Class Scheduler

Final Report

Team Number: SDDEC22-09

Client: Vicky Thorland-Oster

Team Members:

Zachary Bunch- Team Leader and Backend developer

Chris Horvatich - Frontend Developer

Connor Gaecke - Backend Developer

Charles Mulderink - Frontend Developer

Team Email: sddec22-09@iastate.edu

TeamWebsite: http://sddec22-09.sd.ece.iastate.edu

Revised: Dec 6, 2022 /Version 1

Executive Summary

Summary of Requirements

1. Be a document that can be changed with the schedule of classes. If it could be connected to the Schedule of Classes on ISU's website—that would be fabulous. If not, it should be duplicated from the fall before or the spring before.
2. It will need to have the labs associated with the specific rooms.
3. Have them "group" related (so Cpr E 530, 531, 532, 533, 534, 535, 536, 537…show up as one color or some demarcation so that I know I am not putting something on top of another class in that area).
4. It needs to have the times of day that MWF classes are scheduled and the TR scheduled times.
5. Needs to make sure that it doesn't list it twice if it is cross-listed with another.
6. The ability to add Com S, Phys, and Math courses would be sweet

Project Summary

Our project was focused on creating an application that made it easier for an advisor to create the schedule of class for the ECpE department. Our goal was to create a standalone application that would allow the advisor to have a user friendly alternative to the current method of creating the schedule. For the frontend, we wanted to create a GUI using Python and leverage existing frameworks to create a user friendly interface. For the backend, we tried to use CSV files for the internal database and wanted to implement a genetic algorithm to create an optimization functionality.

Unfortunately, our project did not achieve full functionality. We were able to create a GUI and leverage CSV files for the database, however we were unable to get them to work together. In the frontend, we ran into several issues with the framework trying to get it to implement the various features we were trying to deliver. On the backend we struggled with getting the CSV files to work with the GUI and trying to get multiple CSV files to work together. Additionally, we were unable to get the genetic algorithm to work within the project.

Table of Contents

1 Team

1.1 TEAM MEMBERS

Zachary Bunch, Connor Gaecke, Chris Horvatich, and Charles Mulderink

1.2 REQUIRED SKILL SETS FOR YOUR PROJECT

- Programming in Python
- Algorithm Design
- GUI Design using Python
- Backend Database with available classes

1.3 PROJECT MANAGEMENT STYLE ADOPTED BY THE TEAM

During this project we leveraged the Agile methodology. We utilized sprints to focus on the implementation of specific features. This helped us focus on implementing specific features and ensure proper functionality as the project progressed.

1.4 PROJECT MANAGEMENT ROLES

Zachary Bunch- Team Leader and Back-End Development
Connor Gaecke- Back-End Development
Chris Horvatich- Front End GUI Development
Charles Mulderink- Front End GUI development

2 Introduction

2.1 PROBLEM STATEMENT

We were tasked with creating a more efficient and user-friendly way of scheduling classes with the ECpE department at ISU. Our primary goal was to create an application that was more efficient to use than Excel but did not require the user to have to use a complex application. Additionally, one of our unrealized goals was to create a built in class scheduling feature to reduce the amount of work done by hand. Our client currently has to create the schedule by hand in Excel, and we wanted to reduce the amount of time spent to create the class schedule each semester.

2.2 REQUIREMENTS & CONSTRAINTS

- IDE: PyCharm
- GUI Editor: Tkinter
- Work within schedule of classes (Constraint)

- UI will have a schedule of classes and show what time are available and times that are not

Requirements and constraints from Client

1. Be a document which can be changed with the schedule of classes. If it could be connected to the Schedule of Classes on ISU's website—that would be fabulous. If not, it should be duplicated from the fall before or the spring before.
2. It will need to have the labs associated with the specific rooms.
3. Have them "group" related (so Cpr E 530, 531, 532, 533, 534, 535, 536, 537…show up as one color or some demarcation so that I know I am not putting something on top of another class in that area).
4. It needs to have the times of day that MWF classes are scheduled and the TR scheduled times.
5. Needs to make sure that it doesn't list it twice if it is cross-listed with another.
6. The ability to add Com S, Phys, and Math courses would be nice.

2.3 Engineering Standards

PEP 8 – python coding standard to keep our notes and code unified.

Due to the nature of our project being all software and not network enabled we do not have many technical standards we are required to follow.

2.4 Intended Users and Uses

The primary user for this project is the advisor who is responsible for creating the schedule of courses for the ECpE department. The primary use for this project is to leverage the application to reduce the amount of time spent to create the schedule of courses for the ECPE department. Students will also indirectly benefit from this application since it could help the user better visualize the classes within the schedule, and reduce the chance of scheduling conflicts. Professors could also indirectly benefit from this application since it could help the user identify possible time slots where classes could be scheduled.

3 Design

3.1 Design Context

3.1.1 Broader Context

Area	Description	Examples
Public health, safety, and welfare	How does your project affect the general well-being of various stakeholder groups? These groups may be direct users or may be indirectly affected (e.g., solution is implemented in their communities)	The health, safety, and welfare of the public should not be affected by our application. If anything it will save the public time and reduce scheduling conflicts for students.
Global, cultural, and social	How well does your project reflect the values, practices, and aims of the cultural groups it affects? Groups may include but are not limited to specific communities, nations, professions, workplaces, and ethnic cultures.	Our application will reflect the practices of the students, advisors, and professors that are involved with classes or scheduling in the EcpE department.
Environmental	What environmental impact might your project have? This can include indirect effects, such as deforestation or unsustainable practices related to materials manufacture or procurement.	Our application should not have much impact on the environment. If anything it might save more paper by making it all self-contained within the application.
Economic	What economic impact might your project have? This can include the financial viability of your product within your team or company, cost to consumers, or broader economic effects on communities, markets, nations, and other groups.	Our application should directly reduce the time that is spent with scheduling for the EcpE department. In turn, this will help the college save money.

3.1.2 User Needs

Advisors need a way to schedule classes without having to do it by hand while taking into consideration all of the constraints that go along with scheduling classes. The application needs to reduce the overall amount of time spent creating the schedule of classes compared to the current method that uses Excel. The application also needs to be user friendly to encourage the user to leverage the program instead of reverting back to Excel. Students need the application to take scheduling conflicts into consideration to optimize the schedule and to reduce the amount of scheduling conflicts. Professors need the application to be efficient at helping the advisor find open time slots to schedule newly proposed courses.

3.1.3 Prior Work/Solutions

There are other solutions out there that try to solve similar problems as our project but none of the solutions we found worked within the client's requirements. The primary requirement that required a unique solution was the use of an internal database that stored the data in a user friendly file format. This eliminated any solution using an external database like a SQL or NoSQL, and it eliminated any solution where the data was just accessible via the applications memory. That caused us to create a more unique solution that uses CSV files as the internal database.

3.1.4 Technical Complexity

1. The design of our project contains sufficient technical complexity, the use of python and frameworks within python to complete our application will give us new technical skills that may not have been taught in-depth during our course work.
2. Our application will have a front and back end, forward-facing the user will see a GUI and all of the constraints and automation will happen on the backend.

3.2 DESIGN EXPLORATION

3.2.1 Design Decisions

We leveraged the Python language to create both the front-end user interface and back-end algorithm. We went through several Python frameworks, such as Kivy and PyQt5, before choosing our final framework Tkinter. This framework allows us to leverage built in components to help with the overall design of the GUI. We decided on leveraging CSV files for the internal database that way the information was stored in a user friendly and accessible way, and did not require the use of a web server.

3.2.2 Ideation

Below are six iterations we went through to come up with how we would structure the application's internal database:

1. Hard code the classes- this was our original idea but we quickly realized this wouldn't solve the clients needs sufficiently as the classes requirements couldn't be easily updated with the changing curriculum.
2. Have a single profile that the program stores - This would have been easy to implement as the program would not need to write the data into a transferable location and all the classes specifications could be easily managed, but it is not ideal because the profile could not be easily moved or copied which would make iterating on it difficult.
3. Have the user input a formatted Excel spreadsheet with the classes- This option would have been easy for us to implement but it would have created a worse user experience for the client. The whole goal of this project is to reduce the amount of work the client has to do by hand, so this solution does not seem like it really achieves that goal since the client would still be entering data into Excel.

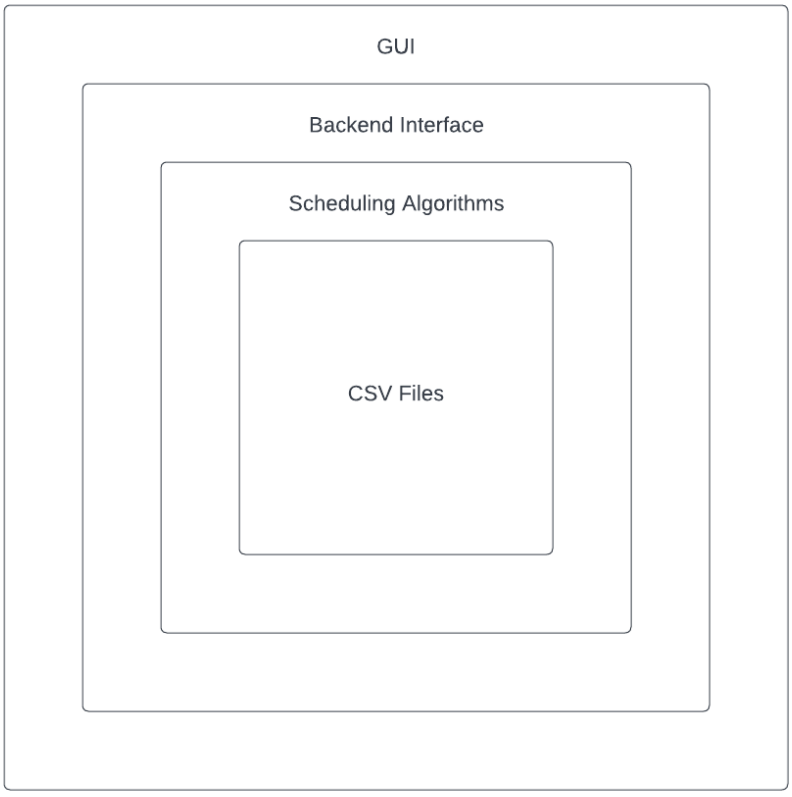
4. Create a database and have it run on a server- This option would have simplified our application but it would have created a worse user experience as well. We cannot expect our client to maintain and debug a server with a database application running on it, so this option would lead to potential problems in the future when the database application needs to be updated. It could also lead to issues where the server might be down and the client isn't able to bring the server online and that would make the application useless. That is why we decided against this solution.
5. Storing the profiles as JSON files - This option was the solution we chose going into this semester. We liked how it created a structured pairing of the data relationships, but was still in a file format that could be updated without having to use the application. However, we quickly realized that for a non-technical user, learning the structure of a JSON file might not be intuitive, so we decided to move away from this solution.
6. Storing the information in CSV files - This is the option we decided to use in the final product. We decided on this option because it gave us the ability to store data without an external database, allows the user to leverage Excel on an ad hoc basis, and is more user friendly for non-technical users to access the information compared to JSON files.

3.2.3 Decision-Making and Trade-Off

Idea:	Pros:	Cons:
Hard-coding the classes	● All class data is stored in a single file	● Has to be updated manually
Single Profile of classes	● All class data is stored in a single area	● Has to constantly edit a single profile with no way to access earlier versions
Excel Spreadsheet	● Ease of access to class data	● Excel spreadsheets are already being used and we are trying to move away from them
Web Server	● Can use database tools/programs to make edits	● Needs Internet Connection to run ● User has to maintain web server
Storing profiles as JSON files	● Multiple class setups ● Can make edits without editing any files	● Many profiles will take up more local storage
Storing profiles as CSV files	● Ease of access to class data	● Harder to control input validation

3.3 FINAL DESIGN

3.3.1 Design Visual and Description



This image represents the overall layout of our application in respect to the various layers with the application. The innermost layer represents the internal database made up of two CSV files (one file contains the course information and the other contains the schedule information). This layer is also the furthest away from the user and that is why it is the innermost layer. The next layer up is the Scheduling Algorithm. This layer refers to where our ideal location for the scheduling algorithm would have been located in the application if it was operational. The next layer up is the Backend Interface. This interface describes the way the frontend interacts with the data stored in the backend. In practice this layer is made of methods that read and send information to the CSV files. The uppermost layer is the GUI, which is the primary interface the user will interact with. This layer is responsible for displaying the information and utilizing the Backend Interface to access the internal database.

3.3.2 Functionality

Our primary goal for this project is to create an easy-to-use desktop application that requires the user to have a minimal understanding of the application to be able to use it. We decided to use a

stand-alone application since it would allow us to wrap all of the layers in the previous image into one executable for the client to run. We did not want them to have to worry about starting up a database on their computer or trying to get them to connect to a database hosted on a remote server. That is why we decided to bundle all of the required components into one application and not split them out between true frontend and backend applications. We also anticipated that this application might be passed on to future advisors and wanted to make the process of sending the application as easy as possible for our client. That also factored into our decision of making this a standalone application instead of multiple moving parts.

3.3.3 Areas of Concern and Development

Based on our client's requirements, our design remained as a single user application. Opting for a local application eliminated the possibility of making the application collaborative. The client's previous solution faced the same problem since input from outside users had to be emailed to the advisor. Since the application needed to prioritize useability and portability for the primary user, we were not able to address our concern of this application being a single user application.

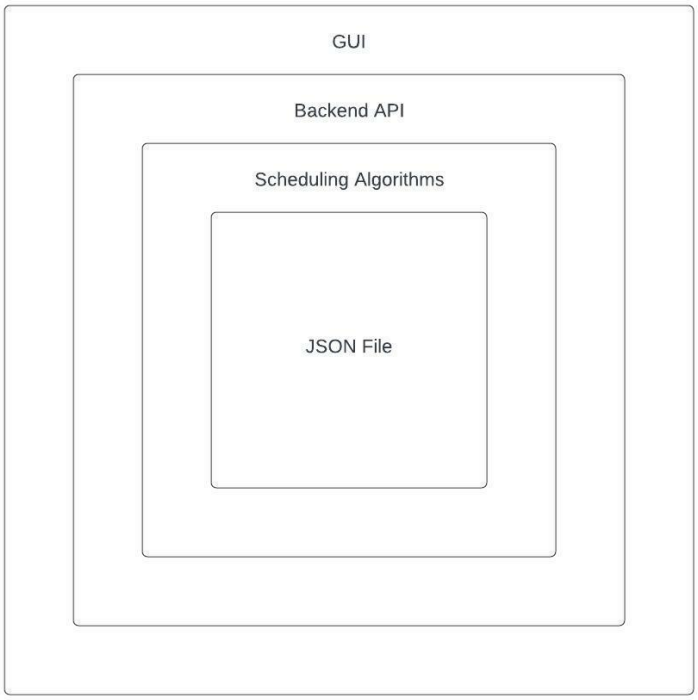
3.4 Technology Considerations

Since our project is a self-contained application, there are little to no considerations to be made in regard to hardware. The only considerations that we had to make are the language that we are developing our application in, and what frameworks to use to build the GUI. During our time working on this project, we determined that Python was in fact the best language to use since not only did it challenge us to work in a language the entire group had little knowledge of, it allowed us to learn about great Python frameworks like Tkinter.

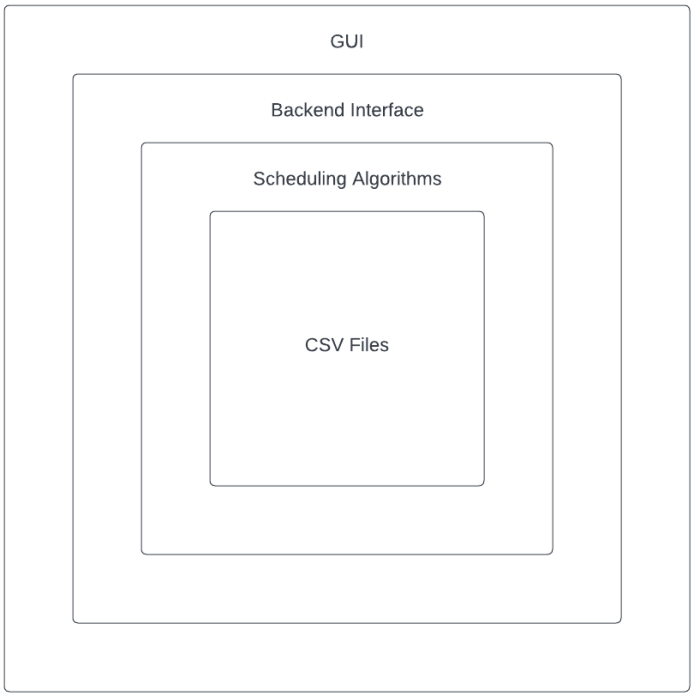
3.5 Design Evolution

Our design this semester did significantly change since CPR E 491. There are two primary changes that caused the significant change. The first was moving away from JSON files and switching to CSV files. Doing this caused the second change and that was how we are accessing the data. Initially we were going to convert the JSON objects within the files into Python dictionaries. From there we were going to create an API like interface where the frontend could call specific methods to change the data. However, since we moved away from JSON files we changed how the frontend would interact with the data. Now the frontend essentially uses an interface that opens the CSV files and uses the Python library Pandas to convert the files into dataframes. The frontend is then able to manipulate the data frames to carry out the various functionality. This not only reduced the complexity of the code but allowed us to reduce the amount of redundancy of data within the application.

Below is our initial design:



Below is our current design:



4 Testing

Testing took place mostly with the integration of the front and back end of the application.

4.1 UNIT TESTING

The units we will be testing are the various methods and functions that are implemented while we code. We believe that taking a granular approach like that will allow us to catch errors earlier on in the process and avoid last-minute debugging errors. Part of our approach will be to create a running document that lists the various methods and functions that are implemented, and within that documentation, it will include a section on how the methods and functions are expected to work. The programmer will also include what test cases they tested their methods and functions with and they will include the results of the tests.

4.2 INTERFACE TESTING

The primary interfaces we will be testing are the APIs that are implemented by the backend. These APIs are responsible for communicating with the database and the scheduling algorithm. These APIs are critical for implementing CRUD functionality within the application.

One interface that will be tested is the interface responsible for converting the JSON file into the application's memory and converting the application's memory back to the JSON file once the application is closed. This interface will be responsible for turning python dictionaries into JSON objects that can be stored within the JSON file and then will also be responsible for converting the JSON objects back into python dictionaries. This interface will be tested by the backend team using several test cases that ensure the required functionality is implemented and that data is not being corrupted during the process of converting it between python dictionaries and JSON objects. The primary goal when testing this interface is to ensure that CRUD functionality has been properly implemented and to make sure that the changes are persistent or ignored depending on how the application is terminated.

Another interface that will be tested is the interface responsible for implementing the CRUD functionality for the application. This interface is responsible for creating, reading, updating, and deleting courses within the database. This interface will be tested using various test cases that ensure the interface is working properly. The goal is to make sure that the various functionalities for the database are working properly and the changes are persistent. Additionally, we want to ensure that none of the data entered by the user is being corrupted by the methods and functions within the interface.

Again we will have the ability to use unit testing here in order to make sure that the implementation of the JSON files is correct.

4.3 INTEGRATION TESTING

Our application will have multiple critical integration paths, the first of which is the path that allows for the integration of the algorithms that work on the back in order to show what times are available for specific classes on the front. We will also have an integration path that relates to how we update and modify the JSON files from the front end.
For integration testing, we will be using the application in order to see if the algorithms in place

are affecting the outcomes of when specific classes can be scheduled. We may try to use the Robot testing framework.

4.4 System Testing

For our system-level testing strategy we are planning on creating multiple unit tests, potentially repurposing ones we have already made, and hooking them together to create a test for full system functionality. We will need to create a unit test that will simulate some user inputs to run the program, and then each step of our application can be tested in sequence and checked for expected outputs.

4.5 Regression Testing

By dividing the project in a large scale between the front end and the back end it makes it easy for the project to be very modular. Buttons on the front end can correlate to methods on the backend. This modularity makes the project resilient to breaking when new functionality is introduced.

4.6 Acceptance Testing

We plan on demonstrating the project to the client and talking through what requirements we have successfully satisfied and what requirements need more work in the future. Doing this repeatedly will result in all of the requirements being satisfied and the client being happy.

4.7 Security Testing (if applicable)

N/A

4.8 Results

The results of our testing strategies will help us locate bugs that are present in our current design. Since we are employing the Agile methodology, we are able to produce many versions of our product. With this testing framework, we were able to debug and develop the next versions of our design faster and more efficiently.

5 Implementation

5.1 Project Implementation

Our application had two main types of implementation's one being the front-end and the other being the back-end.

5.1.1 Front-End Implementation

We started with deciding which coding language we wanted to use and decided on python. The reason for this was because it was a language that we wanted to learn more about, In the beginning we were using a framework within python named PyQt5 but that ended up being rolled back when we found that we could use another python library named tkinter. There

seemed to be much more information online about how to utalize this library in order to make a decent GUI with python.

The functionality of our GUI includes displaying a CSV file that has the times and days of all the EcpE classes. We had also wanted to implement CRUD functionality but were unable to do so because of certin circumstances. In the GUI there is a spot on the tool bar to display CSV files, we are able to display a list of all classes and detales on each class. We wanted there to be a filter at the top of this but ended up not getting the filter functionality to work properly.

5.1.2 Back-End Implementation

The backend of our application is two-fold. There is a system for parsing relevant data from CSV files in order to display the class information through the GUI, and the algorithm used for automatically scheduling classes.

We decided to store all the necessary information for each course in the department within CSV files. These files contain the following for each course: department, course number, course name, course pre-reqs, and the course description. All the information was gathered from the Iowa State University course directory. The frontend is then able to parse the CSV files and display the information within the GUI.

The scheduling algorithm to automatically create a schedule of classes is currently unimplemented in our final design. It ended up being an unattainable goal for the scope of our project. The plan was to implement a genetic algorithm. A genetic algorithm is a heuristic approach to optimization. In the context of our project, many schedules are generated with completely randomized time-slots for each course and are subjected to mutations where only the most “fit” schedules are passed on to the next generation and mutated until a desired fitness level is reached.

5.2 Security Concerns and Countermeasures

5.2.1 Physical Security

We do not have any specific countermeasures for physical security threats. Since our application is standalone and does not run on external hardware, the primary countermeasure for this application is the user following best practices on their device. Depending on how the application is stored by the user, the only concern for physical security is ensuring their device is not damaged or stolen.

5.2.2 Cyber Security

As with physical security, since our application is standalone it does not require specific countermeasures for cyber security threats. The primary cyber security countermeasures for this application will be the user following best practices when it comes to their device. For example, ensuring their device is updated frequently and addressing any security concerns with their device. We considered having a login interface for the application but we decided that it did not

add enough security to outweigh the hit to the application being user friendly. This application is not responsible for turning in the application and that task is handled by our client. So, we believe that even though requiring the client to log into the application could prevent someone else from tampering with the schedule, the user friendliness of not having a login would be more favorable to the client. The client would just need to ensure that the correct schedule is being sent to the Registrar's office and that would address the concern of someone modifying the class schedule.

6 Appendices

6.1 Operation Manual

6.1.1 Editing/Opening application using Python

1. First you are going to need to go online to our https://git.ece.iastate.edu/sd/sddec22-09
    a. Click this icon  to download the zip file that will contain all of the project source code of the application.
    b. The latest version of the application is **mainbuild1.5.py**
    c. You can run this application if you have python installed on your PC

2. Check if you have python installed on your computer
    a. Windows
        i. Open Command Prompt > Type **Python**, or **py** > Hit Enter If Python is installed it will show the version details; otherwise it will open the Python page oin the Microsoft Store for download.
    b. Mac
        i. Python is most likely already installed on your system. To check the installation, go to Applications>Utilities and click on Terminal.
3. Install Python on Mac (Skip to Step 5 if installed)
    a. Check for the installed version of python with the command `python --version`
    b. First Homebrew needs to be installed this will make it easy to install the most recent version of Python start by installing Apple's xcode tools
        i. `xcode-select --install`
    c. Install Homebrew
        i. `ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"`
        ii. You can run brew doctor to see if the installation worked correctly

```
$ brew doctor
Your system is ready to brew.
```

d. Now you can use `brew` to install python packages. Start with installing Python itself.

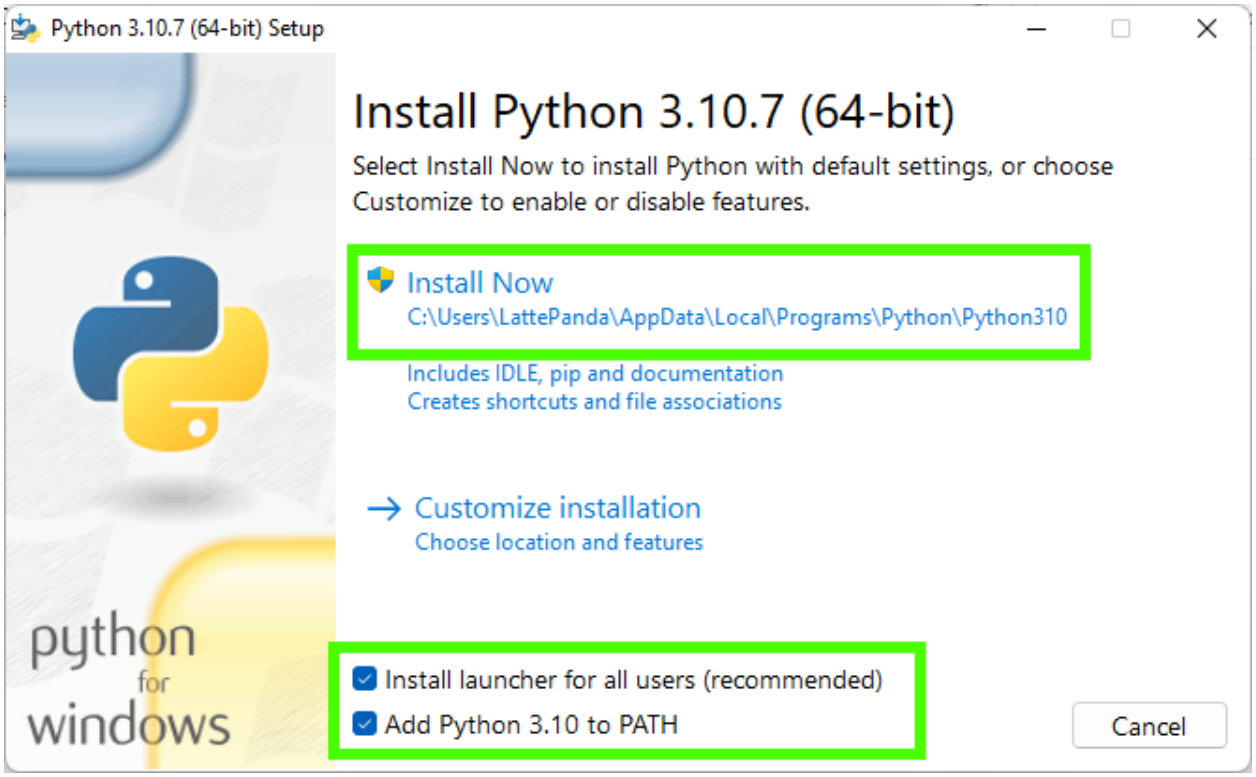
   i. `brew install python3`

e. Finally verify that Python installed correctly

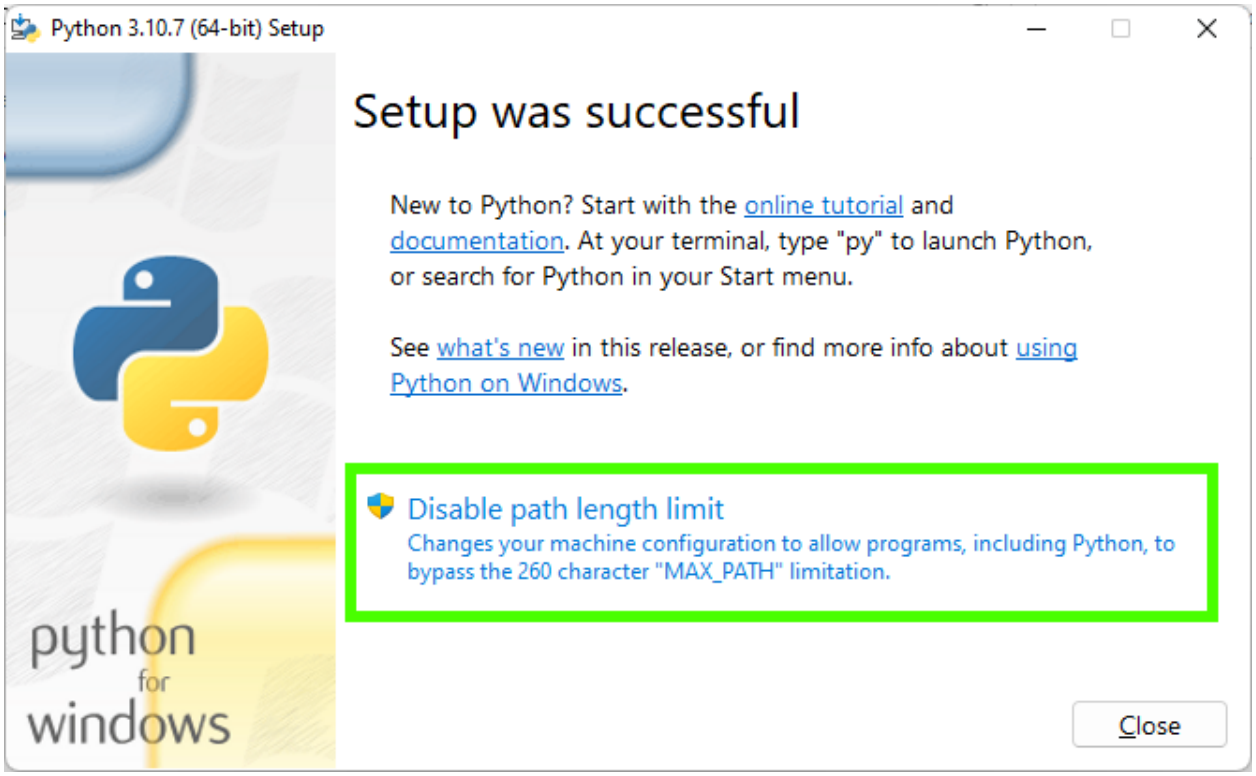
   i. `python3 --version`

```
$ python3 --version
Python 3.5.0
```

4. Install Python on Windows (skip to Step 5 if already installed)
   a. Go to the Python download page and download the Python Windows Executable
      i. https://www.python.org/downloads/
   b. Double click on the installer and click on "Install Now" to begin the installation.
      i. Adding python to the path will enable us to use the Python interpreter from any part of the filesystem.



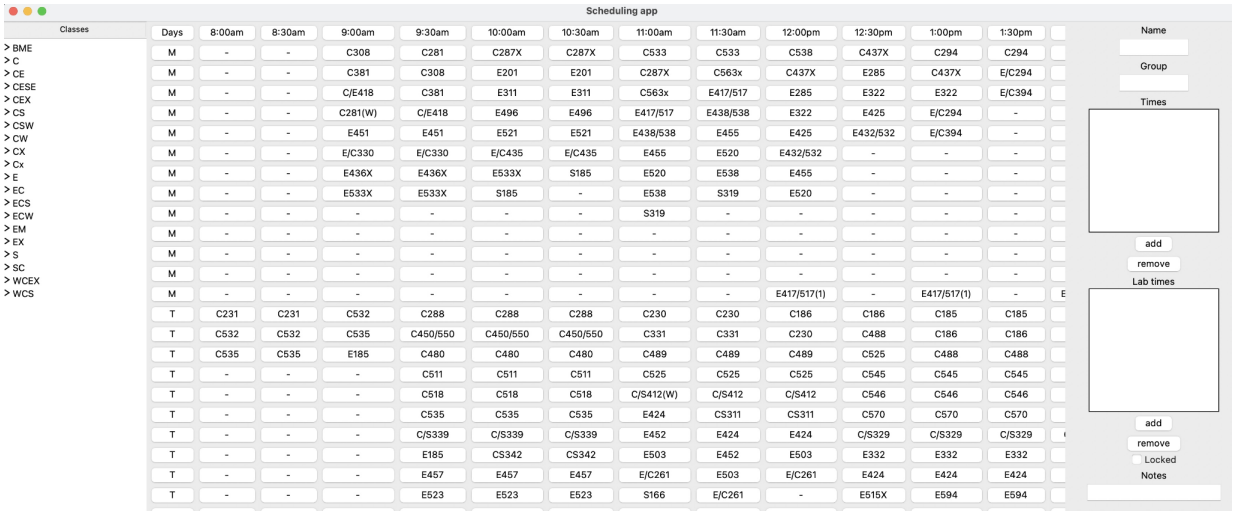
   c. After the installation is complete, click on **Disable Path Length Limit** as shown below and then Close.
      i. Disabling the path length limit means we can use more than 260 characters in a file path.



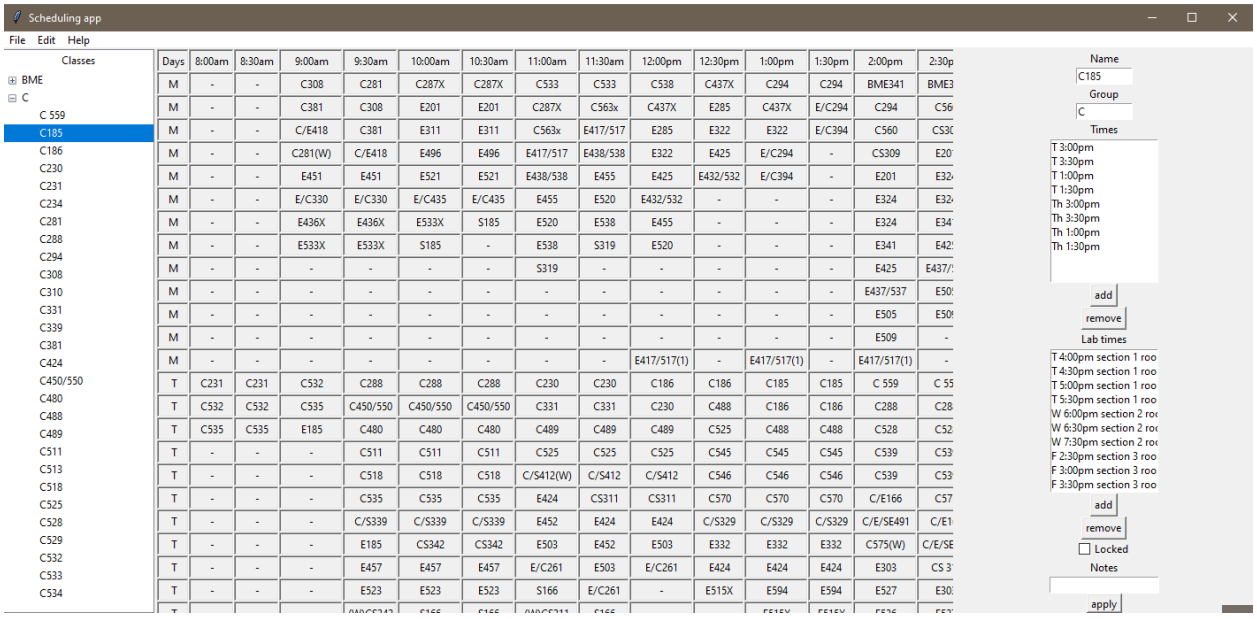
   d. Click Close to end the installation.
   e. Verify that python installed correctly
      i. Go to the Command Prompt and type in `Python -version`
5. After you have python installed you should be able to run the specific version of our application.
6. We used a few python modules that you will need to install using PIP
   a. Check if PIP is already installed by opening a command prompt and typing `pip -v` and pressing enter if installed skip to step 7
   b. Install PIP
      i. `python get-pip.py`
   c. You will now need to install a series of modules using pip run these commands.
      i. `pip install tkinter`
      ii. `pip install tkinter.ttk`
      iii. `pip install pickle`
      iv. `pip install numpy`
      v. `pip install pandas`
      vi. `pip install dataclasses`
      vii. `pip install os`
7. You should now be able to run any version of **mainbuild** that you would like.
8. Our application is open source so you may improve upon it as you wish.

6.1.2 Running and Using our application

1. Windows OS
    a. First you need to run the application ClassScheduler.exe



    b. File>Browse a CSV File
        i. Browse the CSV file that you would like to load up (be sure the CSV file is formatted equivalent to our template)
        ii. Click on apply on the bottom right and then you will see the class schedule displayed.
    c. You will be able to see a list of classes in the left frame.
        i. If you click on a specific class you will be able to see the attributes for that class in the right frame.



    d. Times can be edited by using the buttons in the right frame.
    e. File>Display Class Library
        i. This will bring up an application that will allow you to open other csv files one of interest might be the CSV we created with details about all of the classes

ii. There is a text box at the top of the new window that should allow you to filter results within the CSV file.

f. File>Save As
   i. This will save the current schedule as a .dat file which can then be loaded back into the application with File>Open

g. Help>About
   i. This will bring up the Authors of the program as well as a link to our website which will have more information.

6.2 Alternative Design Versions

6.2.1 Versions considered before client's specifications

The primary version we considered before meeting with the client was a web application approach. This approach would have consisted of a frontend made using AngularJS and Bootstrap. The backend would have been made up of Django and MongoDB. Reflecting back on this solution, we believe this solution would have been the best approach to solve this problem. Based on the client's constraints, we needed to build a standalone application, but that ended up being too restrictive.

6.2.2 Versions that resulted in failure to achieve specifications

There were several versions of this project that resulted in failure. Our first solution was very similar to our final solution, but the backend implemented JSON files and Python dictionaries instead of CSV files. With the constraints put on by the framework being used for the frontend, this solution did not seem viable. Our final solution also failed to achieve specifications. We moved towards a standalone application to make the application more user friendly, but it ended up restricting us too much to implement the features we were trying to achieve. We wanted to use CSV files to make the data more accessible to the client, but that caused us to face complex technical issues when trying to get the CSV files to work with the GUI.

6.3 Other Considerations

Reflecting back on this project, we learned that we should have spent more time in the beginning ensuring that we were on the right track. We did not spend enough time investigating all of the details involved with getting all of the moving parts for this project to work in harmony together. We should have also focused less on user friendliness and focused more on technical complexity of the project. Instead of implementing the project via a standalone application, we should have used a web application approach. This would have allowed us to leverage AngularJS and Bootstrap to make a great looking GUI, we could have used MongoDB to create a proper database for the courses, and leveraged Django to easily connect the database to the GUI.

6.4 Project Plan

6.4.1 PROJECT MANAGEMENT/TRACKING PROCEDURES

We are adopting agile. This fits our goals because for our project we can start with our base code and update it accordingly to accommodate feedback from the client. Being able to present earlier versions of the project to the client will allow us to get a better understanding of what they would like to see changed going forward. This will help reduce the likelihood of last-minute changes to the project and reduce the amount of last-minute debugging errors since we would be testing early and often.

We are utilizing Trello and Git to organize our goals and objectives for this project. These tools will help us stay organized and allow us to easily break large tasks into smaller and more manageable tasks. This will help reduce the complexity of the required tasks for the team to implement and allow us to have a clear path to implement the features within the project.

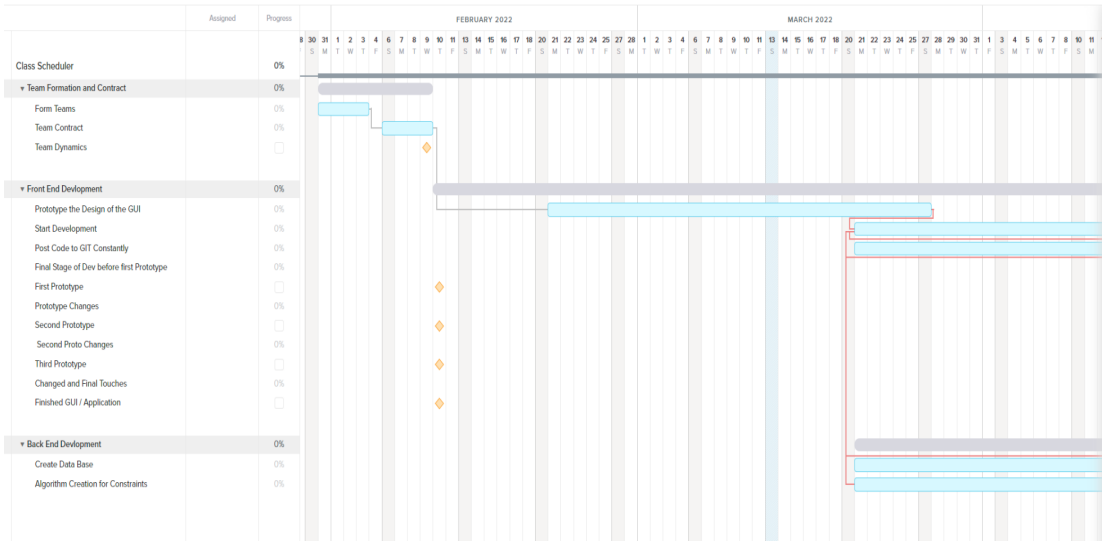
6.4.2 TASK DECOMPOSITION

- Front End:
    - UI/UX design for application
        - Time-table that outputs schedule that algorithm produces
            - Time-table is modifiable by user
        - “Lock” certain classes into specific time slots
            - Algorithm should be reran afterwards at user discretion
- Back End:
    - Create a proper database for storing application information
        - Classes and Labs
        - Corresponding Rooms
        - Class Prerequisites and Corequisites
    - Algorithm that schedules classes based on:
        - Class Prerequisites and Corequisites
        - Predefined times/rooms for certain classes
    - Create output packet that can be interpreted by front end application

6.4.3 PROJECT PROPOSED MILESTONES, METRICS, AND EVALUATION CRITERIA

1. For the frontend, functional GUI that has boxes that are able to be dragged and dropped (not connected to backend yet). For the back end, implement a database structure to store the various classes and labs.
2. For the frontend, functional GUI that is able to interact with the backend. For the backend, implement the CRUD functionality needed by the frontend.
3. For the frontend, implement the time table part of the GUI. For the backend, implement the scheduling algorithm that will generate the output for the class schedules.
4. For the frontend, add remaining features like locking the classes. For the backend, implement remaining features and refine the scheduling algorithm.

6.4.4 PROJECT TIMELINE/SCHEDULE



6.4.5 RISKS AND RISK MANAGEMENT/MITIGATION

Risks:	**Risk Probability:**	**Risk Mitigation:**
The framework we choose does not implement all of the capabilities that we need.	0.25	
The algorithm does not create the desired output.	0.5	Using the Agile methodology, we will test the algorithm early and often to make sure there are no last minute issues with the functionality of the algorithm.
The project expands beyond the original scope for the project and adds more features than we originally anticipated.	0.5	Using the Agile methodology will allow us to present several versions of the project to the client so we can add any additional features into the project earlier on and not at the last minute.
The GUI does not meet the clients expectations.	0.5	Using the Agile methodology will allow us to present several versions of the project

		to the client and allow us to see what they are looking for to improve the GUI.
Poor code quality that delays the debugging phase of the project.	0.2	

6.4.6 PERSONNEL EFFORT REQUIREMENTS

Front End Development	*100h*
Prototype Design	10h
Coding for Frontend / UI	80h
Research on Python Coding GUI	10h

Back End Development	*90h*
Algorithms for Creating Schedule	60h
Database Development	30h

6.4.7 OTHER RESOURCE REQUIREMENTS

The main resources we will need during the project is documentation on how to implement various tools within the framework to implement the functionality required for the project. Beyond that, we do not need any other resources for our project.

6.5 Team Contract

Team Members:

1) Zachary Bunch
2) Connor Gaecke
3) Chris Horvatich
4) Charles Mulderink

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:
    a. **Wednesdays at 4 pm on Discord**
2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
    a. **Discord**
3. Decision-making policy (e.g., consensus, majority vote):
    a. **Majority vote**
4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
    a. **The team leader will keep track of the meeting minutes.**

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
    a. **Yes, everyone is expected to be present and punctual, unless a notice is given in advance.**
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
    a. **Everyone is responsible for fulfilling their responsibilities. If a deadline can not be met the student will inform the group and we will jump in to help on that task.**
3. Expected level of communication with other team members:
    a. **As needed and weekly meetings.**
4. Expected level of commitment to team decisions and tasks:
    a. **Equal commitment among team members.**

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):
    a. **Chris Horvatich - Front end, testing, implementation**
    b. **Connor Gaecke - Back End, design and testing**
    c. **Charles Mulderink - Front end, interface design, prototyping**
    d. **Zachary Bunch- Team Leader, Back End, Client Interaction**
2. Strategies for supporting and guiding the work of all team members:
    a. **Constructive criticism**
3. Strategies for recognizing the contributions of all team members:
    a. **Using git, meetings where team members demonstrate what they have worked on.**

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.
    a. **Charles Mulderink - python knowledge, worked as a platform engineer.**
    b. **Chris Horvatich - A bit of python, CprE 309 gave me software expertise**
    c. **Connor Gaecke - Programming Knowledge, 309 group project experience**
    d. **Zachary Bunch- Python knowledge, 309 group project experience**
2. Strategies for encouraging and support contributions and ideas from all team members:
    a. **Grading and being a good team member/human**
3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)
    a. **Bring issues up to the team leader, and they will resolve the issue between the team members.**

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
    a. **Get the design/prototype ready for the class scheduler and start to implement.**
2. Strategies for planning and assigning individual and team work:
    a. **We will assign individual and team work using Trello.**
3. Strategies for keeping on task:
    a. **We will assign due dates on the Trello cards.**

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
    a. **Bring it up to the team and come to a solution as a team**
2. What will your team do if the infractions continue?
    a. **Talk to our faculty advisor about the problem**

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*
b) *I understand that I am obligated to abide by these terms and conditions.*
c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) Zachary Bunch DATE 2/09/2022
2) Connor Gaecke DATE 2/09/2022
3) Chris Horvatich DATE 02/09/2022
4) Charles Mulderink DATE 02/09/2022