

Travlr Getaways Website

# **CS 465 Project Software Design Document**

Version 1.1

## 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 4](#_Toc36198468)

[Component Diagram 4](#_Toc36198469)

[Sequence Diagram 5](#_Toc36198470)

[Class Diagram 5](#_Toc36198471)

[API Endpoints 5](#_Toc36198472)

[The User Interface 5](#_Toc36198473)

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

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 09/22/2024 | Mitchel Dauk | Summary, constraints, and component diagram |
| 1.1 | 10/06/2024 | Mitchel Dauk | Sequence diagram, Class diagram, and API Endpoints |
| 1.2 | 10/15/2024 | Mitchel Dauk | User Interface |

## Instructions

Fill in all bracketed information on page one (the cover page), in the Document Revision History table, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

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

The travel booking website being constructed for “Travlr Getaways” allows the user to navigate a webpage to ultimately book their dream vacation. The customer should be able to create an account, search for travel packages, search prices, book reservations, and visit the website for an itinerary. There should also be an admin-only site to maintain the customer requests.

The website uses the MEAN stack. The MEAN stack consists of four parts; MongoDB, Express, Angular, and Node.js. MongoDB will work as the database in the back-end and is extremely fast and scalable. Node.js and Express work as an application server to allow the programmer to code what will be included in the website. Node.js specifically works as its own HTTP server so web applications can be built upon it. Angular works as a front-end for users to navigate the website properly. There will also be a front-end webpage that will only be obtainable for admin. All parts of the MEAN stack work well together to provide a smooth and seamless process that works well for the programmer and the user.

In order to architect the MEAN stack, a REST API conjoins into an SPA. REST API stands for representational state transfer and application program interface (Harbor, Holmes, 2019). REST API basically communicates with SPA to return data when it is needed. With quick access and fast protocols, JSON works well to store and exchange data from the programmer, to the user, and eventually to the admin.

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

There are many problems or negatives that can cause the MEAN stack to improperly function. These can include:

Front-end experience – Angular is used in the MEAN stack, but it doesn’t necessarily work with all users due to its use with JavaScript. HTML is rendered, so any issues with JavaScript or support for the language could cause issues. Some search engines may also refuse to run JavaScript (Harbor, Holmes, 2019).

Database – The database used is MongoDB. Sometimes if large files are being processed, it can bog the process down, or even potentially cause a loss of records. Losing records could cause severe issues in the database and some that may not be retrievable again (Knowledgenile, 2023).

Errors and bugs – There may be errors in the JavaScript files that allow most of the website to work, until the user accesses something that causes an issue. With JavaScript, the MEAN stack uses a rendered HTML, so there may be unknown bugs or errors within the code that cause unknown disturbances in the website. This could also be detrimental to security as well.

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

### Component Diagram



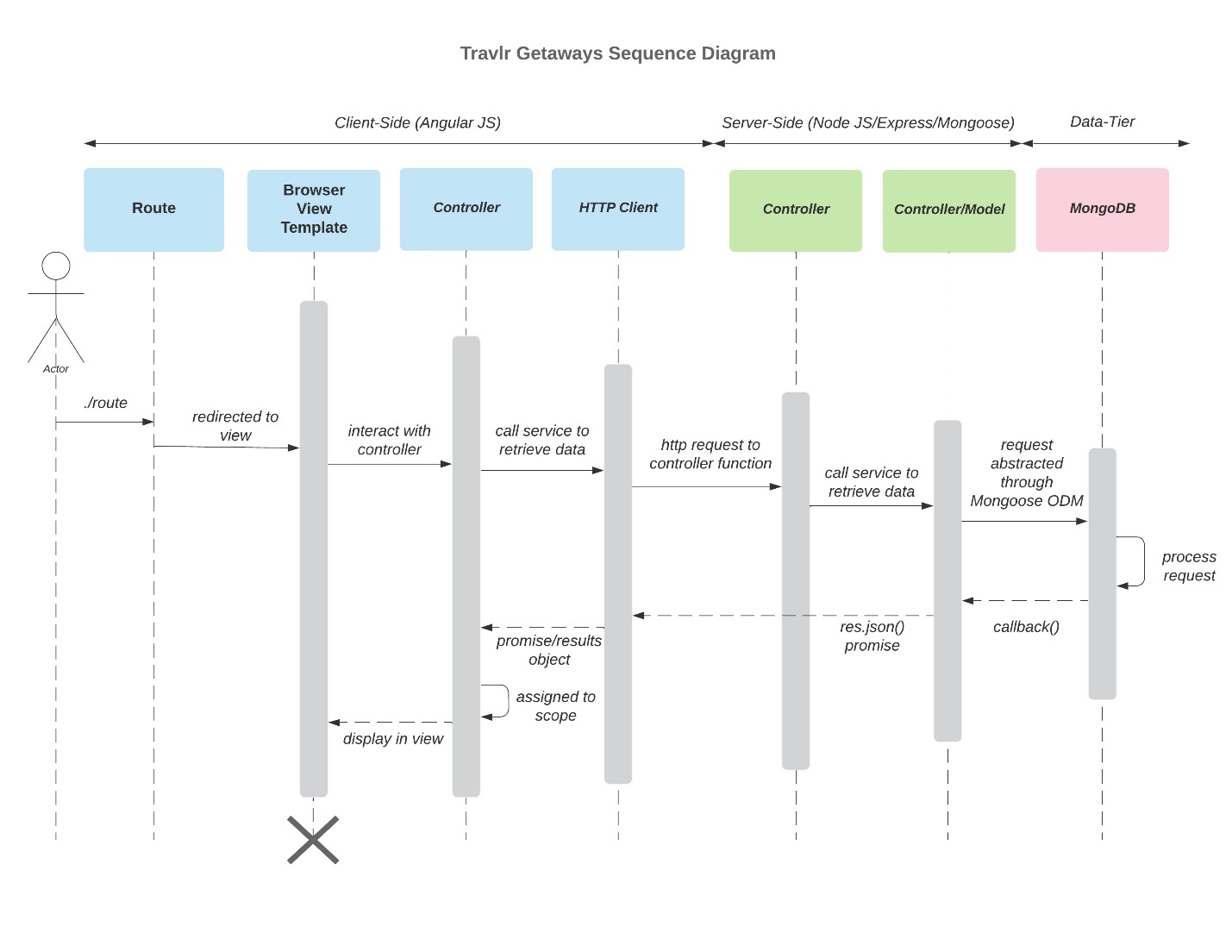
A text version of the component diagram is available: [CS 465 Full Stack Component Diagram Text Version](https://learn.snhu.edu/d2l/lor/viewer/view.d2l?ou=6606&loIdentId=24342).

The component diagram above displays the components of the MEAN stack and how they relate to the software system as a whole. The database uses the M in MEAN as MongoDB. The server uses the E and N in MEAN as Express and Node.js. The client uses the A in MEAN as Angular. All three components are inter-connected to each other, as they all work together. Both the client and the server must be able to access the database. The client and server must also be able to communicate with each other.

Within the server component, Mongoose is used to communicate with MongoDB. Together, they allow the server to access the database. Mongoose is also connected to the server session, which is then connected to the traveler database and authentication server. The client component uses the traveler portfolio to be stored and accessed in the MongoDB database. The traveler portfolio is also connected to the graphic library, web browser, and client session. The client session must be able to access the authentication server to proceed to the website.

The server and database work together through a REST API, and eventually into an SPA. This helps to communicate and send JSON data to allow the website to work for the user. The client uses Angular to pull data and push it back into the REST APIs, through the front-end.

### Sequence Diagram



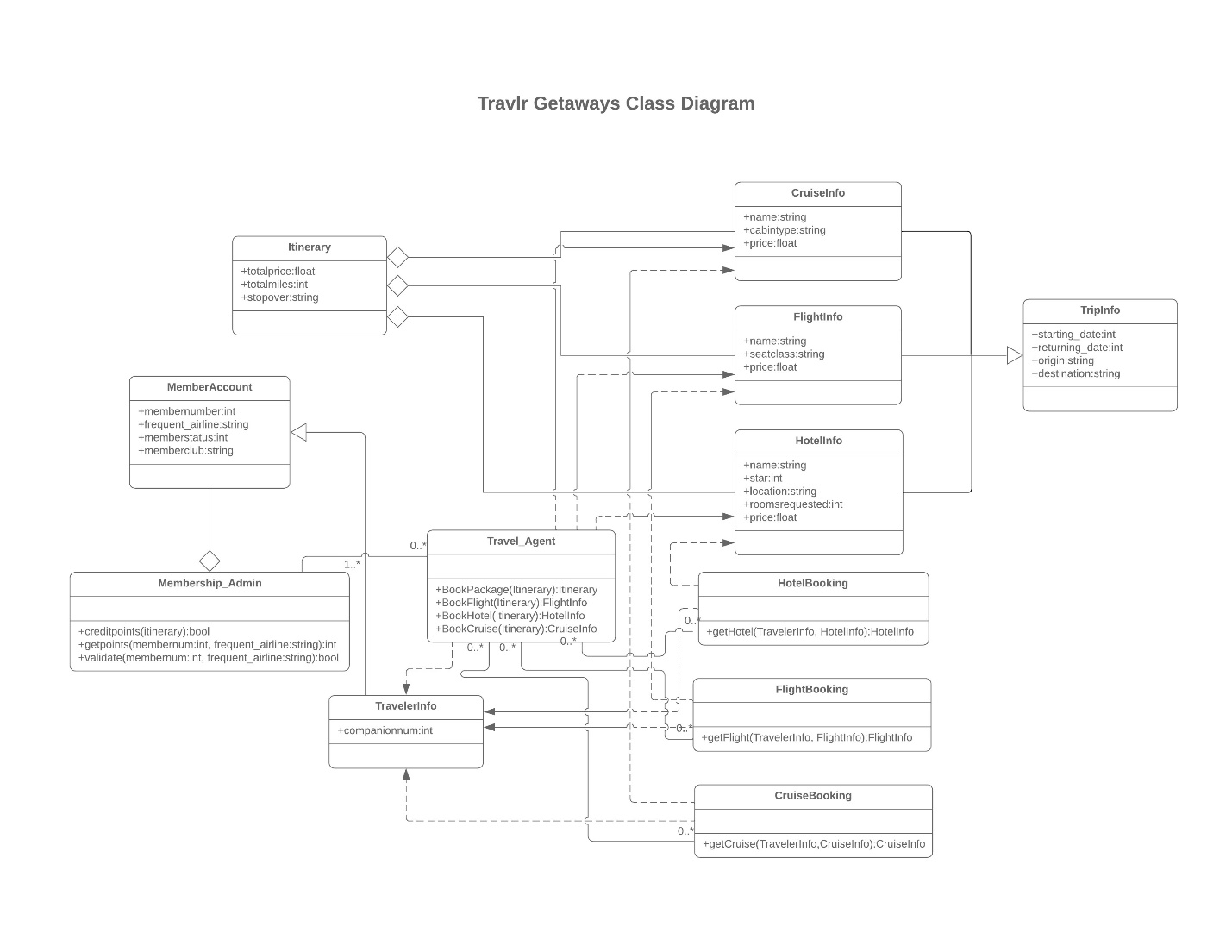
The Sequence Diagram represents the flow of the web application from the client-side, to the server-side, and finally to the data-tier. The Sequence diagram portrays an image of how the user is navigating the website and the steps to access the database of information.

The actor will first be connected through a route request and address to the browser/view/template. The actor will then be able to interact with the controller and finally call the HTTP client to retrieve data. At this point, the actor is signing in and attempting to access the trip information.

The request is then made to interact with the server-side controller. The controller/model can then interact and call to receive the data to be given to the actor. The database is finally reached through Mongoose and the request is granted or denied.

When the information makes its way back to the client, the credentials and options are determined to display the information for the actor in the browser.

## Class Diagram



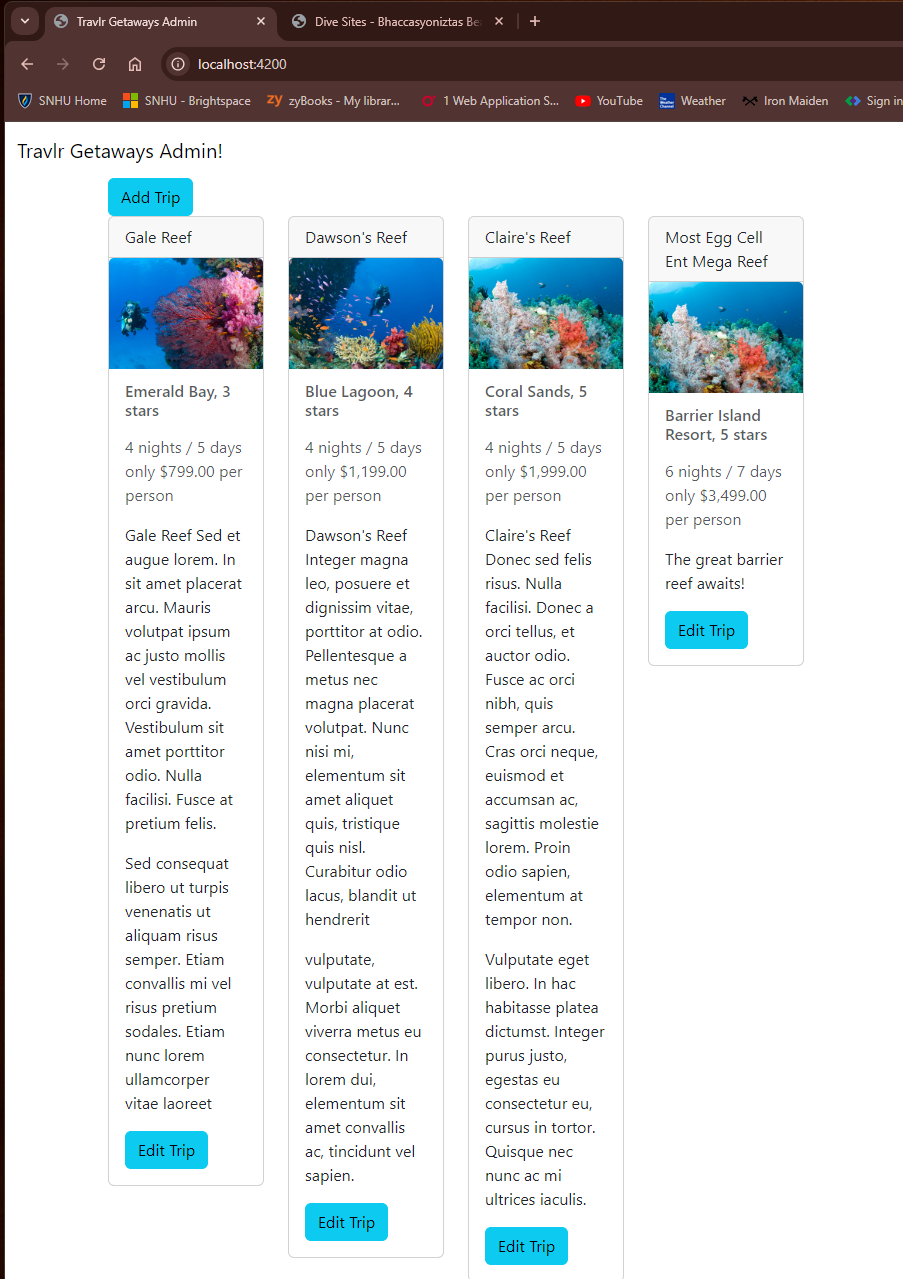
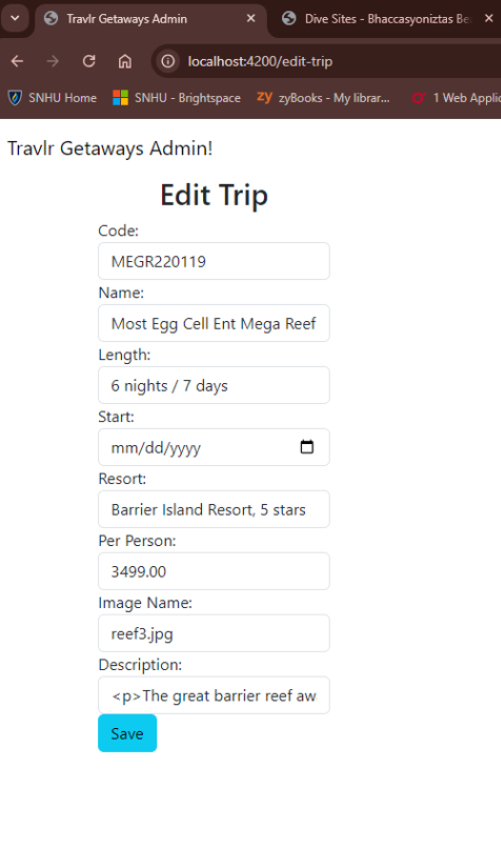
The Class Diagram shows the flow of finding travel information, booking a trip through a travel agent, accessing the member account, and creating an itinerary.

The user will choose a cruise, flight, and hotel, based on their preferences. All of the options chosen will be updated into the user’s trip info, including start times, returning times, origin, and destination. Their travel agent will book the hotel, flight, and cruise and combine it into their traveler info. The member’s account will be updated and admin can access it for points and validation. The itinerary shows a connection through aggregation to the trip info, traveler info, and member account.

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

| **Method** | **Purpose** | **URL** | **Notes** |
| --- | --- | --- | --- |
| **GET** | Retrieve trips | api/trips | Returns all trips |
| **GET** | Retrieve single trip | api/trips/:tripcode | Returns single trip using “tripcode”. |
| **POST** | Add a new trip | api/trips | Creates a trip. |
| **DELETE** | Delete single trip | api/trips/:tripcode | Deletes single trip using “tripcode”. |
| **DELETE** | Delete all trips | api/trips | Deletes all trips |
| **PUT** | Update single trip | api/trips/:tripcode | Updates a single trip using “tripcode”. |
| **PUT** | Update all trips | api/trips | Updates all trips. |

## The User Interface



In this full stack development, a normal HTML page has been transformed into a functional Angular project. The Angular project structure is different from an Express HTML page by the steps involved and the overall outcome. The Angular project structure takes more work, but it is more efficient, with better functionality and security.

The singe-page application, or SPA, has advantages and disadvantages when it compares to an Express HTML webpage. The biggest advantage of SPA is that it has fast loading times. Instead of loading the whole page over again when something is changed, the individual part of the page can be updated automatically. Since only a small portion is changed at a time, it cuts down on how much bandwidth is used (Kaur, 2024).

The disadvantages are that SPA does not connect well with every program. Even though it works great with Angular, other applications may cause slower speeds or disruption of service. There also may be security issues that allow an easier way for an intruder to introduce themselves into confidential data.

The process to make sure SPA is working, can be conducted through tests with various platforms. For this development, Postman was used to verify that GET, PUT, and POST all worked and the proper information was showing up. In the PowerShell window, often times if there is a noticeable error, there will be a basic explanation of the issue. The main errors I ran into was the back-end not communicating properly, or sometimes at all. This can be frustrating, but often times the problem is laid out for the programmer to dissect and change what isn’t linking correctly.

A question I still have is how to upgrade the site and allow the user to do more than just basic tasks. I realize that this is a prototype and there are only basic designs, but when there is more added to the site, there is more bandwidth being used, which contributes to more work being done continuously on the page.

Resources

Holmes, S., & Harber, C. (2019). *Getting mean with Mongo, express, angular, and node Simon Holmes, Clive Harber*. Manning.

*Lets understanding mean stack, its advantages, disadvantages, and use cases*. Lets. (2023, July 3). https://www.knowledgenile.com/blogs/understanding-mean-stack-applications

Kaur, A. (2024, April 26). *What is Single Page Application (SPA)? pros and cons with examples*. Net Solutions. https://www.netsolutions.com/insights/single-page-application/