**Employee Management System Documentation**

**1. Introduction**

**1.1 Purpose of the Document**

This document serves as a comprehensive guide for the Employee Management System developed in core Java. It outlines the system's design, implementation, and usage, providing details for developers and users alike.

**1.2 Project Overview**

The Employee Management System is designed to facilitate efficient management of employee data within an organization. It includes features for adding, removing, and listing employees by department, enhancing organizational workflow.

**1.3 Scope**

The scope of this project encompasses core Java development principles, including object-oriented programming, file handling, and exception management. It does not involve database interactions or complex user interfaces.

**2. System Architecture**

**2.1 High-Level Overview**

The system consists of three main classes: **Employee**, **Department**, and **Management**. These classes collaborate to provide functionalities such as adding and removing employees, department-wise listing, and file serialization.

**2.2 Components**

2.2.1 Employee Class

The **Employee** class represents individual employees. It holds attributes such as Employee ID, Name, Department, and Position.

2.2.2 Department Class

The **Department** class represents organizational departments. It holds attributes such as Department ID and Department Name.

2.2.3 Management Class

The **Management** class acts as a controller class, managing employee and department data. It includes methods for adding, removing, and displaying employee information.

**3. Technologies Used**

**3.1 Core Java**

The project is implemented using core Java, leveraging its fundamental features such as classes, objects, and inheritance.

**3.2 Object-Oriented Principles**

Object-oriented principles, including encapsulation and polymorphism, are employed for designing modular and extensible classes.

**3.3 File Handling**

File handling is used to save and load employee and department data, ensuring persistent storage.

**3.4 Exception Handling**

Exception handling mechanisms are implemented to gracefully manage runtime errors and enhance the system's robustness.

**4. Class Diagram**

**4.1 UML Diagram**

A Unified Modeling Language (UML) diagram visually represents the relationships and interactions between the **Employee**, **Department**, and **Management** classes.

**4.2 Class Relationships**

The relationships between classes are depicted, showcasing how data flows within the system.

**4.3 Class Responsibilities**

Each class's responsibilities are detailed, defining their roles in the overall system.

**5. Project Setup**

**5.1 Environment Requirements**

Ensure that a Java Development Kit (JDK) is installed, and the system environment is configured appropriately.

**5.2 IDE Configuration**

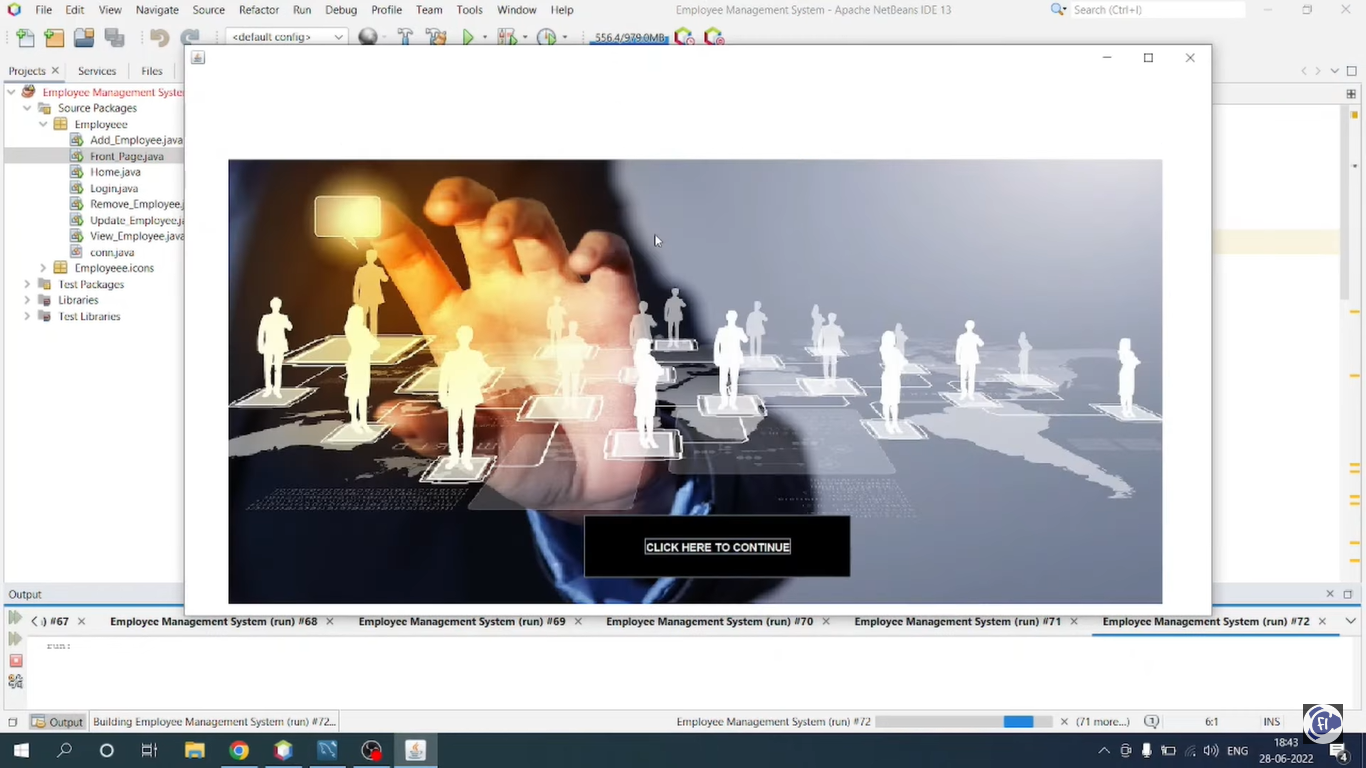
Guidelines for configuring Integrated Development Environment (IDE) settings are provided to facilitate smooth development.

**5.3 Project Structure**

The project is organized into packages and directories, promoting a structured and modular codebase.

**5.4 External Libraries (if any)**

Specify any external libraries used and provide instructions for their integration into the project.

  
Project Output :

