Testing Document

for

Computer Science Schedule Generating System

Version 1.0

Prepared by Jiaqi Yang, Runzhi Zhou, Zijun Liu, Tao Huang, Jieyu Ren

CWRU EECS393

November 6, 2019

Revision History

Date	Version	Description	Author(s)
31 Oct 2019	0.1	Initial Draft	Jiaqi Yang, Runzhi Zhou, Zijun Liu, Tao Huang, Jieyu Ren
1 Nov 2019	0.2	Initial Draft - Revision	Jiaqi Yang, Runzhi Zhou, Zijun Liu, Tao Huang, Jieyu Ren
2 Nov 2019	0.3	Assignment - Test Design Specification	Jiaqi Yang, Runzhi Zhou, Zijun Liu, Tao Huang, Jieyu Ren
3 Nov 2019	0.4	Revision of Test Specification	Jiaqi Yang, Runzhi Zhou,

			Zijun Liu, Tao Huang, Jieyu Ren
4 Nov 2019	0.5	Final Draft	Jiaqi Yang, Runzhi Zhou, Zijun Liu, Tao Huang, Jieyu Ren
6 Nov 2019	1.0	Final Draft - Finalized	Jiaqi Yang, Runzhi Zhou, Zijun Liu, Tao Huang, Jieyu Ren

Table of Contents

1.Introduction

1.1 Purpose

The purpose of this document is to provide the outline and goals for the testing strategy for the Computer Science Schedule Generating System. This document will describe the plan for testing the software, and to specify the test cases and test procedures necessary to demonstrate that the software satisfies the requirements as specified in the project's System Requirements Specification document.

1.2 Scope

This testing plan contains an overall list of the use cases to be tested and the software components associated with each test case. This plan also provides a schedule for the testing and every team member's testing assignment. The process for documenting resolving and the findings of errors is also specified. The test specification includes the description of each main function we designed, a list of the features to be tested for each of the use cases, and the necessary test steps to execute each of the test cases.

1.3 Abbreviations, Acronyms, and Definitions

Abbreviations, Acronyms, and Definitions	
Term	**Definition**

Track	Track entered by a user includes 6 tracks listed as Areas of Computer Science Depth Requirement in Case Western Reserve University official bulletin
Substitute Courses	Substitute courses include courses in General Requirements, Core Requirements, Depth Requirements, Breadth Requirements that can substitute for each other.
Course Priority	Course priority, defined by Computer Science Course Generating System, ranks the different requirements in the decreasing order of General Requirements, Core Requirements, Depth Requirements, Breadth Requirements, SAGES Requirements, Technical Requirements.
High priority courses	High priority courses include courses in General Requirements, Core Requirements, Depth Requirements, and Breadth Requirements.
Low priority courses	Low priority courses include courses in Technical Requirements.
General Requirements	General Requirements include 10 courses listed as Major Requirements and 1 course listed as Statistics Requirement in Case Western Reserve University official bulletin.
Core Requirements	Core Requirements include 6 courses listed as Computer Science Core Requirements in Case Western Reserve University official bulletin.
Depth Requirements	Depth Requirements include 4 courses of the user-input track listed as Computer Science Depth Requirements in Case Western Reserve University official bulletin. If a course can be used to meet both Depth Requirements and Breadth Requirements, it will be preferentially counted toward Depth Requirements if the user's Depth Requirements have not been satisfied.

Breadth Requirements	Breadth Requirements include 5 out of 7 courses listed as Computer Science Depth Requirements in Case Western Reserve University official bulletin.
SAGES Requirements	SAGES Requirements include 2 out of 3 SAGES courses from USNA, USSO, USSY listed as University Seminar Requirements in Case Western Reserve University SAGES Requirements bulletin.
Technical Requirements	Technical Requirements include at least 3 courses from Group 1 and at most 2 courses from Group 2 listed as List of Approved Technical Electives in Case Western Reserve University official bulletin.

2. Test Plan Description

2.1 Product Summary

This program will be used as a computer science schedule generating system. Some of the key features of the system are the following. The website will collect BS student course information and import the information into the database, and automatically generate several recommended schedules for the students.

It will also enable the users to view the information of the course(professor, time and prerequisite), enable users to change the recommended courses and select the best one for them.

Lastly, the website will enable the administrator to have access to the database of the system - it will allow the administrator to update course information, delete the course from the list, read the course information and add the course into the list.

2.2 Responsibilities

The roles and responsibilities of each team member during the testing process are listed in the table below.

Roles and Responsibilities During Testing Process		
Name	**Role**	**Responsibilities**
Jiaqi Yang	Team Leader, Database Manager	Manually test Postgresql database, write automatic junit test for data access objects with a coverage higher than 70%
Runzhi Zhou	Database Manager, Front End Developer	Write automatic junit test for data access objects with a coverage higher than 70%, Write automatic unit test for javaScript and Php with a coverage higher than 70%
Jieyu Ren	Back End Developer	Write automatic junit test for classes in generator package with a coverage higher than 70%, manually integral test the generator’s functionalities
Zijun Liu	Back End Developer	Write automatic junit test for classes in generator package with a coverage higher than 70%, manually integral test the generator’s functionalities
Tao Huang	Front End Developer	Write automatic unit test for javaScript and Php with a coverage higher than 70%, manually integral test the web pages’ functionalities

3. Test Design Specification

3.1 Testing Approach

3.1.1 Compatibility Testing

CSSGS should be able to work in conjunction with user's default browser. We will test that our webpage is functioning correctly by opening the webpage using different browser platforms.

3.1.2 Interface testing

CSSGS has many modules and components, including, but not limited to Login, Edit_info, Generate_plans and choose_courses. Information from these components must be passed to each other and to other components throughout the projects. Interface testing will be used to evaluate whether these components and modules pass data and control correctly to the next component.

3.1.3 Regression testing

Since many components of the CSSGS system are connected, we intend to run all the existing test cases(those thattest the features supported by that build) when there are new changes in the main repository.

3.1.4 Coverage testing

As per the SRS, Team CSSGS will be writing their own unit tests on each new code push. Because of this requirement, Team CSSGS aims to cover 100% of the code. This means that 100%(or extremely close to 100%) of the source code should be passed through a test.

3.1.5 Black box testing

CSSGS is an online schedule generating system. Because of the nature of this application, Team CSSGS will be testing the function on the web page to ensure that the application can provide users with correct and efficient functions.

3.1.6 White box testing

Team CSSGS is using unit and integration testing. All of the unit and integration test cases are constructed with detailed knowledge of the code base, and have been automated to run with each code base change.

3.2 Feature or Combination of Features Not to be Tested

The following feature will not be tested during the test phase.

Load Testing- Ability to cope with large flux of traffic or hardware faults. Test cases will not test the ability of the system to deal with a significant amount of users, as well as any potential hardware issues.

3.3 Environmental Needs

Client Network connection is needed for environment in order to best simulate the condition that a typical user or admin will experience when use CSSGS

3.4 Risks and contingencies

The following are the risks with our testing approach that may arise and may impact our progress in delivering the product on schedule:

Underestimating Planned Schedule

If we can not test our program as described in this testing document, the testing phase for our application will be delayed.

4. Test Specification

4.1 Login & Signup

The Login and Signup feature allows new users to sign up a new account and allow old users and admins to log in to the CSSGS system. It is tested Manually.

4.1.1 User Signup

Manual Test Description			
Test ID	**Description**	**Steps**	**Expected Result**
US-01	User should be able	1. Enter a “valid	User successfully

	to sign up for a new account successfully.	username” 2. Enter a “valid password” 3. Click on the “Sign Up” button	creates a new account and jumps to the user login page.Its username and password should be able to be found in the user database.
US-02	User should not be able to successfully create a new account if the entered username already existed in the system.	1. Enter a “repeated username” 2. Enter a “valid password” 3. Click on the “Sign Up” button	A pop_up window with text ”User with the same username already exists.” appears.

4.1.2 User Login

Prerequisites: 1.the user already has an account in the CSSGS system.

2.the user is currently on the login page.

Manual Test Description			
Test ID	**Description**	**Steps**	**Expected Result**
UL-01	User with existing account should be able to log in to the CSSGS system successfully.	1. Enter the username 2. Enter the correct password. 3. Click on the “Log In” button	User successfully logs into the system and jumps to the User Information Update page.
UL-02	User should not be able to log in to the CSSGS system by using the wrong username and password.	1. Enter the username 2. Enter the correct password. 3. Click on the “Log In” button	A pop_up window with text ”Incorrect username or password” appears.

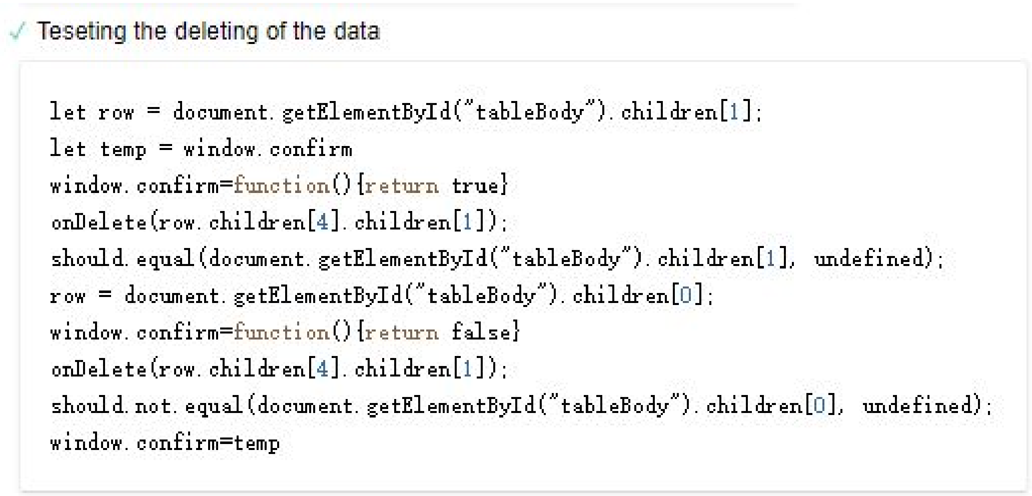
4.2 Admin Page(CRUD)

4.2.1 Test Specification

In this section, we are going to test all functions(adding new data, reading the existing data, resetting the data, choosing and editing the data as well as deleting data) of the CRUD page (admin page) of our system.

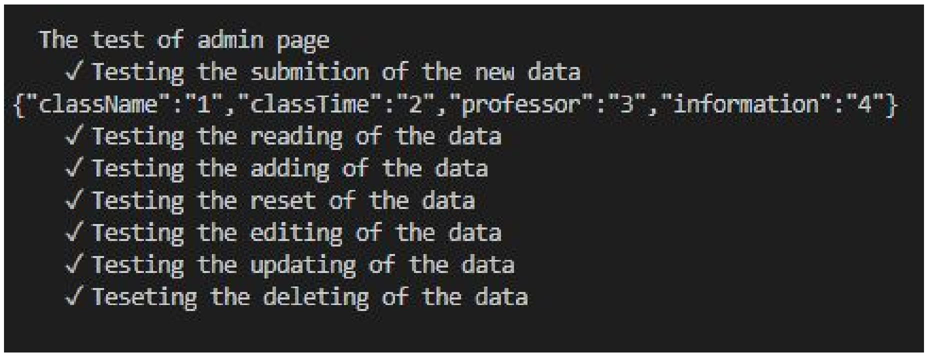
Automated Test Description		
Test	**Description**	**Results**
AP-01	It should return whether it is successful in adding the new data into the web(database)	By calling onFormSubmit method and onEdit method the data is added successfully.
AP-02	It should return whether it is successful in reading the existing data from the web(database)	By calling readForm method the data is read successfully.
AP-03	It should return whether it is successful in adding the new data into the table on the website	By using formData and calling insertNewRecord method, the data is added into the table successfully.
AP-04	It should return whether it is successful in resetting the data	By calling resetform method, the data is reset successfully.
AP-05	It should return whether there is information of the data when we click the data.	By calling onEdit method, the data is read and could be edited.
AP-06	It should return whether it is successful in updating the new data into the web(database).	By calling updateRecord method, the data is updated successfully on the website(database)..
AP-07	It should return whether it is successful in deleting the data into the web(database).	By calling onDelete method, the data is removed successfully from the website(database).

4.2.2 Test Results

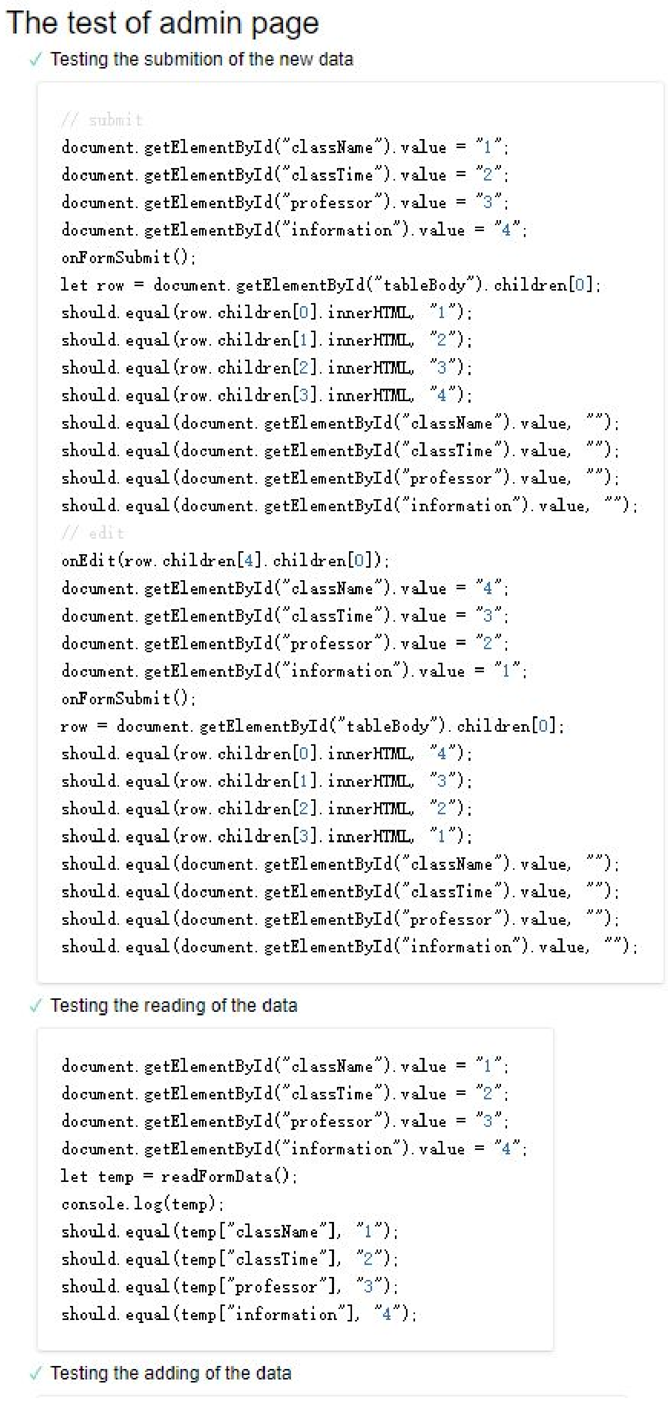


```
✓ Teseting the deleting of the data

let row = document.getElementById("tableBody").children[1];
let temp = window.confirm
window.confirm=function(){return true}
onDelete(row.children[4].children[1]);
should.equal(document.getElementById("tableBody").children[1], undefined);
row = document.getElementById("tableBody").children[0];
window.confirm=function(){return false}
onDelete(row.children[4].children[1]);
should.not.equal(document.getElementById("tableBody").children[0], undefined);
window.confirm=temp
```



```
  The test of admin page
    ✓ Testing the submition of the new data
{"className":"1","classTime":"2","professor":"3","information":"4"}
    ✓ Testing the reading of the data
    ✓ Testing the adding of the data
    ✓ Testing the reset of the data
    ✓ Testing the editing of the data
    ✓ Testing the updating of the data
    ✓ Teseting the deleting of the data
```



The test of admin page

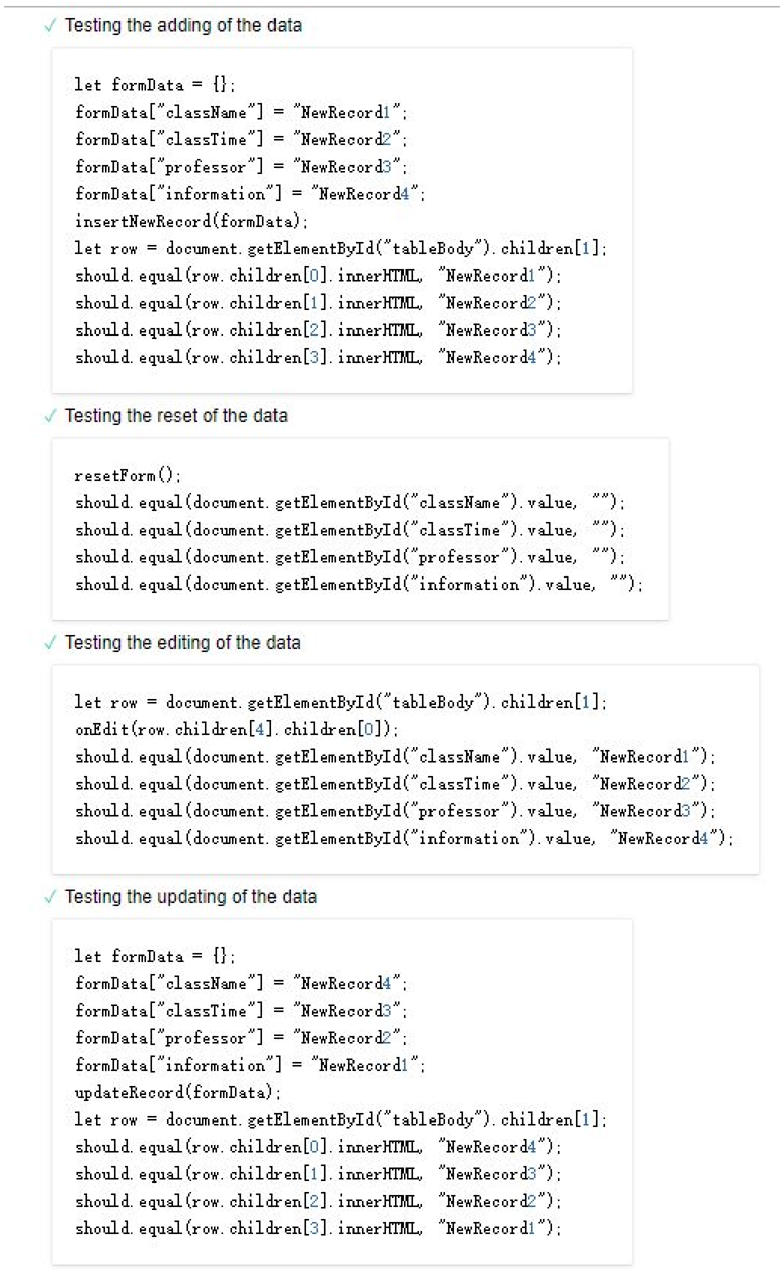
✓ Testing the submition of the new data

```
// submit
document.getElementById("className").value = "1";
document.getElementById("classTime").value = "2";
document.getElementById("professor").value = "3";
document.getElementById("information").value = "4";
onFormSubmit();
let row = document.getElementById("tableBody").children[0];
should.equal(row.children[0].innerHTML, "1");
should.equal(row.children[1].innerHTML, "2");
should.equal(row.children[2].innerHTML, "3");
should.equal(row.children[3].innerHTML, "4");
should.equal(document.getElementById("className").value, "");
should.equal(document.getElementById("classTime").value, "");
should.equal(document.getElementById("professor").value, "");
should.equal(document.getElementById("information").value, "");
// edit
onEdit(row.children[4].children[0]);
document.getElementById("className").value = "4";
document.getElementById("classTime").value = "3";
document.getElementById("professor").value = "2";
document.getElementById("information").value = "1";
onFormSubmit();
row = document.getElementById("tableBody").children[0];
should.equal(row.children[0].innerHTML, "4");
should.equal(row.children[1].innerHTML, "3");
should.equal(row.children[2].innerHTML, "2");
should.equal(row.children[3].innerHTML, "1");
should.equal(document.getElementById("className").value, "");
should.equal(document.getElementById("classTime").value, "");
should.equal(document.getElementById("professor").value, "");
should.equal(document.getElementById("information").value, "");
```

✓ Testing the reading of the data

```
document.getElementById("className").value = "1";
document.getElementById("classTime").value = "2";
document.getElementById("professor").value = "3";
document.getElementById("information").value = "4";
let temp = readFormData();
console.log(temp);
should.equal(temp["className"], "1");
should.equal(temp["classTime"], "2");
should.equal(temp["professor"], "3");
should.equal(temp["information"], "4");
```

✓ Testing the adding of the data



✓ Testing the adding of the data

```
let formData = {};
formData["className"] = "NewRecord1";
formData["classTime"] = "NewRecord2";
formData["professor"] = "NewRecord3";
formData["information"] = "NewRecord4";
insertNewRecord(formData);
let row = document.getElementById("tableBody").children[1];
should.equal(row.children[0].innerHTML, "NewRecord1");
should.equal(row.children[1].innerHTML, "NewRecord2");
should.equal(row.children[2].innerHTML, "NewRecord3");
should.equal(row.children[3].innerHTML, "NewRecord4");
```

✓ Testing the reset of the data

```
resetForm();
should.equal(document.getElementById("className").value, "");
should.equal(document.getElementById("classTime").value, "");
should.equal(document.getElementById("professor").value, "");
should.equal(document.getElementById("information").value, "");
```

✓ Testing the editing of the data

```
let row = document.getElementById("tableBody").children[1];
onEdit(row.children[4].children[0]);
should.equal(document.getElementById("className").value, "NewRecord1");
should.equal(document.getElementById("classTime").value, "NewRecord2");
should.equal(document.getElementById("professor").value, "NewRecord3");
should.equal(document.getElementById("information").value, "NewRecord4");
```

✓ Testing the updating of the data

```
let formData = {};
formData["className"] = "NewRecord4";
formData["classTime"] = "NewRecord3";
formData["professor"] = "NewRecord2";
formData["information"] = "NewRecord1";
updateRecord(formData);
let row = document.getElementById("tableBody").children[1];
should.equal(row.children[0].innerHTML, "NewRecord4");
should.equal(row.children[1].innerHTML, "NewRecord3");
should.equal(row.children[2].innerHTML, "NewRecord2");
should.equal(row.children[3].innerHTML, "NewRecord1");
```

4.3 User Information Update

Manual Test Description			
Test ID	**Description**	**Steps**	**Expected Result**
UIU-01	Full list of all the CS major courses should	1. Log into the GSSCS system	All the CS courses are displayed on the

	appear on the UIU page.		UIU page. All the courses that are marked by the user during the latest update are still marked.
UIU-02	User should be able to mark/unmark the courses in the list to indicate what courses are already taken by the user	1. Click on a course in the list.	If the clicked course is already marked, then the course will be unmarked. If the clicked course is not marked, then the course will be marked.
UIU-03	Users should be able to save the changes made to the list.	1. Click on the "update" button	All the changes made are successfully updated to the user database. The user will jump to the High Priority Course Plan Page.

4.4 Generate High Priority Course Plans

In our design, this section should generate five viable plans for the specified user, with the consideration of priority, prerequisites, timeslots, courses taken and whether the requirement is already met.

Automated Test Description		
Test ID	**Description**	**Expected Result**
GHPC-01	It should return a boolean to determine whether the input user has taken the substitutable courses.	By calling hardCode method, the correct boolean output should be returned.
GHPC-02	It should return a boolean to	By calling isStat method, the

	determine whether the input course is in statistics requirement.	correct boolean output should be returned.
GHPC-03	It should return a boolean to determine whether the input user has completed statistics requirement.	By calling satisfyStat method, the correct boolean output should be returned.
GHPC-04	It should return a boolean to determine whether the input course is in depth requirement.	By calling isDepth method, the correct boolean output should be returned.
GHPC-05	It should return a boolean to determine if the input user takes the input course, whether he or she could complete the depth requirement.	By calling satisfyDepth method, the correct boolean output should be returned.
GHPC-06	It should return a boolean to determine whether the input course is in breadth requirement.	By calling isBreadth method, the correct boolean output should be returned.
GHPC-07	It should return a boolean to determine if the input user takes the input course, whether he or she could complete the breadth requirement.	By calling satisfyBreadth method, the correct boolean output should be returned.
GHPC-08	It should return a boolean to determine whether the input course is already in the current plan by checking courseCode.	By calling isInPlan method, the correct boolean output should be returned.
GHPC-09	It should return a boolean to determine whether the input course is already in the current plan by checking courseID.	By calling isInPlan2 method, the correct boolean output should be returned.
GHPC-10	It should return an arraylist of	By calling noOverlapCourses

	course, which should contain all courses in the input courses that does not overlap with the input timeslot.	method, the correct arraylist of course should be returned.
GHPC-11	It should return a boolean to determine whether the two input timeslots overlap without the consideration of courses that have lab sections.	By calling ifOverlap method, the correct boolean output should be returned.
GHPC-12	It should return a boolean to determine whether the two input timeslots overlap with the consideration of courses that have lab sections.	By calling isSplit method, the correct boolean output should be returned.
GHPC-13	It should return five viable recommended plans for the input user.	By calling generate method, the correct plans should be returned.

Manual Test Description			
Test ID	**Description**	**Steps**	**Expected Result**
GHPC-M-01	The generated course plans should appear on the GHPC page if there are plans.	1.finishing update user information.	Maximum of 5 unique plans are displayed in the GHPC page.
GHPC-M-03	Users should be able to choose one of the plans generated.	1.Click on one of the plans. 2.Click on the “next” button.	User successfully chooses a plan and jumps to the Low Priority Courses section page with courses in the plan shown in the “Courses Selected” list.

4.5 Low Priority Courses Selection

After choosing a plan, the user will jump to the Low Priority Course Selection page to manually add/remove courses from the schedule.There will be two lists shown in the page: one is “Course Selected” list, which lists all the courses selected in the current schedule; Another one is “Available Courses”, which lists all the courses that is still available to be added in the schedule.

Manual Test Description			
Test ID	**Description**	**Steps**	**Expected Result**
LPCS-M-01	Users should be able to add a course into the schedule	1.Chooses course from the “available course” list. 2.Click”Add” button	The selected course appears in the “Courses Selected” list and disappears in the “available course” list. Also, all the courses that have time conflicts with the selected course are removed from the list.
LPCS-M-02	Users should be able to remove a course from the schedule.	1. Chooses a course from the “Course Selected” list. 2.Click”Remove”button	The selected course is successfully removed from the “Course Selected” list. The “available course” list is updated so that it shows all the available courses that are not in conflict with the course in the “Course Selected” list.

4.6 Data Access Layer

In our program, data Access Layer includes CourseDBConnect class and UserInfoDBConnect class, both located in dbconnect package. This layer is responsible for fetching data from our PostgreSql database and converting to the desired format if required.

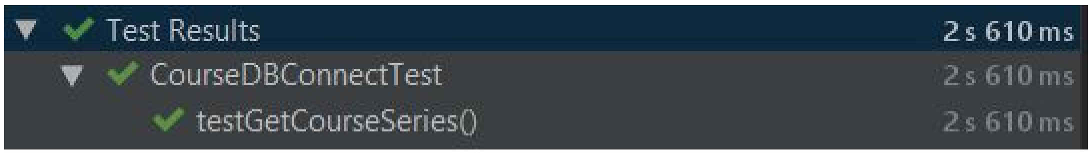
4.6.1 CourseDBConnect Class

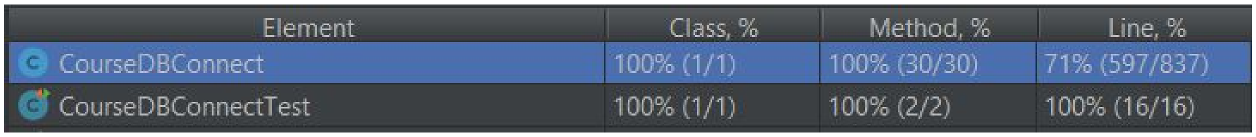
CourseDBConnect class is responsible for fetching course information from course database in our PostgreSql database and converting some of the output into the desired format if required.

Automated Test Description		
Test ID	**Description**	**Expected Result**
CDB-01	It should initialize a singleton instance and return this instance in getCourseDBConnectInstance method.	By calling getCourseDBConnectInstance method, the successfully initialized singleton instance is returned.
CDB-02	It should return the corresponding course name of the input course id.	By calling getCourseName method, the correct course name should be returned.
CDB-03	It should return the corresponding course name of the input course code.	By calling getCourseCode method, the correct course code should be returned.
CDB-04	It should return the corresponding course name of the input course time slots.	By calling getCourseTimeSlots method, the correct course time slots should be returned.
CDB-05	It should return the corresponding course name of the input course prerequisite.	By calling getCoursePrequisite method, the correct course prerequisite should be returned.
CDB-06	It should return the corresponding course name of the input course type.	By calling getCourseTypeFromDB method, the correct course type should be returned.
CDB-07	It should return the corresponding	By calling getCourseDepth method, the

	course name of the input course depth.	correct course depth should be returned.
CDB-08	It should return the corresponding course name of the input course credit hour.	By calling getCourseCreditHour method, the correct course credit hour should be returned.
CDB-09	It should return the corresponding course name of the input course type and convert it to desired format.	By calling getCourseType method, the correct course name should be converted to desired format and returned.
CDB-10	It should return all courses id listed in general requirement.	By calling getGeneralCourseList, the correct and complete list of general course ids should be returned.
CDB-11	It should return all courses id listed in core requirement.	By calling getCoreCourseList, the correct and complete list of core course ids should be returned.
CDB-12	It should return all courses id listed in breadth requirement.	By calling getBreadthCourseList, the correct and complete list of breadth course ids should be returned.
CDB-13	It should return all courses id listed in depth requirement.	By calling getDepthCourseList, the correct and complete list of depth course ids should be returned.
CDB-14	It should return all courses code listed in elective group 1 requirement.	By calling getElectiveGroup1CourseList, the correct and complete list of elective group 1 course ids should be returned.
CDB-15	It should return all courses code listed in elective group 2 requirement.	By calling getElectiveGroup2CourseList, the correct and complete list of elective group 2 course ids should be returned.
CDB-16	It should return all courses code listed in statistics requirement.	By calling getStatisticsCourseCodeList, the correct and complete list of statistics course codes should be returned.
CDB-17	It should return all courses code listed in general requirement.	By calling getGeneralCourseCodeList, the correct and complete list of general course codes should be returned.
CDB-18	It should return all courses code listed in core requirement.	By calling getCoreCourseCodeList, the correct and complete list of core course codes should be returned.

CDB-19	It should return all courses code listed in breadth requirement.	By calling getBreadthCourseCodeList, the correct and complete list of breadth course codes should be returned.
CDB-20	It should return all courses code listed in depth requirement.	By calling getDepthCourseCodeList, the correct and complete list of depth course codes should be returned.
CDB-21	It should return all courses code listed in elective group 1 requirement.	By calling getElectiveGroup1CourseCodeList, the correct and complete list of elective group 1 course codes should be returned.
CDB-22	It should return all courses code listed in elective group 2 requirement.	By calling getElectiveGroup2CourseCodeList, the correct and complete list of elective group 2 course codes should be returned.
CDB-23	It should return all courses code listed in statistics requirement.	By calling getStatisticsCourseCodeList, the correct and complete list of statistics course codes should be returned.
CDB-24	It should return the corresponding Course object of the input course id.	By calling getCourse, the correctly and completely initialized Course object should be returned
CDB-25	It should return the list of all high priority courses in the form of Course object in the decreasing order of priority.	By calling getAllHighPriorityCoursesByPriority method, the correct and complete list of all Course objects in the decreasing order of priority level should be returned.
CDB-26	It should return the list of all depth courses of the input track in the form of Course object.	By calling getCourseListByDepth, the correct and complete list of all Course objects of the input track should be returned.


```
▼ ✔ Test Results                         2 s 610 ms
    ▼ ✔ CourseDBConnectTest              2 s 610 ms
          ✔ testGetCourseSeries()        2 s 610 ms
```



Element	Class, %	Method, %	Line, %
CourseDBConnect	100% (1/1)	100% (30/30)	71% (597/837)
CourseDBConnectTest	100% (1/1)	100% (2/2)	100% (16/16)

4.6.2 UserInfoDBConnect Class

UserInfoDBConnect class is responsible for fetching user information from user_information database in our PostgreSql database and converting some of the output into the desired format if required.

Automated Test Description		
Test	**Description**	**Results**
UDB-01	It should initialize a singleton instance and return this instance in getUserInfoDBConnectInstance method.	By calling getUserInfoDBConnectInstance method, the successfully initialized singleton instance is returned.
UDB-02	It should return the corresponding user name of the input user id.	By calling getUserName method, the correct user name should be returned.
UDB-03	It should return the corresponding user password of the input user id.	By calling getUserPassword method, the correct password should be returned.
UDB-04	It should return the corresponding user type of the input user id.	By calling getUserType method, the correct user type should be returned.
UDB-05	It should return the corresponding core taken of the input user id.	By calling getCoreTaken method, the correct list of core courses taken in the form of Course object should be returned.
UDB-07	It should return the corresponding breadth teaken of the input user id.	By calling getBreadthTaken method, the correct list of breadth courses taken in the form of Course object should be returned.
UDB-08	It should return the corresponding depth taken of the input user id.	By calling getDepthTaken method, the correct list of depth courses taken in the form of Course object should be returned.
UDB-09	It should return the corresponding general taken of the input user id.	By calling getGeneralTaken method, the correct list of general courses taken in the

		form of Course object should be returned.
UDB-10	It should return the corresponding technical elective taken of the input user id.	By calling getElectiveTaken method, the correct list of elective courses taken in the form of Course object should be returned.
UDB-11	It should return the corresponding track of the input user id.	By calling getUserTrack method, the correct track should be returned.
UDB-12	It should return if the administrator username matches the password.	By calling IsCorrectAdminUserPasswordPair method, it correctly returns if the administrator username matches the password.
UDB-13	It should return if the normal username matches the password.	By calling IsCorrectNormalUserPasswordPair method, it correctly returns if the normal username matches the password.
UDB-14	It should return all courses taken in the form of String course code of the input username.	By calling getCourseCodeTaken, it correctly return a complete and correct list of course taken in the form of String course code of the input username.
UDB-15	It should return if an username is used already.	By calling IsUserNameUsed, it correctly returns if an username is used already.

4.7 Course Information

The Course class main job is to assist methods from other classes to access the attributes of a course: credit hours, priority level, course id, course code, course name, time slot, prerequisite(s), course type, and its substitution course’s code. The main method in this class is to “translate” the time slot (a string composed of only integers) into human readable strings.

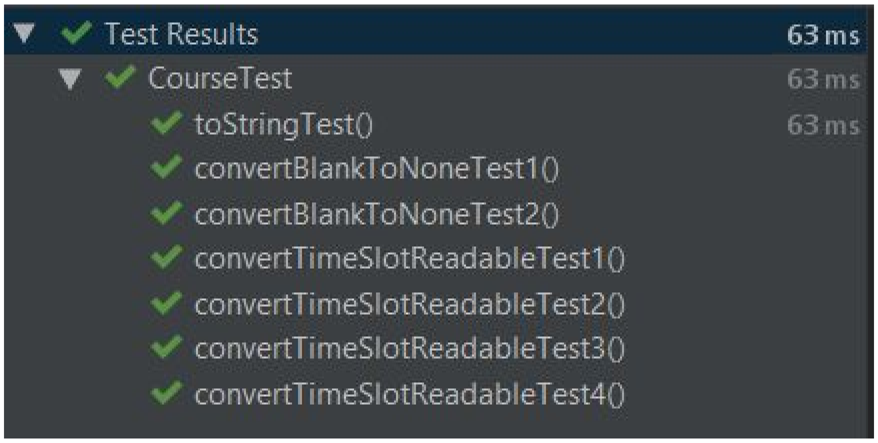
4.7.1 Tests Specifications

Automated Test Description			
Test ID	**Tested Method**	**Description**	**Expected Result**

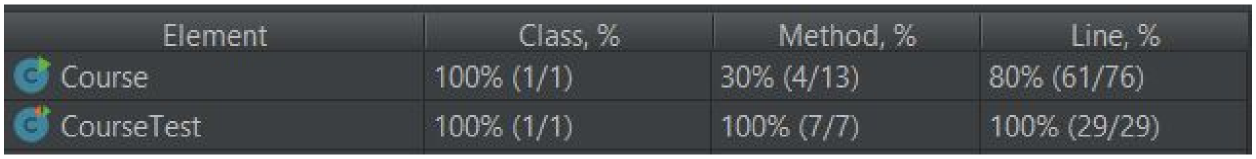
CI-01	convertTimeSlotReadable	This is a structural basis test that tests good data: the input timeslot's length is 22, which indicates the existence of a lab session. Here we can try "1351140123040014001450".	"Regular Section: Monday Wednesday Friday 11:40 - 12:30, Lab/Recitation Section: Tuesday 14:00 - 14:50"
CI-02	convertTimeSlotReadable	This is a structural basis test that tests a good data: the input time slot's length is 11, which indicates that there's no lab sections for this course. Here we can try "13511401230"	"Regular Section: Monday Wednesday Friday 11:40 - 12:30"
CI-03	convertTimeSlotReadable	This is a structural basis test that tests good data: the first digit of the input time slot is zero. Here we can try "01311401230"	""
CI-04	convertTimeSlotReadable	This is a structural basis test that tests bad data: the input time slot's length doesn't fit the format (i.e., the input should be either 11 or 22). Here we can try "1311401230"	"Wrong format of the time slot"
CI-05	convertBlankToNone	This is a structural basis test that tests good data. Here we can try "hello"	"hello"
CI-06	convertBlankToNone	This is a structural basis test that tests good data. Here we can try ""	"None"

CI-07	toString	This is a structural basis test that tests good data: Here we use STAT 312 to be the input course.	"credit: 3 courseID: 1 courseCode: STAT312 courseName: Statistics timeSlot: Regular Section: Tuesday Thursday 14:30 - 15:45 prerequisite: None courseType: statistics requirement substituteCourseCode: STAT380"

4.7.2 Tests Results & Coverage



```
▼ ✓ Test Results                           63 ms
  ▼ ✓ CourseTest                           63 ms
      ✓ toStringTest()                     63 ms
      ✓ convertBlankToNoneTest1()
      ✓ convertBlankToNoneTest2()
      ✓ convertTimeSlotReadableTest1()
      ✓ convertTimeSlotReadableTest2()
      ✓ convertTimeSlotReadableTest3()
      ✓ convertTimeSlotReadableTest4()
```



Element	Class, %	Method, %	Line, %
Course	100% (1/1)	30% (4/13)	80% (61/76)
CourseTest	100% (1/1)	100% (7/7)	100% (29/29)

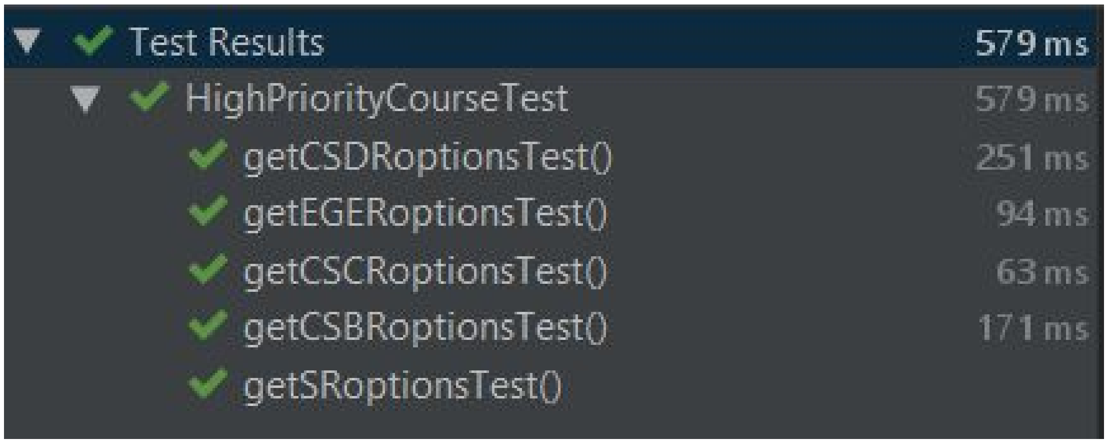
4.8 High Priority Course Information

The HighPriorityCourse class represents courses that fulfill the engineering general requirements, the CS core requirements, the CS depth requirements, the CS breadth requirements, and the statistics requirements. The main methods in this class take the courses already taken by an user and return a list of courses that the user still needs to take to fulfill certain requirement.

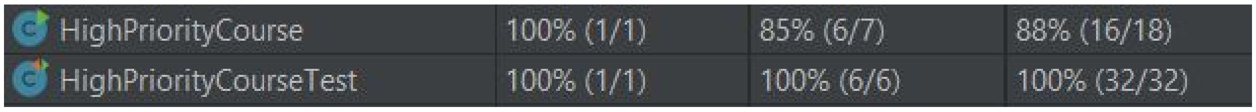
4.8.1 Tests Specifications

Automated Test Description			
Test ID	**Tested Method**	**Description**	**Expected Result**
HP-01	getEGERoptions	This is a structural basis test that tests good data: the student has taken all except two courses from the whole general requirement list.	Course at index 0 should be the first course taken out from the whole requirement list. Course at index 1 should be second course taken out from the whole requirement list.
HP-02	getCSCRoptions	This is a structural basis test that tests good data: the student has taken all except two courses from the whole CS core requirement list.	Course at index 0 should be the first course taken out from the whole requirement list. Course at index 1 should be second course taken out from the whole requirement list.
HP-03	getCSDRoptions	This is a structural basis test that tests good data: the student has taken all except two courses from the whole depth requirement list.	Course at index 0 should be the first course taken out from the whole requirement list. Course at index 1 should be second course taken out from the whole requirement list.
HP-04	getCSBRoptions	This is a structural basis test that tests good data: the student has taken all except two courses from the whole breadth requirement list.	Course at index 0 should be the first course taken out from the whole requirement list. Course at index 1 should be second course taken out from the whole requirement list.
HP-05	getSRoptions	This is a structural basis test that tests good data: the student has taken all except two courses from the whole statistics requirement list.	Course at index 0 should be the first course taken out from the whole requirement list. Course at index 1 should be second course taken out from the whole requirement list.

4.8.2 Tests Results & Coverage



```
▼ ✓ Test Results                         579 ms
  ▼ ✓ HighPriorityCourseTest             579 ms
      ✓ getCSDRoptionsTest()             251 ms
      ✓ getEGERoptionsTest()              94 ms
      ✓ getCSCRoptionsTest()              63 ms
      ✓ getCSBRoptionsTest()             171 ms
      ✓ getSRoptionsTest()
```



HighPriorityCourse	100% (1/1)	85% (6/7)	88% (16/18)
HighPriorityCourseTest	100% (1/1)	100% (6/6)	100% (32/32)

4.9 Low Priority Course Information

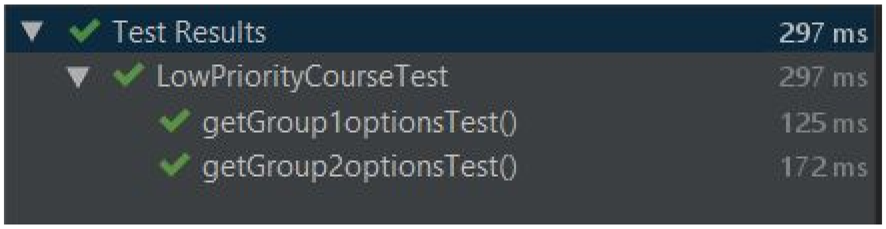
The HighPriorityCourse class represents courses that are in the list of technical electives. The main methods in this class take the courses already taken by an user and return a list of courses that the user can choose to be counted as technical elective course.

4.9.1 Tests Specifications

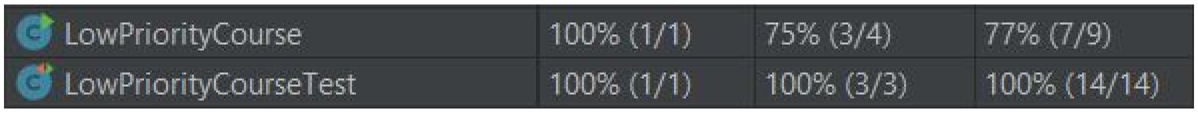
Test ID	Tested Method	Description	Expected Result
LP-01	getGroup1options	This is a structural basis test that tests good data: the student has taken all except two courses from the whole Group 1 elective options list.	Course at index 0 should be the first course taken out from the whole requirement list. Course at index 1 should be second course taken out from the whole options list.
LP-02	getGroup2options	This is a structural basis test that tests good data: the student has taken all except two courses from the whole Group 2	Course at index 0 should be the first course taken out from the whole requirement list. Course at index 1 should be second course taken out from the

		elective options list.	whole options list.

4.9.2 Tests Results & Coverage



```
▼ ✓ Test Results                         297 ms
    ▼ ✓ LowPriorityCourseTest            297 ms
          ✓ getGroup1optionsTest()       125 ms
          ✓ getGroup2optionsTest()       172 ms
```



LowPriorityCourse	100% (1/1)	75% (3/4)	77% (7/9)
LowPriorityCourseTest	100% (1/1)	100% (3/3)	100% (14/14)

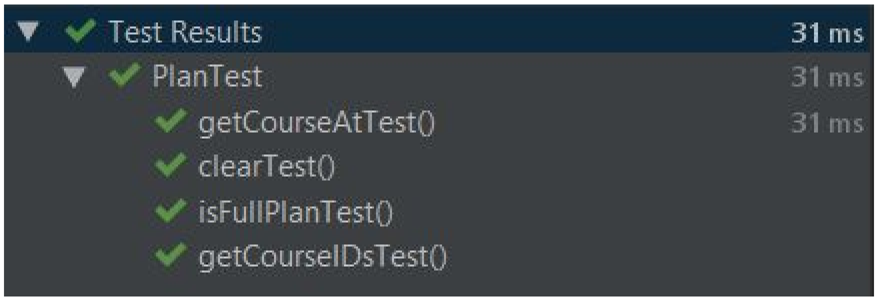
4.10 Plan

4.10.1 Tests Specifications

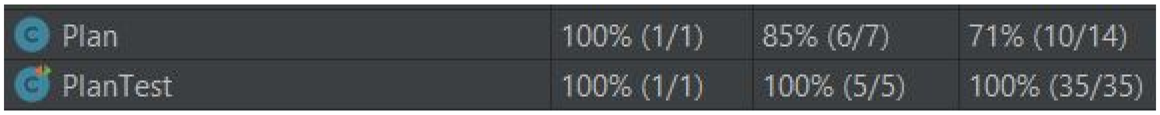
Test ID	Tested Method	Description	Expected Result
PL-01	clear	This is a structural basis test: we want to clear the course list of a plan.	The length of the new plan should be zero.
PL-02	getCourseIDs	This is a structural basis test that tests good data. Here we can try a course list with three courses listed.	The outputs should match the courses' IDs stored in the database.
PL-03	isFullPlan	This is a structural basis test that tests good data. Here we can try a course list with five courses listed. (In our system, a plan can have no more than five courses, because approximately 15-16 credit hours is enough to take during one semester).	true
PL-04	getCourseAt	This is a structural basis test that tests good	3

		data. Here we can try to get the course at index 2 of a course list with five courses whose ids are 1,2,3,4,5, respectively.	

4.10.2 Tests Results & Coverage



```
▼ ✓ Test Results                31 ms
    ▼ ✓ PlanTest                31 ms
          ✓ getCourseAtTest()   31 ms
          ✓ clearTest()
          ✓ isFullPlanTest()
          ✓ getCourseIDsTest()
```



```
Plan        100% (1/1)   85% (6/7)    71% (10/14)
PlanTest    100% (1/1)   100% (5/5)   100% (35/35)
```

4.11 User

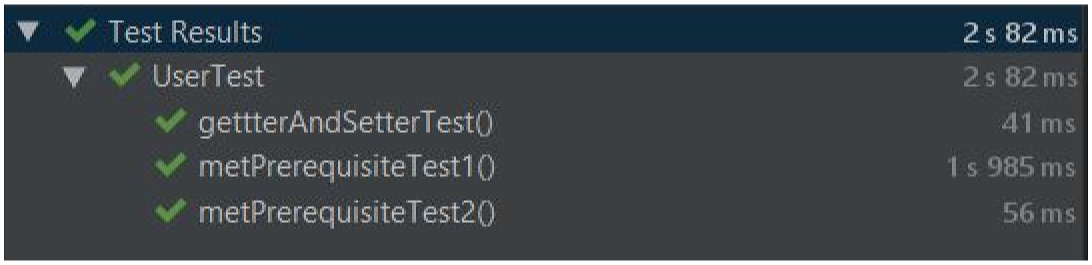
The User class represents a student with his/her username, track, and current plans list. The main method in this class is to test if an user has met the prerequisite of certain course by comparing the courses taken (obtained from the database) and the target course's prerequisite course code.

4.11.1 Tests Specifications

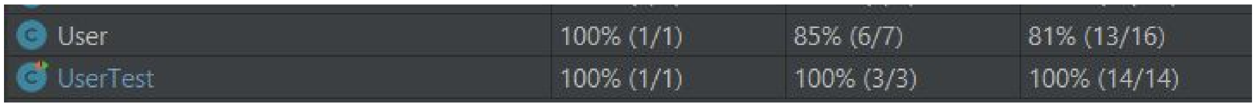
Test ID	**Tested Method**	**Description**	**Expected Result**
UR-01	metPrerequisite	This is a structural basis test: here we can test if a student who has taken EECS 233 has met the prerequisite of EECS 393.	True

UR-02	metPrerequisite	This is a structural basis test: here we can test if a student who has NOT taken EECS 233 has met the prerequisite of EECS 393.	False
UR-03	Getters & Setters	This test verifies that all the getter and setter methods return the correct values.	

4.11.2 Tests Results & Coverage



```
▼ ✓ Test Results                         2 s 82 ms
  ▼ ✓ UserTest                           2 s 82 ms
      ✓ gettterAndSetterTest()               41 ms
      ✓ metPrerequisiteTest1()           1 s 985 ms
      ✓ metPrerequisiteTest2()               56 ms
```



User	100% (1/1)	85% (6/7)	81% (13/16)
UserTest	100% (1/1)	100% (3/3)	100% (14/14)

5. Requirements Traceability

Requirements Traceability Table	
Test ID	**Design Components**
US	Login.html Insert.php style.css
UL	Login.html style.css
AP	Admin.html Script.js style.css
UIU	User_info.html style.css
GHPC	Generator.java
LPCS	LowPriorityCourse.java Plan.java
CDB	CourseDBConnect.java db_backup.sql
UDB	UserInfoDBConnect.java db_backup.sql
CI	Course.java
HP	HighPriorityCourse.java
LP	LowPriorityCourse.java
PL	Plan.java
UR	User.java

6. Inspection Report

Name	Inspection Findings	Resolution
Jiaqi Yang	● Redundant table in part 4.6.2 with similar examples but two one-page table ● Some terminologies' definitions are not clear and not specified in the context ● Some diagrams are too large and they couldn't be displayed right after the context that uses them.	● Remove one of the redundant table ● Add Part 1.3 Acronym and Terminology Dictionary ● Make the diagrams smaller
Zijun Liu	● Description missing for 4.7, 4.8, 4.9, 4.10, and 4.11 ● Missing tests results for 4.7, 4.8, 4.9, 4.10, and 4.11 ● Grammar mistakes	● For each section, add a paragraph of class description. ● Inserting tests results and test coverage screenshots to the end of each section. ● Fix grammar mistakes.
Tao Huang	● Wrong description and some missing description in section 4.1 and 4.2. ● Missing graphs of the test results.	● Deleted the wrong description and added mission information into the section. ● Added missing graphs.
Jieyu Ren	● The description in section 4.4 is not specific. ● Descriptions of class methods not compatible	● Went through the whole section and replaced the description with a more specific one. ● Went through the whole section and fixed the issue.
Runzhi Zhou	● Test documents are included in the Requirements Traceability Table. ● Miss several important methods in User Class ● Some grammar issues	● Deleted all the test documents. ● Added the missing methods in the user class

		● Fixed grammar issues