Summary

Functional Testing Document

Scope

This document describes the functional testing process for the DatabaseManager class in the dev.uninotes.UniNotes.Database package. The goal is to ensure that each public method of the DatabaseManager performs as expected under normal and edge-case scenarios.

---

1. Test Database Connection

Objective: Verify that the connect() method establishes a connection to the SQLite database.

- **Test ID**: TDB01
- **Prerequisites**: Ensure that the SQLite database file exists at ./localdb/unibgnotes.db.
- **Steps**:
    1. Invoke DatabaseManager.connect().
    2. Check if the returned connection object is not null.
- **Expected Result**: Connection object is established and not null.
- **Successful**: yes.

---

2. Test Add User

Objective: Verify that the addUser(String email, String password) method correctly adds a user to the users table.

- **Test ID**: TDB02
- **Prerequisites**: Database initialized.
- **Steps**:
    1. Call DatabaseManager.addUser("test@example.com", "password123").
    2. Query the users table for the added user.
- **Expected Result**: The user with email test@example.com exists in the database.
- **Successful**: yes.

---

3. Test Invalid User

Objective: Ensure the validateUser(String emailOrUsername, String password) method correctly validates user credentials.

- **Test ID**: TDB03
- **Prerequisites**: A user with email thisMailDoesNotExists@example.com and password password123 does not exist in the database.
- **Steps**:
    1. Call DatabaseManager.validateUser("thisMailDoesNotExists @example.com", "password123").
    2. Check the returned boolean false.
- **Expected Result**: The method returns false.
- **Successful**: yes.

---

4. Test Update User

Objective: Ensure the UPDATE_USER() method updates user details successfully.

- **Test ID**: TDB04
- **Prerequisites**: A user with id = 1 exists in the database.
- **Steps**:
    1. Call DatabaseManager.UPDATE_USER(1, "John", "Doe", "john@example.com", "john_doe", "image.jpg").
    2. Query the users table for the updated user data.
- **Expected Result**: The user data is updated as per the provided values.
- **Successful**: yes.

---

5. Test Update User With Unavailable Username

Objective: Ensure the UPDATE_USER() method does not permit to update an user if he wants an username which is already in use.

- **Test ID**: TDB05

- **Prerequisites**: A user with id = 2 exists in the database.
- **Steps**:
    1. Call DatabaseManager.*UPDATE_USER*(2, "TestUpdate", "TestUpdate", "update2@test.com", "teeeest", "image.jpg").
    2. Query the users table.
- **Expected Result**: The user data is not updated because of the unique coinstraint.
- **Successful**: yes.

---

6. Test Update User With Unavailable Email

Objective: Ensure the UPDATE_USER() method does not permit to update an user if he wants an email which is already in use.

- **Test ID**: TDB06
- **Prerequisites**: A user with id = 2 exists in the database.
- **Steps**:
    1. Call DatabaseManager.*UPDATE_USER*(2, "TestUpdate", "TestUpdate", "update@test.com", "teeeest2", "image.jpg").
    2. Query the users table.
- **Expected Result**: The user data is not updated because of the unique coinstraint.
- **Successful**: yes.

---

7. Test Update User With Unavailable Username and Email

Objective: Ensure the UPDATE_USER() method does not permit to update an user if he wants an username and email which are already in use.

- **Test ID**: TDB07
- **Prerequisites**: A user with id = 2 exists in the database.
- **Steps**:
    1. Call DatabaseManager.*UPDATE_USER*(2, "TestUpdate", "TestUpdate", "update@test.com", "teeeest", "image.jpg").

2. Query the users table.

- **Expected Result**: The user data is not updated because of the unique coinstraints.
- **Successful**: yes.

---

8. Test Insert Post

Objective: Verify that the INSERT_POST(String text, int userId) method correctly inserts a post.

- **Test ID**: TDB08
- **Prerequisites**: A user with id = 1 exists in the database.
- **Steps**:
    1. Call DatabaseManager.INSERT_POST("This is a test post", 1).
    2. Query the posts table for the inserted post.
- **Expected Result**: The post exists in the database with the correct text and user ID.
- **Succesfull**: no.
- **What happened**: the test initialy failed, we got false even if the post had been inserted.
- **Solution**: the missing “return true” has been inserted in the method if the post was inserted.

---

9. Test SELECT_POSTS

Objective: Validate that the SELECT_POSTS() method retrieves all posts ordered by ID in descending order.

- **Test ID**: TDB09
- **Prerequisites**: Posts exist in the database.
- **Steps**:
    1. Call DatabaseManager.SELECT_POSTS().
    2. Verify the returned list contains posts ordered by ID in descending order.
- **Expected Result**: The returned list matches the expected order.
- **Successful**: yes.

---

10. Test Insert Note

Objective: Verify that the INSERT_NOTE(String description, int idUser, String course, String type) method adds a note.

- **Test ID**: TDB10
- **Prerequisites**: A user with id = 1, a course, and a note type exist in the database.
- **Steps**:
    1. Call DatabaseManager.INSERT_NOTE("Test note", 1, "Mathematics", "Lecture").
    2. Query the notes table for the inserted note.
- **Expected Result**: The note exists in the database with the correct details.
- **Successful**: yes.

---

11. Test SELECT_NOTES

Objective: Ensure the SELECT_NOTES(String field, String course, String username, String type) method retrieves the correct notes.

- **Test ID**: TDB11
- **Prerequisites**: Notes matching the query parameters exist in the database.
- **Steps**:
    1. Call DatabaseManager*SELECT_NOTES*("Computer Science", "Data Science", "john_doe", "Lecture").
    2. Verify the returned list contains the correct notes.
- **Expected Result**: The returned list matches the expected notes.
- **Successful**: yes.

---

12. Test UPDATE_USER_PASSWORD

Objective: Ensure the SELECT_NOTES(String email, String newpassword, String oldpassword) method updates the password.

- **Test ID**: TDB12
- **Prerequisites**: a user with the chosen email exist in the database.
- **Steps**:
    1. Call DatabaseManager.*UPDATE_USER_PASSWORD("test@example.com", "newpassword123", "password123");*
    2. Verify the boolean result is true.
- **Expected Result**: The returned bool is true.
- **Successful**: yes.

---

13. Test Select Faculties

- **Test ID**: TDB13
- **Objective**: Verify that the SELECT_FACULTIES_NAME() method retrieves a list of faculties.
- **Steps**:
    1. Call DatabaseManager.SELECT_FACULTIES_NAME().
    2. Check if the returned list is not null.
- **Expected Result**: The list of faculties should not be null and should contain valid faculty names.
- **Successful**: Yes

---

14. Test Select Usernames

- **Test ID**: TDB14
- **Objective**: Verify that the SELECT_USERNAMES() method retrieves a list of usernames.
- **Steps**:
    1. Call DatabaseManager.SELECT_USERNAMES().
    2. Check if the returned list is not null.
    3. Verify the list is not empty.

- **Expected Result**: The list of usernames should not be null and should contain valid usernames.
- **Successful**: Yes