ENSE 374: Project PAT Final Report

Team members
Cameron Wilson
Justine Papeleras
Michael Osachoff

Project sponsor
Dr. Tim Maciag

Summary

This is the final report for project PAT done by Team Pulaski. We created a Physical Activity Tracker using the Model View Controller architecture as well as project management skills and documentation. The goal of our project was to allow users a way to track physical activities to stay healthy and encourage working towards a healthier lifestyle. We feel that our first Minimum Viable Product (MVP) successfully meets this goal, however it does not have a social aspect as that was left for the second MVP. We had some struggles as a team, especially with some timing and the calendar functionality but all in all, the MVP works as intended. We learned during the project that we all have to continue to develop our skills and take the lessons we learned into future projects.

Goals and Feedback

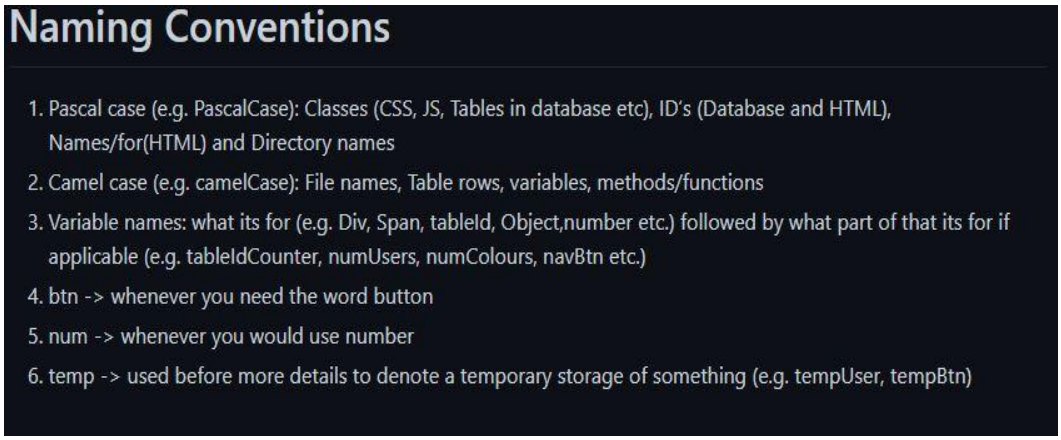
The goal of this project was for us to work together as a team in order to create and ultimately deliver a Minimum Viable Product (MVP) using the technologies and theories we have learned during our labs and lectures. Our project idea was the Physical Activity Tracker, Project PAT, we wanted to solve a modern problem where more and more people do not get the exercise they need, as well as fix struggles than some current tracking methods have. The platform that we intend to create will allow the users to store their current routine, and create their own schedule of future physical activities.

The goals were achieved by us having a completed and fully functioning website. Furthermore, as a group we have followed and completed the project goals and objectives that we have set out during the project initialization phase. The feedback we have received from both our instructor, Tim, and peers, Team Picard, were exceptionally professional, positive, and informative. Tim allowed us to improve the areas that were lacking and generally enhance the deliverables between each scrum meeting, as well as made sure that the team is on track and on time. Tim has also kept us on track and on time. While Team Picard provided distinct responses on areas that were of major

concern when it came to documentation and organization. All things considered, the feedback improved how our Github is organized and clean. It made sure that documents are properly submitted and in the correct format. This also applies to our code, as we tried to stay consistent and clean with our naming conventions and how we approach the style of our programming.

Project Goals	Solve a modern problem and develop a physical tracking activity platform that promotes a long lasting healthier lifestyle and habits. Create a more interactive and motivational tracking activity experience.
Project Objectives	➢ Encourage people to be more physically active. ➢ Add a social aspect via a built-in feature or a 3rd party platform. ➢ Create a more interactive user-interface. ➢ Allow users to input current goals, status, and personal best. ➢ Allow users to track and view their own progress, as well as others. ➢ Allow users to do other users activities and as well share their own.

(Project goals and objectives from the Project Charter)



(Naming conventions we had to follow in our code)

Documentation and Pre-Code Work

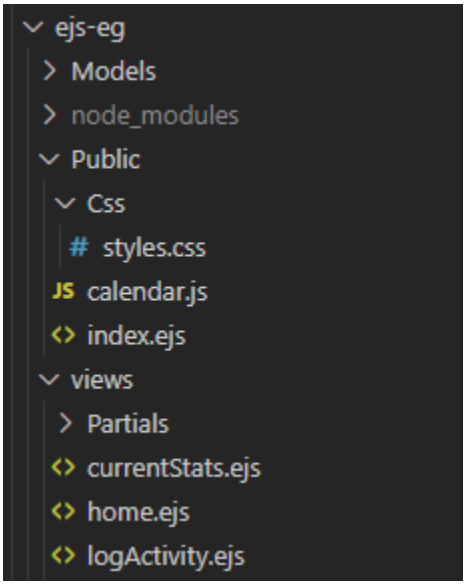
As a group we started off by making our pre-code documentation, including a business case, project charter, a scope statement, stakeholder register, and a Responsible Accountable Consulted Informed (RACI) chart and placing them in our GitHub. These documents allowed us to nail down the specific functionality we wanted to have by the end of the term and also think about the expansions possible for the future. We did a cost-benefit analysis where the main costs were time. The budget set out for the project was a total of zero dollars, and we are happy to say that we maintained that budget while also being on time with our deliverables. Another main

thing that came out of this documentation was some of our risks, one of the main things being the lack of experience of our group and the tight deadlines we would have to make. From there we defined our scope to exclude a social aspect for the time being, as we felt we did not have the know-how on how to implement such features at that point in time. After all of this was done, we moved into looking at how we would apply the model-view-controller (MVC) architecture with a set of diagrams. Using some People Centered Design ideas, we started with a user story map that would show us an end to end experience we wanted to have. We then sorted what we had into two Minimum Viables Products, the first being focused on the solo user and the second adding that social aspect to the project. We did not go further into this project than the first MVP. We also did some state diagrams and a communication diagram to lay out just how our code would communicate across each task the user had, and also the paths they could take to get to a new state. These documents were the last ones done before we had moved onto getting feedback from other groups then finally, into the code.

MVC Architecture

Our code follows a traditional MVC architecture, and it consists of Embedded Javascript (EJS), JavaScript, and CSS3 styling. The model aspects of our project have used JavaScript to implement a MongoDB database that will store all of the user information. This includes schemas for the users, their goals and stats, any logs they create, and activities that they can use to add to their logs. The database creation can be found in makeDb.js and it creates a new MongoDB database named projectPAT that contains the schemas mentioned above, and these are found under ejs-eg/Models.

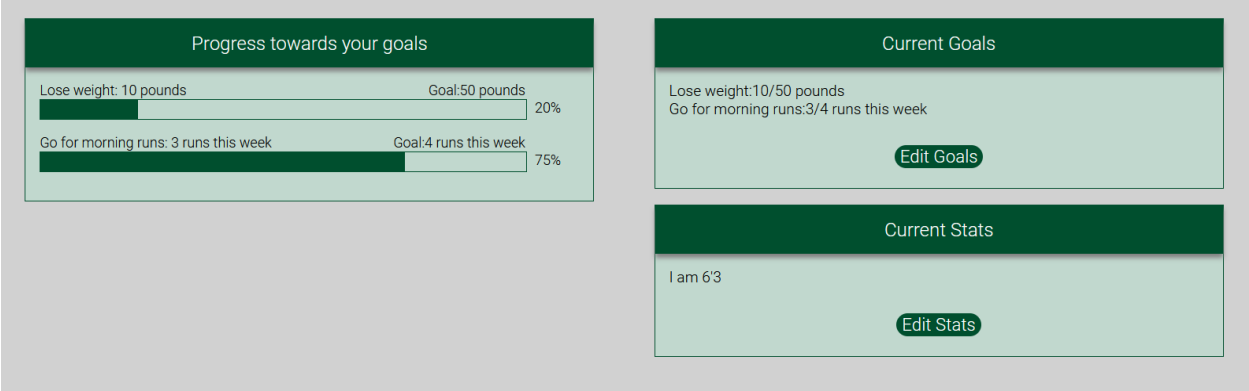
The EJS files of our program are used to display the view for the user to see, and they are located under ejs-eg/Public and ejs-eg/views:



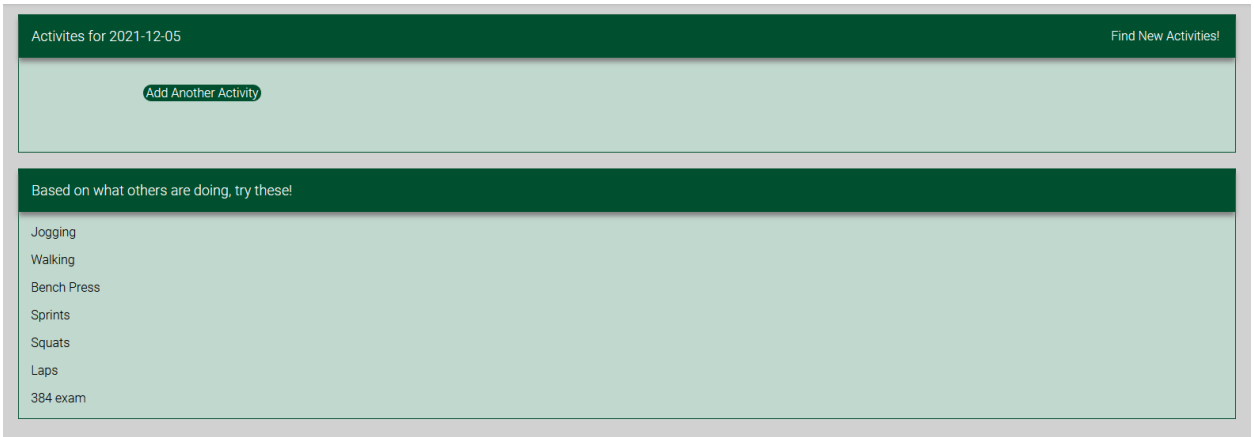
The Public folder contains the index.ejs file which is the login page for the program, the CSS stylesheet used to format the view for the entire program, and a calendar.js file that is used to create the calendar in the home page. The styles.css stylesheet contains all of the classes and id's for the program that have specific styling, but there is also broader styling used for the body, and variables that assist in determining colors for the program. The stylesheet also contains many classes and styles that could be combined, and they may also not always comply with the naming convention that we set out for ourselves, but we believe that it is up to a generally good standard.

The login and signup page (index.ejs) contains minimal content, but we strived for minimalism within the page, and we believe it is acceptable for our first MVP. The home.ejs page is the page you are redirected to upon successful login and it includes many partials that are located under: ejs-eg/views/Partials, and these partials are used in the home as mini-viewers of the other pages, and full-viewers within the separate pages. This includes content boxes for the first activity logged today, progress towards the two oldest goals, the user's stats, a calendar to create/edit a log at a selected date, and an activities list to motivate the user to try new activities. It also includes a header bar that is consistent between all pages once logged in and it is used for quick navigation between each page. Upon interacting with any of the navigation options within the header bar, the user is redirected either to the current stats page, or the activity logging page.

The current stats page contains three separate sections: current stats entered by user, current goals inserted by user, and progress bars to show how close the user is to reaching their goal. They can be seen in the screenshot below:



The log activity page contains the activities logged for the selected dates, and it also shows what users have created and added to the database as custom activities. For reference, a custom activity would be "384 exam" in the screenshot below:



These combined view pages allow the user to see where they are at in reaching their physical activity goals, and it assists in motivating the user through the logged activities, custom stats, and progress bars for their goals.

The final part of the MVC architecture that we have not yet touched on is of course the controller aspects of our program. The controller was the most complicated part of the project. There was a lot of data to not only get, but to send to the view so it could be displayed. Pulling the data was the first thing that we focused on, what do we need to get, how do we get it and when? One of the things that we did in app.js was send lists to an ejs file, this ejs file had controller code in it that was solely responsible for sending the data given to it to the right spot on the page with the correct functionality applied. A good example of sending a list and then working with it in another file is the log activity page. This page gets sent two lists, one of the activities that are logged and one that is the details about each activity. The controller logic in the file goes through all the items of both lists and then determines how to display them based on the type of activity. This allowed us to show a user their own version of the view, but it came with some challenges on the controller side. Getting the details for all the activities was the biggest thing, and what we ended up doing was building an array of details that belong to the log we are trying to view, and then “decoding” them into their respective activities. The following is the code to collect all details in the controller:

```
let activsDetails = [];
for (activity of loggedActivsToRender) {
  let tempDetails = await Details.find({ activityId: activity._id });
  for (det of tempDetails) {
    activsDetails.push(det);
  }
}
```

After getting the correct data from the database, the other controller element that we had troubles with is contained within calendar.js. This file is responsible only for displaying the calendar and changing what month it shows, and it took many hours to get this working and many iterations; however in the end, we managed to get it working. Originally we had tried to keep it within the view, but after the first attempt we noticed that this wasn't going to work. Once it did work, we ended up with a calendar and controller that works as intended and better than we had planned.

Team Reflections

How did we feel and were we successful?

Our team overall is proud of the project at this moment in time however, we also feel that we may have bitten off more than we could chew so to speak. During the documentation phase of the project we were either on time or ahead of deadlines allowing us to think things through, go over things and refine the content of our documentation to reflect our actual thoughts better. When it came to actually writing code we ran into many roadblocks along the way that hindered our success. We underestimated the amount of time it would take to do the back end programming by a large margin. This meant that for the first time in the project, we were playing catch up. The main thing that the group did to overcome this was a natural shift in the roles of each team member. Originally, we were all equals and full stack developers and in the end, Cameron became more of a project manager doing full stack development, Justine became the main front end developer as well as doing most of the writing for the final presentation. Michael took on a full stack development role in the project as well, with more of a focus on the controller aspect of our MVC architecture. Despite the timing challenges we did feel that our project was successful, we delivered the first MVP to the best of our ability and grew our skills in web-development along the way.

At the beginning of the term, none of us would have been able to do what we learned doing this project and being able to not only pick up new skills but deliver a working project is something that we are proud of. As we worked during the project we all came to like the MVC architecture that we used to build the website, the separation of concerns and laying out how each level communicated prior to writing code was a large part of our success. We knew what the code had to do before we wrote it and that saved not only time, but effort and stress as well. The other main thing that our group enjoyed was the freedom that was provided when making this project, the idea was not given to us by the instructor and we were free to choose exactly what documentation we did, for the most part. What documentation we were forced to do became much clearer why we were forced to do it in the end, and some of it even helped us decide on what optional documentation we should do.

The main thing our group did not like was our lack of experience going into the project. We didn't know what we actually knew how to do and that really limited both the ideas we put into the project as well as caused some sleepless nights over errors that we just were not sure why they were happening. Cameron specifically was trying to layer ejs partial in a way that was not intended, and that led to bugs that were hard to hunt down the cause of.

What is the team most proud of?

Team Pulaski is most proud of how dynamic our website ended up being, very little stays the same from user to user aside from general format and we liked that. Every user has their own version of the view as well as their own saved details in the database. We are also proud that we were on time for most of the project without rushing to do things at the last minute, some of the group members tend to leave things until the last minute and as a team, we managed to overcome this and right up to the point where coding started we had success with this. We also managed to come back from our lapse of procrastinating before the last week of class started. Getting back on track is definitely where roles changed but we are happy with the end result of these challenges.

What did we learn?

Team: As a group we learned that it was not the best idea to have everyone as a fullstack developer and no manager. If we would have had a manager from the start, then we may not have fallen behind when we did and actual writing of the code would have gone much smoother. We also learned that documentation is more important than we originally thought. This documentation assisted us in getting back on track and was a large reason we had success in the end.

Cameron: I definitely learned that I need to improve on the front end side of web development more so than on the backend. There was a lot of frustration in styling on my own page, and even points where I was thinking more about the actual data than the users, and it caused a lot of deleting and rewriting HTML and CSS. I like thinking about function more than how it will look, things like colour were something I didn’t spend a lot of time on and in the future I would like to improve on this the most. I also learned that web development is probably not a career path I want to take, I would do it if given the chance, however I would much rather do something else.

Justine: What I have learned during this project was that I am still lacking experience and needing more practice on the back-end side of web development. And because of this most of my work during the coding was the front-end of our site.

Although I mostly work on the front-end, I still would not label myself proficient enough on the front-end as I quite run into problems from time to time. All in all, I definitely learned that more practice and time is needed for myself for both the back and front-end of web development.

Michael: What I learned most in this project was that I am roughly average in my overall capabilities in full-stack development. I was slow but methodical when it came to creating the controller aspects of the program that I contributed to, and I slowly worked line-by-line to understand and work through the logic of the functions that I created. I didn't directly touch the model aspects of the program itself as Cameron volunteered to create them himself as it is his strong suit in full-stack development, but I did provide a pair of eyes that assisted in a pair-programming style to minimize the size and structure of the database while still allowing it to have a large amount of power. My view-based skills in MVC design are similar to the controller in terms of how good I am at completing the task at hand, but more often than not I am throwing styles and classes and formatting at the ejs pages just hoping it will work. Overall, I believe that this project was a great opportunity to get my feet wet in full-stack development, and I am glad that I have at least some practice in web-based programming.

How will we use what we have learned?

This project was excellent for getting all of the members in Team Pulaski exposed to web development in a project environment. Our experiences show that none of us excelled doing everything and it was much better to have front end and back end people. We will use the MVC architecture in the future when developing websites as it feels much superior to the methods taught to us in other classes. This also allows for the separation of concerns, and in the end cleaner code. Paying attention to writing clean code is another thing our group will use in the future. It is very likely we will work as part of a team in the future, and writing cleaner and better code will allow us to excel in a team environment.

What would we like more help with?

As a team we don't feel additional help with class content would be beneficial, we all understand what we did during the project however, we all feel more practice would be beneficial. Something that caused trouble like the calendar functionality could have been avoided had we been more experienced. We feel experience was our biggest downfall when it came to our struggles, and there is not a lot that you can do to replace experience.