Unit Test Report

Software testing final project

AZIZBEK RASULMETOV – 201953082052

04/06/2022

Table of Contents

1. Introduction

1.1 Purpose

Through writing this test report, it is expected to verify and validate that each unit in the software works as intended/designed by the developer. This test report is a document that records data obtained from the test, describes the environmental or operating conditions, and shows the comparison of test results with test objectives.

1.2 Background

This document is the software test report of SMS software project designed by AZIZBEK RASULMETOV. It contains the results of tests, which were executed during the testing.

1.3 Definition

Student ID: Unique ID of students given by their university.

1.4 Reference

The Internet

[1] White Box Testing, Software Testing Fundamentals, accessed 29 May 2020, <http://softwaretestingfundamentals.com/white-box-testing/>

[2] Jain, Mahak, GeeksforGeeks, accessed 29 May 2020, <https://www.geeksforgeeks.org/differences-between-black-box-testing-vs-white-box-testing/>

2. Test Overview

2.1 Test Object

The test object for the unit testing is a simple student management application. Made and designed using IntelliJ IDE Ultimate version software in Java language and it uses Java DB as the databases.

2.2 Test Time

The tests are performed from June 4th through June 6st of 2022.

2.3 Test Methods

The method used in this test is called the white box testing (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing). This method is named so because in the eyes of the tester, the software program is like a white/transparent box, where one can see clearly what's inside.

White box testing is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester [2]. The tester, usually a developer as well, chooses inputs to traverse the paths of the code and determines the appropriate outputs. Knowledge of programming and implementation is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system.

One of the advantages of using white box testing is the test is more comprehensive and can cover most paths. But because the testing can be very complex, it requires highly skilled resources and sufficient programming and implementation knowledge.

Definition by ISTQB (International Software Testing Qualifications Board):

- White-box testing: Testing based on an analysis of the internal structure of the component or system.
- White-box test design technique: Procedure to derive and/or select test cases based on an analysis of the internal structure of a component or system. [1]

2.4 Test Constraint

The overall test constraints for this test are listed below:

1. The test report is based on the tested software version.

2. All tests are based on the same test environment as the development environment (including the operating system, database, and etc.);

2.5 Testers

The tester is also the author of this document and the developer of this project.

3. Test Environment

3.1 Test Hardware Environment

Test Environment	Device	Processor	System Type	Memory	External Storage
Host development environment	Laptop computer	AMD RYZEN 5 Compute Cores 4C	64-bit Operating System, x64-based processor	8GB	None

Table 1 Test hardware environment

3.2 Test Software Environment

Software	Product Name
Operating System	Windows 10 Home Single Language
Programming Language	Java
Software Development Environment	Java version "1.8.0_241" Java(TM) SE Runtime Environment (build 1.8.0_241-b07) Java HotSpot(TM) 64-Bit Server VM (build 25.241-b07, mixed mode)
Database	Java DB

Table 2 Test software environment

4. Test Situation

4.1 Test Content

The test is mainly carried out to check database dao operations using unit test. The test includes userDAO methos such as findUser(), addUser(), updateUser(), findAllUsers(), deleteUsers() as well as studentDAO methods such as addStudent(), updateStudent(), findAllStudents(), deleteStudents().

Registered Users in database

1. Username: admin, Password: 111111
2. Username: aziz, Password: 123456

The table below lists the test cases for find user dao method:

Test Case ID	Test Content	Execution times	Test Results
Find User-1	Find user for username: “admin“ should return user data	5	Achieve the expected result
Find User – 2	Find user for not existing username “aswqw12” returns null	5	Achieve the expected result

Table 3 Find User test cases

The table below lists the test cases for adding user

Test Case ID	Test Content	Execution times	Test Results
Add User – 1	Add User with new username: “arasulmetov” and password: “123456” should return true	5	Achieve the expected result
Add User – 1	Attempt to add user with already existing username: “admin and	5	Achieve the expected result

	password: “111111” throws exception		

Table 4 Add User test cases

The table below lists the test case for find all users

Test Case ID	Test Content	Execution times	Test Results
Find All Users – 1	Attempt to find all users for given username: “admin” and password: “111111” returns true	5	Achieve the expected result

Table 5 Find All Users test case

The table below lists the test case for update user

Test Case ID	Test Content	Execution times	Test Results
Update User – 1	Attempt to update user’ password with username: “testUpdateUser” and password: “123456” to password: “111111” returns true	5	Achieve the expected result

Table 6 Update User test case

The table below lists the test case for delete user

Test Case ID	Test Content	Execution times	Test Results

Delete User -1	Attempt to delete user with username: “testDeleteUser” and password: “123456” returns true	5	Achieve the expected result

Table 7 Delete User test case

The table below lists the test cases for add student

Test Case ID	Test Content	Execution times	Test Results
Add Student -1	Attempt to add student with student id: “20195308205213”, name: “Azizbek”, and major: “CST” returns true	1	Achieve the expected result
Add Student -2	Attempt to add existing student with student id: “20195308205213”, name: “Azizbek”, and major: “CST” throws exception	1	Achieve the expected result

Table 8 Add Student test cases

The table below lists the test case on update student

Test Case ID	Test Content	Execution times	Test Results
Update Student -1	Attempt to update student with student id: “201923123123213” and name: “TestStudent” and major: “TestMajor” to name: “TestStudentUpdated” and major: “TestMajorUpdated” returns true	1	Achieve the expected result

Table 9 Update Student test case

The table below lists the test case for find all students

Test Case ID	Test Content	Execution times	Test Results
Find All Students - 1	Attempt to find all students for student id: “201953082052” and name: “Azizbek” returns more than 0 row.	1	Achieve the expected result

Table 10 Find All Student test case

The table below lists the test case for delete student

Test Case ID	Test Content	Execution times	Test Results
Delete Student - 1	Attempt to delete student returns true	1	Achieve the expected result

4.2 Test Completion Situation

Test Task Name	Content	Progress (% Complete)
Test plans, writing test case	Find resources for the different formats used to display	100%
Prepare test data and environment	See test cases	100%
Execute function test, fill up test data	See test cases	100%
Sort out test data, writing test report	Sorting out test data to write test report	100%

Table 11 Test completion situation

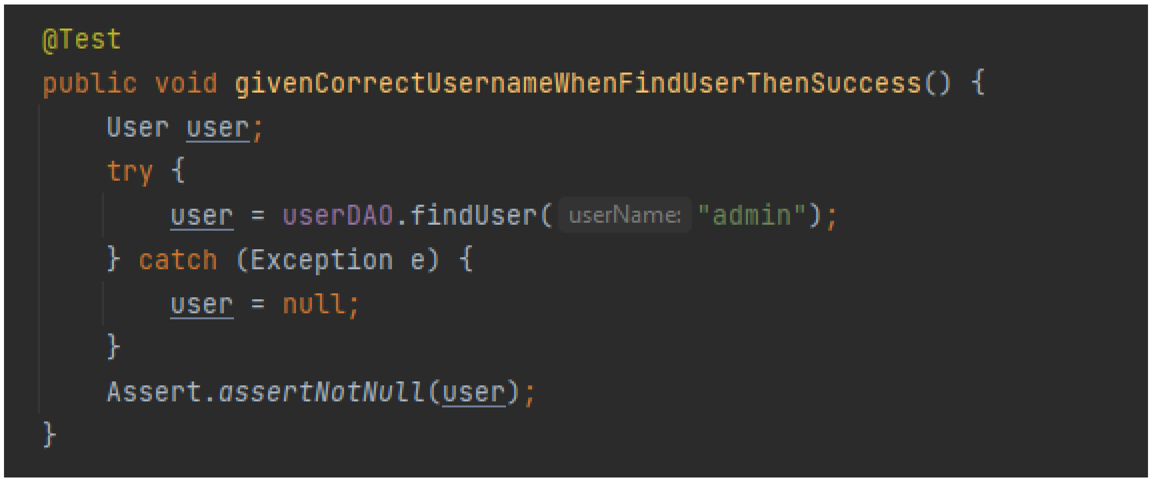
5. Test Result and Analysis

The test cases written in the previous chapter are used to test functions of the software/system/application. It is divided into 4 sub-functions which are add, find, update, delete, and each sub-function is analyzed in detail according to the test results.

5.1 Find User

5.1.1 Find User Instructions

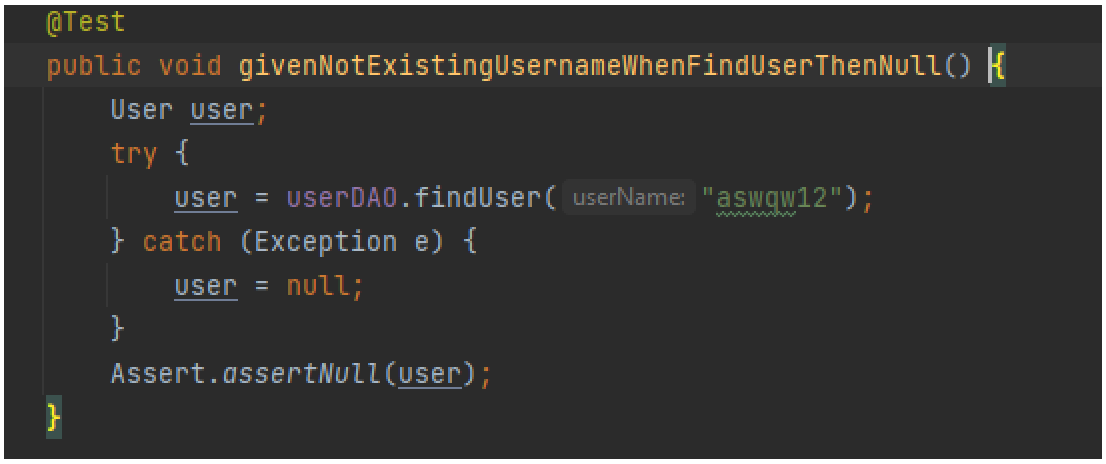
To find user from database we need to provide username which should return a user for success or null otherwise.



```
@Test
public void givenCorrectUsernameWhenFindUserThenSuccess() {
    User user;
    try {
        user = userDAO.findUser( userName: "admin");
    } catch (Exception e) {
        user = null;
    }
    Assert.assertNotNull(user);
}
```

5.1.2 Invalid Test Cases

The test fails if the username provided does not exist in our database

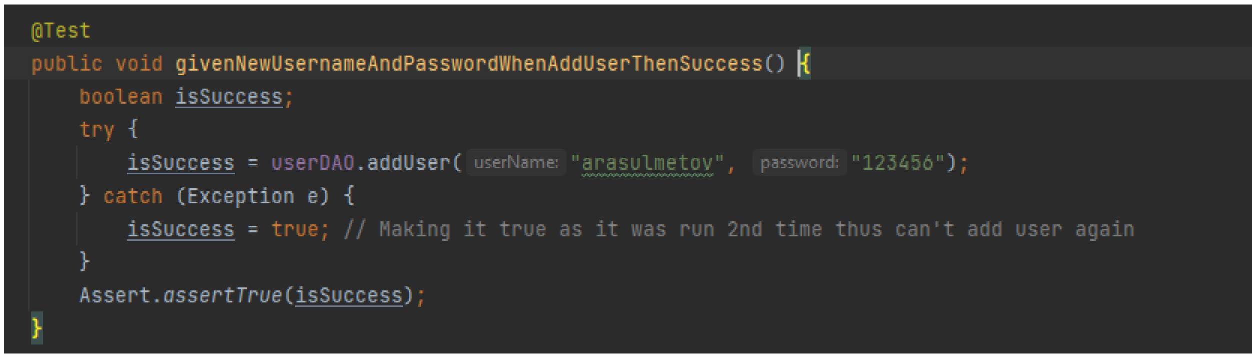


```
@Test
public void givenNotExistingUsernameWhenFindUserThenNull() {
    User user;
    try {
        user = userDAO.findUser( userName: "aswqw12");
    } catch (Exception e) {
        user = null;
    }
    Assert.assertNull(user);
}
```

5.2 Add User

5.2.1 Add User Instructions

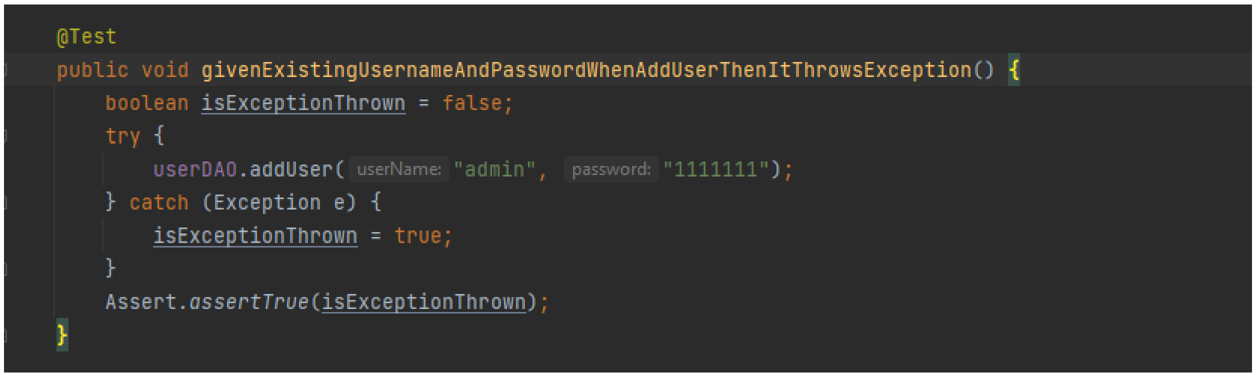
To add the user we need to have unique username and password for that user. If user is added successfully then it returns true or false for failure and may throw SQL exception if database has the user with this username.



```
@Test
public void givenNewUsernameAndPasswordWhenAddUserThenSuccess() {
    boolean isSuccess;
    try {
        isSuccess = userDAO.addUser( userName: "arasulmetov",  password: "123456");
    } catch (Exception e) {
        isSuccess = true; // Making it true as it was run 2nd time thus can't add user again
    }
    Assert.assertTrue(isSuccess);
}
```

5.2.2 Invalid Test Cases

The test fails if the username provided already exists in our database

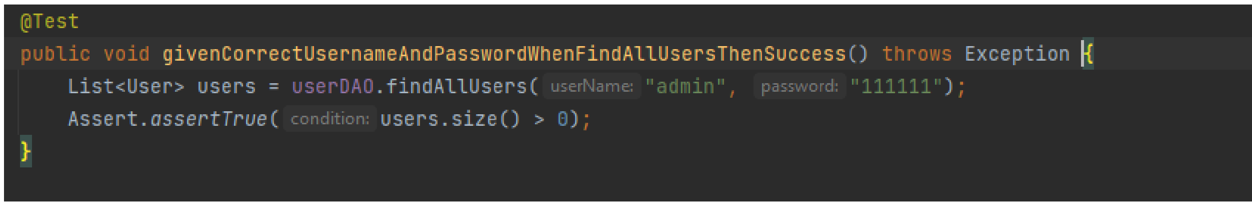


```
@Test
public void givenExistingUsernameAndPasswordWhenAddUserThenItThrowsException() {
    boolean isExceptionThrown = false;
    try {
        userDAO.addUser( userName: "admin",  password: "1111111");
    } catch (Exception e) {
        isExceptionThrown = true;
    }
    Assert.assertTrue(isExceptionThrown);
}
```

5.3 Find All Users

5.3.1 Find All Users Instructions

To find all users we need to provide username and password which in turn returns a list of users.

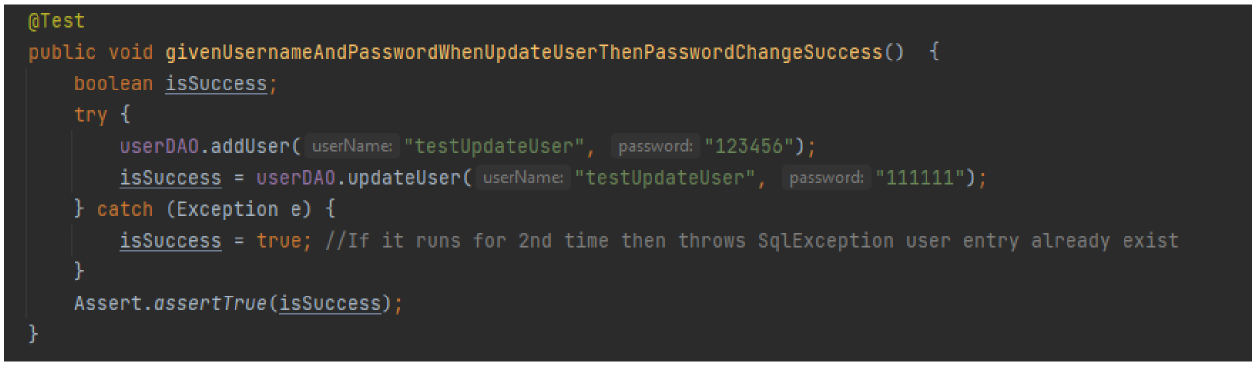


```
@Test
public void givenCorrectUsernameAndPasswordWhenFindAllUsersThenSuccess() throws Exception {
    List<User> users = userDAO.findAllUsers( userName: "admin",  password: "111111");
    Assert.assertTrue( condition: users.size() > 0);
}
```

5.4 Update User

5.4.1 Update User Instructions

To update a user we provide username and password which in turn returns true for success and may throw SQL exception.

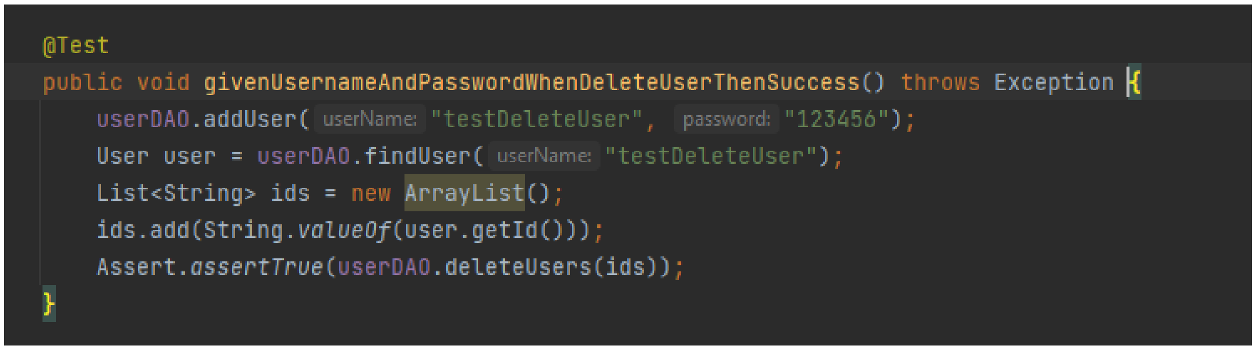


```
@Test
public void givenUsernameAndPasswordWhenUpdateUserThenPasswordChangeSuccess()  {
    boolean isSuccess;
    try {
        userDAO.addUser( userName: "testUpdateUser",  password: "123456");
        isSuccess = userDAO.updateUser( userName: "testUpdateUser",  password: "111111");
    } catch (Exception e) {
        isSuccess = true; //If it runs for 2nd time then throws SqlException user entry already exist
    }
    Assert.assertTrue(isSuccess);
}
```

5.5 Delete User

5.5.1 Delete User Instructions

To delete a user we should provide username and password and It returns true for success or false if it fails.

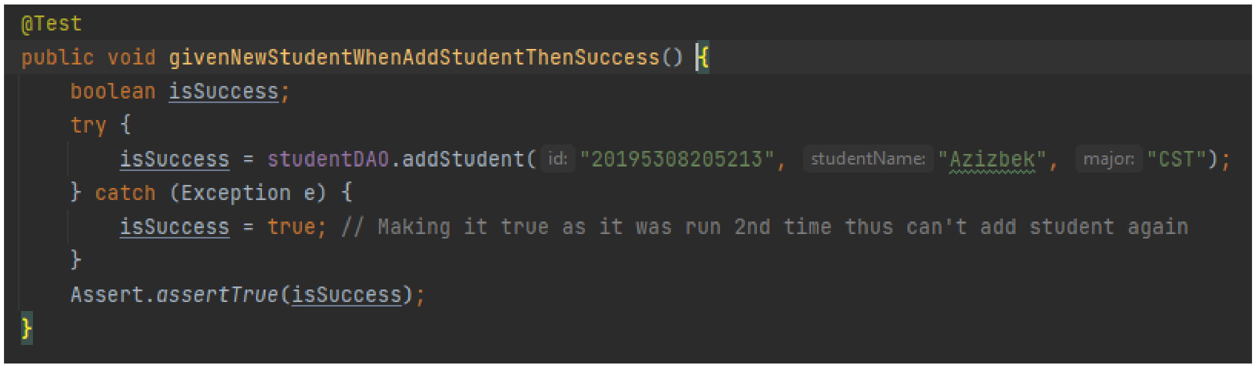


```
@Test
public void givenUsernameAndPasswordWhenDeleteUserThenSuccess() throws Exception {
    userDAO.addUser( userName: "testDeleteUser",  password: "123456");
    User user = userDAO.findUser( userName: "testDeleteUser");
    List<String> ids = new ArrayList();
    ids.add(String.valueOf(user.getId()));
    Assert.assertTrue(userDAO.deleteUsers(ids));
}
```

5.6 Add Student

5.6.1 Add Student Instructions

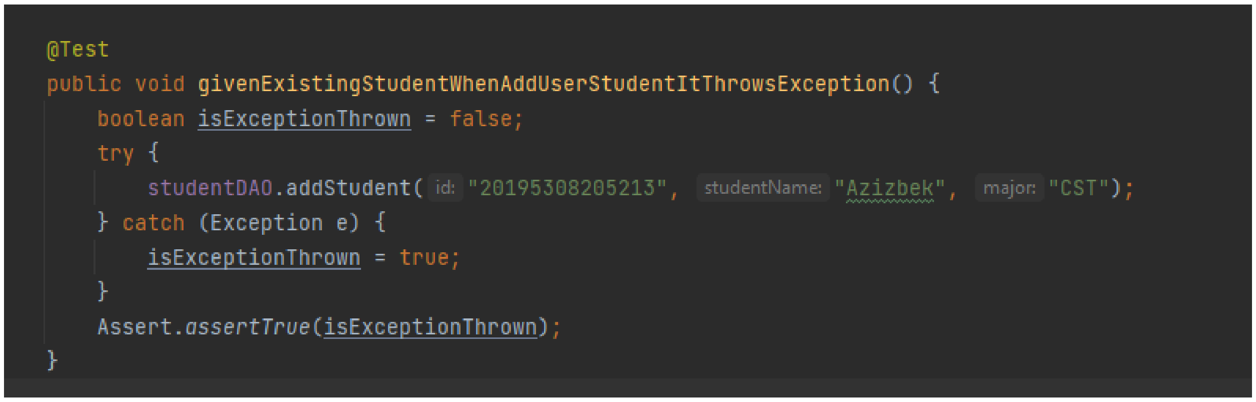
To add student we need student id, student name and major. If is added successfully then it returns true



```
@Test
public void givenNewStudentWhenAddStudentThenSuccess() {
    boolean isSuccess;
    try {
        isSuccess = studentDAO.addStudent( id: "20195308205213", studentName: "Azizbek", major: "CST");
    } catch (Exception e) {
        isSuccess = true; // Making it true as it was run 2nd time thus can't add student again
    }
    Assert.assertTrue(isSuccess);
}
```

5.6.2 Invalid Test Cases

The test fails if the student name provided already exists in our database

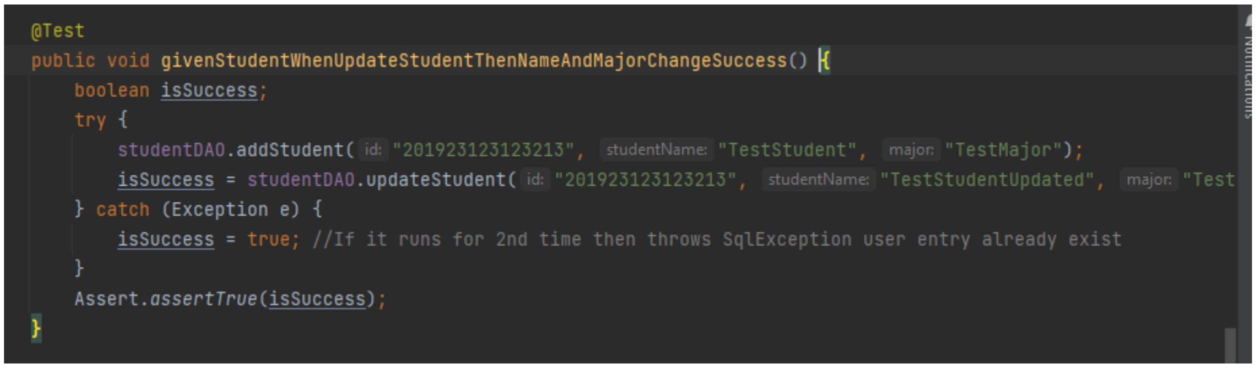


```
@Test
public void givenExistingStudentWhenAddUserStudentItThrowsException() {
    boolean isExceptionThrown = false;
    try {
        studentDAO.addStudent( id: "20195308205213", studentName: "Azizbek", major: "CST");
    } catch (Exception e) {
        isExceptionThrown = true;
    }
    Assert.assertTrue(isExceptionThrown);
}
```

5.7 Update Student

5.7.1 Update Student Instructions

To update a student we provide student name, id and major and which in turn returns true for success and may throw SQL exception.

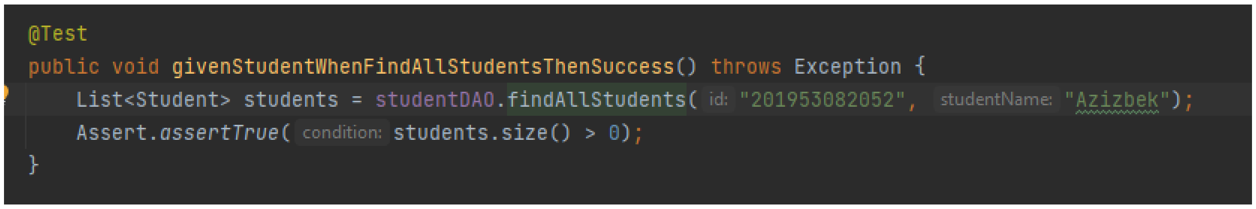


```
@Test
public void givenStudentWhenUpdateStudentThenNameAndMajorChangeSuccess() {
    boolean isSuccess;
    try {
        studentDAO.addStudent( id: "201923123123213", studentName: "TestStudent", major: "TestMajor");
        isSuccess = studentDAO.updateStudent( id: "201923123123213", studentName: "TestStudentUpdated", major: "Test
    } catch (Exception e) {
        isSuccess = true; //If it runs for 2nd time then throws SqlException user entry already exist
    }
    Assert.assertTrue(isSuccess);
}
```

5.8 Find All Students

5.8.1 Find All Students Instructions

To find all students we should provide student id and student name which then returns list of students if it is success

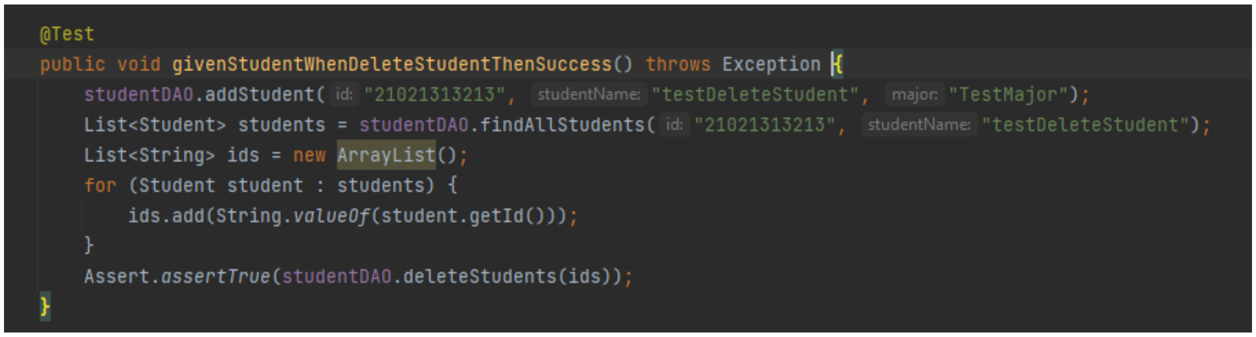


```
@Test
public void givenStudentWhenFindAllStudentsThenSuccess() throws Exception {
    List<Student> students = studentDAO.findAllStudents( id: "201953082052",  studentName: "Azizbek");
    Assert.assertTrue( condition: students.size() > 0);
}
```

5.9 Delete Student

5.9.1 Delete Student Instructions

To delete a student we should provide student id, name and major and It returns true for success or false if it fails.



```
@Test
public void givenStudentWhenDeleteStudentThenSuccess() throws Exception {
    studentDAO.addStudent( id: "21021313213",  studentName: "testDeleteStudent",  major: "TestMajor");
    List<Student> students = studentDAO.findAllStudents( id: "21021313213",  studentName: "testDeleteStudent");
    List<String> ids = new ArrayList();
    for (Student student : students) {
        ids.add(String.valueOf(student.getId()));
    }
    Assert.assertTrue(studentDAO.deleteStudents(ids));
}
```

6. Evaluation

The tests mentioned in this document are performed from 4th June through 6st June of 2022 by the author of this document who is also both the tester and the developer of this project. Below are some of the summaries of the project:

1. The software is a simple student management system. It is developed using IntelliJ IDE Ultimate version Software, java language version 1.8.0_241, and Java DB as its database. Both the tester and the developer use Windows 10 Home Single Language as its Operating System.

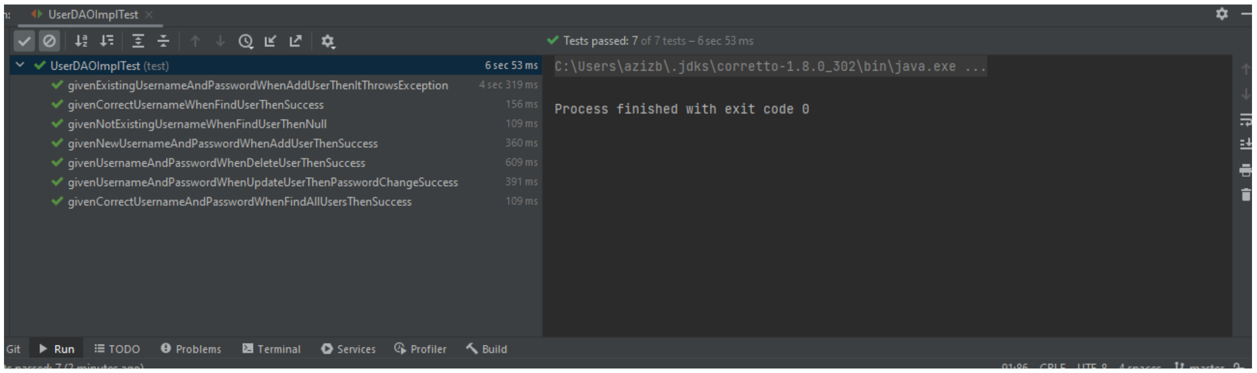
2. The software has complete functions. It provides functions such as find user, add student, update user, update student, delete user, delete student, find all users and other functions. It is comprehensive, reliable, and an easy-to-use software.

3. It is easy to change the functions because they are relatively independent.

4. The system is reliable. It has clear permission restrictions for different users, accurate error, warning, or information prompts.

5. The operation is convenient and easy to comprehend. The interface of each function of the system is simple, the style is consistent and the layout is convenient for users to use.

6. Based on the tests that are performed, all the output of the tests achieves the expected result.

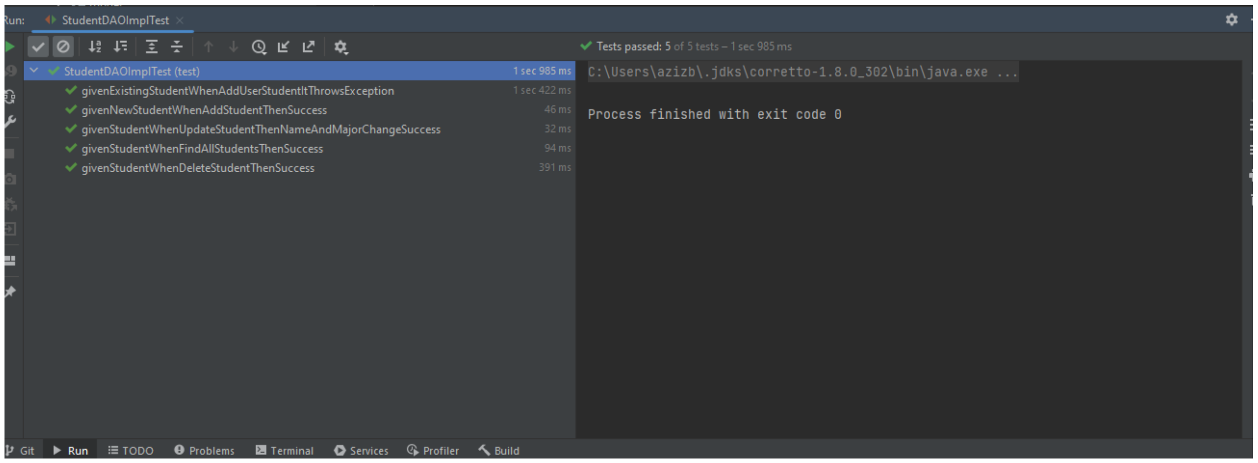


```
Tests passed: 7 of 7 tests – 6 sec 53 ms

UserDAOImplTest (test)                                                   6 sec 53 ms
    givenExistingUsernameAndPasswordWhenAddUserThenItThrowsException     4 sec 319 ms
    givenCorrectUsernameWhenFindUserThenSuccess                          156 ms
    givenNotExistingUsernameWhenFindUserThenNull                         109 ms
    givenNewUsernameAndPasswordWhenAddUserThenSuccess                    360 ms
    givenUsernameAndPasswordWhenDeleteUserThenSuccess                    609 ms
    givenUsernameAndPasswordWhenUpdateUserThenPasswordChangeSuccess      391 ms
    givenCorrectUsernameAndPasswordWhenFindAllUsersThenSuccess           109 ms
```

```
C:\Users\azizb\.jdks\corretto-1.8.0_302\bin\java.exe ...

Process finished with exit code 0
```



```
Run:   StudentDAOImplTest

✓ Tests passed: 5 of 5 tests – 1 sec 985 ms

✓ StudentDAOImplTest (test)                                          1 sec 985 ms
   ✓ givenExistingStudentWhenAddUserStudentItThrowsException         1 sec 422 ms
   ✓ givenNewStudentWhenAddStudentThenSuccess                               46 ms
   ✓ givenStudentWhenUpdateStudentThenNameAndMajorChangeSuccess             32 ms
   ✓ givenStudentWhenFindAllStudentsThenSuccess                             94 ms
   ✓ givenStudentWhenDeleteStudentThenSuccess                              391 ms
```

```
C:\Users\azizb\.jdks\corretto-1.8.0_302\bin\java.exe ...

Process finished with exit code 0
```