11/24/2021

Q-Cor

Web-based Scheduling App

Kushagr Bhatnagar, Brooklyn Coulson, Vaughn Geber
TEAM Q – ENSE 374

Table of Contents

Q-Cor: An Online Scheduling Experience Reimagined

As a student, navigating everything post secondary education has to offer can be a thrilling, yet harrowing experience; especially for those who are new to campus. The initial excitement of joining interesting clubs, making friends, and discovering new passions tends to wear off quickly once the due dates begin to roll in. Managing multiple classes and group projects can quickly become overwhelming when trying to organize due dates for assignments, exam dates, deadlines for project checkpoints, meetings for group projects, and navigating availabilities for coordinating group check-ins. With the Q-Cor online scheduling application, managing daunting course loads has never been easier. This online scheduling app is designed in a way where the student experience comes first.

What is Q-Cor?

Q-Cor is an online scheduling application that can be conveniently accessed by any device housing an internet connection. Users of this application will be able to invite other classmates/group members to view shared calendars. Those invited will be able to edit the calendar in a multitude of ways: adding dates, removing dates, amending pre-existing items and sorting items in terms of importance. There will also be an in-app messaging function that will allow easy communication between members within the same calendar to easily converse on topics like coordinating study sessions, course content discussions, and verify/clarify any assignment/exam requirements. By supporting this in-app chat function, students will no longer need a separate application in order to stay in touch and communicate with classmates and group members.

Planning Phase

As with any project out in industry, it is very important to properly follow a process in order to ensure the best outcome possible with the resources provided. The first phase in any project should be the planning phase; during this phase there should be nothing but planning, documentation and communication on exactly what it is the project should entail/accomplish. The approach Team Q took to accomplishing this project was one intertwined with concepts from Agile including monthly scrums with Dr. Tim Macaig to ensure we stay on track, as well as a Kanban board executed through GitHub to organize and delegate tasks amongst team members. However, the documents we used throughout the project reflect more of a waterfall approach to project management. On top of all of this, our main design architecture for the code deliverables was a Model View Controller approach with defined Minimal Viable Products to be released as the project progressed.

Initial Project Ideas: Business Case Document

As a group we bounced suggestions around, and finally decided on the idea of an online scheduling application. This was something we could all see ourselves using since we have all struggled at one point during our university career with finding a method that works for keeping us organized. After we agreed on the business opportunity, we tried to brainstorm a couple different options to choose from. We knew we wanted to end up with a product that would have your basic features like viewing a traditional calendar with important events marked on their respective dates. This view would also include all the basic functionality required to alter the calendar including adding, removing, and amending pre-existing dates. Another feature we wanted to include in regards to the calendar itself was to add the opportunity for users to prioritize and organize the events if they so choose. To make us different, we thought of including some kind of a group function to cater to university group projects, or for a class setting where several people can have access to the same calendar and date editing. Another aspect of the group functionality that we wanted to include was to have in-app messaging and chat abilities. Most scheduling apps already out on the market lack this function, and we figured it would make communication easier since students wouldn't have to have another application in order to coordinate meetings with one another and to discuss course topics.

With all of this in mind, we came up with two different approaches to this project. The first was to focus on the individual user first by ensuring the calendar is working in its entirety prior to attempting the group activities. Once the fully functioning calendar is developed, then we would explore the ability to share a single calendar to other users, incorporating the in-app messaging, and to eventually add the option for a user to have multiple calendars in order to have a separation of concerns. The second approach was similar to the first except the roles are reversed a little bit. Essentially with the second option, we would make a very basic calendar with limited functionality and then focus on the group activities. Once the sharing of calendars and chat was developed, we would then return to the calendar and finish up the extras like organization and prioritization of events within the calendar as well as having multiple calendars.

Now, with a traditional business case, there is usually a cost-benefit analysis associated with the options previously detailed. However, for this particular project we struggled to come up with any costs since this is a school setting, so the only real cost we could think of related to time (which is constant regardless on which option was chosen). Instead, we thought it would be more beneficial to highlight and focus on the known constraints or risks associated with the project and to factor in all the assumptions given by Dr. Macaig. We reviewed the requirements, and felt that the known constraints included our time frame to complete the project (a little over three months), having an easy-to-read calendar view that isn't too congested, being able to differentiate what is necessary in terms of functionality (needs vs. wants), and to be aware that the chat function could become complex. For assumptions it was noted that we must have at least 1 Minimal Viable Product produced within our timeframe rather than a fully functioning application. It was also assumed that we would focus on the process and motions of the project rather than just focussing on the materialistic outcome. After considering everything mentioned previously, and with some discussion, we ultimately chose to recommend our first option; we would complete the calendar and its functions prior to moving on to the group/sharing activities.

Approving Our Recommendation: Project Charter Document

Once we presented our idea, Dr. Macaig signed off on it and we were able to begin. Through the project charter, we were able to further discuss our imagined outcome for Q-Cor in a more professional manner. We reiterated our project goals that were outlined within the Business Case, and we outlined what our objectives were. These mainly highlighted the means to our envisioned goals which includes proper documentation, sketches of the user interface for all models, and storyboards explaining how a user may interact with our website. We also had obvious objectives like front-end coding, back-end coding, and the testing as we develop to go along with our code. In terms of a project budget, we weren't planning on spending any money since we weren't required to due to this being a course project. We instead tried to come up with some costs that weren't to do with money; we ultimately came up with having a limited team size of three, and a limited timeframe of 74 days to complete at least one deliverable.

We knew it was fair to have Dr. Macaig as our project sponsor and project manager since he was the one to sign off on our idea, as well as he hosted scrums throughout the semester to see our progress. With that being said, we actually struggled on trying to figure out who our additional key project stakeholders would be. We tried to think of who would be interested in our project that would also have some knowledge that could be incorporated into our project. Naturally, Dr. Macaig was a valid choice here (even though he's also considered a sponsor and the manager), and we also decided to include our lab instructor Adam Tilson since he gave us the knowledge and exposure to some concepts that proved to be helpful when we got to the coding stage.

For the overall project milestones, we only found it fitting to include all our due dates highlighted within UR Courses in relating to our project from start to finish. This included the date our group was assigned, our final presentation, and every scrum check in, project activity, and self/peer evaluation in between. Lastly, we reiterated some of our project risks from the Business Case (wants vs. needs, concerns of team experience and the use of skills, etc.), as well as adding some new ones including team member availability for check-ins and mismanagement of time.

Reviewing Our Target Audience: Stakeholder Register

Before moving onto our project scope and officially determining out Minimum Viable Products, we wanted to review who our target audience will be. We hoped that once we did this then it will become easier to not have scope creep, it would allow us to be more focussed on customer needs rather than developer wants, and we would in turn have better deliverables in the long run. However, this was easier said than done. I personally believe we were overthinking what constitutes a proper stakeholder, and who that may be within our project setting. After one of our scrum sessions (and after overhearing some other groups with similar concerns) we decided to include people who may be interested in our application, and who would actually want to use it. Based off of this angle, we chose to include not only post secondary students, but professors to utilize within their classrooms, student unions to distribute as a service for universities, and even corporations that use group collaboration on a daily basis.

Organizing Our Goals: Project Scope Statement

After reviewing our target audience, we were more comfortable with our project goals and this strengthened our reasoning as to why we chose to include our previously defined functionality. Our next step was to organize our project goals into Minimal Viable Products. Initially, within our Project Scope Statement we included: calendar displaying events/dates, calendar manipulation (add, remove, amend dates), better organization of calendars and their items, inviting others to specific calendars, in-app messaging, and supporting the separation of shared/personal calendars. We were then going to have everything up to and including better organization in our first MVP, and the rest in our second MVP. At the end of the semester, we revisited our MVP's and our progress and agreed to move what wasn't completed into the Project Exclusions section. We excluded the inviting others to calendars, messaging, and separating calendars not because they weren't relevant to our customers expectations with our product, but more so because we had moved them to future MVP's and we had not completed them.

Finalizing Expectations: Envisioned and Rationalized MVP's

When we began our project, we took a look at our Project Scope Statement and divided the goals into two Minimal Viable Products. The first one reflected our desire to work on the individual experience by focusing on displaying the events within a rendered calendar, being able to manipulate the calendar (add, remove, amend events), and to better organize these events based on user defined priority. The second MVP would cater to the social aspect of our website and would include the ability to share the calendar with other users, have the in-app messaging up and running, and to have a separation of concerns relating to personal and shared calendars.

Once we dove into the coding stage, it became apparent that it wouldn't be feasible to complete everything we had wanted, and that our MVPs were a little too large since we weren't as experienced as we had thought. We also failed to include basic necessities like the sign up and login features. As such, we saw it to be more reasonable to add a third MVP, and to shuffle around our goals to make our deliverables more realistic. After moving things around, our first MVP included the development of a sign in and login page, validating the credentials when a user attempts to either sign up or login and allowing access only when they meet criteria, rendering the calendar view, and allowing the user to manipulate the calendar. For our second MVP we moved the organization and prioritization of events from the first MVP into this second one, and it also included the original goal of being able to separate calendars (in prep to allow the actual sharing of calendars). Then for our newly created third MVP we included the rest of our goals including the sharing of the calendars with other users and the in-app messaging.

Picturing the Deliverables

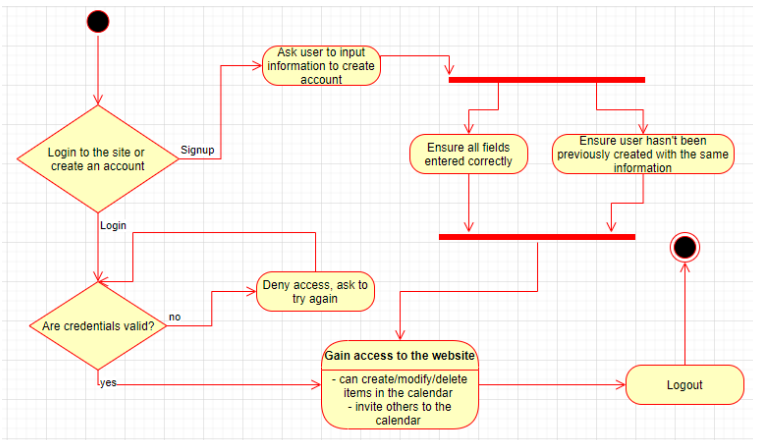
After completing the appropriate documentation in the planning stage, we were ready to start figuring out how we wanted our project to be structured. This would require us to take a look at what sort of data would be kept in our database, as well as how it would be stored. We also needed to explore how the user would be interacting with our site, and how the site would then interact with the back end to be able to provide the user with the results they are requesting. By doing this critical thinking prior to coding, we would be able to be more efficient with our code, and potentially save time by avoiding logical errors.

User Interactions: State Diagrams

The purpose of a state diagram is to visually represent how a user would be interacting with the website while trying to accomplish a task. The task could be anything from signing up for an account, logging into the site, manipulating the calendar or logging out. It essentially tells the story of how this task is executed, and shows the steps taken in order to accomplish said task. For our project, we decided to make two distinct state diagrams: one explains the process of signing up for an account or logging in to the site, and the other state diagram is highlighting how a user may traverse the different options available once they gain access to the website. Typically, one should have a state diagram for every action or task being completed, however since a lot of the tasks within our scheduling app are fairly repetitive, we found it to be more beneficial to have one state diagram with all the options rather than having around four looking fairly identical.

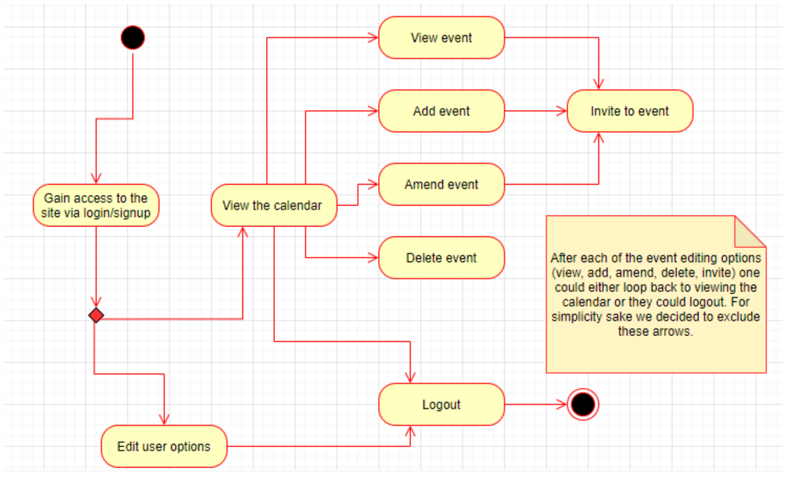
State Diagram for the Login/Signup

Whether a user is signing up for the first time, or logging in for the 100th time, they both have the same starting point (the black dot). Once they get to our website using a web browser, they will be presented with an option of either signing up for a new account, or logging in if they have a pre-existing account. We'll explore the new user signing up first. The site will ask the user to fill out only three fields: username, password and to confirm the password. Once the user fills out these fields, then comes in the validation. The fields will be checked to ensure that they were entered in correctly, and they'll double check to ensure the user's account hasn't been created before. Once both of these passes (it does not matter the order in which they do, but they both must pass prior to moving on), then their information will be added to the database and the user will be redirected to the website. We won't explore what can be done on the site within this diagram, however we do know that when the user is done, then they can logout which will then be our end state (the black dot surrounded by the red ring). If instead the user wanted to login, then the site will ask for a valid username and password. Once the user fills out these fields and attempts to login, then the database will be checked to see if the credentials are valid. If they are not valid, the user will need to try again since they will not be given access. If they prove to be valid, then they will gain access to the website and can do what they wish. After they are done, the process is the same as the sign up where they will log out and reach the end state.


Ask user to input information to create account
Login to the site or create an account
Signup
Ensure all fields entered correctly
Ensure user hasn't been previously created with the same information
Login
Deny access, ask to try again
no
Are credentials valid?
yes
Gain access to the website
- can create/modify/delete items in the calendar
- invite others to the calendar
Logout

State Diagram for the Website Traversal

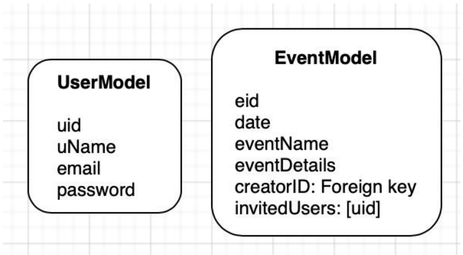
For this state diagram, the start state is when the user has gained access to the website whether it was from creating a new account, or if they had logged in correctly. After logging in the user will have the ability to manipulate their calendar in a multitude of ways. They will be able to add an event, view an event, amend an existing event or even delete the event. They will also be able to invite others to join the calendar so that the invited users can do all of the actions just mentioned. After any of these tasks are completed, the user will go back to the calendar view and can repeat any of these actions if they so desire. When the user decides they no longer wish to use the application, then they will log out; this will mark our end state and conclude the diagram traversal.



View event
Add event
Invite to event
Amend event
Gain access to the site via login/signup
View the calendar
Delete event
After each of the event editing options (view, add, amend, delete, invite) one could either loop back to viewing the calendar or they could logout. For simplicity sake we decided to exclude these arrows.
Logout
Edit user options

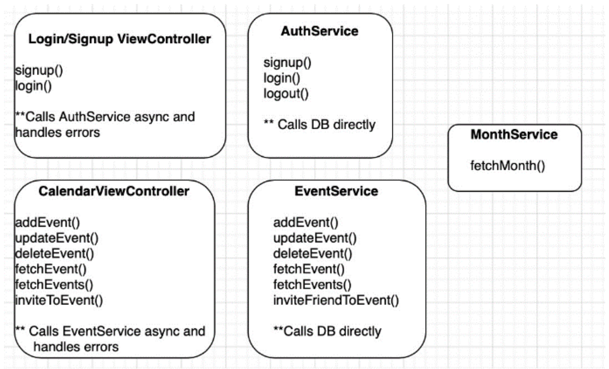
Data Structuring: Data Model and Class Diagrams

For the coding portion of this project, we need to think of what data we need to keep track of within our database, and how this data will be accessed. The data model diagram represented here shows that there are essentially two groups of important data that we must keep track of in the database. The UserModel holds the information needed in order to validate a user's credentials. The uid assigns a unique id to a user within the database (and would be considered a primary key). The uName, email, and password attributes will hold the information provided by the user when creating a new account. It should be noted here that when you physically visit our website, there is no field for the user to fill out an email address; this was omitted due to saving time when testing our website and validation process. Once this was completed, we moved on to tackling the calendar view as well as the event manipulation with the intentions of coming back and adding the email portion to the login/signup validation. Unfortunately, we forgot to come back to this as we attempted to complete more of our first MVP.



The EventModel holds the information needed to keep track of the events that the user adds into their calendar. The eid is the unique id to the events within the database (and is considered the primary key). The date, eventName, and eventDetails attributes are pretty self explanatory in regards to the info they are keeping track of relating to an event. The creatorID is a foreign key, which means it will be holding an id that is the primary key of another table; in this case it will be a placeholder of the uid in the UserModel table.

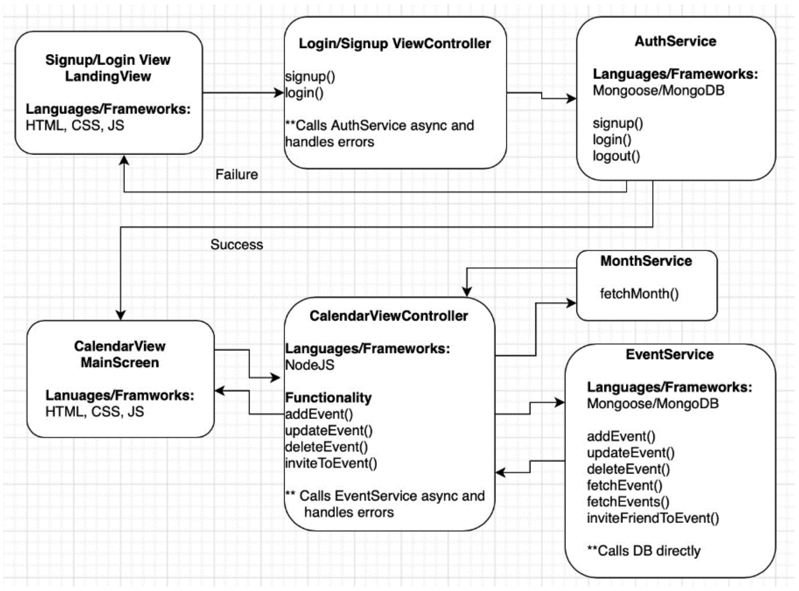
The class diagram outlines the functions within our code that will be the magic behind closed doors. This diagram shows the importance behind creating good names for functions. The fetchMonth() found within MonthService is what will determine the month to be shown in the calendar view. The functions found within the CalendarViewController will handle any errors and if there aren't any errors then it will call the respective function found in EventService; those functions will call our database directly. This is to improve the security of our website since the user



will not be in direct contact with our database. The same concept applies to the Login/Signup ViewController, those functions handle errors and call the respective functions in AuthService so there is no direct contact with the database. The only exceptions are logout() found within AuthService and fetchMonth() found within MonthService. These are exceptions because they aren't dealing with any user provided information (user credentials or events) stored within the database, therefore they can be called directly.

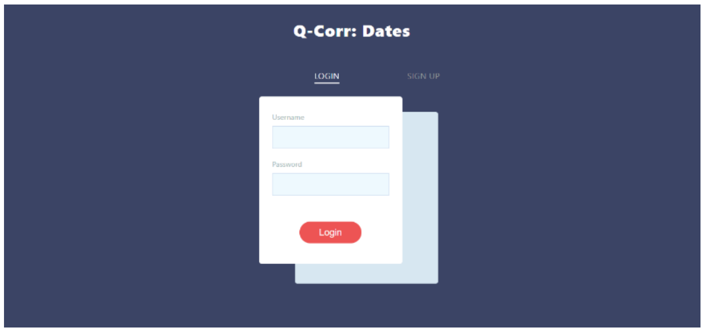
Another Perspective on Structure: Model View Controller Diagram

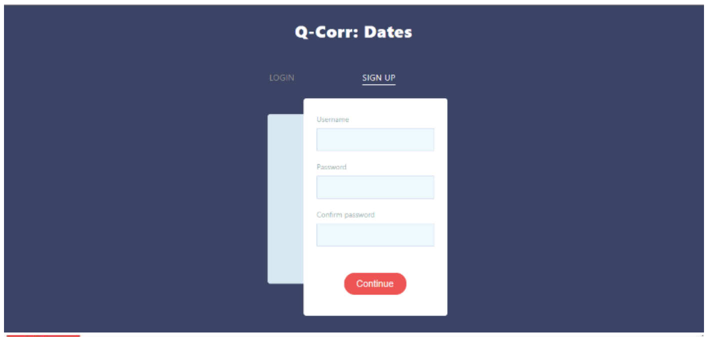
Sometimes it can be confusing to understand how the classes and functions will communicate with the database; be it directly or indirectly. To make this easier the above explanation can be represented in a graph like diagram. The Signup/Login Landing View would be the page the user first sees before they gain access to the website. Depending on how the user tries to access the site, the respective function will be called in the Login/Signup ViewController which will call the matching function in AuthService if there are no validation errors. If there are then they're redirected to the Landing View, otherwise they are sent to the MainScreen. The user can manipulate the calendar and depending on what they do will call a different event function in the CalendarViewController, which could then call the respective function in EventService. If a user wants to change the month then fetchMonth() in MonthService is called. What isn't shown with an arrow is when the user logs out it will call the logout() function directly in AuthService and send the user back to the LandingView.



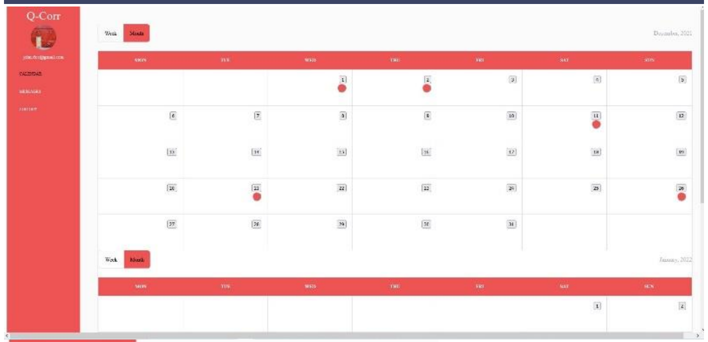
Diving Into the Code: MVP 1

For our project, we used a variety of languages and frameworks to create our website including HTML, CSS, JavaScript, Mongoose and MongoDB. Using HTML to create the skeleton of our site, CSS to make it appealing to the eye, JavaScript for the functionality, and Mongoose and MongoDB for our back-end database, we have created our online scheduling app Q-Cor. Currently our app can only be run

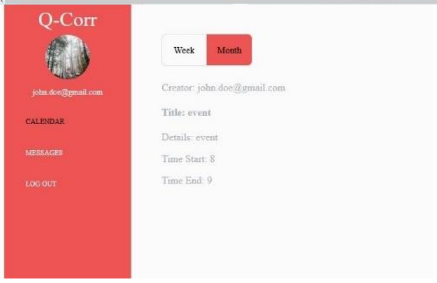




locally, so using node.js we are able to have it accessible for testing and presenting purposes. While running, it can be accessed online by typing localhost:3000 into the web browser. Once doing so, the user will be presented with our login/signup page; the default option shown will be for the user to login. The user can fill in the fields



with their credentials to gain access to the website and view their calendar. If this is the first time the user is coming to our website, then they can simply click the SIGN UP option and then the login portion will slide to the background and the signup input fields will slide to the front. Once the user has entered in their desired username, password and password confirmation then



Creator: john.doe@gmail.com

Title: event

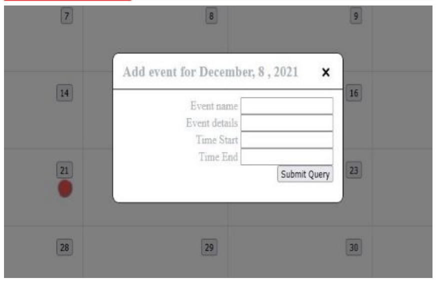
Details: event

Time Start: 8

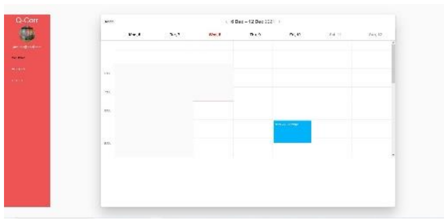
Time End: 9

the website will check to ensure the input is valid. If so, then the user will gain access to the website and will be able to view their calendar.

After successfully gaining access to the website, be it through the sign up or login process, users will be able to have a view of their calendar. From this view, any previously entered



events within the calendar will be indicated with the orange circle. If the user would like to view any of these events, they just need to click the event and then it will bring up the details. If a user would like to add an event to their calendar, they can simply click the date and a pop-up box will prompt the user for the required information pertaining to the new event. It will ask the user to input information regarding the event including the event



name, the event details when it will start and when it will end. Once you submit the query it will bring you back to the calendar view with and it will have been updated with the new event. There also is an option to view the events on a weekly basis rather than a monthly basis. This is what renders; however, the functionality is not fully working as we had hoped. To return to the monthly calendar view at any time the user can simply click the calendar option on the menu on the left, and if they wish to logout, they can do so by clicking the option in the left as well.

Unfortunately, we almost completed the first MVP, however, we were unable to get the remove event function and the amend a pre-existing event functionality done in time.

Reflection of Our Journey

From start to finish, this project did have some ups and downs. When looking back, we were very hopeful at the start, and a little naïve when it came to defining our project goals, creating MVPs based on those goals and trying to divide the work evenly in terms of having everyone work on all aspects together. In terms of the goals we created, realistically we only managed to complete about one third of them. I believe our biggest mistake was how long we waited before actually starting to code. Had we started when winter break had started rather than waiting until afterwards then we likely would have accomplished more of our goals.

Throughout our project journey we received feedback from Dr. Tim Macaig as well as the members from Team Pulaski. In the beginning when we pitched our idea to Dr. Tim Macaig he mentioned for us to be weary of must have's vs. nice to have's when it came to functionality as well as to focus on either the group aspect or the personal aspect of our website. This actually sparked some debate within our group about which functionality we wished to prioritize, however it was much needed to ensure we were all on the same page before we even started. During our second scrum we had some concerns about stakeholders since this was a school setting, and Dr. Tim mentioned that these would include groups of people that might be interested in our products (not just university students exclusively). This helped us to solidify our target audience, and to further think about planned deliverables and how we should organize our project goals into MVPs. As for Team Pulaski's feedback, the main points we chose to act on were the concerns of our MVPs, smoother transitions between team members within our vlogs, and to have the user be sent straight to the calendar view after signing up for the first time rather than having to login after signing up. We believe all the feedback we've gained has helped to push us in the right direction, and to keep us on track; there was definitely value in having the scrums.

This project had some great potential, and given the circumstances I think we did fairly well. We do feel as though there was some missed opportunity with the outcome of our physical website due to the fact that we started the coding portion so late. Looking at the project as whole, including the documentation, the group discussions, becoming more familiar with GitHub, learning to communicate efficiently, and having to make changes as problems became apparent, we feel like this was a successful experience. We liked the challenge of the project being so open in regards to topics, and we also liked how the activities and scrums were structured; this allowed us to stay on track for the most part and know what was expected for each stage. Some things we disliked and struggled with were how to navigate some things like stakeholders and MVPs; we've never done a project like this before so we had nothing to reference from past experience. Due to this, we made some mistakes that cost us later on in the project, however we've learned from those mistakes and tried to adapt as best we could. The experience was good, now we have a better idea of the process, and can potentially have better project scopes and more realistic MVPs moving forwards in our careers. Moving forward, we believe that the communication methods (scrums, KanBan boards, weekly meetings with members) we practiced will definitely come in handy in future projects for sure.

The thing we feel most proud of would have to be the quality of our code. We really tried to go through and ensure we had clean code, and tried to make sure the variables, classes and functions all had relevant names that were self explanatory. We were also proud of the fact that we did not use an API, so everything that is rendered and functioning was all done by hand, from scratch and by us completely. As a group, we learned that it is alright if a person is unable work on every aspect of the project since we all have our strengths and weaknesses. Instead of trying to carry on this way, we utilized our strengths to increase productivity. We also learned about time management, and the importance of creating manageable and reasonable MVPs. Overall, this project was an interesting experience, and we were glad to be given the opportunity to explore how such a project would be done out in industry.