

Travlr Getaways

# **CS 465 Project Software Design Document**

Version 2.0

## Table of Contents

[**CS 465 Project Software Design Document** 1](#_Toc36198462)

[Table of Contents 2](#_Toc36198463)

[Document Revision History 2](#_Toc36198464)

[Instructions 2](#_Toc36198465)

[Executive Summary 3](#_Toc36198466)

[Design Constraints 3](#_Toc36198467)

[System Architecture View 3](#_Toc36198468)

[Component Diagram 3](#_Toc36198469)

[Sequence Diagram 4](#_Toc36198470)

[Class Diagram 4](#_Toc36198471)

[API Endpoints 4](#_Toc36198472)

[The User Interface 4](#_Toc36198473)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/20/24 | Zachary Nicholas | Updated Executive Summary, Design Constraints, and System Architecture View |
| 1.1 | 06/03/2024 | Zachary Nicholas | Updated the Sequence, Class diagram, and the API Endpoints |
| 2.0 | 06/22/2024 | Zachary  Nicholas | Updated the User Interface portion |

## [Executive Summary](#_heading=h.35nkun2)

In order to develop for this application, I used MEAN stack as well as a combination of node.js and MongoDB. Where node.js is the framework that provides for the server side portion of the website and MongoDB manages the database for the website.

## [Design Constraints](#_heading=h.1ksv4uv)

Some of the major design constraints are the Manpower and Budget that are provided and also the deadline for the project that was setout. I believe these to be the most important in terms of design constraints as if any of these are lacking or set too low it can affect the others, especially the Budget in my mind because if the budget is lacking it will cause us to have Manpower issues which would cause us to have to push back on the deadline, similarly if the deadline is too close to the start date it will cost more budget to hire more manpower.

## [System Architecture View](#_heading=h.44sinio)

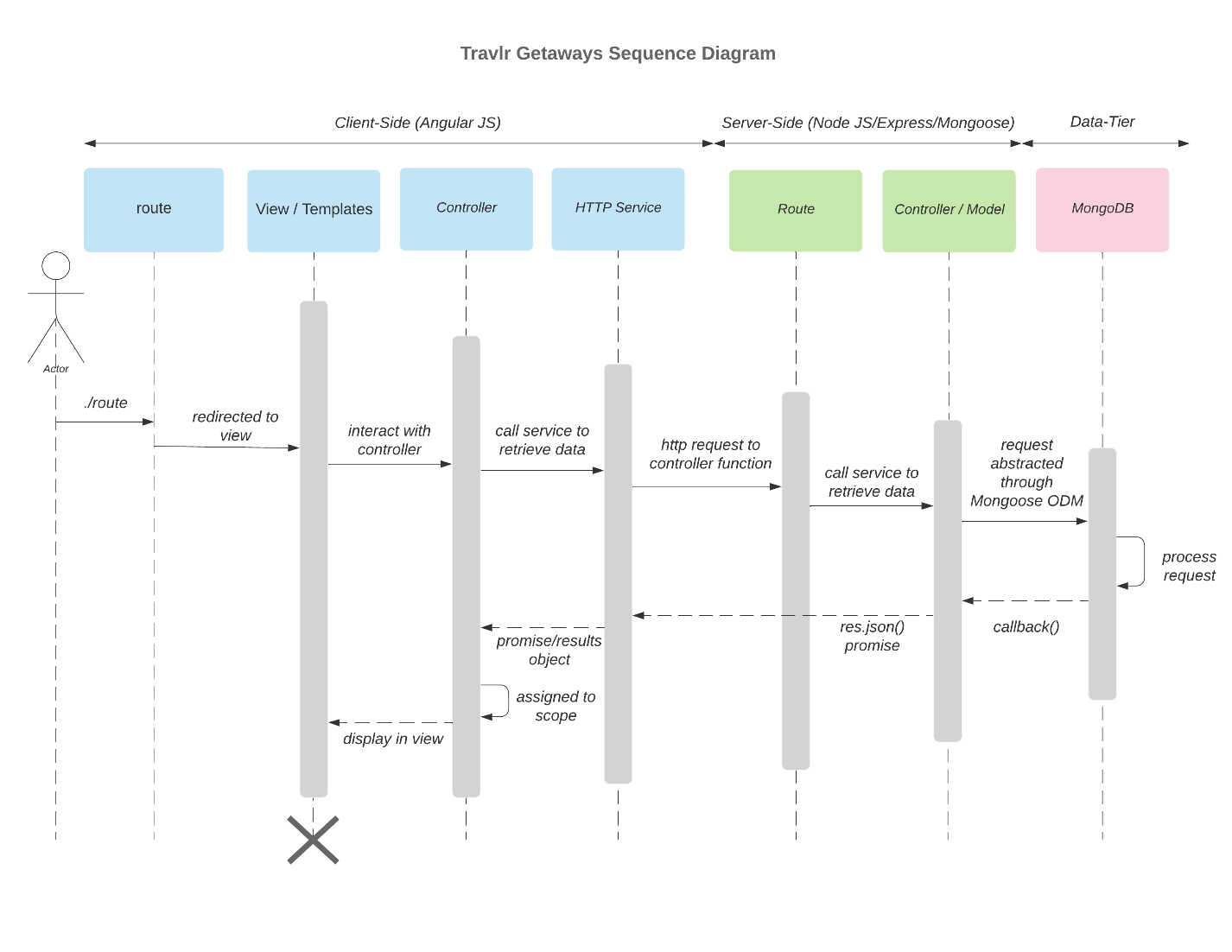
### Component Diagram



A text version of the component diagram is available: [CS 465 Full Stack Component Diagram Text Version](https://learn.snhu.edu/d2l/lor/viewer/view.d2l?ou=6606&loIdentId=24342).

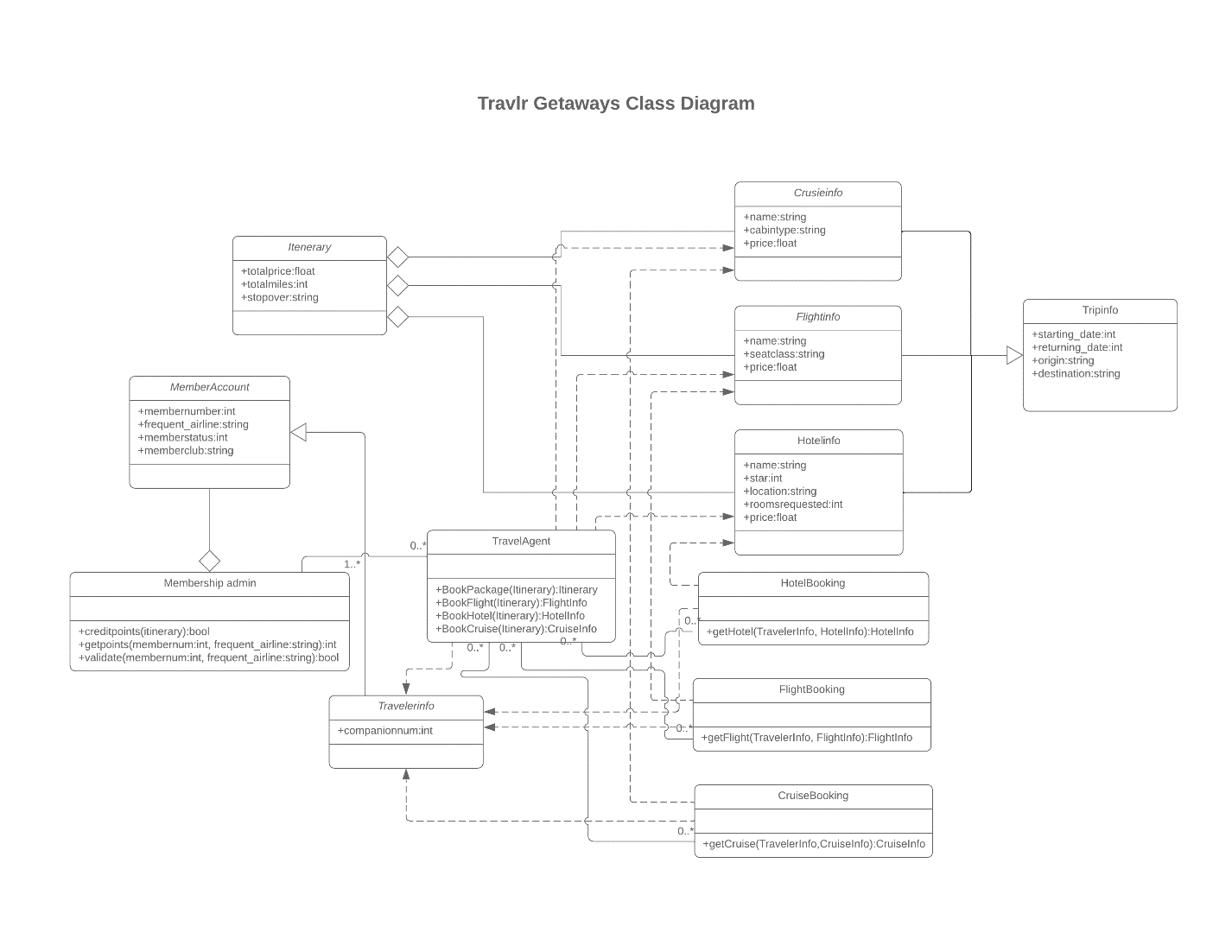
The critical parts that I can see in this diagram are the server, the database, and the client. The client and the server are connected together using ports. The database will feed the server information that will then be served to the client. The server is responsible for authenticating the clients that will connect to it as well as store information provided by the database and passes it to the client.

### Sequence Diagram



When Looking at the sequence diagram it starts with the actor to start the action, the way it flows from here is that the actor or end user will enter the route and is subsequently looks at the view / template section of our site because of the front end router. Following the user looking at the view / template section it will call for the specific controller that controls that portion of the site and applies it and returns it back to the user or actor. Following that the controller responsible for the front end will make calls to the HTTP service in order to retrieve the pieces of information. The way that this is done is the HTTP service connecting the frontend to the backend by making API calls to the specified routes. Again following this the route for the backend will find the correct controller which in turn will query the database if necessary and return the data for the request back to the front end more specifically the HTTP service which will then apply it to a template and make it viewable for the user.

## Class Diagram



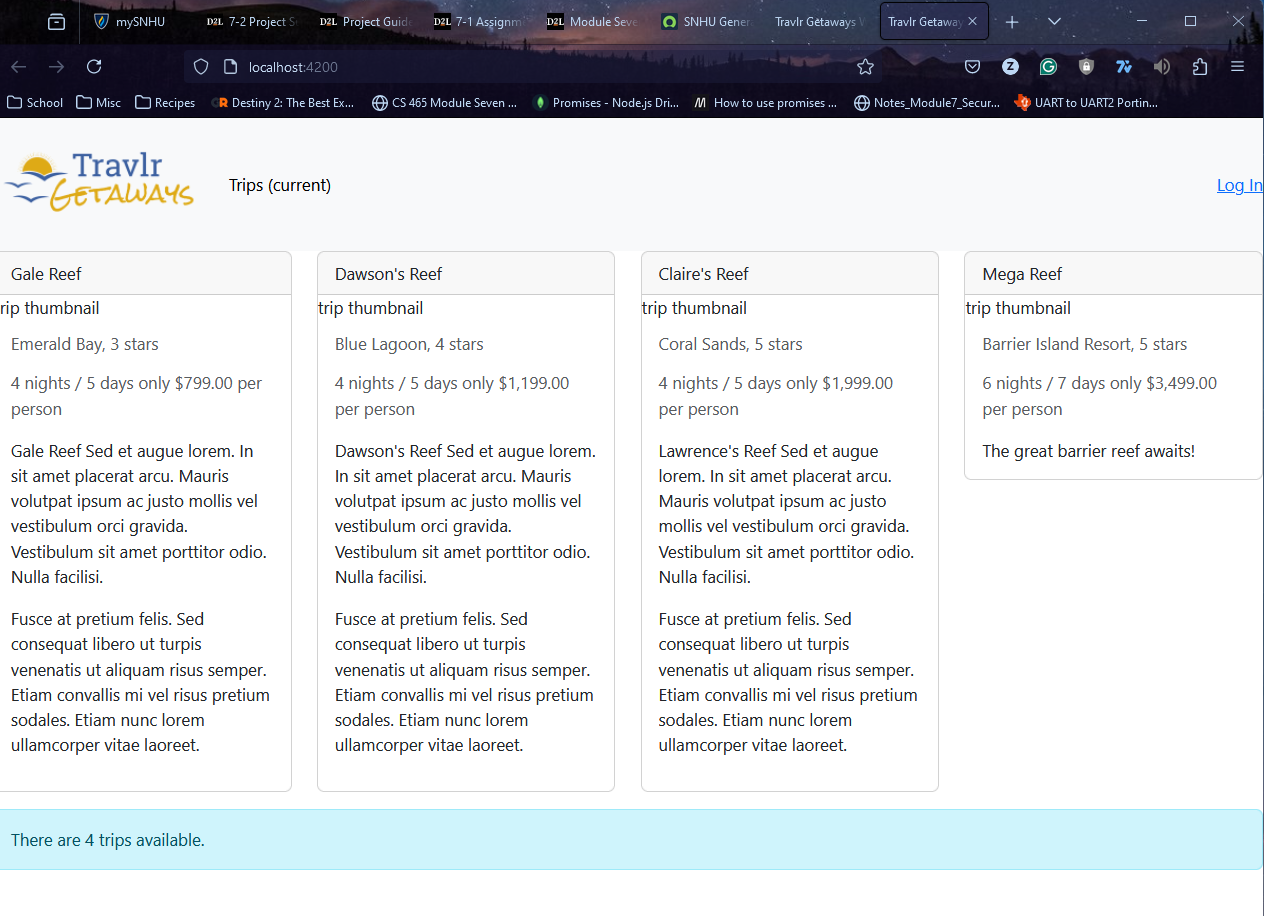
When we look at the class diagram, we can see that the classes (Cruise info, Flight info, and Hotel info) contain specific data such as their name property and price, as well as information that they inherit from Trip info which adds things like a start and end date as well as a origin and destination.

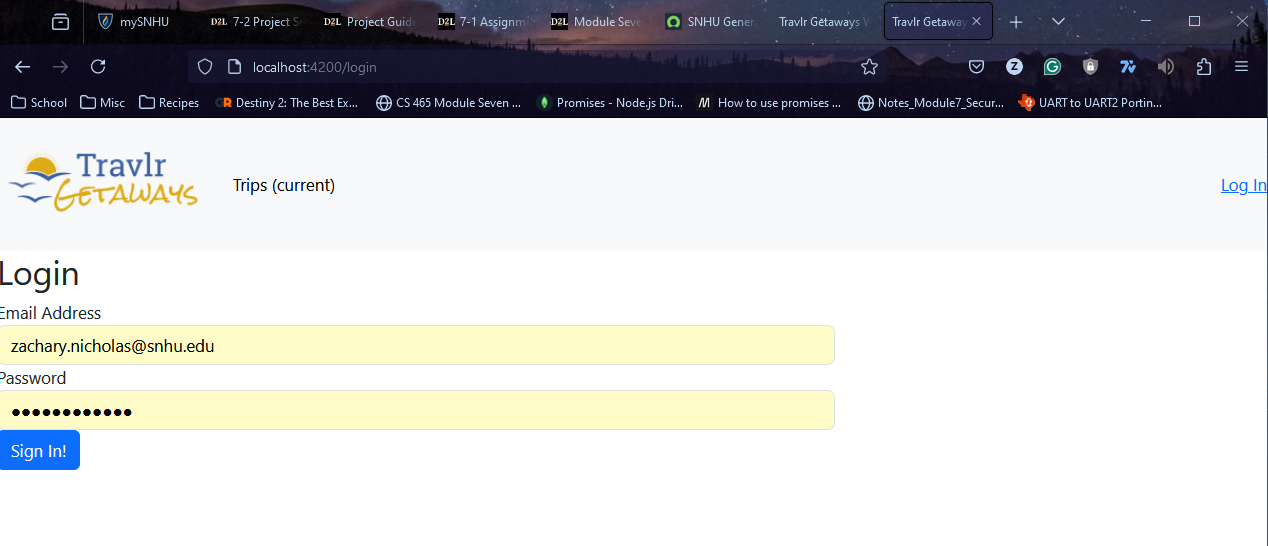
## [API](#_heading=h.2jxsxqh) Endpoints

| **Method** | **Purpose** | **URL** | **Notes** |
| --- | --- | --- | --- |
| **GET** | <Retrieve list of things> | </api/things> | <Returns all active things> |
| **GET** | <Retrieve single thing> | </api/things/:thingId> | <Returns single thing instance, identified by the thing ID passed on the request URL> |
| **POST** | <Creates a new list of things> | </api/things> | <Creates a new list of things> |
| **POST** | <Creates a single thing> | </api/things/:thingID> | <Creates a single item or thing> |
| **PUT** | <Updates a list of things> | </api/things> | <Replaces and changes a list of things (which will override everything in list)> |
| **PUT** | <Updates a single thing> | </api/things/:thingID> | <Replaces and changes a single thing based off ID> |
| **PATCH** | <List of things Modified> | </api/things> | Updated the record with only values passed to it |
| **PATCH** | <Single thing Modified> | </api/things/:thingID> | Updates a single thing with records |
| **DELETE** | <List of things Deleted> | </api/things> | List of things deleted |
| **DELETE** | <Single thing Deleted> | </api/things/:thingID> | Deletes a Single thing |

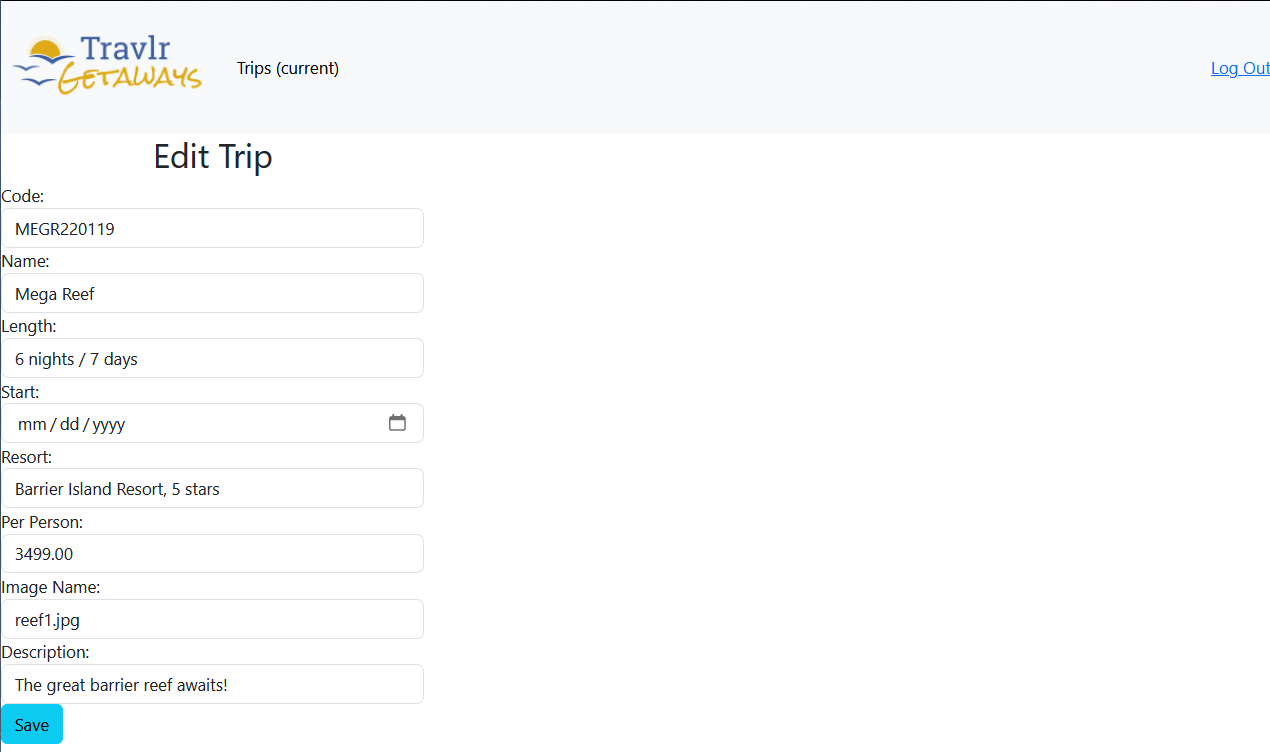
## The User Interface

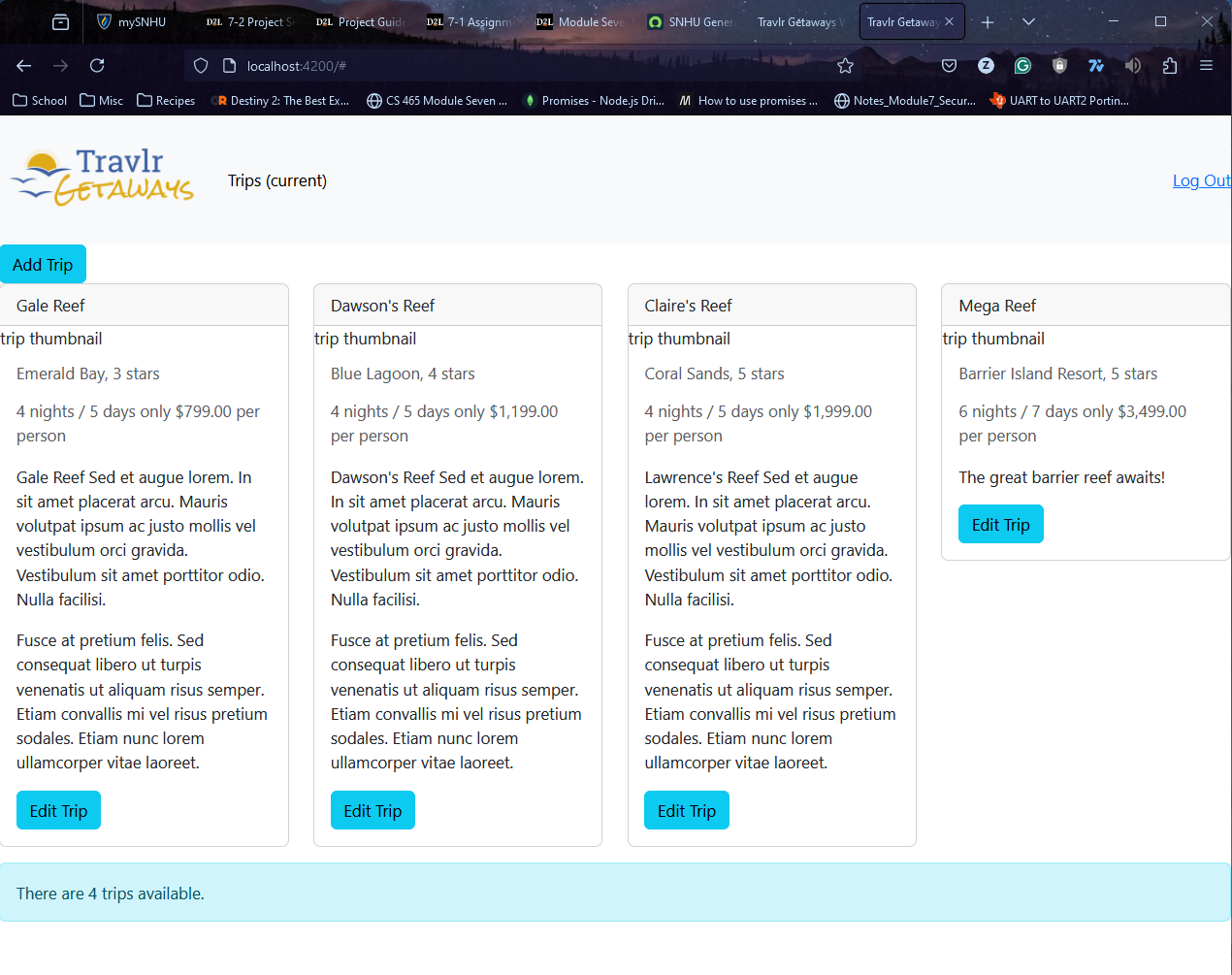
**User login page**





**User adding and updating a trip**





The Angular project is an open-source, JavaScript framework written in TypeScript. It is helpful because it allows different companies build scalable web-based application. The Express project is also a open-source framework, but it is built for the back-end of a website and specifically used Node.js and is written in JavaScript. It is useful for since it has parts specifically for MEAN.

Using SPAs is a good choice as they support a rich client-side functionality where they don’t need to be reloaded when a user interacts or navigates through the application, One of the main benefits of an SPA is that they can load quickly and get the data needed in the background (as long as your internet connection is fast enough).

To start testing our application, I found the easiest way to do this was to use postman which allowed us to test the routing to the API first. With postman we were able to check all of the HTTP requests to our API which included logging in and retrieving data from our database. Another thing that postman allowed us to do is test the GET and PUT without having to use the Web GUI. The final way to test for our API is to use the Web GUI to do the same adding and changing for our trips on our Admin site.

References:

*Angularjs vs expressjs: What are the differences?*. StackShare. (n.d.). https://stackshare.io/stackups/angularjs-vs-expressjs

Codecademy. (n.d.). *What is express.js?* https://www.codecademy.com/article/what-is-express-js

Deshpande, C. (2024, May 29). *What is angular? master this powerful framework [2024]*. Simplilearn.com. https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular

Single-page applications (spas) — what they are and how they work. (n.d.). https://business.adobe.com/blog/basics/learn-the-benefits-of-single-page-apps-spa

*What is the difference between Angular & Express?*. Acquaint Softtech Private Limited. (n.d.). https://acquaintsoft.com/blog/angular-vs-express

Writer, C. B. T., BasuMallick, C., Writer, T., 18, L. U. O., Chiradeep BasuMallick                              Technical Writer                                                                                  opens a new window                                                          opens a new window          , Chiradeep BasuMallick                              Technical Writer                                                                                  opens a new window, Chiradeep BasuMallick                              Technical Writer, & opens a new window                          Chiradeep is a content marketing professional. (2022, October 18). *Pros and cons of single-page applications*. Spiceworks Inc. https://www.spiceworks.com/tech/devops/articles/what-is-single-page-application/