EasyA
Project Plan

Spike Chen (sc), Joey Le (jl), Andrew Liu (al), Peter Nelson (pn), Krishna Patel (kp) - 2-5-2023 - v2.0

Table of Contents

1. Management Plan

1.1 Organization and Work Distribution

Each group member has been assigned subtasks that will contribute to modules that will make up the final project. The following objectives have been assigned to the following group members:

- Data Collection (Peter Nelson/Krishna Patel)
  - Made a python script to parse and turn the gradedata.js file provided in the project spec to a usable data format for the program.
  - Went through the archive of the faculty catalog to make a list containing first and last names of faculty in the natural sciences.
- Data Scraper (Peter Nelson)
  - Looked into implementing the scraper for the system

    - Used the scraper to turn HTML files into .txt files
- User Input (Andrew Liu)
    - Creation of the GUI and GUI elements
    - Output the graph to the GUI
- Documentation (Peter Nelson)
    - Wrote installation and setup documents for end users
    - Provided updates for any changes in installation and setup
    - Created a usage document showing how to how to accomplish real-world tasks using the software
- Data processing (Krishna Patel)
    - Took user input and processed the data accordingly
- Graph Generation (Joey Le)
    - Took processed data and created graphs using matplotlib
- Testing (Spike Chen)
    - Develop test cases to test the system as a whole
    - Provide insight in debugging for various components of the system
    - If there is a problem with a module, will report the issue to the person writing that module
- Making Diagrams (Joey Le)
    - Formed UML diagrams to match the design intentions of the system
- Record keeping (Joey Le)
    - Monitored and recorded the progress of each member throughout the project.
    - Recorded all important information discussed in meetings.
    - Managed documents that will contain the progress of each member and meetings.

We designed the responsibilities of each team member to be flexible in case someone needs help, the workload is unevenly distributed, or any other unexpected issues occur. When a decision that might impact the way the program modules work together needs to be made, the person making the change would propose that decision to the group. The team then suggests alternatives or addresses concerns. If there are neither of these then the original decision can be implemented, otherwise the team will discuss and an alternative will be carried out. Decisions that do not affect other modules do not need to be cleared unless the team member is unclear on an implementation decision.

Additionally, we used GitHub to store the different modules. It also allows us to see each other's work and better understand the rationale for any decisions made within the modules.

1.2 Communications and Meetings

Team members are expected to attend regular scheduled meetings at:

- Tuesdays at 11:00 AM

- Fridays at 4:00PM

All meetings were held in the Price Science Commons and lasted for one hour with the potential of going over a few minutes. If team members felt an additional meeting is helpful, we scheduled one for a time that will work for everyone.

Communication outside of scheduled meetings mainly happened over Discord. There were regular check-ins over Discord text to get an idea of where everyone is at with their progress.

1.3 Discord as a Communication and Management Platform

Discord was selected as our main platform to communicate and manage the project for many reasons. This platform contains the ability to text in group chats and make use of servers. A Discord server is a multi-layered group chat system that stores multiple group chats; each group chat is known as a channel.

The reasons that we used Discord are listed below:

- It was a convenient means of communication that each member had.
- Important documents such as the SDS and meeting logs can be stored in a separate channel in our server for easy access.
- We can receive automated notifications whenever a push was made to our project's repository.
- We can format any text to specify a direct response to a previous text, to address a member directly, and to distinguish code lines from regular text.

When using Discord**,** we expect that our fellow members will be respectful of any active conversations and not disturb our group chats with off-topic information. Normally, each member would ask a specific question and sometimes address it to a specific member or provide an update on their work. A member is also allowed to propose any ideas and plans for the project's development. Whenever a member discusses something that requires some response, other members usually respond within three hours.

With our different schedules, we were quite flexible when expecting any form of response. Members are expected to be kept to date with any public conversations, but whenever they have the time to check. This also applies to voice chat meetings; we only proposed voice chat meetings when we can find a common time that would work for everyone involved.

2. Work Breakdown Schedule

Pending any changes, team meetings will be scheduled on Tuesdays and Fridays. Daily online collaboration will be organized on the Discord web-service. The project due date is February 5, and project milestones will be discussed at in-person meetings.

Original milestones and tentative deadlines

- Parser written and tested (provides correct csv file) - **JAN 19**
- Faculty list collected (create a text file for faculty) - **JAN 19**
- Main .py file written (EasyA.py) - **JAN 18**
- StudentInterface .py file written -**JAN 23**
- EasyA.py receives user interface from StudentInterface.py - **JAN 23**
- Search.py file written - **JAN 25**
- Search.py returns data based on class number search - **JAN 25**
- Search.py returns data based on department search - **JAN 25**
- Search.py returns data based on level search - **JAN 31**
- Web scraper finished - **JAN 31**
- Search.py filters data by faculty list - **JAN 31**
- Graph generation .py file written - **FEB 2**
- EasyA.py can send data to GraphGen.py and receive a graph - **FEB 2**
- Display graph from GraphGen.py - **FEB 2**
- Addition of new data to csv/json through separate program - **FEB 2**
- Project delivery - **FEB 5**

Confirmed Timeline

Note: dates for milestones involving written programs define "written" as when the program performs the basic features as intended; modifications and adjustments could have been made past the provided date.

- Parser written and tested (provides correct csv file) - **JAN 19**
- Faculty list collected (create a text file for faculty) - **JAN 29**
- Rough Main .py file written (EasyA.py) - **JAN 19**
- Search.py file written - **JAN 21**
- GraphGen.py file written - **JAN 24**
- Search.py returns data based on class number search - **JAN 25**
- Search.py returns data based on department search - **JAN 25**
- Search.py returns data based on level search - **JAN 31**
- Web scraper finished - **JAN 29**
- StudentInterface.py file written - **JAN 30**
- Search.py filters data by faculty list - **JAN 30**
- Display graph from StudentInterface.py - **JAN 30**
- GUI can send user options to search.py and receive relevant data results - **JAN 31**
- GUI can send data to GraphGen.py and receive a graph - **JAN 31**
- GraphGen.py and Search.py integrated into StudentInterface.py - **FEB 3**
- Addition of new data to csv/json through separate program - **FEB 3**
- Documentation Created and Completely Revised **- FEB 4**

- Project delivery - **FEB 5**

3. Monitoring and Reporting

The bulk of monitoring and reporting about each member's progress were done primarily through In-person meetings and Discord texts. Here, we provided updates about our experiences in our designated roles. At the start of every meeting, each member gave a status report on their work. Outside of meetings, members would receive automated notifications on significant project updates through GitHub pushes.

All this information was recorded in our meeting logs and progress logs. Meeting logs stored information discussed in each meeting which includes members' progress, where the meeting was held and when, who was attending, and topic discussed. Progress logs were developed in the form of Gantt Chart. This allowed us to compare the work breakdown schedule to monitor and report on the project's progress. With our use of GitHub, it allows every member to see the specific adjustments and progress done for each file and when these occur. We can then compare it to the Gantt Chart to assess our development.

4. Build Plan

The program will be broken into seven main parts which will all be their own individual python files.

4.1 EasyA.py

EasyA.py was implemented simultaneously with the other parts of the program and served as the way to initiate the GUI.

4.2 parser.py

Parse.py will read through the gradedata.js file we have been provided and will store all class and instructor data in a comma separated value (CSV) file that all other parts of the program will rely on.

4.3 scraper.py

Scraper.py will read the different HTML files that we have provided and store the different faculty member names in a text (TXT) file that all other parts of the program will rely on.

4.4 search.py

After those the parser and scraper were completed, the next part was to implement Search.py. This part will look through the CSV file that was generated with input from StudentInterface.py so it is known what the relevant data that needs to be found is.

4.5 GraphGen.py
While the searching was being set up and tested, we were implementing GraphGen.py. This file will handle the visual output that the user expects to see in the form of multiple graphs. The data in the graphs will be obtained from search.py as it will be responsible for analyzing the CSV file.

4.6 StudentInterface.py
StudentInterface.py will prompt the user for the types of graphs they want to see through a text-based interface. This interface will call upon functions needed to search through the data and to generate the graphs; the graphs will appear on this interface. This was developed slightly after search.py and GraphGen.py for this file to easily search for data and search results.

4.7 addData.py
With all of the core functionality for the existing data implemented, we created AddData.py which will be used for adding or modifying class and instructor data to the existing dataset.

5. Build Plan Rationale
The reasoning for dividing our program up into these seven parts was to separate the complex processes into more manageable chunks. This way the testing and development process can be streamlined. Additionally, it adds a level of encapsulation as changes between multiple parts are contained. This also means modification to our code for new features can be easier. As these are contained, it allows much concurrency to occur so that each program can be created as soon as possible, which allows more time to implement and finetune the system as a whole.

The rationale behind the steps of building the system is the way the individual portions rely on each other. We are choosing to implement parse.py first because the data is fundamental to the whole program. Then we are implementing scraper.py, search.py, and graphGen.py because we have ways to develop them independently from each other and the system; these also relied on how we chose to format the data. From there, it gave us a better idea of how to develop the GUI for studentInterface. It also formats how users would input information and the graphs to display, so it needed search.py and graphGen.py to work from. addData.py is implemented last because the addition of new data will not impact the workings of the other parts of the program.

The main risks we have are not being able to meet our deadlines and development issues that may break the program. One of our risk reduction strategies is dividing up the program into smaller parts as incremental development reduces the chance of large errors. Additionally the encapsulation of features means that we know where an issue may be occurring if code is modified. However, we are able to address this with our concurrent operation.

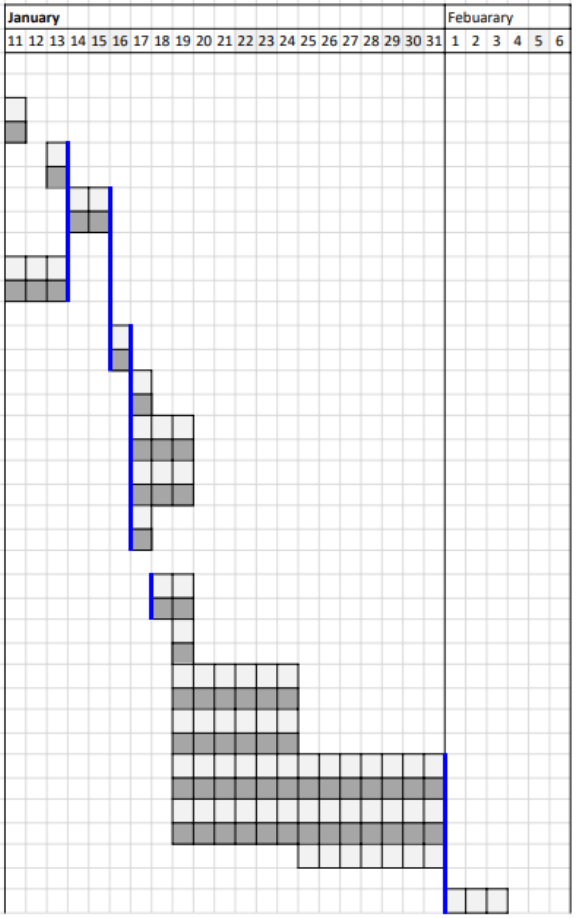
6. Timeline Gantt Chart

EasyA Software Project

(You just need to set the date on the first day, and adjust or remove the weekend markers.)

Project Timeline as of: 1/24/2023
The timeline is maintained by: JL: Joey Le
Team Members: SC: Spike Chen
AL: Andrew Liu
PN: Peter Nelson
KP: Krishna Patel

Tasks and Milestones	Assigned to...	Expected Days
Project Planning		
Assemble team.	Everyone	1
Attend first group meeting and establish roles	Everyone	1
Research and write-up on respective roles for Project Planning and SDS	Everyone	3
Requirements Analysis		
Read provided SRS	Everyone	3
Design		
Establish CSV file format	PN, KP	1
Establish way to store CSV file information within system	JL, KP	1
Establish basic layout for the user inferface	AL	3
Brainstorm potential test cases for Search.py	SC	3
Revise establish designs based on initial submission feedback	Everyone	1
Implementation		
Create Parser.py to parse data into a CSV file	PN, KP	2
Create rough layout of EasyA.py to interact with the entire system	JL	1
Design the user interface to allow the use of Search.py, GraphGen.py, and adding data	AL	6
Create Search.py to store and calulculate the requested data from the user	KP	6
Create GraphGen.py to generate the graph based on Search.py's results	JL	13
Research scraper and implement it into the system	SC, PN	13
Search.py is able to search based on all classes in a specific hundred-level and filter out facu	KP	7
Integrate all system modules together	Everyone	4



Testing				
	Test each important component individually	Everyone	2	
	Test the entire system as a whole	SC		
	Key			
	The tall rows show when the task was planned to be worked on and completed.			
	Planned activity.			
	Work done on the project.			
	Estalish an order where the task/milestone needed to be done before the next one starts