Interactive SEAS Course Program Planner

CS 360
Sprint 4

The Group

Noah Smiley, noah.smiley952@topper.wku.edu
Michaela Snyder, michaela.snyder023@topper.wku.edu
Diego Diaz, diegoalberto.minayadiaz650@topper.wku.edu

The Client

Dr. Michael Galloway, CS Professor , jeffrey.galloway@wku.edu

The Task to be Undertaken

Develop a software that allows students to create a course plan for their SEAS major, taking in consideration previously taken courses, course rotations and prerequisites for the enrollment of each course.

Course registration takes care of the development and learning journey of each student through their years in university. The School of Engineering and Applied Sciences in Western Kentucky university offers 10 undergraduate programs and 2 graduate programs; each one with their own course requirements and degree path. Our software aims to let SEAS students customize their schedule while preserving previous information as well as any future updates about their courses and SEAS programm. Develop a software that allows students to create a course plan for their SEAS major, taking in consideration previously taken courses, course rotations and prerequisites for the enrollment of each course.

Benefits:

The current structure of course planning requires students to check their ICAP (Interactive Curriculum and Academic Progress) and at the same time select their courses for the next semester. Such kind of instructure limits could be confusing and not very intuitive. Therefore our Interactive SEAS Programs Course Planner wll avoid this issue by implementing the following features:

1. The proposed course planning would give students a short term (semester) and long term (4 years) plan to complete their degree. The long term plan will need to fulfill all the course requirements to complete their degree while the short term will need to have at least 12 credit hours to be accepted.

2. The proposed course planning application would provide a framework for students to visually and intuitively locate their courses for their following years based on their course requirements and preferences; thus overcoming the challenges of the current catalog system and promoting the accessibility of these valuable resources.

3. Each change in a degree path, course requirement or course schedule will be reflected in the course plan for those students who decided to take that course and/or whose major requires them to take that course in future, thus preventing misinformation about the courses or requirements to obtain a degree.

4. The proposed course planning application will store the student information, including their previous selected courses , their login information and if they successfully finished a course requirement.

Project Manager:

The project manager for this sprint rotated throughout the team to make sure that every member was held accountable for directing the group's efforts. Week 1 was Noah, Week 2 was Diego, Week 3 was Michaela, and Week 4 was Noah.

Hardware Architecture:

The Raspberry Pi that will be used in this project has a 4GB SDRAM, a quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz processor, gigabit Ethernet, Bluetooth 5.0, a microSD port, and is Power over Ethernet enabled. Unless the team develops inefficient code, this hardware should be more than enough to host the project.

A Preliminary Requirements Analysis:

The system needs to meet the following *functional* requirements:

1. Web Interface
    a) Administrator Side
        i. Allow administrative users (professors) to modify, delete, or add the course schedule information to their affiliated degree program.
    b) Public Side
        1. Allow SEAS students to input grades and previous courses to create compatible schedules and course pathways in the most efficient way possible.
        2. Would not be allowed to add a course if no previous requirements have been successfully accomplished or have been planned to be fulfilled before the beginning of the new course.
        3. Verify that the information provided for the public is accurate and up to date; any information related to course requirements and course availability must be consistent in order to diminish any confusion.
        4. Must be able to visualize a course plan for their semester/academic years and make changes if necessary.
        5. Be easily extensible from both the administrator's perspective and from future developers' perspectives.

Functional Requirements:

1. The project/tool(s) should use a web-based user interface.
2. The project should allow students to select and/or change majors within SEAS.
3. The project should allow students to enter previously taken courses (semester and grade).

4. The project should take course rotation into account and create a plan for finishing a SEAS degree.
5. The project should display all prerequisites for courses in the plan.
6. The project should implement a login system and save student information.

Non-functional Requirements:

1. The project should be implemented on an always-on Raspberry Pi 3 B, B+, or Raspberry Pi 4.
2. The project should be implemented using at least two Docker Containers.
3. The project should use the JavaScript language and D3.js packages
4. Performance metrics should be gathered and optimized.
5. Security metrics should be gathered and optimized.
6. User interface metrics should be gathered and optimized.

Technical Requirements – Feasibility:

1. *Server* – The server for this project will be run on an always on Raspberry Pi, allowing 24 hour access for users. The server will be remotely accessible to all team members at all times, allowing for updates and development at any and all times.

2. *Database* – The database will be appendable; instructors utilizing the system will be able to add available courses and times, which will make the courses available for students using the system for scheduling purposes.

3. *Web* – The entire project will be web-based, with the Raspberry Pi hosting the server. The web-based system will utilize web-building software and languages such as Docker and D3.js.

Scope:

For the purposes of this project we are solely concerned with creating an interactive

planner for use by students and instructors alike. We plan to include a login system that can differentiate between students and instructors. We plan to allow instructors to add their own courses within SEAS, for scheduling purposes. We plan to create an interface that lets students add their past courses and grades within their SEAS program, for prerequisite purposes.

We DO NOT plan to make courses not from SEAS available for scheduling. We DO NOT plan to allow students to add programs outside of SEAS. We DO NOT plan to allow students to have system privileges that instructors have.

Suggested Deliverables:

Management Deliverables:

1. *Requirements Analysis* – Included in this document, it is an overview of the functional and non-functional requirements agreed upon between the stakeholders. Ensures that the development team understands the requirements given by the client, and ensures that the system that is developed fulfills the wishes of stakeholders. Helps ensure a functional system.

2. *Design Document* – a document used to showcase and explain the design formulated by the team to the client. Shows the development plans for the system without code, and helps officially decide what is possible within the system and what is not. A visualization of the system.

3. *Source Code* – addended to the team's project document, this is the code developed for the system. This will include all code used in the system, including classes, methods, and test cases used. The code will have been tested extensively with multiple test cases.

4. *Progress Presentations* – presentations at the end of each sprint to update the client on the team's progress throughout the sprint. Will be around ten minutes and will summarize team efforts for that sprint.

Technical Deliverables:

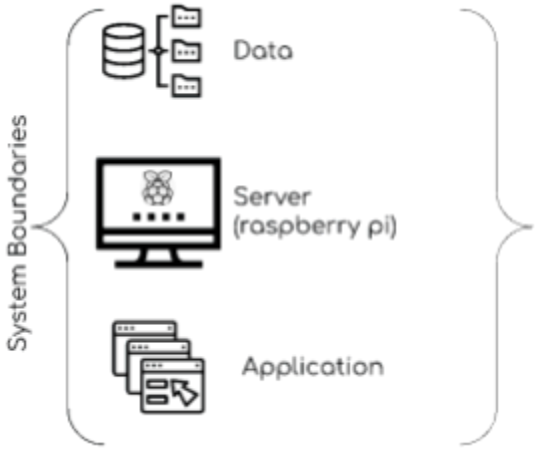
1. A *database* that courses for SEAS programs can be stored in. It will also need to be able to store student and administrator information for future logins and to store student schedules and information under their respective logins.

2. An *administrative interface* to add, modify, delete and search for SEAS courses. It must allow administrators to add not only courses, but the times and semesters when the courses are available. This interface must be available to only administrators.

3. A *course plan interface* for students in SEAS programs. This must allow students to view the courses necessary to complete a SEAS degree, and it must take course rotation into account. These course plans will also be accessible to administrators for advising purposes. Other students will not be able to view the course plans for other students.

4. A *login system* that can differentiate between student users and administrative users. Data will be saved under the corresponding login, and will be accessible to only that user – unless the user is a student, then the student's information will be available to administrators for advising purposes.

System Boundaries:



Physical:

Physical boundaries the team discovered include the shared disk, server, and application.

Logical:

Logical boundaries the team discovered include the web-based UI and the administrator and user classes, namely inheritance, strings, and integers.

Walk-Through:

To ensure understanding between the Client and the Team, this walkthrough has been prepared. This is not an exhaustive example of how the system will be used, but rather an explanation of basic usage.

Student user walk-through

The student is first presented with a login screen. If this is the student's first time using the system, the student will be asked to create an account and input their student ID number. Upon creation of the student's account, the student will be able to login to the same account at later times. Once logged in, the student will be able to add past courses and grades, according to semester and year. The student will be able to select a SEAS program. Once the student has fully entered all relevant information – courses, grades, program – the student will be able to view a course plan to complete the SEAS program that they selected.

Administrative walk-through

The administrator will be presented with a login screen. If the administrator has not used the system before, they will be prompted to create an account. Upon creation of the account, the administrator will be able to access an interface that allows them to enter courses and corresponding course details such as semester, year, credit hours, time, and weekdays. The administrator will also be able to utilise the system for advising purposes, and will be able to access course plans created by students using the student's student ID number.

Software Development Process:

The project will utilize the *modified waterfall model* because the requirements are well-defined by both the client and the team. Since the client has very specific needs and specifications for the system, and the system is a utility rather than a research program, this model gives the team the following benefits:

1. *Process visibility* – the client and the team are always aware of what sprint they are in.

2. *Separation of tasks* – the team may work on one problem and/or area at a time, which allows the team greater control over the project as a whole.

3. *Quality control* – the modified waterfall model allows the team to spend more time on each task because there is no rush to finish each sprint, which thus allows the team to create a system that is of a greater quality.

Outline Plan (Principal activities and Sprints)

I. Sprint 1 (August 24th, 2020 – September 12th, 2020) – Feasibility Study and Requirements Analysis. A final draft for the study and analysis will be due at the end of the sprint, along with a presentation that summarizes the team's efforts throughout the sprint. A minimum of ten pages for the study.

II. Sprint 2 (September 13th, 2020 – October 10th, 2020) – Software Modelling and Design. The existing document from Sprint 1 will be revised and amended to include software modelling and design data by the end of the sprint, along with a presentation that summarizes the team's efforts throughout the sprint. A minimum of twenty pages for the revised document.

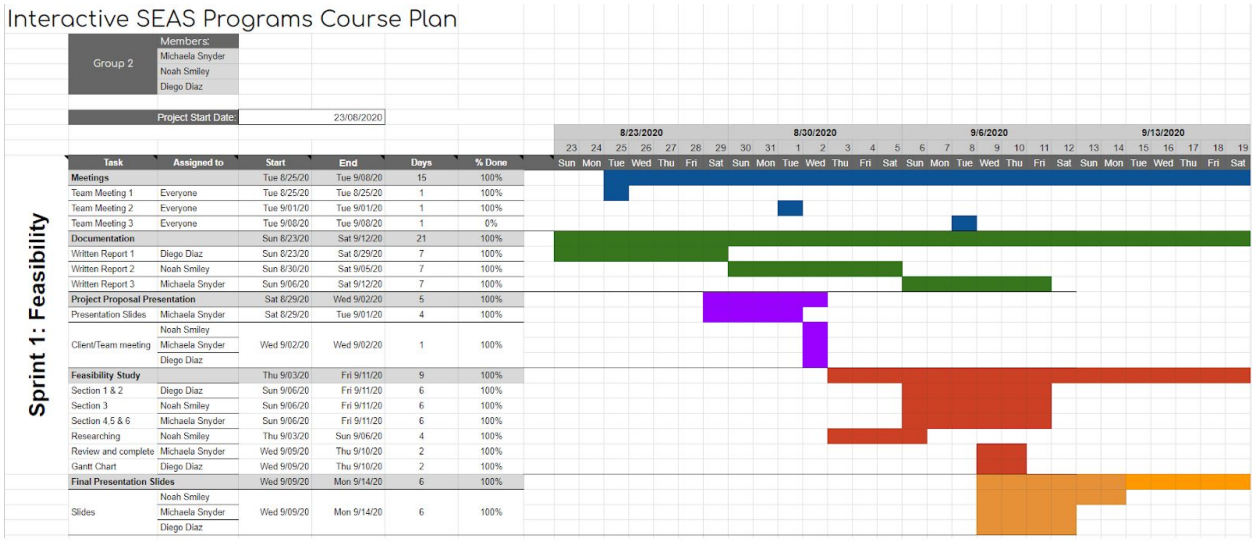
III. Sprint 3 (October 11th, 2020 – November 7th, 2020) – Software Implementation. The existing document will be revised to include a description of implementation activities undertaken by the team by the end of the sprint, along with a presentation that summarizes the team's efforts throughout the sprint. A minimum of thirty pages for the revised document.

IV. Sprint 4 (November 8th, 2020 – December 12th, 2020) – Software Testing. The existing document will be submitted in its final form, including all previous reports, studies, charts, and source code. A final presentation will also be due by the end of the sprint. The final document

has a minimum of forty pages.

Gantt Chart for Team Progress and Time Management:

Sprint 1:



Interactive SEAS Programs Course Plan

Group 2	Members:
	Michaela Snyder
	Noah Smiley
	Diego Diaz

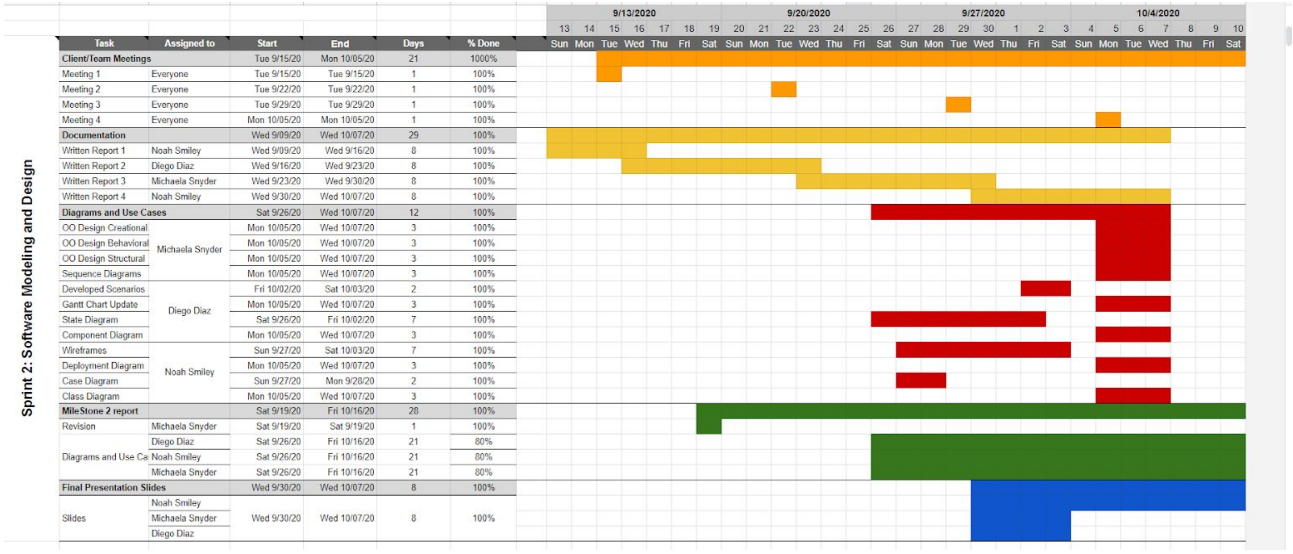
Project Start Date: 23/08/2020

Sprint 1: Feasibility

Task	Assigned to	Start	End	Days	% Done
Meetings		Tue 8/25/20	Tue 9/08/20	15	100%
Team Meeting 1	Everyone	Tue 8/25/20	Tue 8/25/20	1	100%
Team Meeting 2	Everyone	Tue 9/01/20	Tue 9/01/20	1	100%
Team Meeting 3	Everyone	Tue 9/08/20	Tue 9/08/20	1	0%
Documentation		Sun 8/23/20	Sat 9/12/20	21	100%
Written Report 1	Diego Diaz	Sun 8/23/20	Sat 8/29/20	7	100%
Written Report 2	Noah Smiley	Sun 8/30/20	Sat 9/05/20	7	100%
Written Report 3	Michaela Snyder	Sun 9/06/20	Sat 9/12/20	7	100%
Project Proposal Presentation		Sat 8/29/20	Wed 9/02/20	5	100%
Presentation Slides	Michaela Snyder	Sat 8/29/20	Tue 9/01/20	4	100%
Client/Team meeting	Noah Smiley Michaela Snyder Diego Diaz	Wed 9/02/20	Wed 9/02/20	1	100%
Feasibility Study		Thu 9/03/20	Fri 9/11/20	9	100%
Section 1 & 2	Diego Diaz	Sun 9/06/20	Fri 9/11/20	6	100%
Section 3	Noah Smiley	Sun 9/06/20	Fri 9/11/20	6	100%
Section 4,5 & 6	Michaela Snyder	Sun 9/06/20	Fri 9/11/20	6	100%
Researching	Noah Smiley	Thu 9/03/20	Sun 9/06/20	4	100%
Review and complete	Michaela Snyder	Wed 9/09/20	Thu 9/10/20	2	100%
Gantt Chart	Diego Diaz	Wed 9/09/20	Thu 9/10/20	2	100%
Final Presentation Slides		Wed 9/09/20	Mon 9/14/20	6	100%
Slides	Noah Smiley Michaela Snyder Diego Diaz	Wed 9/09/20	Mon 9/14/20	6	100%

This Gantt chart shows team efforts throughout the first Sprint. The total amount of time spent between all team members is 31 hours. We spent the most time creating the Feasibility study and preparing our Weekly Reports, although towards the end of the sprint we spent the majority of our time on the Sprint 1 Final Presentation.

Sprint 2:

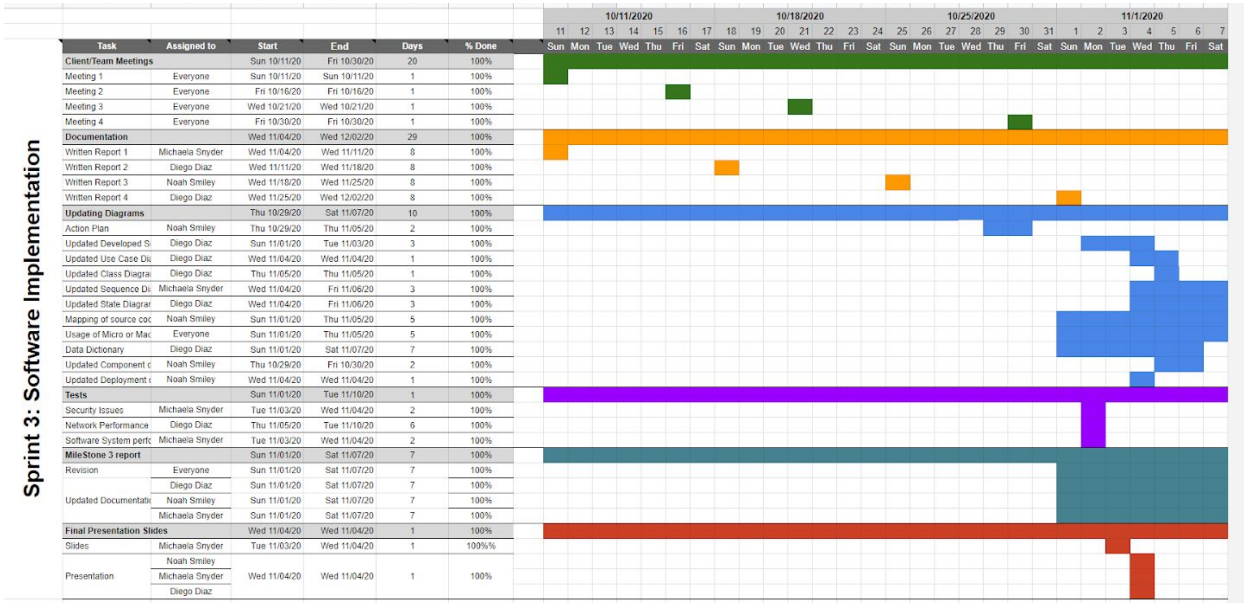


Sprint 2: Software Modeling and Design

Task	Assigned to	Start	End	Days	% Done
Client/Team Meetings		Tue 9/15/20	Mon 10/05/20	21	1000%
Meeting 1	Everyone	Tue 9/15/20	Tue 9/15/20	1	100%
Meeting 2	Everyone	Tue 9/22/20	Tue 9/22/20	1	100%
Meeting 3	Everyone	Tue 9/29/20	Tue 9/29/20	1	100%
Meeting 4	Everyone	Mon 10/05/20	Mon 10/05/20	1	100%
Documentation		Wed 9/09/20	Wed 10/07/20	29	100%
Written Report 1	Noah Smiley	Wed 9/09/20	Wed 9/16/20	8	100%
Written Report 2	Diego Diaz	Wed 9/16/20	Wed 9/23/20	8	100%
Written Report 3	Michaela Snyder	Wed 9/23/20	Wed 9/30/20	8	100%
Written Report 4	Noah Smiley	Wed 9/30/20	Wed 10/07/20	8	100%
Diagrams and Use Cases		Sat 9/26/20	Wed 10/07/20	12	100%
OO Design Creational	Michaela Snyder	Mon 10/05/20	Wed 10/07/20	3	100%
OO Design Behavioral		Mon 10/05/20	Wed 10/07/20	3	100%
OO Design Structural		Mon 10/05/20	Wed 10/07/20	3	100%
Sequence Diagrams		Mon 10/05/20	Wed 10/07/20	3	100%
Developed Scenarios	Diego Diaz	Fri 10/02/20	Sat 10/03/20	2	100%
Gantt Chart Update		Mon 10/05/20	Wed 10/07/20	3	100%
State Diagram		Sat 9/26/20	Fri 10/02/20	7	100%
Component Diagram		Mon 10/05/20	Wed 10/07/20	3	100%
Wireframes	Noah Smiley	Sun 9/27/20	Sat 10/03/20	7	100%
Deployment Diagram		Mon 10/05/20	Wed 10/07/20	3	100%
Case Diagram		Sun 9/27/20	Mon 9/28/20	2	100%
Class Diagram		Mon 10/05/20	Wed 10/07/20	3	100%
MileStone 2 report		Sat 9/19/20	Fri 10/16/20	28	100%
Revision	Michaela Snyder	Sat 9/19/20	Sat 9/19/20	1	100%
	Diego Diaz	Sat 9/26/20	Fri 10/16/20	21	80%
Diagrams and Use Ca	Noah Smiley	Sat 9/26/20	Fri 10/16/20	21	80%
	Michaela Snyder	Sat 9/26/20	Fri 10/16/20	21	80%
Final Presentation Slides		Wed 9/30/20	Wed 10/07/20	8	100%
Slides	Noah Smiley Michaela Snyder Diego Diaz	Wed 9/30/20	Wed 10/07/20	8	100%

This Gantt chart shows team efforts throughout the second Sprint. The total amount of time spent between all team members is approximately 38 hours. We spent most of the time preparing the diagrams and designing the wireframes. The team's time management has improved since Sprint 1.

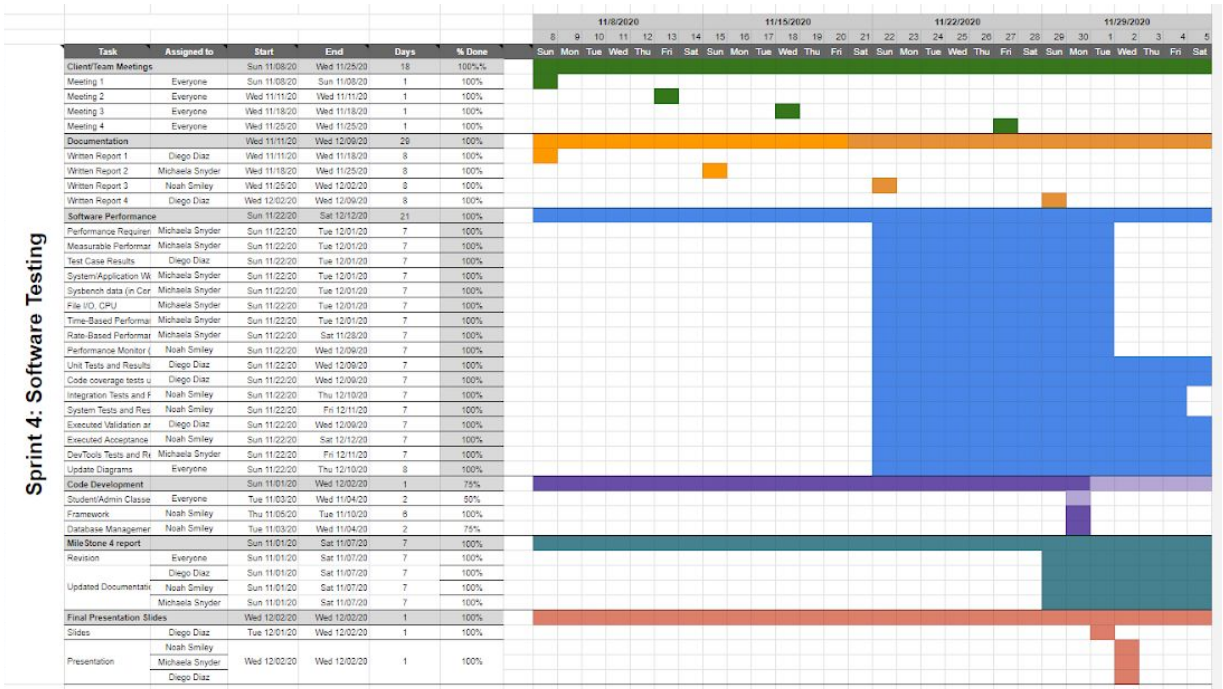
Sprint 3:



Task	Assigned to	Start	End	Days	% Done
Client/Team Meetings		Sun 10/11/20	Fri 10/30/20	20	100%
Meeting 1	Everyone	Sun 10/11/20	Sun 10/11/20	1	100%
Meeting 2	Everyone	Fri 10/16/20	Fri 10/16/20	1	100%
Meeting 3	Everyone	Wed 10/21/20	Wed 10/21/20	1	100%
Meeting 4	Everyone	Fri 10/30/20	Fri 10/30/20	1	100%
Documentation		Wed 11/04/20	Wed 12/02/20	29	100%
Written Report 1	Michaela Snyder	Wed 11/04/20	Wed 11/11/20	8	100%
Written Report 2	Diego Diaz	Wed 11/11/20	Wed 11/18/20	8	100%
Written Report 3	Noah Smiley	Wed 11/18/20	Wed 11/25/20	8	100%
Written Report 4	Diego Diaz	Wed 11/25/20	Wed 12/02/20	8	100%
Updating Diagrams		Thu 10/29/20	Sat 11/07/20	10	100%
Action Plan	Noah Smiley	Thu 10/29/20	Thu 11/05/20	2	100%
Updated Developed S	Diego Diaz	Sun 11/01/20	Tue 11/03/20	3	100%
Updated Use Case Di	Diego Diaz	Wed 11/04/20	Wed 11/04/20	1	100%
Updated Class Diagra	Diego Diaz	Thu 11/05/20	Thu 11/05/20	1	100%
Updated Sequence Di	Michaela Snyder	Wed 11/04/20	Fri 11/06/20	3	100%
Updated State Diagra	Diego Diaz	Wed 11/04/20	Fri 11/06/20	3	100%
Mapping of source coc	Noah Smiley	Sun 11/01/20	Thu 11/05/20	5	100%
Usage of Micro or Mac	Everyone	Sun 11/01/20	Thu 11/05/20	5	100%
Data Dictionary	Diego Diaz	Sun 11/01/20	Sat 11/07/20	7	100%
Updated Component d	Noah Smiley	Thu 10/29/20	Fri 10/30/20	2	100%
Updated Deployment d	Noah Smiley	Wed 11/04/20	Wed 11/04/20	1	100%
Tests		Sun 11/01/20	Tue 11/10/20	1	100%
Security Issues	Michaela Snyder	Tue 11/03/20	Wed 11/04/20	2	100%
Network Performance	Diego Diaz	Thu 11/05/20	Tue 11/10/20	6	100%
Software System perfo	Michaela Snyder	Tue 11/03/20	Wed 11/04/20	2	100%
MileStone 3 report		Sun 11/01/20	Sat 11/07/20	7	100%
Revision	Everyone	Sun 11/01/20	Sat 11/07/20	7	100%
Updated Documentatio	Diego Diaz	Sun 11/01/20	Sat 11/07/20	7	100%
	Noah Smiley	Sun 11/01/20	Sat 11/07/20	7	100%
	Michaela Snyder	Sun 11/01/20	Sat 11/07/20	7	100%
Final Presentation Slides		Wed 11/04/20	Wed 11/04/20	1	100%
Slides	Michaela Snyder	Tue 11/03/20	Wed 11/04/20	1	100%
Presentation	Noah Smiley Michaela Snyder Diego Diaz	Wed 11/04/20	Wed 11/04/20	1	100%

This Gantt chart shows team efforts throughout the third Sprint. The total amount of time spent between all team members is approximately 48 hours. Most of that time was spent developing our initial web pages and making sure external services (remote connection, sysbench, Django) work properly. The team's time management has improved since sprint 2 and the team has managed to take care of more tasks.

Sprint 4:

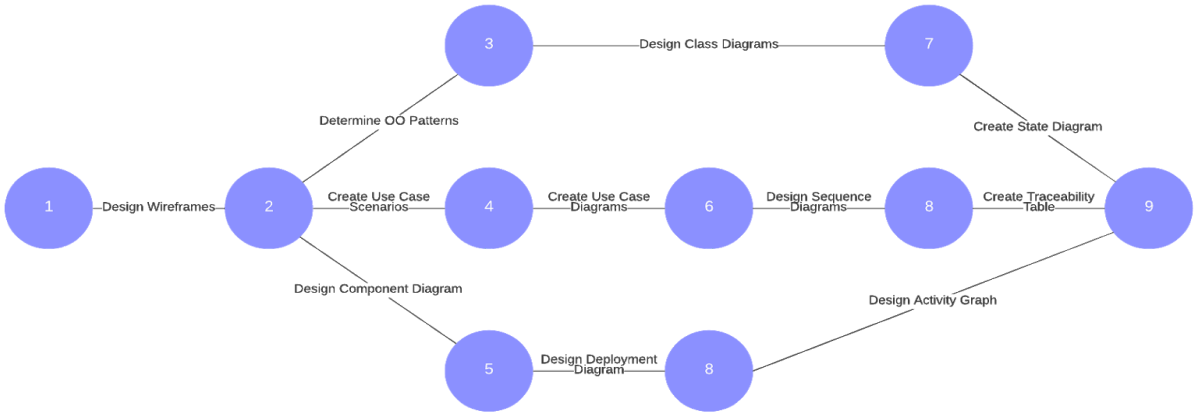


Task	Assigned to	Start	End	Days	% Done
Client/Team Meetings		Sun 11/08/20	Wed 11/25/20	18	100%%
Meeting 1	Everyone	Sun 11/08/20	Sun 11/08/20	1	100%
Meeting 2	Everyone	Wed 11/11/20	Wed 11/11/20	1	100%
Meeting 3	Everyone	Wed 11/18/20	Wed 11/18/20	1	100%
Meeting 4	Everyone	Wed 11/25/20	Wed 11/25/20	1	100%
Documentation		Wed 11/11/20	Wed 12/09/20	29	100%
Written Report 1	Diego Diaz	Wed 11/11/20	Wed 11/18/20	8	100%
Written Report 2	Michaela Snyder	Wed 11/18/20	Wed 11/25/20	8	100%
Written Report 3	Noah Smiley	Wed 11/25/20	Wed 12/02/20	8	100%
Written Report 4	Diego Diaz	Wed 12/02/20	Wed 12/09/20	8	100%
Software Performance		Sun 11/22/20	Sat 12/12/20	21	100%
Performance Requirem	Michaela Snyder	Sun 11/22/20	Tue 12/01/20	7	100%
Measurable Performar	Michaela Snyder	Sun 11/22/20	Tue 12/01/20	7	100%
Test Case Results	Diego Diaz	Sun 11/22/20	Tue 12/01/20	7	100%
System/Application We	Michaela Snyder	Sun 11/22/20	Tue 12/01/20	7	100%
Sysbench data (in Cer	Michaela Snyder	Sun 11/22/20	Tue 12/01/20	7	100%
File I/O, CPU	Michaela Snyder	Sun 11/22/20	Tue 12/01/20	7	100%
Time-Based Performar	Michaela Snyder	Sun 11/22/20	Tue 12/01/20	7	100%
Rate-Based Performar	Michaela Snyder	Sun 11/22/20	Sat 11/28/20	7	100%
Performance Monitor (	Noah Smiley	Sun 11/22/20	Wed 12/09/20	7	100%
Unit Tests and Results	Diego Diaz	Sun 11/22/20	Wed 12/09/20	7	100%
Code coverage tests u	Diego Diaz	Sun 11/22/20	Wed 12/09/20	7	100%
Integration Tests and F	Noah Smiley	Sun 11/22/20	Thu 12/10/20	7	100%
System Tests and Res	Noah Smiley	Sun 11/22/20	Fri 12/11/20	7	100%
Executed Validation ar	Diego Diaz	Sun 11/22/20	Wed 12/09/20	7	100%
Executed Acceptance	Noah Smiley	Sun 11/22/20	Sat 12/12/20	7	100%
DevTools Tests and Re	Michaela Snyder	Sun 11/22/20	Fri 12/11/20	7	100%
Update Diagrams	Everyone	Sun 11/22/20	Thu 12/10/20	8	100%
Code Development		Sun 11/01/20	Wed 12/02/20	1	75%
Student/Admin Classe	Everyone	Tue 11/03/20	Wed 11/04/20	2	50%
Framework	Noah Smiley	Thu 11/05/20	Tue 11/10/20	6	100%
Database Managemer	Noah Smiley	Tue 11/03/20	Wed 11/04/20	2	75%
MileStone 4 report		Sun 11/01/20	Sat 11/07/20	7	100%
Revision	Everyone	Sun 11/01/20	Sat 11/07/20	7	100%
Updated Documentatio	Diego Diaz	Sun 11/01/20	Sat 11/07/20	7	100%
	Noah Smiley	Sun 11/01/20	Sat 11/07/20	7	100%
	Michaela Snyder	Sun 11/01/20	Sat 11/07/20	7	100%
Final Presentation Slides		Wed 12/02/20	Wed 12/02/20	1	100%
Slides	Diego Diaz	Tue 12/01/20	Wed 12/02/20	1	100%
Presentation	Noah Smiley Michaela Snyder Diego Diaz	Wed 12/02/20	Wed 12/02/20	1	100%

This Gantt chart shows the effort of the team to complete the project for the fourth Sprint. The total amount of time spent between all team members is approximately 55 hours. Most of the developing time was designated to have a working product and having the results of different tests for the software/hardware performance. The number of tasks assigned for the team this semester was greater than the previous ones but the team made more meetings during this sprint to make sure that all the requirements were meet before hand.

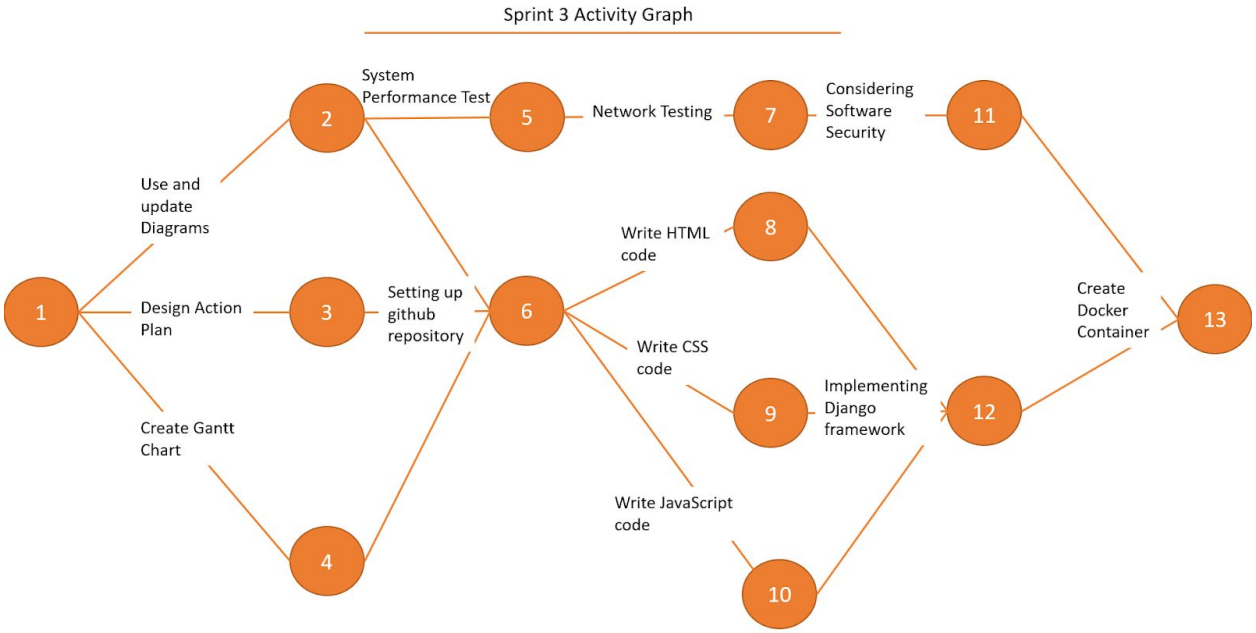
Activity Graph

Sprint 2:



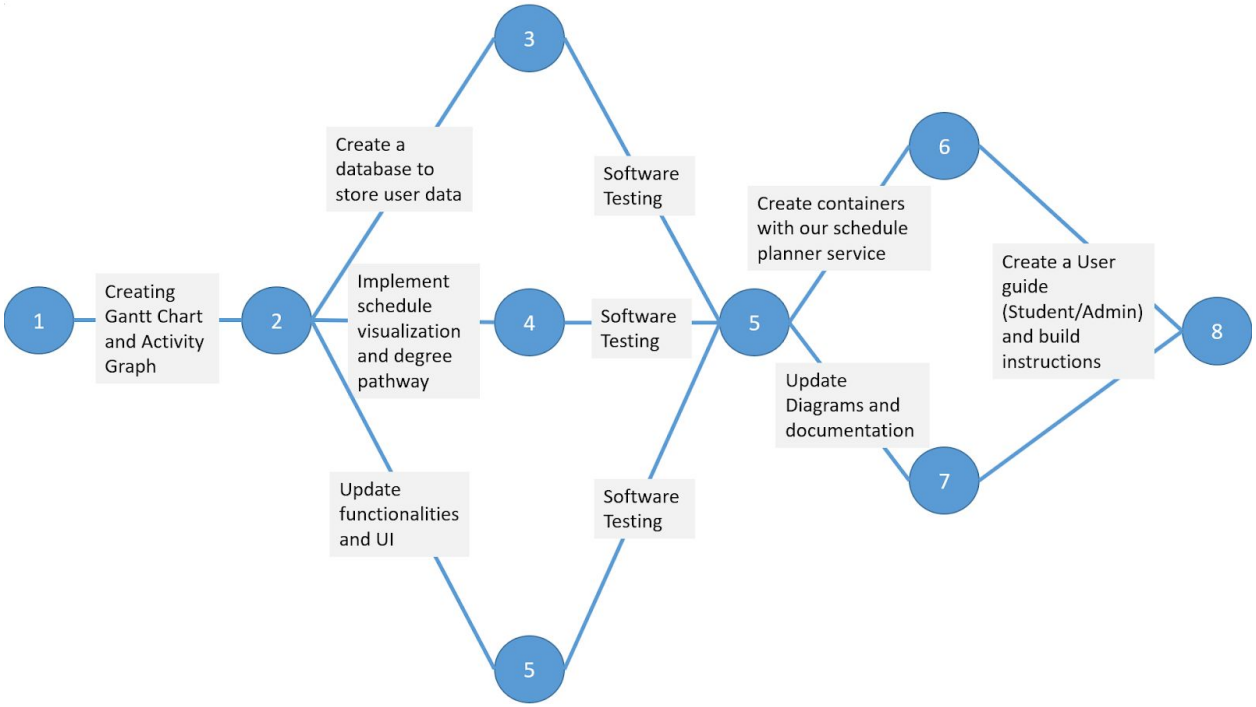
This activity graph shows a graphical representation of the team's workflow throughout the second Sprint. It took 4 - 5 hours on average to complete each activity. Most of the activities consist of creating and designing diagrams for our project. Each node represents a different step in our workflow and each edge the activity that the team made to progress through each node.

Sprint 3:



This activity graph shows a graphical representation of the team's workflow throughout the third Sprint. The time it took to complete each activity is too wide to include an average but the total amount of time was approximately 48 hours . Most of the activities consist of developing code and checking external services. Each node represents a different step in our workflow and each edge the activity that the team made to progress through each node

Sprint 4:



This activity graph shows the plan that the team followed during the sprint 4 to develop this documentation and to improve the development of the project. The time it took to complete each activity is between 1 to 9 hours depending on the task. Most of the activities consist of developing code and making different tests. Each node represents a different step in our workflow and each edge the activity that the team made to progress through each node

Team Evaluation

The team believes that we work well together and will have no issues developing and completing this project. As a team, we communicate well and have had no difficulties in contacting each other.

Visibility Plan

External:

Our group will be meeting at least once a week in a designated online Zoom meeting. If problems were to arise we have backup communication on SMS text and email as well. If more complex or strenuous matters were to arise we can also meet in person in the designated class time on Tuesdays at 9:30 A.M. Aside from our local group meetings, we will also be participating in a Client - Team meeting every wednesday to discuss progress in development of the project.

Internal:

The Group will meet weekly, likely Tuesday, anywhere from 9:00 am to 6:00 pm at least 15-25 minutes to discuss progress and problems in development. Our meeting minutes will be recorded and logged for reference in client team meetings. And supplementary communication will likely be done through text message or email. In regards to source code we will communicate on the best way to distribute to the other members depending on the circumstances of the code and development stage in the process. The group will be monitoring our progress on a weekly basis ensuring smooth development throughout the cycle. For maximum efficiency our group will be comparing our ongoing progress to the Gantt chart.

Business Considerations

As Western Kentucky students, the Group owns the copyright in the software that we create in this project. The Group agrees to transfer the copyright to the Client and to provide the Client with unrestricted license to use the system.

(It is a possibility that this project may incidentally form concepts that could be patented. If such a situation arises, the Group collectively owns the rights to all patents associated with the project.)

Risk Analysis

1. Altering Requirements:

Risk: The client may (throughout the developmental process) have a change of heart in regards to different concepts or requirements altering the route of development in regards to the process. The client's obligation to such changes is acknowledged but is to try to be maintained to an absolute minimum as altering the development process can skew results and time frames associated with the project. *Solution:* To minimize changes to initial protocol and requirements by collecting and confirming them clearly before the development process begins. Our group made some software and hardware testing to ensure that our system will work properly for this project.

2. Misinterpreted/Unfulfilled Requirements:

Risk: Within the client's initial requirements, there may be implied or misinterpreted requirements or functions that are not accurately developed to the clients intended specification by the group due to lack of understanding or initial miscommunication. *Solution:* The group can prevent such matters by clearly communicating the interpreted requirements given by the client back to the client themselves to confirm or revise the initial interpretations of such requirements formed by the group.

3. Lack of Knowledge in regards to implementation:

Risk: In regards to the back end development of the project, the complexity of course rotation and entropy of individual schedules amongst the nine degree programs could lead to some shortcomings or inaccuracy in regards to implementation whether a blanket lack of knowledge or specific coding errors. *Solution:* Thoroughly checking our code as we progress and develop the project could help mitigate the vast majority of shortcomings due to incompetence or simple mistakes.

4. System Integration:

Risk: Being that the project is so expansive and certain tasks will have to be distributed to individual group members the project will have to be hosted 24/7, incorporating this effectively within the project could be a concern if not done correctly. Being that the group will be sharing one Raspberry Pi - not conflicting with the other group members' progress while incorporating individual progress may lead to some conflict in compiling and creating complimentary code to the other group members. *Solution:* In order to avoid system integration conflictions, the group needs to clearly understand the raspberry pi’s role and how to combine individual code effectively to create a working and efficient program.

Risk: Being that the project is so expansive and certain tasks will have to be distributed to individual group members the project will have to be hosted 24/7, incorporating this effectively within the project could be a concern if not done correctly. Being that the group will be sharing

one Raspberry Pi - not conflicting with the other group members' progress while incorporating individual progress may lead to some conflict in compiling and creating complimentary code to the other group members. *Solution:* In order to avoid system integration conflictions, the group needs to clearly understand the raspberry pi's role and how to combine individual code effectively to create a working and efficient program.

5. Non-functional Requirements:

Risk: Similar to "*Lack of Knowledge in regards to implementation"* possible risk , the non-functional requirements of the project may cause a few issues to arise in regards to lack of knowledge in unfamiliar concepts and languages specifically the use of JavaScript, D3.js packages, and Dockers. Even though The group has been learning through sprint 3 how to use the programming languages and frameworks needed there are still some errors in our implementation. *Solution:* Our team will try to look for different ways to implement our code and frameworks whenever any problem appears. One of our team members shared a playlist to learn about the front-end implementation of a web-page.

Human resources:

Risk: Being that several of the requirements (especially the non-functional ones) are somewhat foreighn to all members of the group the development process, especially in the early stages of the project could be slow to start.

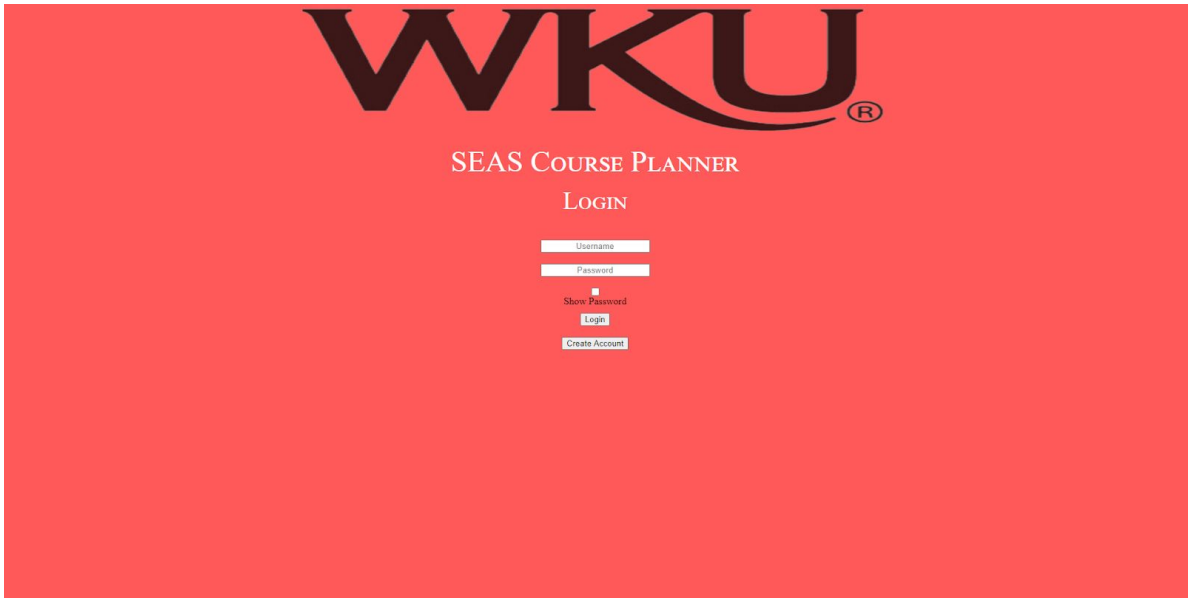
Solution: The group has acknowledged our shortcomings in regards to the knowledge and skills to develop the project in an efficient and accurate way and will take steps to verify we understand these skills clearly and their role in creating the project and maintaining the project.

Conclusion:

Based on thorough review of the project's timeline and required functions both functional and non-functional, we've concluded that the project can be completed in an efficient and accurate manner while meeting clients specification in both the time constraints of the project and technical specifications.

Wireframe Diagrams

Login Webpage

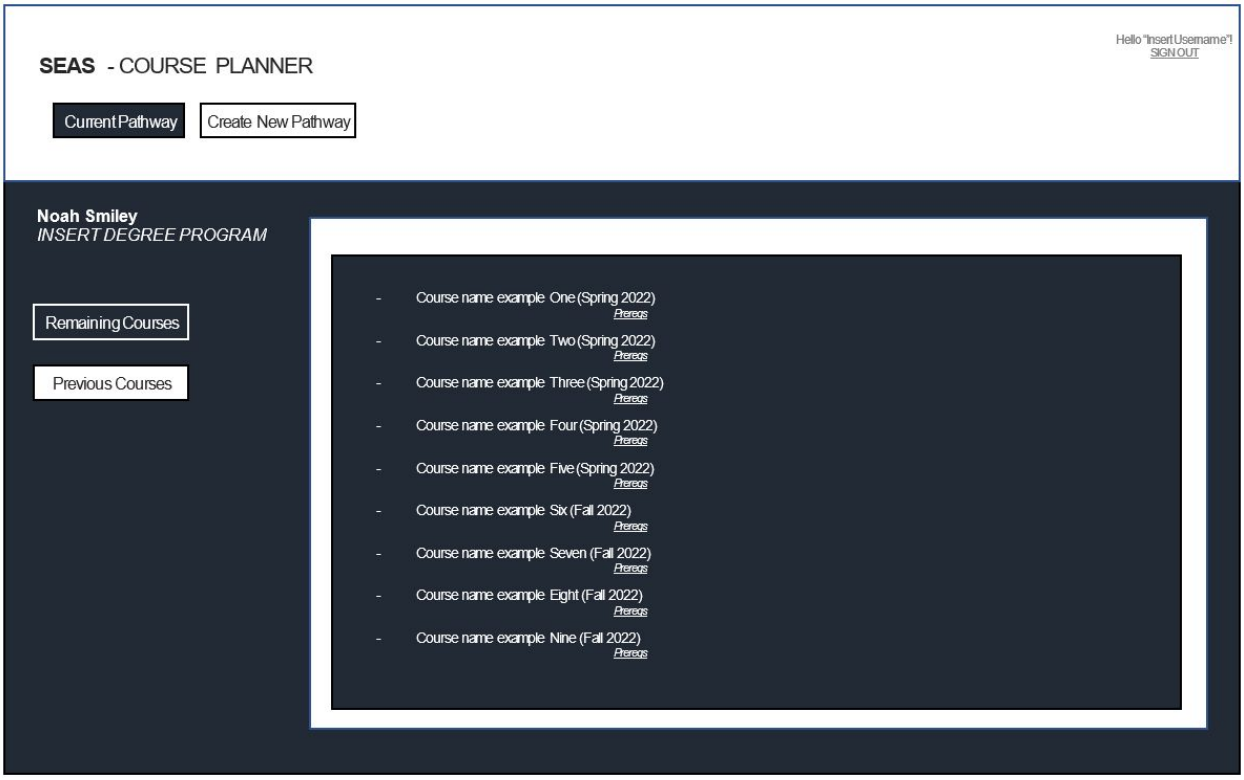


Description: This is the login page where students and professors will be asked to enter their username and password. If there is any user that is not able to login, they can click on the "Create Account" button which will redirect them to the registration page

Registration Page

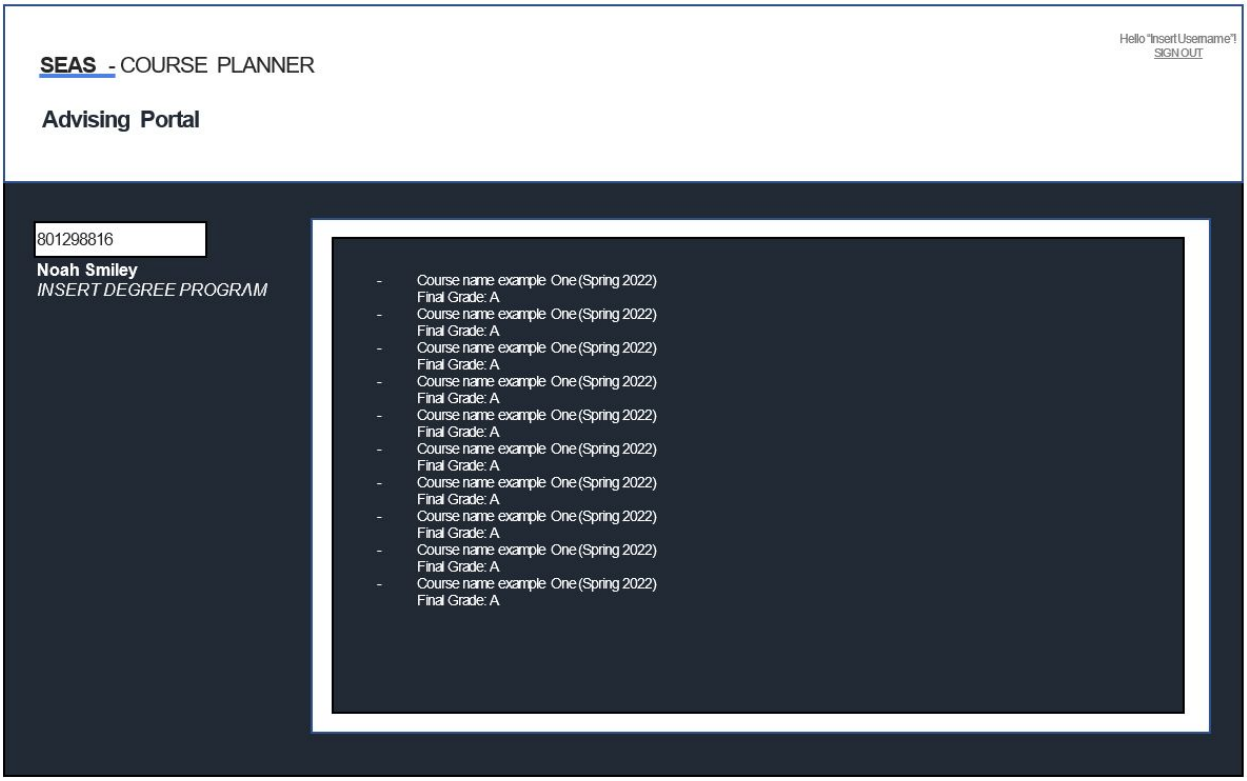
Description: The registration page will ask each user to select its account type (student or admin) and enter a username and password. Once a new user is registered it will be added to our database and will be able to login to their account.

Student home-page



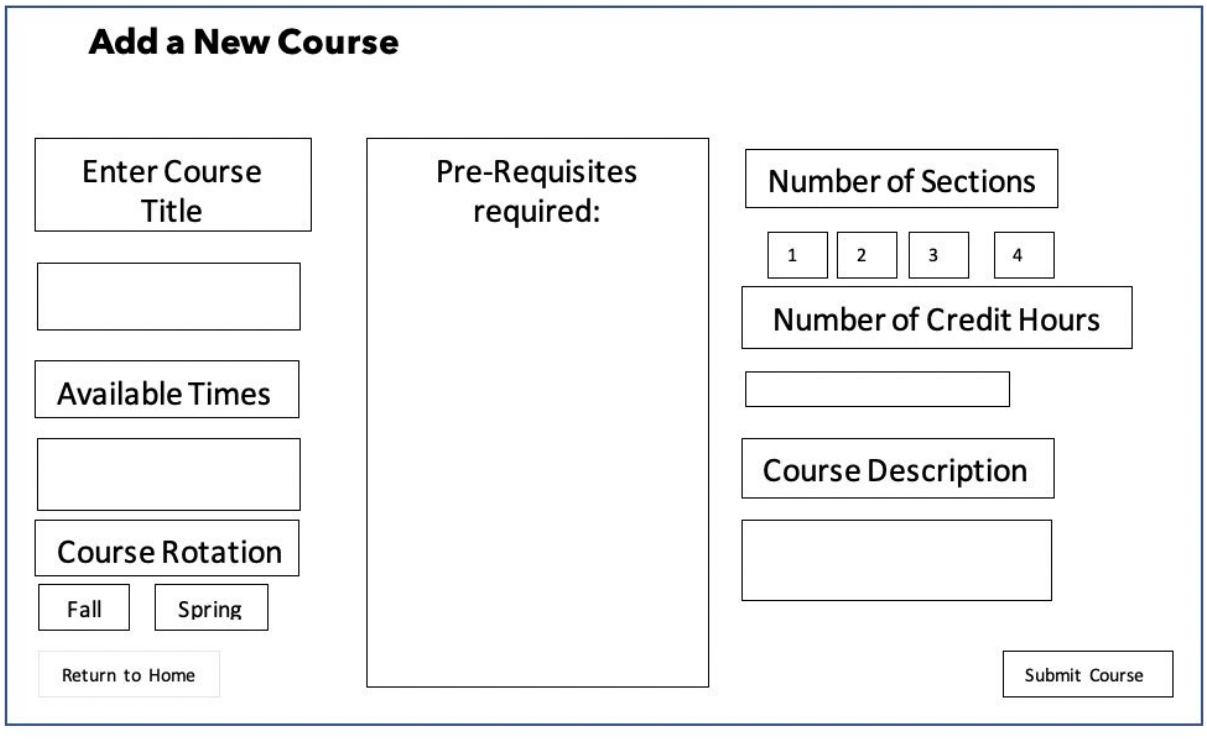
Description: This home page will allow each student to quickly access their current pathway to complete their academic degree which will be displayed in the middle of the screen.. Student information and courses records will be shown at the left side of the screen. Additional options such as creating a new pathway or getting an in depth look at the student's current pathway will be located at the top left side of the screen.

Administrator home-page



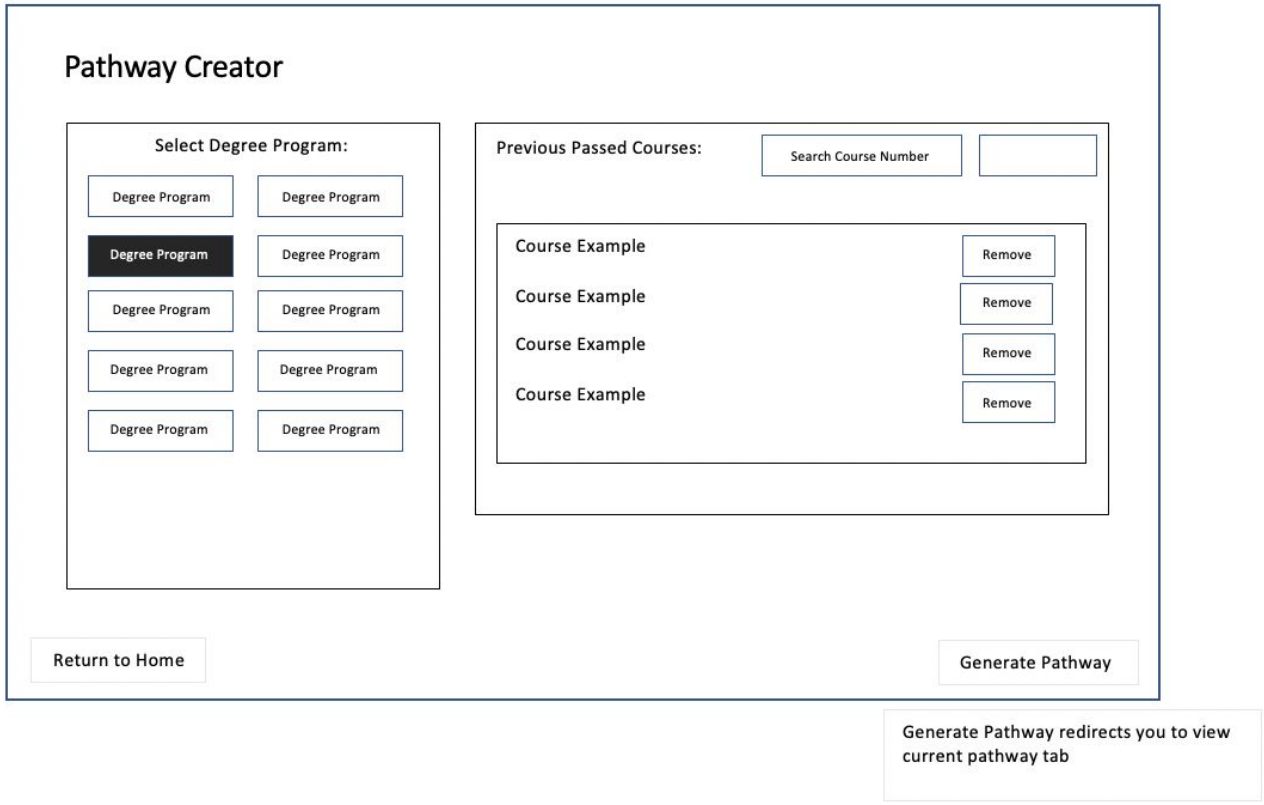
Description: Administrators will have access to a home page where they can get details about student grades and enrollment. The purpose of this portal is to make it easier for advisors to help in the registration and planning process of each student.

Administrative Add a New Course:



Description: Administrators will be able to add a new course to the database. There will be certain requirements for it such as the course title, prerequisites, course rotation, etc. Each course can be modified by any administrator and the system will automatically notify students about the changes.

Create Pathway:



Description: Students and administrators can create different paths to complete a degree. The pathway creator will show the required course to complete the degree for each major.

Use Case Scenarios

We designed different use case scenarios to test all the functionalities of our program and at the same time to consider the preconditions to complete before accomplishing another task. These use case scenarios propose examples where users try to perform different tasks using our program. All of them will be tested while we develop our application to ensure that the functionalities provided by our code satisfies the necessities of future users

Use Case: New Administrator User
ID: UC001
Actor: Administrator
Preconditions: User is in admin page
Primary Scenario: Administrator clicks on “sign up” System asks for administrator credentials Administrator credentials are checked by our data base Administrator is not accepted since it is not part of our data base
Secondary Scenarios: Administrator is part of our database

Use Case: Access Student Information
ID: UC002
Actor: Administrator
Preconditions: User is logged in At least one student is enrolled in one of the classes given by the administrator or the administrator is an advisor
Primary Scenario: Begins when the administrator selects “enrolled students” The system displays Names, IDs, GPAs, and Major for each student
Secondary Scenarios: Student does not have any GPA information (Freshman)

Use Case: Access Login
ID: UC003

Actor: Administrator/ Student
Preconditions: User has an ID, a password User is part of SEAS
Primary Scenario: The system asks for user credentials The user inputs its credentials The system checks user privileges (student or administrator)
Secondary Scenarios: Invalid user credentials

Use Case: User Registration
ID: UC004
Actor: Student
Preconditions: Student is in Login page
Primary Scenario: The system ask for user credentials Student clicks on “Create Account” button The system redirects student to the registration webpage Student selects the account type “student” Then the student creates a name and a password Student clicks on create account button
Secondary Scenarios: Student name is already in use Student selects incorrect account type

Use Case: Get Course and Schedule suggestions
ID: UC005
Actor: Student
Preconditions: Student is logged in Student has a declared major

Primary Scenario: The system gets major requirements for student's major The system evaluates the requirements for each course and student credits The system displays a list with the suggested courses student should take The system evaluates the best path for a student to complete all the major requirements The system displays a suggested schedule based on the path previously evaluated
Secondary Scenarios: Student has no major declared Major requirements could change Elective classes could affect student priorities

Use Case: Modify Course Attributes
ID: UC006
Actor: Administrator
Preconditions: Students are registered in classes and administrator is in main administrators page
Primary Scenario: The system displays different courses with options to add new ones The administrator clicks on a computer science course The administrator changes the name of the course to software architecture The administrator clicks the save button and the system saves the changes
Secondary Scenarios: The administrator does not save the modification they have done. The administrator creates a new course

Use Case: Log out
ID: UC007
Actor: Student
Preconditions: Student is already in the login page
Primary Scenario: Student logs in Student selects Current Pathway System displays Pathway

Student selects ‘logout’ link at the top of the page
Secondary Scenarios: Student makes changes for pathway or schedule Student press exit button while changing major or making changes to schedule

Requirements Traceability Table

The requirement traceability table compares the preliminary requirement analysis that we made on page 2 to our use cases. The purpose of this table is to ensure that all our cases test the requirements for the project and clearly projects how our program is expected to be used.

Use Cases								
Requirements		UC 001	UC 002	UC 003	UC 004	UC 005	UC 006	UC 007
	R1					X	X	
	R2	X	X		X	X		X
	R3	X		X		X		X
	R4		X		X	X	X	

OO Design Patterns

Creational

The team will utilize the Factory Method design pattern in the system. This pattern requires the use of a parent class or interface that is then implemented or extended to more subclasses. This design pattern will be used because this system requires two different kinds of users, but rather than define two completely different classes, the team can create one general User class and then create two subclasses for Students and Administrators.

Source Code:

```
from django.apps import AppConfig

class UsersConfig(AppConfig):
    name = 'Users'
-------------------
from django import forms
```

```
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm

class UserRegisterForm(UserCreationForm):
        pass
class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']
---------------
from django.contrib import admin
from django.contrib.auth import views as auth_views
from django.urls import path, include
from Users import views as users_views

urlpatterns = [
        path('admin/', admin.site.urls),
   path('', include('SEAS_Course_Planner.urls')),
   path('register/', users_views.register, name= 'register'),
path('login/', auth_views.LoginView.as_view(template_name='users/login.html'), name=
'login'),
path('logout/',auth_views.LogoutView.as_view(template_name='users/logout.html'),
name= 'logout'),
   path('newpathway/', include('SEAS_Course_Planner.urls')),

]
```

Behavioral

The team will utilize the Iterator design pattern in the system. This pattern creates an easy way to iterate through and search a collection without the search being visible. This design pattern will be used because although we are using the Flyweight pattern, we have to include a way to iterate through the Users in order to determine if the current User is attempting to duplicate an existing account.

```
Source Code:{% extends "SEAS_Course_Planner/base.html" %}
{% load crispy_forms_tags %}

{% block content %}
   <div class="content-section">
```

```
        <form method="POST">
            {% csrf_token %}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Log In</legend>
                {{ form|crispy }}
            </fieldset>
            <div class="form-group">
                <button class="btn btn-outline-info" type="submit">Login</button>
            </div>
        </form>
        <div class="border-top pt-3">
            <small class="text-muted">
                Need An Account? <a class="ml-2" href="{% url 'register' %}">Sign Up Now</a>
            </small>
        </div>
    </div>
{% endblock content %}
----------
from django.shortcuts import render, redirect
from django.contrib import messages
from .forms import UserRegisterForm
# Create your views here.

def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            messages.success(request, f'Your account has been created! You are now able to log in')
            return redirect('login')
    else:
        form = UserRegisterForm()
    return render(request, 'users/register.html' , {'form': form})

def profile(request):
    return render(request, 'users/profile.html')
```

Structural

The team will utilize the Flyweight design pattern in the system. This pattern decreases the amount of memory required when a large amount of objects are created that share some internal states. This design pattern will be used because it creates a pool of objects that can be used; if a User tries to create an account that already exists, then the flyweight pattern will allow the system to supply the User object that has already been created.

Source Code:

```
from django.shortcuts import render, redirect
from django.contrib import messages
from .forms import UserRegisterForm
# Create your views here.

def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            messages.success(request, f'Your account has been created! You are now able to log in')
            return redirect('login')
    else:
        form = UserRegisterForm()
    return render(request, 'users/register.html' , {'form': form})


def profile(request):
    return render(request, 'users/profile.html')
```

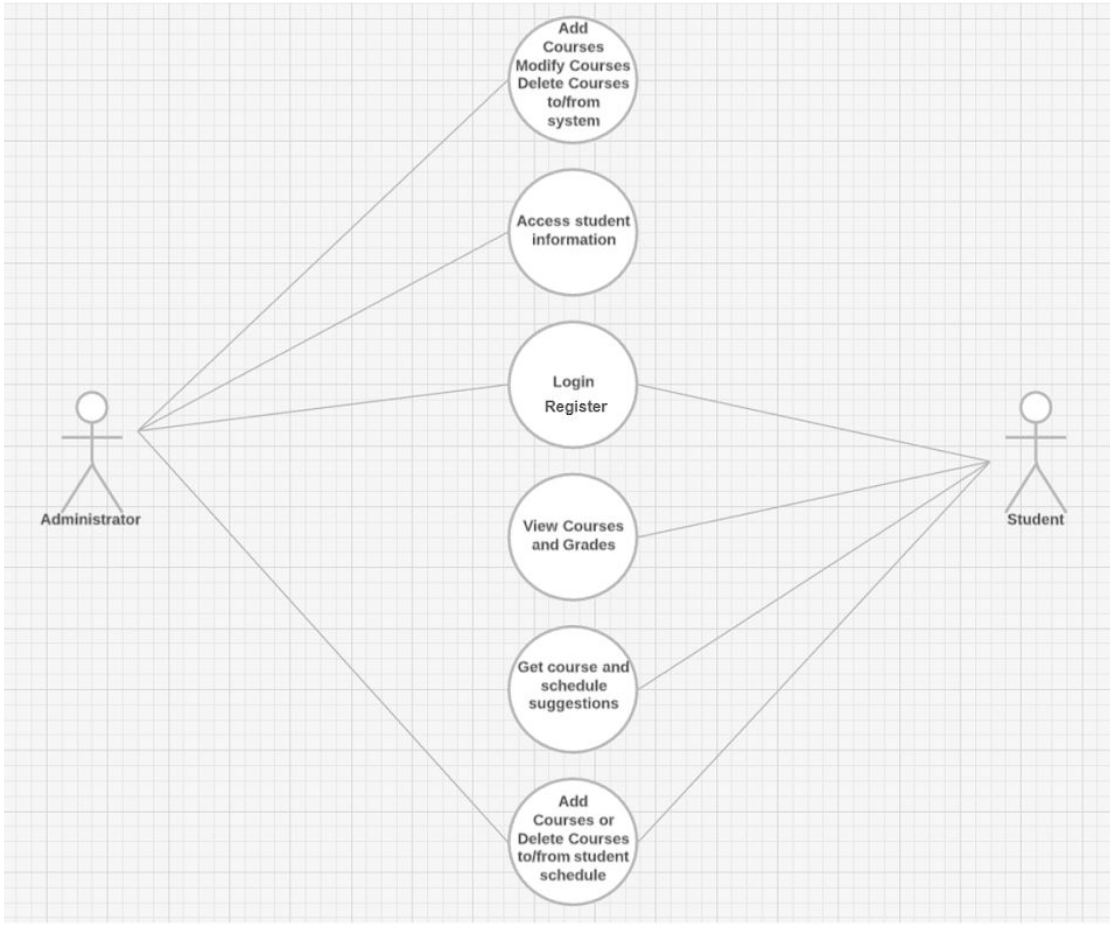
Use of macro and micro software control

Within our project, there's a base HTML file which houses the navbar, and basic structure of the websites. The base HTML file is then referenced and built upon by the other pages, so it doesn't have to be coded everytime - by developing this way, we can have control on each page and makes debugging the site much easier as the files are much smaller in length as they’re built upon the base file.

Use Case Diagram

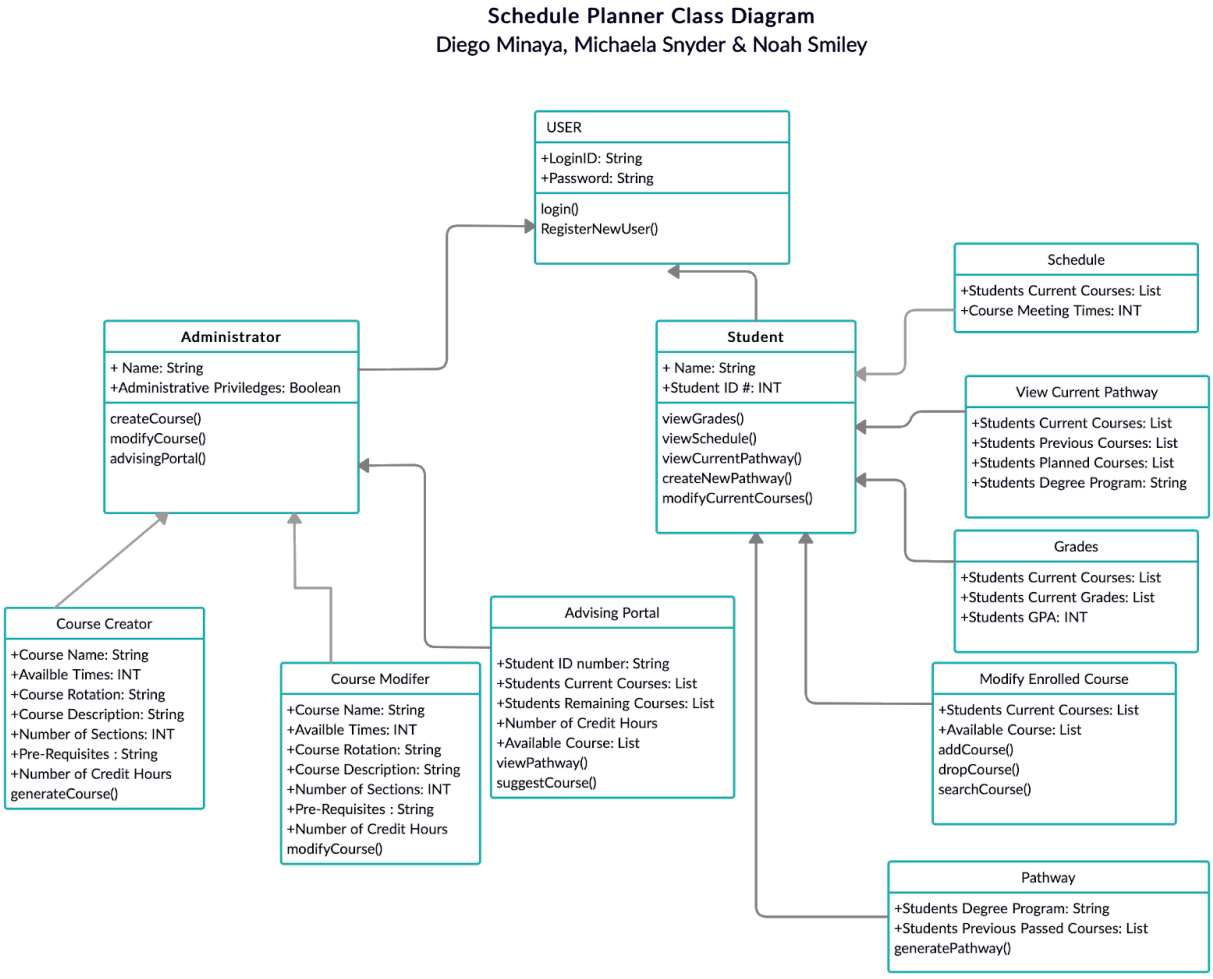
Schedule Planner Use Case Diagram

Diego Minaya, Michaela Snyder & Noah Smiley



This case diagram shows the functionalities that administrators and students will be able to access and what kind of information would be displayed to them. It also shows the difference between both classes, allowing administrators to have more control over our system.

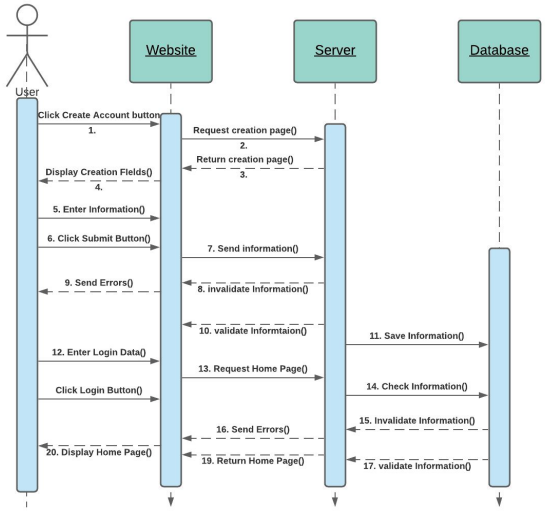
Class Diagrams



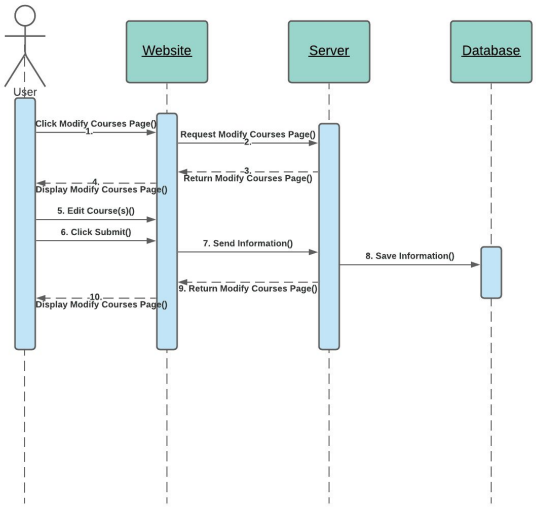
This Class Diagram provides the attributes and methods needed for all the classes that will be used to develop our code. Each object in our diagram has a name , attributes and methods (some objects would not need methods). They are ordered from top to bottom and the edges that connect our objects show the relationship between different objects

Sequence Diagrams

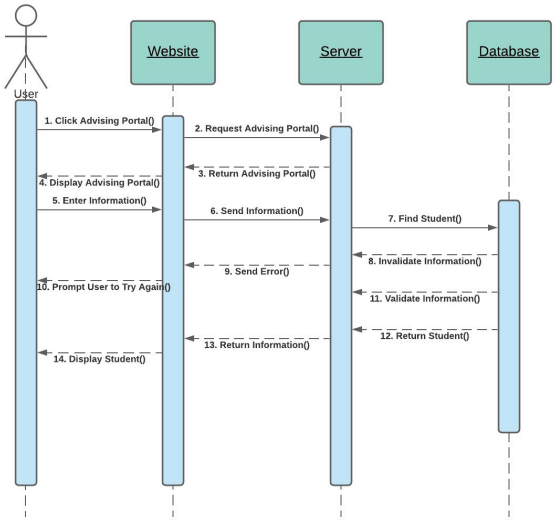
User Login Sequence



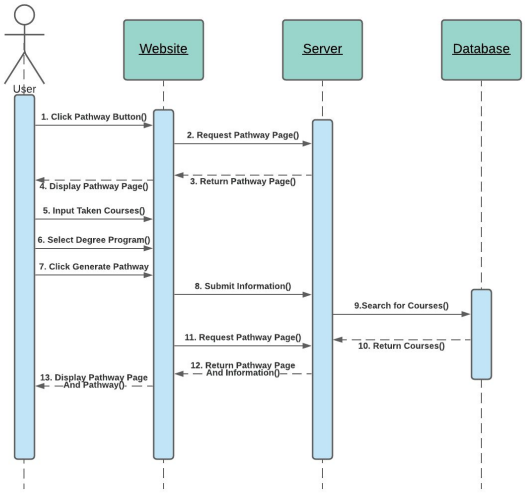
Modify/Edit Course Sequence



Find Student Sequence

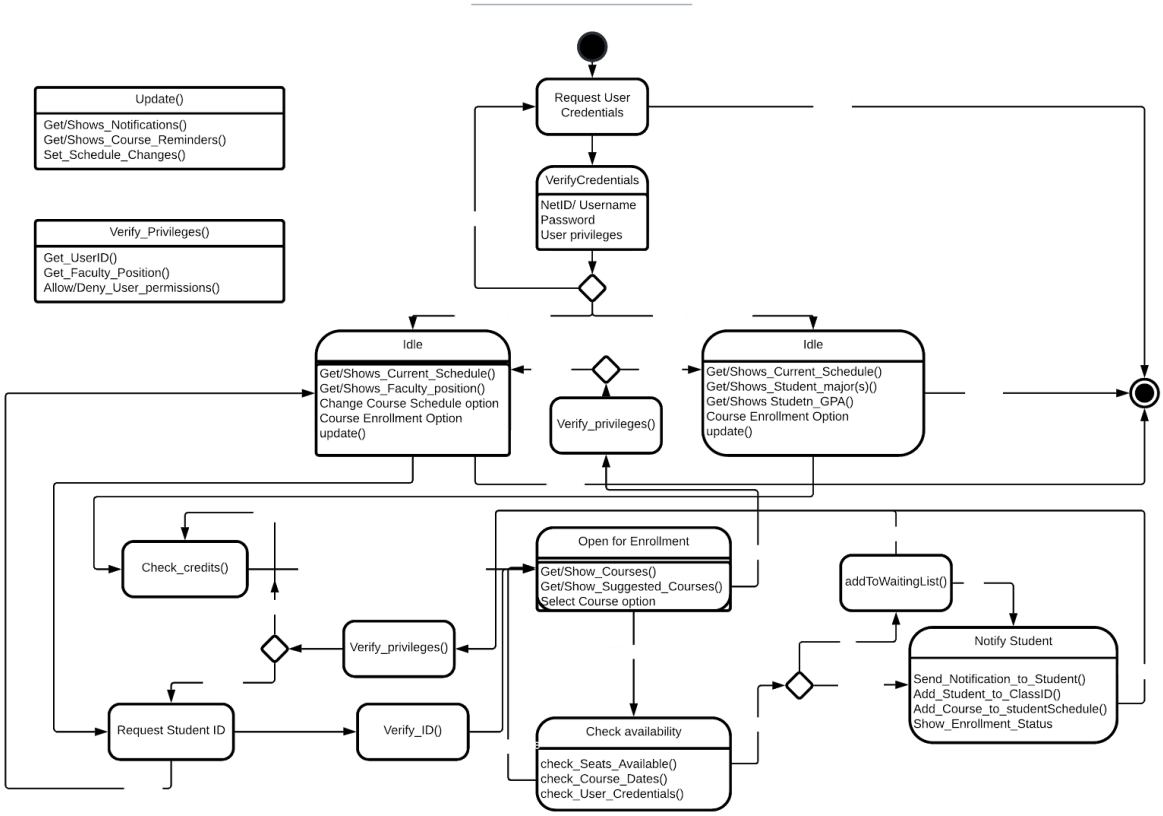


Generate Schedule Sequence



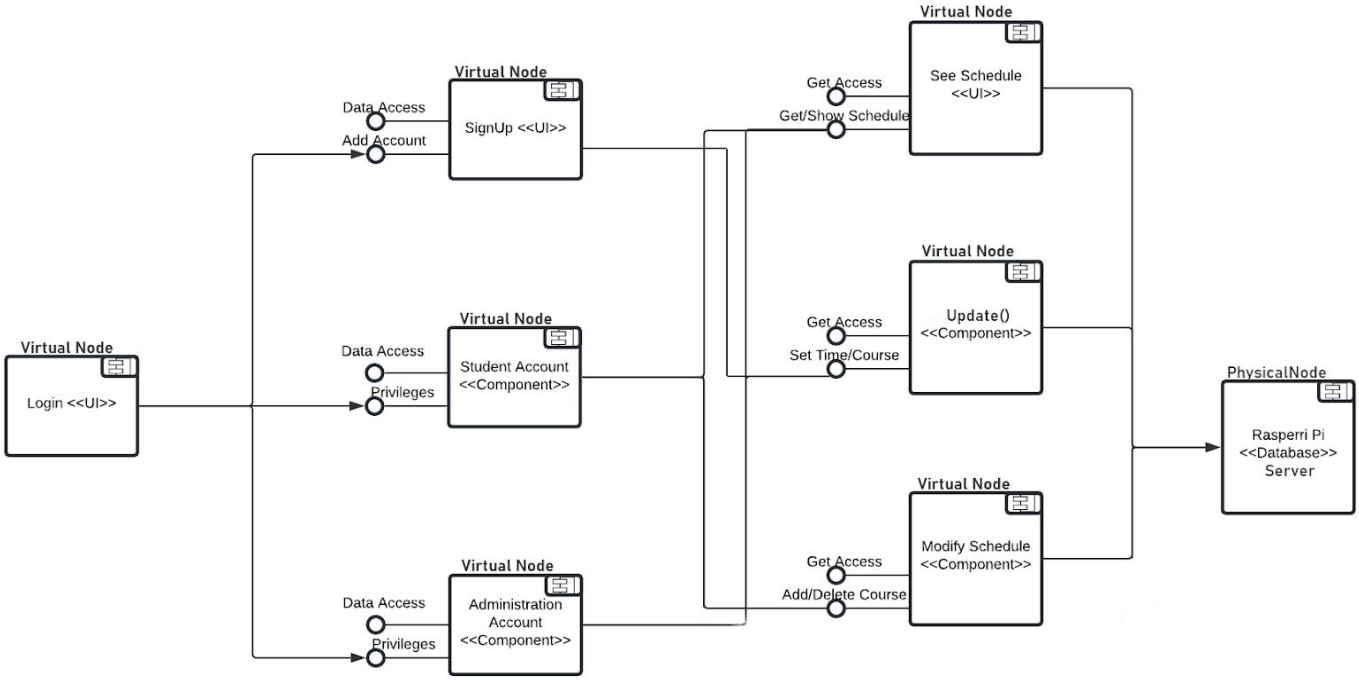
In each Sequence Diagram, the relationship is defined between the User, the Website, the Server, and the Database. Although there are five Use Case Scenarios, there are only four distinct functional relationships: the User Login, Modifying/Editing Courses, Finding a Student, and Generating a Student.

State Diagram



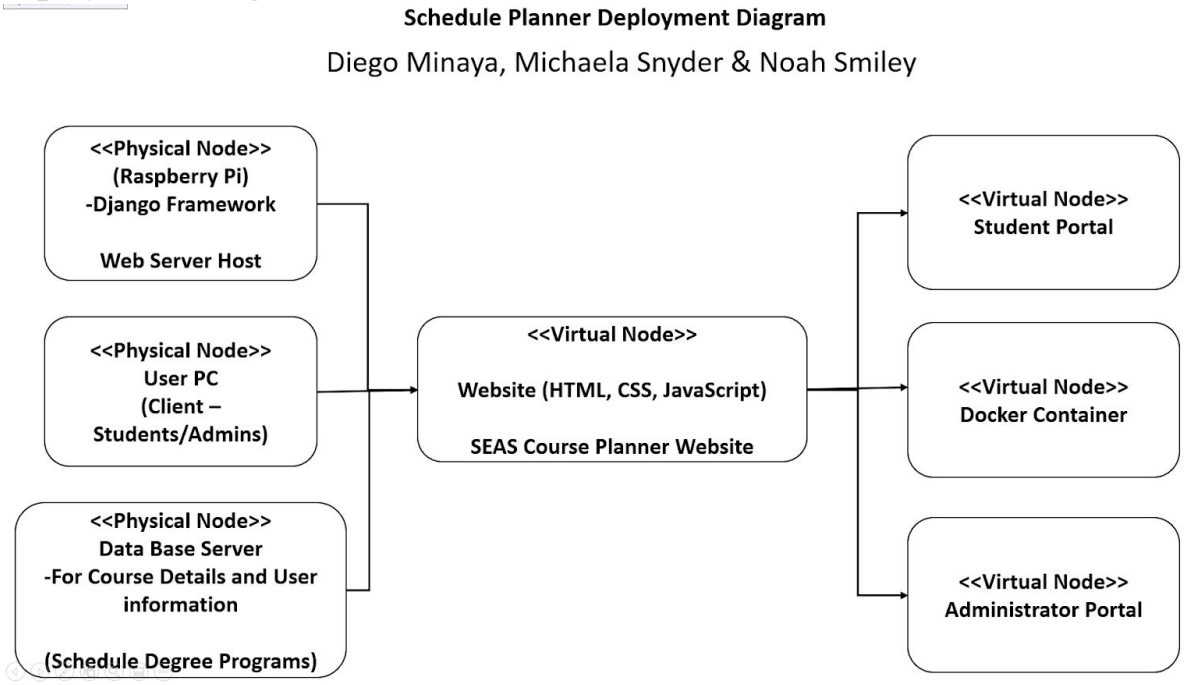
This state diagram describes our system by using different states that are mostly connected to each other. Each of them shows important attributes and operations. However, there are 2 abstract functions for our system to work properly (Update() and Verify_Privileges()) both of them are planned to be used for student and administrator classes.

Component Diagram



Our component diagram shows virtual and physical nodes across our system. Since our system will start from the login page, this page will be the starting node that will pass user data to all the other nodes. This user data will be checked or stored depending on the functionalities for each node. At the end of this diagram is our server, where all these nodes are going to be executed.

Deployment Diagram



The deployment diagram shows the physical and virtual nodes of our system but with a special emphasis on the frameworks and programming languages that each one of them will have. There will be 3 physical nodes (server, client and database) that work together to show our Course Planner website. This website is a virtual node connected to other virtual nodes based on the resources needed for the user task.

Data Dictionary

The following chart shows a collection of the variables that are used and are planned to be used for our code. This data dictionary provides a better transition of data between our different codes and a better understanding on the purpose of each variable.

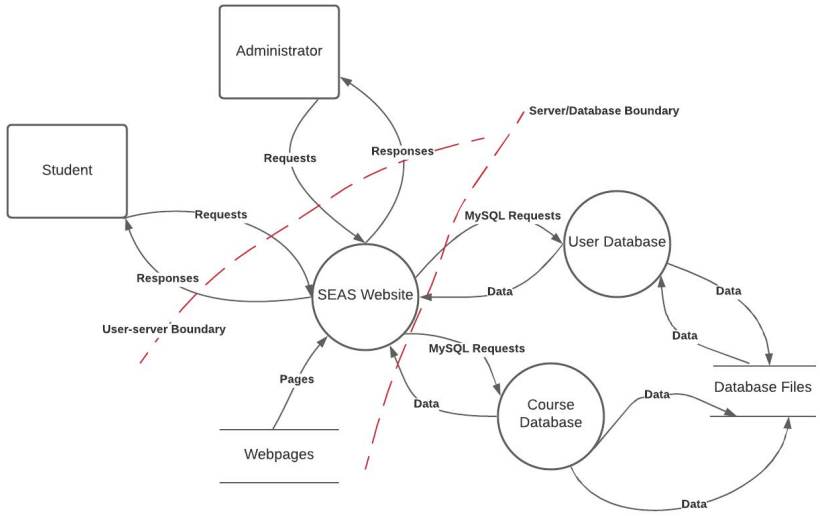
Data Dictionary

Element or value display name	Description	Data type	Location	Acceptable values	Required?	Accepts null value?	Read only?	Developed and integrated?	Notes
LoginID	ID of student or administrator	Int	Login Page and database	800#####	Yes	No	No	Yes	
Password	Combination of characters and numbers	String	Login Page and database	At least 1 character and 1 int	Yes	No	No	Yes	Can be created through registers page
Name	Name of the administrator	String	Administrator class	At least 2 characters	Yes	No	No	Yes	
Name	Name of the student	String	Student class	At least 2 characters	Yes	No	No	Yes	
administrative_Priviledges	Checks if administrative is authorized	Boolean	Administrator class	True or False	Yes	No	Yes	No	
courseN	Name of course	String	Database	Name###	Yes	No	Yes	No	Only administrators can modify it
GPA	Students Grade	Float	Database	#.##	Yes	Yes	Yes	No	It will display null if the student has no records
currentCourses	Array of courses	Array	Database	Course Obj	No	Yes	Yes	No	
courseD	Description of the course	String	Database		Yes	No	No	No	
courseR	Requirements for course	String	Database	Course Obj	Yes	No	No	No	

courseID	Number of the course	Int	Database	###	Yes	No	No	No	
pathwayArr	Array of courses	String Array	Database, main page	Course Names/IDs	No	Yes	No	No	It is null if no pathway was created
degreeProgram	Name of degree	String	Database, main page		Yes	No	Yes	No	
previousPassedCourses	List of passed courses	Array of strings	Database	List of different course nems	Yes	No	Yes	No	The array can be empty but can not be null

Security Issues and Resolution

Security issues that the team encountered had mainly to do with administrator access, hiding passwords during login and registration, and making sure there was a minimum length for passwords as well as guidelines for a strong password. These issues were addressed in the design of the system. Passwords are hidden unless a "Show Password" button is selected, which ensures that passwords are hidden unless users want them shown. Similarly, during registration, users are required to create passwords that are at least 8 characters long and are alpha-numeric.



This security model showcases the relationship between all available parties in the system such as the student and administrator users, the website, the webpages, the user database, the course database, and the database files. There is a boundary between the users and the server and the server and the database.

Software System Performance

Performance Requirements

Since our server needs to store the data of different users and at the same time modify that data according to administrator's and student's request, it is necessary to ensure that our server is capable of processing different threads at the same time. The storage capabilities of our system must be able to handle changing and accessing its data constantly. Therefore we require that our system reads and writes data persistently. Finally, to ensure that our users receive instant feedback on any changes or any request they made, it is important to make sure that transferring data from our server across the web does not take too long. For that reason, it is required to ensure that the WAN connection works relatively fast.

Performance Objectives

Test processing, storage, and connection capabilities of our server to ensure that it will work properly when our web SEAS course plan service is deployed. We will run different tests in our system using Sysbench and Iperf

Test-Case Results

In our early build, we successfully completed our use case 3 and 4. We are still working on the framework and developing our database, therefore we can not test other cases until we finish entering our data for different courses and users. However simple information such as the ID of the user or password are working properly.

System/Application Workload

We test our system by running from 1 to 64 threads, the system successfully completed all the tasks in 456.55 seconds on our server and 4201.1 seconds in a container. We also ran a storage performance test by creating a file from 64 MB up to 2GB and our system was able to allocate all that information in 154.81 seconds. Finally, we ran a network test using LAN and WAN connections inside a container and in our system. We transferred from 1 MB to 64 MB. It took a total of 52.22 seconds to ran our tests on LAN and a total of 56.31 seconds to run our tests on WAN.

Hardware/Software Bottlenecks

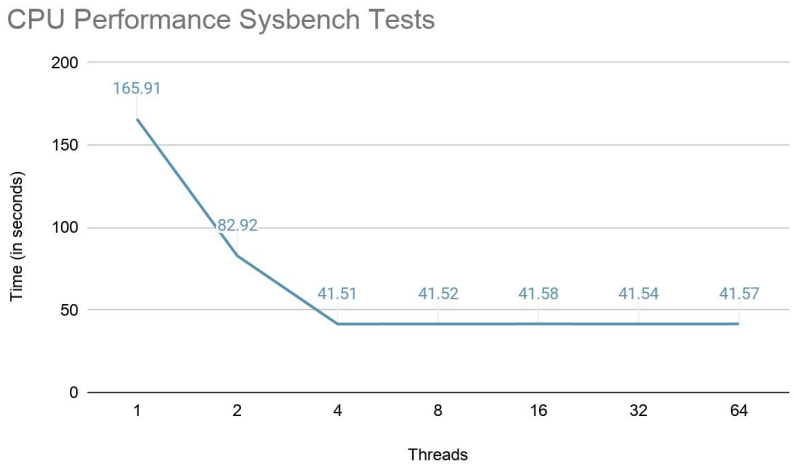
Hardware

On average, the Raspberry Pi is working at 11% capacity, without running the container. With the container running, the Raspberry Pi averages out at 34% capacity. As shown in the Sysbench tests, although CPU usage for the Raspberry Pi is manageable, the Docker container may run into performance issues if there is a surplus of users on the system.

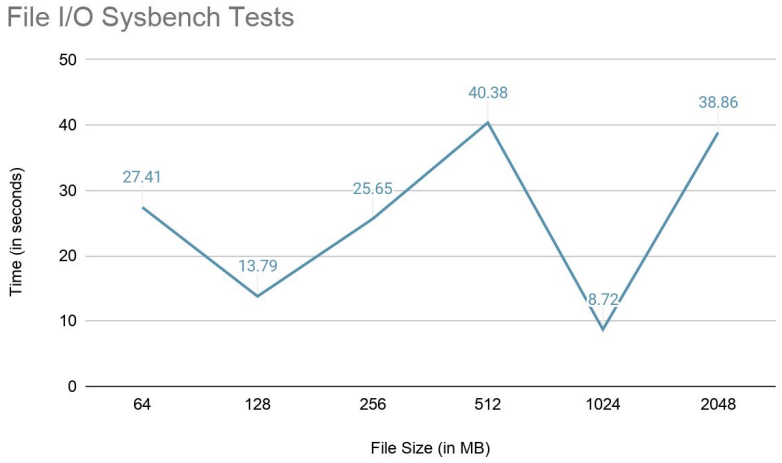
Software

Code may end up being inefficient, although extensive tests have not been done on existing code due to it still being in development.

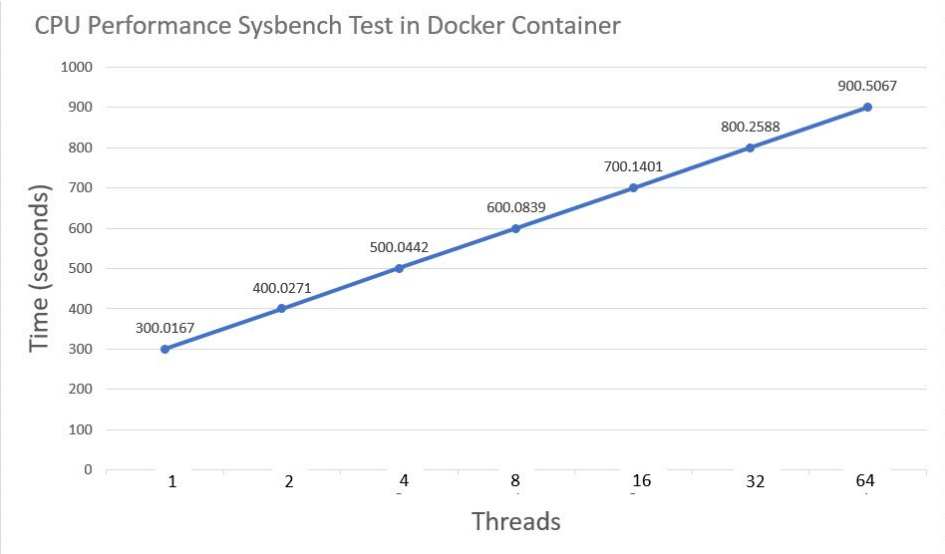
Sysbench Data



The basic test-case for the Raspberry Pi system performance entailed running a short terminal command through Sysbench that found all prime numbers up to 10,000. The numbers on the x-axis are the number of worker threads.

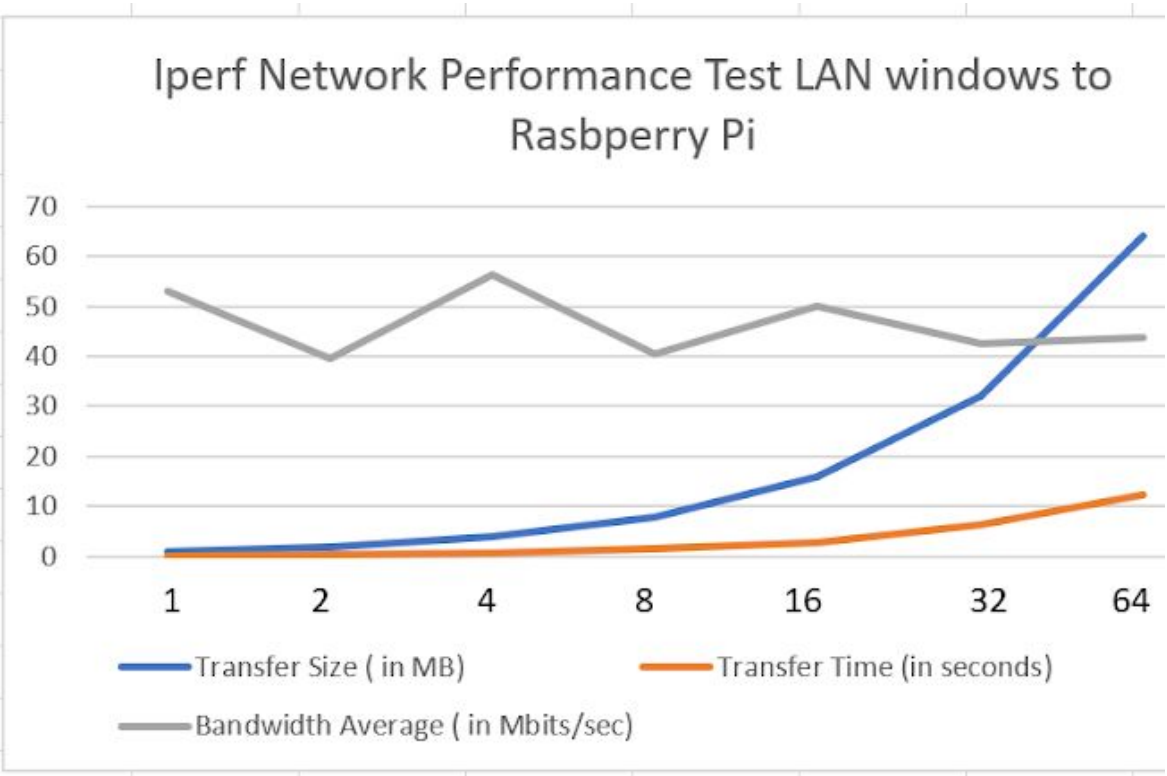


The file I/O Sysbench tests entailed creating files of ascending size from 64MB to 2GB and using a Sysbench terminal command to randomly read and write in the files. At 512MB and 2GB, there were sudden decreases in system performance.

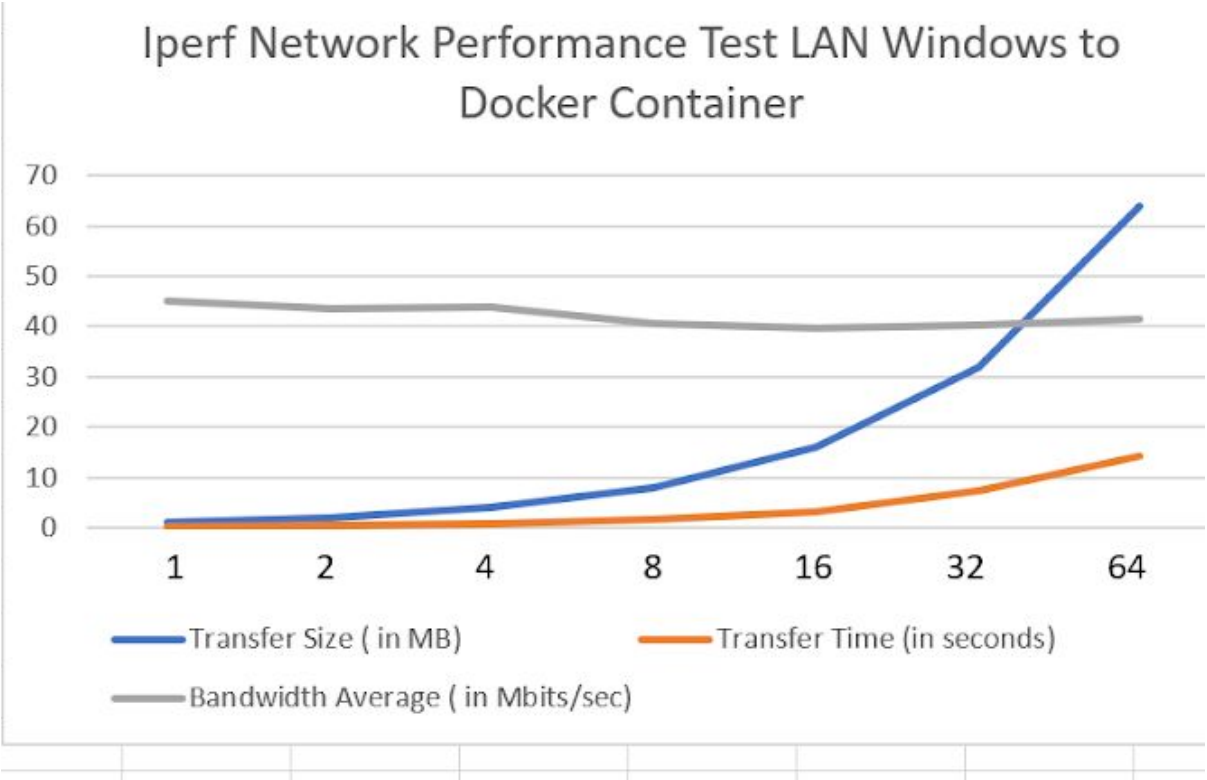


The CPU performance Sysbench test was deployed in a docker container to check the execution of different threads inside a container and to compare its performance to our previous CPU sysbench test deployed in our server.

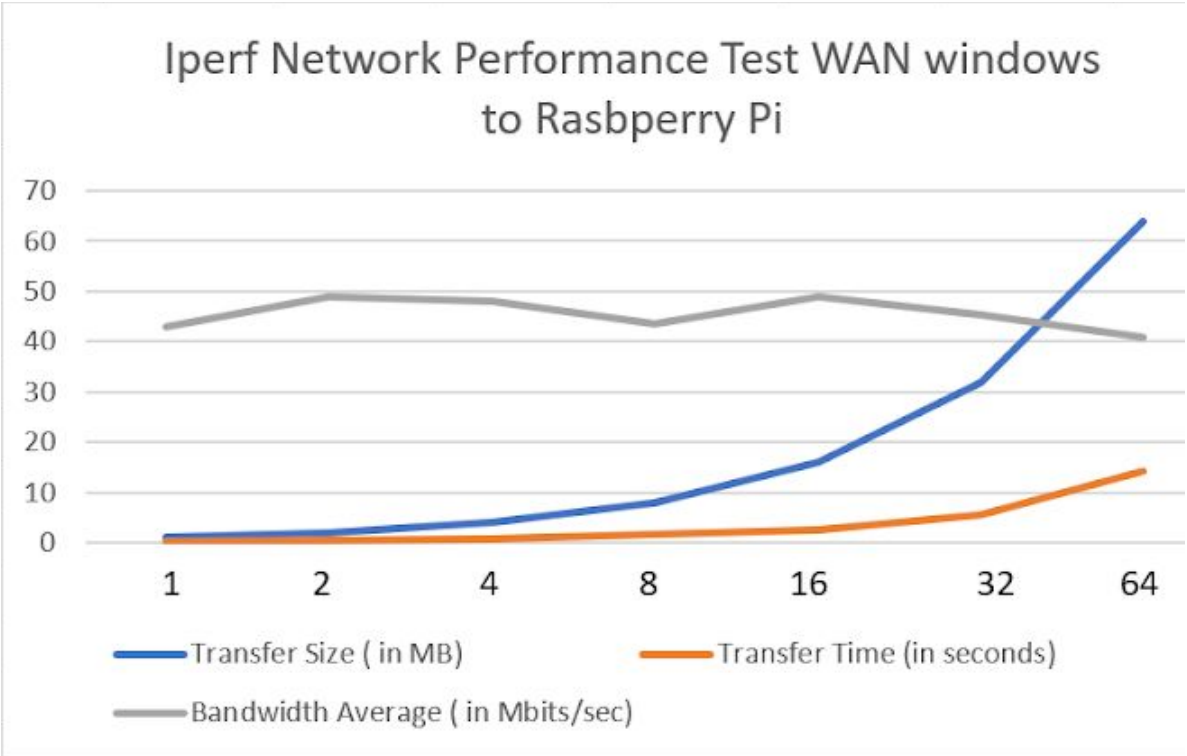
Iperf3 Tests



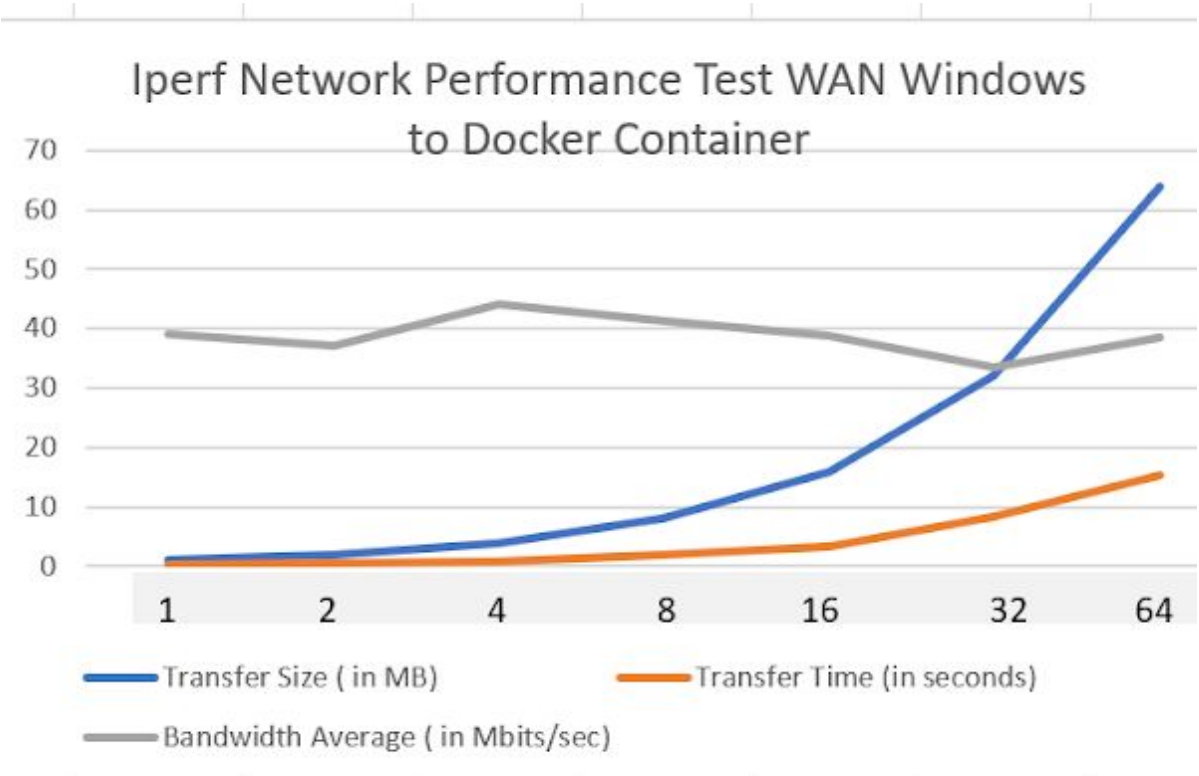
The iperf network test allows us to compare the transfer time that is necessary depending on how big is the file size that need to be transferred, this test was made using LAN connection and due to an error on iperf (iperf3:error - select failed: Bad file descriptor) the maximum data that we could transfer was 64 MB.



The iperf network test allows us to compare the transfer time that is necessary depending on how big is the file size that will be transferred, this test was made using a container in our server and a pc connected to the same LAN. Due to an error on iperf (iperf3:error - select failed: Bad file descriptor) the maximum data that we could transfer was 64 MB.



This iperf network test allows us to compare the previous LAN connection performance with our WAN connection, this test was made using WAN connection between a pc and our server. Due to an error on iperf (iperf3:error - select failed: Bad file descriptor) the maximum data that we could transfer was 64 MB.



This iperf network test allows us to compare the previous LAN connection performance with our WAN connection, this test was made using WAN connection between a pc and a container running in our server. Due to an error on iperf (iperf3:error - select failed: Bad file descriptor) the maximum data that we could transfer was 64 MB.

Performance Monitor Development

All of group members made the following collaborations:

Michaela Snyder: Security Issues, Software System performance, preparing Slides, developing HTML, CSS code for login webpage, updating documentation

Noah Smiley:Action Plan, developing register, user and administrator web pages, preparing our framework using DJango and deploying it in our server

Diego Minaya: Preparing Server (port forwarding), Testing Network Performance, updating diagrams and its documentation, data dictionary

Time-Based performance Data and Description

Generally speaking, the time-based performance tests were the Sysbench tests that measured the time to copy and/or transfer the data from generated files to generated empty files.

The average for the CPU performance tests on the Raspberry Pi was 59.29s over different thread counts from 1 to 64.

The average for the File I/O performance tests on the Raspberry Pi was 25.8s for files of differing sizes from 64MB to 2GB.

The data shows that the Raspberry Pi is able to run time-sensitive programs in an effective manner, which is important for our project considering users expect applications to run smoothly and efficiently.

Rate-Based performance Data and Description

The following rate based performance test were done in our server (Raspberry pi):

Threads test :

total time: 21.2052s

avg: 8.48ms

RAM Memory test.

3072.00 MB transferred

1224.77 MB/sec

SD write test:

537 MB written

21.3 MB/s

SD read test;

537 MB readed

45.5 MB/s

The data shows that the raspberry pi is able to run a multithreaded program in a timely way. This multi thread test is a simulation of the number of clients that will be connected to the server . For the memory and data tests, our raspberry pi is able to process, write and load a good amount of data that is more than enough for the data that is expected to be processed for this project in the schedule planner itself and in the database.

Software Testing

Software Testing Checklist

To test the use cases we created for this project, we made a software checking checklist where we go through each one of the necessary steps for the use cases and see if our system is able to provide us with the expected output. 63% of the steps from the use case scenarios were successful (show the expected output).

Software Testing Checklist

Software Name:	Interactive SEAS Course Program Planner

ID	Test Case	Input Value/Action	Expected Output	Pass/ Fail	Date Tested (mm/dd/yyyy)
	Edit Course				
1	Preconditions meet before testing use case:	Not applicable	Administrator page shown	P	12/10/2020
2	Administrator creates a new account	Clicks on “sign up“ button	Sign up page for administrators is displayed	F	12/10/2020
3	System asks for administrator credentials	Username: Diego Password: SEASCP12345	System rejects administrator registration since it is a student	F	12/10/2020

	Secondary Scenario				
4	Administrator inputs right credentials	Username: Noah Password: ****	New account is created	F	12/09/2020
	Access Student Information				
5	Preconditions meet before testing use case	Not applicable	Administrator is logged in	F	12/09/2020
6	Administrator selects “enrolled students”	Clicks on enrolled student button	Shows the information of enrolled students	F	12/09/2020
	Access Login				
7	Preconditions meet before testing use case	Not applicable	Database has info about user	P	12/09/2020
8	System asks for user credentials	Not applicable	Web Page asks for username and password	P	12/09/2020
9	User inputs its credentials	Username: Diego Password: SEASCP12345	Shows user input	P	12/09/2020
10	The system checks user privileges (student or administrator)	Not applicable	User has access to the Course	F	12/09/2020

			Program Planner webpage		
	Secondary Scenario				
11	Invalid user credentials	Username: ABC Password: 12345	User input is deleted	P	12/09/2020
	User Registration				
12	Preconditions meet before testing use case	Not applicable	Shows Login Page	P	12/09/2020
13	System ask for user credentials	Not applicable	Input is required from user	P	12/09/2020
14	Student clicks on “Register” link	Clicks on “Register” or “Sign Up Now”	System redirects student to the registration webpage	P	12/09/2020
15	Student selects the account type “student”	Selects Student from list	Shows student account type	F	12/09/2020
16	Student creates a name and a password	Username: Usertester Password: SEASCP54321	Input for name and password is required	P	12/09/2020
17	Student clicks on create account button	Clicks on “create account”	Account is created	P	12/09/2020

	Secondary Scenarios				
18	Student name is already in use	Username: Diego Password: SEASCP12345	Web page ask for another name	P	12/09/2020
19	Student selects incorrect account type	Selects Admin from list	System checks user credentials	F	12/09/2020
	Get Course and Schedule suggestions				
20	Preconditions meet before testing use case	Username: Diego Password: SEASCP12345	Student is in main page and has a stored major	P	12/09/2020
21	System gets major requirements for student's major	Not applicable	System evaluates the requirements for each course and student credits	P	12/09/2020
22	System evaluates the best path for a student	Not applicable	System displays a list with the suggested courses	F	12/09/2020
	Secondary Scenarios				

23	Student has no major declared	Not applicable	No path should be evaluated	P	12/09/2020
24	Major requirements change	Clicks on “declare major”	Update course requirements and notify students	F	12/09/2020
	Edit Course				
25	Preconditions meet before testing use case	Not applicable	System shows classes with registered students	F	12/09/2020
26	System displays different courses with options to add new ones	Not applicable	Administrator should be able to click on any of those courses	P	12/09/2020
27	Administrator changes the name of the course to software architecture	Course Name : “Software Architecture”	System shows the changes made by the administrator	P	12/09/2020
28	Administrator saves changes	Clicks on save button	System stores the changes made for the course	P	12/09/2020
	Secondary Scenarios				

29	Administrator logs out before saving	Clicks on log out	System does not save the modifications	P	12/09/2020
30	Administrator creates a new course	Administrator clicks on “+” and inputs the attributes for the new course	System displays the necessary inputs for the creation of the course	P	12/10/2020
	Log out				
31	Preconditions meet before testing use case		User is enters its credentials on login page	P	12/09/2020
32	Student selects Current Pathway		System shows current Pathway	P	12/09/2020
33	Student selects ‘logout’ link at the top of the page		System shows exit page	P	12/09/2020
	Secondary Scenarios				
34	Student makes changes for pathway or schedule and saves them before logging out		System saves changes	F	12/09/2020
35	Student press exit button while changing major or making changes to schedule		Student exits the schedule planner	P	12/09/2020

Unit Tests

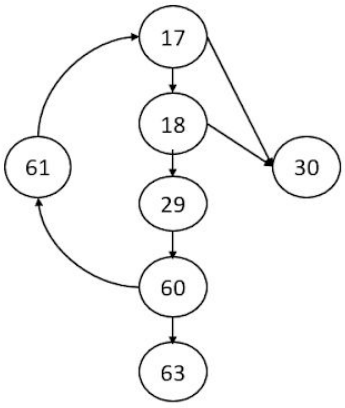
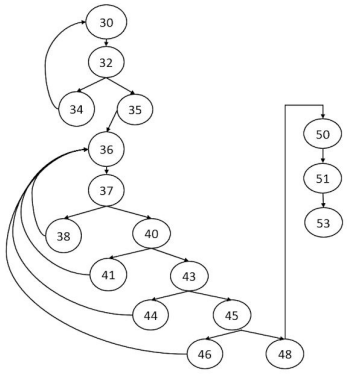
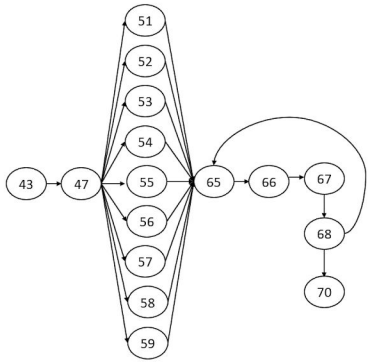
Testing Plan: Essentially the testing plan in regards to unit tests is parsing through the system and finding activities that the program must execute to have an efficient run.
In these, we've selected to test Manage.py file as it manages the intialial main method and init functions.We've selected to test the admin.py functionality as its essential to providing a smooth admin experience.We've selected the forms.py for testing as its essential in the user registration and login pages functionality. We've selected the models.py for testing as its responsible for creating the models used in the django database. We've selected the views.py as its responsible for displaying the various pages of the website; And finally we've decided to test the apps.py as its responsible for the reaction of the app config.

No.	Test Scenario	Program	Last Run	Duration	Result
1	Tests for the manage.py	Course_Planner_Project/manage.py	12/11/20	2.3ms	P
2	Tests Registration of the admin page	SEAS_Course_Planner/admin.py	12/11/20	1.2ms	P
3	Tests for user operations	Course_Planner_Project/Users/forms.py	12/11/20	2.3s	P
4	Tests for system specific model creation	**SEAS_Course_Planner**/models.py	12/11/20	10s	F
5	Tests for displaying views	/SEAS_Course_Planner/views.py	12/11/20	2s	P
6	Tests on	SEAS_Course_Planner/apps.py	12/11/	1.2ms	P

	creation of app config		20		

Code coverage tests using Flow Graphs to generate test cases

The following shows three different flows graphs made using the source code for the project. These graphs show the logic for the login, register and new pathway webpages for the project. The paths below represent the different steps that the user can take using each one of these webpages. These flow graphs helped to understand the limitations of user input on each page and have a better understanding on how to perform the unit testing.

Flow Graphs		
Login Page	Register Page	New Pathway
 Independent Paths: 17, 18, 29, 60, 63 // successful login 17, 18, 29, 60, 61,17 // unsuccessful login 17, 30 // go to sign up page	 Independent Paths: 30, 32, 35, 36, 37, 40, 43, 45, 48, 50, 51, 53// successful registration 30, 32, 34, 30// username already exists 30, 32, 35, 36, 37, 40, 41// password too short	 Independent Path: 43, 47, 54, 65, 66, 67, 68, 65, 66, 67, 68, 70 //successful computer science major declaration

Integration Tests

T1: Login Page

Test Description: The point of this test was to determine the performance of the login page; the objective is to make sure that the login page runs effectively and correctly.

Test Item: The test item is a series of methods written in Python that directly correlate and define the username and password sections.

Scope of Test: The scope of this test is to make sure that the login page works efficiently and doesn't result in any errors. This will not test the database.

Approach: The approach for this test will be to test the login page from the perspective of a user, specifically testing the performance and UI of the page.

Exit Criteria: Once the page worked efficiently and effectively, the tests were complete.

Test Schedule: The test took place on December 13th, and took no longer than five minutes.

Approval: The test was successful and the Login Page passed.

T2: Registration Page

Test Description: The point of this test was to determine the effectiveness of the registration page; the objective is to ensure the efficient performance of the registration page and ensuring that users can use the page.

Test Item: The test item is a series of methods written in Python that directly correlate and define the username and password sections, specifically validating the passwords and ensuring that they meet the password requirements.

Scope of Test: The scope of this test is to make sure that the registration page works efficiently and doesn't result in any errors. This will not test the ability to login with the user information.

Approach: The approach for this test will be to test the registration page from the perspective of a user, specifically testing the performance and UI of the page.

Exit Criteria: Once the page worked efficiently and effectively, the tests were complete.

Test Schedule: The test took place on December 13th, and took no longer than ten minutes.

Approval: The test was successful and the Registration Page passed.

T3: New Pathway Page

Test Description: The point of this test was to determine the efficiency of the new pathway page; the objective is to ensure the efficient performance of the new pathway page and ensuring that users can use the page and generate new pathways.

Test Item: The test item is a series of methods written in Python that directly correlate and define the course and degree information.

Scope of Test: The scope of this test is to make sure that the new pathway page works efficiently and doesn't result in any errors. This will not test pathways other than computer science since the project did not reach completion.

Approach: The approach for this test will be to test the new pathway page from the perspective of a user, specifically testing the performance and UI of the page.

Exit Criteria: Once the page worked efficiently and effectively, the tests were complete.

Test Schedule: The test took place on December 13th, and took no longer than fifteen minutes.

Approval: The test was successful and the New Pathway Page passed.

Executed System Tests

To perform a system test that validates the fully integration of all the component of our software, we performed the following tests:

1. Usability Testing :

   Our system has a minimalistic UI to make it easier for users to interact with it. It shares the same interface across all pages even though the content of these pages vary depending on their

functionality. There is always a header that could be use to access the main functionalities of the system

2. Load Testing:
   The page size of our webpage is 112.6 KB and after making 9 requests at the same time, the loaf time was 171 ms. Accessing user data and uploading data takes 217 ms on average for the website to respond.
3. Regression Testing:
   No bugs were found on the main web pages. The web pages were accessed multiple times to make sure that they all responded in the way and the results were consistent. However there are some functionalities that do not provide the expected outcome for the user.
4. Functional Testing
   Many of the expected functionalities for the software are not implemented and more than 40% of the use cases failed. The results showed that our system does not apply any storage methods for the courses or pathways. But the functionalities of user logging page and signup work properly and store user information as expected.
5. Hardware/Software Testing:
   The system is being hosted by pythonanywhere.com servers and the data is stored on the host using Django. The user interaction is heavily guided by the system since there are very specific requirements for creating a user, declaring a major, updating grades, etc. These requirements are shown to the end user through text and notifications.

Executed Validation and Verification Tests

Validation Testing:

1. Unit Testing:
   The unit testing allowed us to validate the functional implementation of the different components in our student and administrator webpages. 84% of the unit tests were successful, therefore we can conclude that most of the components work properly.
2. System Testing:
   Our system testing was performed to test how well the components work together. The data transferred between our system is handled by Django and it works as expected. Users and administrators have different web pages that share the same course information. However the information about the courses such as grading, dates, rotations is not properly shown in our entire system since it does not show the changes made by the student or administrator.
3. Integration Testing:
   We performed three tests to determine the effectiveness of the Logn, Registration and New Pathway webpages. All these tests were successful in a controlled environment, the development of the system is not completed yet therefore it was taken in consideration what functionalities and expectations were given for each of these specific tests.
4. User Acceptance Testing:
   The acceptance testing done by an external tester showed that 75% of the tests were successful.The user acceptance testing suggest that the team should spend more time on the

development of the project to allow the user to manipulate the data and reflect any changes of these data to both students and administrators

Verification Testing:

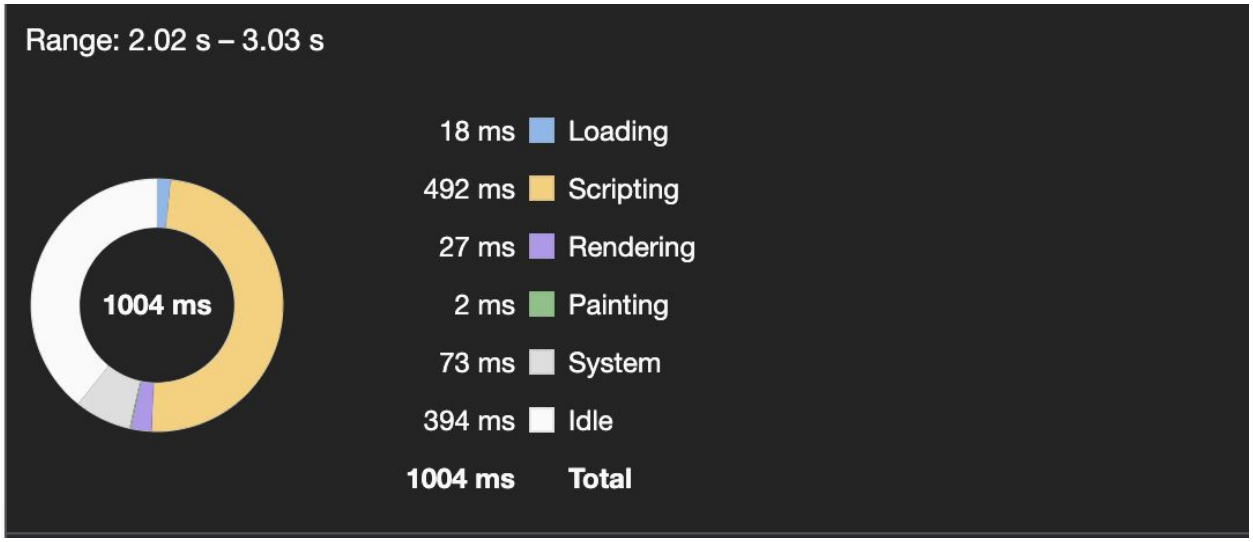
1. Functional Requirements
   This project meets 4 of the 6 initial requirements for the project. Even though the expectation of these 4 requirements is accurately presented in the final project (with some inconsistencies)it also requires our team to work on the other 2 requirements that we were unable to meet. The 2 requirements that we were not able to meet in our final project were: considering course rotation to create a plan and display all prerequisites for courses. The development of the project shows that these issues are mostly due to the lack of data management.
2. Design Review
   Throughout the development of the project the team made a total of 25 graphs including diagrams, charts,and wireframes. These graphs were used as a reference for the development of the system as a whole and for the source code. Some of these graphs were deleted or updated depending on their purpose and whether or not they meet the requirements of the project.
   The team also created 15 progress reports , 1 action plan and 3 versions of this documentation. The development of the project required a constant and clear communication between the team and the client. The client and the team meet over zoom on Wednesdays during 14 weeks to make sure that the development of the project is progressing according to the project requirements and expectations.
3. Testing Plan
   The testing plan executes different test for the system , this includes testing the software and the hardware used for the project. Data was gathered based on the performance of the system running in our hosting webpage, our original database(Raspberry pi) and the source code itself. Most of the tests were successful and they show that the final product is able to meet most of the requirements for this project however it still needs a more extensive development to fully meet all the requirements of the project and improve the consistency of the project outcomes.

Executed Acceptance Tests

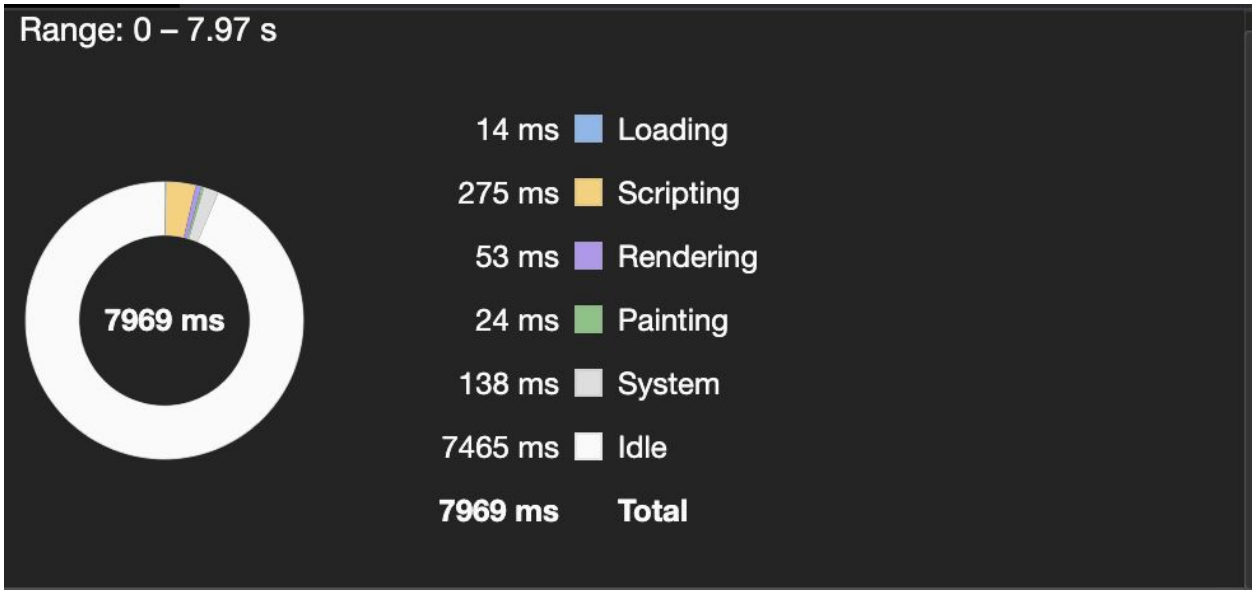
Number	Acceptance Requirement	Critical Yes	Critical No	Test Result Accept	Test Result Reject	Comments
1	The system must allow for Users to register an account or Log in.	X		X		

2	The system must allow for users to select from 9-degree programs and choose courses they've previously taken – and input grades.	X		X		
3	The system must allow admins (professors to add/modify courses to the database using the Django admin web page)	X		X		Allows selection but is not manipulated
4	The system must generate and display a current Pathway based on previously taken courses and grades, specific to the user	X			X	Displays but not accurate

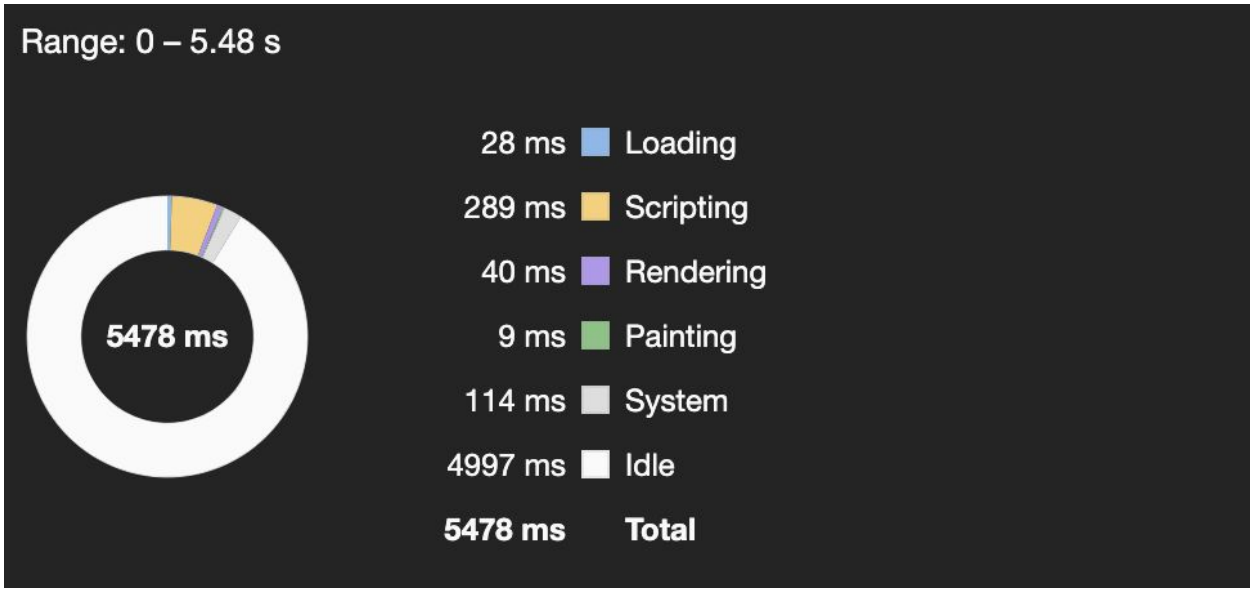
Chrome Browser DevTools Tests



This initial test measured the performance of the login landing page. To measure this, the performance was recorded as the login landing page was refreshed, measuring how long it took in the range of 1.01 seconds. As can be seen in the image above, most of the time used by the webpage was used by the scripting, second was – disregarding idleness – the system, third was rendering, fourth was loading, and fifth was painting.



The second test measured the performance when creating a new account. To measure this, the performance was recorded after the input of the information, then the information was submitted. Ignoring the time spent idling, the majority of the time when loading and processing the new account information was scripting followed by system, rendering, painting, and loading.



The third test measured the performance when logging in using an existing account. To measure this, the performance was recorded after the input of the login information, then the information was submitted. Ignoring the time spent idling, the majority of time was spent, once again, on scripting followed by system, rendering, loading, and painting.

Concluding Remarks:

The development of the SEAS Course planner has laid the foundation for all developers of the project and made us more familiar with the software development cycle and process. It's strengthened our documentation skills as well as general development skills as a whole - and was a great learning experience for all of us.

Appendix

Product Build Instructions:

Raspbian OS: To install raspbian on a raspberry pi, it is necessary to have an SD card with the OS in it (it can be downloaded from this page: https://www.raspberrypi.org/software/). Then this SD card should be inserted on the sd card slot of the raspberry. Once it turns on, it will automatically start the installation of the Raspbian OS and the SD card will be used as a storage.

Network: Connecting remotely to the raspberry pi requires a router that port forwards the ip address of the pi. For this project, our group used the remote desktop connection from windows. The router was configured to forward the 3389 port (commonly used for Windows Remote Desktop and Remote Assistance connections). Then on the windows application we inserted our IP address followed by the port number. For example: 50.30.###.###:3389

Docker: Installing docker on a linux system requires to update and upgrade the system and then proceed to download the docker software using curl -fsSL https://get.docker.com -o get-docker.sh
Then sudo usermod -a -G docker $USER and newgrp docker to allow access to all users without needing to use sude. Then to create a container it was necessary to pull an image from docker or to create a new one using nginx.

Nginx: To build the final container for project our team opted to use nginx instead of apache since it created our container for our website using our html,css,js and python files. To create the container using nginx a Dockerfile was created inside the same directory as our project. The content of the docker file is:
FROM nginx:alpine
COPY ./usr/share/nginx/html
Then on the terminal we run -- docker build -t *appName* .
This will create an image of the project, so the final step is to run it using the port 80
-- docker run -d -p 80:80 *appName*

Django: To install django in a ubuntu system it is necessary to have python installed first. Once python is installed in the server, use the following command to install django --sudo apt install python3-django. Since Django is a framework that helps software project, one of the advantages of it is that it automatically create a set up for the project using -- django-admin startproject mysite

Student/Teacher/Admin user guide Guide:
Student: Upon connecting to the sight the student is asked to login, or register an account with SEAS Course Planner Website. After creating a new account, or logging into a previous account - the student is then presented the new pathway page, which allows a user to select from a dropdown box, one of the nine SEAS degree programs - and then select the courses in the degree program they've already taken, and also input a grade. After the user has selected their courses, they must hit the submit button and then a pathway is generated - and the user is brought to the current pathway page. Which displays the generated pathway split into four years, the current pathway displays associate grades with previous courses and courses the Student should take to complete the degree program.

Admin/Teacher: The admin (teacher) must access the SEAS Course planner admin page, once there is an option to add courses to the database of courses provided. Once selected the admin is prompted to select which of the nine degree programs the course being offered pertains to, the course name, the semester the course will be offered, and finally the prerequisites for the course. Once all this data is filled out the admin can hit submit and the course will be added to the database and displayed to users as an available course. Along with adding courses - admins can delete and manage courses offered.

How to connect:

Along with the raspberry Pi, Our website is accessed by a 3rd party hosting application - “pythonanywhere.com” for ease of use and connection for other individuals. The website functions as normal using the django framework. Users connect by going to noahsmiley.pythonanywhere.com.

Tables

Sysbench Container Performance table

	Number of threads						
	1	2	4	8	16	32	64
CPU Time (in seconds)	10.0001	10.0001	10.0001	10.0001	10.0001	10.0001	10.0001
	10.0001	10.0001	10.0001	10.0001	10.0001	10.0001	10.0001
	10.0001	10.0001	10.0001	10.0001	10.0001	10.0001	10.0001
	10.0013	10.0013	10.0013	10.0013	10.0013	10.0013	10.0013
	10.001	10.001	10.001	10.001	10.001	10.001	10.001
	10.0002	10.0002	10.0002	10.0002	10.0002	10.0002	10.0002
	10.0005	10.0005	10.0005	10.0005	10.0005	10.0005	10.0005
	10.0003	10.0003	10.0003	10.0003	10.0003	10.0003	10.0003
	10.0005	10.0005	10.0005	10.0005	10.0005	10.0005	10.0005
	10.0004	10.0004	10.0004	10.0004	10.0004	10.0004	10.0004
	10.0004	10.0004	10.0004	10.0004	10.0004	10.0004	10.0004
	10.0008	10.0008	10.0008	10.0008	10.0008	10.0008	10.0008
	10.0009	10.0009	10.0009	10.0009	10.0009	10.0009	10.0009
	10.0009	10.0009	10.0009	10.0009	10.0009	10.0009	10.0009
	10.0011	10.0011	10.0011	10.0011	10.0011	10.0011	10.0011
	10.0007	10.0007	10.0007	10.0007	10.0007	10.0007	10.0007
	10.0013	10.0013	10.0013	10.0013	10.0013	10.0013	10.0013
	10.0009	10.0009	10.0009	10.0009	10.0009	10.0009	10.0009
	10.0007	10.0007	10.0007	10.0007	10.0007	10.0007	10.0007
	10.0006	10.0006	10.0006	10.0006	10.0006	10.0006	10.0006
	10.0006	10.0006	10.0006	10.0006	10.0006	10.0006	10.0006
	10.0009	10.0009	10.0009	10.0009	10.0009	10.0009	10.0009
	10.0005	10.0005	10.0005	10.0005	10.0005	10.0005	10.0005
	10.0001	10.0001	10.0001	10.0001	10.0001	10.0001	10.0001
	10.0004	10.0004	10.0004	10.0004	10.0004	10.0004	10.0004
	10.0002	10.0002	10.0002	10.0002	10.0002	10.0002	10.0002
	10.0003	10.0003	10.0003	10.0003	10.0003	10.0003	10.0003
	10.0005	10.0005	10.0005	10.0005	10.0005	10.0005	10.0005
	10.0001	10.0001	10.0001	10.0001	10.0001	10.0001	10.0001
	10.0003	10.0003	10.0003	10.0003	10.0003	10.0003	10.0003
		10.0011	10.0011	10.0011	10.0011	10.0011	10.0011
		10.0009	10.0009	10.0009	10.0009	10.0009	10.0009
		10.0009	10.0009	10.0009	10.0009	10.0009	10.0009
		10.0012	10.0012	10.0012	10.0012	10.0012	10.0012
		10.0011	10.0011	10.0011	10.0011	10.0011	10.0011
		10.0008	10.0008	10.0008	10.0008	10.0008	10.0008
		10.001	10.001	10.001	10.001	10.001	10.001
		10.0008	10.0008	10.0008	10.0008	10.0008	10.0008
		10.0015	10.0015	10.0015	10.0015	10.0015	10.0015
		10.0011	10.0011	10.0011	10.0011	10.0011	10.0011
			10.0013	10.0013	10.0013	10.0013	10.0013
			10.0007	10.0007	10.0007	10.0007	10.0007
			10.002	10.002	10.002	10.002	10.002
			10.0022	10.0022	10.0022	10.0022	10.0022
			10.0021	10.0021	10.0021	10.0021	10.0021
			10.0017	10.0017	10.0017	10.0017	10.0017
			10.0018	10.0018	10.0018	10.0018	10.0018

			10.0024	10.0024	10.0024	10.0024	10.0024
			10.0019	10.0019	10.0019	10.0019	10.0019
			10.001	10.001	10.001	10.001	10.001
				10.0033	10.0033	10.0033	10.0033
				10.0036	10.0036	10.0036	10.0036
				10.0033	10.0033	10.0033	10.0033
				10.0081	10.0081	10.0081	10.0081
				10.004	10.004	10.004	10.004
				10.0029	10.0029	10.0029	10.0029
				10.0031	10.0031	10.0031	10.0031
				10.0038	10.0038	10.0038	10.0038
				10.0033	10.0033	10.0033	10.0033
				10.0043	10.0043	10.0043	10.0043
					10.0055	10.0055	10.0055
					10.0058	10.0058	10.0058
					10.0053	10.0053	10.0053
					10.0057	10.0057	10.0057
					10.0055	10.0055	10.0055
					10.0054	10.0054	10.0054
					10.0054	10.0054	10.0054
					10.0059	10.0059	10.0059
					10.0058	10.0058	10.0058
					10.0059	10.0059	10.0059
						10.0113	10.0113
						10.0114	10.0114
						10.01	10.01
						10.0123	10.0123
						10.0128	10.0128
						10.0132	10.0132
						10.012	10.012
						10.0113	10.0113
						10.0105	10.0105
						10.0139	10.0139
							10.0238
							10.0245
							10.0251
							10.0238
							10.0232
							10.0252
							10.0241
							10.0258
							10.0307
							10.0217
Total:	300.0167	400.0271	500.0442	600.0839	700.1401	800.2588	900.5067

Network Performance Tables:

LAN (Local Area Network)

Windows to
Raspberry Pi

Transfer Size (in MB)	Transfer Time (in seconds)	Bandwidth Average (in Mbits/sec)
1	0.16	53
2	0.45	39.5
4	0.6	56.3
8	1.66	40.5
16	2.69	50
32	6.32	42.5
64	12.32	43.6

WAN (Wide Area Network)

Windows to Docker Container

Transfer Size (in MB)	Transfer Time (in seconds)	Bandwidth Average (in Mbits/sec)
1	0.2	45
2	0.51	43.7
4	0.78	44
8	1.66	40.5
16	3.24	39.7
32	7.43	40.3
64	14.2	41.5

Windows to Raspberry Pi

Transfer Size (in MB)	Transfer Time (in seconds)	Bandwidth Average (in Mbits/sec)
1	0.23	43
2	0.51	48.9
4	0.73	48
8	1.8	43.5
16	2.53	49
32	5.45	45.3
64	14.3	41

Windows to Docker Container

Transfer Size (in MB)	Transfer Time (in seconds)	Bandwidth Average (in Mbits/sec)
1	0.31	39
2	0.63	37
4	0.78	44
8	2.03	41.3
16	3.41	38.7
32	8.4	33.4
64	15.2	38.4

Source Code

"""
ASGI config for Course_Planner_Project project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/3.1/howto/deployment/asgi/
"""

import os

```
from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Course_Planner_Project.settings')

application = get_asgi_application()


"""
Django settings for Course_Planner_Project project.

Generated by 'django-admin startproject' using Django 3.1.3.

For more information on this file, see
https://docs.djangoproject.com/en/3.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.1/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '$cxvtg#jt3ta(=p^d3ue9a!!6(!lm1eo0^5qvkaxwc*&am!p!r'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'Users.apps.UsersConfig',
    'crispy_forms',
    'SEAS_Course_Planner.apps.SeasCoursePlannerConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

```
ROOT_URLCONF = 'Course_Planner_Project.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'Course_Planner_Project.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True
```

```
USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.1/howto/static-files/

STATIC_URL = '/static/'

CRISPY_TEMPLATE_PACK = 'bootstrap4'

LOGIN_REDIRECT_URL = 'SEAS_Course_Planner-currentpathway'

LOGIN_URL = 'login'


from django.contrib import admin
from django.contrib.auth import views as auth_views
from django.urls import path, include
from Users import views as users_views



urlpatterns = [
        path('admin/', admin.site.urls),
    path('', include('SEAS_Course_Planner.urls')),
    path('register/', users_views.register, name= 'register'),
    path('login/', auth_views.LoginView.as_view(template_name='users/login.html'), name= 'login'),
    path('logout/', auth_views.LogoutView.as_view(template_name='users/logout.html'), name= 'logout'),
    path('newpathway/', include('SEAS_Course_Planner.urls')),


]

"""
WSGI config for Course_Planner_Project project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/3.1/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Course_Planner_Project.settings')

application = get_wsgi_application()


#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
```

```
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Course_Planner_Project.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()


from django.contrib import admin
from .models import Architectural_Science_Courses_Offered
from .models import Civil_Engineering_Courses_Offered
from .models import Computer_Information_Technology_Courses_Offered
from .models import Computer_Science_Courses_Offered
from .models import Construction_Management_Courses_Offered
from .models import Electrical_Engineering_Courses_Offered
from .models import Engineering_Technology_Management_Courses_Offered
from .models import Manufacturing_Engineering_Technology_Courses_Offered
from .models import Mechanical_Engineering_Courses_Offered

# Register your models here.
admin.site.register(Architectural_Science_Courses_Offered)
admin.site.register(Civil_Engineering_Courses_Offered)
admin.site.register(Computer_Information_Technology_Courses_Offered)
admin.site.register(Computer_Science_Courses_Offered)
admin.site.register(Construction_Management_Courses_Offered)
admin.site.register(Electrical_Engineering_Courses_Offered)
admin.site.register(Engineering_Technology_Management_Courses_Offered)
admin.site.register(Manufacturing_Engineering_Technology_Courses_Offered)
admin.site.register(Mechanical_Engineering_Courses_Offered)


from django.apps import AppConfig


class SeasCoursePlannerConfig(AppConfig):
    name = 'SEAS_Course_Planner'


# Generated by Django 3.1.3 on 2020-11-04 09:44

from django.db import migrations, models


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='coursesOffered',
```

```
            fields=[
                                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
                ('courseName', models.CharField(max_length=250)),
                ('professorName', models.CharField(max_length=100)),
                ('preRequisites', models.TextField()),
            ],
        ),
    ]


# Generated by Django 3.1.3 on 2020-11-05 03:54

from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('SEAS_Course_Planner', '0001_initial'),
    ]

    operations = [
        migrations.AddField(
            model_name='coursesoffered',
            name='semester',
                field=models.CharField(choices=[('SPR 21', 'SPR 21'), ('FA 21', 'FA 21'), ('SPR 22', 'SPR 22'),
('FA 22', 'FA 22'), ('SPR 23', 'SPR 23'), ('FA 23', 'FA 23'), ('SPR 24 ', 'SPR 24'), ('FA 24', 'FA 24')],
default='SPR 21', max_length=20),
        ),
    ]


# Generated by Django 3.1.3 on 2020-11-05 04:06

from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('SEAS_Course_Planner', '0002_coursesoffered_semester'),
    ]

    operations = [
        migrations.AddField(
            model_name='coursesoffered',
            name='degreeProgram',
                    field=models.CharField(choices=[('Architectural Science', 'Architectural Science'), ('Civil
Engineering', 'Civil Engineering'), ('Computer Information Technology', 'Computer Information
Technology'), ('Computer Science', 'Computer Science'), ('Construction Management', 'Construction
Management'), ('Electrical Engineering', 'Electrical Engineering'), ('Engineering Technology
Management', 'Engineering Technology Management'), ('Manufacturing Engineering Technology',
'Manufacturing Engineering Technology'), ('Mechanical Engineering', 'Mechanical Engineering')],
default='Architectural Science', max_length=50),
        ),
    ]


# Generated by Django 3.1.3 on 2020-11-05 04:14
```

```
from django.db import migrations


class Migration(migrations.Migration):

    dependencies = [
        ('SEAS_Course_Planner', '0003_coursesoffered_degreeprogram'),
    ]

    operations = [
        migrations.RemoveField(
            model_name='coursesoffered',
            name='professorName',
        ),
    ]


# Generated by Django 3.1.3 on 2020-11-06 20:50

from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('SEAS_Course_Planner', '0004_remove_coursesoffered_professorname'),
    ]

    operations = [
        migrations.CreateModel(
            name='Architectural_Science_Courses_Offered',
            fields=[
                                ('id',  models.AutoField(auto_created=True,  primary_key=True,  serialize=False,
verbose_name='ID')),
                ('courseName', models.CharField(max_length=250)),
                 ('semester', models.CharField(choices=[('SPR 21', 'SPR 21'), ('FA 21', 'FA 21'), ('SPR 22', 'SPR
22'), ('FA 22', 'FA 22'), ('SPR 23', 'SPR 23'), ('FA 23', 'FA 23'), ('SPR 24 ', 'SPR 24'), ('FA 24', 'FA 24')],
default='SPR 21', max_length=20)),
                ('preRequisites', models.TextField()),
            ],
        ),
        migrations.CreateModel(
            name='Civil_Engineering_Courses_Offered',
            fields=[
                                ('id',  models.AutoField(auto_created=True,  primary_key=True,  serialize=False,
verbose_name='ID')),
                ('courseName', models.CharField(max_length=250)),
                 ('semester', models.CharField(choices=[('SPR 21', 'SPR 21'), ('FA 21', 'FA 21'), ('SPR 22', 'SPR
22'), ('FA 22', 'FA 22'), ('SPR 23', 'SPR 23'), ('FA 23', 'FA 23'), ('SPR 24 ', 'SPR 24'), ('FA 24', 'FA 24')],
default='SPR 21', max_length=20)),
                ('preRequisites', models.TextField()),
            ],
        ),
        migrations.CreateModel(
            name='Computer_Information_Technology_Courses_Offered',
            fields=[
                                ('id',  models.AutoField(auto_created=True,  primary_key=True,  serialize=False,
verbose_name='ID')),
                ('courseName', models.CharField(max_length=250)),
```

```
            ('semester', models.CharField(choices=[('SPR 21', 'SPR 21'), ('FA 21', 'FA 21'), ('SPR 22', 'SPR
22'), ('FA 22', 'FA 22'), ('SPR 23', 'SPR 23'), ('FA 23', 'FA 23'), ('SPR 24 ', 'SPR 24'), ('FA 24', 'FA 24')],
default='SPR 21', max_length=20)),
            ('preRequisites', models.TextField()),
        ],
    ),
    migrations.CreateModel(
        name='Computer_Science_Courses_Offered',
        fields=[
                    ('id',  models.AutoField(auto_created=True,  primary_key=True,  serialize=False,
verbose_name='ID')),
            ('courseName', models.CharField(max_length=250)),
            ('semester', models.CharField(choices=[('SPR 21', 'SPR 21'), ('FA 21', 'FA 21'), ('SPR 22', 'SPR
22'), ('FA 22', 'FA 22'), ('SPR 23', 'SPR 23'), ('FA 23', 'FA 23'), ('SPR 24 ', 'SPR 24'), ('FA 24', 'FA 24')],
default='SPR 21', max_length=20)),
            ('preRequisites', models.TextField()),
        ],
    ),
    migrations.CreateModel(
        name='Construction_Management_Courses_Offered',
        fields=[
                    ('id',  models.AutoField(auto_created=True,  primary_key=True,  serialize=False,
verbose_name='ID')),
            ('courseName', models.CharField(max_length=250)),
            ('semester', models.CharField(choices=[('SPR 21', 'SPR 21'), ('FA 21', 'FA 21'), ('SPR 22', 'SPR
22'), ('FA 22', 'FA 22'), ('SPR 23', 'SPR 23'), ('FA 23', 'FA 23'), ('SPR 24 ', 'SPR 24'), ('FA 24', 'FA 24')],
default='SPR 21', max_length=20)),
            ('preRequisites', models.TextField()),
        ],
    ),
    migrations.CreateModel(
        name='Electrical_Engineering_Courses_Offered',
        fields=[
                    ('id',  models.AutoField(auto_created=True,  primary_key=True,  serialize=False,
verbose_name='ID')),
            ('courseName', models.CharField(max_length=250)),
            ('semester', models.CharField(choices=[('SPR 21', 'SPR 21'), ('FA 21', 'FA 21'), ('SPR 22', 'SPR
22'), ('FA 22', 'FA 22'), ('SPR 23', 'SPR 23'), ('FA 23', 'FA 23'), ('SPR 24 ', 'SPR 24'), ('FA 24', 'FA 24')],
default='SPR 21', max_length=20)),
            ('preRequisites', models.TextField()),
        ],
    ),
    migrations.CreateModel(
        name='Engineering_Technology_Management_Courses_Offered',
        fields=[
                    ('id',  models.AutoField(auto_created=True,  primary_key=True,  serialize=False,
verbose_name='ID')),
            ('courseName', models.CharField(max_length=250)),
            ('semester', models.CharField(choices=[('SPR 21', 'SPR 21'), ('FA 21', 'FA 21'), ('SPR 22', 'SPR
22'), ('FA 22', 'FA 22'), ('SPR 23', 'SPR 23'), ('FA 23', 'FA 23'), ('SPR 24 ', 'SPR 24'), ('FA 24', 'FA 24')],
default='SPR 21', max_length=20)),
            ('preRequisites', models.TextField()),
        ],
    ),
    migrations.CreateModel(
        name='Manufacturing_Engineering_Technology_Courses_Offered',
        fields=[
                    ('id',  models.AutoField(auto_created=True,  primary_key=True,  serialize=False,
verbose_name='ID')),
            ('courseName', models.CharField(max_length=250)),
```

```
            ('semester', models.CharField(choices=[('SPR 21', 'SPR 21'), ('FA 21', 'FA 21'), ('SPR 22', 'SPR 22'), ('FA 22', 'FA 22'), ('SPR 23', 'SPR 23'), ('FA 23', 'FA 23'), ('SPR 24 ', 'SPR 24'), ('FA 24', 'FA 24')], default='SPR 21', max_length=20)),
            ('preRequisites', models.TextField()),
        ],
    ),
    migrations.CreateModel(
        name='Mechanical_Engineering_Courses_Offered',
        fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
            ('courseName', models.CharField(max_length=250)),
            ('semester', models.CharField(choices=[('SPR 21', 'SPR 21'), ('FA 21', 'FA 21'), ('SPR 22', 'SPR 22'), ('FA 22', 'FA 22'), ('SPR 23', 'SPR 23'), ('FA 23', 'FA 23'), ('SPR 24 ', 'SPR 24'), ('FA 24', 'FA 24')], default='SPR 21', max_length=20)),
            ('preRequisites', models.TextField()),
        ],
    ),
    migrations.DeleteModel(
        name='coursesOffered',
    ),
]


from django.db import models

# Create your models here.


SEMESTER_CHOICES = (
    ("SPR 21", "SPR 21"),
    ("FA 21", "FA 21"),
    ("SPR 22", "SPR 22"),
    ("FA 22", "FA 22"),
    ("SPR 23", "SPR 23"),
    ("FA 23", "FA 23"),
    ("SPR 24 ", "SPR 24"),
    ("FA 24", "FA 24"),
)
# declaring a Student Model



class Architectural_Science_Courses_Offered(models.Model):

        courseName = models.CharField(max_length=250)

        semester = models.CharField(
    max_length = 20,
    choices = SEMESTER_CHOICES,
    default = 'SPR 21'
    )

        preRequisites = models.TextField()


        def __str__ (self):
                return self.courseName

class Civil_Engineering_Courses_Offered(models.Model):
```

```
        courseName = models.CharField(max_length=250)

        semester = models.CharField(
    max_length = 20,
    choices = SEMESTER_CHOICES,
    default = 'SPR 21'
    )

        preRequisites = models.TextField()


        def __str__ (self):
                return self.courseName

class Computer_Information_Technology_Courses_Offered(models.Model):

        courseName = models.CharField(max_length=250)

        semester = models.CharField(
    max_length = 20,
    choices = SEMESTER_CHOICES,
    default = 'SPR 21'
    )

        preRequisites = models.TextField()


        def __str__ (self):
                return self.courseName

class Computer_Science_Courses_Offered(models.Model):

        courseName = models.CharField(max_length=250)

        semester = models.CharField(
    max_length = 20,
    choices = SEMESTER_CHOICES,
    default = 'SPR 21'
    )

        preRequisites = models.TextField()

        def __str__ (self):
                return self.courseName

class Construction_Management_Courses_Offered(models.Model):

        courseName = models.CharField(max_length=250)

        semester = models.CharField(
    max_length = 20,
    choices = SEMESTER_CHOICES,
    default = 'SPR 21'
    )

        preRequisites = models.TextField()

        def __str__ (self):
                return self.courseName
```

```
class Electrical_Engineering_Courses_Offered(models.Model):

        courseName = models.CharField(max_length=250)

        semester = models.CharField(
    max_length = 20,
    choices = SEMESTER_CHOICES,
    default = 'SPR 21'
    )

        preRequisites = models.TextField()

class Engineering_Technology_Management_Courses_Offered(models.Model):

        courseName = models.CharField(max_length=250)

        semester = models.CharField(
    max_length = 20,
    choices = SEMESTER_CHOICES,
    default = 'SPR 21'
    )

        preRequisites = models.TextField()

        def __str__ (self):
                return self.courseName


class Manufacturing_Engineering_Technology_Courses_Offered(models.Model):

        courseName = models.CharField(max_length=250)

        semester = models.CharField(
    max_length = 20,
    choices = SEMESTER_CHOICES,
    default = 'SPR 21'
    )

        preRequisites = models.TextField()

        def __str__ (self):
                return self.courseName


class Mechanical_Engineering_Courses_Offered(models.Model):

        courseName = models.CharField(max_length=250)

        semester = models.CharField(
    max_length = 20,
    choices = SEMESTER_CHOICES,
    default = 'SPR 21'
    )

        preRequisites = models.TextField()

        def __str__ (self):
                return self.courseName


* {
```

```
  margin: 0;
  box-sizing: border-box;
}

h1{
  font-size: 50px;
  margin: -2px;
}

h2 {
  padding-bottom: 20px;
  padding-top: 30px;
  margin: 0px;
}

.select-box {
  display: flex;
  width: 400px;
  flex-direction: column;
}

.select-box .options-container {
  background: #2f3640;
  color: #f5f6fa;
  max-height: 0;
  width: 100%;
  opacity: 0;
  transition: all 0.4s;
  border-radius: 8px;
  overflow: hidden;

  order: 1;
}

.selected {
  background: #2f3640;
  border-radius: 8px;
  margin-bottom: 8px;
  color: #f5f6fa;
  position: relative;

  order: 0;
}

.selected::after {
  content: "";
  background: url("img/arrow-down.svg");
  background-size: contain;
  background-repeat: no-repeat;

  position: absolute;
  height: 100%;
  width: 32px;
  right: 10px;
  top: 5px;

  transition: all 0.4s;
}

.select-box .options-container.active {
  max-height: 240px;
```

```
  opacity: 1;
  overflow-y: scroll;
}

.select-box .options-container.active + .selected::after {
  transform: rotateX(180deg);
  top: -6px;
}

.select-box .options-container::-webkit-scrollbar {
  width: 8px;
  background: #0d141f;
  border-radius: 0 8px 8px 0;
}

.select-box .options-container::-webkit-scrollbar-thumb {
  background: #525861;
  border-radius: 0 8px 8px 0;
}

.select-box .option,
.selected {
  padding: 12px 24px;
  cursor: pointer;
}

.select-box .option:hover {
  background: #414b57;
}

.select-box label {
  cursor: pointer;
}

.select-box .option .Degree {
  display: none;
}


body {
  background: #fafafa;
  color: #333333;
  margin-top: 0rem;
}

h1, h2, h3, h4, h5, h6 {
  color: #444444;
}

ul {
  margin: 0;
}

.bg-steel {
  background-color: #003366;
}

.site-header .navbar-nav .nav-link {
  color: #cbd5db;
}
```

```
.site-header .navbar-nav .nav-link:hover {
  color: #ffffff;
}

.site-header .navbar-nav .nav-link.active {
  font-weight: 500;
}

.content-section {
  background: #ffffff;
  padding: 10px 20px;
  border: 1px solid #dddddd;
  border-radius: 3px;
  margin-bottom: 20px;
}

.article-title {
  color: #444444;
}

a.article-title:hover {
  color: #428bca;
  text-decoration: none;
}

.article-content {
  white-space: pre-line;
}

.article-img {
  height: 65px;
  width: 65px;
  margin-right: 16px;
}

.article-metadata {
  padding-bottom: 1px;
  margin-bottom: 4px;
  border-bottom: 1px solid #e3e3e3
}

.article-metadata a:hover {
  color: #333;
  text-decoration: none;
}

.article-svg {
  width: 25px;
  height: 25px;
  vertical-align: middle;
}

.account-img {
  height: 125px;
  width: 125px;
  margin-right: 20px;
  margin-bottom: 16px;
}

.account-heading {
  font-size: 2.5rem;
```

```
}


from django.urls import path
from . import views


urlpatterns = [
    path('', views.currentpathway, name ='SEAS_Course_Planner-currentpathway'),
    path('newpathway/', views.newpathway, name ='SEAS_Course_Planner-newpathway'),
]


from django.shortcuts import render
from django.http import HttpResponse
from django.contrib.auth.decorators import login_required
# Create your views here.
@login_required
def currentpathway(request):

        return render(request, "SEAS_Course_Planner/home.html")
@login_required
def newpathway(request):
        return render(request, "SEAS_Course_Planner/newpathway.html")


from django.apps import AppConfig


class UsersConfig(AppConfig):
    name = 'Users'


from django import forms
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm

class UserRegisterForm(UserCreationForm):
        pass
class Meta:

        model = User
        fields = ['username', 'email', 'password1', 'password2']



from django.shortcuts import render, redirect
from django.contrib import messages
from .forms import UserRegisterForm
# Create your views here.

def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            messages.success(request, f'Your account has been created! You are now able to log in')
            return redirect('login')
    else:
        form = UserRegisterForm()
```

```
    return render(request, 'users/register.html' , {'form': form})


def profile(request):
    return render(request, 'users/profile.html')
```