APPROACH PAPER

Task Description

Objective	Task Name	Description
Development(Backend)	Perform CRUD Operation	1. The project should have a process of user registration, updation, deletion, and retrieval.
		2. The project should consider the unique username even after the deletion of one user the same should not be used.
		3. After registration a password should be generated (need not implement the sms part, return the password in response). This password should be directly linked with the user name. i.e. username can be retrieved via password. Ex. userName: ANURAG, password: 114211817 Note: This password mapping should reflect the strong encryption, logic behind the encryption should have a strong base. The retrieval should not be ambiguous (like the given example).
		4.Log Level and effectiveness of logs.

Index

Technical Challenge(s) (Background, Context, and Motivation)

Description of Task:

To develop a **User Management Application** using Spring Boot that allows users to create, update, delete, and retrieve accounts while ensuring that each user has a unique username and an encrypted password. The system will utilise the Caesar cipher algorithm for encrypting and decrypting usernames and passwords.

Current System:

Currently, there is no existing system. I am starting from scratch.

Current Scenario:

Diagrammatic representation of the current system ***(wherever applicable)***

Proposed Solution:

The proposed solution aims of this project is to develop a **User Management Application** using Spring Boot that enables users to create, update, delete, and retrieve accounts while ensuring that each user has a unique username and a securely encrypted password. This system will utilise the Caesar cipher algorithm for generating encrypted passwords and decrypting usernames, providing a basic level of security for user credentials.

(Why: Rationale for why your solution will work)

The user management Application will effectively provide secure account management by enforcing unique usernames, utilizing the Caesar cipher for password encryption, and implementing intuitive CRUD operations, ensuring a user-friendly experience while safeguarding sensitive data.

Key Requirements:

1. **User Registration**:
    - Each user must have a unique username.
    - Upon account creation, the system generates an encrypted password linked to the username with a variable length.
    - Passwords must be encrypted using the Caesar cipher algorithm.
2. **User Management**:
    - Users can create accounts using their unique usernames.
    - Users can update their account details (e.g., email, role, salary phone number, etc) but not the username.
    - Users can delete their accounts,
    - Users can get all the existing user's storage in-memory databases.
    - Users can retrieve their usernames using the encrypted password.

Technology Details

- **Backend Framework**: Spring Boot
- **Database**: H2 (in-memory)
- **Programming Language**: Java
- **Encryption**: Custom implementation of the Caesar cipher algorithm

Proposed Architecture/

1. Architecture Layers:

The system follows a layered architecture pattern, typically comprising three main layers:

- **Presentation Layer** (Controller)
- **Business Logic Layer** (Service)
- **Data Access Layer** (Model/Repository)

2. Component Breakdown:

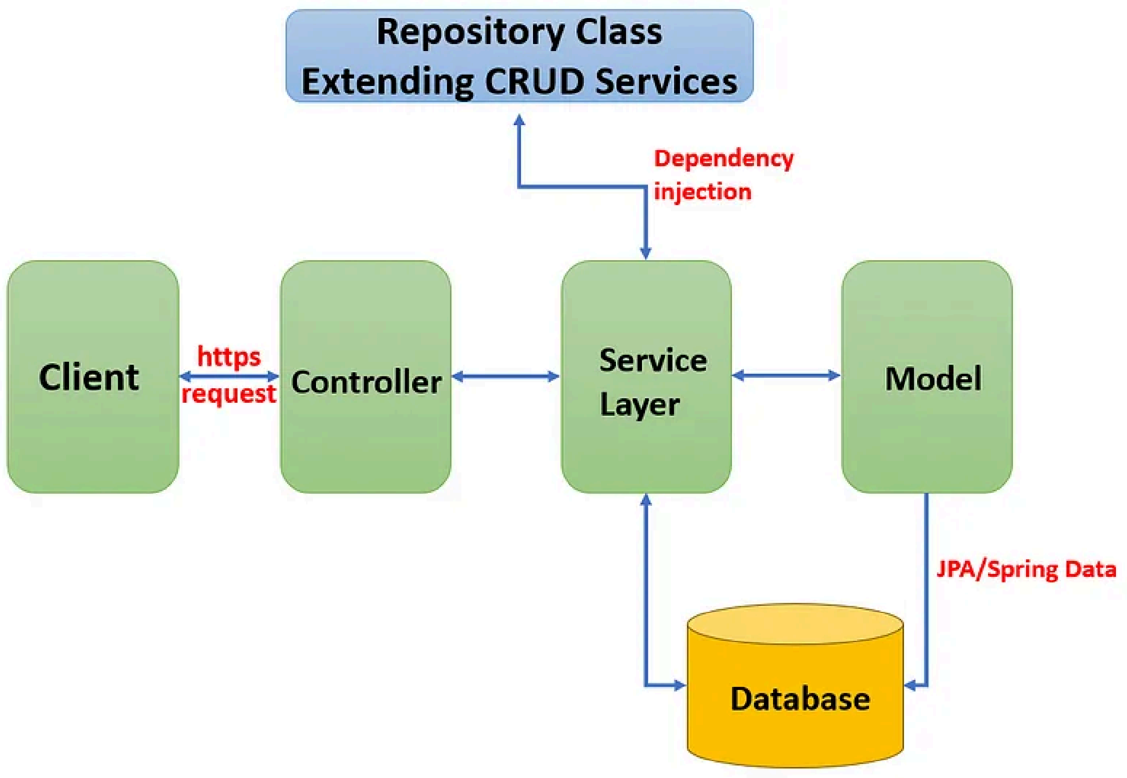
1. **Presentation Layer** (Controller):
    - Handles incoming HTTP requests and sends responses back to clients.
    - Maps URLs to specific actions, providing user registration, account updates, deletions, and username retrieval endpoints.
    - Example Controllers:UserController: Manages user-related operations (create, update, delete, retrieve).
2. **Business Logic Layer** (Service):
    - Contains the core logic for user registration and account management.
    - Validates inputs, processes requests, and interacts with the data access layer.
    - Utilizes the Caesar cipher utility for encrypting and decrypting passwords.
    - Example Services:UserService: Handles business operations related to user accounts.
3. **Data Access Layer** (Model/Repository):
    - Represents the structure of user data and manages data storage.
    - Utilizes in-memory storage (e.g., H2 database) for user details.
    - Implements CRUD operations for user data.
    - Example Models:User: Represents user attributes (id username, password, email, profession(optional)).

Application Workflow:

1. **User Registration**:
    - Represents the structure of user data and manages data storage.
    - Utilizes in-memory storage (e.g., H2 database) for user details.
    - Implements CRUD operations for user data.
    - Example Models:UserEntity: Represents user attributes (username, password, email, role salary phonenumber).
2. **Account Update**:
    - The user sends a request to update account details.
    - **UserController** receives the request and invokes **UserService** to process the update.
    - **UserService** performs the update and saves the changes in the data access layer.

3. **Account Deletion**:
    - The user sends a request to delete their account.
    - **UserController** calls UserService, which marks the account as deleted and retains the encrypted password.
4. **Username Retrieval**:
    - User sends a request with the encrypted password to retrieve their username.
    - **UserController** processes the request and calls **UserService** to decrypt the password and retrieve the username.

Sequence Diagram:



Hardware Requirements

Processor:

- **Minimum:** Dual-core (Intel i3 or equivalent)
- **Recommended:** Quad-core (Intel i5 or higher)

RAM:

- **Minimum:** 4 GB
- **Recommended:** 8 GB or more

Disk Space:

- **Minimum:** 1 GB free
- **Recommended:** 10 GB or more for project files and dependencies

Software Requirements, including Operating System :

Operating System:

- Any modern operating system that supports the software requirements (e.g. Linux distribution like Ubuntu 22.04 LTS.).

Software Components:

- **Java Development Kit (JDK):**
    - **Version:** JDK 17.0.12 (recommended for Spring Boot 3.0)
- **Integrated Development Environment (IDE):**
    - Visual Studio Code (with Java extensions)
- **Build Tool:**
    - Maven 3.9.4
- **Database (for persistent storage):**
    - H2 Database (in-memory for development/testing)
- **Spring Boot:**
    - **Version:** 3.3.2
- **Other Dependencies:**
    - Depending on your application, you may need additional libraries such as:
        - Spring Data JPA (for database access)
        - Spring Web (for REST APIs)
        - Lombok (to reduce boilerplate code)
        - Jackson (for JSON processing)

Optional Tools

- **Postman:** For testing RESTful APIs.
- **Version Control System:** Git for source code management.

Networking Requirements including Internet services :

- **Ethernet Connection:** Stable internet access for downloading dependencies and updates.