MacsPlan

Team Dantam:
Taewoo Kim
Nicholas Kaiser
Michael Lesartre
Sebastian Cabrol

6/19/18

Introduction:

A large number of people who attend Colorado School of Mines have the issue that their long term class schedule is not laid out; this is because online flowcharts are often out of date and confusing to read, and minors and ASIs make the complete list of classes a student needs to take difficult to formulate. Because of this students can easily take classes in an undesirable order. This would result in students not being able to take critical classes in order, due to pre-requisites, or classes only being offered one semester. This can result in a student being behind in classes and force them to stay in school for a semester or even a full year, costing them thousands of dollars in tuition and lost revenue from a job when they could have been working up to a year or more prior to when they graduate. Furthermore, this complicated maneuver can be especially difficult for first-generation college students and others who have no prior experience about college from parents or peers.

Our clients for this project are Dr. Christopher Painter-Wakefield and Dr. Neil Dantam. Being advisors to multiple students, a planner to remedy the above mentioned dilemma would conserve time on the part of the advisors and the students.

Dr. Dantam has already provided us with a nascent version for the planner logic which our website would communicate to receive a schedule. Our intended plan was to create a front-end web interface to interact with this “backend planner” residing in the server.

MACSPLAN is a web application that generates a hypothetical schedule for a student based on their major, minor, and choice of electives. The application is also able to display to the user what classes he or she should take each semester in order to graduate based on the number of semester that the user decides. The student will also be able to edit the given schedule to better fit his or her preferences, such as taking a class in a specific semester, and the application will generate a new schedule based on the new constraints.

Requirements:

We had to create a single-page web application with an interface that allows the user to input their chosen major, minor, ASI, and elective choices, as well as other information about their intended plan of study. This other information includes classes already taken either through transfer credits or traditional classes at Mines, as well as the bulletin year, the number of semesters the student intends to finish in, and whether the next semester is spring or fall.

This web application had to be able to take the information gathered by the interface and send it to a backend planner, which had already created by Dr. Dantam, that returns a proposed schedule for the user's remaining time at Mines. This proposed schedule is then displayed to the user on our web application, and has the capability to allow the user to decide that they do not want to take specific classes during certain semesters. After the user has made these changes the web application is then able to send the altered constraints back to the backend planner, and receive a different schedule that will then be displayed to the user again, for their approval or continued alteration.

Because the information given to the website pertains to the student's school record, it is considered private and is protected by FERPA. Our application does not include user authentication, thus, the website is not able to store user data of any kind in a database or other long term storage so as to protect the security of the user's information. However, we still want the student to be able to save their information to reuse it later. To accommodate this, we included an option to export their schedule choices as a JSON file. Instead of having the user log on to access their information, they can export the file, which contains the user's current inputs and constraints. If the user wants to use it again they can import the same file. This import option will automatically fill in the same fields. Thus, all information is stored on the user's machine, and there is no risk of a FERPA violation.

We also, as part of the process of designing the application, gathered feedback, both from users on the usability of the application, and from ourselves about the ease of use and functionality of the backend. We provided feedback on the planner to Dr. Dantam, and integrated users' suggestions into the application.

System Architecture:

Figure 1 illustrates how each component of the system's architecture communicates with each other. The frontend web application is able to take input from the user and an imported JSON file, and send its output to the backend planner or export it to a JSON file. The backend planner is able to take input from the frontend and give its output to the frontend.

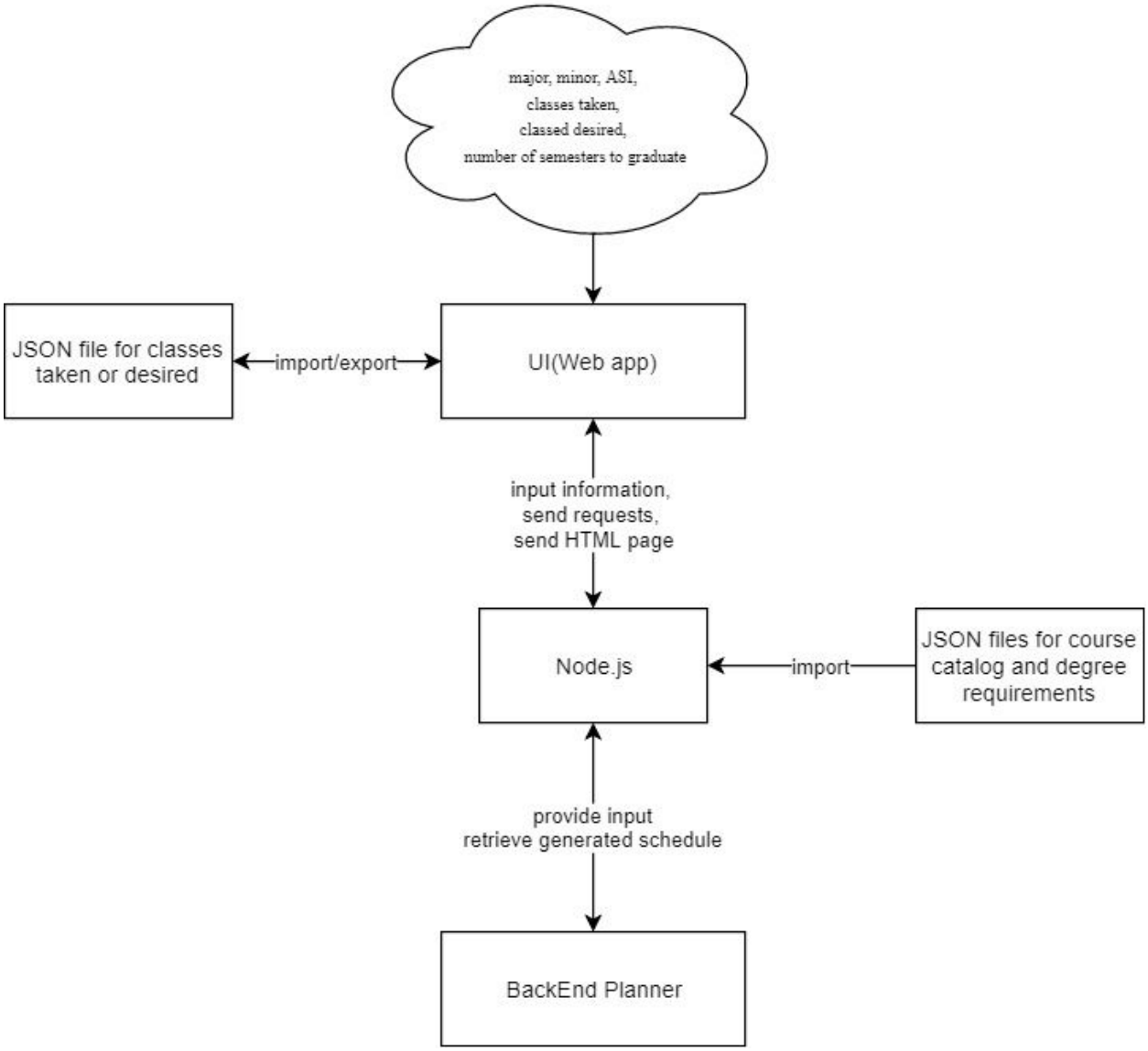

Figure 1.

Components of Architecture:

Front-End Web Application:

This is the web interface that appears when the user logs into “macsplan.mines.edu”. On the website, the user can input his or her bulletin year, major, minor, and ASI. The user can also input the number of semesters that he or she wishes to stay at Mines as well as the reference semester. The reference semester is whether the starting semester is at Fall or Spring as that will affect the consecutive semesters. All the inputs are obtained through the use of datatables, dropdowns and

checkboxes, which in turn show the user what they have input to the website. The import/export functionality at the top of the page will be saving all the user inputs to an external file so that the user can come back to the website and reupload them via import.

Server:

Node.js serves as the connection among the planner, website, and the JSON data files, specific url are linked to each JSON files which are retrieved by the website via ajax request to occupy the data table once the information are inputted. The Node.js also listens to the XML-RPC server which is hosting the backend planner.

Backend Planner:

The backend planner is a prototype source code developed as an extension of Dr. Dantam's research for Robot Motion and Planning. The planner will take in all of the classes that the user has taken, and needs to take, and the course catalog that they need to follow. It will then return a possible schedule that the user can follow.

Technical Design:

One interesting aspect of our design is the lack of a database. As seen in figure 2, there is no sql or nosql database that stores the catalog information. It's innate as an external JSON file, as this makes the application lightweight. Instead of a database, the website is connected to the backend code that runs in the background and provides the logic for the generated schedule.

Our product consists of three main components. The first is the backend planner. Since this was designed by Dr. Dantam, our insight into its design is limited. The second is the server-side code, which we wrote using Node.js. This code is responsible for delivering HTML to the client, as well as communicating with the backend planner using XML-RPC. Finally, the client-side JavaScript and HTML is responsible for providing the user interface and making requests to the server.

Figure 2 shows how each of the three parts of our product work together. After the browser makes a request with user inputs, the server side application returns JSON files for the browser to display on the user's machine. The server-side code is able to then give the backend the formatted information given to it by the client-side code. Then the backend given to us by Dr. Dantam will take that data and make a personalized schedule out of it, and send that data back to the server to be sent to the browser and displayed to the user.

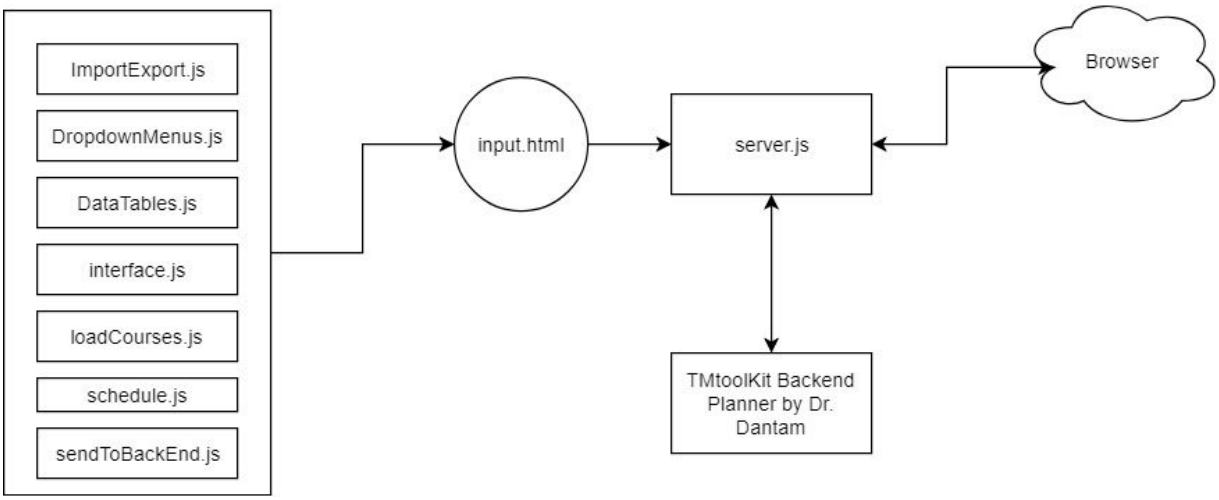

Figure 2.

Figure 3 illustrates how a user would be able to use the web application. First, if the user has already made a schedule, or partially put in his or her information and exported it to a JSON file, the user is able to import that file and the website will automatically load in that data into the appropriate fields so the user doesn't have to. If the user doesn't have a JSON file to import, he or she is able to put in his or her information manually. Either way, the user is then able to generate a schedule, which will display on the website. If the user is happy with the schedule, he or she is able to export the data that generated it into a JSON file for later use. If not, the user is able to make edits to the schedule. For example, one could tell the website that he or she does not want to take a class in a certain semester. The website would then be able to generate a new schedule for the user to see. This loop would continue until the user is satisfied with their generated schedule.

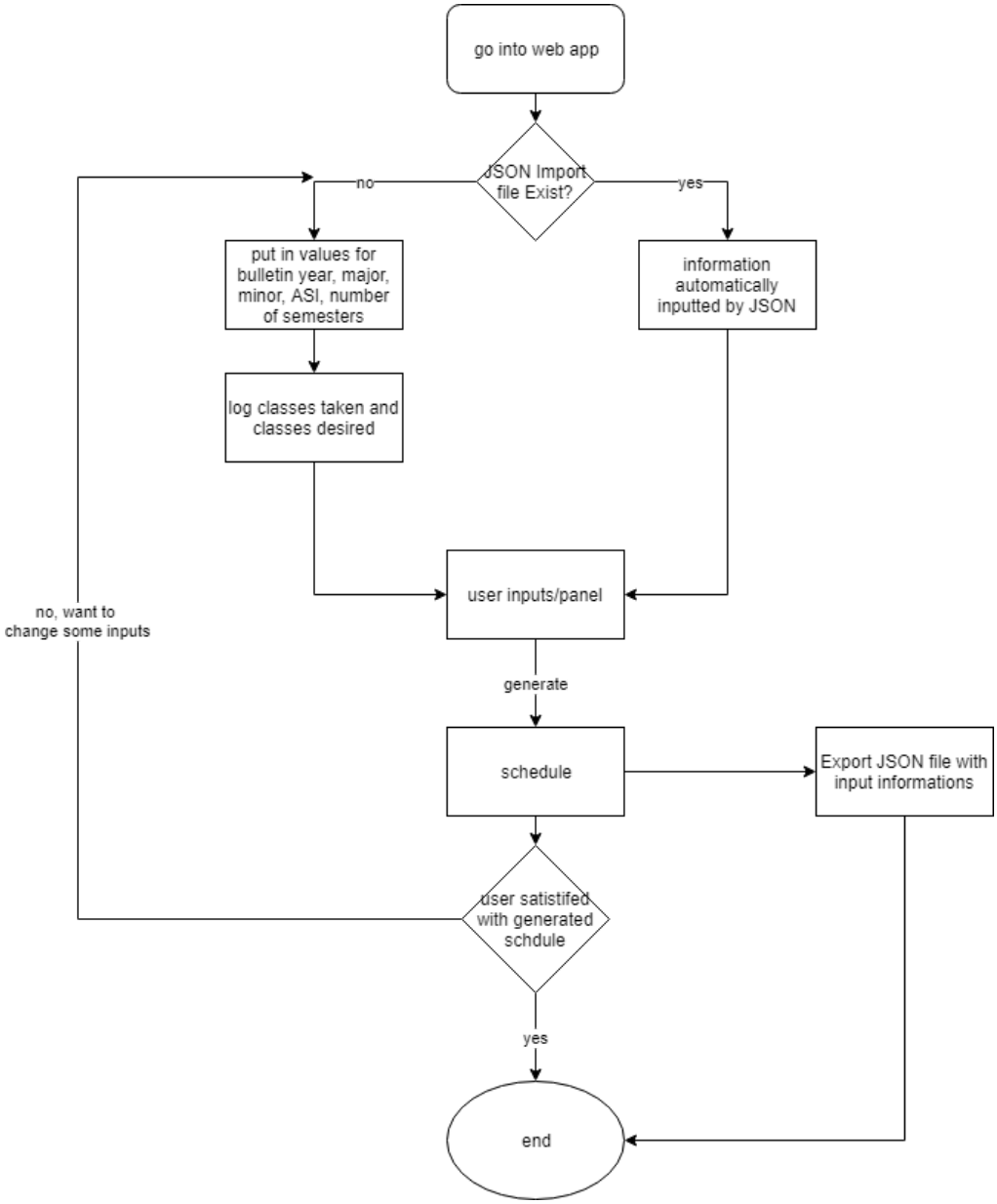
go into web app
JSON Import file Exist?
no
yes
put in values for bulletin year, major, minor, ASI, number of semesters
information automatically inputted by JSON
log classes taken and classes desired
user inputs/panel
no, want to change some inputs
generate
schedule
Export JSON file with input informations
user satistifed with generated schdule
yes
end

Figure 3.

Client-Side Design Decisions:

Dropdown menu selections for Bulletin year, major, minor, ASI, number of semesters, and starting semester:

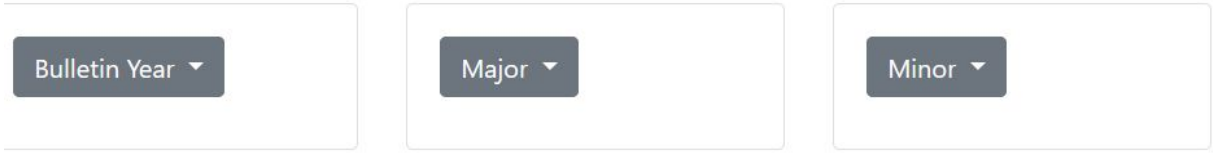

Figure 4.

Limited choices for each category, and thus a mouse click is more convenient than manually typing in and validating the choices for each.

Importing/Exporting json files of saved inputs:

Figure 5.

We cannot save data of the user. If the user wishes to revisit the website, he or she can export the input data, and then later importing it when they come back.

Datatable for course catalog, course desired, and course taken:

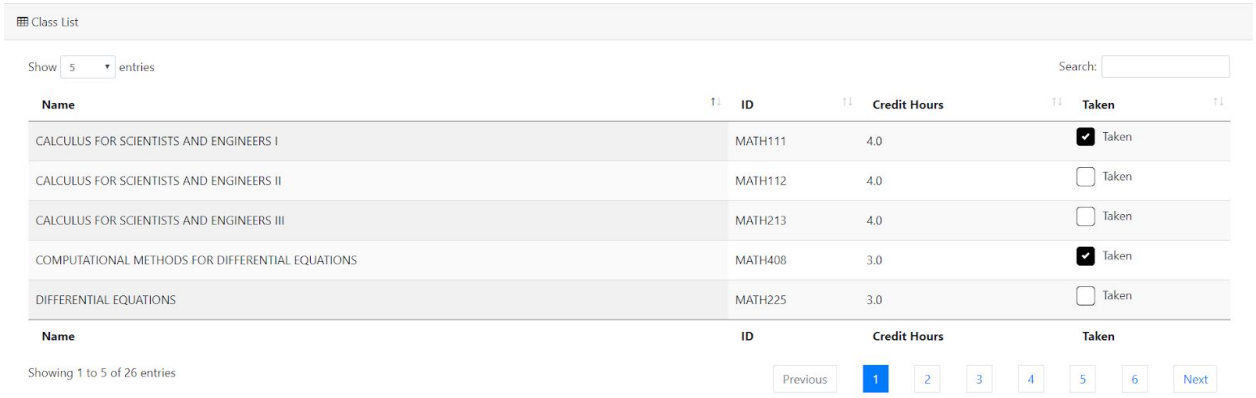

Figure 6.

With the amount of classes that are available, a datatable makes the information condensed and easily navigable for the user.

Schedule Layout:

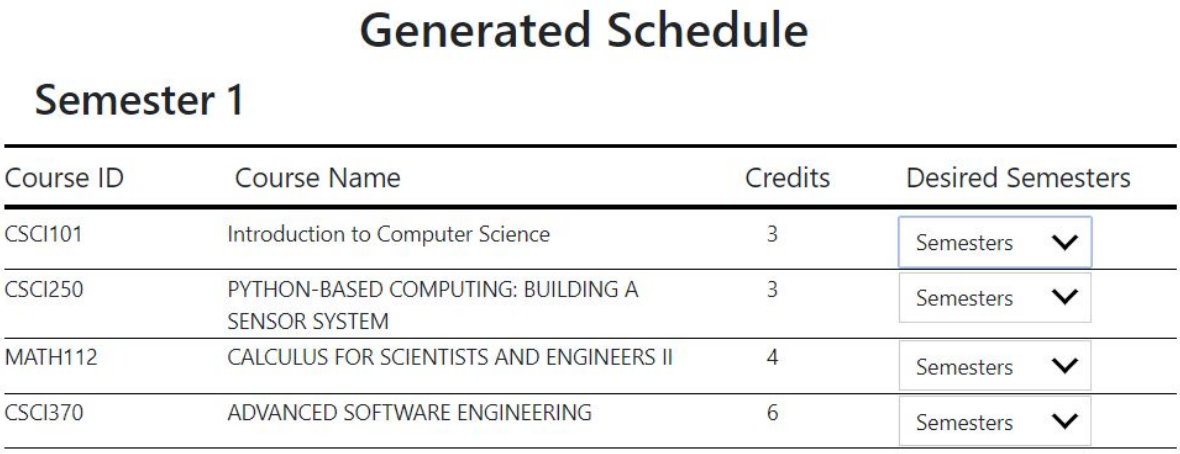

Figure 7.

The layout of the schedule is easy to read for the user. It allows the user to add additional constraints on a generated schedule to move classes to different semesters.

Technology/Framework Decisions:

Bootstrap 4:

- Convenient way to produce an aesthetically pleasing format for the HTML elements using pre-defined CSS classes.

AngularJS:

- Data-binding and client-side request to backend server

NodeJS:

- Due to team's limited knowledge in web development, we wished to unify the front-end and the back-end with JavaScript.

jQuery:

- It's responsible for event handlers and interactions with the websites

DataTable from DataTable.net:

- It provided most of the functionality right out of the box

BeautifulSoup and Python:

- Used for HTML parser to scrape course and degree information from catalog.mines.ed

ForeverJS:

- Npm library that allows to run Node server even with closing out of putty session

Results:

Features that we implemented:

- Generated schedule using these inputs:
- Majors
- Classes taken

- Classes desired
- GUI for all user inputs
- Communication with backend planner
- Website hosted on school server

Features We Did Not Implement:

- Minor/ Asi
- Elective data file (small details)
- Compatibility with edge

Performance Testing:

- Each individual tested own part before merging
- Once all parts were working and merged, more testing on the whole frontend

Summary of Testing:

- console.log();
- A lot of visual testing for frontend
- Made sure everything looked how it should

Usability Tests:

- Had roommates/friends look over front end
- Added some functionality based off of users desires

Future Work:

- Multipass integration
- Use additional information in backend
- More complete dataset (from registrar)
- UI appearance improvements
- More user-defined constraints
- Including summer sessions/classes

Lessons Learned:

- jQuery is a useful tool in JavaScript to interact with the HTML file
- Real-world data is extremely messy and difficult to deal with
- Having backup plans for outside dependencies is a good idea - depending completely on data from the registrar would not have worked
- Meeting more often, even if no work is done at some meetings, can be very beneficial to communication

- Projects will not always use the latest technology, working with a variety of options is important