TEAM APOLLO: Final Report

We, team Apollo, have developed a web application for Math and Computer Science department of Emory University.

Table of Content:

Brief Description of Project

We developed a web-based user-friendly scheduler to allow optimizing class schedules for each semester with no conflicts and while all stakeholder's preference in mind. In order to develop our webpage, we have used mean stack with Mongo database.

Our scheduler has a navigation bar at the top with 4 tabs: Calendar View, Table View, Configuration and Help.

- **Calendar View:** It displays a grid of days by timing for each classroom to be able to add classes. Each individual class has an edit in place functionality which allows her the flexibility to change 4 different attributes of the class: Class Number, Professor Assigned, Days and Time
- **Table View:** It displays a list of all the classes with the professor, room, days and times attached. It can be sorted based on each column as well as exported into an excel file. It has similar edit in place functionality as the calendar display
- **Configuration:** This tab allows Erin or any other user to add classrooms and professors to the database that she can later use to assign to different classes.
- **Help:** The help segment describes the functionality of the web page in detail for the ease of users of the application

Team Organization and Roles:

- Product Lead: Tamer Avci (60%), Ben Chang (20%), Saachi Madan(20%)
- Project Demo Lead: Ben Chang (40%) Tamer Avci (30%), Saachi Madan (30%).
- Front-End Developer: Ben Chang (60%), Tamer Avci (30%), Saachi Madan (10%).
- Database Integration Developer: Ben Chang (90%), Tamer Avci (10%)
- User Stories Lead: Saachi Madan (70%), Tamer Avci (20%), Ben Chang (10%).
- Product Documentation Lead: Saachi Madan (50%), Tamer Avci (30%), Ben Chang (20%).
- Client Communication: Saachi Madan (50%), Tamer Avci (25%), Ben Chang (25%).

Team Communication Methods:

- **Team Communication:**
  - **Messengers:** Our team was in constant communication about our progress, difficulties or gaining opinions through an instant messenger which allowed us to stay constantly in touch and available for each other.
  - **Google Share:** We also communicated some of our ideas and thoughts through google documents during the initial phase of the project. We used it to jot down our ideas and resources for others to share and collaborate
- **Client Communication:**

- **Email:** Our primary method of communication with our client, Erin Nagle, was email. We scheduled our meetings as well as sent updates and follow-up documents back and forth.
- **Sprint Meeting:** We established 4 client face-to-face meeting about 1 hour each. During these meetings, we discussed our progress, likes and dislikes and changes to user requirements.

Team Development Schedule

- **Weekly Meetings:**
  - We scheduled weekly meeting on Thursday at 5:30-6:30pm.
  - Our peer programming sessions were held weekly on Saturday/Sunday 2-4pm.
- **Scrums:** We have bi-weekly scrums on Tuesday and Thursday from 5:00-5:15pm.
- **Sprint:** We scheduled an 2 hour long session discussing the action steps for each sprint and the direction we would head.

Project Management Tools:

- **Trello:** We used trello as our primary project management tool
  - User Stories: Our user stories were converted into requirements which followed the to-do → in-progress → test → completed sections assigned priority levels and checklists.
  - Bug History: We developed a separate dashboard to store our bug history on Trello. We identified bugs in the program and developed a process of eliminating them.
  - User Test Feedback: Through our user test feedback, we developed requirements which followed a similar process to the original user stories to be incorporated in our software.

Updated User Stories

Priority Level	User Story	Derived Requirement	Generation Source	Status
High Priority	Some other professors helping in scheduling prefer list view. It is absolutely necessary for us to have calendar and list view.	**View : Table & Calendar** - Calendar and List View Required	Modified : Client Meeting (Sprint 3)	Accomplished. Different tab for each views.
	She was intrigued with the interpretation and likes the concept of using the	**View : Table & Calendar** **-** Develop the	Modified : Client Meeting (Sprint 3)	Partially Accomplished.

	multiple view of the calendar to represent classes based timing with respect to professor and room**.**	skeleton for class - - - With professor (for validation) - Develop the skeleton for class with room (for planning)		
	Ideally she would love to add a new event by clicking inside the calendar. But she would be okay with having something on the same screen to add a new event. She hates dropdown-lists that she is currently using for adding a new class.	**User Friendly Functionality** - Adding an easy user friendly way to add a new class	Modified : Client Meeting (Sprint 2)	Accomplished
	We need a way to print out these tables. Many professors prefer the table-view and a lot of it is done online but we pass these tables around and use them for multiple functions.	**Functionality** - Export Table View into some type of document that can be edited and pritned	Modified : Client Meeting (Sprint 2)	Accomplished. Exported as csv file.
	Autocomplete needs work. Complete in place and drag and drop	**Functionality** **-** Autocorrect, edit in place are not reflecting on other pages	User Test	Partially Accomplished.
Mediu m Priority	We were unsure about putting restriction on class object (MWF and TuTh). She prefers to have the flexibility to overwrite the restriction and therefore, we decided to look into more options to weigh the pros and cons and come up with a solution	**Validation & Filtration** - Figure out the consequences for setting up the class retractions based on MWF and TuTh · How can the restrictions be overwritten by the user (warnings?) · Derive options	Modified : Client Meeting (Sprint 2)	Partially Accomplished. .

		for the next meeting		
		Functionality: - Sortable function for table in the table view	Modified : Client Meeting (Sprint 3)	Accomplished.
	She did not like the default week, month, year view but she it is not a huge concern.	**View : List & Calendar** - Find a way to, if not delete, hide the unwanted information of the calendar framework.	Modified : Client Meeting (Sprint 2)	Accomplished. Changed the approach to a custom made view rather than using a package.
	New users would find it difficult to navigate through. I like the navigation bar but it might become a little complicated to understand the entire new system	**User Friendly Navigation:** **-** Find a way to make it easier for the user to adapt to the new software.	User Test	Accomplished. Additional of the help section will help the user navigate better
	Each class is really big and we can't see the entire schedule in 1 page. The scrolling is a little annoying as you can't see the entire schedule at once. The white spaces in each class are unnecessary and waste space.	**View : List & Calendar** - add a + sign or something to expand the columns, make the class smaller or pop up when clicked	User Test	Future Steps. Not accomplished due to time restrictions
	Color: Black on grey is a poor choice, font color can be darker, color palette needs work, need warmer colorsV	**Visual Appeal** - change the color scheme to warmer colors	User Test	Accomplished. Minor color scheme changes.
	Can't read the font, too much white space. We present this on projectors so people should be able to read the text in the meeting.	**Visual Appeal** - Text too small, attendees in the meeting can't read information of the presentation	User Test	Accomplished. Increased text size.

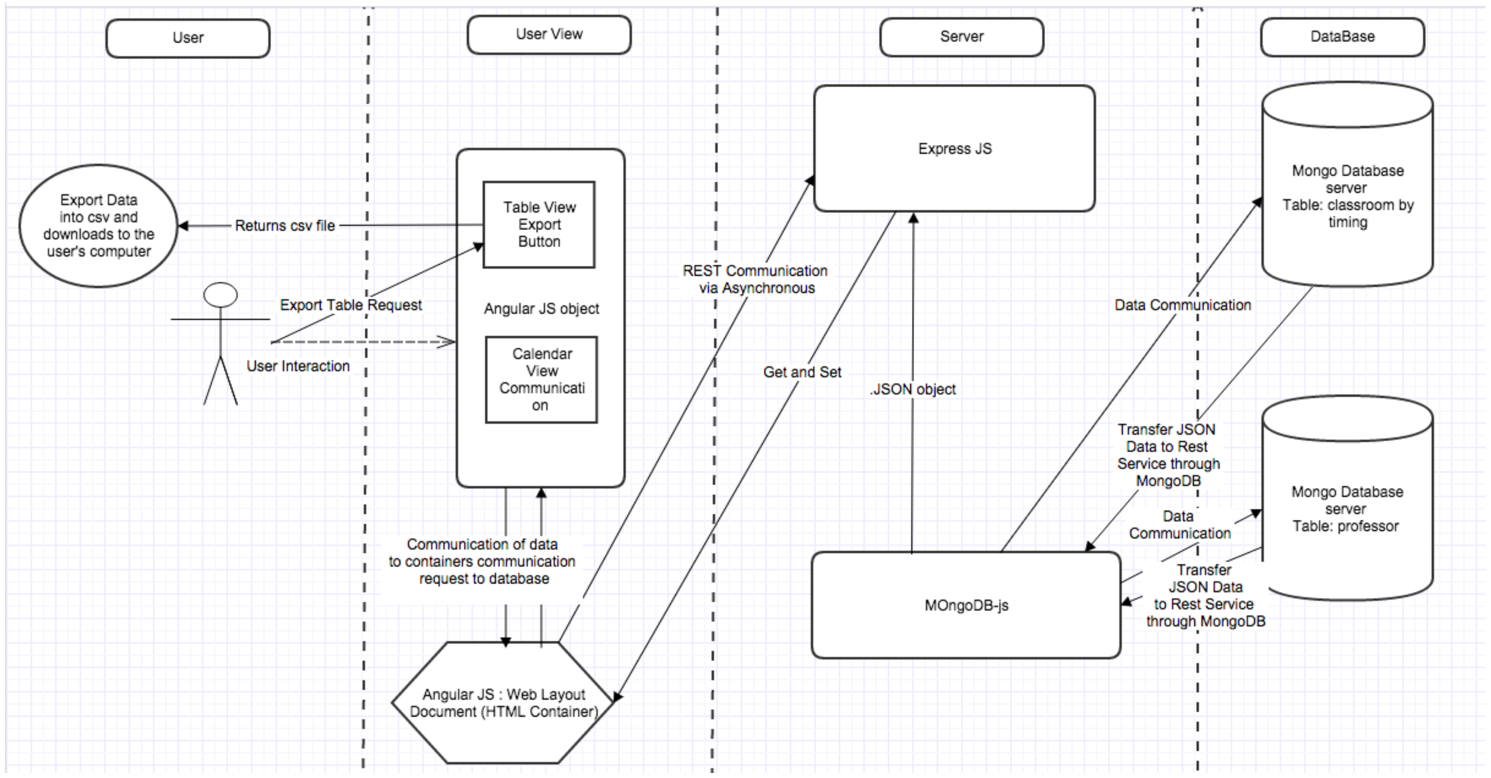
	Apollo logo too big	**Visual Appeal** - The sign distracts from the purpose	User Test	Accomplished. Apolo logo's size decreased
	Lines are not aligned, Add rooms and add teachers are very close to each other. The table view adding things seems chunky and confusing. Which add function is for which list?	**Visual Appeal** - Format the page better. Align things at different page sizes.	User Test	Accomplished. Alignment changed and margin & padding added
Low Priority	I don't even know what the color-coding means exactly! It is not very functional or efficient. An ugly yellow block of scroll bar appears which does not exactly line with the times. It becomes so difficult to make sense of the scheduler	**Color Coding** - Develop and propose multiple systems of color coding and allowing that would fit the clients intuitive process and help her vizualize the schedule.	Initial Client Meeting	Future Steps. Not accomplished due to time restrictions
	There needs to be better way to display the lab sessions.	**Filtrations System** **-** Checkbox at the bottom? For labs, class levels (cores, electives)	Initial Client Meeting	Future Steps. Not Accomplished due to time restrictions
	Change the list view in the navigator bar to table view. 2 names for the same thing is confusing. Is list view different from tableview?	**Visual Display:** **-** Consistency in naming convention	User Testing	Accomplished. Inconsistent words changes to match table view.

Database:

- We use MongoDB, a NoSQL database to store our data. NoSQL databases are more flexible with structures of the objects that are stored which is particularly useful for us because we were not sure what our final data schema would look like.
- The database holds three collections, two to store classroom and professor autocomplete data, and one to store our calendar blocks for the semester.

- Our auto-complete data simply holds documents that have a key value pair. The key being classroom and professor respectively, and then the respective value. The calendar block is a single document that stores data under the blocks key. The value is an array that holds data on professor, classroom, class name, time, and days.
- The way we have it set up is easy to expand on because we would simply store a semester ID key with each document. This would be filtered when we make API requests to retrieve the proper data.
- Example: db.Apollo.calendar.find() -- [blocks:[{professor:"Nagy", classroom:"E201", className:"Math315", time:"11:30am", days:"MW"}]]

Software Architecture:


User
User View
Server
DataBase
Express JS
Mongo Database server Table: classroom by timing
Export Data into csv and downloads to the user's computer
Returns csv file
Table View Export Button
Export Table Request
Angular JS object
User Interaction
Calendar View Communication
REST Communication via Asynchronous
Data Communication
Get and Set
.JSON object
Transfer JSON Data to Rest Service through MongoDB
Mongo Database server Table: professor
Data Communication
Communication of data to containers communication request to database
Transfer JSON Data to Rest Service through MongoDB
MOngoDB-js
Angular JS : Web Layout Document (HTML Container)

Testing Reports

To develop a successful error free project, we have used the following testing methods:

- **Verification and Validation Testing:**
  a. During each Sprint, we perform an informal Verification and Validation Test. In the verification we ensure that the software implemented is correctly working and there are no unidentified bugs.
  b. For validating our process, we reframe our previous requirements based on the communication with our clients and add the new ones to the list.

    c. We develop tasks or functions needed to be implemented through this requirement list. Therefore, at the start of each new sprint, we check if all our functionality in the software is traceable to the requirements.

- **Peer Testing:**
    a. Pair Programming
        i. We used the pair programming method to develop a large part of our code. We sit together and discussed how we want the code to look. Then one of the pair develops the code while the other one looks over their shoulder and visualizes the code.
        ii. After the code is developed, the observer questions the peer about the code and points out parts where they might have interpreted the solution differently and discuss the options.
        iii. This allows us to eliminate minor bugs that might have been difficult for us to catch later. It also ensures that all of us understand and execute the requirements similarly.
    b. Peer Review
        i. If we have not developed the code together and one of the developers has completed the code for a requirement and moves it into the test phase in our product management tool (trello), it is the responsibility of the other 2 programmers to go and test that code for any errors.
        ii. If they find an error, the bug is stored in a bug history board which either gets resolved or the remains in the bug history. Once the bug is resolved it is moved in the test phase of the bug history board. Then the other developers test the bug again and move it to complete or edit the bug and start over.

- **Regression Testing:**
    a. After completing a big functionality or many related requirements we test the entire code on basic manual testing to ensure we have not broken something. These testing are generally done upon finishing of the entire calendar view, list view and/or configuration to ensure the necessary changes are made across the board and there is the correct communication between the pages.
    b. We also do a regression manual testing of the complete software when we have finished a sprint and start the next one.

- **User Testing**:
    a. We developed a demo to display our product to 20 users and get their feedback on the software. We had 4 questions targeting different category of user interactions. The following are the results of the user testing based on their categories.
        i. User Friendly Navigation
            1. Overall 75% satisfaction
            2. Positive feedback: “love the navigation through the tabs” , “almost there”, “simple and straightforward navigation”
            3.

Negative Feedback	Change implemented
chunky to add things	Yes, additions are made to test through.
New users would find it difficult to navigate through	Added a help section to help the new users navigate.

ii. Data Represented
1. Overall 82% satisfaction
2. Positive Feedback: “love the data assortment that allows user to dictate how they want to organize their information”
3.

Negative Feedback	Change implemented
Naming: list and table view is inconsistent	Naming has been made consistent
Show more data without scrolling, make the boxes smaller	Currently out of scope due to the shortage of time. But definitely a good next step.

iii. Q3: Visual Appeal
1. Overall 65% satisfaction
2. Positive feedback : Clean distinctions
3.

Negative Feedback	Change implemented
Color: Black on grey is a poor choice, font color can be darker, color palette needs work, need warmer colors	In progress, changing the color palette to be warmer.
Can’t read the font, too much white space	In progress, working on increasing the font size.
Apollo logo too big	Completed, decreased the size of logo
Lines are not aligned, Add rooms and add teachers are very close to each other	Completed, lines are aligned and the add tag is on the same line as the text fields.

iv. Q4: Flexibility - 81%
1. Positive Feedback:

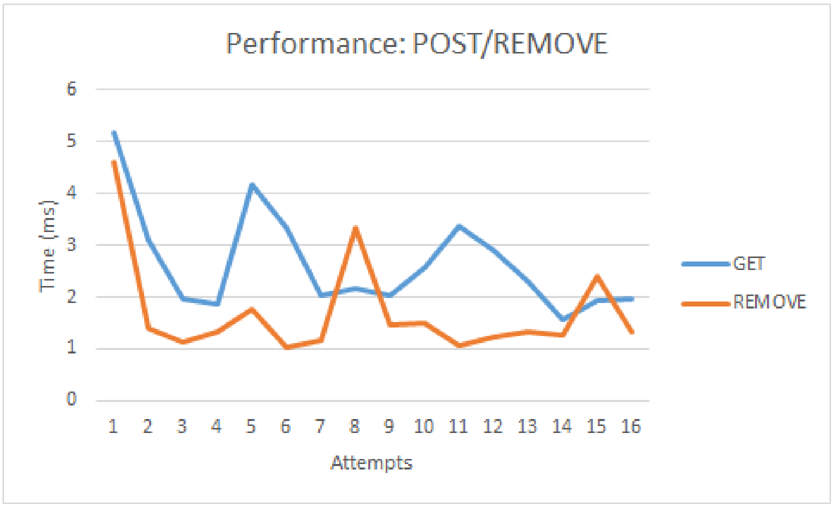
2.

Negative Feedback	Change implemented
Autocomplete needs work. Complete in place and drag and drop	Implementation is in progress
add a + sign or something to expand the columns, make them smaller or pop up	Currently, out of scope due to time restrictions but definitely the next step we would be on if we had time.

b. We also got feedback about their score, what they like and what they hate in the program. We used these feedback to create user requirements.
c. As a group we have discussed these requirements and taken them into consideration. These requirements have been given priority levels. Most of these requirements have been executed and some remain.

- **Hallway Testing:**
  a. We asked random non-technical users to try using our program
  b. Based on feedback, we moved certain buttons to different locations

- **Performance Testing:**
  a. The only performance metric we tested was the time it takes to load the application and data load at server/transaction.
  b. Because we haven't hard coded anything on the frontend, time was quite satisfactory.
  c. Pages that you can navigate through the navigation bar are embedded HTMLs inside a landing HTML, so they do not refresh or load another HTML within the index.html
  d. Results:
     i. Performance Testing Loading Time: P(N) = N x T x D
        N # of concurrent users, T # of transactions/unit time, D data load at server/transaction
     ii. Given N=1, T=1, Test sample size: 24 Transaction type: ADD/POST
        Mean time for P(1) = **2.68ms** with a std.dev = **0.23ms**
     iii. Given N=1, T=1, Test sample size: 16 Transaction type: GET/REMOVE
        Mean time for P(1) = **1.71ms** with a std.dev = **0.24ms**



- **Integration Testing:**
  a. We use Mocha (Javascript testing framework) to run api requests to test full integration pipeline
  b. Check to make sure all our response codes at 200
  c. Integration tests ensure that our code still works even with small modifications

Client Documentation Report:

- **Description of platform, software need to run it**
  a. We are using a full stack application framework called MEAN. That is, MongoDB + ExpressJS + AngularJS + NodeJS.
  b. Node.js runs on Google's V8 runtime engine which is written entirely in C code leading to high speed and non-clocking code. This has to be downloaded before running our app. It handles any gaps in communication between our other tools.It also handles the packages that we use to run our projects.
  c. ExpressJS handles routing, requests, and API endpoints so that our database can talk to our frontend with a layer of security.
  d. AngularJS is a robust frontend framework that allows us to modularize our code. We separate our code into MVC or model, views, and controllers. That way our code is easily maintanable and our code is easily reusable.
  e. MongoDB is a NoSQL database. It is performance-driven model that is extremely scalable and can easily cope with large volumes of structured or unstructured data. This also needs to be downloaded and run before npm start.

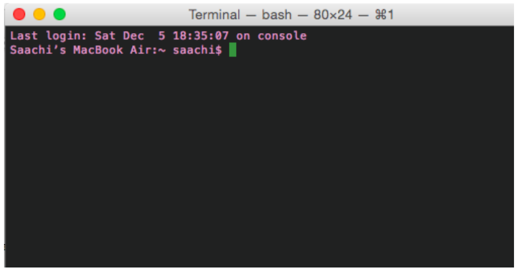
- **List of installation steps[1]:** These steps will have to be followed once for every computer you install and plan to run the software on currently. We would recommend that you have an technical expert from Emory Technology Service/ Our Team help you with these steps on any future computer.
  a. Downloading MongoDB: This will be your database that stores all the information which can be retrieved anytime.
    - You need to download a version that is compatible to your computer from the link MongoBD (https://www.mongodb.org/downloads?_ga=1.110079948.896005422.1445371742#production)
    - Once you read the page, select your operating system type (OS or windows) and the version from the drop down list (the first entry on the list should be fine). Then click download on the upper right corner of the page.
    - This will download a zip file on your computer. Double click it and your download is done.

  b. Download Node (https://nodejs.org/en/download/)
    - You will need to download a version that is compatible to your computer from the link above.

[1] These are steps to install the software on the computer and using it locally. We are in touch with Emory to have the software run server which will make it easier/more feasible for the client to implement and use.

- Choose your operating system and select your system architecture 32-bit or 64-bit.
- Once you download the file, double click and follow the instructions to install.

c. Start a terminal: Terminal is an interface you often use to execute some commands you can't access by opening folders.

- To open terminal on a mac
  - Click on the search button
  - type terminal and press enter.
  - A screen similar to the one below should appear.
  - 

```
Terminal — bash — 80×24 — ⌘1
Last login: Sat Dec  5 18:35:07 on console
Saachi's MacBook Air:~ saachi$
```

- To open terminal on a windows:
  - Search command prompt in the start menu.

d. Downloading GIT: Git is a software we used to collaborate and work together on the program. All are files are stored on this page. Therefore, in order to download our code and run it on your computer, you need to download git

- https://git-scm.com/downloads is a link where you can download git.

e. Clone the repository: Our repository is saved on git and after you download it, go on the terminal and type the following command:

- git clone https://github.com/EmorySE/Apollo.git
- This command allows you to copy all of our source code into your local computer.

f. `npm install bower -g` in the root directory (in the same directory we cloned our git)

g. `npm install` in the root directory

h. `bower install` in the public directory

i. Run mongod.exe from the directory you have installed the MongoDB.

j. `npm start` in the root directory to serve the code to localhost:3000

- **Running steps**

  a. After npm start, the software should initialize and the database should populate our calendar.

  b. Simply open up a browser (preferably Google Chrome as the testing is done using Chrome)

  c. Go to localhost:3000

  d. Any changes you make will be saved automatically to our database.

  e. Next time you like to launch the application, open up a terminal and go to your root directory for Apollo.

   f. **npm start and repeat the steps a through e.**

- In order for your interaction with the software, we have included a help section in the software itself. This help section has any question you would have during your normal interaction with the software.

Read Me

Our product features a navigation bar which has 3 sections : Calendar View, List View, Configuration and Help. Below we have attached explanation as well as common questions you would have while using this product. These questions are divided by each view. These questions and their answers can also be found on the help tab.

1. How do I get started?
   a. The navigation bar on top helps you look at the schedule in different ways as well as add classes and professors. All of this information is stored on the backend.
2. Will I lose my work if I close the tab? Will I return to the same page?
   a. We would hate for you to lose any work / forget where you were if you got interrupted suddenly and closed your tab. Therefore, we have a cookie system which will always get you to the same page that you left.
3. Scheduling for the next semester, do I have to start over?
   a. You don't need to start over if you don't wish to. You can go into the configuration page and delete/ add the classrooms and professors. Move to the calendar/list view and start making changes to the schedule.

Configuration View:

In the configuration view, you can add/delete classes or professors to your database which are also used as data for the autocomplete in the calendar and the list view. Simply type in the data and press the check button to add. You can go down the list to delete desired entries.

1. How and when do I add a Professor?
   a. To add a professor enter their name and press Add. You only need to do this once you have a new professor. The professor will then be visible in autocompletes in the other tabs
2. How and when do I add a Classroom?
   a. Everytime you have access to a new classroom to schedule classes you can add them here. You only need to add it once and it'll be available on all the other pages.

3. How do I delete something?
   a. To delete a professor or classroom simply click the remove button next to them. Once you delete a professor or classroom any classes assigned to them will change to their default values.
4. If I mistyped a name, do I remove and re-add it?
   a. You can certainly do it that way but you could also double click on the name of the professor or the class code and edit it in place. Once you have edited it click on the check to save to edit or the "X" to delete the edit.

Calendar View:

Calendar view allows the user to see the classes according to 2 types, by a certain classroom or by days. To select the view, you click on the Calendar in the navigation by and select by days or classrooms from the from down list. In the classroom view, you can select the desired classroom from the drop down list on the top left corner of the grid. The grid would show the classes scheduled in that room with days as the column and times as the rooms. In the days view, you can select the day you wish to view from the drop down list to the top left corner of the grid, and you'll be able to view all the scheduled classes with the rooms as the columns and time as the rows. In each of the views, you can edit, delete or add each class.

4. How do I add a class in the Calendar?
    a. In either the calendar view by classroom or by classrooms, double anywhere in the calendar grid to add a class. It will add the class with default values: Professor for Professor Name, Class for Class Name, M/T/W/Th/F for days depending on where you clicked and Time of slot for Time of the class.
5. How do I change the default values after adding the Class?
    a. To change the values of anything (Professor name, class number, room number, Days, Times) in a class cell just double click inside the cell and then edit. After you are satisfied with the change just click Enter to save and view your changes.
6. I made some change to a class and it disappear. I need to change something else too. I made the change by mistake. What do I do?
    a. The class would only disappear if you change the room, time or days of that class. As soon as you make the change, the class would go to the appropriate slot, as it no longer belongs in the same slot. If you remember the change you can go there and make further changes. If you don't remember the change you can go to the list view, find the entry and make the appropriate changes there.
7. How do I add a class?
    a. If you want a class to represent in multiple cells, say the class CS170 taught by professor Valerie Summet happens every Monday, Wednesday, Friday in room W201 at 1:00 am. Follow the steps below:
        i. Click anywhere inside the grid on calendar view you will see a light blue box with default values appear.
        ii. Double click the box to enter in the edit mode of the box
        iii. Change professor to Valerie Summet, Class to CS170
        iv. Change the time to 1:00pm
        v. Change the days to MWF (With or without space)
8. What do I type for multiple day class?
    a. To represent different days of the week type:
        i. M or Monday for Monday
        ii. T or Tuesday for Tuesday
        iii. W or Wednesday for Wednesday
        iv. Th or Thursday for Thursday
        v. F or Friday for Friday
    b. You can type a combination of these without space to represent multiple days. For example, you can type, M Thursday or MTh for Monday and Thursday classes.

List View:

List view shows you a the list of all the classes that have been added either in the list or in the calendar. The list view is sortable by each of the columns. You can also all a class by clicking on "+" button on the bottom of the page or delete it by clicking on the "X" at the end of each row. You can double click on each row and edit it then click check button to save or "x" button undo the change. The list is also downloadable to your computer in a csv document by clicking the export button.

1. How do I edit the item in the list?
    a. Double click on the item you wish to edit. Make the necessary changes. Then click on the check button if you want to save the changes and the "X" button if you want to delete the changes.
2. How do I add a row?
    a. You can add a row by clicking the "+ Add" sign at the bottom of the table.
3. How do I delete a row?
    a. To delete a row, click on the "X" button on the right corner of the row.
4. Can I sort the list?
    a. Yes, you can sort the list based on any of the columns.
    b. To the rows in ascending order based on a column click on the heading of that column.
    c. To the rows in descending order based on a column click twice on the heading of that column
5. How can I download the file?
    a. The export button next to the list view title allows you to download your list which you can open in an excel file

Future Action Steps:

Due to the limited duration of project duration we were unable to incorporate all the features that we believe would be implemented in an ideal project. Therefore, we made a conscious decision to leave out certain features in to complete the essential features in time to deliver a usable version of the product. We have listed the features we suggest to add if the product is expanded in the future.

- Optimized saving mechanisms (not deleting then creating, use update instead)
    - Being able to update the professor names and classroom number is not a feature we currently have in the configuration page. If you misspell a name, you have to delete it from the list (which deletes it from the database) and add the correct name again.
- Add login page
    - Currently there is no login page or security/ password protection access to the page and hence the database. Anyone with the server url can access it and use any feature they desire. If the project is used for scheduling classes, there will be a need to add a password protected entry point with admin id

and passwords which will have to be taken into account to access the database as well.

- Add security for APIs
    - The APIs we are using also don't have any security element. As mentioned before, with an addition to the login page, there needs to be a layer of passwords(validations) and security attached to the project to be usable.
- Separate data into semesters so db does not need to be reset every semester
    - We would like to save multiple views for each semester as well as plan more than 1 semester in advance. Therefore, a feature that accounts for saving data and viewing it based on semesters would be a great help to the clients as they currently scheduled 2 semesters in advance.
- Accommodations for labs and other unstructured schedule times
    - Special accommodations have to be made for labs and other unscheduled classes. These classes are shorter duration and are attached to other classes and therefore there can be potential scheduling conflicts.
- Implement JS Minification to optimize
    - JS minification compresses our files so that they are on a single line. There are no spaces or return statements that are stored. These are only applied to deployed files, there are still files that are not compressed so that these files are still editable.
    - JS minification package tools exists in npm so expansion would be as simple as adding a dependency.
- Calendar validations
    - Calendar validation does not complete work currently. Our validation only works if you enter an incorrect start time and classrooms. In the future we would like to add validation for conflicts with professors assigned to classes.
- Flexible time intervals
    - We made the time intervals to represent the 75 minute classes. In the future, it would be more convenient for the client to be able to add classes that are of different time intervals too.
- Allow reordering of classrooms
    - Currently the classroom are fixed in the column based on which classroom

was added first in the configuration page on the Calendar view by days. We made an assumption, to simplify the project, that the client would add the rooms they use more frequently first. However, in the future it would be great if they could reorder classrooms in the list.

- View by time patterns than simple days
    - It would be easier to be able to classes that are MW, MWF, TTh, F or classes with lab patterns rather than being able to view classes only by individual days. We ran out of time before being able to this feature. We believe this would ease the process further.
- Using color in classes scheduling
    - We thought about adding color code to classes based on where they were in the scheduling process. If they were certained, tentative or placeholders. We could have different colors representing that.
- Room and Class capacity validation
    - We thought about adding room and class capacity to each room and class. This would add an additional level of validation so that if a classroom can not take the amount of students planned for a class, it would throw an error.
- Class Level Optimization
    - We would also like to classify classes based on core or electives and different levels as we can add 1 class and it's pre-requisite at the same time in different classrooms as the same student would not be interested in both classes. We would want to minimize the conflict, for example, between 2 electives offered at the same time. In order to minimize time conflicts students and giving them the option to take their desired classes easily would be another great addition.