SRU Class Scheduler

Final Report



April 25, 2022

Slippery Rock University

Autumn Kriebel alkriebel36@gmail.com

Katherine Settle katuesettle77@gmail.com

Dean Lisko dlisko2526@gmail.com

Group Number: 5

Class: CpSc488 section 01

Problem: Efficient and Accurate Class Scheduling

Contents

[Contents i](#_Toc101808376)

[1 References 1](#_Toc101808377)

[2 Completion 2](#_Toc101808378)

[2.1 Contribution 2](#_Toc101808379)

[2.1.1 Percentage Breakdown 2](#_Toc101808380)

[2.1.2 Explanation 2](#_Toc101808381)

[2.2 Complete Modules 2](#_Toc101808382)

[2.3 Incomplete Modules 3](#_Toc101808383)

[2.3.1 Not tested 3](#_Toc101808384)

[2.3.2 Not Coded 3](#_Toc101808385)

[3 Problem Explanation 4](#_Toc101808386)

[3.1 Overview of the Program 4](#_Toc101808387)

[3.1.1 The Classes 4](#_Toc101808388)

[3.1.2 The Classrooms 4](#_Toc101808389)

[3.1.3 The Professors 4](#_Toc101808390)

[4 Description of the Solution 6](#_Toc101808391)

[4.1 Overview of the Solution 6](#_Toc101808392)

[4.1.1 Methods of solving 6](#_Toc101808393)

[4.1.2 Adaptors 6](#_Toc101808394)

[4.1.3 Path to Solution 6](#_Toc101808395)

[5 System Requirements 7](#_Toc101808396)

[5.1 Required Software 7](#_Toc101808397)

[5.2 Introduction 7](#_Toc101808398)

[5.2.1 Description 7](#_Toc101808399)

[5.2.2 Broad Objectives 7](#_Toc101808400)

[5.2.3 Specific Objectives 7](#_Toc101808401)

[5.3 Scope 7](#_Toc101808402)

[5.3.1 Items Addressed 7](#_Toc101808403)

[5.3.2 Recommended Features 8](#_Toc101808404)

[Classroom Association 8](#_Toc101808405)

[Dynamic Error Log 8](#_Toc101808406)

[Class Grid Display 8](#_Toc101808407)

[Fully Dynamic Times 8](#_Toc101808408)

[5.3.3 Terms 8](#_Toc101808409)

[5.3.4 Assumptions 8](#_Toc101808410)

[5.4 Prose Description Appendix 9](#_Toc101808411)

[5.5 SRU Class Scheduler Glossary 10](#_Toc101808412)

[5.5.1 Important Classes 10](#_Toc101808413)

[5.6 Testing 11](#_Toc101808414)

[5.6.1 Human Factor Testing 11](#_Toc101808415)

[Selecting Excel File 11](#_Toc101808416)

[Checking Error Log 11](#_Toc101808417)

[Changing Classroom Information 11](#_Toc101808418)

[Adding a Classroom 11](#_Toc101808419)

[Changing Class Times 11](#_Toc101808420)

[Traditional Classes 11](#_Toc101808421)

[Single-Day Classes 11](#_Toc101808422)

[Changing Class Attributes 12](#_Toc101808423)

[From Scheduler Tab 12](#_Toc101808424)

[From Class List Tab 12](#_Toc101808425)

[Add a Class 12](#_Toc101808426)

[Changing Professor Attributes 12](#_Toc101808427)

[Saving Your Changes 12](#_Toc101808428)

[5.6.2 Static Analysis 12](#_Toc101808429)

[Compiler Warnings 12](#_Toc101808430)

[5.6.3 Integration Testing 12](#_Toc101808431)

[Adding Information 12](#_Toc101808432)

[Main Window to Add Class 12](#_Toc101808433)

[Main Window to Add Classroom 13](#_Toc101808434)

[Main Window to Add Professor 13](#_Toc101808435)

[Editing Information 13](#_Toc101808436)

[Main Window to Edit Class 13](#_Toc101808437)

[Main Window to Edit Classroom 13](#_Toc101808438)

[Main Window to Edit Professor 13](#_Toc101808439)

[5.6.4 Black Box Testing 14](#_Toc101808440)

[Scheduler Window 14](#_Toc101808441)

[Save Changes 14](#_Toc101808442)

[5.6.5 Continuous Testing 14](#_Toc101808443)

[Changes to Class Scheduler Logic Code 14](#_Toc101808444)

[Changes to Class Scheduler User Interface Code 14](#_Toc101808445)

[5.7 Caveats/Minefields 14](#_Toc101808446)

[5.7.1 Formatting Issues 14](#_Toc101808447)

[Excel File Not Formatted Correctly for Program to Read 14](#_Toc101808448)

[Problem 14](#_Toc101808449)

[Solution 14](#_Toc101808450)

[5.7.2 Complex Time Issues 15](#_Toc101808451)

[Overlapping Class Times 15](#_Toc101808452)

[Problem 15](#_Toc101808453)

[Solution 15](#_Toc101808454)

[Single Day Classes 15](#_Toc101808455)

[Problem 15](#_Toc101808456)

[Solution 15](#_Toc101808457)

[Disappearing Class Labels 15](#_Toc101808458)

[Problem 15](#_Toc101808459)

[Solution 15](#_Toc101808460)

[6 Description of Results 17](#_Toc101808461)

[6.1 Output Files 17](#_Toc101808462)

[6.1.1 Professor Report 17](#_Toc101808463)

[6.1.2 Updated Scheduling File 17](#_Toc101808464)

[6.1.3 Updated Header File 17](#_Toc101808465)

[6.2 Code Reusability 17](#_Toc101808466)

[6.3 UML Diagrams 17](#_Toc101808467)

[6.3.1 Changing Times State Chart Diagram 18](#_Toc101808468)

[6.3.2 Add Class State Chart Diagram 19](#_Toc101808469)

[6.3.3 Add Professor State Chart Diagram 20](#_Toc101808470)

[6.3.4 Add and Save Sequence Diagram 21](#_Toc101808471)

[6.3.5 Time Change Sequence Diagram 22](#_Toc101808472)

[6.3.6 View Room Info Sequence Diagram 23](#_Toc101808473)

[6.3.7 Check Room Info Activity Diagram 24](#_Toc101808474)

[6.3.8 Context Menu Activity Diagram 25](#_Toc101808475)

[6.3.9 Time Change Activity Diagram 26](#_Toc101808476)

[7 File Path Names 27](#_Toc101808477)

[7.1 Necessary Files 27](#_Toc101808478)

[8 Post Deployment 28](#_Toc101808479)

[8.1 Program Deployment/Maintenance 28](#_Toc101808480)

[8.2 Post-Mortem Analysis 28](#_Toc101808481)

# References

Deenathayalan, M (2019) *How to Create Word Document Using C#,* <https://www.c-sharpcorner.com/UploadFile/muralidharan.d/how-to-create-word-document-using-C-Sharp/,> C# Corner.

Dykstra, T (2022) *C# Programming Guide*, <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/,> Microsoft Docs.

Laleye, Y., McNamara, M., Pitluga, A (2004). *Hermes Final Report.*  Program documentation, Slippery Rock University of Pennsylvania, Slippery Rock.

Thangiah, S. *SRU Class Scheduler.* Program, Slippery Rock University of Pennsylvania, Slippery Rock.

*(2022) WPF ToolKit Documentation*, Xceed, <https://xceed.com/en/support/documentation-center>.

# Completion

## Contribution

The SRU Class Scheduler program was built upon from an existing project. The following is a breakdown of the contributions made to the project from outside sources.

### Percentage Breakdown

60% of the current code was modeled off of outside sources

* 40% was from the existing project that was provided
  + Thangiah
* 20% was gathered from the references listed:
  + Deenathayalan (2019)
  + Dykstra (2022)

### Explanation

In regard to the contributions that were provided from the existing code, the code was reusable. The group was able to utilize the existing code by modifying aspects to fit the newly intended use. For example, if the group wanted to create a new tab in a grid, the code to create a tab was available within the existing code but needed modified to display the desired content.

Aside from the contributions that were acquired from the existing project that was provided, the specific instances of the use of outside sources are explained below:

* Creating the professor report was heavily influenced by the guide written by Deenathayalan (2019). This guide gave the framework for how to create a word document, how to write out to a word document, and how to save a word document.
* The C# guide by Dykstra (2022) helped the group write code that could be considered rudimentary to the language. For example, the use of certain key words to understand basic C# principles contributed to reading in the Headers file for the startup of the program.

## Complete Modules

Completion Status: *Incomplete*

The following modules have been completed and tested for the current version of the SRU Class Scheduler.

* Add a Professor
* Edit a Professor
* Add a Classroom
* Edit a Classroom
* Add a Class
* Edit a Class
* Professor Report
* Save Changes
* Add Row

## Incomplete Modules

The following modules have not been completed and tested for this current version of SRU Class Scheduler.

Fully Dynamic Times – At least 1 bug is still present in this feature. Duplicate timeslots can appear on the grid from user made changes and the program will only use the lowest timeslot in the column so time conflicts can't be created by the duplicate. However, the grid will look odd if a duplicate timeslot is present. Users can change the duplicate timeslot to whatever timeslot is needed to use the program correctly.

### Not tested

* Adv. Options
* Conflicts

### Not Coded

* Allowing classes to be associated with specific classrooms
* Dynamic Error Log
* Class grid display

# Problem Explanation

## Overview of the Program

The SRU Class Scheduling Program has been created to assist with the process of class scheduling for the head of any college department. This program allows users to perform more accurate class scheduling with increased efficiency through an attractive user interface. The Class Scheduler is adaptable to the user’s input information via a cooperative Excel file.

The program consists of three major attributes of class scheduling: Classes, Classrooms, and Professors. Each attribute exists on its own but can be incorporated into other attributes of the scheduling process. Professor and Classroom can be incorporated into the Classes attribute. Similarly, the Classes can be incorporated into the Classroom and Professors attributes.

### The Classes

The Classes are represented on a grid in the main window of the program so that the department chair can schedule classes with little room for error. Each attribute of the class must be compliant with the expected value when editing or adding a class to the list.

There are several types of classes that can be created or loaded into the program through the Excel spreadsheet. These include the following: In-person, Online, Appointment, and Single-Day classes. All classes can be altered through the Edit Class Dialog but may have different available attributes depending on the class type. All aspects of a class can be altered through the SRU Class Scheduler Program.

### The Classrooms

The Classrooms are represented as buttons on the grid that correspond to the columns that in-person classes are placed in. When editing or adding a classroom, each attribute of the room must be compliant with the expected values before the classroom attributes will be accepted.

The four specified types of classrooms include “WEB”, “APPT”, “APPT2”, and any accurate building and room number that is allowed by the program. Depending on the classroom location of the class, the class may appear on the large grid in the main window, in the Unassigned tab, or in the WEB/APPT/APPT2 grid at the bottom center of the main window.

### The Professors

The Professors can be easily viewed in the Professor Key in the main window of the Scheduler tab as well as within the Professors tab. When editing or adding a professor, each attribute of the professor must be compliant with the expected values before the professor attributes will be accepted.

Professors have five attributes that are directly related to each instance: Name, SRU ID, Color, Max Credits, and Max Prep. These attributes can be viewed within the Professors tab of the main window. In relation to classes, the in-person classes that are on the main grid will appear in the color that corresponds to the professor that is teaching it.

# Description of the Solution

## Overview of the Solution

The group began working on a previously existing program that was previously intended to allow for an easier, more accurate scheduling process as well. However, this existing system did not have the functionality required for it to be used in a way that would prove helpful in the scheduling process. The existing system did not provide the user with the ability to interact with many aspects of the program including classes, classrooms, professors, timing, and error knowledge. In addition, the system would frequently crash if the user did not do exactly what the system was expecting.

### Methods of solving

The SRU Class Scheduler Program allows the user to have full privileges when using the program. While the program does validate user input and make user aware of the errors that may be occurring as they work, the user may utilize the user interface in whatever way they wish. The friendly user interface allows for altering and adding classes, classrooms, and professors. In addition, the user will always be able to save any new work that was completed. Progress will not be saved within the program. This allows the user to select a new Excel file every time they wish in order to complete their scheduling needs.

### Adaptors

Each time that the user wishes to interact with the program in a way that cannot be conducted on the current screen, a separate dialog box will appear to gather user input and allow for that desired user interaction. For example, when the user clicks on the “Add Class” button, the Add Class Dialog will appear for the user to provide input. In order for this interaction to work, the AddClassDialog.xaml.cs class is implemented to bridge the gap between the main window user interface and the Add Class Dialog user interface. This occurs for a large majority of user interaction with the program.

### Path to Solution

When the appropriate Excel spreadsheet is loaded into the program, the main window will appear with an attractive user interface for the user to begin their work within the program. All aspects of the scheduler are intertwined so that if the user edits something in one area of the program, it will update in all other areas.

When all of the desired changes have been made within the program, the user has the ability to save their work. This is crucial if they wish to provide this information to the dean of the college for implementation. The work will be saved as an Excel spreadsheet under the name that the user inputs in their file explorer before saving.

# System Requirements

## Required Software

* Microsoft® Windows® 2000
* Microsoft® Office® 2000 or later

## Introduction

### Description

The SRU Class Scheduler Program is designed to allow department chairs at Slippery Rock University to schedule classes in relation to their assigned classrooms, professors, preferences, and more. The user interface allows for easier interaction with the class data than what is found in working with an Excel spreadsheet. By checking all of the user’s entered information before it can be accepted by the program, the class scheduler eliminates the potential for scheduling conflicts. The user will also be able to save their work for further use so that the scheduling does not all have to be completed within one session.

### Broad Objectives

* Allow users to freely interact with class scheduling program in a way that eliminates potential scheduling conflicts
* Stop user from needing to change values within the Excel spreadsheet by giving all of that access within the program
* User interface must be attractive to the eye and uniform across the program.

### Specific Objectives

* User must be able to edit and add all aspects of classes, classrooms, and professors.
* Single-day classes must be able to be read in and edited by the program
* Program must be able to handle several formats of spreadsheets that are not compliant with the program without crashing.
* User must be aware of the errors that are occurring during runtime through an error log.

## Scope

### Items Addressed

* Users may freely interact with the class scheduling program in a way that eliminates potential scheduling conflicts. They can edit/add all aspects of classes, classrooms, and professors.
* The program handles files that are not perfectly formatted by communicating to the user what the issue is and allowing the user to have the chance to pick a new file.
* Several types of input can be handled now such as single day classes, the crosslist attribute of a class, and a professor’s correct SRU ID number.
* When there is an error that occurs during runtime, it will print to the error log tab and save to the error log history when the program is saved and closes.
* All dialog boxes that may appear during user interactions are uniform.

### Recommended Features

*Future work could be done to ensure that these options are functional to assist in the class scheduling process.*

#### Classroom Association

Allowing classes to be associated with specific classrooms would allow for the program to recognize specific courses and automatically associate them with a classroom. For example, a biology lab class must be taught in a laboratory.

#### Dynamic Error Log

A dynamic error log could be created by allowing the error log to “refresh” when errors are fixed. This would make the error log easier for the user to follow and more accurate.

#### Class Grid Display

Class grid display currently does not reflect the length of the class when it is displayed on the scheduler, this allows for a 50-minute class and a 2-hour class to appear as the same size. By changing the class grid, it will let users have a clearer view of what the schedule will look like.

#### Fully Dynamic Times

At least 1 bug is still present in the final version of this project. Moving forward when making time labels / timeslots for the GUI grid a Date Time object may be more helpful in creating timeslots that don’t create duplicates. Currently each timeslot uses strings for their timeslot values allowing a user to make duplicate timeslots on the grid. While no conflicts can be created by the user a duplicate timeslot can make the GUI look odd.

### Terms

* User - Most likely the department chair, this is the person that will be accessing the program to utilize the features for the purpose of creating a class schedule for the desired semester.
* Scheduler - The user interface within the “Scheduler” tab of the main window of the program; allows for the majority of user interaction with class, classroom, and professor attributes

### Assumptions

* The SRU Class Scheduler Program is assumed to be functioning properly.
* The SRU Class Scheduler Program is assumed to be available to any user who has properly installed the specified requirements listed in the appropriate manuals.

## Prose Description Appendix

The users of the SRU Class Scheduler program will have full access to all aspects of the user interface. They may edit and/or add anything that they see fit and can save changes to anywhere on the computer they are using. If any header or professor information would need to be changed permanently, the user could access the Headers file within the program at the following path: Schedule\_WPF/bin/Debug/Headers.xlsx.

The current program utilizes a GUI for users to interact with the program and effectively schedule the classes that they have loaded in. The program now allows users to edit and add several attributes of the class scheduling process while avoiding conflicts that may have occurred if the changes were only happening within an Excel spreadsheet. The ability to avoid conflicts within the program creates a much faster process of creating an accurate class schedule for the intended department.

The values of the class information will be stored within the user’s personal Excel spreadsheet that they have created in compliance with the Class Scheduler Program. The values of the expected headers, expected professors, and expected classroom information will be stored within the Headers file that is located within the program files at the following path: Schedule\_WPF/bin/Debug/Headers.xlsx

Once the user has conducted all of the changes that they wish, they may save their changes into an Excel spreadsheet. This spreadsheet will be able to be read back into the program for future use.

The time to generate the user interface of the program from the Excel file will vary by the Excel file that is being loaded into the system as well as the number of errors that occur during startup. On average, the loading of the file into the program takes about 20 seconds.

Possible failures that may occur during the runtime of the program are improper Excel spreadsheet format and scheduling errors that result in classes needing to be manually assigned a classroom. To ensure that the user is aware of all issues that occur, the error log will be added to every time there is something that needs to be communicated to the user. For any previous issues that the user may need to reference, there will be an error log history text file.

For an improper file format error, a message will be generated for the log describing where and why the error occurred. The message will also be displayed to the user describing the incorrect format encountered for the scheduler information file. If the error persists after the file format has been corrected, either then the system must undergo maintenance to correct the error, or the documentation must be corrected.

If there is a failure between the main window and the file selected, the user will be prompted to select a new file until the file is compliant with the expected values.

If there is an error detected in the schedule while the Excel spreadsheet is being read in, the error will be printed to the error log. The user will also be prompted to correct the error before their next startup so that the schedule can be as accurate as possible.

The environment that the SRU Class Scheduler program will initially operate in can be on any machine that has Virtual Studio 2022 as well as Microsoft Access 2000 or later.

The SRU Class Scheduler program is expected to have minimal security in relation to user privileges as there is no user-specific information that is stored within the program. In the future, it may be an option to require a specific pin to be emailed to the user so that the program can be sure the user is a person and not a robot.

The only software constraint currently involved with the program is its dependence on the user having access to Microsoft Access 2000 or later.

## SRU Class Scheduler Glossary

### Important Classes

**MainWindow.xaml.cs-** Main class for the class scheduling program that will open the main user interface when the program begins

**FileSelect.xaml.cs-** Creates an instance of the MainWindow class so that a request window appears to ask the user to choose the appropriate class scheduling Excel file

**EditProfessorDialog.xaml.cs-** Provides the opportunity for the user to change a professor’s attributes

**EditNotesDialog.xaml.cs-** Provides the opportunity for the user to add notes to a specific class

**EditClassTimeDialog.xaml.cs-** Provides the opportunity for the user to view and/or change a class’s weekday, start time, and end time. The class’s attributes are also validated.

**EditClassSeating.xaml.cs-** Provides the opportunity for the user to change a classroom’s maximum seating capacity

**EditClassRoomInfo.xaml.cs-** Provides the opportunity for the user to view and/or change a classroom’s attributes

**EditClassDialog.xaml.cs-** Provides the opportunity for the user to change a class’s attributes

**ChangeTimeDialog.xaml.cs-** Alters a classroom’s availability based on the times the classroom is scheduled to be in use.

**AddProfessorDialog.xaml.cs-** Handles the manual addition of a professor by the user and adds it to the appropriate areas of the scheduling platform. The professor is also assigned the appropriate attributes.

**AddClassRoomDialog.xaml.cs-** Handles the manual addition of a classroom by the user and adds it to the appropriate areas of the scheduling platform

**AddClassDialog.xaml.cs**- Handles the manual addition of a class by the user and adds it to the appropriate areas of the scheduling platform

## Testing

### Human Factor Testing

#### Selecting Excel File

1. Start the SRU Semester Scheduler by selecting the icon.
2. Click the Select File Button.
3. Select Excel file with semester information.
4. Note that a gear icon replaces the Select File Button.
5. Note that after a few seconds the start window closes and is replaced by the main window.

#### Checking Error Log

6. Click the “Error Log” tab in the upper left corner of the main window.

7. Note that all errors that occurred during startup will appear in this window.

8. Note that errors that occur during runtime will appear in the error log.

#### Changing Classroom Information

9. Click the “Scheduler” tab in the upper left corner of the main window.

10. Click any of the classrooms that appear as buttons on the top of the table underneath the table title.

11. Alter the classroom information to your liking or leave it the same.

12. Click submit or exit in the top right corner of the popup.

13. If changes were submitted, the user would see those implemented.

#### Adding a Classroom

14. Click on the “Add Classroom” button in the lower right corner of the main window.

15. Enter the desired information and click submit.

#### Changing Class Times

##### Traditional Classes

16. Click the “Scheduler” tab in the upper left corner of the main window.

17. Click and drag any of the classes onto an empty spot on the grid.

##### Single-Day Classes

18. In the “Single Day” tab on the bottom left of the main window, right-click on the class.

19. Select “Edit Class Time.”

20. Enter the desired information and click submit.

#### Changing Class Attributes

##### From Scheduler Tab

21. Right-click any traditional, online, unassigned, or single-day class in the window.

22. Select “Edit Class” option.

23. Enter the desired information and click submit.

##### From Class List Tab

24. Right-click on any part of the class you wish to edit.

25. Select “Edit Class” option.

26. Enter the desired information and click submit.

##### Add a Class

27. Click on the “Add Class” button in the lower right corner of the main window.

28. Enter the desired information and click submit.

#### Changing Professor Attributes

29. Click the “Scheduler” tab in the upper left corner of the main window.

30. Right-click on any professor in the Professor Key in the bottom right corner.

31. Select “Edit Professor” option.

32. Enter the desired information and click submit.

#### Saving Your Changes

33. Click on the “Save Changes” button in the lower right corner of the main window.

34. Navigate to the desired location in your file explorer and enter the title of what you would like to save the Excel spreadsheet as.

35. Save your changes.

36. Note that now you can safely exit the program and all of your changes will be found within the Excel spreadsheet that you named as well as the Headers file.

### Static Analysis

#### Compiler Warnings

On many of the variables that are only used in certain circumstances, there were compiler warnings. To eliminate these, the severity level was set to “None.” Many cases were regarding exceptions in the code.

### Integration Testing

#### Adding Information

##### Main Window to Add Class

* Information being sent
  + Add Class Dialog Box: to allow for user input
  + Requirements: to validate user input
  + Rejection Icons: to communicate to user that the input is invalid
* Information being received
  + Add Class Dialog Box: user input becomes a class

##### Main Window to Add Classroom

* Information being sent
  + Add Classroom Dialog Box: to allow for user input
  + Requirements: to validate user input
  + Rejection Icons: to communicate to user that the input is invalid
* Information being received
  + Add Classroom Dialog Box: user input becomes a classroom

##### Main Window to Add Professor

* Information being sent
  + Add Professor Dialog Box: to allow for user input
  + Requirements: to validate user input
  + Rejection Icons: to communicate to user that the input is invalid
* Information being received
  + Add Professor Dialog Box: user input becomes a professor

#### Editing Information

##### Main Window to Edit Class

* Information being sent
  + Edit Class Dialog Box: to allow for user input
  + Requirements: to validate user input
  + Rejection Icons: to communicate to user that the input is invalid
* Information being received
  + Edit Class Dialog Box: user input updates the current class

##### Main Window to Edit Classroom

* Information being sent
  + Edit Classroom Dialog Box: to allow for user input
  + Requirements: to validate user input
  + Rejection Icons: to communicate to user that the input is invalid
* Information being received
  + Edit Classroom Dialog Box: user input updates the current classroom

##### Main Window to Edit Professor

* Information being sent
  + Edit Professor Dialog Box: to allow for user input
  + Requirements: to validate user input
  + Rejection Icons: to communicate to user that the input is invalid
* Information being received
  + Edit Professor Dialog Box: user input updates the current professor

### Black Box Testing

#### Scheduler Window

After reading in a sample class scheduling file. The information in the file was quality checked to ensure that it reflected the information in the sample class scheduling file.

#### Save Changes

After changing information within the program, the new information was saved to a new Excel file. It was then quality checked to ensure the information was corrected and all changes were highlighted in red.

### Continuous Testing

#### Changes to Class Scheduler Logic Code

For all changes within the logic of the code, such as reading in the excel file. Unit testing and Integration testing will be run to check if there are errors.

#### Changes to Class Scheduler User Interface Code

For all changes to the user interface, such as the creation of new buttons or windows. Human Factor testing will be utilized to test the accuracy of the program.

## Caveats/Minefields

While creating this program there were some issues that had to be dealt with throughout the entirety of the project. The two main problems surrounding the excel sheet and the timetables. This section will speak more in depth on the problems and our solutions to them.

### Formatting Issues

#### Excel File Not Formatted Correctly for Program to Read

##### Problem

Originally, the Excel sheet was only able to be read by the program if all information was in a specific order. If the headers were not accurate the program would crash. If the information was in the correct order but not able to be read, then the program would crash. This was a large problem as users cannot be trusted to always enter correct and accurate data

##### Solution

The way we overcame this problem was by creating the Headers.xlxs file and the error log. If headers are out of order the excel file will still be able to be read in as there are indexes to save the location of the header. Solving the issue of invalid information was done by outputting to the error log if information was missing or incorrect. If the information entered was bad enough to cause a fatal error, then a message box would appear with information on how to fix it and it would be written out to the error log history.

### Complex Time Issues

#### Overlapping Class Times

##### Problem

Timeslots in the program are determined via labels placed on the grid using a specified naming procedure. This procedure involves the class day, hour side of the time, classroom, meridian of the time. Due to this the program when moving classes does not look for time conflicts present when a class that starts at 06:00 PM and ends at 07:15 PM. Only the 06 of the class starting time is used for the label.

##### Solution

Within the DetermineTimeConflicts() method extra checks were added so that the entire time is checked against every other class time so that a class starting at 07:00 PM won't have a conflict with a class from 06:00 PM to 07:15 PM as the class will be moved to the unassigned class List this way even though the label naming convention isn’t ideal conflicts between overlapping class times would be caught.

#### Single Day Classes

##### Problem

The program was writing used the parent Grid name as the days for each class. These grid names are “MWF“ and “TR” respectively. This would cause single day classes to be unable to interact with any part of the program.

##### Solution

The way we overcame this problem was rewriting the code so Single day classes have their own section in the program and any instance of a method using “MWF” or “TR” for the class day will also be checked for “M” “W” or “F” as the day for “MWF” and for “TR” it also checks for “T” and “R” so that single day classes could be read in, and compared to other classes within the program.

#### Disappearing Class Labels

##### Problem

The program in some instances, instead of making a label contain the class text box name and have the same background color as the assigned professor would leave the label blank while also changing the class information to the specified timeslot.

##### Solution

Two solutions to this problem were enacted as there were multiple reasons for this problem existing. First labels are checked multiple times throughout runtime to make sure no duplicate labels exist before populate Time Table is called. The other solution needed was a modification to how the program determines if a label is empty or not. Originally the program only checked if the string was empty to say the label was empty, instead the program checks that the label is an empty string or the label’s content is equal to the class text box name of the current class being moved, to prevent a “conflict” from being detected and removing the class from the label when it shouldn’t have.

# Description of Results

## Output Files

Upon successful creation of the professor report and save changes the user will receive two output files. A word document titled Professor\_report.docx and their created excel file with the updated scheduling information. The Headers.xlxs file will also be updated with the current information.

### Professor Report

This report returns information regarding the professors and the classes they teach. This document outputs a list of professors used in the scheduling and for each professor a list of classes they teach. This allows the user to see the information in a unique way from the class scheduling excel file.

### Updated Scheduling File

Upon saving your changes the current scheduling file will be overwritten or it will allow for the user to save their changes as a new excel sheet. This excel sheet will look similar to the excel sheet read in, however, it will contain all of the changes highlighted in red.

### Updated Header File

Upon saving your changes the current headers file will be overwritten. This will allow for classroom profiles and professor profiles to be updated with new classroom and professor information. This is important as the next time you run the program these profiles will be saved.

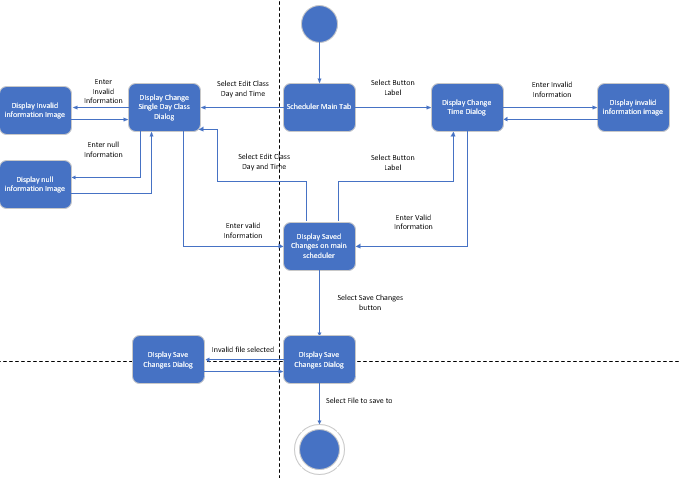
## Code Reusability

The SRU Class Scheduler was created in a manner that allows for reusability of the program. Because there is a save changes feature, the user can save their changes externally onto their own computer. The saved file may be uploaded for further alterations to the schedule. The user also has the option to select a new file to load into the program. Regardless of whether the file was used previously in the program or not, the SRU Class Scheduler is expected to operate correctly and aid the user in their scheduling needs.

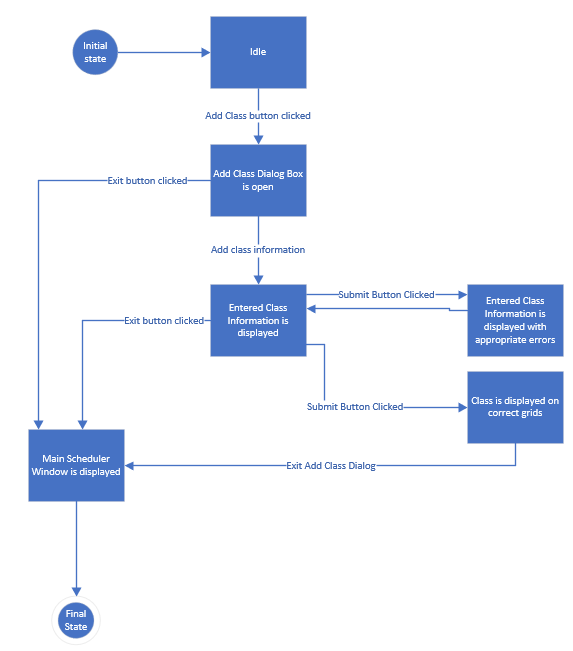
## UML Diagrams

*The following UML diagrams will assist in the understanding of how the Class Scheduling Program functions. If you would like a closer look at any of the diagrams, all of the diagrams can be found in the folder titled “UML.”*

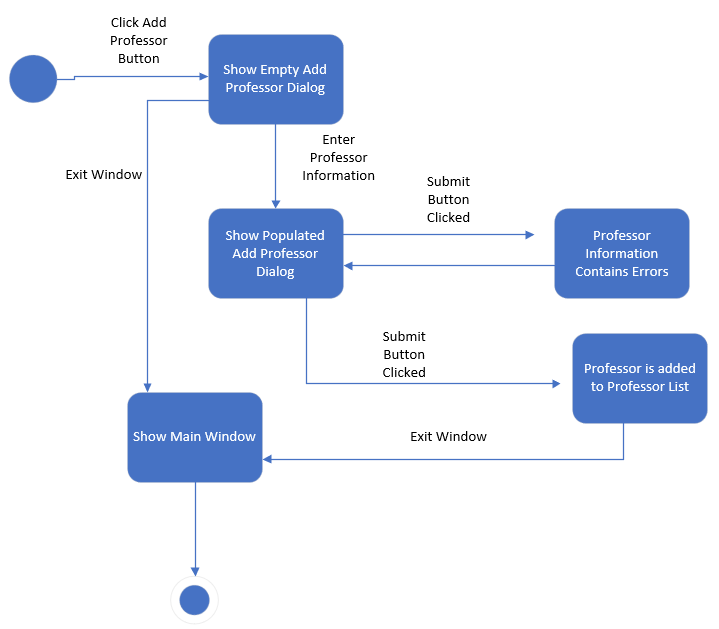
### Changing Times State Chart Diagram



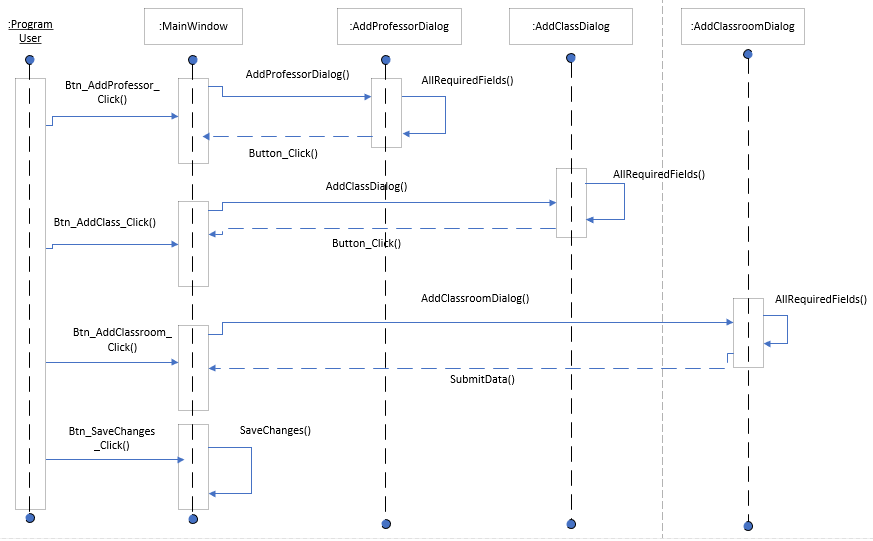
### Add Class State Chart Diagram



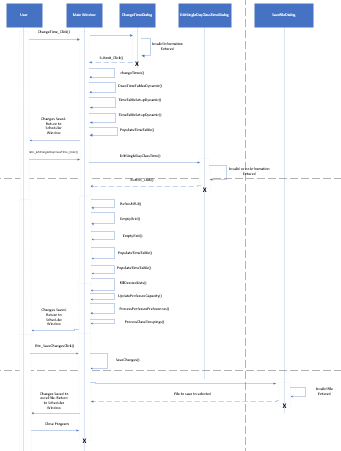
### Add Professor State Chart Diagram



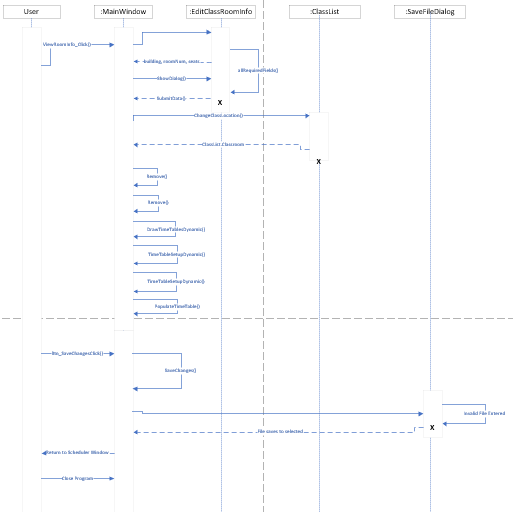
### Add and Save Sequence Diagram



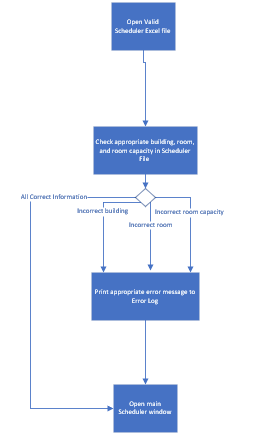
### Time Change Sequence Diagram



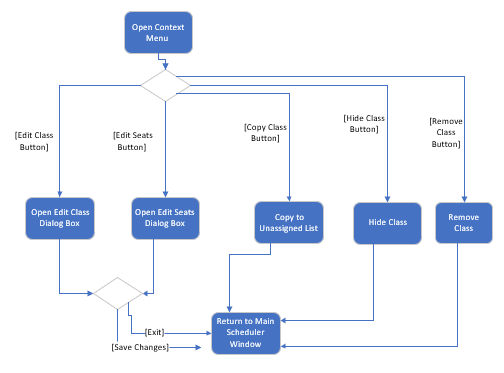
### View Room Info Sequence Diagram



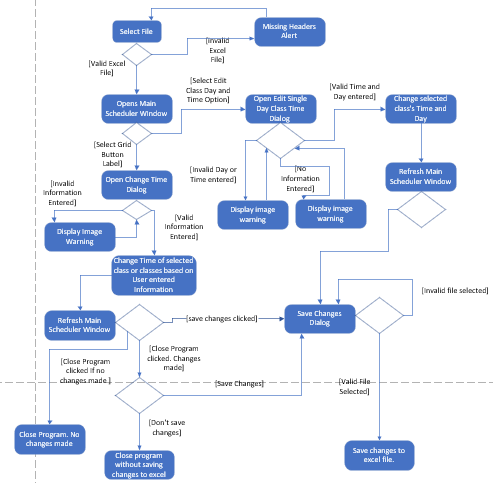
### Check Room Info Activity Diagram



### Context Menu Activity Diagram



### Time Change Activity Diagram



# File Path Names

## Necessary Files

Needed files can be found within the main folder that contains the program. These files are Headers.xlxs, errorLog.txt, and errorLogHistory.txt. You will also need a file that contains your class scheduling information which should be provided by the university.

# Post Deployment

## Program Deployment/Maintenance

The issues that may be run into regarding maintenance after deployment would be in regard to the Headers.xlsx file. If the program is used by many departments and these departments are providing feedback that is not uniform, there would need to be several Headers files for users to choose from. This may get complicated because a department chair may think that the program has become too complicated if they are required to choose their own header file too.

If there are instances of two-day or four-day classes that are loaded into the system, the entire program would need to be re-worked. There is no current handler for those types of classes so if that became an issue based on user feedback, there would be extensive maintenance.

## Post-Mortem Analysis

While the final product of the SRU Class Scheduler is functional, the group did start with existing code that was built off of. If the group were to complete the project over again, we would have started from scratch so that the grid would have a more free-flowing layout. Currently, each class must be placed in a specific block of the data grid. It would be better visually understood if the amount of space that the class took up on the free-flowing grid would correspond to the length of time that the class would be in session.

The group would also have created a separate method for creating class times so that the method could be called each time instead of typing it out for every instance it is needed. The same can be said for reading in values from the header file. If the group were to start over, we would have created a separate method (or class) to open the header file and store all values instead of opening and closing the file several times throughout the program.