Student Manager Application.

Test Plan.

Introduction

This is a Student Manager application. It is designed to keep records of students attending a given school. It is capable of adding, deleting, and updating student records, as well as being queried for a selected student's or all student's information. Records can be queried by Student ID, as well as by First Name and Last Name. If there are more than one records with the same First Name and Last Name all records with the given First Name and Last Name are shown with the prompt to select the correct record by Student ID. Button List All displays all records.

The application allows to differentiate between types of students (i.e. Undergraduate, Graduate, Part-time), student status (Resident, Non-resident), and student level (Freshman, Sophomore, Junior, Senior).

The application also allows to calculate the tuition based on the number of credit hours.

Overview Of Testing Strategy.

Black box testing implies testing an application without knowledge of its inner workings. In other words testing it from the point of view of the user. It is done through encapsulation and abstraction. In the case of this particular application the GUI screen plays the role of the black box. The user doesn't need to know how the record gets into the database or gets retrieved from it. All he/she needs to know is which button to press in order to achieve the expected result.

White box testing on the other hand uses information about the structure of the program. It's done in part by running unit tests, which verify that the class will work correctly in isolation. However even if the application passes all test cases white box testing cannot guarantee that the program will run correctly for the user. "Testing can show only the presence of bugs—not their absence" (Edsger Dijkstra).

Integration testing occurs after unit testing and is a technique, in which separate modules are combined and tested as a group. The goal of integration testing is to verify functional, performance, and reliability requirements to major design items are met.

For this particular application - considering its relative simplicity - I chose black box testing. I am using the application as a user would use it to verify it works as intended.

List Of Features To Test

1. Add Undergraduate Resident Freshman (all default dropdown values and Add button).

2. Clear (use Reset button).

3. Add Undergraduate Non-resident Freshman (use Status dropdown list and Add button).

4. Add Undergraduate Resident Junior (use Level dropdown list and Add button).

5. Add Graduate Resident (use Student Type dropdown list and Add button).

6. Add Graduate Non-resident (use Student Type and Status dropdown lists and Add button).

7. Add Undergraduate Non-resident Senior (use Status and Level dropdown lists and Add button).

8. Find Student ID 1 (use Find button).

9. Find Student ID 2 (make sure the dropdown window changes accordingly).

10. Find Student ID 5 (make sure the dropdown windows change accordingly).

11. Find Student ID 6 (make sure the dropdown windows change accordingly).

12. List all records (use List All button).

13. Find Student ID 3. Enter "12" in the Credit Hours window and press Calculate Tuition button.

14. Test Remove button. Find student ID 4 and press Remove. See the prompt confirming delete.

15. Find Student ID 4 (which no longer exists). Make sure the error prompt is showing.

16. List all. Enter the first and last name of the Student ID 5 and press Find.

16. List all and add a student with the same first and last name as Student ID 1 (but different gpa, Student Type, and Status).

17. Reset. Enter the first and last name of Student ID and press Find. See that both records with the same name list in the table on the bottom and the prompt shows asking to choose the student by ID.

|  |
| --- |
| TC Number 1  Test Add functionality and Reset functionality. |
| Requirement to be tested  Add with default dropdowns, Add with other values from dropdown values. |
| Preconditions: conditions that must be present before test case can successfully run  The application needs to be connected to the MySQL database and the GUT\_Task4\_Registrar\_Attachment.sql script must be run |
| Steps: The steps the tester must execute to test the feature.   1. Add Undergraduate Resident Freshman (all default dropdown values and Add button). 2. Clear (use Reset button). 3. Add Undergraduate Non-resident Freshman (use Status dropdown list and Add button). 4. Add Undergraduate Resident Junior (use Level dropdown list and Add button). 5. Add Graduate Resident (use Student Type dropdown list and Add button). 6. Add Graduate Non-resident (use Student Type and Status dropdown lists and Add button). 7. Add Undergraduate Non-resident Senior (use Status and Level dropdown lists and Add button). |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc.  6 Student ID's created. After each add a prompt shows confirming the add. When Reset is pressed the screen clears of entries. |
| Pass/Fail: Mark whether the test case passed or failed. The results can be compiled and used to determine if the application is ready for delivery/release.  Pass. |

|  |
| --- |
| TC Number 2  Test Find functionality. |
| Requirement to be tested  Query database by Student ID. |
| Preconditions: conditions that must be present before test case can successfully run  The application needs to be connected to the MySQL database and the GUT\_Task4\_Registrar\_Attachment.sql script must be run |
| Steps: The steps the tester must execute to test the feature.  1. Find Student ID 1 (use Find button).  2. Find Student ID 2 (make sure the Student Type dropdown changes to Non-resident).  3. Find Student ID 5 (make sure the Student Type dropdown changes to Graduate and Status dropdown changes to Non-resident).  4. Find Student ID 6 (make sure the Status dropdown changes to Non-resident and Level dropdown changes to Senior). |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc.  The application displays the information for the selected Student ID's including what changing the dropdown boxes as described above. |
| Pass/Fail: Mark whether the test case passed or failed. The results can be compiled and used to determine if the application is ready for delivery/release.  Pass. |

|  |
| --- |
| TC Number 3  Test list all records functionality. |
| Requirement to be tested  Displaying all records at once. |
| Preconditions: conditions that must be present before test case can successfully run  The application needs to be connected to the MySQL database and the GUT\_Task4\_Registrar\_Attachment.sql script must be run |
| Steps: The steps the tester must execute to test the feature.  Press List All button |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc.  The information for all existing records must show in the table at the bottom of the GUI screen. |
| Pass/Fail: Mark whether the test case passed or failed. The results can be compiled and used to determine if the application is ready for delivery/release.  Pass. |

|  |
| --- |
| TC Number 4  Test Calculate Tuition functionality. |
| Requirement to be tested  Application must calculate tuition based on credit hours entered, as well as based Student Type and Status . |
| Preconditions: conditions that must be present before test case can successfully run  The application needs to be connected to the MySQL database and the GUT\_Task4\_Registrar\_Attachment.sql script must be run |
| Steps: The steps the tester must execute to test the feature.  1. Find Student ID 1.  2. Enter 12 in Credit Hours.  3. Press Calculate Tuition.  4. See prompt with $2400.  5. Find Student ID 5 (Graduate Non-resident)  6. Enter 12 in Credit Hours.  7. Press Calculate Tuition  8. See prompt with $4200. |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc.  A prompt with appropriate calculated tuition amounts for the selected Student Type and Status. |
| Pass/Fail: Mark whether the test case passed or failed. The results can be compiled and used to determine if the application is ready for delivery/release.  Pass. |

|  |
| --- |
| TC Number 5  Test Remove functionality. |
| Requirement to be tested  The ability to remove a selected record. |
| Preconditions: conditions that must be present before test case can successfully run  The application needs to be connected to the MySQL database and the GUT\_Task4\_Registrar\_Attachment.sql script must be run |
| Steps: The steps the tester must execute to test the feature.  1. Find Student ID 4.  2. Press Remove button.  3. See the prompt confirming delete.  4. Enter 4 in Student ID box.  5. Press Find.  6. Confirm that the error message displays.  7. Press List All button.  8. Confirm that the Student ID 4 is no longer in the database. |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc.  The record for the Student ID 4 gets removed from the database. |
| Pass/Fail: Mark whether the test case passed or failed. The results can be compiled and used to determine if the application is ready for delivery/release.  Pass. |

|  |
| --- |
| TC Number 6  Find by First Name and Last Name functionality (not required) |
| Requirement to be tested  Optional - ability to find students by First Name and Last Name. Also, if there are more than one student with the same name, display all records with the same name and prompt to choose one by Student ID. |
| Preconditions: conditions that must be present before test case can successfully run  The application needs to be connected to the MySQL database and the GUT\_Task4\_Registrar\_Attachment.sql script must be run |
| Steps: The steps the tester must execute to test the feature.  1. List all and add a student with the same first and last name as Student ID 1 (but different gpa, Student Type, and Status).  2. Reset.  3. Enter the first and last name of Student ID and press Find.  4. See that both records with the same name list in the table on the bottom and the prompt shows asking to choose the student by ID. |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc.  Both records with the same name are displayed in the table at the bottom of the screen and the prompt appears asking to select one record by Student ID. |
| Pass/Fail: Mark whether the test case passed or failed. The results can be compiled and used to determine if the application is ready for delivery/release.  Pass. |