4-2 MILESTONE

4-2 Milestone Three Enhancement Two Algorithms and Data Structures

CS-499-12473-M01 Computer Science Capstone 2024 C-5 (Sept-Oct)

Michael Zietz

Southern New Hampshire University

Sept 29, 2024

* Briefly describe the artifact. What is it? When was it created?

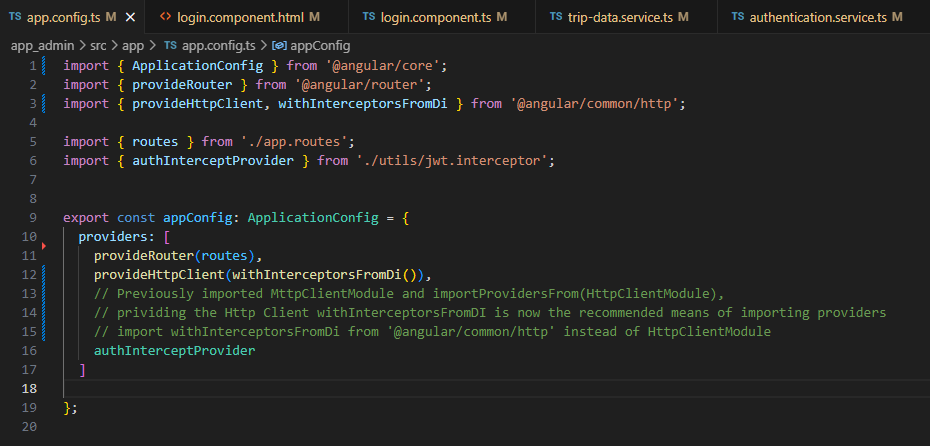
This artifact creates a web-based application, named Travlr Getaways, that creates a platform where vacation trips are packaged and published for travel agents to add and modify. It was created for a previous course at SNHU, CS 465 Full Stack Development, and its original state was the final submission for the class. It utilizes the MEAN stack, which includes MongoDB, Express JavaScript, Angular JavaScript, and Node JavaScript. Angular serves as the client-side framework written in JS, while Express is used as the framework for the backend, and MongoDB functions as the database. Node, also written in JavaScript, acts as the primary web server framework. In its current state, Travlr’s web application is capable of hosting these travel pack ages, allowing non existing users to view trip details, but requiring authenticated sign in to edit existing or create new trip packages.

* Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in software development? How was the artifact improved?

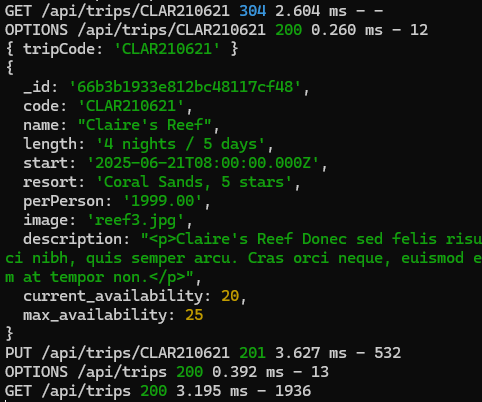
This artifact was selected with the intention to correct known functionality issues in addition to new features that create purpose for new users outside of the current one designed for a trip agent. I sought to improve the clarity and structure of the project as well as correcting some lingering issues left after its original submission. Due to deprecated use of login authorization in the project’s final version, this left an issue with users not being able to submit POST commands to create new or edit existing trip objects after signing in. I seek to implement current tools and APIs used to perform this authentication and enable full functionality to the website developed. I also would like to revisit other pages from the website and implement SQL functionality to other areas other than trip ID management. I plan to implement booking reservations and reconciling availabilities. In its current state, the website is only able to take advantage of a small portion of its promised features and does not separate different types of users signing in. I seek to introduce user roles, those intended to edit trip and those intended to book them, and allow functionality related to them. The website will also need to have features implemented to allow booking once all conditions allow it.

The first enhancement was to rework how Angular’s app.config.ts file handled providing interceptors for user authentication. Before enhancements, it utilized the HttpClientModule library which has been deprecated starting with Angular’s version eighteen. The fix that I discovered was to import withInterceptorsFromDi from Angular’s common, http folder because it “includes class-based interceptors configured using a multi-provider in the current injector into the configured HttpClient instance” which was compatible with the current use of provideHrrpClient libraries. (Angular, 2024)

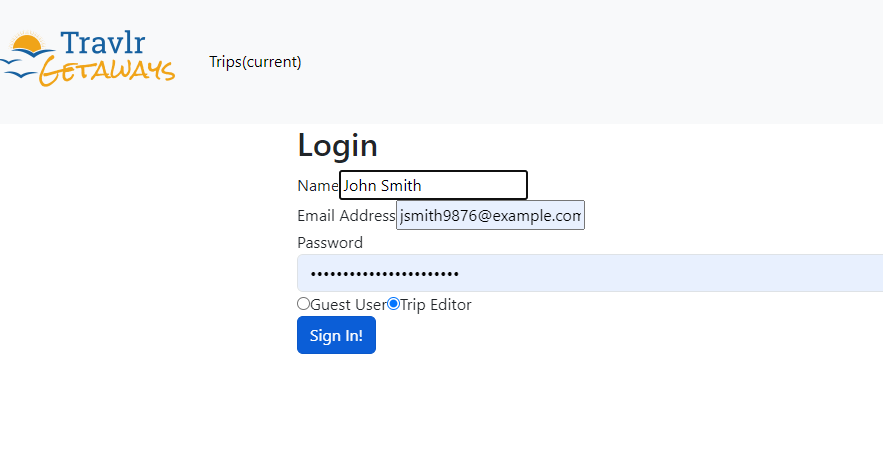
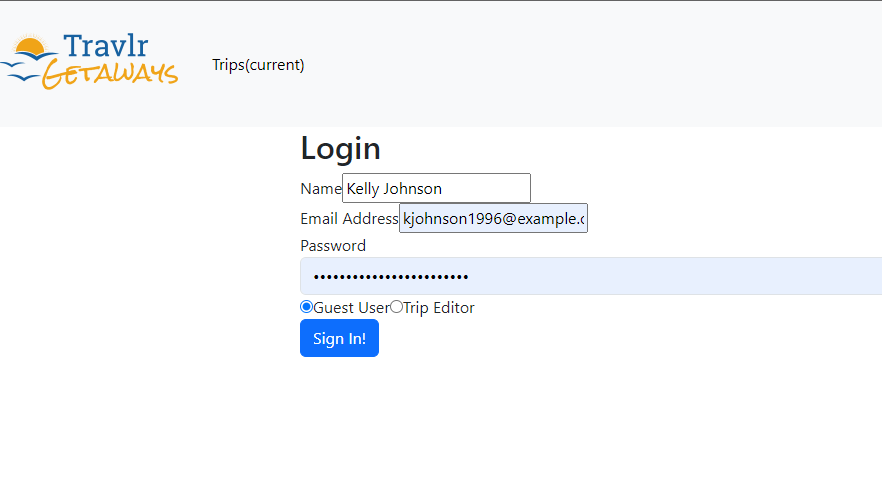
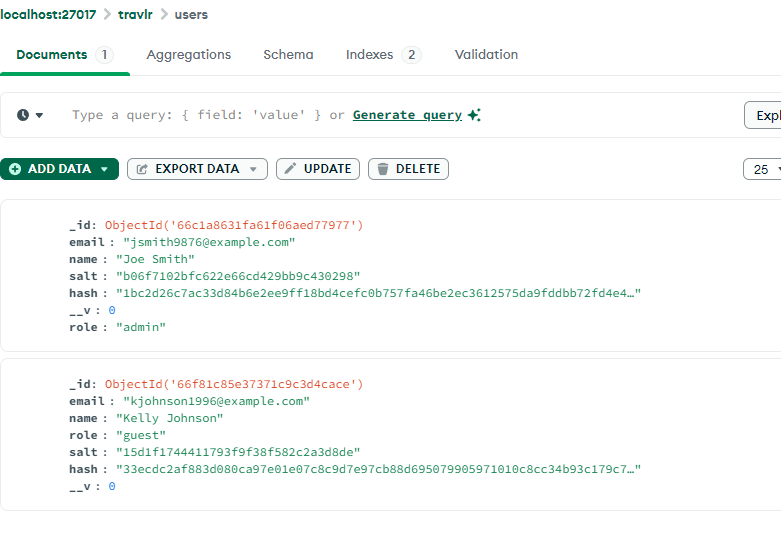
(Origin: app.config.ts)



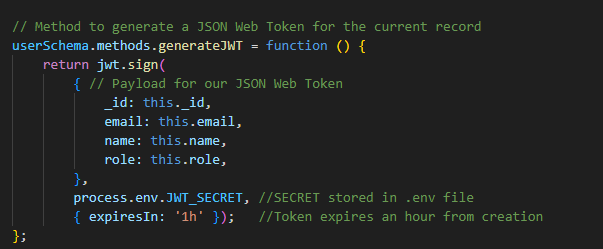
After this change, the artifact was able to authenticate and approve PUT SQL requests base on users signed in and modify data elements stored in MongoDB’s trips table. Here is an example of the backend logs approving the PUT command.



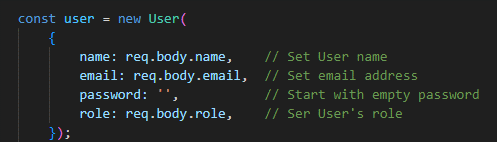
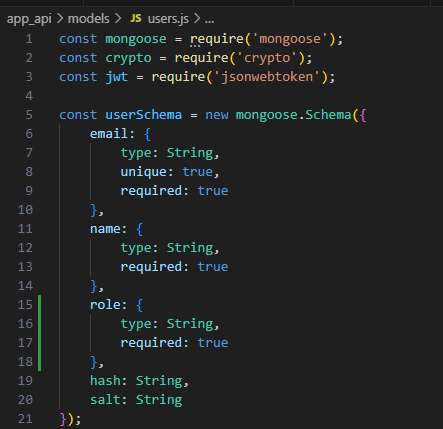
The next feature I wanted to implement was the possibility for multiple types of users to use this platform. I created a new variable specific to the user class named “role,” updated the current users in MongoDB to possess this new trait and registered a new account in the system possessing the other user trait relevant to the website. First, I wanted these roles to determine what functionality the user had access to and decided these would be the “guest” and “admin” descriptors. The admin role described the travel agent that would need to access and update the trip info hosted on the website, and the guest would be able to modify the availability for a trip as though they were reserving it for their party. This meant I had to implement these roles throughout the app\_admin, app\_api, and app\_server’s files and require their submission for authenticated signing in.



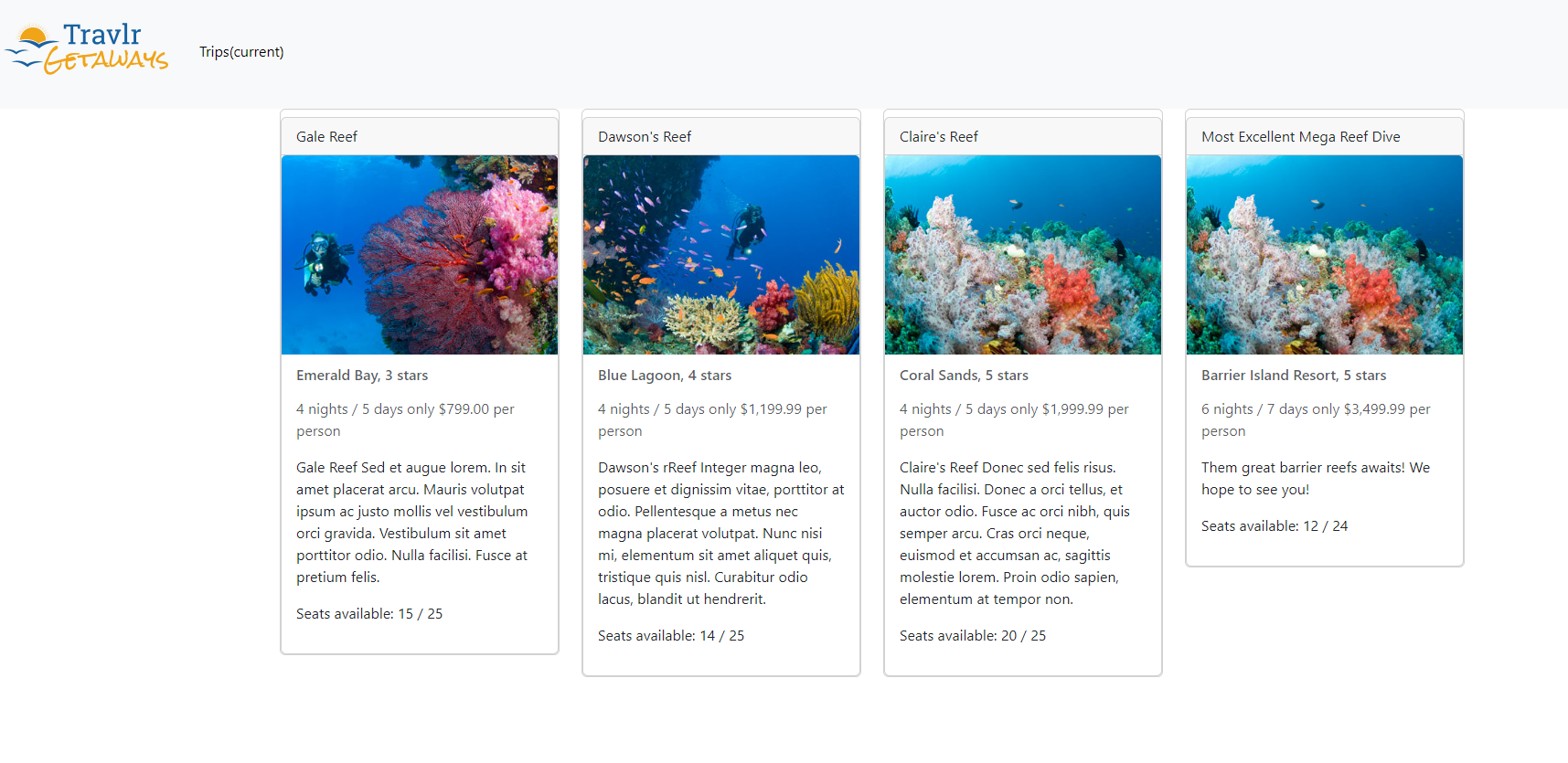
(Origin: app\_api/controllers/authentication.js)

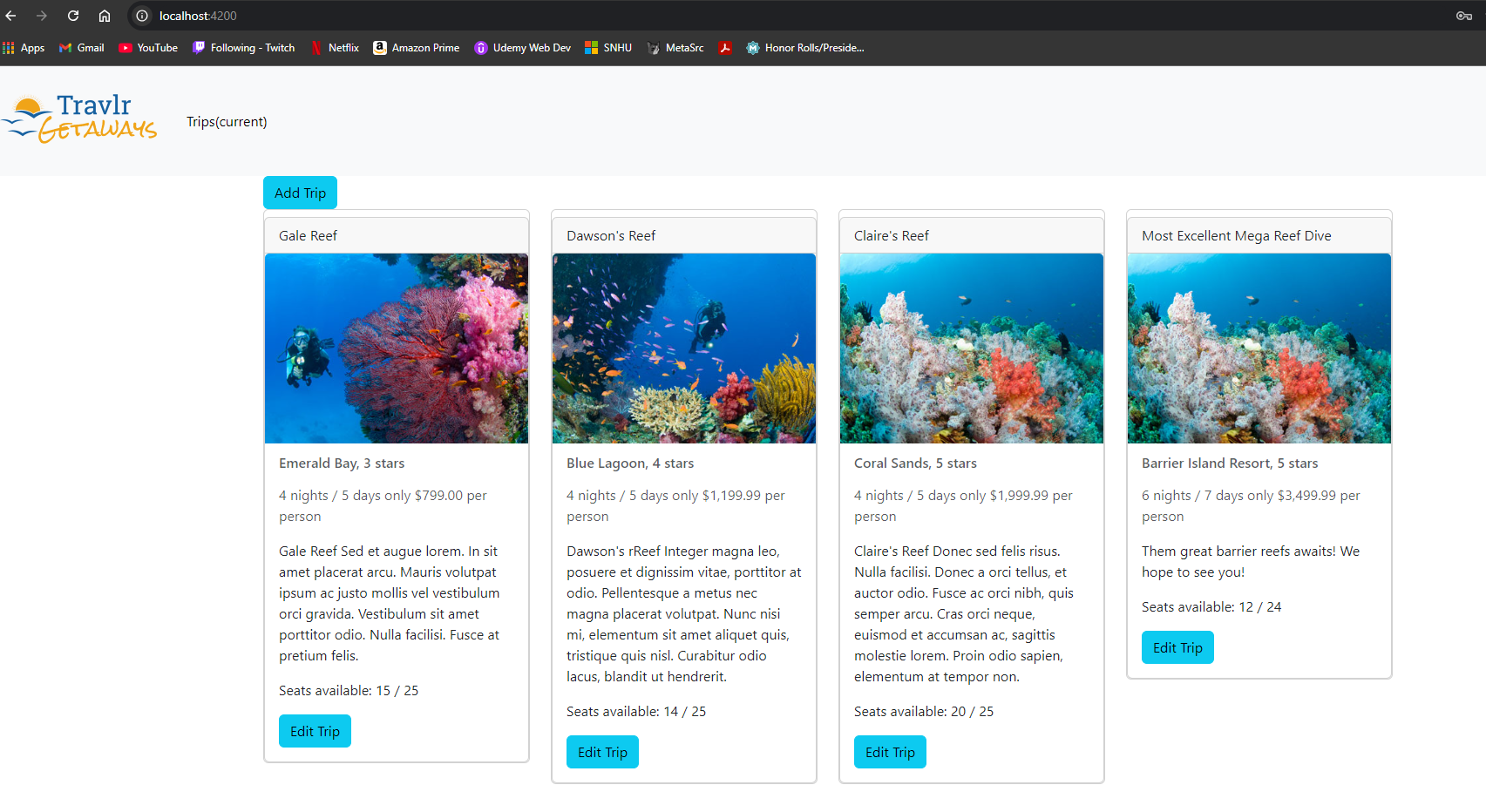


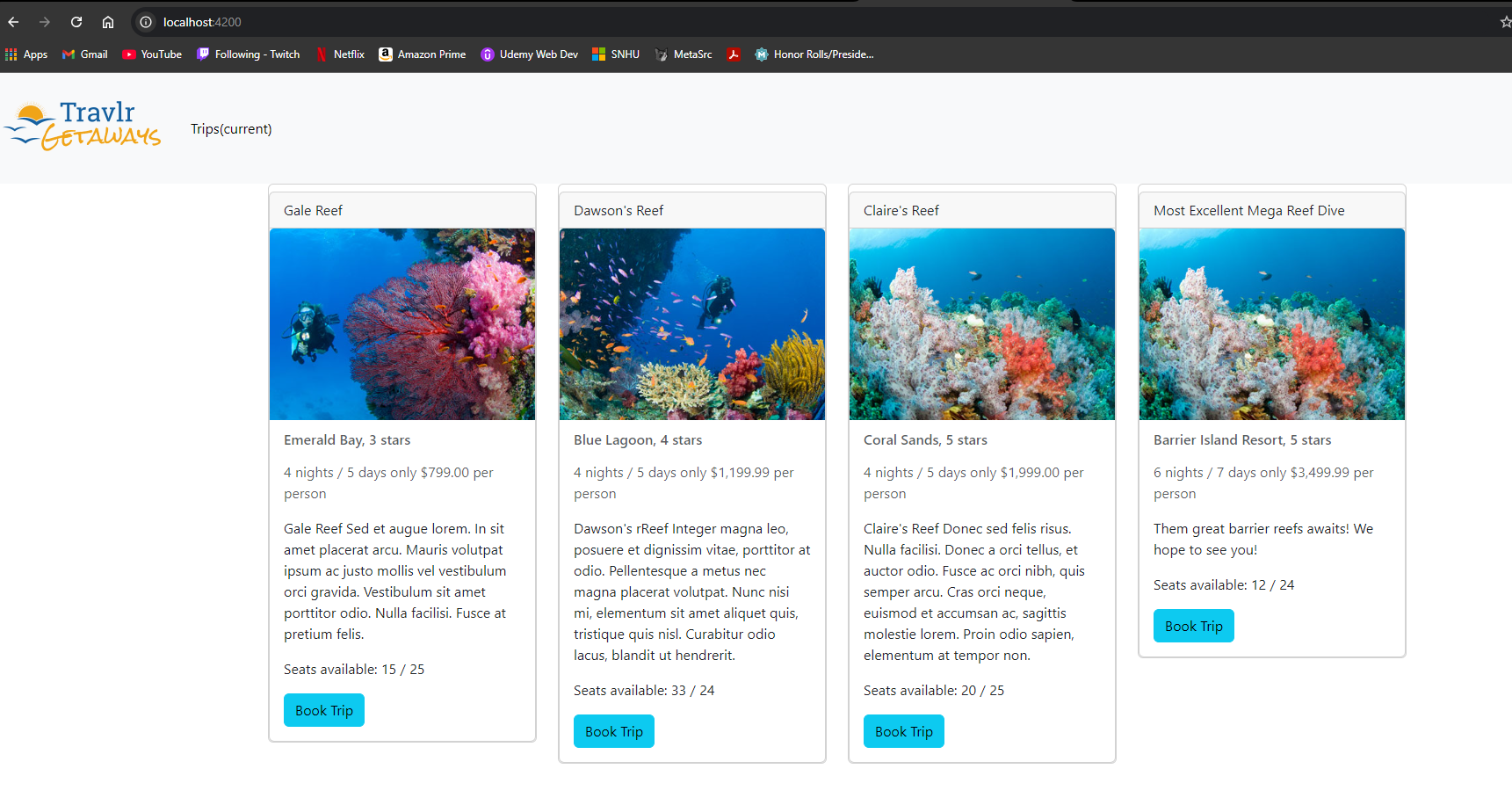
(Origin: app\_api/models/users.js)



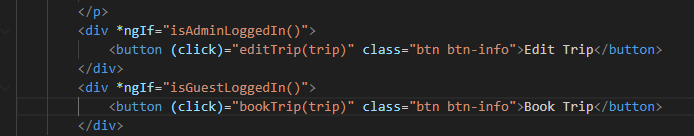
Using this new role functionality, I activated and disabled different features in the form of buttons determined by who was signed in a given moment. Guests would be able to see the “Book Trip” button while admins could still select “Edit Trips.” If there was no authenticated user signed in browsing and viewing the current trips page was still available, but all editing was unavailable. The Add Trip feature I desired to only be available to the admin role, as guests should not be able to create new trips, so I added additional logic for the trip-listing component to only show this button if an admin was signed in as well. In the following photos, note the presence of buttons available for each trip as well as the new and included trip info regarding seat availability. This was implemented by giving the Trip class the new variables, current\_availability and max\_availability, to imitate the seating pool that guests and admins could access.

 (No user)

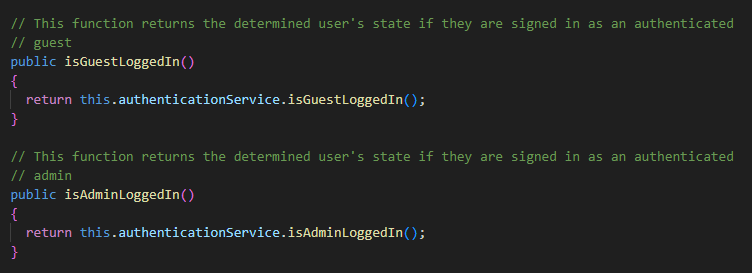
(Admin)

(Guest)

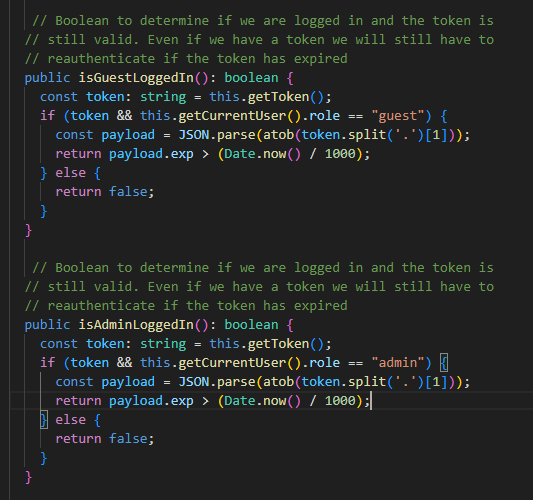
(Origin: trip-card.component.html)

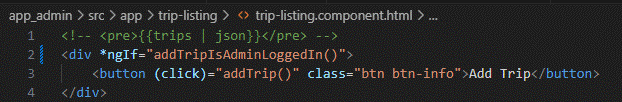


(Origin: trip-card.component.ts)

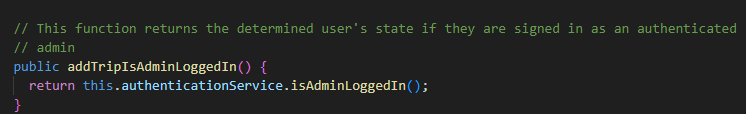


(Origin: authentication.service.ts)

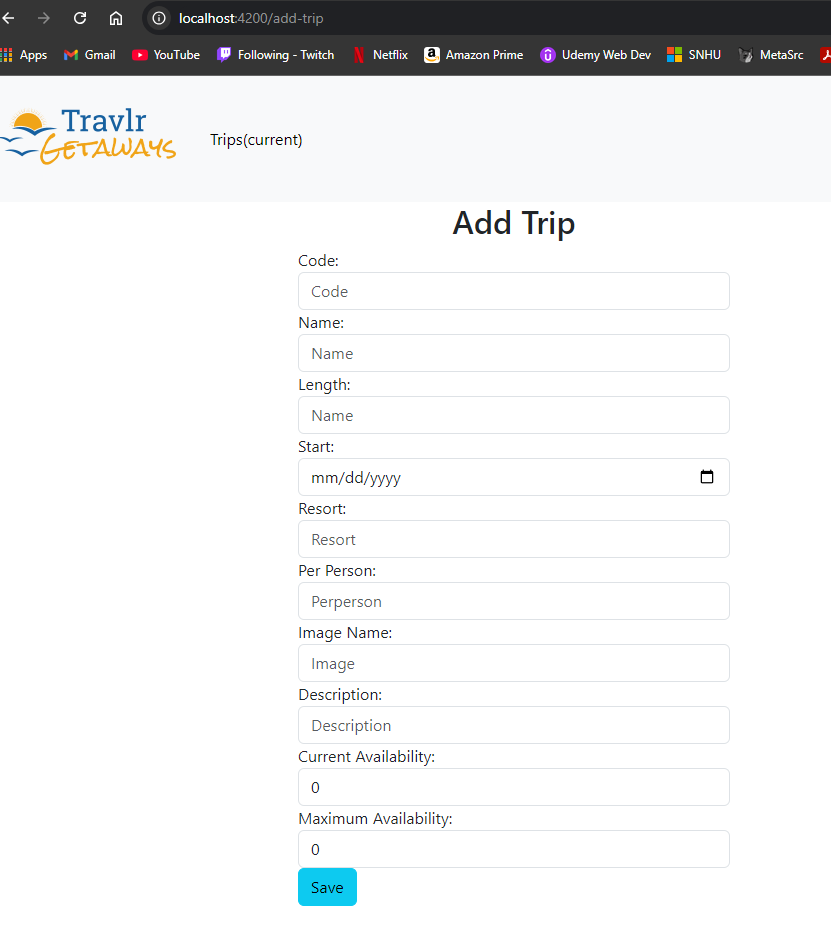
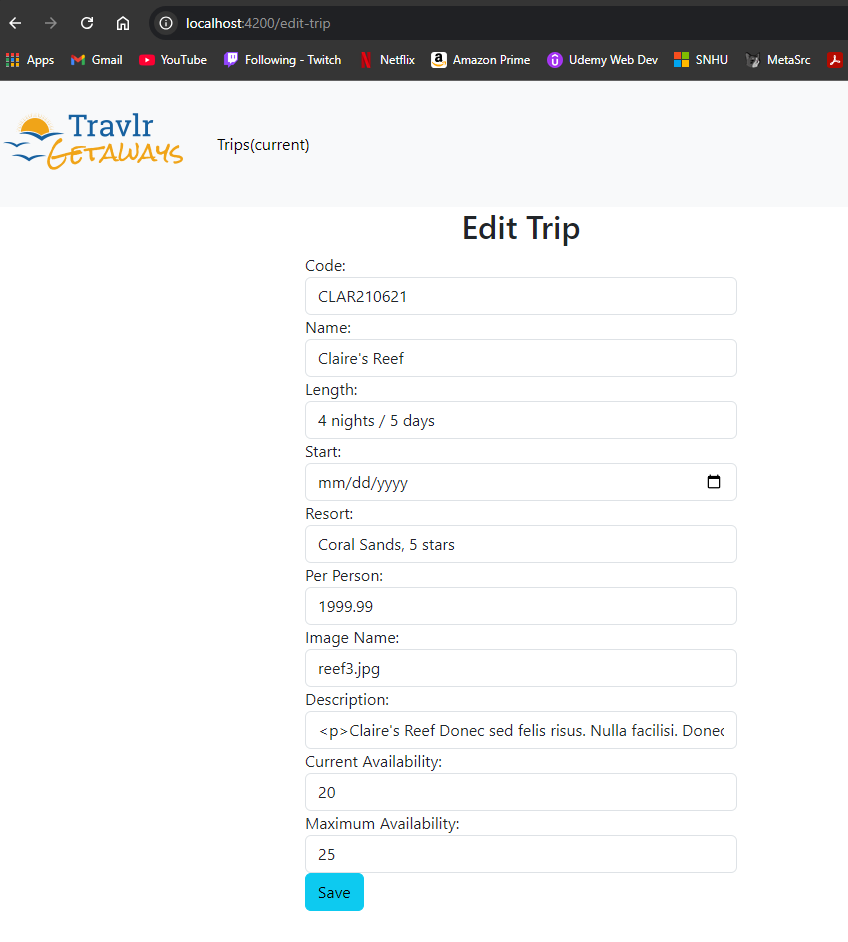


(Origin: trip-listing.component.html)

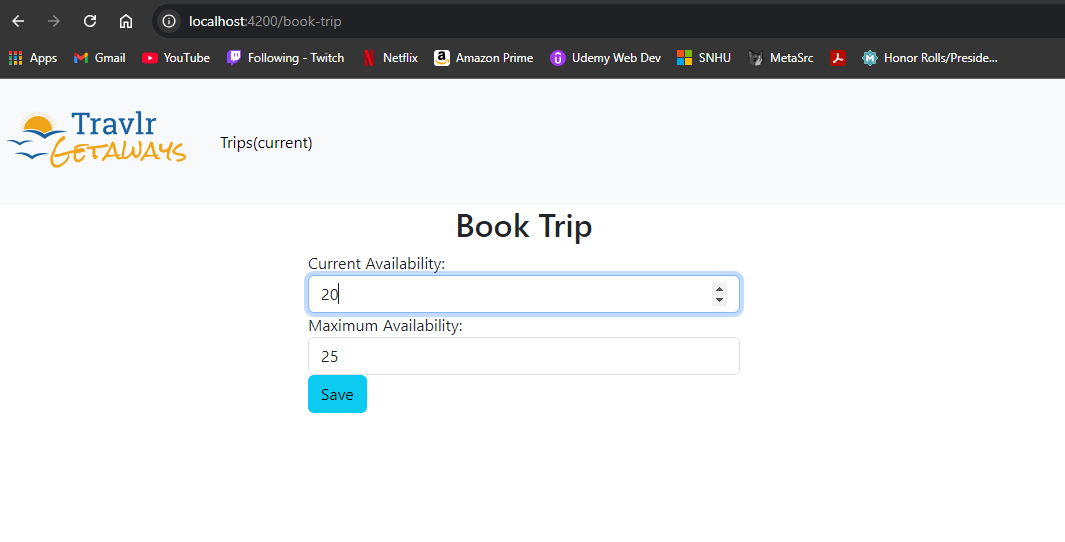
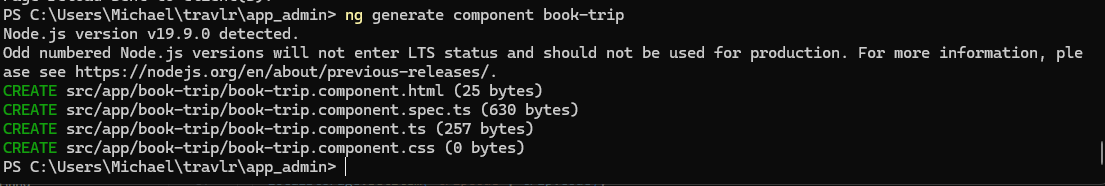
(Origin: trip-listing.component.ts)

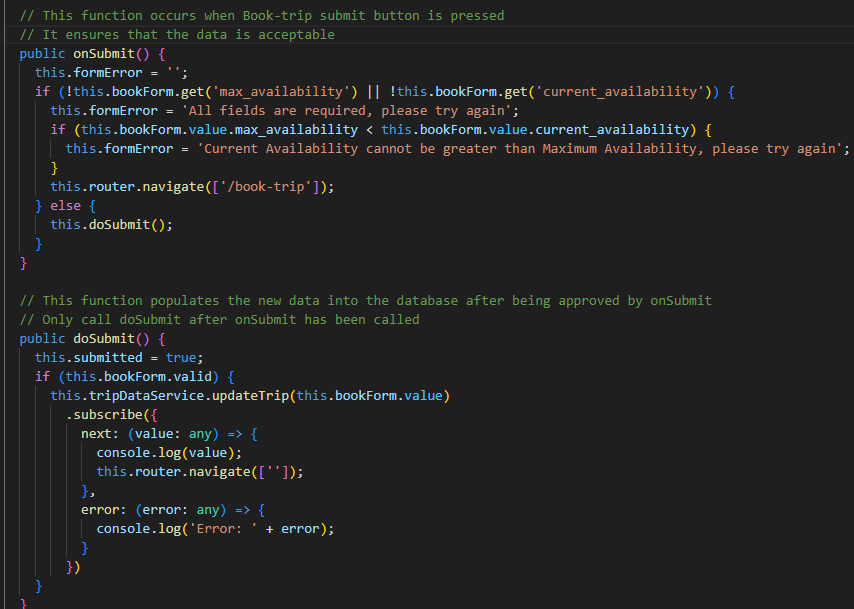


I then needed to implement the new trip variables in the Edit Trip and Add Trip pages, so admins could edit and create trips that accounted for the availability feature. These are required variables, just like the others present, so a PUT command cannot be submitted where form entries are empty/null. As you can see in the next two screenshots, the Edit and Add Trip pages now take into account the new availability variables.



The final new feature I wanted to add to this artifact’s enhancements was the guest user’s booking page. This was accessed by submitting the available Book Trip button for a specific trip. This meant creating a new component called “book-trip” and creating accessibility to the page via routes. Node then populates the necessary folders and files in the app\_admin folder according to the book-trip naming convention. From there, I needed to add additional code establish the route for this destination as well as the functionality code of actually booking and editing these trips’ availability. This defined in book-trip.component.ts which uses html pieces that have been implemented in book-trip.component.html where buttons, forms, and even logic requiring data is located. In the following screenshots, it depicts the component’s creations, the visualization of the html code, and the functional submit functions that edit the trip variables from the SQL database.



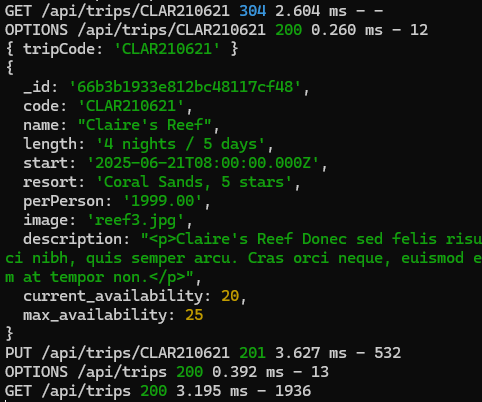
(Origin: book-trip.component.ts)

(Origin: book-trip.component.html)

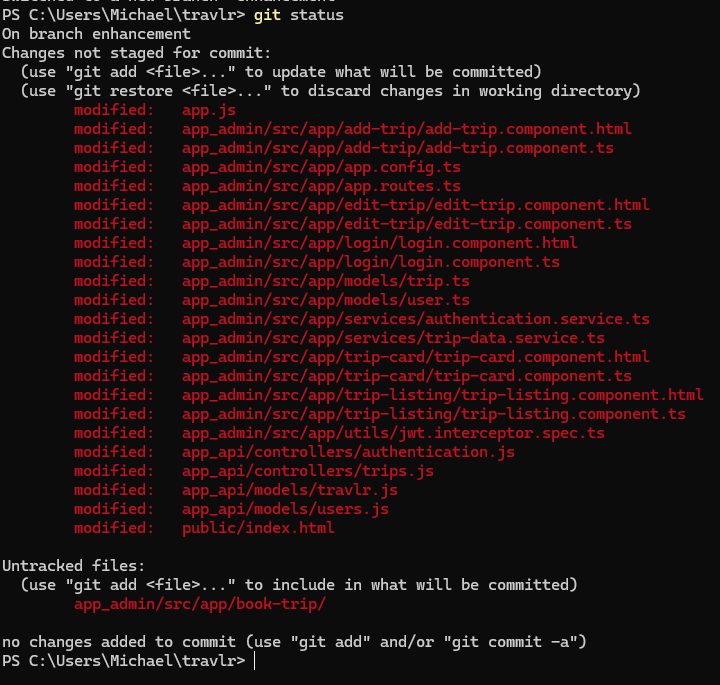


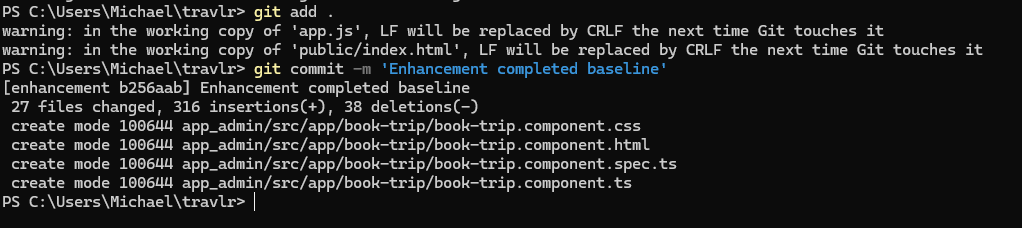
Lastly, I updated the Travlr Getaways Home Page, index.html, to include hreferences to localhost:4200 which is the Angular and Node site. This way, users can navigate from the website directly to where sign in and data entry can begin.

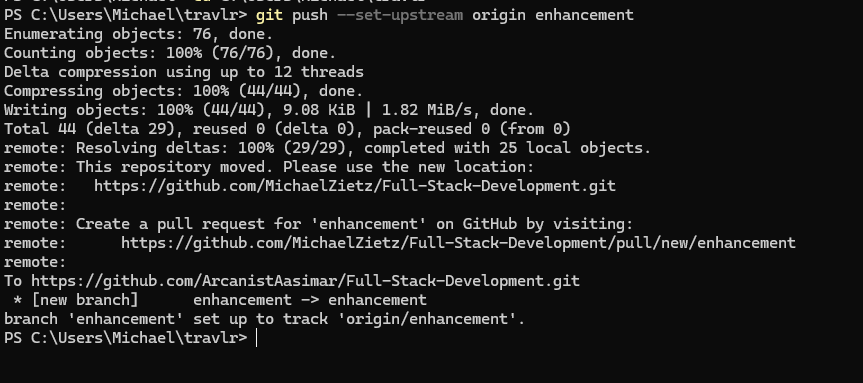
For optimization, time complexity, as well as efficiency of the algorithmic logic, I fine tuned the logic present for each POST, PUT, and GET to ensure the full-stack application performed in a timely manner. Timeouts were set, so in the event data required was missing the Angular services would terminate the attempt without compromising the application or running indefinitely. During my tests on my local network, I was able to secure an average of 3 milliseconds per interaction with the database. This is suitable as these requests would not delay a user to a point where it would hinder their performance accessing and editing data. In the following screenshot it showcases several interaction and requests from the database where trip data is requested and updated within an efficient manner.



Content with the enhancements that have been met, I uploaded the new version of Travlr Getaways’ website to the GitHub repository. This is located in a separate repository from a previous course due to the file’s size. The updated version is stored in the “enhancement” branch.







* Did you meet the course outcomes you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?

As declared in my enhancement outline in Module One, I satisfied the listed improvements and new features to the existing artifact. I corrected known errors due to deprecated libraries with satisfactory replacements, updated comments for clarity, and implemented new features like user roles, trip availability, limited user’s ability to their determined role on the platform and created booking trip functionality. It is through these enhancements I have fulfilled the course outcomes I set out to align this artifact’s enhancements with. They are as follows:

• Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science.

• Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution while managing the trade-offs involved in design choices.

• Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources.

* Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?

During this artifact’s enhancement development, there were a plethora of problems that I came across. I was totally unsure of how to correctly pivot the Travlr Getaways platform away from the deprecated without having to rebuild the authentication process entirely within the limited enhancement time frame. I researched why the HttpClientModule library was deprecated in the first place and what common http intercepts looked like after Angular’s eitheenth version. Before starting this enhancement, I had completed this project for another course that outlined the website’s creation from start to finish, and other than finding fixes for bugs, I was not required to locate infrastructure outside of Angular’s common libraries. Once I ran into the authentication error, I lacked the knowledge to correct it at the time, but since then I’ve educated myself on http intercepts and that they transform a request into an observable event that can allow algorithms to dictate program flow.

In the original enhancement’s outline, I had intended to create different functionality based on user roles that I wanted to create, but I was unsure of the full scope and how it would take form in the program outside the literal functionality I had intended, like booking trips for guests. I had to realize that surrounding components would also need to updated to take these variables into account. Realizing that the app\_api folder’s components needed to be updated as well was a hurdle for me because I would implement logic and features desired, but when running tests, I would not receive errors while the program would not work as intended. An example of this was the new trip variables. The PUT http request would appear successful from the Powershell terminal, but the database would not be updated with the new information despite including them in both the app\_server and app\_admin structure. I then had to piece together that the interface between the two was missing and included the necessary code for the algorithm to work correctly.

Resources

*Angular*. (2024). Angular.dev; Angular. https://angular.dev/api/common/http/withInterceptorsFromDi?tab=description

‌