

Travlr Getaways Web Application

# **CS 465 Project Software Design Document**

Version 1.0

## 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 | 09/12/2023 | Vincent Snow | Executive summary, design constraints, components |
| 2.0 | 09/30/2023 | Vincent Snow | Chart, revisions to summary and constraints |
| 3.0 | 10/14/2023 | Vincent Snow | API endpoints, screenshots, Angular & Express explanation |

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

The application requirements are as follows:

- Users must be able to create an account and view their itineraries.

- Users must be able to search travel packages by location and price.

- Users must be able to book packages on the website.

- Administrators must be able to log in to a private admin area of the site.

- Administrators must be able to manage users, travel packages, and their details.

The web application will be built using MEAN – a group of javascript technologies that work together to render web pages with dynamic (changing) parts securely and quickly while minimizing repetitive code and file sizes and allowing modular pieces of the code to be reused and edited easily. The MongoDB database will securely store data about the travel packages and users in the application and easily integrate with the other technologies used. Express is a popular web application framework within NodeJS, an open-source javascript runtime environment for multiple platforms that helps quickly build APIs. Angular is a javascript framework for creating reactive, dynamic single-page web applications. The use of this stack together is very popular and therefore has plenty of community support and experience.

The customer area and admin area of the website will be separated by user privilege levels so that the admin area will not be accessible to customers or anyone without administrator permissions and credentials. Customers will be able to create an account for themselves and add booked travel packages to it, while administrators will be able to interact with the entire database to edit user accounts and travel packages.

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

* The application must not allow unauthorized access of user data.
* The application must encrypt sensitive user information according to legal standards.
* The appearance of the application must include the company’s logo and trademarks.
* This application may not run well on mobile devices due to the technology being used.
* Customers cannot book a travel package without logging into an account.
* Heavy loads on MongoDB may cause loss of records
* MEAN stack may not be able to support large-scale heavy usage

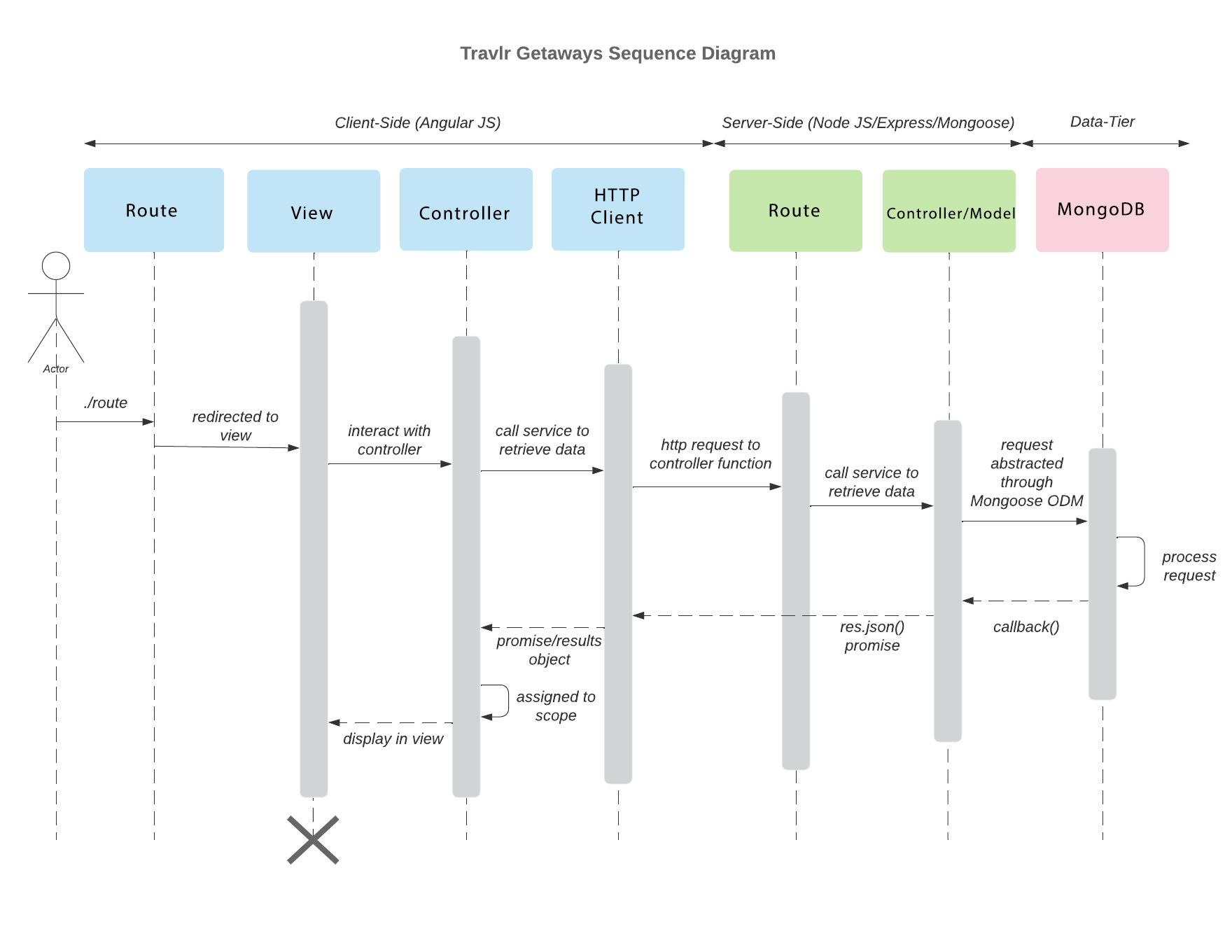
## [System Architecture View](#_heading=h.44sinio)

### Component Diagram



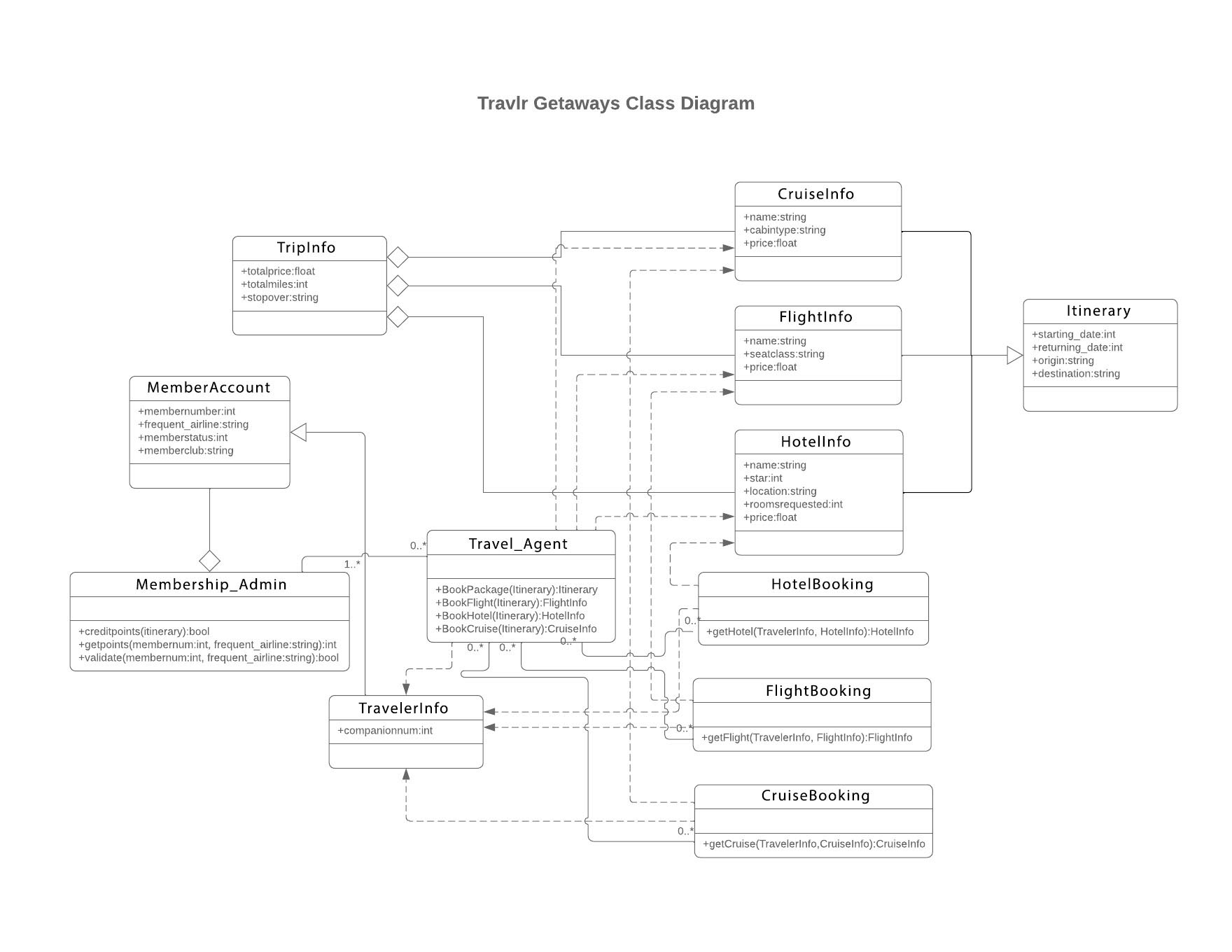
### The three components of the system are the client, the server, and the database. A client, the user or customer, opens the application using a web browser on their device. Through HTTP, they request to see web pages held on the server. The server pages may send a request for information about users and travel packages from the database, and then send that to the user as well to render the webpage interface. Each time a request is made by one major component to the other, an authentication check will be made for security to ensure that a legitimate authorized user receives the data.

### Sequence Diagram



The user sends a route request using their browser, which redirects them to a view and might also call the controller to send methods or data to use with it. From this view, they can interact with various elements that send another request for a route. These routes may call the server to retrieve data or direct the user to another view. If calling the server, an HTTP request is used, and the server’s controller uses a model to format the data being sent to or retrieved from the database. This data is sent and retrieved in the form of a JSON object and may be passed back to the client and displayed in the view.

## Class Diagram

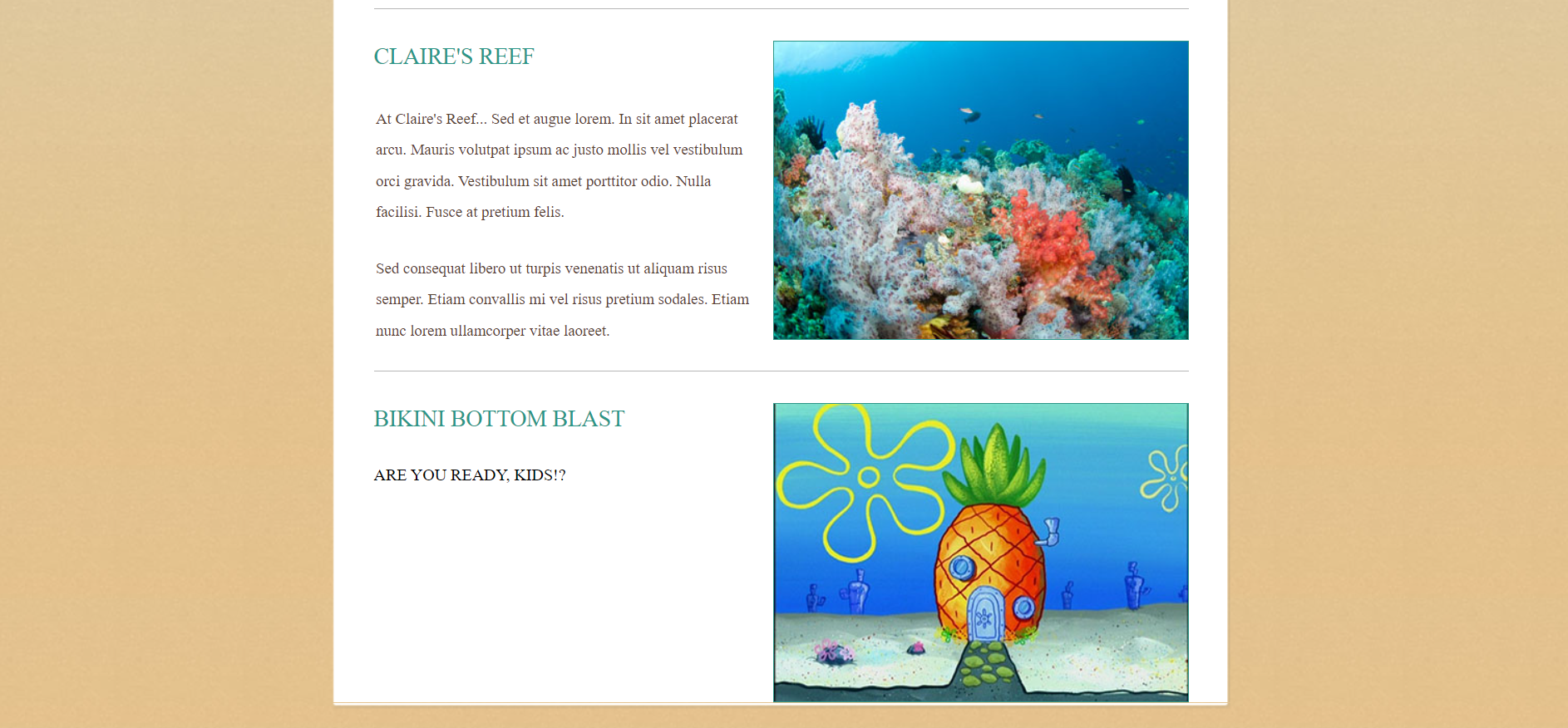


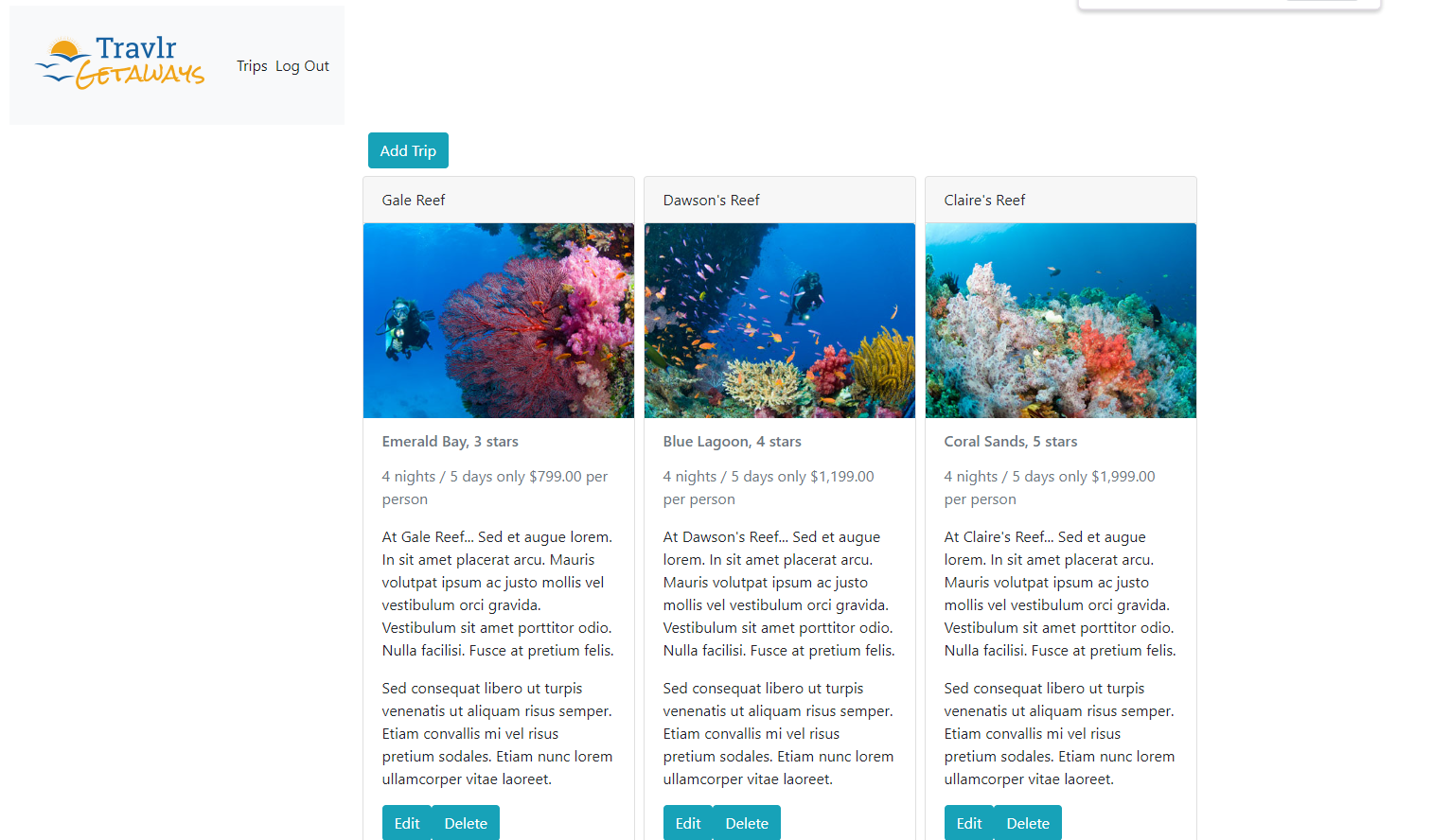
Each MemberAccount represents a user’s account. It has a Membership\_Admin class connected to it that validates their points for bookings. Booking methods are held in the Travel\_Agent class, which is used to make and view bookings for a traveler & companions. The Itinerary class holds all details for the trip: the Cruise, Flight, and HotelInfo classes and the details of those in a TripInfo class.

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

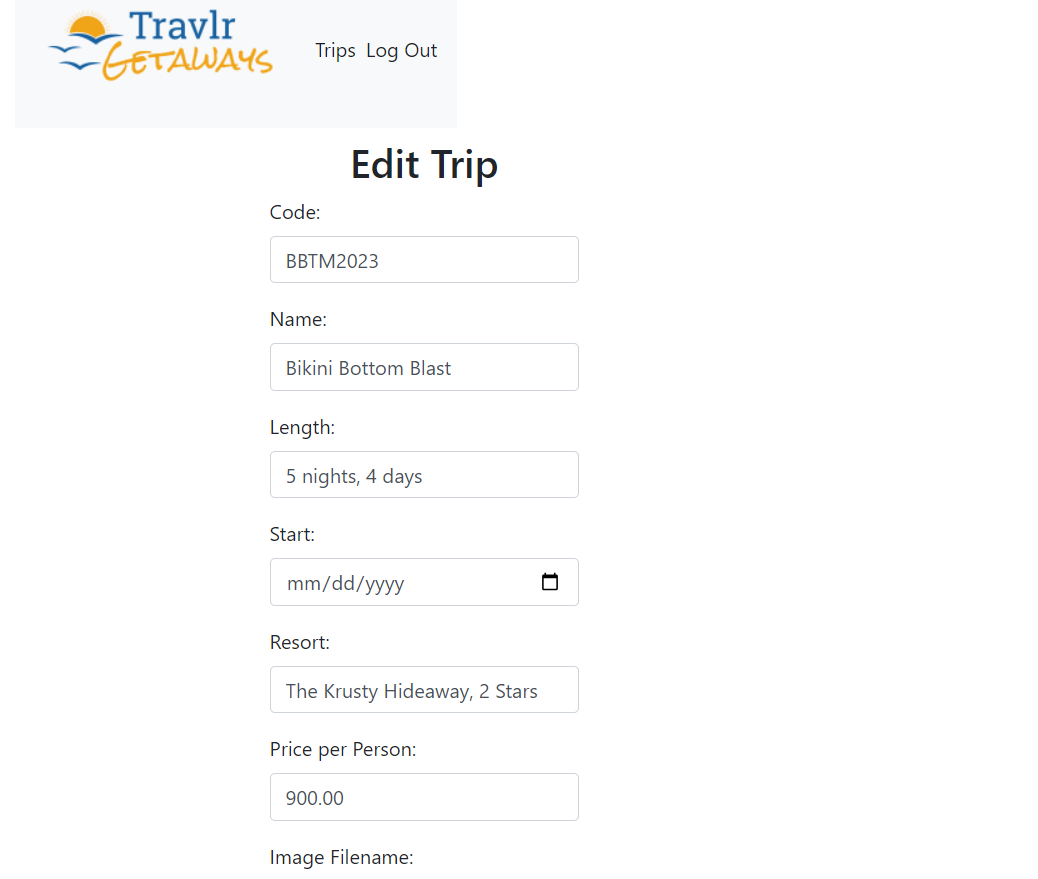
| **Method** | **Purpose** | **URL** | **Notes** |
| --- | --- | --- | --- |
| **POST** | Log in a user | api/login | Validates against user schema and returns an auth key |
| **POST** | Register a new user | api/register | Returns an auth key & creates new user document in database |
| **GET** | Get the list of trips to render on the travel view or use in other functions | api/trips | Returns the list of trips in the trips db collection |
| **POST** | Add a new trip to the user’s account | api/trips | Creates a new document in travlr db trips collection |
| **GET** | Get a particular trip to modify | Api/trips/:tripCode | Returns a trip object that matches the code |
| **DELETE** | Erase a trip package from the database | Api/trips/:tripCode | Deletes the trip document with the matching code key |
| **PUT** | Add a new trip package | Api/trips/:tripCode | Creates a new document in the trips collection of travlr db |

The User Interface

A custom trip added to the database and displayed in the client site.  


The trip edit/admin page:  


The update screen on the admin site:



Angular organizes the code into services and components that can work together, be reused, and accessed from any other file. While somewhat similar to the way the Express front-end works with handlebars injecting code snippets into pages, Angular has more tools for organizing and instantly creating different types of useful elements for a page.

A simple web application might live up to its name by being simple, but the same thing in a simple web application format with static html pages requires much longer documents and repeating code throughout each of the pages. Using an SPA built with Express means that we can reuse pieces of code and separate the concerns for client and server into different folders for organization. The data we need is really rendering on one homepage and being injected into it to display dynamically. It requires a bit more work installing tools and knowledge in how to use these various tools and libraries, however.

Thorough testing of an application involves incrementally testing that each component works by running it locally, using automated tests in the code, and tools such as POSTMAN to determine if the routes themselves are functional. When encountering a problem, we will step through the code to find where an error in logic or typo could be stopping it from functioning if no specific error message is given on the console.