DefaultTableModel model = new DefaultTableModel() {
        @Override
        public boolean isCellEditable(int row, int column) {
            // Only make the "Actions" column editable
            return column == getColumnCount() - 1;
        }
    };

    try {
        if (res != null) {
            ResultSetMetaData metaData = res.getMetaData();

            // Prepare data
            int columnCount = metaData.getColumnCount();
            for (int i = 1; i <= columnCount; i++) {
                model.addColumn(metaData.getColumnName(i));
            }
            model.addColumn("Actions"); // Add the Actions column

            while (res.next()) {
                Object[] row = new Object[columnCount + 1];
                for (int i = 1; i <= columnCount; i++) {
                    row[i - 1] = res.getObject(i);
                }
                row[columnCount] = "Actions"; // Placeholder for the Actions column
                model.addRow(row);
            }

            // Set model to JTable
            table.setModel(model);

            // Add custom renderer and editor for the Actions column
            table.getColumnModel().getColumn(model.getColumnCount() - 1).setCellRenderer(new
            ActionRenderer());
            table.getColumnModel().getColumn(model.getColumnCount() - 1).setCellEditor(new
            ActionEditor(table));

            // Center renderer
            DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
            centerRenderer.setHorizontalAlignment(SwingConstants.CENTER);
            for (int i = 0; i < table.getColumnCount(); i++) {
                table.getColumnModel().getColumn(i).setCellRenderer(centerRenderer);
            }
        }
    } catch (SQLException e1) {
        e1.printStackTrace();
    }
}
}

```

```

// Custom Renderer for the Actions column
static class ActionRenderer extends JPanel implements TableCellRenderer {

    private static final long serialVersionUID = 1L;

    public ActionRenderer() {
        setLayout(new FlowLayout(FlowLayout.CENTER, 5, 0));
    }

    @Override
    public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected,
                                                    boolean hasFocus, int row, int column) {

        this.removeAll(); // Clear previous buttons
        add(createButton(row, "/images/icons8-view-24.png"));
        add(createButton(row, "/images/icons8-edit-30.png"));
        add(createButton(row, "/images/icons8-delete-24.png"));
        return this;
    }

    private JButton createButton(int row, String path) {
        JButton button = new JButton("");
        button.setFont(new Font("Tahoma", Font.PLAIN, 14));
        button.setFocusable(false);
        button.setBackground(new Color(255, 255, 255));
        button.setBounds(885, 563, 57, 28);
        button.setIcon(new ImageIcon(HomePage.class.getResource(path)));
        return button;
    }
}

// Custom Editor for the Actions column
static class ActionEditor extends AbstractCellEditor implements TableCellEditor {
    private JPanel panel;
    private JButton viewButton, editButton, deleteButton;
    private JTable table;

    public ActionEditor(JTable table) {
        this.table = table;
        panel = new JPanel(new FlowLayout(FlowLayout.CENTER, 5, 0));

        viewButton = new JButton("");
        editButton = new JButton("");
        deleteButton = new JButton("");

        viewButton.setFont(new Font("Tahoma", Font.PLAIN, 14));
        viewButton.setFocusable(false);
        viewButton.setBackground(new Color(255, 255, 255));
        viewButton.setBounds(885, 563, 57, 28);
        viewButton.setIcon(new ImageIcon(HomePage.class.getResource("/images/icons8-view-24.png")));

        editButton.setFont(new Font("Tahoma", Font.PLAIN, 14));
        editButton.setFocusable(false);
        editButton.setBackground(new Color(255, 255, 255));
        editButton.setBounds(885, 563, 57, 28);
        editButton.setIcon(new ImageIcon(HomePage.class.getResource("/images/icons8-edit-30.png")));

        deleteButton.setFont(new Font("Tahoma", Font.PLAIN, 14));
        deleteButton.setFocusable(false);
        deleteButton.setBackground(new Color(255, 255, 255));
        deleteButton.setBounds(885, 563, 57, 28);
        deleteButton.setIcon(new ImageIcon(HomePage.class.getResource("/images/icons8-delete-
24.png")));

        // Add Action Listeners for buttons
        viewButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int row = table.getSelectedRow(); // Get the selected row
                if (row != -1) {
                    // Extract data from the row
                    Object idObject = table.getValueAt(row, 0); // Assuming the first column is
                    "id"

                    int id = Integer.parseInt(idObject.toString());
                    frame.setVisible(false);
                    new ViewPage(id, frame);

                    stopCellEditing();
                }
            }
        });
}

```



```

editButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        int row = table.getSelectedRow(); // Get the selected row
        if (row != -1) {
            // Extract data from the row
            Object idObject = table.getValueAt(row, 0); // Assuming the first column is "id"
            Object nameObject = table.getValueAt(row, 1);
            Object deptObject = table.getValueAt(row, 2);
            Object salObject = table.getValueAt(row, 3);
            int id = Integer.parseInt(idObject.toString());
            String name = nameObject.toString();
            String dept = deptObject.toString();
            String sal = salObject.toString();

            frame.setVisible(false);
            new UpdatePage(id, name, dept, sal, frame);

            stopCellEditing();
        }
    }
});

deleteButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        int row = table.getSelectedRow(); // Get the selected row
        if (row != -1) {
            // Extract data from the row
            Object idObject = table.getValueAt(row, 0); // Assuming the first column is "id"
            int id = Integer.parseInt(idObject.toString());

            MessageBox deleteBox = new MessageBox();
            deleteBox.delete(id);

            stopCellEditing();
        }
    }
});

panel.add(viewButton);
panel.add(editButton);
panel.add(deleteButton);
}

@Override
public Component getTableCellEditorComponent(JTable table, Object value, boolean isSelected, int
row, int column) {
    return panel;
}

@Override
public Object getCellEditorValue() {
    return null; // No specific value for the Actions column
}
}
}

```

## Explanation:

The `HomePage` class represents the graphical user interface (GUI) for managing employees in the Employee Management System. It utilizes the Swing library to create a window with features like employee search, displaying employee data in a table, and adding, updating, and deleting employee records.

## Key Components:

1. **JFrame frame:** The main window of the application.
2. **TextField searchTxt:** A text field where users can input an employee ID to search.
3. **JTable table:** A table used to display employee records retrieved from the database.

---

## UI Elements and Layout:

- **Labels and Titles:**
  - `EMPLOYEE MANAGEMENT SYSTEM`: A large title at the top of the window, centrally aligned.
  - `@KODNEST | ALL RIGHTS RESERVED`: A footer message at the bottom.
  - `EMPLOYEES`: A label indicating the section where the employees will be listed.
- **Search Functionality:**
  - A `TextField` (`searchTxt`) for entering an employee ID and a `Button` (`searchBtn`) with a search icon to initiate the search.
  - The user inputs an ID, and upon clicking the search button, the program retrieves data for the corresponding employee from the database and displays it in the table.
- **Add Employee:**
  - A `Button` (`addBtn`) with a "NEW +" label that directs the user to a new page for adding an employee. When clicked, it hides the current frame and opens the `OnboardPage` for adding a new employee.
- **Employee Table:**
  - A `JTable` is used to display employee information like ID, name, department, and salary, with an additional "Actions" column.
  - The table is wrapped inside a `JScrollPane` to allow scrolling when there are many records.
- **Refresh Functionality:**
  - A refresh button (`refreshBtn`) with an icon that reloads the employee data and updates the table view.

---

## Table Structure & Behavior:

1. **Populating the Table:**
    - The `tableStructure(ResultSet res)` method is used to populate the table based on data from the `ResultSet` object returned by the `EmployeeApp` class.
    - It uses the `DefaultTableModel` to create the table structure dynamically and adds an "Actions" column.
    - The table displays records retrieved from the database, and each row includes buttons for "View", "Edit", and "Delete" operations.
  2. **Actions Column:**
    - The "Actions" column is populated with three buttons:
      - **View**: Opens a detailed view of the selected employee's data.
      - **Edit**: Allows the user to edit the selected employee's data.
      - **Delete**: Deletes the selected employee from the database.
    - **ActionRenderer**: A custom renderer class that creates and displays the action buttons in each row.
    - **ActionEditor**: A custom editor class that provides the functionality for the action buttons. When clicked, these buttons trigger the corresponding actions like viewing, editing, or deleting the record.
  3. **Centering Table Data:**
    - The `DefaultTableCellRenderer` is used to center-align all data in the table.
-

## Search Functionality:

- The `searchBtn` triggers an action that checks if the entered text is a valid employee ID. If valid, it retrieves the employee data from the database using the `getEmployee(id)` method from the `EmployeeApp` class and updates the table.
  - If the input is invalid (empty or non-numeric), it displays a message prompting the user to enter a valid ID.
- 

## Event Handling:

- **Add Employee:** Clicking the "NEW +" button hides the current window and opens a new window (`OnboardPage`) for adding a new employee.
  - **Search:** The search functionality is tied to the `searchTxt` field and `searchBtn` button. If a valid ID is provided, the table is updated with the corresponding employee's data.
  - **Refresh:** The refresh button reloads all employee data from the database and updates the table.
- 

## Custom Table Cells:

- **Action Buttons:** The "Actions" column contains interactive buttons for each row. The buttons are rendered and edited using custom cell renderers and editors.
    - **View:** Opens a new page to view detailed information about the employee.
    - **Edit:** Allows editing the employee's information.
    - **Delete:** Prompts a confirmation box and deletes the selected employee if confirmed.
- 

## Supporting Classes and Functions:

1. **MessageBox:** A helper class used to show message dialogs. It's used in multiple places (e.g., when the user enters an invalid ID or confirms deletion).
  2. **OnboardPage, UpdatePage, ViewPage:** These classes represent different pages for adding a new employee, updating an existing employee, and viewing detailed information, respectively.
-



```

import java.awt.EventQueue;
import java.awt.Image;
import java.awt.Insets;
import java.awt.RenderingHints;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.Border;
import javax.swing.border.EmptyBorder;

import Backend.EmployeeApp;

import javax.swing.JLabel;

import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;

import javax.swing.SwingConstants;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.awt.event.ActionEvent;

public class ViewPage extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JLabel salaryField;
    private String empName;
    private String department;
    private String salary;

    /**
     * Create the frame.
     */
    public ViewPage(int id, JFrame callerFrame) {

        ImageIcon img = new
ImageIcon("E:\\Kodenest\\EmployeeManagementSystem\\src\\images\\profileicon.png");
        Image scaledImage = img.getImage().getScaledInstance(250, 250, Image.SCALE_SMOOTH); // Change
100x100 to desired size
        ImageIcon scaledIcon = new ImageIcon(scaledImage);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 565, 654);
        contentPane = new JPanel();
        contentPane.setBackground(new Color(255, 255, 255));
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel profileicon = new JLabel();
        profileicon.setIcon(scaledIcon);
        profileicon.setBounds(139, 10, 250, 260);
        contentPane.add(profileicon);

        EmployeeApp emp = new EmployeeApp();
        ResultSet res = emp.getEmployee(id);
        try {
            res.next();
            empName = res.getString(2);
            department = res.getString(3);
            salary = String.valueOf(res.getInt(4));
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        JLabel name_field = new JLabel(empName);
        name_field.setForeground(new Color(0, 0, 255));
        name_field.setHorizontalAlignment(SwingConstants.CENTER);
        name_field.setFont(new Font("Bernard MT Condensed", Font.BOLD, 26));
        name_field.setBounds(101, 282, 358, 50);
        contentPane.add(name_field);

        JLabel lblNewLabel_1 = new JLabel("ID:");
        lblNewLabel_1.setFont(new Font("Calibri", Font.BOLD, 18));
        lblNewLabel_1.setBounds(25, 396, 70, 35);
        contentPane.add(lblNewLabel_1);

        JLabel lblNewLabel_1_1 = new JLabel("DEPARTMENT:");
        lblNewLabel_1_1.setFont(new Font("Calibri", Font.BOLD, 18));
        lblNewLabel_1_1.setBounds(25, 456, 142, 35);
        contentPane.add(lblNewLabel_1_1);

        JLabel lblNewLabel_1_2 = new JLabel("SALARY:");
        lblNewLabel_1_2.setFont(new Font("Calibri", Font.BOLD, 18));
        lblNewLabel_1_2.setBounds(25, 516, 98, 35);
        contentPane.add(lblNewLabel_1_2);

```

```

JLabel id_field = new JLabel(String.valueOf(id));
id_field.setFont(new Font("Modern No. 20", Font.PLAIN, 16));
id_field.setHorizontalAlignment(SwingConstants.CENTER);
id_field.setBounds(58, 398, 90, 29);
contentPane.add(id_field);

JLabel dept = new JLabel(department);
dept.setHorizontalAlignment(SwingConstants.CENTER);
dept.setFont(new Font("Modern No. 20", Font.PLAIN, 16));
dept.setBounds(139, 458, 171, 29);
contentPane.add(dept);

salaryField = new JLabel(salary);
salaryField.setHorizontalAlignment(SwingConstants.CENTER);
salaryField.setFont(new Font("Modern No. 20", Font.PLAIN, 16));
salaryField.setBounds(101, 518, 81, 29);
contentPane.add(salaryField);

JButton btnNewButton = new JButton("EDIT");
btnNewButton.setBackground(new Color(255, 255, 255));
btnNewButton.setForeground(new Color(0, 0, 255));
btnNewButton.setFont(new Font("STKaiti", Font.BOLD, 14));
btnNewButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        new UpdatePage(id, empName, department, salary, callerFrame);
        setVisible(false);
    }
});
btnNewButton.setIcon(new ImageIcon(ViewPage.class.getResource("/images/icons8-edit-30.png")));
btnNewButton.setBounds(279, 569, 121, 35);
contentPane.add(btnNewButton);

JButton btnDelete = new JButton("DELETE");
btnDelete.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        MessageBox deleteBox = new MessageBox();
        deleteBox.delete(id);
    }
});
btnDelete.setBackground(new Color(255, 255, 255));
btnDelete.setIcon(new ImageIcon(ViewPage.class.getResource("/images/icons8-delete-24.png")));
btnDelete.setForeground(new Color(255, 0, 0));
btnDelete.setFont(new Font("Sitka Text", Font.BOLD, 14));
btnDelete.setBounds(410, 569, 134, 35);
contentPane.add(btnDelete);

JButton btnBack = new JButton("BACK");
btnBack.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        callerFrame.setVisible(true);
    }
});
btnBack.setIcon(new ImageIcon(ViewPage.class.getResource("/images/icons8-back-50.png")));
btnBack.setForeground(new Color(0, 128, 255));
btnBack.setFont(new Font("STKaiti", Font.BOLD, 12));
btnBack.setBackground(Color.WHITE);
btnBack.setBounds(10, 10, 121, 50);
contentPane.add(btnBack);

setVisible(true);
}
}

```

## Explanation:

This code defines a `ViewPage` class which is part of an Employee Management System application. It is designed to display detailed information about an employee based on their ID, including their profile image, name, department, and salary. Here's a breakdown of the main components and functionality:

## Key Features of the `viewPage` class:

1. **Employee Details:**
  - The employee's name, department, and salary are retrieved from the database using the provided `id` and displayed in labels.
2. **Buttons:**
  - **Edit Button:** Allows the user to update the employee's information by opening an `UpdatePage` (not defined in this code). When clicked, it passes the employee's details to `UpdatePage`.

- **Delete Button:** Deletes the employee's record from the database. The action is handled by the `MessageBox` class.
  - **Back Button:** Takes the user back to the previous screen (the `callerFrame`).
3. **Database Interaction:**
- The employee data is fetched from a backend class (`EmployeeApp`) which interacts with the database. The `getEmployee(id)` method retrieves the details for the employee with the specified ID.
  - The details are extracted from the `ResultSet` and stored in variables to display in the labels.
4. **Styling:**
- The layout and font styles for labels and buttons are customized for a clean, professional UI.
  - The buttons have icons to enhance the user experience (edit, delete, back).

## Flow:

- When the page is loaded (`ViewPage(int id, JFrame callerFrame)` constructor), the employee's details are fetched using the provided ID and displayed in a `JPanel`.
- The user can:
  - **Edit** the employee's details by clicking the "EDIT" button.
  - **Delete** the employee's record by clicking the "DELETE" button.
  - **Go Back** to the previous page using the "BACK" button.

## Onboard Page

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;

import Backend.EmployeeApp;
import javax.swing.SwingConstants;
import javax.swing.ImageIcon;

public class OnboardPage extends JFrame {

    private JTextField nameField;
    private JTextField deptField;
    private JTextField salField;
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;

    /**
     * Create the frame.
     */
    public OnboardPage(JFrame frame) {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 719, 570);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel lblNewLabel = new JLabel("Name *");
        lblNewLabel.setFont(new Font("Segoe UI Black", Font.BOLD, 20));
        lblNewLabel.setBounds(49, 60, 89, 28);
        contentPane.add(lblNewLabel);

        JLabel lblDepartment = new JLabel("Department *");
        lblDepartment.setFont(new Font("Segoe UI Black", Font.BOLD, 20));
        lblDepartment.setBounds(49, 186, 169, 28);
        contentPane.add(lblDepartment);

        JLabel lblNewLabel_2 = new JLabel("Salary *");
        lblNewLabel_2.setFont(new Font("Segoe UI Black", Font.BOLD, 20));
        lblNewLabel_2.setBounds(49, 301, 89, 28);
        contentPane.add(lblNewLabel_2);
    }
}
```

```

nameField = new JTextField();
nameField.setText("");
nameField.setFont(new Font("Tahoma", Font.PLAIN, 15));
nameField.setBounds(42, 102, 364, 39);
contentPane.add(nameField);
nameField.setColumns(10);

deptField = new JTextField();
deptField.setText("");
deptField.setFont(new Font("Tahoma", Font.PLAIN, 15));
deptField.setColumns(10);
deptField.setBounds(42, 237, 364, 39);
contentPane.add(deptField);

salField = new JTextField();
salField.setText("");
salField.setFont(new Font("Tahoma", Font.PLAIN, 15));
salField.setColumns(10);
salField.setBounds(42, 351, 364, 39);
contentPane.add(salField);

JButton addButton = new JButton("ONBOARD");
addButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        String name = nameField.getText();
        String dept = deptField.getText();
        String salary = salField.getText();
        EmployeeApp emp = new EmployeeApp();
        MessageBox message = new MessageBox();

        if (name.isBlank() || name.isEmpty() || name.equals("") || dept.isBlank() ||
dept.isEmpty() || dept.equals("")
            || salary.isBlank() || salary.isEmpty() || !salary.matches("[0-9]*")) {
            message.showMessageDialog("Enter all valid Details");
            return;
        }

        int sal = Integer.parseInt(salary);

        emp.addEmployee(name, dept, sal);
        message.showMessageDialog("Employee Onboarded Successfully");
        frame.setVisible(true);
        setVisible(false);
    }
});
addButton.setBackground(new Color(255, 255, 255));
addButton.setFont(new Font("Trebuchet MS", Font.BOLD, 18));
addButton.setForeground(new Color(128, 0, 255));
addButton.setBounds(574, 476, 121, 47);
contentPane.add(addButton);

JLabel lblNewLabel_1 = new JLabel("ONBOARD PAGE");
lblNewLabel_1.setHorizontalAlignment(SwingConstants.CENTER);
lblNewLabel_1.setForeground(new Color(128, 0, 255));
lblNewLabel_1.setFont(new Font("Tahoma", Font.BOLD, 30));
lblNewLabel_1.setBounds(167, 10, 406, 39);
contentPane.add(lblNewLabel_1);

JButton btnBack = new JButton("BACK");
btnBack.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        frame.setVisible(true);
        setVisible(false);
    }
});
btnBack.setIcon(new ImageIcon(OnboardPage.class.getResource("/images/icons8-back-50.png")));
btnBack.setForeground(new Color(128, 0, 255));
btnBack.setFont(new Font("STKaiti", Font.BOLD, 12));
btnBack.setBackground(Color.WHITE);
btnBack.setBounds(431, 476, 121, 50);
contentPane.add(btnBack);
setVisible(true);
}
}

```



## Explanation:

The `OnboardPage` class in this code represents a page in the Employee Management System for onboarding new employees. It provides an interface where the user can enter the name, department, and salary of an employee and then submit this information to add the employee to the system. Here's a breakdown of the main features and functionality:

## Key Components:

### 1. Input Fields:

- **Name Field:** A `TextField` for entering the employee's name.
- **Department Field:** A `TextField` for entering the employee's department.
- **Salary Field:** A `TextField` for entering the employee's salary. It validates that the input is numeric.

### 2. Labels:

- Labels are displayed above each input field to indicate what information should be entered. These include:
  - Name
  - Department
  - Salary
- An additional label at the top (`ONBOARD PAGE`) provides a title for the page.

### 3. Buttons:

- **Onboard Button:** When clicked, this button validates the input fields (checks that no field is left empty and that the salary is a valid number). If everything is valid, the employee is added to the system by calling the `addEmployee` method from the `EmployeeApp` backend class. It then displays a success message using the `MessageBox` class and returns to the previous screen.
- **Back Button:** When clicked, this button closes the current page and brings back the previous screen (`frame`).

### 4. Validation:

- Before adding the employee, the `OnboardPage` validates the fields:
  - The `name`, `department`, and `salary` fields cannot be empty or blank.
  - The `salary` field must be a valid integer.
- If validation fails, a message box is shown to prompt the user to enter all valid details.

### 5. Styling:

- The page uses various fonts, colors, and iconography to improve the user experience:
  - The `Onboard` button has a purple foreground and a white background.
  - The `Back` button has an icon and matches the purple color theme.
  - Labels and buttons are styled to have clear, readable fonts and consistent layout.

### 6. MessageBox:

- The `MessageBox` class (presumably defined elsewhere) is used to display messages to the user (e.g., success or error messages). When the employee is successfully onboarded, a success message is shown.

### 7. Navigation:

- When the "BACK" button is pressed, the current page is hidden (`setVisible(false)`), and the previous screen is shown again (`frame.setVisible(true)`).

## Flow:

1. The user is presented with a form to input the employee's name, department, and salary.
2. When the "ONBOARD" button is clicked:
  - The inputs are validated.

- If valid, the employee is added to the database.
  - A success message is displayed, and the previous screen is shown again.
  - If invalid, an error message prompts the user to correct the fields.
3. If the user clicks the "BACK" button, the form is hidden, and the previous page is displayed.

## Update Page

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import Backend.EmployeeApp;

import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.Color;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.ImageIcon;
import javax.swing.SwingConstants;

public class UpdatePage extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField nameField;
    private JTextField deptField;
    private JTextField salField;

    /**
     * Create the frame.
     */
    public UpdatePage(int id, String name, String department, String salary, JFrame frame) {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 719, 570);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel lblNewLabel = new JLabel("Name *");
        lblNewLabel.setFont(new Font("Segoe UI Black", Font.BOLD, 20));
        lblNewLabel.setBounds(49, 60, 89, 28);
        contentPane.add(lblNewLabel);

        JLabel lblDepartment = new JLabel("Department *");
        lblDepartment.setFont(new Font("Segoe UI Black", Font.BOLD, 20));
        lblDepartment.setBounds(49, 186, 169, 28);
        contentPane.add(lblDepartment);

        JLabel lblNewLabel_2 = new JLabel("Salary *");
        lblNewLabel_2.setFont(new Font("Segoe UI Black", Font.BOLD, 20));
        lblNewLabel_2.setBounds(49, 301, 89, 28);
        contentPane.add(lblNewLabel_2);

        nameField = new JTextField();
        nameField.setText(name);
        nameField.setFont(new Font("Tahoma", Font.PLAIN, 15));
        nameField.setBounds(42, 102, 364, 39);
        contentPane.add(nameField);
        nameField.setColumns(10);

        deptField = new JTextField();
        deptField.setText(department);
        deptField.setFont(new Font("Tahoma", Font.PLAIN, 15));
        deptField.setColumns(10);
        deptField.setBounds(42, 237, 364, 39);
        contentPane.add(deptField);

        salField = new JTextField();
        salField.setText(salary);
        salField.setFont(new Font("Tahoma", Font.PLAIN, 15));
        salField.setColumns(10);
        salField.setBounds(42, 351, 364, 39);
        contentPane.add(salField);
```

```

        JButton addButton = new JButton("UPDATE");
        addButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String name = nameField.getText();
                String dept = deptField.getText();
                MessageBox messageBox = new MessageBox();
                EmployeeApp emp = new EmployeeApp();

                if (name.isBlank() || name.isEmpty() || name.equals("") || dept.isBlank() ||
                    dept.isEmpty() || dept.equals("") || salField.getText().isBlank() || salField.getText().isEmpty() ||
                    !salField.getText().matches("[0-9]*")) {
                    messageBox.showMessageDialog("Enter all valid Details");
                    return;
                }

                int sal = Integer.parseInt(salField.getText());
                try {
                    emp.updateEmployee(id, name, dept, sal);
                    messageBox.showMessageDialog("Updated Successfully");
                    frame.setVisible(true);
                } catch (Exception e2) {
                    // TODO: handle exception
                    e2.printStackTrace();
                }
            }
        });
        addButton.setBackground(new Color(255, 255, 255));
        addButton.setFont(new Font("Trebuchet MS", Font.BOLD, 18));
        addButton.setForeground(new Color(128, 0, 255));
        addButton.setBounds(574, 476, 121, 47);
        contentPane.add(addButton);

        JButton btnBack = new JButton("BACK");
        btnBack.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                frame.setVisible(true);
                setVisible(false);
            }
        });
        btnBack.setIcon(new ImageIcon(UpdatePage.class.getResource("/images/icons8-back-50.png")));
        btnBack.setForeground(new Color(128, 0, 255));
        btnBack.setFont(new Font("STKaiti", Font.BOLD, 12));
        btnBack.setBackground(Color.WHITE);
        btnBack.setBounds(443, 477, 121, 50);
        contentPane.add(btnBack);

        JLabel lblNewLabel_1 = new JLabel("EDIT PAGE");
        lblNewLabel_1.setForeground(new Color(128, 0, 255));
        lblNewLabel_1.setFont(new Font("Tahoma", Font.BOLD, 30));
        lblNewLabel_1.setHorizontalAlignment(SwingConstants.CENTER);
        lblNewLabel_1.setBounds(178, 10, 406, 39);
        contentPane.add(lblNewLabel_1);
        setVisible(true);
    }
}

```

## Explanation:

The `UpdatePage` class is a part of the Employee Management System that allows the user to update an employee's details. This page provides input fields for the employee's name, department, and salary, and enables the user to save the updated information. Below is a detailed breakdown of the class functionality:

## Key Components:

### 1. Input Fields:

- **Name Field:** A `JTextField` pre-populated with the employee's current name.
- **Department Field:** A `JTextField` pre-populated with the employee's current department.



- **Salary Field:** A `UITextField` pre-populated with the employee's current salary.
- 2. **Labels:**
  - **Name Label:** Describes the name field.
  - **Department Label:** Describes the department field.
  - **Salary Label:** Describes the salary field.
  - **Edit Page Title:** A title label at the top indicating the purpose of the page ("EDIT PAGE").
- 3. **Buttons:**
  - **Update Button:** When clicked, this button validates the input fields, checks if the name, department, and salary are valid, and then updates the employee's details in the backend.
  - **Back Button:** When clicked, this button hides the current page and returns the user to the previous screen (`frame`).
- 4. **Validation:**
  - Before updating the employee's details, the `UpdatePage` performs validation:
    - **Name and Department Fields** cannot be empty.
    - **Salary Field** must be numeric and cannot be empty.
  - If any validation fails, a message is shown to the user asking for valid details.
- 5. **MessageBox:**
  - The `MessageBox` class (presumably defined elsewhere in your code) is used to display messages to the user:
    - If the inputs are valid, a success message ("Updated Successfully") is shown.
    - If the validation fails, a message prompts the user to enter valid details.
- 6. **Backend Interaction (EmployeeApp):**
  - The `EmployeeApp` class is used to interact with the backend system, specifically the `updateEmployee` method, which updates the employee details in the database.
- 7. **Styling:**
  - The page uses various fonts and colors to improve the user experience:
    - The **Update** button has a purple foreground and a white background.
    - The **Back** button uses an icon and matches the purple color theme.
    - Labels and buttons are styled with clear, readable fonts and appropriate alignment.

## Flow:

1. The `UpdatePage` constructor is invoked with the current employee's details (ID, name, department, salary) and the previous screen (`frame`).
  2. The fields are pre-populated with the current employee information.
  3. The user can modify the fields if necessary:
    - If the user clicks **UPDATE**:
      - The input values are validated.
      - If valid, the employee details are updated in the backend via the `EmployeeApp` class.
      - A success message is shown, and the page returns to the previous screen (`frame`).
      - If invalid, an error message is shown.
  4. If the user clicks **BACK**, the current page is hidden, and the previous screen (`frame`) is shown again.
-

## 7. Screenshots

### Home Page:

EMPLOYEE MANAGEMENT SYSTEM

SEARCH

NEW +


EMPLOYEES

id	name	department	salary	Actions
1	Abhisek Nayak	Developement Team	5120000	Actions
6	Om Panda	Designing	856321	Actions
8	Harihar Maharana	Backend Developement	600000	Actions

©KODNEST | ALL RIGHTS RESERVED

### View Page:

BACK



Abhisek Nayak

ID: 1

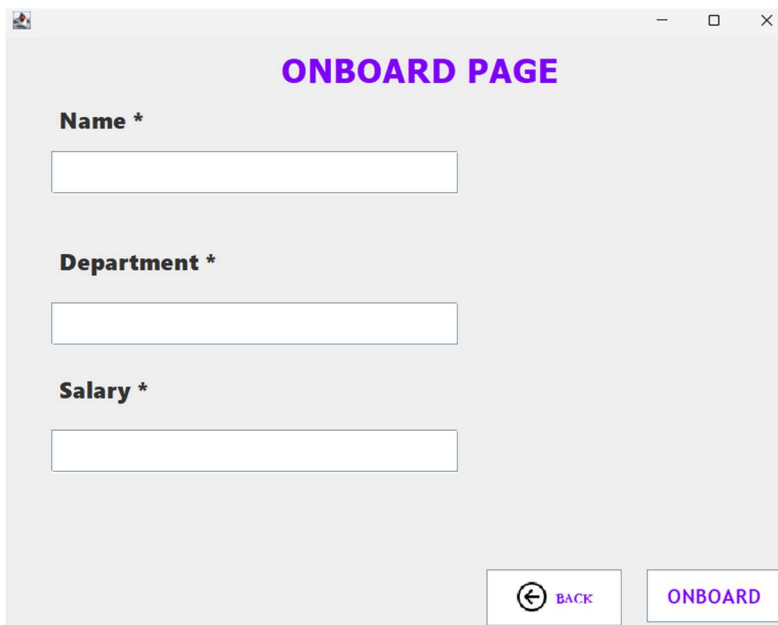
DEPARTMENT: Developement Team

SALARY: 5120000

EDIT

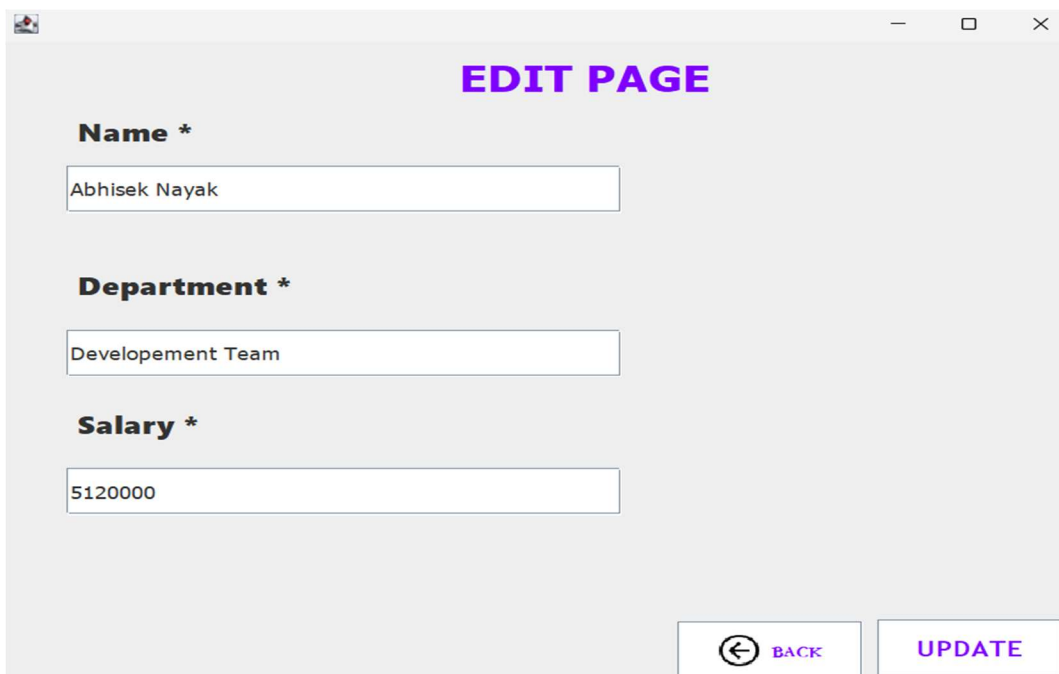
DELETE

## Onboard Page:



A screenshot of a web application window titled "ONBOARD PAGE". The window has a light gray background and standard window controls (minimize, maximize, close) in the top right corner. The title "ONBOARD PAGE" is displayed in bold purple text at the top center. Below the title, there are three form fields, each with a label and an asterisk: "Name \*", "Department \*", and "Salary \*". Each label is in bold black text. The input fields are empty white rectangles. At the bottom right of the window, there are two buttons: a "BACK" button with a left-pointing arrow icon and the word "BACK" in purple, and an "ONBOARD" button with the word "ONBOARD" in purple.

## Update Page:



A screenshot of a web application window titled "EDIT PAGE". The window has a light gray background and standard window controls (minimize, maximize, close) in the top right corner. The title "EDIT PAGE" is displayed in bold purple text at the top center. Below the title, there are three form fields, each with a label and an asterisk: "Name \*", "Department \*", and "Salary \*". Each label is in bold black text. The input fields contain the following text: "Abhisek Nayak", "Development Team", and "5120000". At the bottom right of the window, there are two buttons: a "BACK" button with a left-pointing arrow icon and the word "BACK" in purple, and an "UPDATE" button with the word "UPDATE" in purple.

---

## 8. Conclusion

The Employee Management System is a robust application that simplifies employee data management. This project helped in enhancing my knowledge of Java GUI development, database management, and CRUD operations. It also provided valuable insights into creating user-friendly interfaces and efficient backend systems.

---

