**Testing Document: Project Name**

**Profile Functions**

**Module:**User Authentication and Profile Management

**Test Case 1:** User Profile Creation

**Description:** Test the createUserProfile function to ensure a new user can be created.

**Input:**

* **Name:** John Doe
* **Username**: johndoe1
* **Password:** unknown1
* **DOB:** 1993-05-15
* **Phone:** 55555555555
* **Email:** unknownperson1@yahoo.com

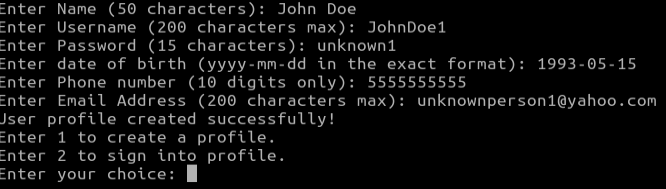
**Expected Output:** A new user record should be created in the database.

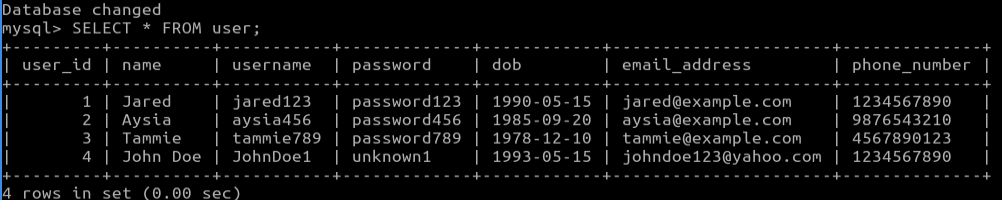
**Actual Output**:Verified the new user record is present in the database. [Actual output]

**Result:Pass**

**TEST CODE:** createUserProfile(conn), isUsernameUnique(conn,user.username), dateFormatValid(user.dob),

**Code used to conduct the above test**

****

****

**Test Case 2:** User Sign-in Authentication

**Description:** Test the signIn function to authenticate a user.

**Input:**

Username: Jared321

Password: password123

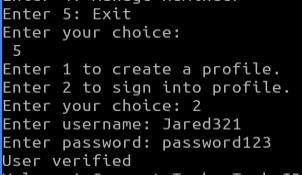
**Expected Output:** User authentication should be successful.

**Actual Output:** User 'Jared321’ successfully authenticated.

**Result:**Pass

[TEST CODE]

Code used to conduct the above test



**Test Case 3:** Username Uniqueness Check

**Description:** Verify that the isUsernameUnique function correctly identifies unique usernames.

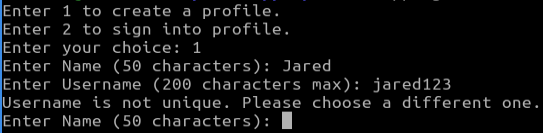
**Input:** Username: jared123

**Expected Output**:The function should return false, indicating the username is already taken.

**Actual Output**:Function returned false as expected.

**Result:**Pass

Code used to conduct the above test



**Test Case 4:** Update User Email

**Description:** Ensure the updateEmail function updates the user's email address correctly.

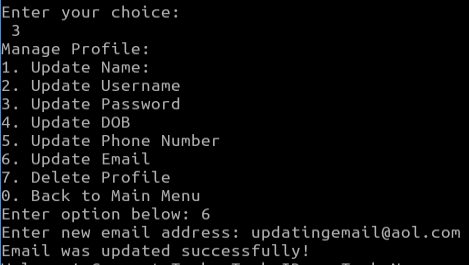
**Input:**

* User ID: 1
* New Email: [updatingemail@](mailto:updatingemail@yahoo.com)aol.com
* **Expected Output:** The email address for User ID 1 should be updated in the database.

**Actual Output:** The email address was successfully updated to updatingemail@aol.com

**Result**: Pass

**Test Code:** updateEmail(conn, current\_user);



**Test Case 5:** Delete User Profile

**Description:** Confirm that the deleteProfile function removes the user's profile from the database.

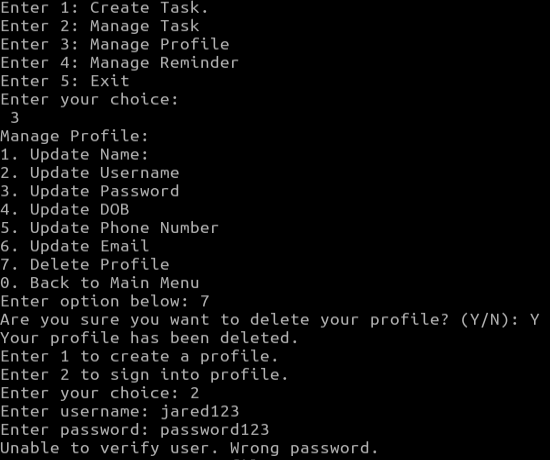
**Input:** User ID: 1

**Expected Output:** The user profile for User ID 1 should no longer exist in the database.

**Actual Output:** User profile for User ID 1 was successfully deleted.

Result: Pass

**Test Code:** deleteProfile(conn, current\_user);



**Test Case 6:** Update User Name

**Description:** Verify the updateName function correctly updates a user's name.

**Input:**

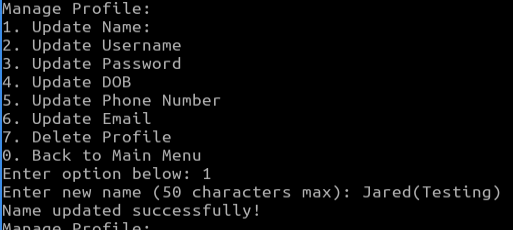
* User ID: 1
* New Name: Jared(Testing)

**Expected Output:** The name for User ID 1 should be updated in the database.

**Actual Output:** Name updated to Jared(Testing) in the database.

**Result:** Pass

**Test Code:**

****

**Test Case 7:** Update User Password

**Description:** Confirm the updatePassword function changes the user's password successfully.

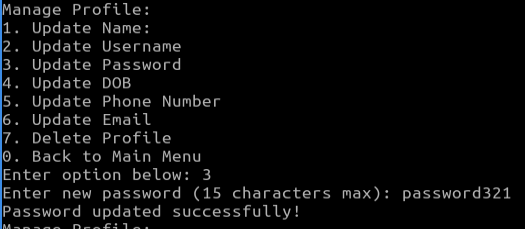
**Input:**

* User ID: 1
* New Password: password321
* **Expected Output:** The password for User ID 1 should be updated in the database.

**Actual Output:** Password updated successfully in the database.

**Result:** Pass

**Test Code:**

****

**Test Case 8**: Update User Date of Birth

**Description:** Validate the updateDOB function properly updates the user's date of birth.

**Input:**

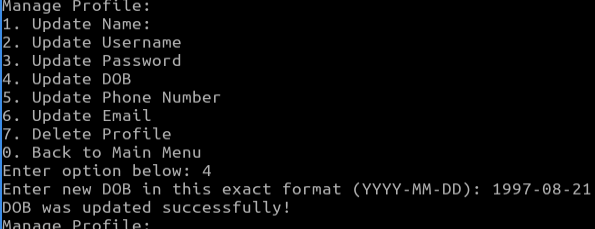
* User ID: 1
* New DOB: 1997-08-21

**Expected Output:** The date of birth for User ID 1 should be updated in the database.

**Actual Output**: Date of birth updated to 1997-08-21 in the database.

**Result**: Pass

**Test Code:**

****

**Test Case 9:** Update User Phone Number

**Description:** Test the updatePhoneNum function to ensure the user's phone number is updated.

**Input:**

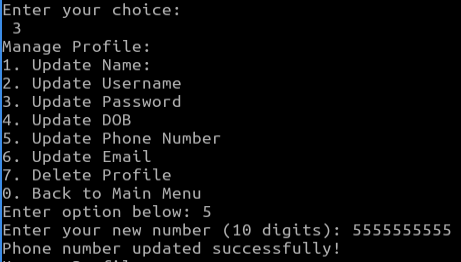
* User ID: 1
* New Phone Number: 5555555555

**Expected Output:** The phone number for User ID 1 should be updated in the database.

**Actual Output:** Phone number updated to 5555555555 in the database.

**Result:** Pass

**Test Code:**



Comments

**Module: Task Handling**

**Test Case 1:** Task Creation

**Description:** Test the createTaskForm function to ensure a new task can be created.

**Input:**

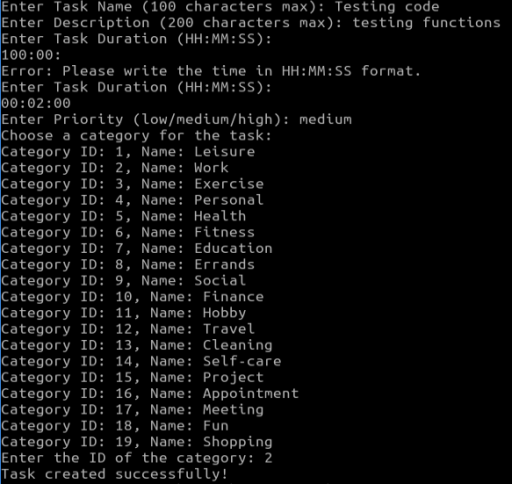
* Task Name: "Testing"
* Description: "Testing functions"
* Duration: "00:02:00”
* Priority: "medium"
* User ID: 1 (current\_user)

**Expected Output:** A new task record should be created in the database.

**Actual Output:** Verified the new task record is present in the database with the user ID 1.

**Result:** Pass

**Test Code:**

****

**Test Case 2:** Task Management

**Description:** Test the manageTask function to allow updating and deletion of tasks.

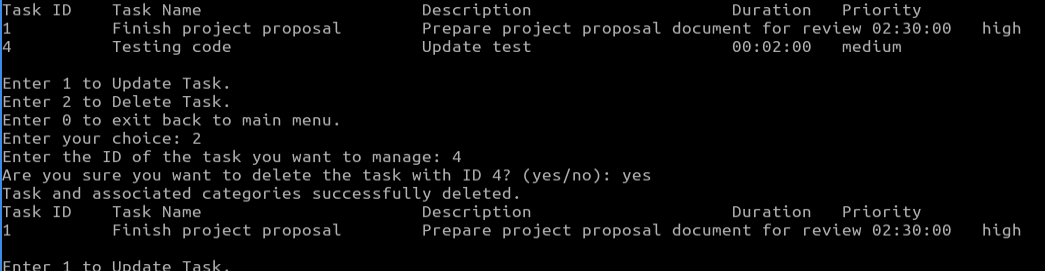
**Input:**

* User ID: 1 (current\_user)
* Task to manage: 4 “Testing code”

**Expected Output:** User should be able to update and delete their tasks.

**Actual Output:** User was able to update and delete the selected task.

**Result:** Pass

**Test Code:**

**Test Case 3:** Update Task

**Description:** Verify the updateTaskForm function correctly updates task details.

**Input:**

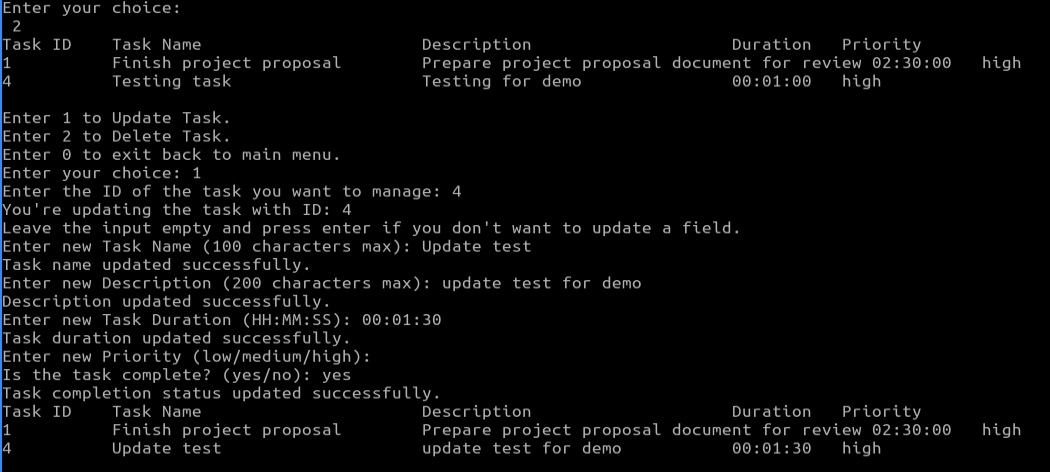
* Task ID: 1
* New Description: “Finish project proposal document for review”
* New Priority: “high”

**Expected Output:** Task with ID 1 should have its details updated in the database.

**Actual Output:** Task updated successfully with the new description and priority.

**Result:** Pass

**Test Code:**

****

**Test Case 4:** Delete Task

**Description:** Confirm the deleteTaskByID function removes a task from the database.

**Input:**

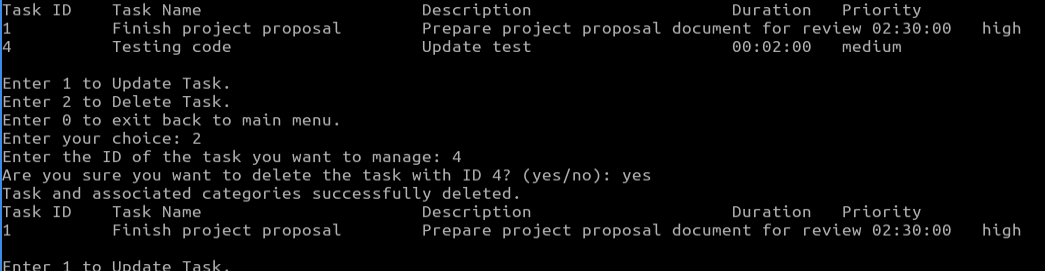
* Task ID: 1

**Expected Output:** The task with ID 1 should no longer exist in the database.

**Actual Output**: Task with ID 1 was successfully deleted from the database.

**Result:** Pass

**Test Code:**

****

**Test Case 5:** Validate Task ID

**Description:** Ensure the validateTaskID function correctly validates a task ID.

**Input:**

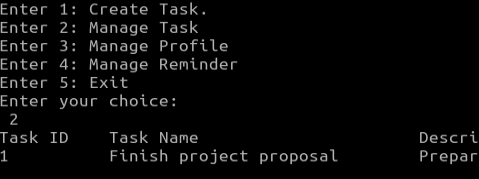
* Task ID: 1

**Expected Output:** The function should return true, indicating the task ID exists and belongs to the current user.

**Actual Output:** Function returned true, task ID 1 is valid and belongs to the current user.

**Result:** Pass

**Test Code:**

****

**Test Case 6:Description:** Test the fetchTaskById function to retrieve a task's details.

**Input:**

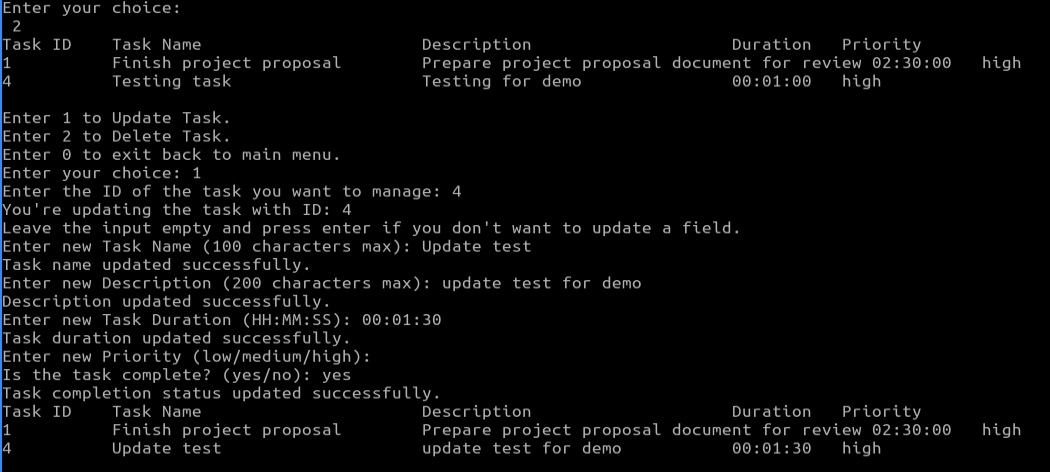
* Task ID: 1

**Expected Output**: The function should return the details of the task with ID 1.

**Actual Output:** Retrieved task details match the expected task.

**Result:** Pass

**Test Code:**

****

**Test Case 7:** Print Task Categories

**Description**: Verify that the printCategories function correctly prints all task categories.

**Input:** [No direct input, triggers function to execute]

**Expected Output:** All available categories should be listed from the database.

**Actual Output**: All categories listed successfully.

**Result:** Pass

**Test Code:**

****

**Test Case 8:** Show User Tasks

**Description:** Confirm that the showUserTasks function correctly displays all tasks for the current user.

**Input:**

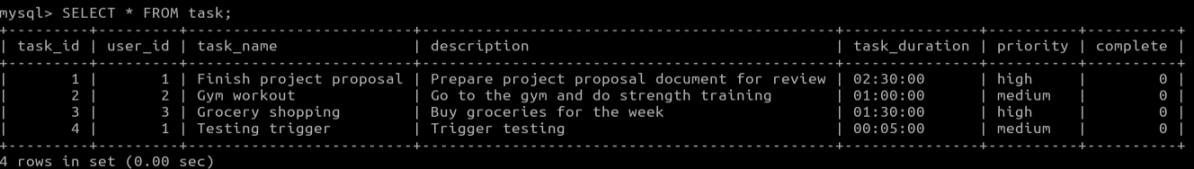
* User ID: 1 (current\_user)

**Expected Output:** All tasks associated with User ID 1 should be displayed.

**Actual Output:** All tasks for the current user were displayed correctly.

**Result:** Pass

**Test Code:**

****

**Module: Reminder Management**

**Test Case 1:** Manage Reminders Interface

**Description:** Test the manageReminders function to ensure the reminder management interface works correctly.

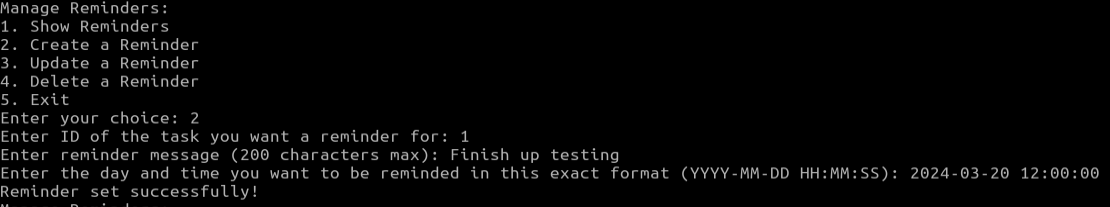
**Input:**

* User ID: 1 (current\_user)
* **Expected Output:** The function should allow the user to view, create, update, and delete reminders.

**Actual Output:** The reminder management interface displayed all options correctly, and the user could navigate through them.

**Result**: Pass

**Test Code:**

****

**Test Case 2:** Display User Reminders

**Description:** Verify the showUserReminders function correctly displays all reminders for the current user.

**Input:**

* User ID: 1 (current\_user)

**Expected Output:** All reminders associated with User ID 1 should be displayed.

**Actual Output**: The reminders for the current user were displayed correctly.

**Result:** Pass

**Test Code:**

****

**Test Case 3:** Reminder Creation

**Description:** Confirm that the createReminder function can successfully create a new reminder.

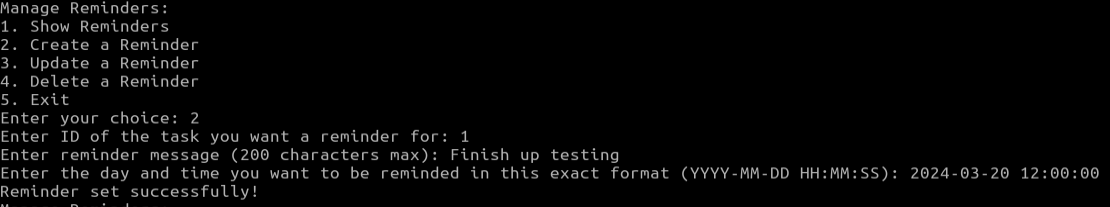
**Input:**

* User ID: 1
* Task ID: 2
* Message: "Finish up testing”
* Status: "scheduled"
* Date and Time: 2024-03-21 12:00:00
* **Expected Output:** A new reminder for the specified task should be created with the given message and scheduled time.

**Actual Output:** The new reminder was successfully created and scheduled.

**Result:** Pass

**Test Code:**

****

**Test Case 4**: Reminder Update

**Description**: Test the updateReminder function to ensure an existing reminder can be updated.

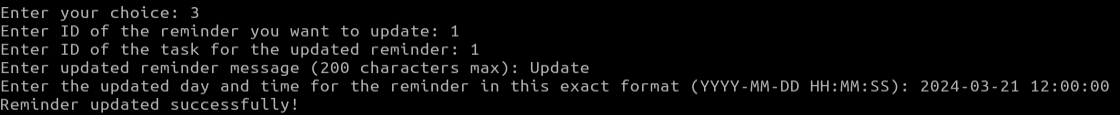
**Input:**

* Reminder ID: 1
* New Message: "Update”
* New Status: "rescheduled"
* New Date and Time: 2024-03-21 12:00:00
* **Expected Output**: The reminder with ID 1 should be updated with the new message, status, and scheduled time.

**Actual Output**: The reminder was updated with the new details as expected.

**Result:** Pass

**Test Code:**

****

**Test Case 5:** Reminder Deletion

**Description:** Verify that the deleteReminder function correctly removes a reminder from the system.

**Input:**

* Reminder ID: 1

**Expected Output:** The reminder with ID 1 should be deleted from the database.

**Actual Output:** Reminder was successfully deleted, no longer appears in the list of user reminders.

**Result:** Pass

**Test Code:**

****