Progress Report Two

for

4P02 Project (Course of Action Dashboard)

Prepared by Team Rocket:

Liam Yethon (6255384)
David Saldaña (6155964)
Jason Grightmire (5388327)
Katie Lee (6351696)
Jesse Masciarelli (6243109)
James Zhao (6240527)
Wenjie Lin (6135859)
Calvin Feng (6240790)

Brock University

April 3, 2021

Table of Contents

1. Introduction

1.1 Overview

In the time since our last progress report, Team Rocket has continued development for the Course of Action Dashboard. Our goal was to continue implementing features while taking into consideration the feedback we received from the last report. This document will go over the latest processes and development regarding the design, implementation, and release of the second stage of the project, as well as our next steps and any recorded problems and discussion that may have emerged.

In the Design and Functionality section, we will go over important features added such as drag and drop functionality and saving a student's course schedule to a database. The Implementation section will go over the tools and resources used, as well as the reasons behind why they are believed to be a great fit for this project. The Release section will breakdown our scrum meetings and also include examples of our first iteration for the web application. One piece of criticism we received was a lack of unit testing. This has now been added to our project and is described in more detail towards the end of the report. The Discussions section will go over problems that we encountered during development. The section will also include our plans for future release. Lastly, the User Stories section will contain a table of tasks for the project that have been completed or are still in progress.

2. Design and Functionality

2.1 Drag and Drop Functionality

One of the core components of our system is the course planner tool. The course planner component is designed to allow the user (or student) to visualize and plan out the layout of their academic schedules. In order to achieve this, we knew that we had to provide our users with a method to arrange and adjust their schedules with freedom and simplicity. We decided that the best way to fulfill this need was to take the drag and drop approach, where users can move courses around in whichever order, place them in any semester and year, and add, remove, or restore courses freely.

We first implemented the drag and drop functionality within the course planner itself, given a default, "recommended", schedule provided by Brock University's *2020-2021 Undergraduate Calendar for Computer Science*. The only difference with the schedule provided by the Undergraduate Calendar and the one we generated is that any general credits listed (i.e., 1 Humanities context credit, 0.5 elective credits, etc.) are not included as a drag and drag item. Rather, we decided to omit them from the course planner entirely, and instead, allow the user to search up a particular credit from the course list component and drag it into the course planner.

That being said, the next feature we implemented was the functionality of dragging and dropping a course list item into the course planner. The moment a user drops a course list item into the course planner, the item gets converted into a course planner element, and thus enables its drag and drop functionality.

As of now, when the user grabs a course list item, it holds onto its original appearance, rather than the appearance of a course planner item. The next step (and most likely the last step) of the drag and drop portion, is to adjust the appearance of said item.

2.2 Backend Controllers

Since our Course Planner only consists of a couple pages we were able to group most of the backend methods and functionality using only our "Home Controller". This Home Controller serves as a way for us to be able to connect several aspects of our HTML file with all the information we have in our local database that we implemented. Some of the methods that are located in this controller are: The login functionality, which enables us to check the email and password the user using the application enters with all the entries in our database to make sure the user is a valid student. After a successful verification the student is redirected to the course planner page where their student information will be displayed as well as the current schedule they have planned in the website. Other controller methods include the manual creation of a JSON file that contains the new schedule the student created using the planner and wishes to save to their records. This method allows the application to access the database we are using and create, store, delete and update values based on the desired action.

2.3 Course Scheduler JSON

When looking at different options for us to be able to save a student's schedule which is a complex object with several attributes into our local database we came across the JSON file format. By utilizing the JSON file format this allows us to save an object that contains several attributes that need to be accessed later on as a string into our database. Having this sort of flexibility saves us from having to deal with SQL and local DB errors related to saving complex objects. A JSON object is easily deserialized from the database and serialized from the application by utilizing the proper methods. For us to be able to keep track of a student schedule we decided to do the following: Based on several unique identifiers on the HTML file we are able to identify the different courses the user has selected in their planner as well as the year these courses belong to. After getting this information we save these unique IDs into a data structure and send it to the backend controllers which then look up the selected IDs from the database and gather all the course data. After having all the information from each course we manually created a JSON file which stores all the courses and the year they belong too. Finally these newly created JSON files are stored into the users schedule database entry.

2.4 Dynamic Progress Tracking

An update to a key design element seen in our first release is the dynamic functionality of the student credit tracking and progress bar section at the top of the page. In our first iteration this feature was designed and tested through manual data input, but did not dynamically pull data from the students schedule. Now that we have the ability to change and save student schedules in this release, we are also able to use this data and display it through the progress tracker. It can be seen that statistics such as overall credits, humanities credits, first year credits etc. can be seen whenever a student saves their schedule and refreshes the statistics and progress session. These stats are pulled directly through the student's schedule which is stored in the projects database.

The next steps for this feature are to have it update automatically every time a student makes a change to their schedule, rather than only when they choose to save and refresh it. We plan to implement this feature by utilizing our schedule saving function every time a new course item is dropped into the schedule, and adding the progress bar update as a part of this saving sequence. This feature makes great use of our database functionality and presents user data in an easy to interpret, organized fashion.

2.5 Course Planner

Revisiting the course planner, we've implemented some additional features that would make the tool more user friendly. The first thing we implemented aside from the drag and drop functionality was its ability to add additional year containers. With a default and minimum duration of four years, the student is able to add additional years with the maximum capacity of ten. Any additional year added after the fourth year can also be removed (along with any courses that are listed within that year container).

As mentioned above in section 2.3, we implemented the ability to save a student's schedule. To put this feature up to use, we added a new 'Save' button to the course planner that will enable the student to save their schedule as is.

Another button we added was the 'Reset' button, which will refresh and populate the student's course planner with the default, recommended schedule for their given program (mentioned in section 2.1).

As for the course planner elements, we also implemented a feature that highlights any duplicate items found within the user's schedule. Rather than preventing the user from adding duplicate courses to their planner, we wanted to allow the user to have as much freedom as possible when planning things out.

The next steps for each feature mentioned above are as follows:

- 'Save' button: provide the user with feedback confirming that their schedule has been successfully saved (for instance, with a Sweetalert2 notification).
- 'Reset' button: before resetting, warn the user that their current progress may be lost without saving and confirm if they would like to load the default, recommended schedule. If the user confirms, the reset will occur, otherwise, the reset will get canceled.
- Highlighting duplicate items: integrate tooltips for any duplicate items. If a duplicate exists within the student's schedule, then they may hover over a duplicate item (indicated by the highlight) to display the tooltip, describing the issue.

Some additional features that we are considering in our next steps are as follows:

- Integrate tooltips for any other flaws within the user's schedules. For instance, if a user has more than the maximum number of credits planned for a year, then the credits which cause the overflow would be highlighted, and when hovered, the user can read its tooltip describing the flaw.
- 'Export' button: when selected, the layout of the student's schedule will get exported in some format (i.e., in a table - printed as a '.pdf' file). Currently, we have the 'Export' button displayed on the course planner menu. We would basically have to carry out its functionality.
- Schedule library/bank: have a library/bank of previously saved schedules, including the default, recommended schedule. We would like to allow the user to revisit any previously made plans and give them a sense of control over knowing the integrity of the schedules that they are working with.

2.6 Course List

Moving onto the course list component, users are able to drag an item from the course list and drop it into any section in the course planner. Courses that a user already has present within their course planner will be indicated with a highlight within the course list. For example, if a user has 'COSC 1P02' present in their course planner, the 'COSC 1P02' course list item will then be highlighted in green. The purpose of this feature was to improve the user experience by providing more important information to the user. As of now, everything on the course list is working well as we expected with the exception of the issues surrounding the styling of the course list and its components. That being said, the next steps include improving the visual user interface of the component, modifying the information that is displayed to the user in each course list item (i.e., rather than just displaying the course subject and code, on hover, we will be displaying its subject, code, and name). For instance, if a user hovers over the item, "COSC 1P02", they will get presented with the name, "COSC 1P02; Introduction to Computer Science".

3. Implementation

3.1 Local Database

In order to store information about the course, including prerequisites and the course context, we decided to use the local database included in Visual Studio. The course database can be accessed just like a hosted database, however it requires each person to set up the database individually when they open the project. This has an added benefit of allowing us to change individual values when needed for testing without causing any potential errors to affect the entire team. We also have a student database stored in the same location as the course database, which holds the students login credentials, full name and schedule that can be accessed from a later date.

To store the student's schedule we will be using JSON to store a string that can be deserialized into an object inside of the Home Controller when accessed. We decided to use this as it is very simple and frees us from having to map out a complicated database schema, since our goal for the database is to just hold student login information and a list of offered courses.

3.2 SweetAlert2

After looking for some libraries that would enable us to obtain customized pop ups we decided to utilize the SweetAlert2 library. SweetAlert2 enables us to create and customize different types of pop ups that can be used to display warnings, errors, information and even prompt the user with choices. We will be using these pop ups to display several warnings regarding course and schedule restrictions such as course prerequisites, schedule inconsistencies (i.e duplicate courses), etc. For us to be able to install these libraries into our application we were only required to link the libraries CDN link in our Layout page which enables our entire application to make use of these libraries.

4. Release

4.1 Scrum Meetings

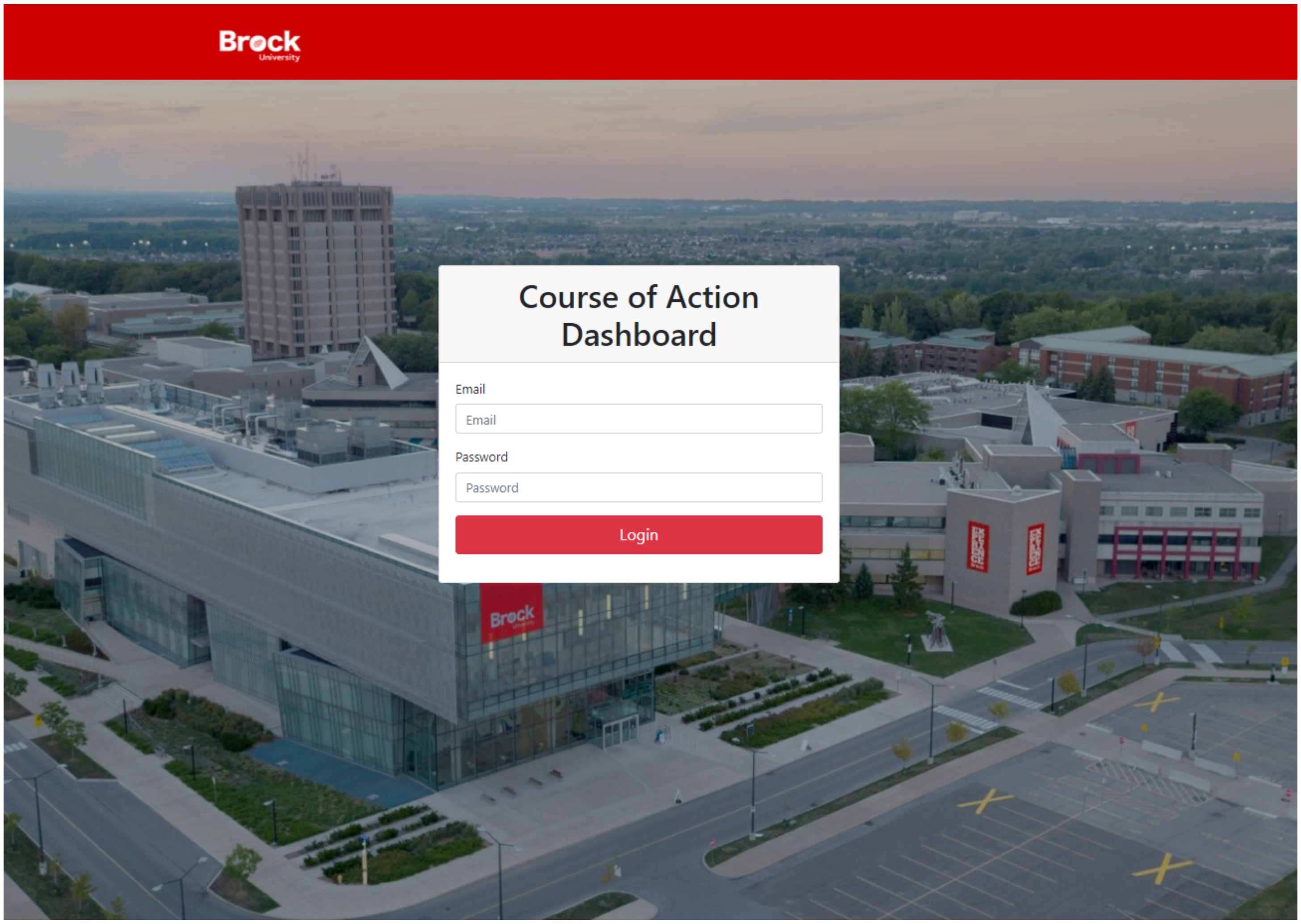
During our scrum meetings for the second sprint, each task assigned to a person or a group of people are discussed, such as how the progress is going, what issues were encountered and also the goals for the week before the next scrum meeting. In the second sprint, we planned to implement the database and have the front end of the project utilize the data and present it to the user. As we are nearing the end of the completion of the project, most of our tasks link to each other which require further communication. The scrum meeting allowed us to quickly group these similar tasks and have the involved group members explain and help each other to finish the entire section of the project.

4.2 Second Sprint

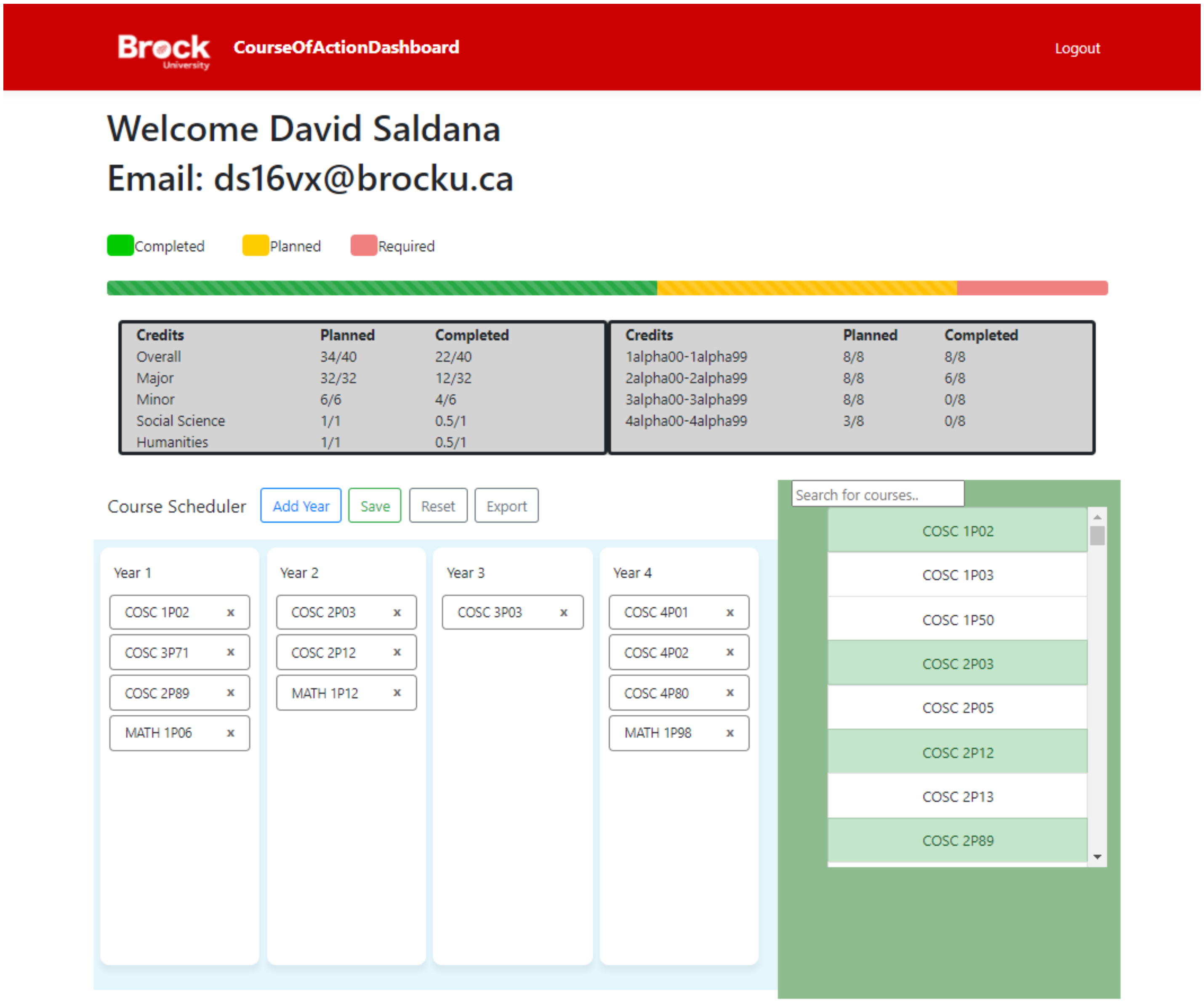
For our second sprint, we have assigned new tasks to each member to work on throughout the month. Each week, we would have a scrum meeting discussing the progress each member has made. Once someone has done their task, we would update our current version with the feature implemented. Afterwards, we would review our current state and make sure some tasks were done for next week such as the drag and drop feature. An example of this can be seen with the database. The connection has been made to each component of our application such as the course list and login page. For the login page, we are able to verify if the email and password exist within our database. If it does, it will redirect them to the course planner where they can start planning out their desired schedule. By the end of our second sprint, we have completed the majority of the functionality.

4.3 First Functional Version

- **Login Page**



- **Course Planner Page**



Welcome David Saldana
Email: ds16vx@brocku.ca

🟩Completed 🟨Planned 🟥Required

Credits	Planned	Completed
Overall	34/40	22/40
Major	32/32	12/32
Minor	6/6	4/6
Social Science	1/1	0.5/1
Humanities	1/1	0.5/1

Credits	Planned	Completed
1alpha00-1alpha99	8/8	8/8
2alpha00-2alpha99	8/8	6/8
3alpha00-3alpha99	8/8	0/8
4alpha00-4alpha99	3/8	0/8

Year 1	Year 2	Year 3	Year 4
COSC 1P02 ×	COSC 2P03 ×	COSC 3P03 ×	COSC 4P01 ×
COSC 3P71 ×	COSC 2P12 ×		COSC 4P02 ×
COSC 2P89 ×	MATH 1P12 ×		COSC 4P80 ×
MATH 1P06 ×			MATH 1P98 ×

COSC 1P02
COSC 1P03
COSC 1P50
COSC 2P03
COSC 2P05
COSC 2P12
COSC 2P13
COSC 2P89

5. Discussion

5.1 Problems

As for problems, most of the issues we encountered while developing this application were related to being able modify student data inside the local database we implemented in the application. At first we were able to retrieve all the data but we lacked the knowledge required to modify it and save it. Thankfully we were able to resolve this issue by doing some research and learning a couple SQL related methods that ASP.NET applications are able to utilize to save changes made to entries.

Other problems we encountered were related to the database and the creation of the several tables we needed such as the course table. The first iteration of the SQL course table didn't work properly because the table entry names didn't match the ones implemented in our Course.cs class which caused an issue when trying to access the information. Aside from issues related to the database, we've encountered some issues surrounding the styling of the course list component. On grabbing a course list item, there is an issue with altering its default appearance in order for it to blend in with the consistent theme of the site.

5.2 Future Release Plan

Our main focus for this first functional version was to get the main functionality of the course planner working. Our current version now allows users to login, create and save schedules which now enables us to change our focus on adding additional functionality such as warnings and allowing users to have several planned schedules in their account.

Our next steps moving forward are as follows:

- Work on displaying warnings whenever schedule restrictions take place in the users course planner.
    - <u>How we will do this</u>: we will be generating and compiling a list of all possible errors or issues that the user may run into with regard to their active course schedule. Once the errors are generated and compiled, we will then develop the trigger functions that will execute once a particular case arises. This will be displayed within the Error/Warning section of the page.
- Implement the "Export" functionality that will enable users to export the schedule they planned to an excel document.
    - <u>How we will do this</u>: we will be reading the course IDs of each element from the student's course planner and pulling the necessary data needed from the database to populate the exported document. We will then be considering the methods of

    which the data will be presented to the user visually and will carry out with the method of choice.

- Work on some design/styling related issues to make our final design look how we envision it.
    - How we will do this: we will come up with various user interface design drafts and decide which provides the best user experience. Once a design gets approved, we will work on adjusting and/or implementing the design of choice.
- Fix minor bugs related to saving student schedules.

6. Unit Testing

For the unit tests, all methods inside of the Home Controller that had a substantial body were tested for relevant cases. Visual Studio provides some built in testing tools, including IntelliTest, which provides an easy to use test suite that can be run quickly at any point. Visual Studio also provides live unit testing, which will run the unit tests whenever the corresponding code is changed in the main project.

Name	Input	Expected	Output	Notes
Index Null Test	Student = null	NullReferenceException	NullReferenceException	Works as intended.

Save Schedule Confirm Test	studentId = 1005
	Courses = {{1,2}, {3,4}, {5,6}, {7,8}}
		Method Success	Method Success	Works as intended.

Save Schedule Null Test	studentId = 1005
	Courses = null
		NullReferenceException	NullReferenceException	Works as intended.

Save Schedule Invalid Parameters Test	studentId = 1005
	Courses = {{-1}}
		Argument Exception (Should not have course with CID of -1)	Method Success	Test not working as intended, potentially due to usage of FirstOrDefault when accessing database.

Login Confirm Test	Email = "test@brocku.ca"
	Password = "test"
		redirectResult = null (Method Success)	redirectResult = null (Method Success)	Works as intended, login page returns Index with student found through database on success.

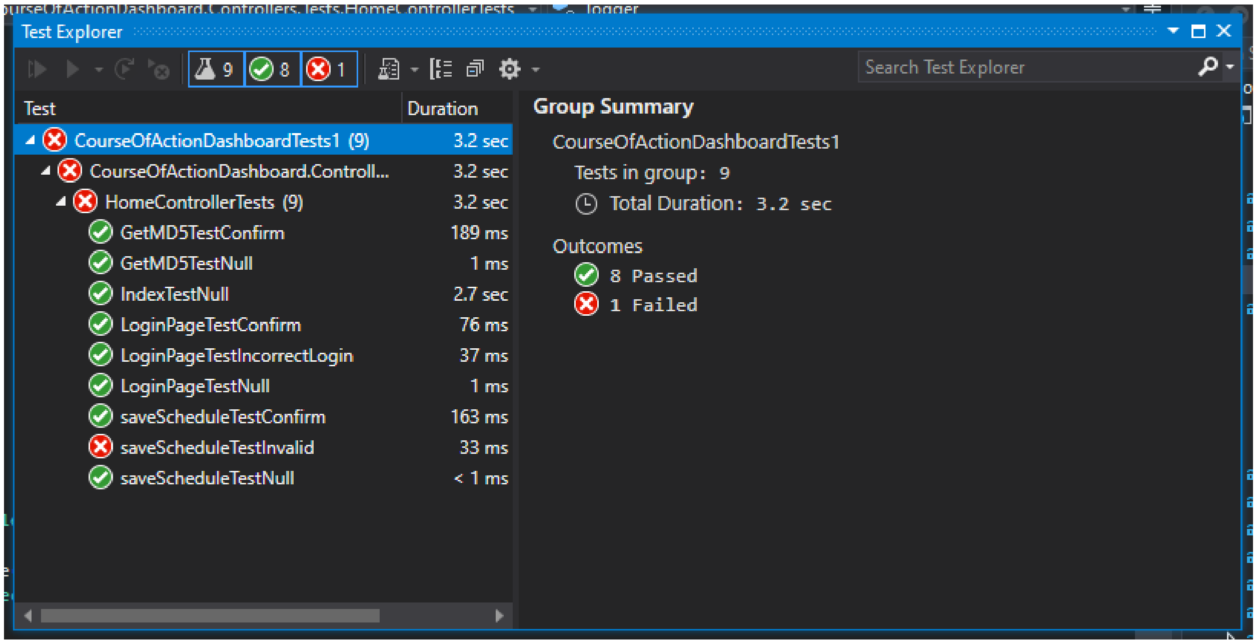
Login Null Test	Email = null
	Password = null
		redirectResult != null	redirectResult != null	Works as intended, login page returns RedirectToAction object if student object could not be found.

Login Invalid Parameters Test	Email = "test@brocku.ca"
	Password = "wrong"
		redirectResult != null	redirectResult != null	Works as intended, see above.

MD5 Hash Confirm Test	testInput = "testing"
	testOutput = "ae2b1fca515949e5d54fb22b8ed95575"
		Method Success, output = testOutput	Method Success, output = testOutput	Works as intended.

MD5 Hash Null Test	testInput = null	Argument Null Exception	Argument Null Exception	Works as intended.

As of this progress report, all but one of the unit tests pass without issue. More work is needed to resolve the Save Schedule Invalid Parameters Test as it does not appear to be functioning correctly. As stated in the notes column, this is most likely due to the usage of the FirstOrDefault method when accessing a given course in the database by its CID. This will either require a test rewrite or the removal of the FirstOrDefault method.



```
Test Explorer
9  ✓ 8  ✗ 1                                   Search Test Explorer

Test                                    Duration   Group Summary
✗ CourseOfActionDashboardTests1 (9)     3.2 sec      CourseOfActionDashboardTests1
  ✗ CourseOfActionDashboard.Controll... 3.2 sec        Tests in group: 9
    ✗ HomeControllerTests (9)           3.2 sec        Total Duration: 3.2 sec
      ✓ GetMD5TestConfirm               189 ms
      ✓ GetMD5TestNull                    1 ms       Outcomes
      ✓ IndexTestNull                   2.7 sec        ✓ 8 Passed
      ✓ LoginPageTestConfirm             76 ms         ✗ 1 Failed
      ✓ LoginPageTestIncorrectLogin      37 ms
      ✓ LoginPageTestNull                 1 ms
      ✓ saveScheduleTestConfirm         163 ms
      ✗ saveScheduleTestInvalid          33 ms
      ✓ saveScheduleTestNull           < 1 ms
```

Pictured above is an example run of the Visual Studio testing suite.

7. User Stories

Note that this table summarizes the major user stories covered during this phase of the project. An extensive list of completed issues: https://github.com/LYethon/COSC4P02Project/issues

Component	Description	Status	Notes
Warnings/Errors	Recognize and throw warnings based on the user's schedule (i.e. throw a warning if a user has duplicates present in their schedule).	In progress	Includes compiling the types of errors that a user could encounter and generating the trigger functions that will report said errors to the display.
Course Planner	Course Planner - implement the functionality of the Export Button	In progress	Building the functionality.
Course Planner	Course Planner - implement Tooltips for duplicate courses present in the planner as well as tooltips for any highlighted courses that cause max capacity overflow for each year.	In progress	See section 2.5
Course Planner	Course Planner - fix the Reset Button	In progress	The reset button has some minor bugs that need some fixing.
Course Planner	Course Planner - organize Sections into sub components/containers (i.e. for the fall and winter session and the spring and summer session)	In progress	For better organization.
Course List	Course List - update the UI (visual design)	In progress	
Search Bar Filter	Implement the ability for the course list to filter through matching results	Complete	Filters by search term, not yet by type etc.
Mapping out Database	Create the tables and keys necessary to store student and course data	Complete	
Course search list scrollability	Allow a user to scroll through the results of the searchable courses	Complete	

Drag and drop courses within schedule	Allow a user to move courses to different years and positions within their schedule	Complete	
Drag and drop courses from searchable lost	Allow a user to drag course objects from the searchable list into their schedule	Complete	Can use cosmetic changes
Course Scheduler - button to reset course schedule	Allow a user to reset their schedule back to the default recommended schedule	Complete	This feature may be reformatted, but is functional.
Course Scheduler - add year function	Give the user the ability to add more years to their course planner	Complete	
Frontend database communication	Implement Database Update Functionality From Frontend	Complete	

8. Recorded Meetings

Meeting 7 - https://web.microsoftstream.com/video/1e3c0551-c995-4fea-a445-ee3b89e44f50
Meeting 8 - https://web.microsoftstream.com/video/4696b156-5f9c-43de-87e4-73242c0e3bad
Meeting 9 - https://web.microsoftstream.com/video/900e1a0b-b1e7-4d97-bf41-5ea354757edc
Meeting 10 - https://web.microsoftstream.com/video/b9cf5e64-9fca-4d9f-8583-1fb7145a07e0