# **DBMS Project Documentation**

## **1. Project Overview**

The goal of this project is to create a lightweight, user-friendly DBMS with the following features:

* User authentication (login, signup, forgot password).
* Secure password storage using SHA-256 hashing.
* User-specific folders to store databases and tables.
* Support for creating, reading, updating, and deleting (CRUD) tables in .csv or other common formats.
* Transaction management with concurrency control and rollback mechanisms.
* Logging of all changes in .txt files.
* Role-based access control (admin can grant/revoke permissions).

## **2. System Architecture**

The system will consist of the following components:

1. **User Authentication Module**: Handles user signup, login, and password recovery.
2. **Database Storage Module**: Manages user-specific folders and table storage.
3. **Table Management Module**: Implements CRUD operations on tables.
4. **Transaction and Concurrency Module**: Ensures data integrity during concurrent operations.
5. **Logging Module**: Tracks changes made to the database.
6. **Access Control Module**: Manages user permissions.

## **3. Detailed Design**

### **3.1 User Authentication Module**

#### **Features:**

* **Signup**:
  + Username (unique, primary key).
  + Password validation:
    - Minimum 8 characters.
    - At least 1 capital letter.
    - At least 1 special character.
    - At least 1 number.
  + Email (optional, can be added later).
  + Passwords stored as SHA-256 hashes.
* **Login**:
  + Authenticate users using username and password.
* **Forgot Password**:
  + Allow users to reset their password after email verification.

#### **Implementation Steps:**

1. Create a users table to store user details:
   * username (Primary Key)
   * password\_hash (SHA-256 hash)
   * email (Optional)
2. Implement password validation during signup.
3. Use SHA-256 hashing to store passwords securely.
4. Implement a password reset mechanism using email verification.

### **3.2 Database Storage Module**

#### **Features:**

* Each user has a dedicated folder to store their databases and tables.
* Tables are stored in .csv or other common formats.

#### **Implementation Steps:**

1. Create a root directory for the DBMS (e.g., DBMS\_Root).
2. For each user, create a subfolder named after their username (e.g., DBMS\_Root/user1).
3. Store all user-specific tables and databases in their respective folders.

### **3.3 Table Management Module**

#### **Features:**

* Users can create, read, update, and delete tables.
* Support for .csv or other formats.
* Implement CRUD operations:
  + **Create**: Create a new table.
  + **Read**: Read data from a table.
  + **Update**: Update records in a table.
  + **Delete**: Delete records or entire tables.
* Additional functionalities:
  + Rename tables.
  + Insert new records.

#### **Implementation Steps:**

1. Use libraries like pandas (Python) to handle .csv files.
2. Implement functions for each CRUD operation.
3. Validate user inputs to prevent errors.

### **3.4 Transaction and Concurrency Module**

#### **Features:**

* Support for transactions (begin, commit, rollback).
* Concurrency control using locking mechanisms.
* Ensure data integrity during concurrent operations.

#### **Implementation Steps:**

1. Use a locking mechanism (e.g., file locks) to prevent concurrent writes.
2. Implement transaction logs to track changes.
3. Provide rollback functionality using transaction logs.

### **3.5 Logging Module**

#### **Features:**

* Log all changes made to the database.
* Log format:
  + Timestamp
  + Username
  + Action (e.g., "Updated email from X to Y")
* Store logs in .txt files.

#### **Implementation Steps:**

1. Create a logs folder within each user's directory.
2. Write logs to a .txt file after every operation.

### **3.6 Access Control Module**

#### **Features:**

* Admin can grant or revoke permissions for users.
* By default, users cannot access other users' databases.
* Permissions include:
  + Read access
  + Write access
  + Delete access

#### **Implementation Steps:**

1. Create a permissions table:
   * username
   * table\_name
   * read\_access (boolean)
   * write\_access (boolean)
   * delete\_access (boolean)
2. Implement functions for the admin to update permissions.

## **4. Technology Stack**

* **Programming Language**: Python (recommended for its simplicity and libraries).
* **Libraries**:
  + pandas for handling .csv files.
  + hashlib for SHA-256 hashing.
  + os and shutil for file and folder management.
  + logging for creating logs.
* **Database Storage**: File-based storage (e.g., .csv files).

## **5. Implementation Plan**

1. **Phase 1**: User Authentication Module
   * Implement signup, login, and forgot password functionality.
2. **Phase 2**: Database Storage Module
   * Create user-specific folders and table storage.
3. **Phase 3**: Table Management Module
   * Implement CRUD operations.
4. **Phase 4**: Transaction and Concurrency Module
   * Add transaction support and locking mechanisms.
5. **Phase 5**: Logging Module
   * Implement logging functionality.
6. **Phase 6**: Access Control Module
   * Add role-based access control.

## **6. Testing Plan**

* Test each module individually (unit testing).
* Perform integration testing to ensure all modules work together.
* Test edge cases (e.g., invalid inputs, concurrent access).

## **7. Future Enhancements**

* Add support for SQL-like queries.
* Implement indexing for faster search operations.
* Add a graphical user interface (GUI).
* Support for additional file formats (e.g., JSON, Excel).

## **8. Conclusion**

This document provides a comprehensive guide to building a DBMS with the specified features. By following the implementation plan and testing each module thoroughly, you can create a robust and user-friendly system. Good luck with your project!