FIT Schedule Planner

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

Date of Meeting with the Client for developing this Plan:
N/a

Goal and Motivation:

Students use PAWS to register for classes in upcoming semesters. They can access CAPP degree evaluations and FIT programs to ensure they take the right courses. The current system works but can be a headache since there is a lot of room for misunderstanding and confusion. That being said, some students take unnecessary classes and have difficulty fitting all their classes together. The FIT Schedule Planner aims to bring everything a student needs into one place with a user-friendly interface to streamline the class registration process.

Approach:

The user will be able to view all available classes and times that fulfill certain credits from their program. From there, they can add classes to their schedule and look for where they have time gaps (a weekly schedule will be depicted as classes are added/removed). When adding classes, the FIT Schedule Planner will check if prerequisites are met, whether or not the class is available (full or offered), and highlight any overlapping classes. Filters will also be available in the search process so students can look for classes by subject, professor, times, course number, credit range, or RateMyProfessor score. Users can also add personal time blocks (e.g., basketball practice) to ensure classes do not overlap with extracurriculars.

The user will be able to view a checklist that depicts their progress toward their program completion. The checklist is generated from the same programs that students on the FIT website can access. The student would be required to load their CAPP Degree Evaluation in order to make use of this feature's main functionality. The checklist will take the student's CAPP Degree Evaluation and compare it to their selected program to determine which credits they have fulfilled and which credits still need some work. It will then post a scrollable list for the student to review.

The user will have access to a simple interface with a consistent layout to minimize confusion. A navigation bar will be located at the top to make it easy to jump in between the schedule planner and the checklist. There will also be a FAQs tab if additional assistance is needed. Each tab will include instructions at the top to inform the student how to navigate and use that tab. All components in the tab will be labeled making it easier for the student to make a connection between the instructions and the content. Pop-ups will appear when necessary to inform the student if an error has occurred and what caused it. Loading bars will appear when necessary to depict that the interface is working rather than frozen (during checklist creation and class search).

Algorithms and Tools:

The FIT Schedule Planner is built using the MERN stack:

- **MongoDB**: Database to store user data, schedules, and course information.
- **Express.js**: Backend framework for building APIs.
- **React**: Frontend library.
- **Node.js**: Server-side JavaScript execution.
- **Axios**: Handles HTTP requests.
- **RateMyProfessor API**: To fetch professor ratings for class filtering.
- **FIT's Online Resources**:
    - Fall Schedule
    - Spring Schedule
    - Degree Programs

Novel Features/Functionalities:

There are several systems that exist that are in use today that do everything the FIT Schedule Planner can do. What makes the FIT Schedule Planner special is how it combines all these systems into one. When registering for classes now, students have to go to three different domains in order to access FIT programs, CAPP degree evaluations, and PAWS class registration. A lot of students use Rate My Professor, so that makes four. With the FIT Schedule Planner, students can access all of this information in one place, which saves them time and energy.

Technical Challenges:

1. **Getting Real-time Data**
    - We don't have a way to get instant access to real-time data, so we are retrieving data using a scraper, which takes time to run. This can cause potential issues such as data becoming outdated or inaccuracies arising during the scraping process.

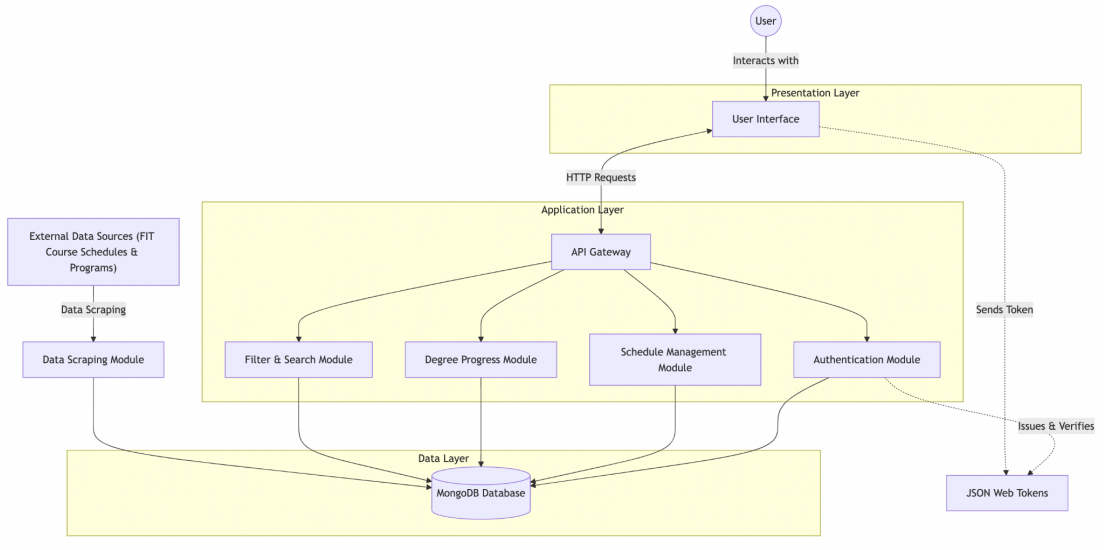
2. **Web Application Development**
    - Our team struggled to transition from making simple GUIs to building an actual web-based platform. More specifically integrating React with backend systems and also managing deployment pipelines.

3. **Dynamic Data Handling**
    - Implementing real-time features, such as detecting schedule conflicts and prerequisites, requires efficient querying and data processing.

Design

System Architecture

The application follows a modular architecture:



- **Frontend (React)**: User interfaces for course search, schedule management, and progress tracking.
- **Backend (Node.js & Express.js)**: APIs for fetching schedules, managing user data, and handling file uploads.
- **Database (MongoDB)**: Stores class data, user schedules, and uploaded evaluations.

Interfacing with Users

- **Students**: Access the web interface to manage schedules and track degree progress.

Evaluation

Success metrics for the FIT Schedule Planner include:

- **Usability**: Measured ease of navigation, and clarity of features, among others, through user surveys.
- **Reliability**: Make sure that the system functions correctly 90% of the time when testing.
- **Accuracy**: Compare/Match/Validate course filtering, prerequisite checks, and schedule conflict detection.
- **Performance**: Measure response times for class searches and checklist generation. We are looking for an average of 2 and 5 seconds, respectively.

Progress Summary

Feature	Completion %	To-dos
Degree Checklist	60%	Fix client-side deployment bugs, incorporate CAPP data
Schedule Conflict Detection	70%	Debug and enhance time conflict resolution
Advanced Filtering	90%	Finalize RateMyProfessor filter integration
Prerequisite Tree Visualization	90%	Improve course selection flow
Weekly Schedule Grid	100%	Completed

Milestone 4:

- Implement, test, and demo loading the CAPP Degree Evaluation
- Implement, test, and demo accessing the program checklist
- Implement test, and demo additional filtering option, address bugs with formatting
- Fix user context

Milestone 5 (Mar 26):

- Implement, test, and demo prerequisite checking system
- Implement, test, and demo schedule updates
- Conduct evaluation and analyze results

Milestone 6 (Apr 21):

- Implement, test, and demo which features/modules
- Test/demo of the entire system
- Conduct evaluation and analyze results
- Create user/developer manual
- Create demo video

Task Matrix for Milestone 4:

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%

- 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. (Dr. Fitz's recommendation).
  - Idea 3:
    - Improve current regex implementation (Dr. Chan's recommendation)

- 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 existing features should not be hard.

- 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.

Approval from Faculty Advisor:

I have discussed this with the team and approved this project plan. I will evaluate the progress and assign a grade for each of the three milestones.

Signature: _____ Date: _____