Java Mini Project - : **Any Real case Application Management System**

****Project Overview:****

You are tasked with developing a Java application,**Any Real case Application Management System**, that allowsusers(administrators) to manage user records efficiently. The system will include functionalities such as addingnewrecord,updating existing records, deleting records, searching of records and viewing records. The applicationwill interactwithadatabase to persist data and will incorporate Object-Oriented Programming (OOP) principles, Java CollectionsFramework,and robust exception handling.

****Core Requirements:****

1. **User Authentication:**

- The application should allow administrators to log in using a username and password. - Implement basic authentication logic, including exception handling for incorrect login attempts. **2. **User Record Management:****

- **Add New Record:**

- The system should provide an option to add new records, including user details - The data should be stored in a relational database (e.g., MySQL, PostgreSQL).

- **Update existing RecordDetails:**

- Allow updating of user details based on the ID.

- **Searching of RecordDetails:**

- Allow searching of user details based on the ID.

- **Delete Record:**

- Implement functionality to delete a record from the database.

- **View all Records:**

- The application should allow viewing of all records, with options to filter and sort using Java Collections. **3.**

****Database Connectivity:****

- Use JDBC (Java Database Connectivity) to connect the application to the chosen relational database. - Ensure that database operations are efficiently managed and exceptions (e.g., SQLExceptions) are properlyhandled. **4. **Object-Oriented Design:** -** - Design the system using OOP principles:

- **Classes and Objects:** Define required classes as per choosen application.

- **Inheritance:**Extend a base class to reuse it features in its child classes.

- **Encapsulation:** Use private fields and provide getter/setter methods.

- **Polymorphism:** Implement method overriding to customize behavior in derived classes. - **Abstraction:** Use abstract classes or interfaces for defining common behaviors.

5. **Use of Java Collections:**

- Store records in a suitable Java collection (e.g., `ArrayList`, `HashMap`) before performing database operations. - Implement features such as sorting and searching using Java Collections utilities.

6. **Exception Handling:**

- Implement exception handling across the application to manage different error scenarios: - **Input Validation:** Handle cases where invalid data is provided by the user.

- **Database Exceptions:** Properly handle SQLExceptions and ensure the application continues to runsmoothly. -**Custom

Exceptions:** Define and throw custom exceptions where appropriate (e.g., `UserNotFoundException`).