Technical Manual for Class Scheduling Application

By Katherine Settle, Dean Lisko, Autumn Kriebel

March 6, 2022

# Overview

* This manual contains instructions for installing and utilizing the class scheduling application. The highlighted portions of the manual are currently under review.

# System Requirements

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

# 3. Assumptions

* It is assumed that all system requirements have been met and that the user has access to a device that is compatible with the above system requirements.

# 4. Program 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

# 5. Interactions Between Program Classes

**File Selection Window to Scheduling Timetable**

* Upon starting the program, ReadExcel() method communicates with FileSelect class so that the user can choose the desired scheduling file.
* The Headers.xlsx file is read in to validate the scheduling file.
* Once the scheduling file is validated in the ReadExcel() method with the list of headers, it is uploaded into the timetable.

**Scheduling Timetable View to Changing Classroom Attributes**

* When the user clicks on the classroom label, the classroom’s current information will be displayed along with the option to change the classroom’s attributes.
* If submitted, the ViewRoomInfo() method will create an instance of the EditClassRoomInfo class, use UpdateClassRoomInfo() method to see if the classroom’s attributes were updated, and updates all classes that used to be in that room.
* If the classroom’s attributes are altered, a new classroom with the updated attributes are created using AddClassroom() method and the old classroom is removed by calling RemoveClassroom() method.
  + The classes that had a location of the old classroom will be updated to have the location of the new classroom by calling ChangeClassLocation() method.

**Scheduling Timetable View to Changing Class Attributes**

* This has not been implemented yet.

**Scheduling Timetable View to Changing Professor Attributes**

* This has not been implemented yet.

**Scheduling Timetable View to Adding a Classroom**

* When the user clicks on the “Add a Classroom” button in the lower right-hand corner of the main window, the AddClassroom() method is called.
* When AddClassroom() method is called, the new classroom information is added to the classroom list and displayed on both the MWF timetable and the T/TR timetable.
* If a classroom is added to the classroom Excel spreadsheet manually (not within the main window), it will automatically be loaded into the timetables upon starting the program by utilizing the ReadExcel() method.

**Scheduling Timetable View to Adding a Class**

* When the user clicks on the “Add a Class” button in the lower right-hand corner of the main window, the AddClass() method is called.
* When AddClass() method is called, the new class information is added to the class list and displayed on the appropriate timetable.
* If a class is added to the scheduling Excel spreadsheet manually (not within the main window), it will automatically be loaded into the timetables upon starting the program by utilizing the ReadExcel() method.

**Scheduling Timetable View to Adding a Professor**

* When the user clicks on the “Add a Professor” button in the lower right-hand corner of the main window, the Btn\_AddProfessor\_Click() method is called.
* Btn\_AddProfessor\_Click() method assigns the new professor a color to be used in the timetable before calling the AddProfessor() method.
* When AddProfessor() method is called, the new professor information is added to the professor list that is displayed in the Professor List Tab at the top of the main window.
* If a professor is added to the scheduling Excel spreadsheet manually (not within the main window), it will automatically be loaded into the timetables upon starting the program by utilizing the ReadExcel() method.
  + This feature is not in working condition for the current version of the scheduling program.

**Scheduling Timetable View to Save Changes**

* When the user clicks on the “Save Changes” button in the lower right-hand corner of the main window, the SaveChanges() method is called.
* The SaveChanges() method exports the current information being displayed in the timetables to a new Excel spreadsheet that the user can save under a new filename or the previously used filename.
  + The format that was originally read in should be the same for when it is exported to the new Excel file.

**6. Handling Additional Headers**

The following instructions explain how to update the program in the case of additional headers being added to the class information excel file.

1. **Updating the Headers.xlxs file**
   1. The first thing that you will need to do is update the Headers.xlxs file. If you have successfully installed and configurated the program you will file this file within the Schedule\_WPH folder => bin folder => debug folder. This process is demonstrated below.

Graphical user interface, application

Description automatically generated

Figure 1‑0‑1: Selecting the Schedule\_WPF folder

Graphical user interface, application, Excel

Description automatically generated

Figure 1‑0‑2: Selecting the bin folder

Graphical user interface, application, Word

Description automatically generated

Figure 1‑0‑3: Selecting the Debug folder

Graphical user interface, application

Description automatically generated

Figure 1‑0‑4: Selecting Headers.xlsx

* 1. Once you have opened the Headers.xlxs file, you are able to add your new header to the list of headers. Ensure that you are in Sheet 1. You will see that the first column of the sheet is full of headers that are within the class information excel file.

Graphical user interface, application, table, Excel

Description automatically generated

Figure 1‑0‑5: Sheet 1 of the Headers.xlxs file

* 1. To add your new header, scroll to the bottom of the page and in the first open cell type your new header.

Graphical user interface, application, table, Excel

Description automatically generated

Figure 1‑0‑6: Space for a new header to be added

**note:** The program is case sensitive so be sure that the header you typed exactly matches the new header.

* 1. Once you are done you can save and exit from the excel file.

1. **Updating Header Indexes**
   1. Now, you need to create an index for the new header within the program. Inside the ReadExcel method in the MainWindow.xaml.cs there will be a list of indexes.

A screenshot of a computer

Description automatically generated

2

1

Figure 1‑0‑7: 1) shows the class you need to be inside to find the ReadExcel method and indexes. 2) shows where there is space to add the index for the new header.

* 1. You will need to add in a line of code following the format below.

int index(NAME) = excelHeaderList.IndexOf("(NAME)") + 1;

For example, if you are adding the header “NewHeader” you would type:

int indexNewHeader = excelHeaderList.IndexOf("NewHeader") + 1;

* 1. The name of the index can be any name; however, it helps to create one that is similar to the header name as to not cause confusion when reading the code.

1. **Reading in the Information**
   1. A screenshot of a computer

      Description automatically generated with medium confidenceTo ensure that the new data is being properly read in by the program, determine what classification the new header belongs to: Classes, Classrooms, Professors, or Timing. Once this is determined, add the appropriate code into the corresponding section of ReadExcel() inside of MainWindow.xaml.cs by following the format that is currently in place.

Figure 1- 0-8: Example of a section for reading in header information related to creating professors

* 1. In order to read in information that corresponds to the new header, use the following line: worksheet.Cells[I, index(NAME)] as Excel.Range).Value
     1. Be sure to replace (NAME) with the name of the index of your new header.
  2. It is important to check that the value is not null by using an if statement, similar to figure 1-0-8. This allows for the program to read in any values that will cause problems when trying to display it on the scheduler.

1. **Updating Edit/Add Class Information**
   1. To ensure that the new data is properly interacting with the program, determine what classification the new header belongs to: Classes, Classrooms, Professors, or Timing. Once this is determined, add the appropriate code into the corresponding classes by following the format that is currently in place.
      1. New Classes Attribute:
         1. Reference the following classes for any additions that will need to be made:
            1. MainWindow.xaml (and MainWindow.xaml.cs)
            2. Classes.cs
            3. AddClassDialog.xaml (and AddClassDialog.xaml.cs)
            4. EditClassDialog.xaml (and EditClassDialog.xaml.cs)
            5. EditSingleDayClassTimeDialog.xaml (if applicable)
            6. EditSingleDayClassTimeDialog.xaml.cs (if applicable)
      2. New Classrooms Attribute:
         1. Reference the following classes for any additions that will need to be made:
            1. MainWindow.xaml (and MainWindow.xaml.cs)
            2. ClassRoom.cs
            3. AddClassRoomDialog.xaml (and AddClassRoomDialog.xaml.cs)
            4. EditClassRoomInfo.xaml (and EditClassRoomInfo.xaml.cs)
            5. EditClassDialog.xaml (and EditClassDialog.xaml.cs)
      3. New Professor Attribute:
         1. Reference the following classes for any additions that will need to be made:
            1. MainWindow.xaml (and MainWindow.xaml.cs)
            2. Professors.cs
            3. AddProfessorDialog.xaml (and AddProfessorDialog.xaml.cs)
            4. EditProfessorDialog.xaml (and EditProfessorDialog.xaml.cs)
      4. New Timing Attribute:
         1. Reference the following classes for any additions that will need to be made:
            1. MainWindow.xaml (and MainWindow.xaml.cs)
            2. Timeslot.cs
            3. EditClassTimeDialog.xaml (and EditClassTimeDialog.xaml.cs)
            4. EditSingleDayClassTimeDialog.xaml (and EditSingleDayClassTimeDialog.xaml.cs)
            5. ChangeTimeDialog.xaml (and ChangeTimeDialog.xaml.cs)
2. **Updating Save Changes**
   1. As previously mentioned, it is important to note that if the information from the new header is to be saved upon a “Save Changes” event, the proper references will need to be added to the data table in the following method in MainWindow.xaml.cs:

public System.Data.DataTable getDataTableFromClasses()

**7. UML Diagrams on Functionality of Class Scheduling Program**

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 Diagrams.”

**Use Case Conditions**

**Exit**

Precondition: The system is running.

Activity: This service would terminate the system and all agents running.

Post condition: The system is now not running.

**Write Error**

Precondition: The system is running.

Activity: This service would detect an error in the program’s functionality and write it to the error log.

Post condition: The user is aware of the error and the error is recorded.

**Read Excel**

Precondition: The excel file to be read in has been chosen.

Activity: Information from excel sheet is read in. Time conflicts, missing headers, and timeslots are checked. All classes that have a designated room and time with no conflicts are placed on the main window grid.

Post condition: Main program window opens in “Scheduler” tab

**“Scheduler” Tab Label changed**

Precondition: One of the button labels for the class times was clicked and the required information was input into the change time Dialog and submitted.

Activity: Button label changes time to match user input information based on parameters other button labels may be changed. The time slot, the button label and any other changed labels are attached to receive a new name. All duplicate changed times are removed. The master time slot list is updated to have the user inputed timeslots.

Post condition: Scheduler tab grid is modified to fit user input changes

**Class Moved**

Precondition: A colored class is clicked and dragged to a time slot by the user

Activity: calls determine time to find the new timeslot for the class from the default and master Time lists, and applies the new time to the class

Post condition: Changes moved class to have a starting and ending time based on the position it was moved to

**Save Changes**

Precondition: User clicks save changes button and specifies where the information will be saved to

Activity: Writes all current class information to a data table, then writes that data table to an excel file

Post condition: New excel file is created in specified location with modified class information

**UML Diagrams**Diagram

Description automatically generated

Figure 1‑0‑9: Use case Diagram to preform actions in the scheduler tab not involving changing the time label buttons

Diagram

Description automatically generated

Figure 1‑0‑10: Use case diagram to preform actions in the scheduler tab involving the time label buttons

**Diagram

Description automatically generated**

Figure 1‑0‑11: State chart diagram for reading in the excel file

Diagram

Description automatically generated

Figure 1‑0‑12: Activity diagram for starting the program

Diagram

Description automatically generated

Figure 1‑0‑13: Class Diagram for program functionality

Diagram, schematic

Description automatically generated

Figure 1‑0‑14 Sequence Diagram on adding new information and saving

Diagram

Description automatically generated

Figure 1‑0‑15 Activity diagram on context menu that appears when right clicking on a class within the schedular window

Diagram

Description automatically generated

Figure 1-0‑16: State chart diagram that shows how adding a professor works

Diagram

Description automatically generated

Figure 1-0‑17: Activity diagram that shows how checkroomInfo checks to ensure correct information

Chart, diagram

Description automatically generated

Figure 1-0‑18: State Chart diagram that shows how to add a class

Diagram

Description automatically generated

Figure 1-0‑19 State chart diagram on how to change times

Diagram

Description automatically generated

Figure 1-0‑20: Activity Diagram on how to change times

Diagram

Description automatically generated

Figure 1-0‑21: Sequence Diagram on how to change times

Diagram

Description automatically generated with low confidence

Figure 1- 0‑22: Sequence Diagram on viewing room info