In Memory Db with Indexing

1. Requirements

Project Requirements: Low-Level Design for an In-Memory Database

1. **Database Creation**:

   The system must allow users to create new **in-memory databases**, ensuring fast data access and manipulation. These databases will exist in memory during runtime and will be cleared when the application shuts down or restarts.

2. **Database Security**:

   Users should be able to set a **password** for their in-memory databases to restrict unauthorized access and protect sensitive information.

3. **Table and Schema Management**:

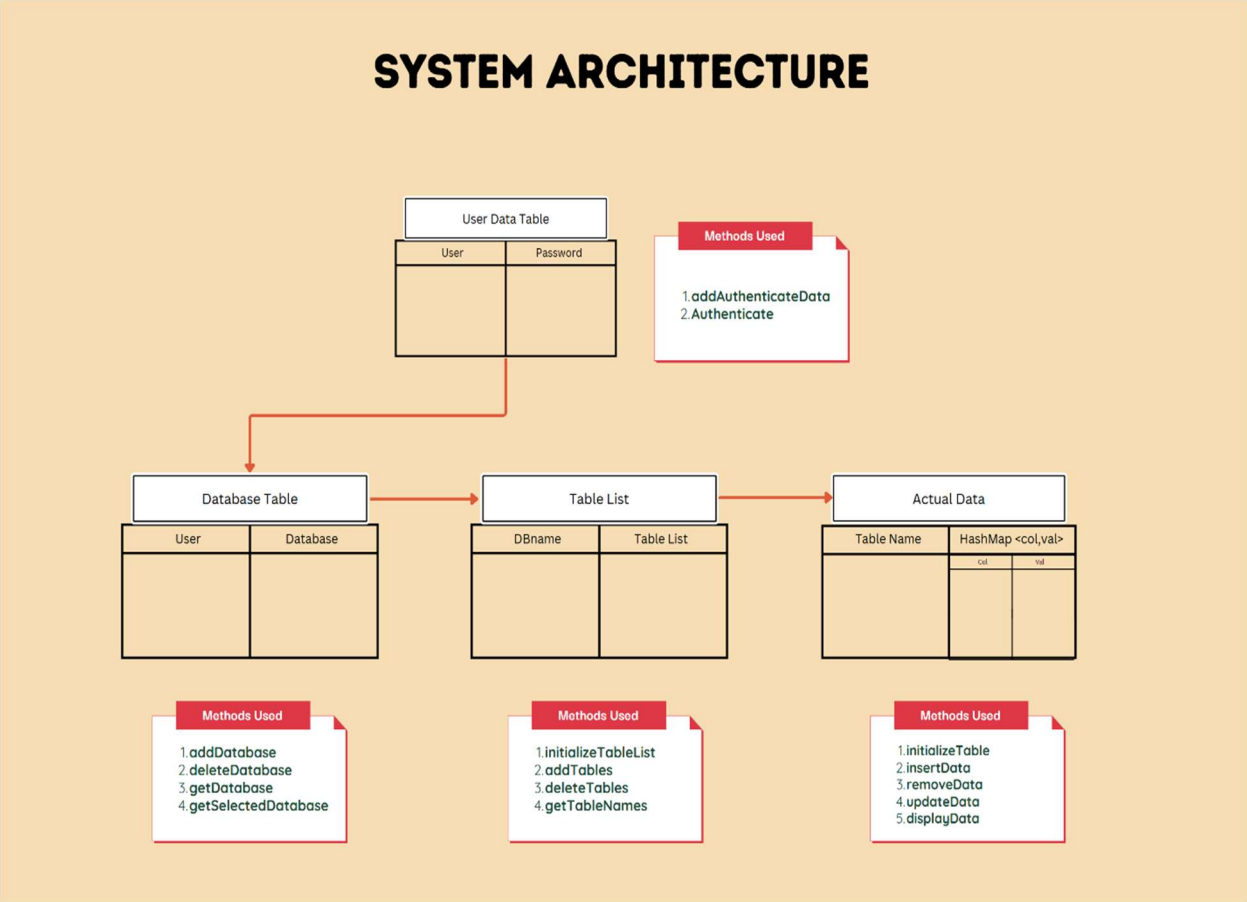
   Users must have the ability to create **tables** within the database. The system should allow the definition of **schemas**, specifying column names and their respective data types.

4. **CRUD Operations on Tables**:

   The system should support the following core operations on tables:

   a. **Insert**: Add new rows to the table.
   b. **Delete**: Remove specific rows from the table based on conditions or row identifiers.
   c. **Update**: Modify data in specific rows based on conditions.

5. **Indexing for Optimized Performance**:
   Users should be able to create **indexes** on specific columns of a table. These indexes must be implemented using efficient data structures to enhance the performance of queries, ensuring fast lookups and retrieval of rows based on column values.



Explaining the UML diagram

Database Class Methods

1. **addDatabase(user, databaseName)**
   Creates a new database for the specified user.

   Parameters:

   user: The user for whom the database is being created.

2. **databaseName**: The name of the database to be created.

   deleteDatabase(user, databaseName)
   Deletes an existing database associated with the specified user.

   Parameters:

   user: The user who owns the database.

   databaseName: The name of the database to be deleted.

3. **getDatabase(user, databaseName)**
   Retrieves all the databases associated with the specified user.

   Parameters:

   user: The user whose databases are being retrieved.

   databaseName: The name of the database to retrieve.

TableData Class Methods

1. **initializeTable(String tableName)**

2. Creates a new table with the specified name.

   a. **Parameters**:
      i. tableName: The name of the table to be created.

3. **insertData(String tableName, String colname, String val)**

   Inserts a new value into the specified column of the table.

   a. **Parameters**:
      i. tableName: The name of the table where data will be inserted.
      ii. colname: The column in which the value will be added.
      iii. val: The value to be inserted.

4. **removeData(String tableName, String colname, String val)**

   Removes data from the specified column of the table based on the value provided.

   a. **Parameters**:
      i. tableName: The name of the table from which data will be removed.
      ii. colname: The column where the value will be searched.
      iii. val: The value to be removed.

5. **updateData(String tableName, String colname, String val)**

   Updates data in the specified column of the table.

   a. **Parameters**:
      i. tableName: The name of the table where the data will be updated.
      ii. colname: The column to be updated.
      iii. val: The new value to replace the existing one.

6. **displayData(String tableName)**

   Displays all the data from the specified table.

   a. **Parameters**:
      i. tableName: The name of the table to display data from.

TableList Class Methods

1. **initializeTableList(String dbname)**

2. Initializes a list of tables for the specified database.

   a. **Parameters**:
      i. dbname: The name of the database for which the table list will be initialized.

3. **boolean addTables(String dbname, String tableName)**

Adds a new table to the specified database. Returns true if the table is successfully added, and false otherwise.

   a. **Parameters**:
      i. dbname: The name of the database where the table will be added.
      ii. tableName: The name of the table to be added.

4. **boolean deleteTables(String dbname, String tableName)**

Deletes a table from the specified database. Returns true if the table is successfully deleted, and false otherwise.

   a. **Parameters**:
      i. dbname: The name of the database from which the table will be deleted.
      ii. tableName: The name of the table to be deleted.

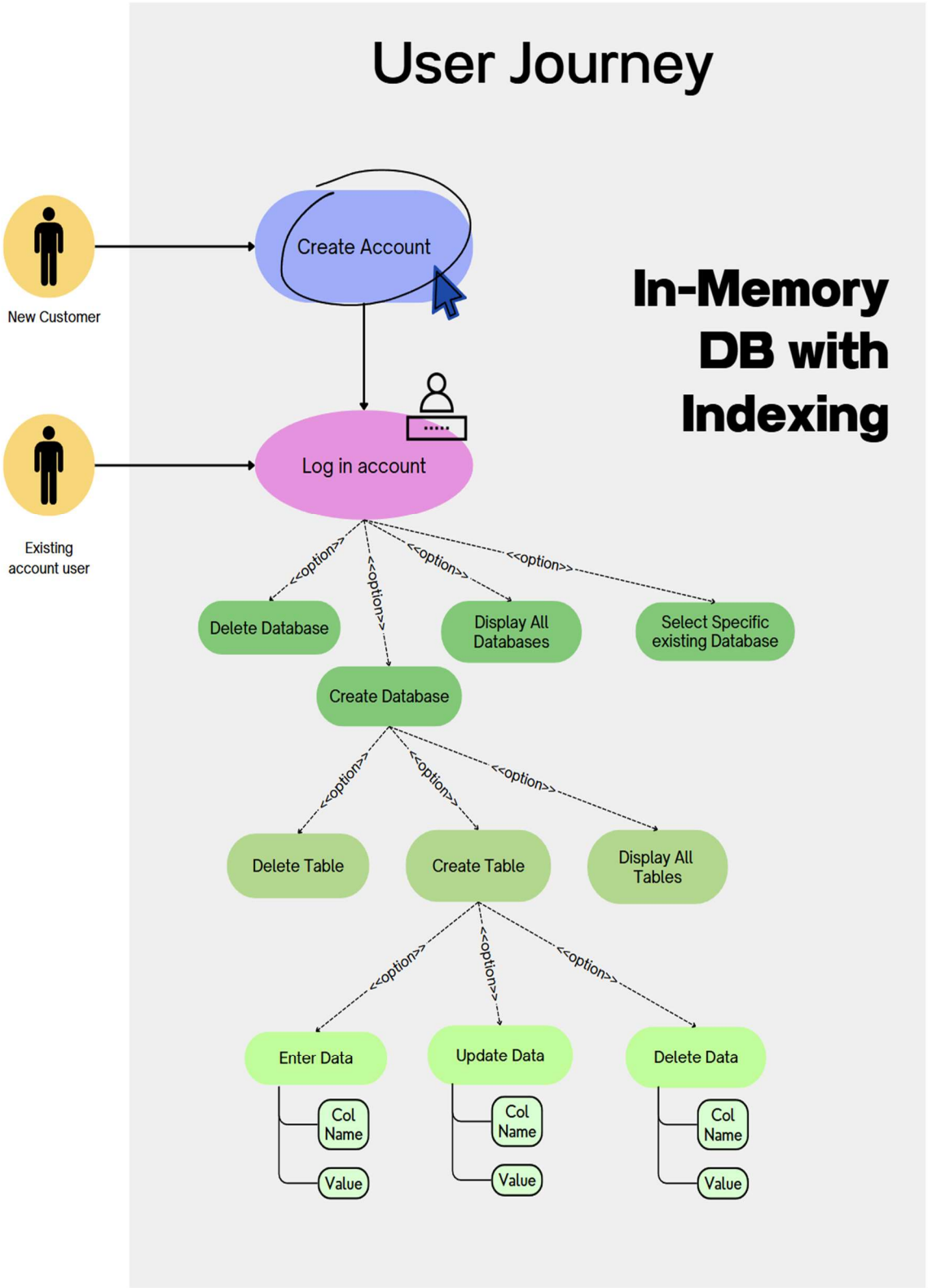
5. **getTableNames(String dbname)**

   Retrieves the names of all tables in the specified database.

   a. **Parameters**:
      i. dbname: The name of the database whose table names will be retrieved.

UserData Class Methods

1. **boolean authenticate(String name, String password)**
2. Authenticates the user by validating their credentials.
   a. **Parameters**:
      i. name: The username of the user attempting to authenticate.
      ii. password: The password associated with the user's account.
   b. **Returns**:
      i. true if the provided credentials are valid.
      ii. false if the credentials are invalid.


User Journey
New Customer
Create Account
In-Memory DB with Indexing
Existing account user
Log in account
<<option>>
Delete Database
Display All Databases
Select Specific existing Database
Create Database
Delete Table
Create Table
Display All Tables
Enter Data
Update Data
Delete Data
Col Name
Value

Explanation of the user journey diagram

1. User Authentication

Before interacting with the database system, the user must authenticate themselves:

- **Action**: boolean authenticate(String name, String password) in the UserData class.
- **Purpose**: Ensures only authorized users can interact with the database system.

2. Creating a Database

- **Action**: addDatabase(String user, String databaseName) in the Database class.
- **Purpose**: Allows the user to create a new in-memory database to organize their data.
- **Example**:
    - User creates a database named EmployeeDB.

3. Setting a Password for the Database

- **Action**: A secure password can be set at the time of user creation (handled via the UserData class).
- **Purpose**: Ensures that only the authenticated user can access and manipulate the database.
- **Example**:
    - User sets the password password123 for their account during creation.

4. Creating a Table

- **Action**: initializeTable(String tableName) in the TableData class.
- **Purpose**: Adds a new table to the database.
- **Example**:
    - User creates a table named Employees in the database EmployeeDB.

5. Inserting Data into the Table

- **Action**: insertData(String tableName, String colname, String val) in the TableData class.
- **Purpose**: Adds rows of data into the specified table.
- **Example**:
    - User inserts data into the Employees table:
        - Name column: John Doe
        - Position column: Software Engineer

6. Adding an Index to a Column

- **Action**: createIndex(String tableName, String colname) in the TableData class.
- **Purpose**: Creates a B-tree index on the specified column for faster searching.
- **Example**:
    - User creates an index on the Name column of the Employees table.

7. Searching for Data

- **Action**: getValueWhere(String tableName, String colname, String colValue) in the TableData class.
- **Purpose**: Searches for rows matching a specific value in a column.
    - If an index exists on the column, the search uses the B-tree.
    - Otherwise, the system scans the entire table.
- **Example**:
    - User searches for employees named John Doe in the Employees table.

8. Deleting Data from the Table

- **Action**: removeData(String tableName, String colname, String val) in the TableData class.
- **Purpose**: Removes rows from the table based on the specified condition.
- **Example**:
    - User deletes data where Name is John Doe from the Employees table.

9. Deleting a Table

- **Action**: deleteTables(String dbname, String tableName) in the TableList class.
- **Purpose**: Removes a table from the specified database.
- **Example**:
    - User deletes the Employees table from EmployeeDB.

10. Deleting a Database

- **Action**: deleteDatabase(String user, String databaseName) in the Database class.
- **Purpose**: Removes the entire database, including all associated tables and data.
- **Example**:
    - User deletes the EmployeeDB database.