Deliverable

Project Management/Scrum

a) Assignment of Roles
- Product Owner: John Cowgill
- Scrum Master: Christopher Son
- Development Team: Khalid Abduljabbar, Abdulaziz Ameri, Michael Antonio Mateo, Haroon Mohammed Hussein, Thuong Nguyen, Josiah Villarante

b) Sprint Planning Meeting *In Scrum Diary*

c) Sprint Review *More in-depth in Scrum Diary*
- Sprint 1 Review

Start Time	**Duration**	**Activity**	**Description**	**Role**
2:00 PM	15 min	Review Sprint Goals	Went over the Sprint Goals Reviewed the Roadmap on JIRA	Product Owner
2:15 PM	20 min	Go over main priorities	Go over the highest prioritized goals Check to see what has been completed and what is to be done	Product Owner
2:35 PM	30 min	Demonstrate	Showed the initial code created Demonstrated the features created	Dev Team
3:05 PM	10 min	Sprint Overview	Went over the timetable for the rest of the project Presented hindrances to the Sprint and how to solve them Make sure everything is being done in increments and following the Scrum Guidelines	Scrum Master
3:15 PM	15 min	Unfinished Tasks	Go over what has not been completed Reprioritize unfinished work	Product Owner
3:30 PM	-	Closing	-	-

- Sprint 2 Review

Start Time	**Duration**	**Activity**	**Description**	**Role**
8:00 PM	15 min	Review Sprint Goals	Went over the Sprint Goals Checked to see what Goals were not achieved	Product Owner
8:15 PM	30 min	Main Priorities Completed	Review the highest prioritized tasks Distinguish what is leftover in the prioritized tasks.	Product Owner
9:00 PM	45 min	Demonstrate	Demonstrated the current code that is available	Dev Team
9:05 PM	5 min	Sprint Overview	See what hindrances affected the process and determine solutions Make sure Scrum Process is being followed	Scrum Master
9:10 PM	10 min	Unfinished Tasks	Separate unfinished tasks from deliverable requirements	Product Owner
9:20 PM	-	Closing	Last Sprint	-

Sprint 1 Review:

1) What went well during the Sprint?
   Everyone was able to start their code and was able to create a basis to establish all of the class objects. The login GUI was fully completed and Java Swing was successfully implemented in many aspects of our program. The search algorithm had been created and was implemented to search Class Schedules. The create account interface was also established, same with a universal Student Class.

2) What problem did the team run into? How were those problems solved?
   Some problems that the team ran into were practical in the sense that Eclipse was not functioning correctly for many programmers in the Development Team. Therefore, there was some struggle in aspect of the Github pushes/commits. Also a couple developers were only able to establish a skeleton to their code, rather than fill out the skeleton to start creating a more established set of codes. These problems were approached by working together to figure out how to fix the Eclipse errors through troubleshooting and attempting to look up solutions online. Also, we set up extra meetings throughout the next sprint to use as a progress check-in to keep programmers accountable for their work.

Sprint 2 Review:

1) What went well during the Sprint?
   A lot of the code was able to come together and the majority of the skeletons were filled with functioning code to allow for collaboration between members creating similar functions. For example the Search function was implemented in Class Search as well as the Professor Search. Also, each of the GUI components (Login, Search Class, Search Professor, Create Account) have been created and now function as they were programmed *more in-depth in the Scrum Diary*. There were a couple new features added to the login page (Forgot Password, Change Password)
2) What problem did the team run into? How were those problems solved?
   There were still many problems with pushing code to the Github repository but as a group we were able to reconvene after our Sprint Review/Retrospective and figured out everything out. However, there was a few new problems we ran into. First, a developer did not do anything to update his code, pushing back the requirements needed to bring the rest of the completed code together to create a complete program. Another problem was that many people were unsure of how to test and see the EclEmma coverage. This problem was solved as one of the developers was able to create a couple test cases for the ClassSearch and ClassInfo classes.

d) Sprint Retrospective

- Retrospective #1
    - Worked Well
        - Everyone was able to create a skeleton for the classes that were required.
        - Established the objects that are necessary for the entirety of the project.
        - Majority of the goals with high precedence had been completed.
        - Development team was able to collaborate with each other to establish a common ground to work around and develop code.
    - Needs Improvement
        - Everyone needs to commit as changes are being made rather than pushing when Sprint is coming to near end.
        - Team needs to update time logged in to establish a burndown chart.
        - Tasks that were prioritized should be completed before implementing other tasks into code.
        - Some classes did not coincide correctly and needs to be changed in order obtain processable code.
        - Couple developers were unable to upload code to Github as there were internal issues in Eclipse.
    - What's Next
        - Finish the classes that were not completed (Search, Login Screen, Student).
        - Fix errors in Eclipse.
        - Team should push code as needed and input time logged.
- Retrospective #2
    - Worked Well
        - A lot of new code was pushed and skeleton codes were being implemented with code
        - Demonstration for the GUI interface for both the search function as well as the login were fully functional.
        - Those that collaborated were able to benefit one another by teaching Eclipse functions that many were unaware of
        - Many goals have been completed and a common ground has been established for all.
        - Design Patterns were established and defined as the coding had come together.
        - Code was able to be pushed.
    - Needs Improvement
        - Not all the issues and tasks were accomplished and require more effort from some developers.
        - A storage class is needed.
        - Some classes did not coincide with each other.
        - JIRA needs to be updated on an incremental basis, rather than at end of Sprint
    - What's Next
        - Testing

Handling User Stories

a) Two Examples of CCC

1) As a user, I want to verify my information so that the school has my correct information.

- Card:
  - Requirements: The user should be able to confirm that the information the school has is correct before creating the account.
  - Acceptance criteria: If working properly, the system will display all of the information that the user has entered from the previous story. The user can confirm whether or not the information is correct. If the information is correct, the "Yes" button will display a confirmation message (eg, "the application has been sent") and send an email to the student. If the information is incorrect (there's a typo), the "No" button will send the user back to the beginning of the application. This feature requires that the user has already filled out the application.

- Conversation:
  - FeVER: The user begins on the (completed) page that has the drop down lists "Term applying for" and "Major category". The user clicks the button on the bottom of the screen that says "Verify". The system responds by updating the page to display all of the applicant information that the user has entered. From top to bottom, the system displays the user's first and last name, date of birth, social security number, contact information, the term the user is applying for, and the major category the user has selected. There is a message at the bottom of the screen, "Is the following information correct?", followed by two buttons labeled "Yes" and "No". If the user selects "No", the system takes them back to the beginning of the application page. Else if the student selects "Yes", the system displays a confirmation message, stating "the application has been sent to the school". The system sends a verification email to the student.
- Confirmation:
  - To test if the acceptance criteria has been met, submit an application with incorrect information (i.e., the student's first name is spelt wrong) and click the "No" button. The application should be wiped clear and the user should be able to refill in the information. Next, submit an application with illegal characters or missing information and click the "Yes" button. The system should return an error and prompt the user to fix it. Next, submit an application that's fully completed and correct and click the "Yes" button. The system should display a confirmation message the student has successfully sent their application and there should be a prompt to return to the login screen.

2) As a student, I want to see all of the classes that are offered for this semester so that I can pick which schedule/professor fits best in my schedule.

- Card:
  - Requirements:
    - The user should be able to search for a class and its course number in enrolling classes when they click "search for classes" tab.
    - The user should also be able to search for professor's last name to search for a class
  - Acceptance Criteria: The student must have an account, which is assumingly fulfilled because they have access in the homepage.

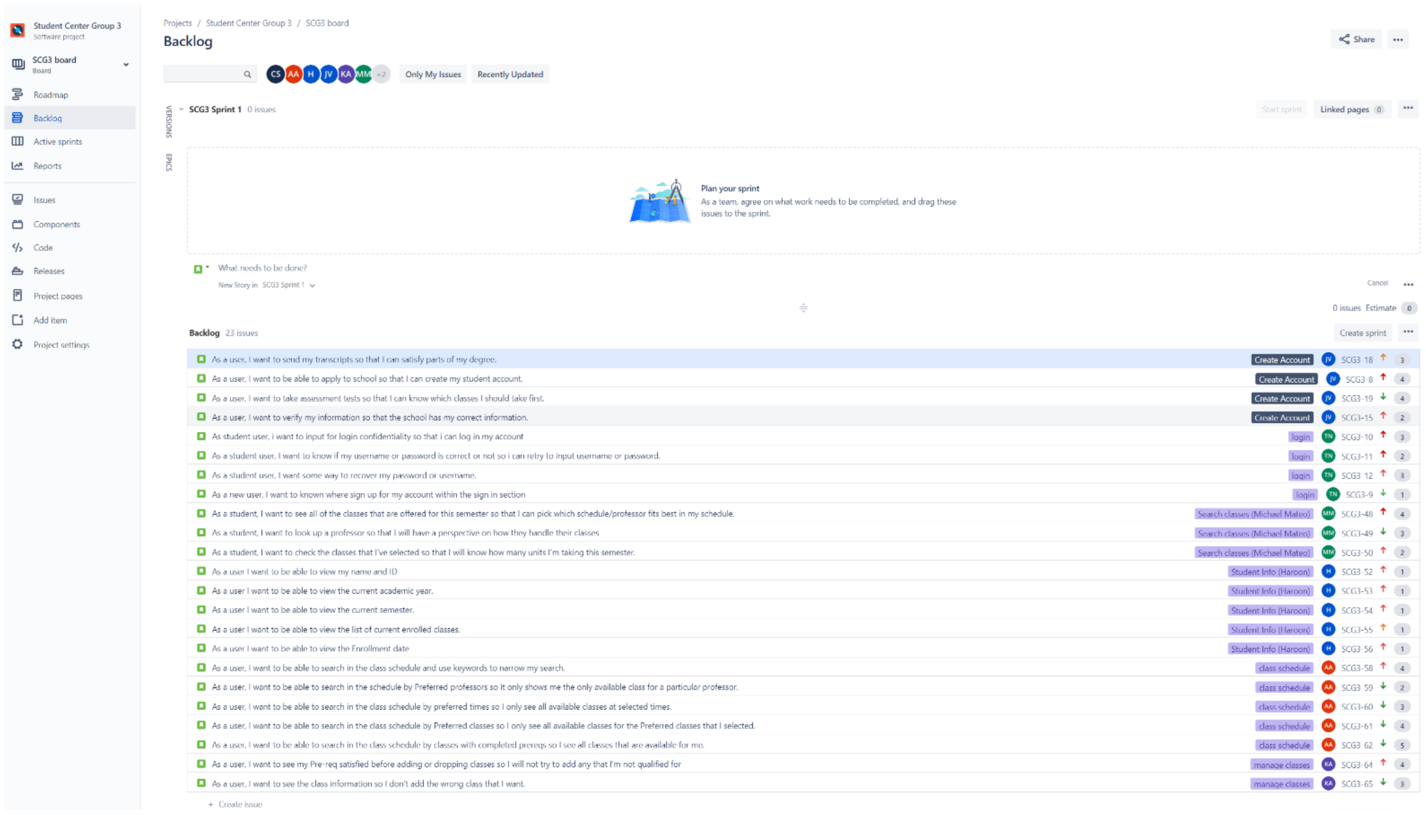
- Conversation:
    - FeVER: When the user clicks on “search for classes” tab in the homepage/dashboard, the system will provide 2 options for the user to search for classes: a drop-down list for subjects along with a text box for course number, and a text box for the professor’s last name. The user will pick a subject from the drop-down list; and they will have an option to enter the course number for that subject to make the search narrower. If the user chooses to search a class by looking up a professor, the system will provide a search box for the user to search a professor by their last name. After the user satisfies one of these two options, the system will display all of the classes offered for the semester. The list includes the subject, subject description, meeting time, the professor and their rating, and the units.
- Confirmation:
    - <u>Test before proceeding to Search Class code:</u>

        - Verify if the user has an account by implementing a simple boolean method. For example, boolean *has_an_account* (returns true if a username is found)

    - <u>Tests to check if Search Class works:</u>

        - First, search a class by using the drop-down list. Check if the drop-down list is working properly and all of the subjects are listed. If the user selected a subject, the program must return all of the available classes for that specific subject.
        - If searching a class by professor, there will be a search box for the user input to enter the professor’s first and last name. Check if the user’s input matches one of the professors in the storage by implementing a boolean function like *is_professor_found* that returns true if the string input by the user in first AND last name matches the professor’s name. The program should display the classes offered by that professor.

b) Story points, Prioritization and Task Assignment using Tool



Backlog

SCG3 Sprint 1 0 issues

Plan your sprint
As a team, agree on what work needs to be completed, and drag these issues to the sprint.

What needs to be done?
New Story in SCG3 Sprint 1

0 issues Estimate 0

Backlog 23 issues

As a user, I want to send my transcripts so that I can satisfy parts of my degree.	Create Account	JV	SCG3-18	↑	3
As a user, I want to be able to apply to school so that I can create my student account.	Create Account	JV	SCG3-8	↑	4
As a user, I want to take assessment tests so that I can know which classes I should take first.	Create Account	JV	SCG3-19	↓	4
As a user, I want to verify my information so that the school has my correct information.	Create Account	JV	SCG3-15	↑	2
As student user, i want to input for login confidentiality so that i can log in my account	login	TN	SCG3-10	↑	3
As a student user, I want to know if my username or password is correct or not so i can retry to input username or password.	login	TN	SCG3-11	↑	2
As a student user, I want some way to recover my password or username.	login	TN	SCG3-12	↑	3
As a new user, I want to known where sign up for my account within the sign in section	login	TN	SCG3-9	↓	1
As a student, I want to see all of the classes that are offered for this semester so that I can pick which schedule/professor fits best in my schedule.	Search classes (Michael Mateo)	MM	SCG3-48	↑	4
As a student, I want to look up a professor so that I will have a perspective on how they handle their classes	Search classes (Michael Mateo)	MM	SCG3-49	↓	3
As a student, I want to check the classes that I've selected so that I will know how many units I'm taking this semester.	Search classes (Michael Mateo)	MM	SCG3-50	↑	2
As a user I want to be able to view my name and ID	Student Info (Haroon)	H	SCG3-52	↑	1
As a user I want to be able to view the current academic year.	Student Info (Haroon)	H	SCG3-53	↑	1
As a user I want to be able to view the current semester.	Student Info (Haroon)	H	SCG3-54	↑	1
As a user I want to be able to view the list of current enrolled classes.	Student Info (Haroon)	H	SCG3-55	↑	1
As a user I want to be able to view the Enrollment date	Student Info (Haroon)	H	SCG3-56	↑	1
As a user, I want to be able to search in the class schedule and use keywords to narrow my search.	class schedule	AA	SCG3-58	↑	4
As a user, I want to be able to search in the schedule by Preferred professors so it only shows me the only available class for a particular professor.	class schedule	AA	SCG3-59	↓	2
As a user, I want to be able to search in the class schedule by preferred times so I only see all available classes at selected times.	class schedule	AA	SCG3-60	↓	3
As a user, I want to be able to search in the class schedule by Preferred classes so I only see all available classes for the Preferred classes that I selected.	class schedule	AA	SCG3-61	↓	4
As a user, I want to be able to search in the class schedule by classes with completed prereqs so I see all classes that are available for me.	class schedule	AA	SCG3-62	↓	5
As a user, I want to see my Pre-req satisfied before adding or dropping classes so I will not try to add any that I'm not qualified for	manage classes	KA	SCG3-64	↑	4
As a user, I want to see the class information so I don't add the wrong class that I want.	manage classes	KA	SCG3-65	↓	3

↑ : Highest Priority

↑ : High Priority

↑ : Medium Priority

↓ : Low Priority

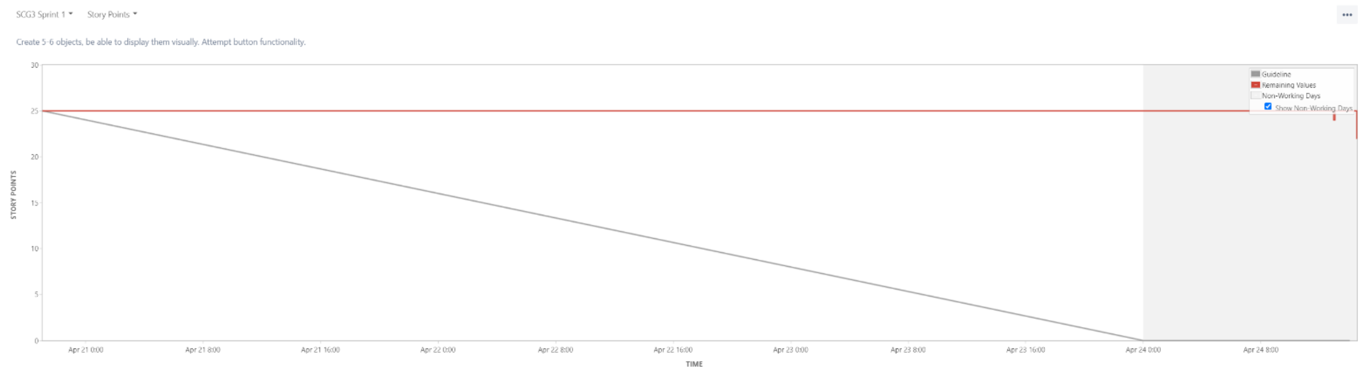
↓ : Lowest Priority

c) Risk Table on 3 P’s

	Predictable	Unpredictable	Solutions
People risk	-Many of the team members cannot attend meetings due to other assignments/projects. This may result in a risk on an unfinished project, leading to a delay of deeming it “Done”.	-Team members may get sick or would have to attend an emergency that was unforeseen, delaying the completion of some tasks.	-These risks can be mitigated as we would create a tentative schedule of when each task is due by and what is needed for completion. Also team members can stay accountable by checking in on progress of others
Process Risk	-Many of the developers could have designs that may compromise each other's designs and would detrimentally take an effect on the procedural aspects in creating our product.	-Developers may run into an error or bug implementing their designs into their code, thus hindering the completion of a more important story requiring that design.	-These risks can be managed by having the developers all collaborate and make sure each design fits well with each other. Also through communication, many issues may be resolved.
Product Risk	-Being on a time table, the final product may face some productivity issues as the code has not been thoroughly tested and does not completely fulfill any of the tests that have been required	-Every developer is of different skill level and the final product may not compile due to a marginal difference in competency of code level. Therefore many problems may arise as the final product may not be fully functional	-To mitigate product risks, a schedule has been set to finalize and deploy the project fully passing every test and with the collaborative efforts from the development team, the code will be entirely completed with instances of peer programming to aid those in implementing the specific code necessary.

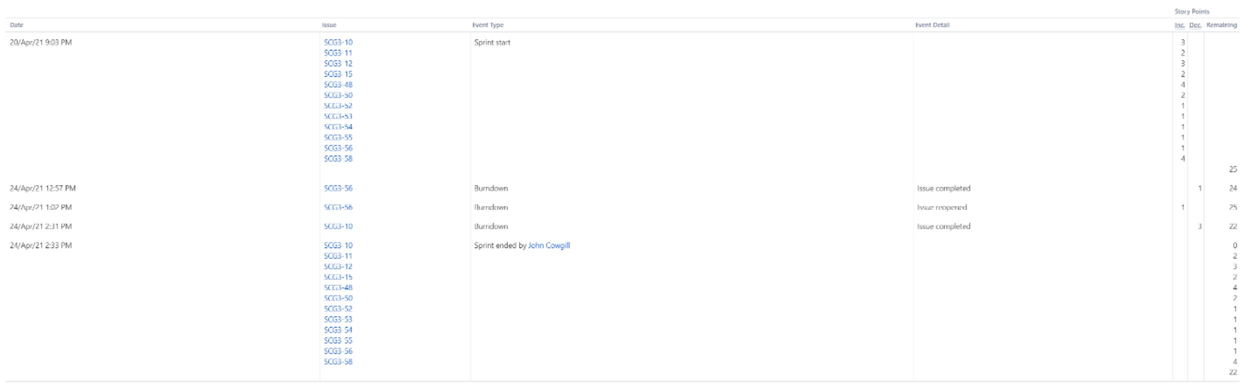
d) Burndown Chart

Sprint 1



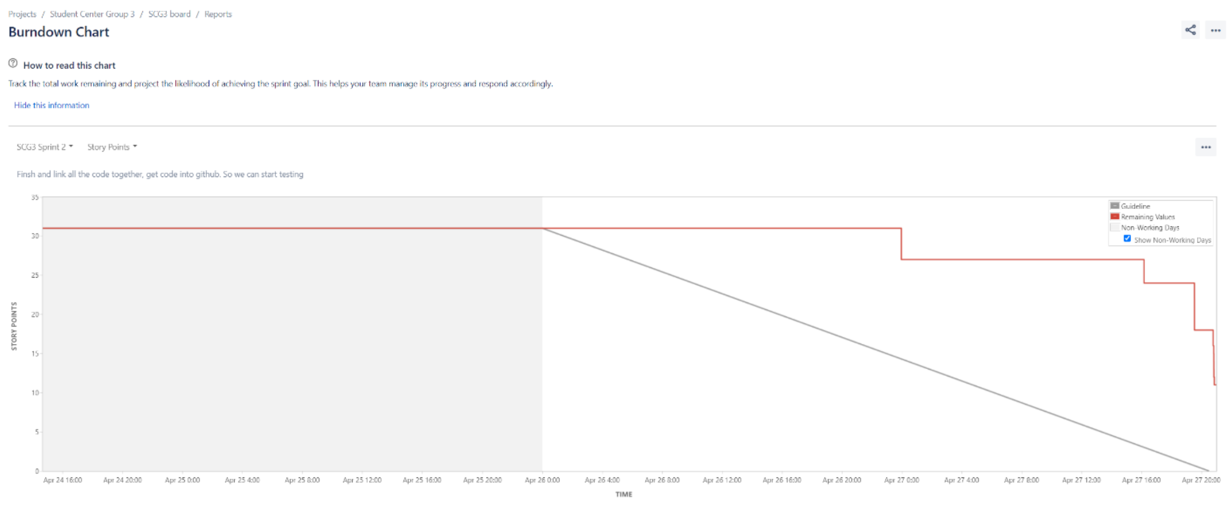
Addressed in Sprint Review

Team did not update times and inputted at the end of the sprint



				Story Points		
Date	Issue	Event Type	Event Detail	Inc.	Dec.	Remaining
20/Apr/21 9:03 PM	SCG3-10	Sprint start		3		
	SCG3-11			2		
	SCG3-12			3		
	SCG3-15			2		
	SCG3-48			4		
	SCG3-50			2		
	SCG3-52			1		
	SCG3-53			1		
	SCG3-54			1		
	SCG3-55			1		
	SCG3-56			1		
	SCG3-58			4		
						25
24/Apr/21 12:57 PM	SCG3-56	Burndown	Issue completed		1	24
24/Apr/21 1:02 PM	SCG3-56	Burndown	Issue reopened	1		25
24/Apr/21 2:31 PM	SCG3-10	Burndown	Issue completed		3	22
24/Apr/21 2:33 PM	SCG3-10	Sprint ended by John Cowgill				0
	SCG3-11					2
	SCG3-12					3
	SCG3-15					2
	SCG3-48					4
	SCG3-50					2
	SCG3-52					1
	SCG3-53					1
	SCG3-54					1
	SCG3-55					1
	SCG3-56					1
	SCG3-58					4
						22

Sprint 2



Projects / Student Center Group 3 / SCG3 board / Reports

Burndown Chart

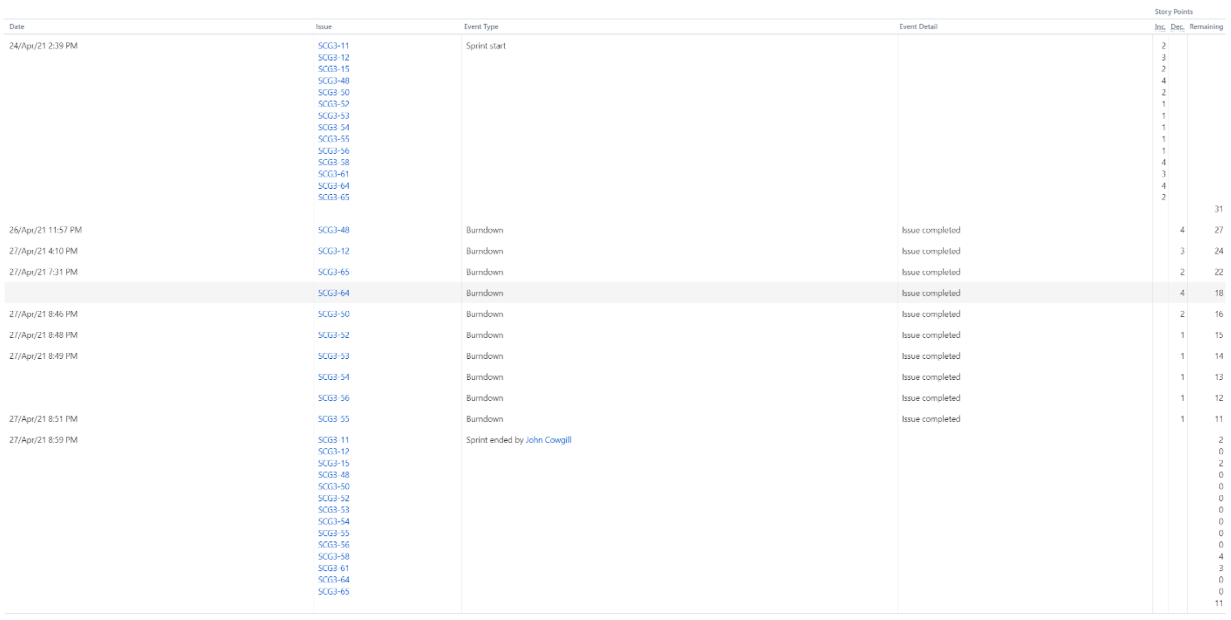
How to read this chart

Track the total work remaining and project the likelihood of achieving the sprint goal. This helps your team manage its progress and respond accordingly.

Hide this information

SCG3 Sprint 2 ▾ Story Points ▾

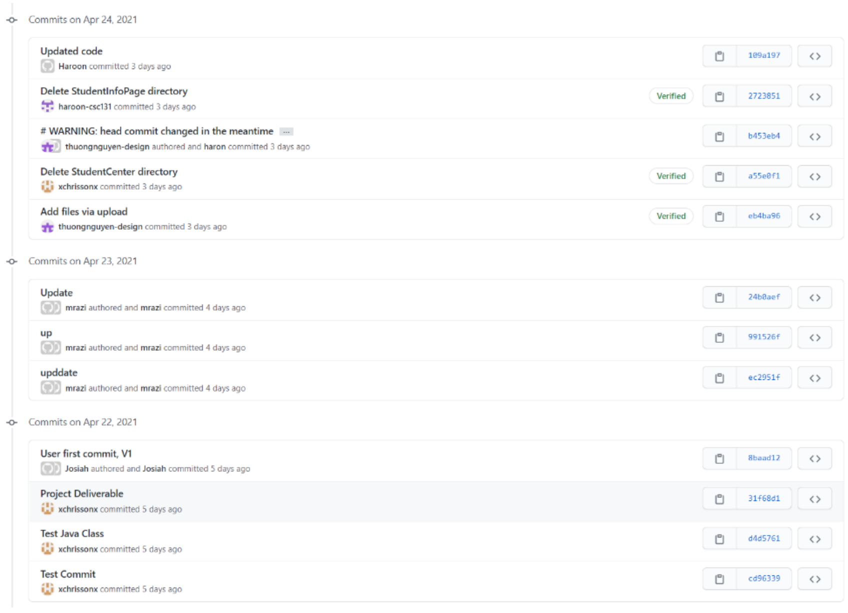
Finsh and link all the code together, get code into github. So we can start testing



				Story Points		
Date	Issue	Event Type	Event Detail	Inc.	Dec.	Remaining
24/Apr/21 2:39 PM	SCG3-11 SCG3-12 SCG3-15 SCG3-48 SCG3-50 SCG3-52 SCG3-53 SCG3-54 SCG3-55 SCG3-56 SCG3-58 SCG3-61 SCG3-64 SCG3-65	Sprint start		2 3 2 4 2 1 1 1 1 1 4 3 4 2		31
26/Apr/21 11:57 PM	SCG3-48	Burndown	Issue completed		4	27
27/Apr/21 4:10 PM	SCG3-12	Burndown	Issue completed		3	24
27/Apr/21 7:31 PM	SCG3-65	Burndown	Issue completed		2	22
	SCG3-64	Burndown	Issue completed		4	18
27/Apr/21 8:46 PM	SCG3-50	Burndown	Issue completed		2	16
27/Apr/21 8:48 PM	SCG3-52	Burndown	Issue completed		1	15
27/Apr/21 8:49 PM	SCG3-53	Burndown	Issue completed		1	14
	SCG3-54	Burndown	Issue completed		1	13
	SCG3-56	Burndown	Issue completed		1	12
27/Apr/21 8:51 PM	SCG3-55	Burndown	Issue completed		1	11
27/Apr/21 8:59 PM	SCG3-11 SCG3-12 SCG3-15 SCG3-48 SCG3-50 SCG3-52 SCG3-53 SCG3-54 SCG3-55 SCG3-56 SCG3-58 SCG3-61 SCG3-64 SCG3-65	Sprint ended by John Cowgill				2 0 2 0 0 0 0 0 0 0 4 3 0 0 11

Coding using GitHub:

a) Proof of Collaboration
SOME_LONER: Josiah
mrazi: Abdulaziz Ameri
mickeymacho6: Michael Mateo



Commits on Apr 24, 2021

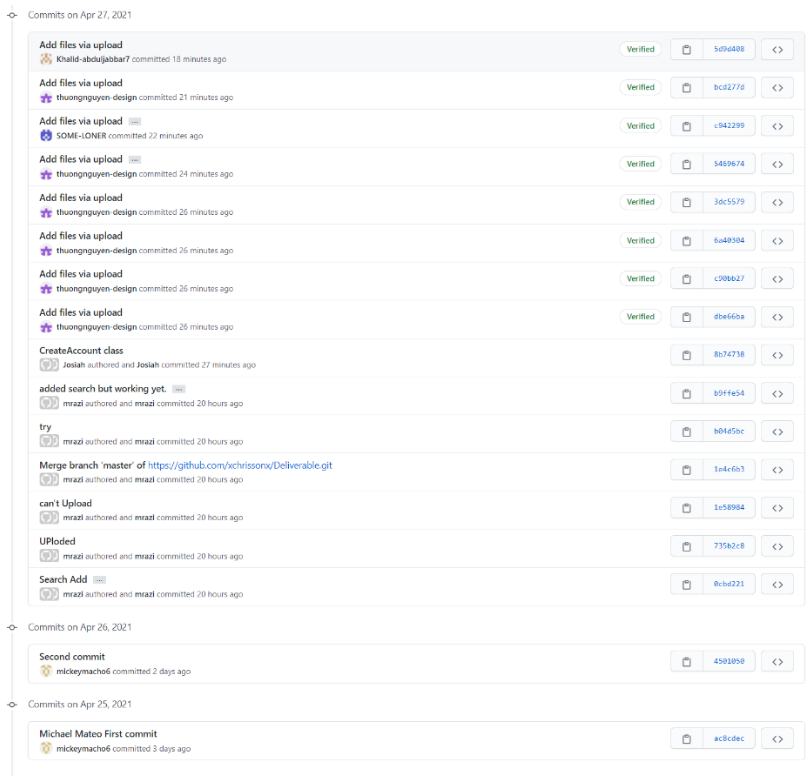
Updated code Haroon committed 3 days ago		109a197
Delete StudentInfoPage directory haroon-csc131 committed 3 days ago	Verified	2723851
# WARNING: head commit changed in the meantime … thuongnguyen-design authored and haron committed 3 days ago		b453eb4
Delete StudentCenter directory xchrissonx committed 3 days ago	Verified	a55e0f1
Add files via upload thuongnguyen-design committed 3 days ago	Verified	eb4ba96

Commits on Apr 23, 2021

Update mrazi authored and mrazi committed 4 days ago		24b0aef
up mrazi authored and mrazi committed 4 days ago		991526f
upddate mrazi authored and mrazi committed 4 days ago		ec2951f

Commits on Apr 22, 2021

User first commit, V1 Josiah authored and Josiah committed 5 days ago		8baad12
Project Deliverable xchrissonx committed 5 days ago		31f68d1
Test Java Class xchrissonx committed 5 days ago		d4d5761
Test Commit xchrissonx committed 5 days ago		cd96339


Commits on Apr 27, 2021

Add files via upload
Khalid-abduljabbar7 committed 18 minutes ago — Verified — 5d9d408

Add files via upload
thuongnguyen-design committed 21 minutes ago — Verified — bcd277d

Add files via upload …
SOME-LONER committed 22 minutes ago — Verified — c942299

Add files via upload …
thuongnguyen-design committed 24 minutes ago — Verified — 5469674

Add files via upload
thuongnguyen-design committed 26 minutes ago — Verified — 3dc5579

Add files via upload
thuongnguyen-design committed 26 minutes ago — Verified — 6a40304

Add files via upload
thuongnguyen-design committed 26 minutes ago — Verified — c90bb27

Add files via upload
thuongnguyen-design committed 26 minutes ago — Verified — dbe66ba

CreateAccount class
Josiah authored and Josiah committed 27 minutes ago — 8b74738

added search but working yet. …
mrazi authored and mrazi committed 20 hours ago — b9ffe54

try
mrazi authored and mrazi committed 20 hours ago — b04d5bc

Merge branch 'master' of https://github.com/xchrissonx/Deliverable.git
mrazi authored and mrazi committed 20 hours ago — 1e4c6b3

can't Upload
mrazi authored and mrazi committed 20 hours ago — 1e58984

UPloded
mrazi authored and mrazi committed 20 hours ago — 735b2c8

Search Add …
mrazi authored and mrazi committed 20 hours ago — 0cbd221

Commits on Apr 26, 2021

Second commit
mickeymacho6 committed 2 days ago — 4501050

Commits on Apr 25, 2021

Michael Mateo First commit
mickeymacho6 committed 3 days ago — ac8cdec

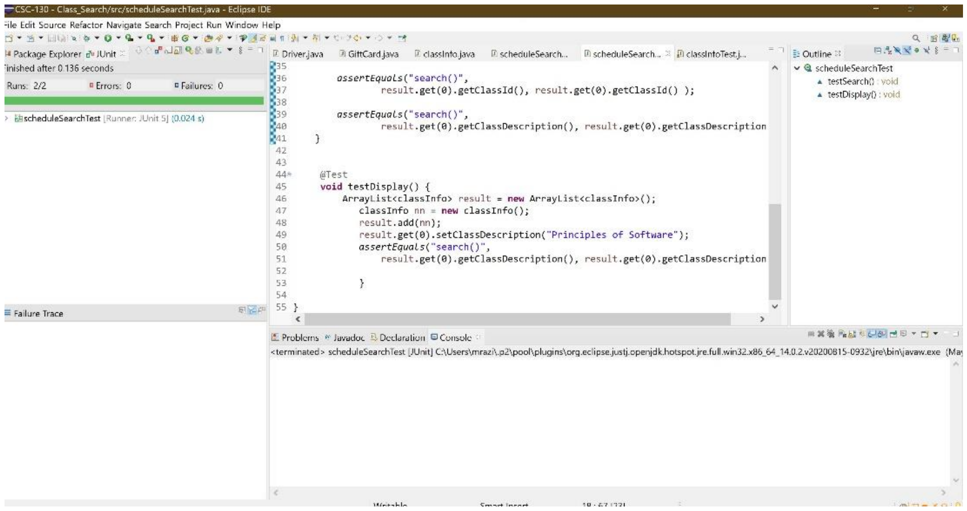
b) High Cohesion and Low Coupling

When looking at the code and the diagrams provided, it is quite visible that there is a strong basis surrounding the aspects of both High Cohesion and Low Coupling Principles. It is quite evident that each of the classes are subject to a specific task and does it with great aptitude. Looking at each class, it is apparent that the grouped elements (i.e. User, Search, Login) all pertain to specific functionality that would not coincide with each other. It is apparent that the modules do not cross levels of abstraction within each other as each module is defined and clearly separated. More specifically, there are instances of logical cohesion, that only gather those components with actual relations. With this high level of cohesivity, it is clear there is a sense of low coupling. In aspect of coupling, it is apparent that rather than every class being knowledgeable of every single modularity, many of the classes are able to stand with small amounts of dependencies. For instance, there are instances of loose coupling in regards to the user extending to only the professors and students; other instances may include the classInfo class extending, as well as the login. This is significant towards the design of the project as it allows for the code to be easily maintained.

c) Using Junit

Not everyone tested only providing the test cases that were completed

ClassSearch Test



CSC-130 - Class_Search/src/scheduleSearchTest.java - Eclipse IDE

Finished after 0.136 seconds

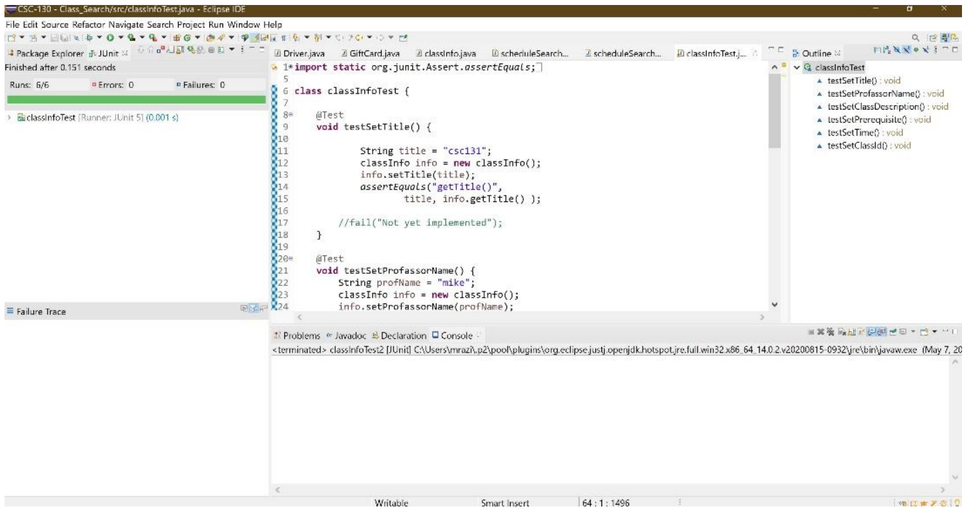
Runs: 2/2 Errors: 0 Failures: 0

scheduleSearchTest [Runner: JUnit 5] (0.024 s)

```java
        assertEquals("search()",
                result.get(0).getClassId(), result.get(0).getClassId() );

        assertEquals("search()",
                result.get(0).getClassDescription(), result.get(0).getClassDescription
    }


    @Test
    void testDisplay() {
        ArrayList<classInfo> result = new ArrayList<classInfo>();
            classInfo nn = new classInfo();
            result.add(nn);
            result.get(0).setClassDescription("Principles of Software");
            assertEquals("search()",
                result.get(0).getClassDescription(), result.get(0).getClassDescription

            }

}
```

```
<terminated> scheduleSearchTest [JUnit] C:\Users\mrazi\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20200815-0932\jre\bin\javaw.exe (May
```

ClassInfo Test



CSC-130 - Class_Search/src/classInfoTest.java - Eclipse IDE

Finished after 0.151 seconds

Runs: 6/6 Errors: 0 Failures: 0

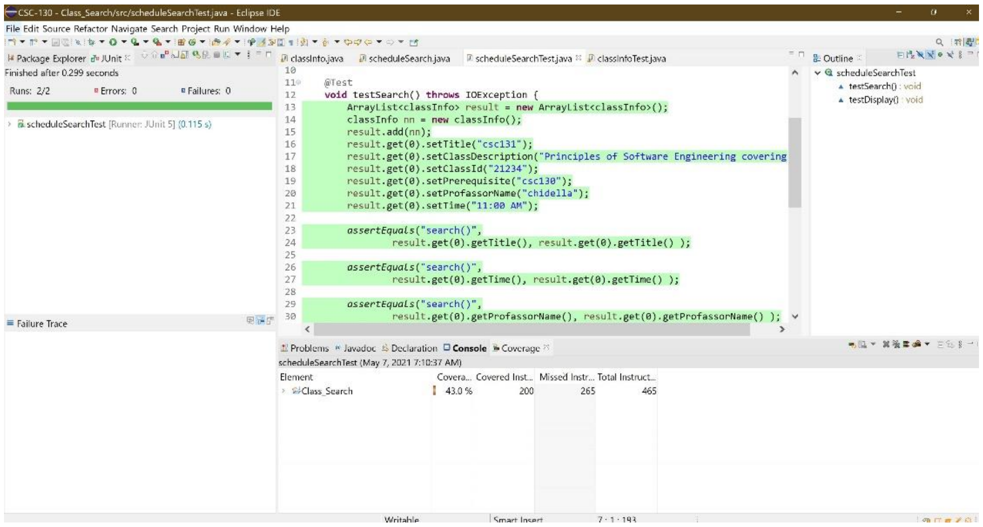
classInfoTest [Runner: JUnit 5] (0.001 s)

```java
import static org.junit.Assert.assertEquals;

class classInfoTest {

    @Test
    void testSetTitle() {

            String title = "csc131";
            classInfo info = new classInfo();
            info.setTitle(title);
            assertEquals("getTitle()",
                    title, info.getTitle() );

        //fail("Not yet implemented");
    }

    @Test
    void testSetProfassorName() {
        String profName = "mike";
        classInfo info = new classInfo();
        info.setProfassorName(profName);
```

```
<terminated> classInfoTest2 [JUnit] C:\Users\mrazi\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20200815-0932\jre\bin\javaw.exe (May 7, 20
```

d) EclEmma Coverage

ClassSearch



CSC-130 - Class_Search/src/scheduleSearchTest.java - Eclipse IDE

Finished after 0.299 seconds

Runs: 2/2 Errors: 0 Failures: 0

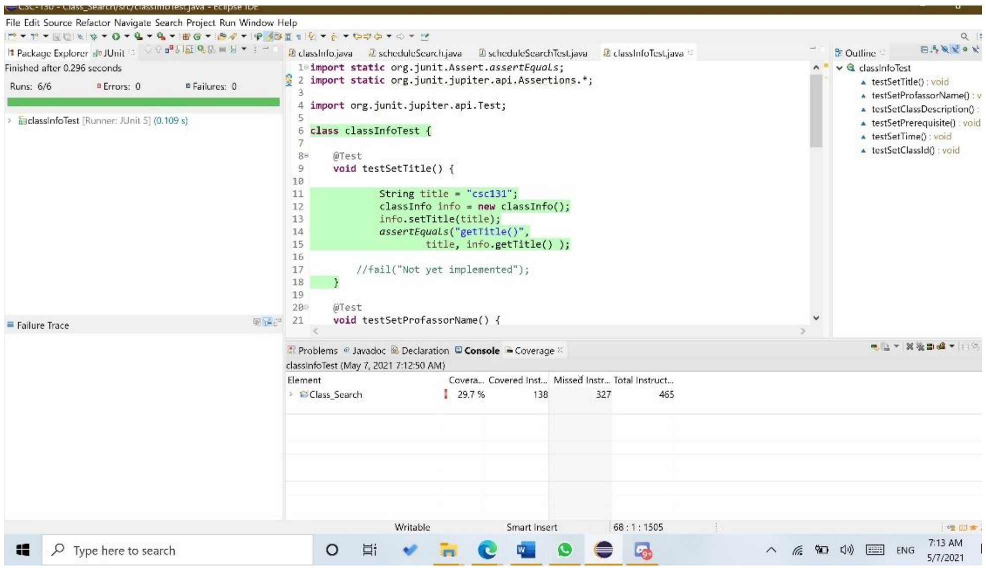
scheduleSearchTest [Runner: JUnit 5] (0.115 s)

```java
    @Test
    void testSearch() throws IOException {
        ArrayList<classInfo> result = new ArrayList<classInfo>();
        classInfo nn = new classInfo();
        result.add(nn);
        result.get(0).setTitle("csc131");
        result.get(0).setClassDescription("Principles of Software Engineering covering
        result.get(0).setClassId("21234");
        result.get(0).setPrerequisite("csc130");
        result.get(0).setProfassorName("chidella");
        result.get(0).setTime("11:00 AM");

        assertEquals("search()",
                result.get(0).getTitle(), result.get(0).getTitle() );

        assertEquals("search()",
                result.get(0).getTime(), result.get(0).getTime() );

        assertEquals("search()",
                result.get(0).getProfassorName(), result.get(0).getProfassorName() );
```

scheduleSearchTest (May 7, 2021 7:10:37 AM)

Element	Covera...	Covered Inst...	Missed Instr...	Total Instruct...
Class_Search	43.0 %	200	265	465

ClassInfo



CSC-130 - Class_Search/src/classInfoTest.java - Eclipse IDE

Finished after 0.296 seconds

Runs: 6/6 Errors: 0 Failures: 0

classInfoTest [Runner: JUnit 5] (0.109 s)

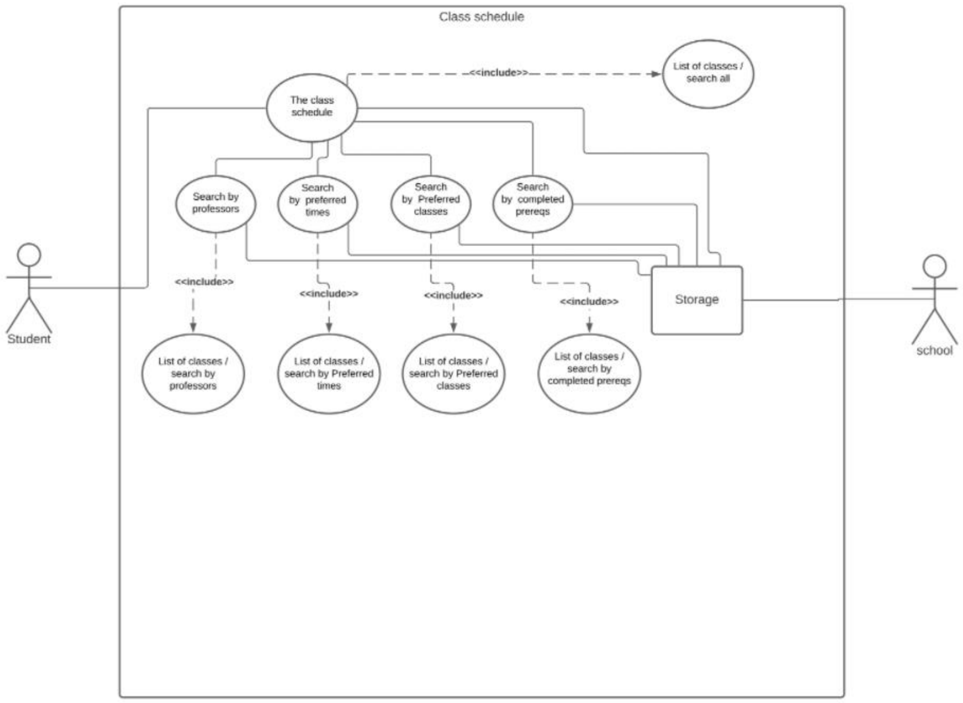
```java
import static org.junit.Assert.assertEquals;
import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

class classInfoTest {

    @Test
    void testSetTitle() {

            String title = "csc131";
            classInfo info = new classInfo();
            info.setTitle(title);
            assertEquals("getTitle()",
                    title, info.getTitle() );

        //fail("Not yet implemented");
    }

    @Test
    void testSetProfassorName() {
```

classInfoTest (May 7, 2021 7:12:50 AM)

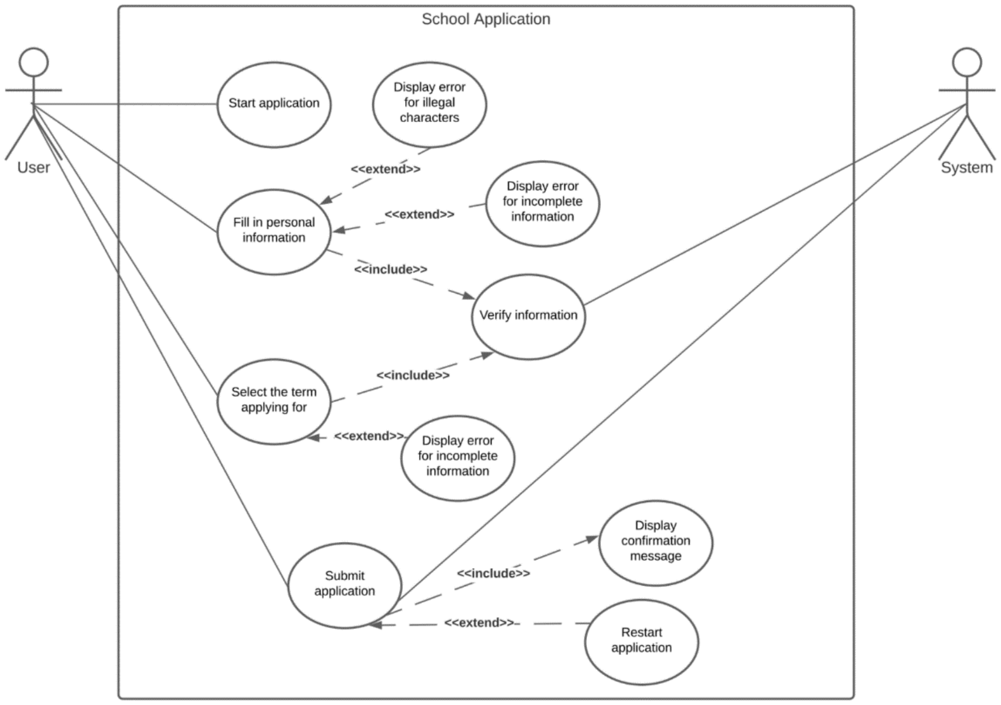
Element	Covera...	Covered Inst...	Missed Instr...	Total Instruct...
Class_Search	29.7 %	138	327	465

Four types of UML Diagrams

a) Use Case

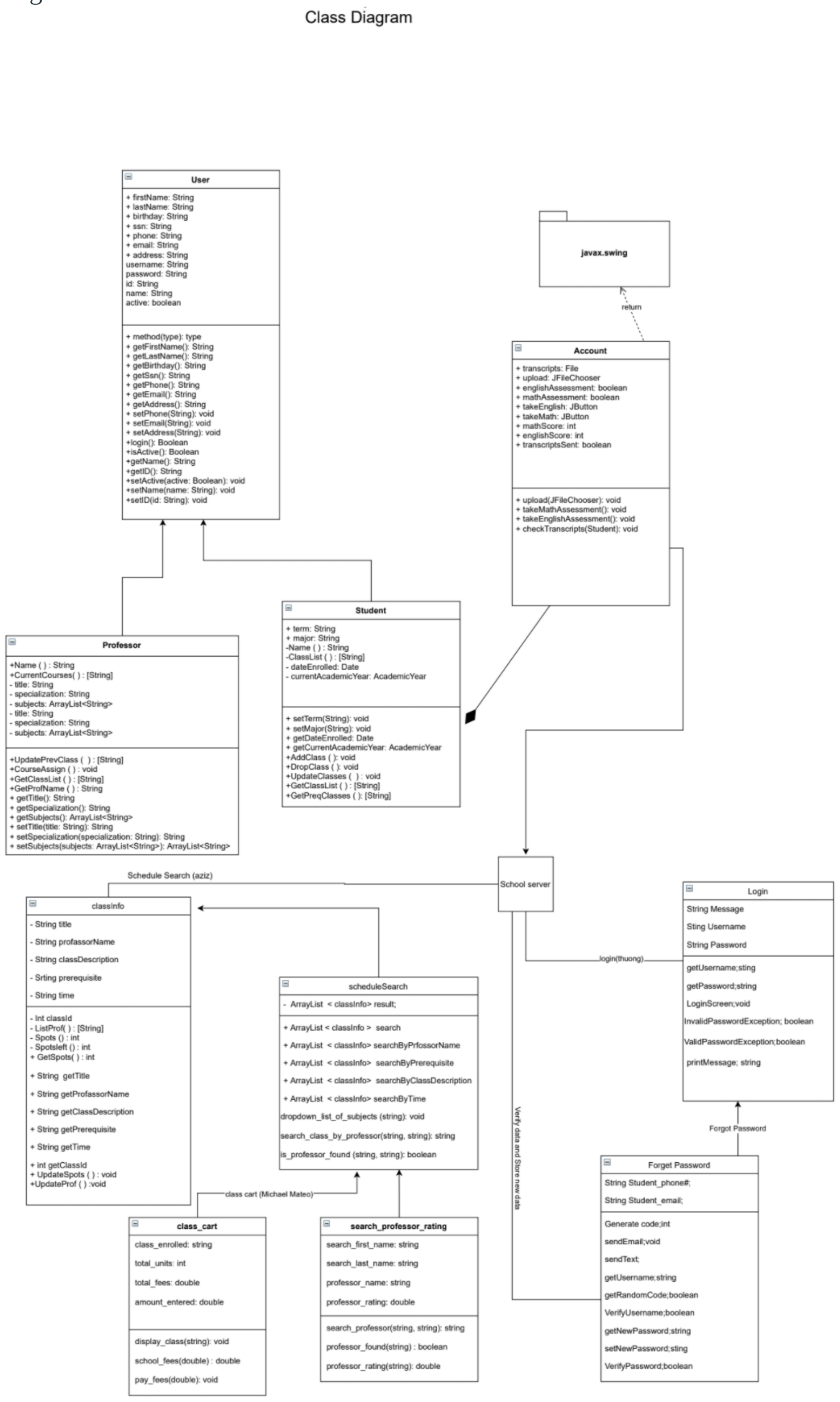


Class schedule
The class schedule
<<include>>
List of classes / search all
Search by professors
Search by preferred times
Search by Preferred classes
Search by completed prereqs
<<include>>
<<include>>
<<include>>
<<include>>
List of classes / search by professors
List of classes / search by Preferred times
List of classes / search by Preferred classes
List of classes / search by completed prereqs
Storage
Student
school

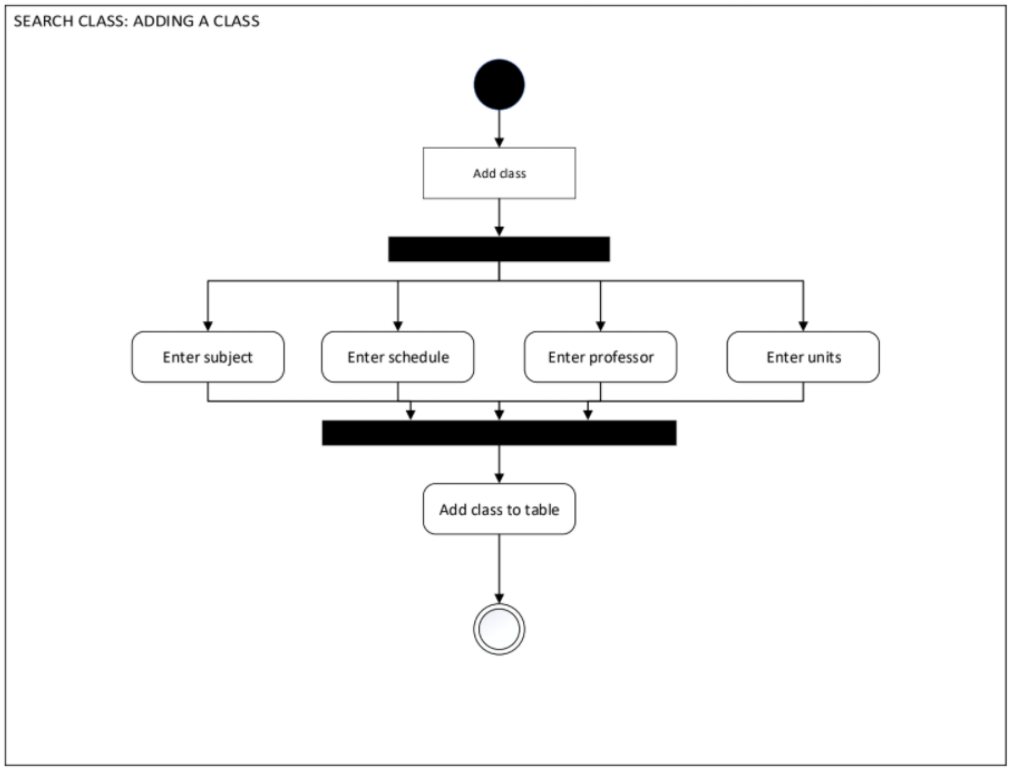


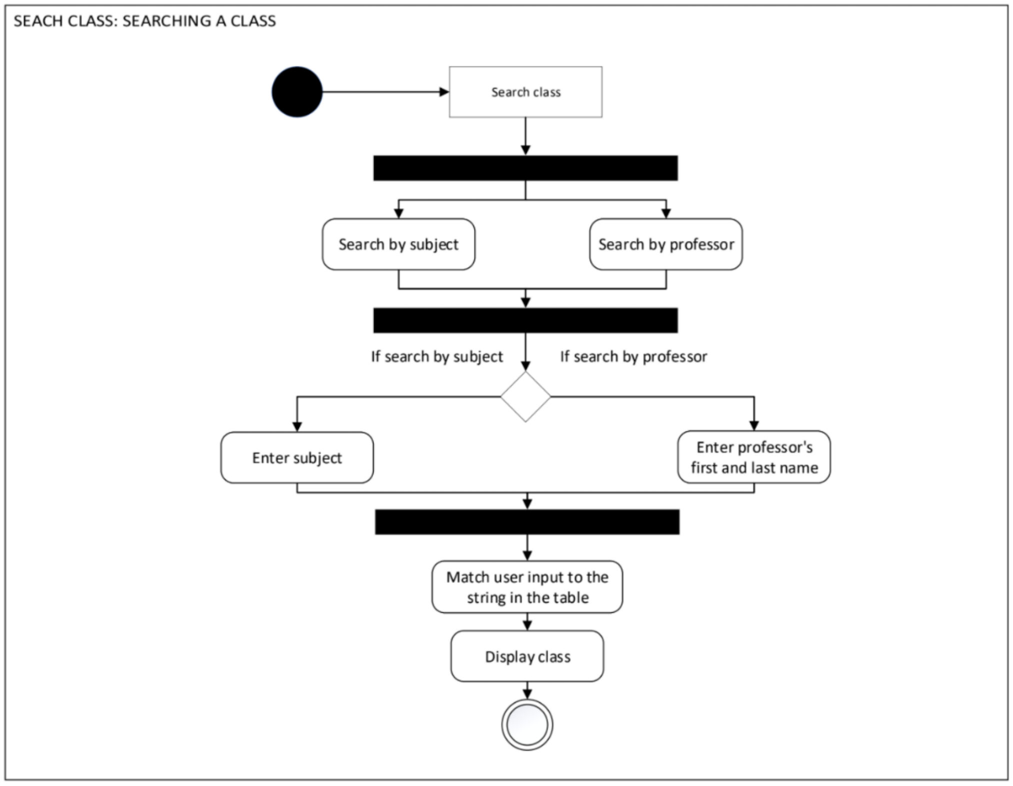
School Application
User
System
Start application
Display error for illegal characters
<<extend>>
Display error for incomplete information
<<extend>>
Fill in personal information
<<include>>
Verify information
<<include>>
Select the term applying for
<<extend>>
Display error for incomplete information
Display confirmation message
<<include>>
Submit application
<<extend>>
Restart application

b) Class Diagram

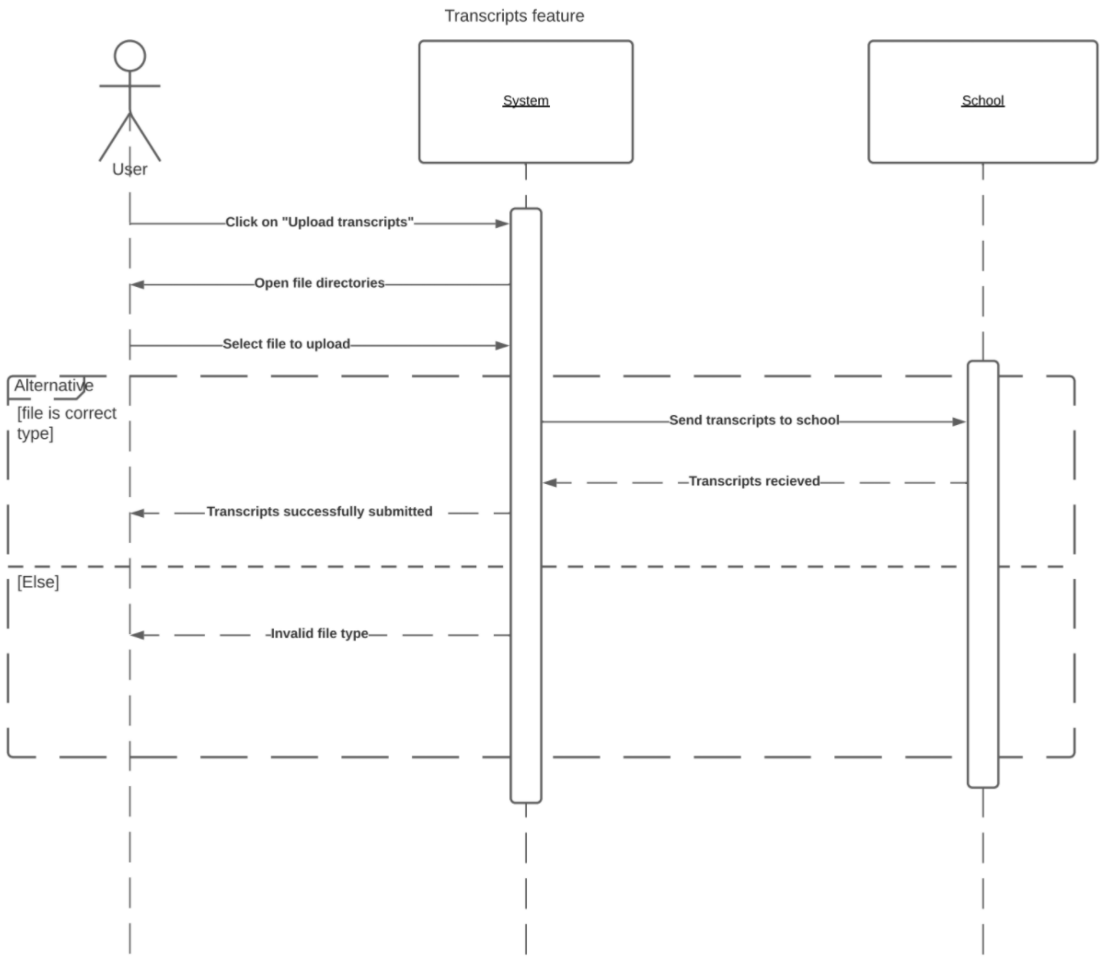


c) Activity Diagram

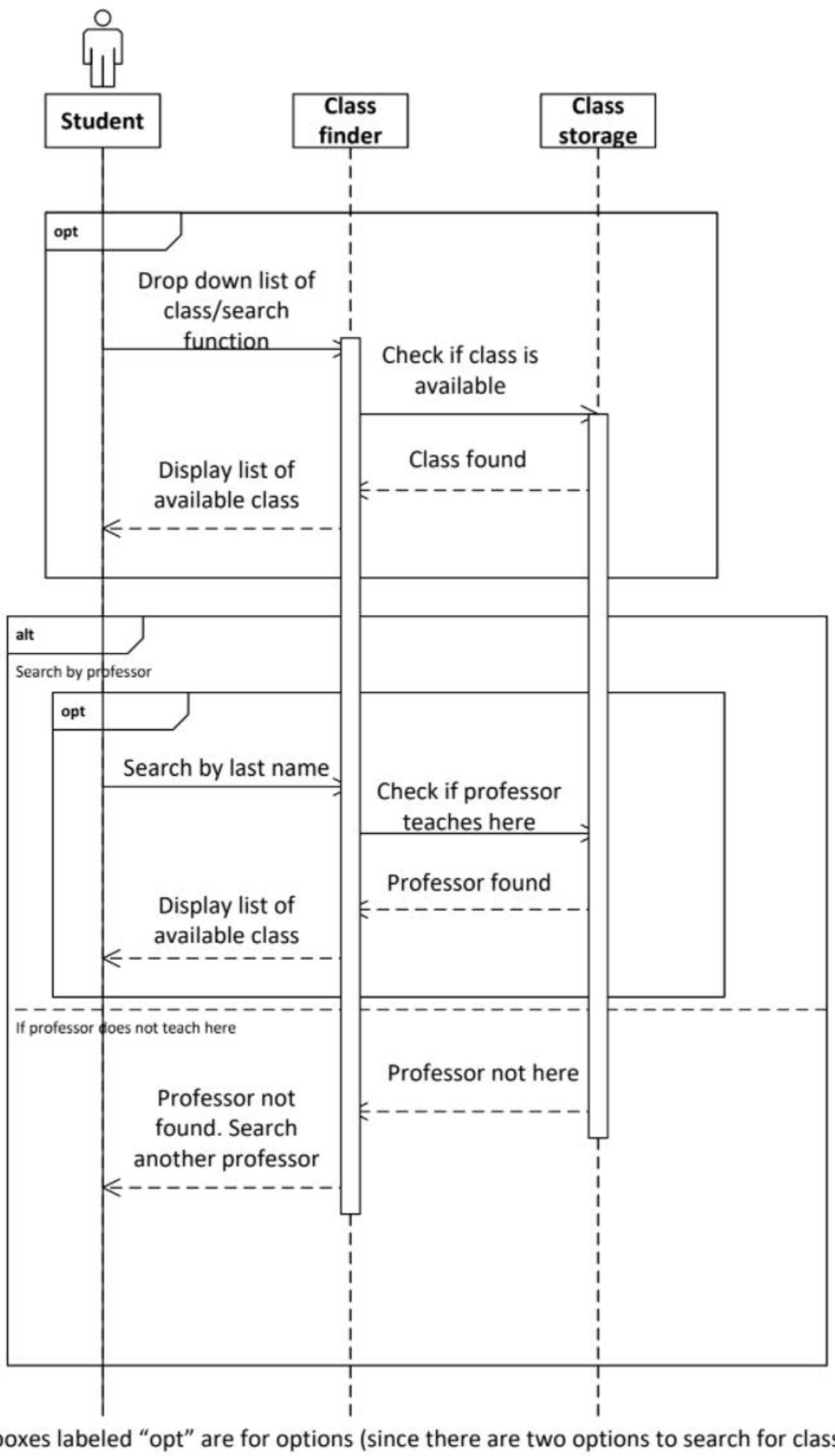




d) Sequence Diagram



Search class feature


Student
Class finder
Class storage
opt
Drop down list of class/search function
Check if class is available
Class found
Display list of available class
alt
Search by professor
opt
Search by last name
Check if professor teaches here
Professor found
Display list of available class
If professor does not teach here
Professor not here
Professor not found. Search another professor

* Two boxes labeled "opt" are for options (since there are two options to search for classes): search by subject and search by professor.

Minimum 2 Design Patterns

a) Design Pattern A: **Observer**
One of the designs that is followed is the behavioral pattern of Observer Design. The Observer design is implemented when looking into the Student and a class was to be dropped or added, then the actual User’s information would have to be updated as well. Therefore, with the iteration of that one instance, the rest of the dependent objects that are relevant to the User (or the Student) would also be updated as well.
The Observer has been implemented in our code throughout many of the objects (specifically the Student, Class Cart, Login, and User). These objects pertain to the Observer pattern as many of these objects would implement each other and changing the object in any way may affect the specific instances that may already be present. Therefore these objects will now carry a new value or instance and would project the same throughout the other implemented objects.

b) Design Pattern B: **Decorator**
Another design that is followed would be the Iterator pattern, pertaining to the login and search aspect of the program. The Decorator design pattern is able to access different capabilities of objects and create a system in a “dynamic” sense, which embodies how the object will be presented. This design is accessible in the login and class search as there are multiple GUI components that are visible.
The Decorator has been implemented in our code specifically pertaining to any GUI interface that may emerge to grab information (Login, Search, Class Cart). In these instances, a new pop up window would emerge taking in inputs provided by the user, all induced from the implementation of Java Swing and JFrame. Example code and interface are posted down below in the Bonus Points (#3).

Bonus Points

1) User Stories
    a) Validate User Story with INVEST and MoSCoW Principles

User Story: As student user, I want to input for login confidentiality so that I can log in my account

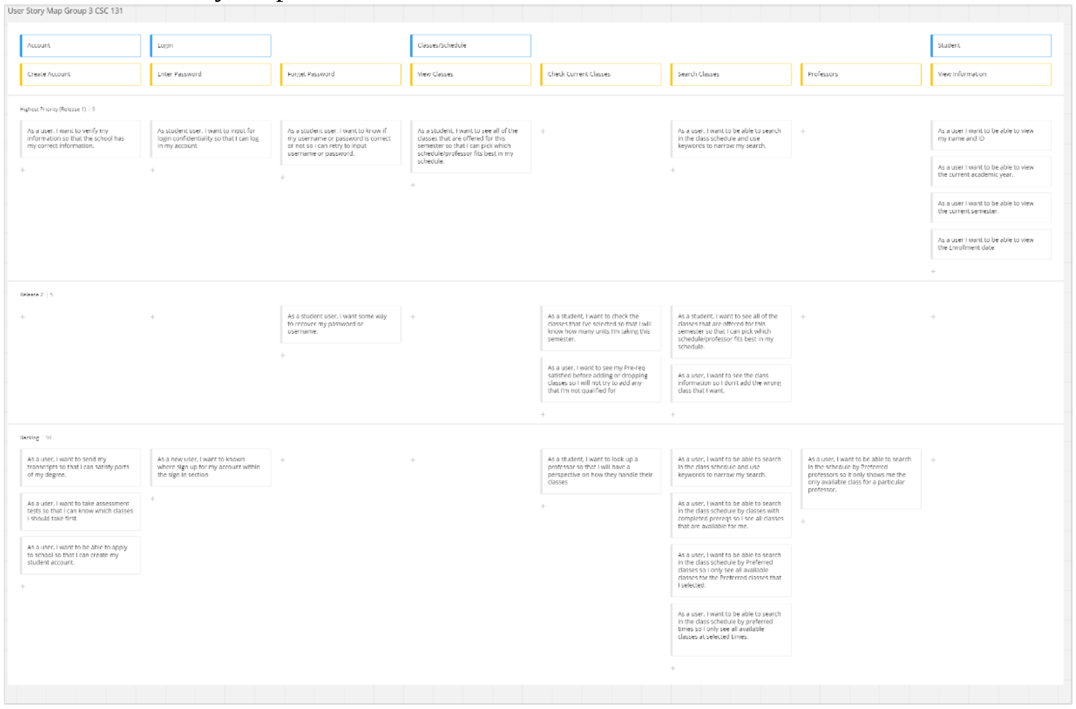
INVEST:

- Independent: This user story is independent as it relies on User's input of login and password
- Negotiable: As it is built on Java Swing and JFrame, different inputs on how to build the interface can be discussed by the Development and stakeholders to develop a functioning product accepted by all parties.
- Valuable: Provides value by creating a bridge between the user and the rest of the functions in the program through the basis of login and password.
- Estimable: Original estimate of 6 hours, was updated to a total of 9 hours
- Small: Was able to be completed by the duration of the first Sprint (4/20/21 → 4/24/21)
- Testable: Satisfies the acceptance criteria of outputting a working GUI interface that responds to Sample Login Accounts created by the Development Team.

MoSCoW

- Must Have: Validating inputs (username and password) by comparing with data stored for account in database. This is important as it is the primary function of this user story.
    - User Story: As a student user, I want to know if my username or password is correct or not so I can retry to input username or password.
- Should Have: System that will allow user to reset account passwords by pulling information (the email) from database to send a link to reset credentials. This function is important, but the product can still function without it. *This has been completed in the second sprint.
    - User Story: As a student user, I want some way to recover my password or username
- Could Have: A secondary validation system (i.e. Security Question and it's answer) that would allow for users to reset credentials using a different method. This is important but not necessary as password can be reset from the 'Should Have' requirement.
- Would Have: The user to have the ability to change username and/or email, but not necessary because our product emphasizes on adding, dropping, and searching classes rather than focusing on the aspect of logging in.

b) Provide User Story Map



User Story Map Group 3 CSC 131

	Account	Login		Classes/Schedule				Student
	Create Account	Enter Password	Forget Password	View Classes	Check Current Classes	Search Classes	Professors	View Information
Highest Priority (Release 1) | 9	As a user. I want to verify my information so that the school has my correct information.	As student user. I want to input for login confidentiality so that I can log in my account	As a student user. I want to know if my username or password is correct or not so I can retry to input username or password.	As a student. I want to see all of the classes that are offered for this semester so that I can pick which schedule/professor fits best in my schedule.		As a user. I want to be able to search in the class schedule and use keywords to narrow my search.		As a user I want to be able to view my name and ID As a user I want to be able to view the current academic year. As a user I want to be able to view the current semester. As a user I want to be able to view the Enrollment date
Release 2 | 5			As a student user, I want some way to recover my password or username.		As a student, I want to check the classes that I've selected so that I will know how many units I'm taking this semester. As a user. I want to see my Pre-req satisfied before adding or dropping classes so I will not try to add any that I'm not qualified for	As a student, I want to see all of the classes that are offered for this semester so that I can pick which schedule/professor fits best in my schedule. As a user, I want to see the class information so I don't add the wrong class that I want.		
Backlog | 10	As a user, I want to send my transcripts so that I can satisfy parts of my degree. As a user, I want to take assessment tests so that I can know which classes I should take first. As a user, I want to be able to apply to school so that I can create my student account.	As a new user, I want to known where sign up for my account within the sign in section			As a student, I want to look up a professor so that I will have a perspective on how they handle their classes	As a user, I want to be able to search in the class schedule and use keywords to narrow my search. As a user, I want to be able to search in the class schedule by classes with completed prereqs so I see all classes that are available for me. As a user, I want to be able to search in the class schedule by Preferred classes so I only see all available classes for the Preferred classes that I selected. As a user. I want to be able to search in the class schedule by preferred times so I only see all available classes at selected times.	As a user, I want to be able to search in the schedule by Preferred professors so it only shows me the only available class for a particular professor.	

c) Include Definition of Done with Acceptance Criteria

Definition of Done:

- Code compiles
- Passes JUnit and EclEmma
- Peer reviewed by rest of the group
- Must pass all Acceptance Criteria specific for the Story
- Product owner must verify it meets the requirements

Acceptance Criteria

User Story: As a user, I want to apply to school so that I can create my student account.

Acceptance Criteria:

- If code compiles, the user should be able to input and submit all the required information (with legal characters).
  - This feature can stand alone and it will lead into the next user story for verification.
- Peer reviewed without any flags.
- Sample users will enter their information and we will see if our system has captured the information correctly by using JUnit and EclEmma.

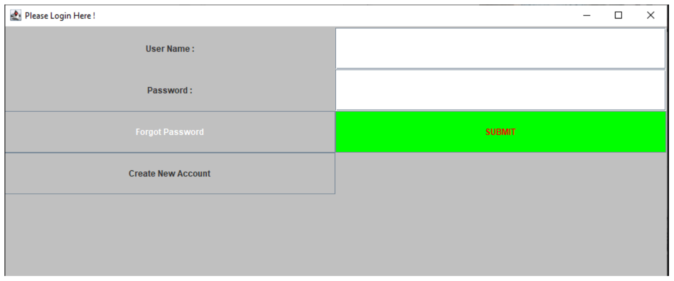
With working code, it would mean that it has compiled and passed both the JUnit and EclEmma test, meaning it would pass the first two bullet points in the Definition of Done (DoD). Since compiling and passing a test doesn't mean that all requirements are met, peer review and Product Owner verification are necessary to satisfy our DoD as well the Acceptance Criteria.

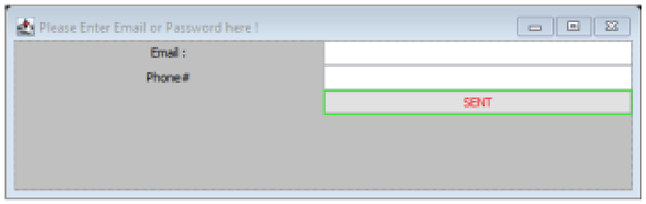
2) Additional 2 Design Patterns in Code

3) Interaction Design Principles and User Interface

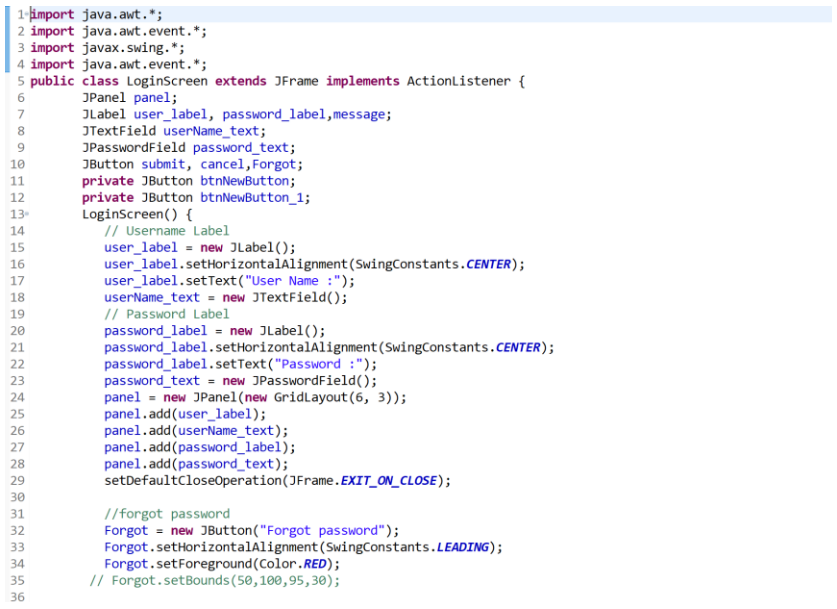
- Used Java Swing

Screenshots:

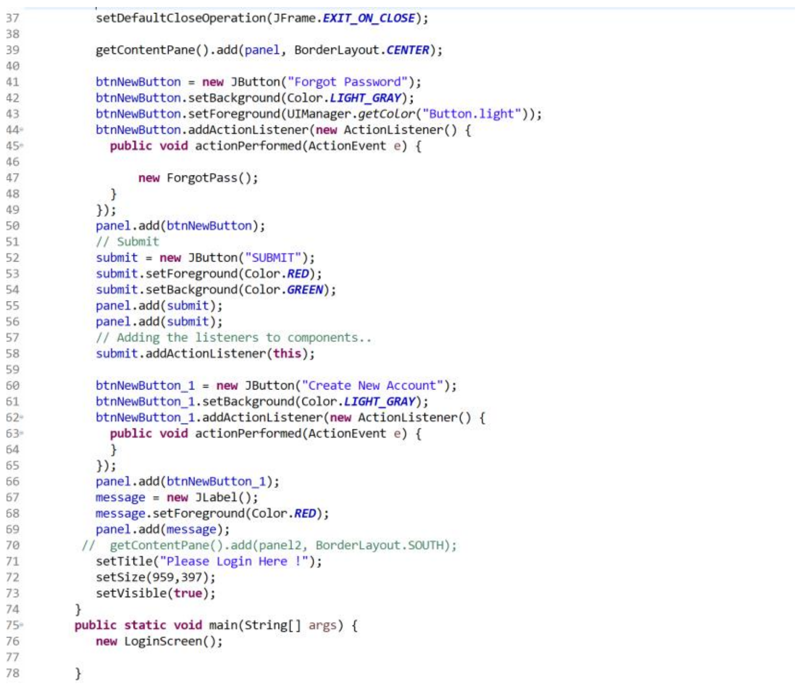




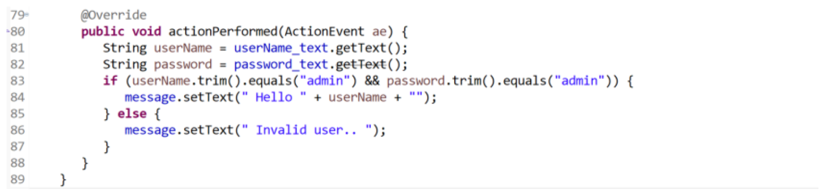
Code:



```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.event.*;
public class LoginScreen extends JFrame implements ActionListener {
        JPanel panel;
        JLabel user_label, password_label,message;
        JTextField userName_text;
        JPasswordField password_text;
        JButton submit, cancel,Forgot;
        private JButton btnNewButton;
        private JButton btnNewButton_1;
        LoginScreen() {
           // Username Label
           user_label = new JLabel();
           user_label.setHorizontalAlignment(SwingConstants.CENTER);
           user_label.setText("User Name :");
           userName_text = new JTextField();
           // Password Label
           password_label = new JLabel();
           password_label.setHorizontalAlignment(SwingConstants.CENTER);
           password_label.setText("Password :");
           password_text = new JPasswordField();
           panel = new JPanel(new GridLayout(6, 3));
           panel.add(user_label);
           panel.add(userName_text);
           panel.add(password_label);
           panel.add(password_text);
           setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

           //forgot password
           Forgot = new JButton("Forgot password");
           Forgot.setHorizontalAlignment(SwingConstants.LEADING);
           Forgot.setForeground(Color.RED);
         // Forgot.setBounds(50,100,95,30);

```



```java
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        getContentPane().add(panel, BorderLayout.CENTER);

        btnNewButton = new JButton("Forgot Password");
        btnNewButton.setBackground(Color.LIGHT_GRAY);
        btnNewButton.setForeground(UIManager.getColor("Button.light"));
        btnNewButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

                new ForgotPass();
            }
        });
        panel.add(btnNewButton);
        // Submit
        submit = new JButton("SUBMIT");
        submit.setForeground(Color.RED);
        submit.setBackground(Color.GREEN);
        panel.add(submit);
        panel.add(submit);
        // Adding the listeners to components..
        submit.addActionListener(this);

        btnNewButton_1 = new JButton("Create New Account");
        btnNewButton_1.setBackground(Color.LIGHT_GRAY);
        btnNewButton_1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
            }
        });
        panel.add(btnNewButton_1);
        message = new JLabel();
        message.setForeground(Color.RED);
        panel.add(message);
      //  getContentPane().add(panel2, BorderLayout.SOUTH);
        setTitle("Please Login Here !");
        setSize(959,397);
        setVisible(true);
    }
    public static void main(String[] args) {
        new LoginScreen();

    }
```



```java
    @Override
    public void actionPerformed(ActionEvent ae) {
        String userName = userName_text.getText();
        String password = password_text.get~~Text~~();
        if (userName.trim().equals("admin") && password.trim().equals("admin")) {
            message.setText(" Hello " + userName + "");
        } else {
            message.setText(" Invalid user.. ");
        }
    }
}
```