**Proof of Concept (PoC) Document**

**User CRUD Operations - Spring Boot Web Application**

**1. Introduction**

This Proof of Concept (PoC) document outlines the implementation of a **User Management System** built using **Spring Boot and MySQL**. The application supports **CRUD (Create, Read, Update, Delete) operations** through a REST API.

**2. Technology Stack**

* **Backend**: Spring Boot (Spring MVC, Spring Data JPA)
* **Database**: MySQL
* **Build Tool**: Maven
* **Logging**: SLF4J with Logback
* **Exception Handling**: Custom exception handling with @RestControllerAdvice
* **Testing**: Postman

**3. Features & API Endpoints**

**User Management APIs**

| **HTTP Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /api/users/fetch | Fetch all users |
| GET | /api/users/fetch/{id} | Fetch user by ID |
| POST | /api/users/create | Create a new user |
| PUT | /api/users/update/{id} | Update user by ID |
| DELETE | /api/users/delete/{id} | Delete user by ID |

**4. Application Architecture**

**Layered Architecture**

1. **Controller Layer**: Handles HTTP requests and responses.
2. **Service Layer**: Implements business logic.
3. **Repository Layer**: Interacts with the database using Spring Data JPA.
4. **Exception Handling**: Centralized error handling with @RestControllerAdvice.

**5. Implementation Details**

**Entity Class: User**

Defines user attributes and database mapping.

**Repository: UserRepository**

Extends JpaRepository for database interaction.

**Service: UserServiceImpl**

Implements business logic for CRUD operations.

**Controller: UserController**

Handles API requests for user operations.

**Exception Handling: GlobalExceptionHandler**

Handles custom exceptions like UserNotFoundException.

**6. Conclusion**

This PoC demonstrates a User Management System using Spring Boot, providing a scalable and efficient backend solution with structured exception handling and logging.