Database Bugs

Test Case ID	TC_002
Test Case Name	DBM_AddUser_WithInvalidData
Test Expected Output	Invalid user data should be rejected, and the database should remain unchanged.
Actual Output	The database allows invalid users to be added in certain scenarios (e.g., negative age, invalid date, empty name).
Steps to Reproduce	1. Attempt to add a user with invalid data (e.g., negative age or empty name) using DBM_Add_User. 2. Observe that the user is added, and the database count increases.
Root Cause	Lack of proper input validation in DBM_Add_User allows invalid data to be added.
Suggested Fix	- Add robust input validation in DBM_Add_User to reject invalid data before modifying the database. - Ensure error codes or messages are provided for invalid input cases.

Test Case ID	TC_004
Test Case Name	DBM_AddUser_WithDuplicateUsername
Test Expected Output	The function should reject the addition of a user with a username that already exists in the database. The current_user_test count should remain unchanged, and the database state should not be modified.
Actual Output	The database allows adding users with duplicate usernames. The function does not reject the user, and the current_user_test count increases.
Steps to Reproduce	1. Add a user with a unique username (e.g., "EdgesAcademy") using DBM_Add_User. 2. Attempt to add another user with the same username. 3. Observe that: - The function does not return 0 (indicating rejection). - The current_user_test count increases. - The duplicate user's details are present in the database.
Root Cause	Lack of validation to check for duplicate usernames when adding new users.
Suggested Fix	1. Add a validation step in DBM_Add_User to ensure that usernames are unique. 2. Iterate through the existing users in the database and compare the User_Name field of the new user with those of existing users. 3. If a match is found, return 0 (failure) and ensure no changes to the database state.

Test Case ID	TC_006
Test Case Name	DBM_DeleteUser_WithNonExistentID
Test Expected Output	The function should reject the deletion operation and leave the database unchanged.
Actual Output	The database count is decremented, or the state is modified, when attempting to delete a nonexistent user ID.
Steps to Reproduce	1. Attempt to delete a user ID that does not exist in the database (e.g., current_user_test + 1) 2. Observe that the user count changes, or the database state is altered.
Root Cause	Lack of validation to check if the user ID exists before performing deletion.
Suggested Fix	Add checks in DBM_Delete_User to ensure the user ID exists before proceeding with the deletion.

Test Case ID	TC_009
Test Case Name	DBM_DeleteUser_FromEmptyDatabase
Test Expected Output	The function should return a failure (e.g., return 0) and leave the database unchanged when attempting to delete from an empty database.
Actual Output	The function allows a deletion operation on an empty database, which is incorrect. The database state is altered.
Steps to Reproduce	1. Clear the database by deleting all users. 2. Attempt to delete a user from the empty database using DBM_Delete_User(0). 3. Observe that the function does not reject the operation.
Root Cause	Lack of check for an empty database before allowing a deletion.
Suggested Fix	Add a condition in DBM_Delete_User to check if the database is empty (current_user_test == 0) before processing a delete operation.

Test Case ID	TC_010
Test Case Name	DBM_initDB_WithValidData
Test Expected Output	The DBM_initDB function should correctly reset the current_user_test variable to reflect the initial number of users added to the database.
Actual Output	The DBM_initDB function initializes the database but fails to correctly update the current_user_test variable, resulting in a mismatch between the expected and actual user count.
Steps to Reproduce	1. Add a user to the database to modify its state. 2. Call DBM_initDB to reset the database. 3. Observe that: - The default users are added back to the database. - The current_user_test variable does not match the number of default users.
Root Cause	The DBM_initDB function does not correctly update the current_user_test variable after resetting the database.
Suggested Fix	Update the DBM_initDB function to explicitly set the current_user_test variable to the correct value after adding the default users. This ensures consistency between the user count and the database state.

Test Case ID	TC_013
Test Case Name	DBM_AddToCourse_AlreadyEnrolled
Test Expected Output	The function should return a failure (e.g., RET = 0) and prevent further enrollment if the user is already enrolled in the course.
Actual Output	The function DBM_AddToCourse allows a user to enroll in the same course multiple times, which is incorrect. The user's enrollment state is altered even when they are already enrolled in the course.
Steps to Reproduce	1. Add a test user to the database using DBM_Add_User. 2. Enroll the user in a course using DBM_AddToCourse. 3. Attempt to enroll the same user in the same course again using DBM_AddToCourse. 4. Observe that the function does not reject the operation and allows double enrollment.
Root Cause	The function DBM_AddToCourse does not check if the user is already enrolled in the course before allowing enrollment.
Suggested Fix	Add a condition in DBM_AddToCourse to check if the user is already enrolled in the course before proceeding with the enrollment. If the user is already enrolled, return RET = 0 to prevent double enrollment.

Test Case ID	TC_015
Test Case Name	DBM_DeleteReservation_WithValidData
Test Expected Output	The function should update the Enrollments array to set the student's enrollment status to 0 (indicating the student is no longer enrolled in the course). The reservation should be successfully deleted, and the student's details should no longer reflect the enrollment in the specified course.
Actual Output	The function DBM_DeleteReservation does not properly delete a student's reservation from the course. The student's enrollment status in the Enrollments array remains unchanged as 1 (indicating the student is still enrolled), even after the reservation deletion function is called. This causes the test to fail, as the reservation is not removed from the database.
Steps to Reproduce	1. Add a test user to the system using DBM_Add_User(). 2. Enroll the user in a course using DBM_AddToCourse(). 3. Call DBM_DeleteReservation() to delete the student's reservation for the course. 4. Observe that: - The function returns 1 indicating success. - The student's enrollment status in the Enrollments array is still 1 (indicating they are still enrolled). - The reservation is not actually removed, and the student's status remains unchanged.
Root Cause	The function DBM_DeleteReservation does not properly update the Enrollments array, and the reservation deletion is not properly reflected in the database.
Suggested Fix	1. Replace the comparison operator == with the assignment operator = in the line: Enrollments[StudentID][CourseID - 1] == FALSE; Change it to: Enrollments[StudentID][CourseID - 1] = FALSE; 2. After making this correction, the Enrollments array will correctly reflect the removal of the reservation and will show 0 for the student's enrollment status.

Test Case ID	TC_016
Test Case Name	DBM_DeleteReservation_StudentNotEnrolled
Test Expected Output	1. The function should return 0 (indicating failure) when attempting to delete a reservation for a student who is not enrolled in the course. 2. The Enrollments array should remain unchanged for the student, reflecting that they are not enrolled in the course. 3. The enrollment status should be correctly updated when a reservation is deleted (using the correct assignment operator).
Actual Output	1. The function returns 1, indicating success, when attempting to delete a reservation for an unenrolled student. 2. The function uses the comparison operator == instead of the assignment operator =, meaning the enrollment status is not correctly updated. 3. The student's enrollment status in the Enrollments array is not modified, and the deletion is falsely marked as successful.
Steps to Reproduce	1. Add a test user to the database using DBM_Add_User(). 2. Ensure the test user is not enrolled in any course (e.g., Course ID 1). 3. Call DBM_DeleteReservation() to delete the student's reservation for the course they are not enrolled in. 4. Observe that: - The function returns 1, indicating success. - The Enrollments array does not change, but the deletion should not have been successful for an unenrolled student. - The enrollment status for the student is not updated due to the incorrect use of the == operator instead of =.
Root Cause	The function DBM_DeleteReservation does not validate if the student is enrolled in the course before proceeding with the deletion. Additionally, the comparison operator == is used instead of the assignment operator =, which causes incorrect behavior in updating the enrollment status.
Suggested Fix	1. Add validation in DBM_DeleteReservation to check if the student is actually enrolled in the course. - If the student is not enrolled (Enrollments[StudentID][CourseID - 1] != TRUE), return 0 (failure). 2. Correct the assignment operator from == to = in the following line: Enrollments[StudentID][CourseID - 1] = FALSE; This ensures that the enrollment status is correctly updated when a reservation is deleted for an enrolled student.

Test Case ID	TC_017
Test Case Name	DBM_DeleteReservation_NonExistentCourse
Test Expected Output	1. The function should return 0 (indicating failure) when attempting to delete a reservation for a non-existent course. 2. The Enrollments array should remain unchanged, reflecting that no deletion was made.
Actual Output	1. The function returns 1, indicating success, when attempting to delete a reservation for a non-existent course. 2. The Enrollments array is unchanged, but the deletion should not have been successful for a non-existent course.
Steps to Reproduce	1. Add a test user to the database using DBM_Add_User(). 2. Attempt to delete a reservation for a non-existent course (e.g., CourseID = 100). 3. Observe that: - The function returns 1, indicating success. - The Enrollments array is unchanged, but the deletion should not have been successful for a non-existent course.
Root Cause	The function DBM_DeleteReservation does not validate if the course exists before proceeding with the deletion, leading to incorrect success status and unchanged enrollment data.
Suggested Fix	1. Add a course ID validation in the DBM_DeleteReservation function to ensure that the course exists before processing the deletion. 2. If the course ID is invalid (i.e., out of range), the function should return 0 (failure) without modifying the Enrollments array.

Test Case ID	TC_018
Test Case Name	DBM_DeleteReservation_InvalidStudentID
Test Expected Output	1. The function should return 0 (failure) when an invalid StudentID is provided. 2. The Enrollments array should not be altered for an invalid user ID.
Actual Output	1. The function returns 1, which is incorrect for an invalid StudentID. 2. The Enrollments array might incorrectly show changes for the invalid student.
Steps to Reproduce	1. Add a valid test user using DBM_Add_User(). 2. Set up an invalid StudentID (e.g., one that is out of bounds). 3. Attempt to delete a reservation for the non-existent student. 4. Observe the following: - The function returns 1, which is incorrect for an invalid StudentID. - The Enrollments array might incorrectly show changes for the invalid student.
Root Cause	The function DBM_DeleteReservation does not check if the StudentID is valid before attempting to modify the Enrollments array, leading to unpredictable behavior.
Suggested Fix	1. Add a check at the beginning of DBM_DeleteReservation to ensure that StudentID is within valid bounds (i.e., the student exists). 2. If the StudentID is invalid, return 0 immediately and do not attempt to modify Enrollments.

Test Case ID	TC_019
Test Case Name	DBM_DeleteReservation_NoEnrollments
Test Expected Output	1. The function should return 0 (failure) or Error when attempting to delete a reservation for a course with no enrolled students. 2. The Enrollments array should remain unchanged, reflecting that no deletion was made.
Actual Output	1. The function returns TRUE instead of 0. 2. The Enrollments array is unchanged, but the deletion should not have been successful.
Steps to Reproduce	1. Add a test user to the system using DBM_Add_User(). 2. Ensure that the course has no enrolled students. 3. Call DBM_DeleteReservation() to delete the student's reservation for the course with no enrollments. 4. Observe that: - The function returns TRUE instead of 0. - The Enrollments array is unchanged, but the deletion should not have been successful.
Root Cause	The DBM_DeleteReservation function does not check if the course has enrolled students before processing the deletion, allowing it to incorrectly return TRUE for a non-existent enrollment.
Suggested Fix	Change the return value for courses with no enrollments to 0 (failure) in the DBM_DeleteReservation function. This ensures the function behaves as expected when attempting to delete from an empty course.

Create Account Bugs

Test Case ID	TC_028
Test Case Name	Add_Account_Same_Username
Test Expected Output	The second user should not be allowed to register with the same username. The database should reject this operation.
Actual Output	The second user is added to the database with the same username.
Steps to Reproduce	1. Start the software. 2. Initialize the database (make sure it is empty). 3. Add the first user with a valid username "User1234". 4. Add the second user with the same username "User1234". 5. Check the database to see if both users are added.
Root Cause	The source code does not implement a check for the uniqueness of the username before adding a new user to the database.
Suggested Fix	Implement a check in the Add_Account function to verify if the username already exists in the database before adding a new user. If the username exists, the function should return FALSE and prevent adding the duplicate user.

Test Case ID	TC_032
Test Case Name	Delete_Account_SameUser
Test Description	Testing delete Account functionality with the same user ID after deletion
Test Expected Output	The first deletion should succeed, and the second should fail as the user doesn't exist anymore
Actual Output	The second deletion passes, indicating the user deletion logic does not handle subsequent deletions
Steps to Reproduce	1. Add a user to the database. 2. Delete the user once. 3. Try deleting the same user again.
Root Cause	The Delete_Account function does not check if the user has already been deleted, allowing the same ID to be deleted multiple times.
Suggested Fix	Add a check to ensure the user exists before attempting deletion, or track the deletion status of users in the database.

Course Registration Bugs

Test Case ID	TC_035
Test Case Name	AddStudentToCourse_InvalidCourseID
Test Description	Tests registering a valid user to a non-existent course (invalid course ID).
Test Expected Output	The user should not be registered to the course, and the result should indicate failure.
Actual Output	The registration failed as expected, but the result did not reflect an appropriate error code.
Steps to Reproduce	1. Add a valid user to the database. 2. Attempt to register the user to a non-existent course (Course ID = MAX_COURSES + 1). 3. Check the result and enrollments.
Root Cause	The system does not handle invalid course IDs correctly. It should have returned an error specific to an invalid course ID.
Suggested Fix	Implement a check in the AddStudentToCourse function to return an error code for invalid course IDs (greater than MAX_COURSES).

Test Case ID	TC_036
Test Case Name	AddStudentToCourse_InvalidStudentID
Test Description	Tests registering an invalid student to a valid course.
Test Expected Output	The user should not be registered to the course, and the result should indicate failure.
Actual Output	The registration failed as expected, but the result did not properly handle the invalid student ID.
Steps to Reproduce	1. Add a valid user to the database. 2. Assign an invalid user ID (MAX_USERS + 1). 3. Attempt to register the user to a valid course (Course ID = 1).
Root Cause	The system does not check whether the Student_id is valid (within the bounds of MAX_USERS). This results in out-of-bounds access when using an invalid Student_id.
Suggested Fix	Implement a check to ensure Student_id is within the valid range (< MAX_USERS) in the AddStudentToCourse function.
Severity	High (The bug leads to undefined behavior and memory issues when an invalid student ID is provided.)

Backend Bug

Test Case ID	TC_062
Test Case Name	BackEnd_CustomerRunner_Login_PasswordIncorrect_ThirdTrial
Test Description	This test verifies that the system correctly handles the third incorrect password attempt.
Test Expected Output	- After three incorrect password attempts: - "Your session is Terminated" - "User Will Be Deleted, Contact Admin to return it" - The user count should be decremented by 1 in the database.
Actual Output	- "Your session is Terminated" - "User Will Be Deleted, Contact Admin to return it" - The user count is **not** decremented due to a bug in the DBM_Delete_User function.
Steps to Reproduce	1. Add a valid test user to the database. 2. Attempt to log in with incorrect passwords for three consecutive trials. 3. Assert that the system displays "Your session is Terminated" and indicates the user will be deleted. 4. Verify that the user count in the database is decremented.
Root Cause	The issue lies within the DBM_Delete_User function. It fails to correctly delete the user from the database after the third incorrect password attempt.
Suggested Fix	Review and fix the logic in the DBM_Delete_User function to ensure the user is properly removed from the database after the third failed login attempt.
Severity	High - The user deletion mechanism is crucial for the correct flow of the application and impacts the overall functionality of the login process.