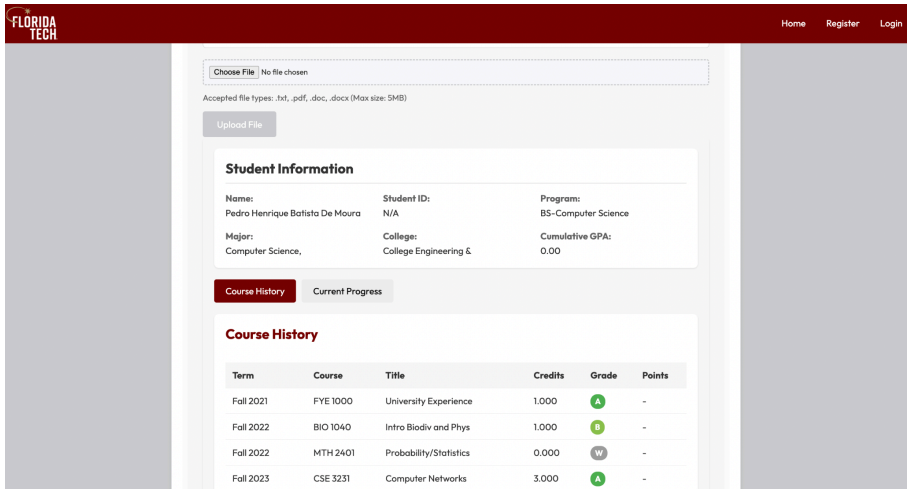
Milestone #03 - Progress Evaluation

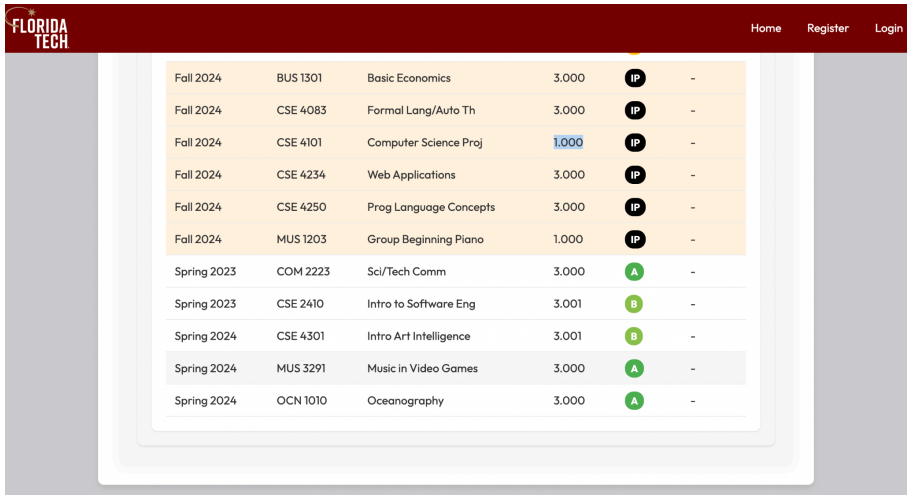
FIT Schedule Planner
Pedro Moura - pmoura2020@my.fit.edu
Jordan Synodis - jsynodis2021@my.fit.edu
Dr. Fitz - fnembhard@fit.edu

Task	Completion	Pedro	Jordan	Todos
1. Implement, test, and demo loading the CAPP Degree Evaluation	40%	100%	0%	Find a better way to do this & compare processed file with the checklist
2. Implement, test, and demo accessing the program checklist	50%	0%	100%	Fix client-side files for deployment, incorporate data from degree evaluation for progress tracking
3. Implement, test, and demo advanced filtering options	90%	0%	100%	Fix some font colors, fix some text boxes where text is cut off, fix bug with time conflict, add rateMyProf filter
4. Fix Database	100%	100%	0%	none
5. Implement, test, and demo Tree Prerequisites	90%	100%	0%	Improve Course Selection
6. Fix the Weekly Schedule grid	100%	50%	50%	none

- Discussion of each accomplished task (and obstacles) for the current Milestone:
  - Task 1. Implement, test, and demo loading the CAPP Degree:
    - Developed a parser that every time a user tries to upload a file (ex. Unofficial transcript), it parses and tries to extract the file content. Currently is not getting all the data from the file because of data format. The file had inconsistent whitespace and newlines, which was causing the regex pattern matching to break. After trying for many times, our team believes that we should come up with a better solution than trying to read the uploaded file.

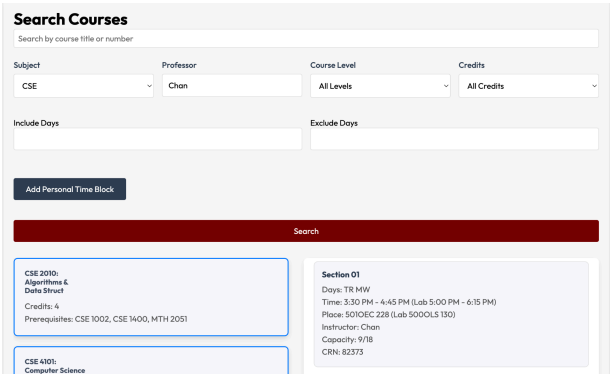
- Demo:

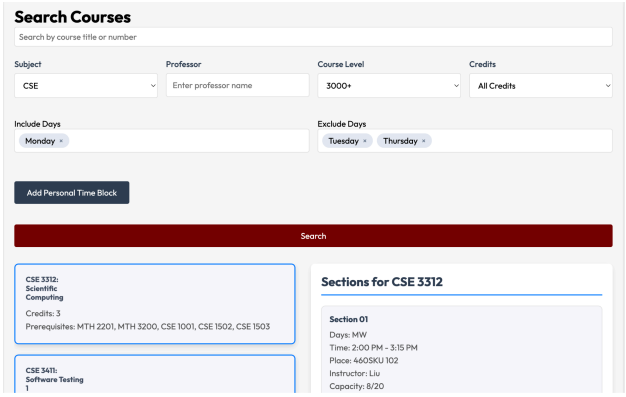


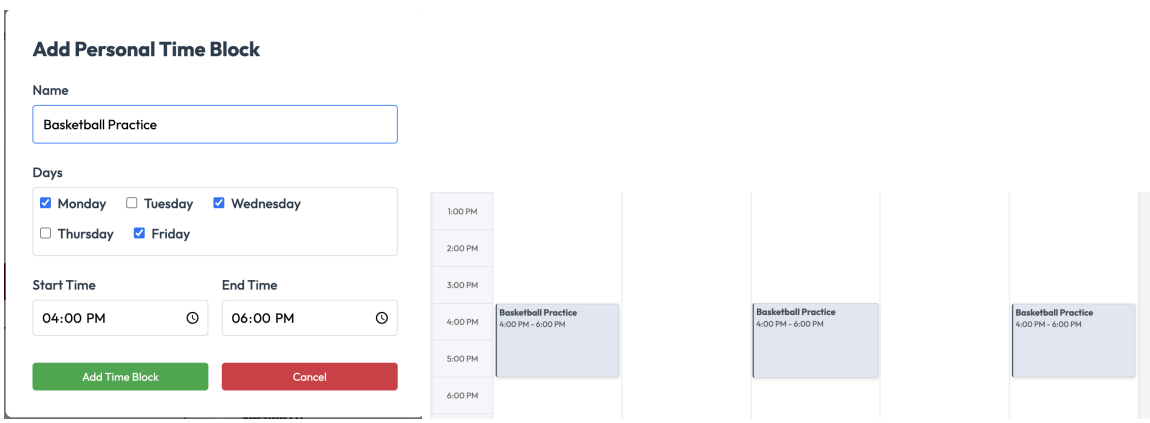


- Task 2: Implement, test, and demo accessing the program checklist
  - The checklist is supposed to access the database to allow the user to select their program and then take their degree evaluation and depict what they have completed and what they still need to do
  - The program json file has been uploaded to the database and server-side files were created, allowing the web application to access the needed data
  - Client-side files were edited to add the actual interface that the user interacts with, but we're running into some bugs with the deployment

- Task 3: Implement, test, and demo advanced filtering options
  - We had the search bar working but needed to add filters to help narrow down the long list of classes. Users should be able to include/exclude days, search by professors, subject, etc. In addition to filters, users should be able to add a time block to the schedule so they can add their extracurriculars like work or sports
  - All these features were added and are fully functional
  - Demo





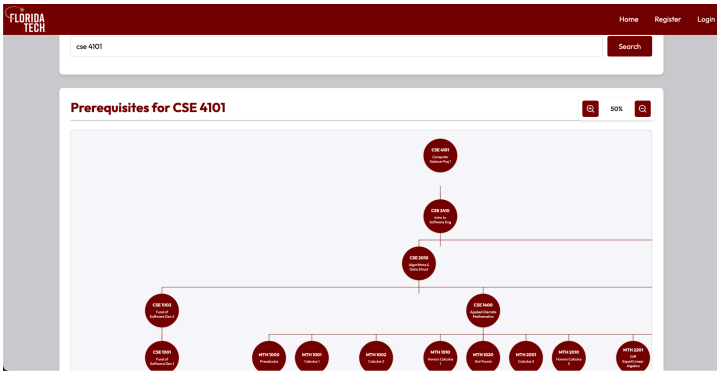


- Task 4. Fix Database:
  - We were having two problems with our database, connection and fetching correct data:
    - Connection:
      - Sometimes, our team could not access the data or connect to the database. The problem was that our team had to keep adding the IP addresses they were on every single time, and sometimes it kept changing the IP due to the firewall. The solution was to add the “0.0.0.0/0” on the database network access so that we could connect to the database from anywhere.
    - Fetching correct data:
      - Because our data was generated by scraping, sometimes it gets some random white spaces or newlines, resulting in

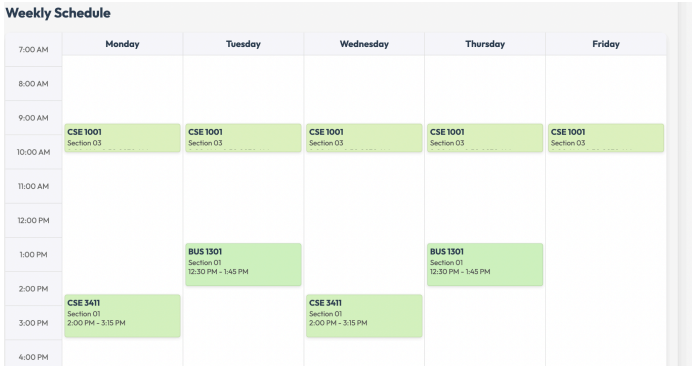
inaccurate results when fetching the data. So, a script was created to deal with this weird spacing and format data better.

- Task 5. Implement, test, and demo Tree Prerequisites:
  - After receiving a lot of the same feedback about making a Tree for the prerequisites, we gave it a thought, and it actually could be helpful for future features (ex. adding classes on schedule). The initial problem was showing course prerequisites using a clear visual hierarchy. The connecting line between nodes needed to be corrected. Hence, the correct restructuring of CSS and appropriate validation solved the problem. Implementation was improved by replacing horizontal scrolling with zoom functionality, allowing users to navigate large chains of prerequisites more straightforwardly. Also, designed backend logic to avoid infinite loops on course circular prerequisites.

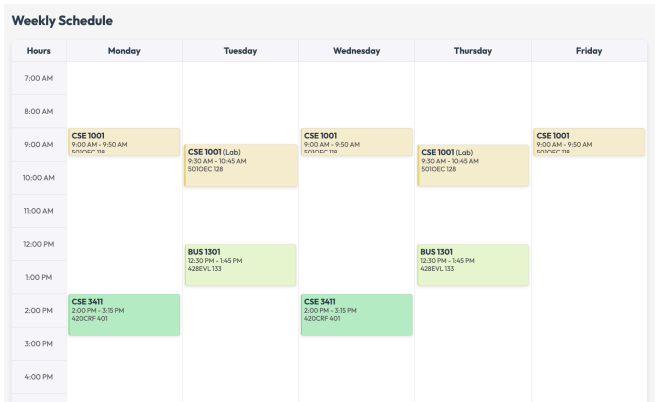
  - Demo:



- Task 6. Fix the Weekly Schedule grid:
  - The actual fix was simple, the main issue was to find what and where was causing the problem. The first approach was to add some elements (ex: <dig> tags) to observe how it behaves, but it did not help much. What actually helped was modifying the CSS file by adding borders on each element to see where it was affecting. While fixing this, feedback was received that it could have been more visually pleasant to see all classes on the schedule having different colors.
  - Demo:
    - Before:



- After:



- Discussion of contribution of each team member to the current Milestone:
  - Pedro Moura:
    - Built the file scraper for the uploaded file (CAPP)
    - Found and fixed the bug on the database that was causing inaccurate fetching
    - Found and fixed the bug on the database that was causing failed connections
    - Developed course pre-requisite tree visualization
    - Fix the weekly schedule grid bug that was causing
    - Implemented algorithm to display different colors for each class added to the schedule
    - Added a feature to show on schedule if it is a class or a lab

  - Jordan Synodis:
    - Built the course's advanced filtering feature
    - Added personal time blocks
    - Added time conflict checks between classes and personal blocks
    - Found and fixed a bug with text cut-offs in the schedule grid
    - Added programs to the database
    - Created client and server-side files for the program checklist

        - Added program checklist to progress tab in web application

- Plan for the next Milestone (task matrix):

Task	Pedro	Jordan
1. Implement, test, and demo loading the CAPP Degree Evaluation	50%	50%
2. Implement, test, and demo accessing the program checklist	50%	50%
3. Implement test, and demo additional filtering option, address bugs with formatting	0%	100%
4. Fix User Context	100%	0%

- Description for each task in the task summary for the next milestone:
    - Task 1: Find a better way to extract data from the uploaded file or come up with a better solution for this problem (track the user's progress). Current ideas:
        - Idea 1:
            - If the user is a new student, they won't need to upload anything, so our application will work. For current students, show something like a "to-do" section, where users can add classes they already took or are registered so they can keep track and we can use this information to make our recommendation system works.
        - Idea 2:
            - Use LLM to read the uploaded file and write a specific format (Ex. JSON) so we can use it. (Explore the area of Agent AI).

    - Task 2: The server end of this feature is running as expected, but the actual deployment is causing the whole program to crash. This bug needs to be identified and handled to properly deploy the progress checklist. In addition to this bug, the data from the CAPP Degree Evaluation should be concatenated with the checklist so the web application actually displays the user's progress toward their selected program.

    - Task 3: The last filter that needs to be added is the RateMyProfessor score. We would have to look into the API for RateMyProfessor and see exactly how we are going to get that data and store it in the database. Once we get that, adding to already existing features should not be a hard feat.

    - Task 4: After the user login, the web app should save the user's session. That means if the page is refreshed, the user should not need to enter an email and password to log in again.

- Date(s) of meeting(s) with Client during the current milestone:
    - See Faculty Advisor Feedback below

- Client feedback on the current milestone:
    - See Faculty Advisor Feedback below
- Date(s) of meeting(s) with Faculty Advisor during the current milestone: …
    - Nov 25, 2024
- Faculty Advisor feedback on each task for the current Milestone
    - Task 1: Recommended using LLM to help read the uploaded document.
        - Tools mentioned: GROBID and pdfminer
    - Task 2: …
    - Task 3: Suggested using or creating our own version of a RateMyProfessor API. The reason to make our own is because the available API is in Python and our application is using MERN stack. We may be able to invoke the Python API from our Node App, but may get better performance converting the Python API to native Node..
        - https://github.com/tisuela/ratemyprof-api/tree/master
    - Task 4: …
    - Task 5: …
    - Task 6: …
    - Task 7: …

Faculty Advisor Signature: _____ Date: 11-25-2024

1. **Evaluation by Faculty Advisor**
   - **Faculty Advisor: detach and return this page to Dr. Chan (HC 209) or email the scores to pkc@cs.fit.edu**
   - **Score (0-10) for each member: circle a score (or circle two adjacent scores for .25 or write down a real number between 0 and 10)**

Pedro Moura	**0**	**1**	**2**	**3**	**4**	**5**	**5.5**	**6**	**6.5**	**7**	**7.5**	**8**	**8.5**	**9**	**9.5**	**10**
Jordan Synodis	**0**	**1**	**2**	**3**	**4**	**5**	**5.5**	**6**	**6.5**	**7**	**7.5**	**8**	**8.5**	**9**	**9.5**	**10**

Faculty Advisor Signature: _____ **Date:** 11-25-2024