# SOSY Scheduler

Requirements and Specification Document
12/03/2024, Iteration 1
24/03/2024, Iteration 2
05/04/2024, Iteration 3

## Project Abstract

SOSY Scheduler is an online web application catered for students in the Software Systems (SoSy) Major at Simon Fraser University. Building upon the foundation of their current course offerings display (https://sfussss.org/courses), SoSy Scheduler enhances the existing webpage by providing personalized degree planning. Students can view required SOSY courses, track academic progress, and preview upcoming courses for semester schedules. The planning experience that this application provides aims to alleviate the stress of a student's university journey by facilitating an all-in-one seamless navigation.

## Competitive Analysis

GoSFU's MySchedule tool is the primary schedule planning platform for SFU students. It offers upcoming semester courses, visual scheduling, and the ability to generate multiple schedules for any SFU student. However, MySchedule is cluttered with course data from all SFU departments and requires users to navigate back through GoSFU to see their overall degree requirements. Addressing these issues, SOSY Scheduler restricts courses planning to required SOSY courses and provides a reference checklist all in one place.

## Customer

Organization: SSSS is a student-led society at Simon Fraser University, where it helps students and faculty by addressing issues and concerns between the students and faculty, building community in and around our society, and sharing resources provided by students and external organizations.
Name: Joshua Li
Role: President of Software Systems Student Society (SSSS)
Email: joshua_li@sfu.ca

## User Stories

**Type of User:** Student User (SFU student in Software Systems Major)

**Story:** A regular student wants to log in.
**Iteration:** 1
**Subtype:** Login

**Triggers/PreConditions:** From the root page, press on login.
**Actions/Postconditions**: If successful login, take to their landing page. If unsuccessful login, return and notify the user.
**Tests:** In Javascript, *pattern.test()* ensures that the login email ends with @sfu.com. In the Java Controller, *userList.isEmpty()* tests whether a valid login email and password have been inputted. Users are notified of errors by the front-end.

**Type of User:** Student User (SFU student in Software Systems Major)
**Story:** A regular student wants to log out.
**Iteration:** 2
**Subtype:** Logout
**Triggers/PreConditions:** The user must have had a successful login (see story above). The user then clicked on the logout.
**Actions/Postconditions:** The user's session is successfully terminated. The page is then rerouted back to the index.
**Tests:** As a precondition, the program must test that it is in a valid session by *testSessionUser(*). The logout *session.invalidate()* command implicitly tests whether the logout was successful. Users are notified by returning to front-end

**Type of User:** Student User (SFU student in Software Systems Major)
**Story:** A regular student wants to view a list of completed and uncompleted SOSY courses specified as such.
**Iteration:** 2
**Subtype:** Course Viewing
**Triggers/PreConditions:** The user must be logged in.
**Actions/Postconditions:** The user has a list of completed and uncompleted SOSY courses that they can update as complete and incomplete (see stories below).
**Tests:** As a precondition, the program must test that it is in a valid session by *testSessionUser(*). As an invariant condition, the program must test that the completed and uncompleted SOSY list make up the complete SOSY list. This test is implemented by ensuring the completed list is always constructed and then constructing the incomplete list from it at runtime. If an error occurs, users are returned to the home page.

**Type of User:** Student User (SFU student in Software Systems Major)
**Story:** A regular student wants to mark a class as complete.
**Iteration:** 2
**Subtype:** Course Viewing
**Triggers/PreConditions:** From the list page (see story above), the user clicked on an incomplete class from their course list.

**Actions/Postconditions:** The user's account with the newly completed course is updated. The view of completed and uncompleted SOSY courses is reloaded (see story above).
**Tests:** As a precondition, the program must test that it is in a valid session by *testSessionUser()*. As an invariant condition, the program must test that the completed and uncompleted SOSY list make up the complete SOSY list. This test is implemented by ensuring the completed list is always constructed and then constructing the incomplete list from it at runtime.  If an error occurs, users are returned to the list page.

**Type of User:** Student User (SFU student in Software Systems Major)
**Story:** A regular student wants to mark a class as incomplete.
**Iteration:** 2
**Subtype:** Course Viewing
**Triggers/PreConditions:** From the list page (see story above), the user then clicked on a complete class from their course list.
**Actions/Postconditions:** The user's account is updated removing the completed course. The view of completed and uncompleted SOSY courses is reloaded (see story above).
**Tests:** As a precondition, the program must test that it is in a valid session by *testSessionUser()*. As an invariant condition, the program must test that the completed and uncompleted SOSY list make up the complete SOSY list. This test is implemented by ensuring the completed list is always constructed and then constructing the incomplete list from it at runtime. If an error occurs, users are returned to the list page.

**Type of User:** Student User (SFU student in Software Systems Major)
**Story:** A regular student wants to register a new account.
**Iteration:** 2
**Subtype:** Registration
**Triggers/PreConditions:** From the root page, click on registration.
**Actions/Postconditions:** User inputs unique email and passwords and submits. If successfully created, notify and reroute to login (see story above). If not successfully created, route back to the registration page.
**Tests:**  In Javascript, *pattern.test()* ensures that the registered email ends with @sfu.com. and a password that includes an uppercase, lowercase, and a number . In the Java Controller, *userList.isEmpty()* tests whether the inputted username is taken. Users are notified of error or success by the front-end.

**Type of User:** Student User (SFU student in Software Systems Major)
**Story:** A regular student wants to view all of the courses offered in the next semester.
**Iteration:** 3
**Subtype:** Schedule

**Triggers/PreConditions:** From the list page (see story above), click on the schedule courses page.
**Actions/Postconditions:** The user navigates to the scheduled courses page with a checkbox form of all SOSY courses offered in the upcoming semester.
**Tests:** As a precondition, the program must test that it is in a valid session by *testSessionUser()*. The list of SOSY courses is directly inputted into the program and thus will always generate correctly. If an error occurs, users are returned to the list page.

**Type of User:** Student User (SFU student in Software Systems Major)
**Story:** A regular student wants to see their course schedule given a set of selected courses.
**Iteration:** 3
**Subtype:** Schedule
**Triggers/PreConditions:** From the scheduled courses page (see story above), user selected some number of courses and submitted them as a form.
**Actions/Postconditions:** The user navigates to the chosen courses page, where a weekly lecture schedule is displayed for all courses selected in the form. Any courses that have overlapping lectures (including overlapping by travel time) are listed under the day they overlap.
**Tests:** As a precondition, the program must test that it is in a valid session by *testSessionUser()*. The list of SOSY courses is directly inputted into the program and thus will always generate correctly. If an error occurs, users are returned to the list page.

**Type of User:** Student User (SFU student in Software Systems Major)
**Story:** A regular student wants to save their schedule as a PDF File
**Iteration:** 3
**Subtype:** File Download
**Triggers/PreConditions:** From the chosen courses page, click on the download button.
**Actions/Postconditions:** After the button is pressed, the user will be redirected to the file.io website (web API) with a PDF of their schedule loaded and ready for download.
**Tests:** As a precondition, the program must test that it is in a valid session by *testSessionUser()*. As a postcondition, the code must log the user schedule output for debugging along with providing the user with a correct PDF file of their schedule. Users are notified of any errors in creating or uploading the PDF File.

**Type of User:** Administrator
**Story:** An administrator wants to log into their account
**Iteration:** 1
**Subtype:** Login.
**Triggers/PreConditions:** From the root page, press on login.

**Actions/Postconditions**: If successful login, take to their landing page. If unsuccessful login, return and notify the user.
**Tests:** In Javascript, *pattern.test()* ensures that the login email ends with @sfu.com. In the Java Controller, *userList.isEmpty()* tests whether a valid login email and password have been inputted and *user.isAdmin()* tests whether the user is an admin. Users are notified of errors by the front-end.

**Type of User:** Administrator
**Story:** An administrator wants to log out.
**Iteration:** 2
**Subtype:** Logout
**Triggers/PreConditions:** The admin must have had a successful login (see story above). The admin then clicked on the logout.
**Actions/Postconditions:** The admin's session is successfully terminated. The page is then rerouted back to the index.
**Tests:** As a precondition, the program must test that it is in a valid session by *testSessionUser()*. The logout *session.invalidate()* command implicitly tests whether the logout was successful. Users are notified by returning to front-end

**Type of User:** Administrator
**Story:** An administrator wants to see the current list of users in the database.
**Iteration:** 2
**Subtype:** Administration
**Triggers/PreConditions:** The admin must have had a successful login (see story above).
**Actions/Postconditions:** The admin is provided with a list of current users in the system.
**Tests:** As a precondition, the program must test that it is in a valid session by *testSessionAdmin()*. If an error occurs, users are returned to the index page.

**Type of User:** Administrator
**Story:** An administrator wants to remove a user.
**Iteration:** 2
**Subtype:** Administration
**Triggers/PreConditions:** From the list of users, the admin clicked on a user.
**Actions/Postconditions:** The database is updated removing the user. The view of users is reloaded (see story above).
**Tests:** As a precondition, the program must test that it is in a valid session by *testSessionAdmin()*. The program tests that the chosen user is in the database and is not the admin by *userlist.isEmpty()* and *toRemove.isAdmin()*. If an error occurs, users are returned to the list of users.

## Iteration 3 Major Features

Flowing in over from iteration 2, the logout feature is now fixed to prevent users from navigating back to there content after logging out.

Each user now can generate a weekly lecture schedule for the upcoming term given a list of SOSY courses. The scheduled courses pages contain a checklist of all SOSY courses offered in the upcoming term that the user can select for their schedule. Once submitted, a weekly schedule is displayed with overlapping lectures (either by time or distance and time) marked as such. The plan was originally to improve on GoSFU by allowing multiple semester scheduling, however, at closer inspection it was revealed that courses are not decided for more than 1 semester at a time. On the schedule display, there is a download button that redirects and uses a Web API to generate a PDF version of the current page that is available to download on file.io.

Tests were also updated this iteration. For any method in the controller, the session user was explicitly verified and an incorrect user error was handled by returning to the index page. All other back-end tests were in the try…catch blocks that only can only fail by programming or database errors. If such an error occurs, the user is continually redirected back to the last safe page. User registration and login tests were altered to change HTML instead of sending an window alert for a smoother user interface. On the front-end, an error in the Web API and a navigation to an external page still sends a window alert.

Additionally, the website's design was updated.

## Iteration 3 Velocity

As mentioned at the end of Iteration 3, the development iteration 3 was far quicker than the previous iterations. There are only technically 3 new story points from a user point of view but both the creation of the schedule and its extraction to a PDF required a considerable more amount of programming (see files under courseScheduling). Additionally, team members continually updated documentation, tests, and design this iteration to produce a polished website.

## Iteration 2 Major Features

Flowing over from iteration 1, the login/logout/register feature is now complete with session verification. The admin account has the username adminHere@sfu.com and the password Hello 123. Please create your new account with a random sfu.com email to test internal functionality.

For the administrator, a current list of users is dynamically provided. From this list, the administrator can click on a user to remove them from the database. The administrators themselves cannot be removed.

A current list of all SOSY courses completed and uncompleted is provided for the users. From this list, users can click on an uncompleted course to log it as completed. Additionally from this list, users can click on completed courses to log them as uncompleted. This information is saved and carries over from session to session. All SOSY courses are either completed or uncompleted.

## Iteration 2 Velocity

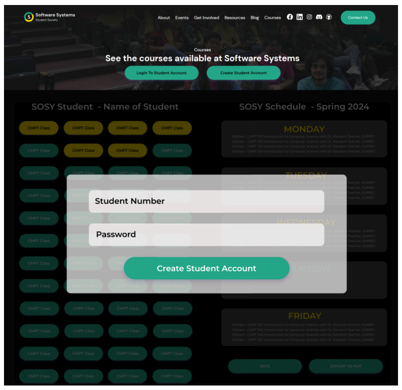
The velocity for Iteration 2 sped up quite considerably from Iteration 1 as there are now 8 additional story points completed in this iteration all of which have set up the back-end data needed to create the front-end scheduling view for Iteration 3.  The velocity for Iteration 3 is estimated to be quite quicker since it requires less information from the database. The velocity of the development increased as the team members became more comfortable working on different parts of the project simultaneously and the bottleneck of setting up the system was cleared.

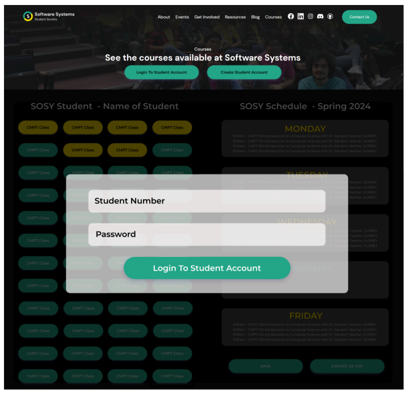
Development briefly slowed when the Render database and site thus delaying pushes to git. Team Members instead developed functionalities on local devices using locally saved data before pushing to git after Render.com recovered.

## User Interface Requirements

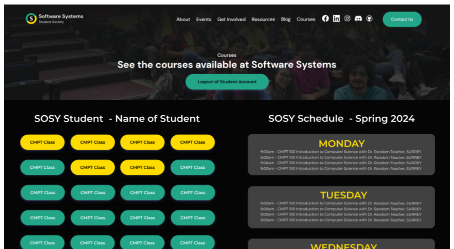
The User Interface must have all of the following views/methods on the site.

1. A method to log in and create a new user. In the implementation, this method can be accessed via two buttons on the main page of the site.


Software Systems
Student Society
About Events Get Involved Resources Blog Courses
Contact Us
Courses
See the courses available at Software Systems
Login To Student Account
Create Student Account
SOSY Student - Name of Student
SOSY Schedule - Spring 2024
CMPT Class
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
9:00am - CMPT 100 Introduction to Computer Science with Dr. Random Teacher, SURREY
Student Number
Password
Create Student Account
SAVE
EXPORT AS PDF



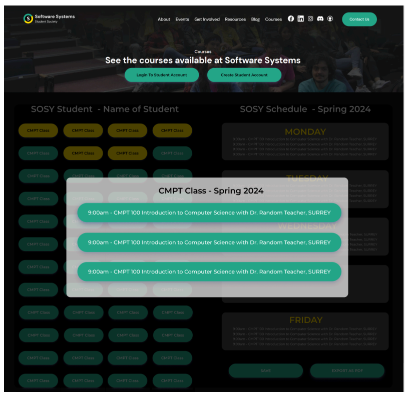
2. A view to see a complete list of courses completed and to complete a Software Systems degree. In this implementation,


Software Systems
Student Society
About Events Get Involved Resources Blog Courses
Contact Us
Courses
See the courses available at Software Systems
Logout of Student Account
SOSY Student - Name of Student
SOSY Schedule - Spring 2024
CMPT Class
MONDAY
TUESDAY
WEDNESDAY
9:00am - CMPT 100 Introduction to Computer Science with Dr. Random Teacher, SURREY

a list of courses as individual buttons is displayed on the main page with completed courses in yellow and uncompleted courses in green.

3. A view to see classes scheduled over multiple different academic calendar terms. In this implementation, a weekly schedule shows selected classes with timings for a certain term.

4. A method to save and export the created schedule. In the implementation, a button is used to generate either result from the back-end.

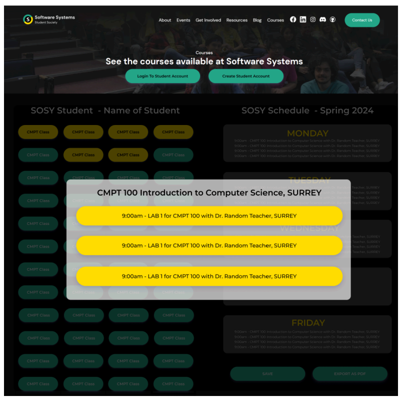


CMPT Class - Spring 2024

9:00am - CMPT 100 Introduction to Computer Science with Dr. Random Teacher, SURREY

9:00am - CMPT 100 Introduction to Computer Science with Dr. Random Teacher, SURREY

9:00am - CMPT 100 Introduction to Computer Science with Dr. Random Teacher, SURREY



CMPT 100 Introduction to Computer Science, SURREY

9:00am - LAB 1 for CMPT 100 with Dr. Random Teacher, SURREY

9:00am - LAB 1 for CMPT 100 with Dr. Random Teacher, SURREY

9:00am - LAB 1 for CMPT 100 with Dr. Random Teacher, SURREY

5. A method to view and select from all lecture and lab offerings of a given course for a given term. In the implementation, users clicking on a course (2) while in a given term (3) will result in a list of lecture offerings followed by lab offerings in that semester.