

Travlr Getawas

# **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 | 05/25/24 | Jamie Javis | Preparing documentations for Travlr Getaway |

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

The Travlr Getaways web-based application is aimed to offer a effortless vacation planning experience for users while providing the company with the proper tools to manage their application and the information. This application will need to use MEAN stack (MongoDB, Express.js, Angular, and Node.js) to guarantee an accessible, effective, and strong application.

**Mean Stack**

The use of this MEAN Stack for this web application provides the application with a modern Java Script full stack method. This approach allows for frontend and backend integrated development and launch. The important parts of this application will include Angular for the frontend framework for a responsive user interface, Node.js and Express.js for a quality backend framework that’s known for its scalable and responsive structure, and MongoDB to handle large amounts of user data and will increase performance as the application grows.

**Appropriate Architecture**

For the front-end architecture, the use of Angular is needed. This will ensure that the customer facing application will be responsive. As a Single Page Application, customer will be able to sign up or into their account, search through packages or for specific packages based on the price or location, book their vacation stay and reserve the travel package, and be able to edit and oversee their itinerary lists.

For the Administrator Single Page Application side, building in Angular is also needed. Much like the customer side, this will allow the admins to be responsive with managing their platform. Admins will be able to manage the customer accounts and their data, set and change their package pricing information, and edit and maintain the information for the site and package information. In this admin side, the admins will be able to see data information, content update information, and user management while still being simple and user friendly.

**Backend Information**

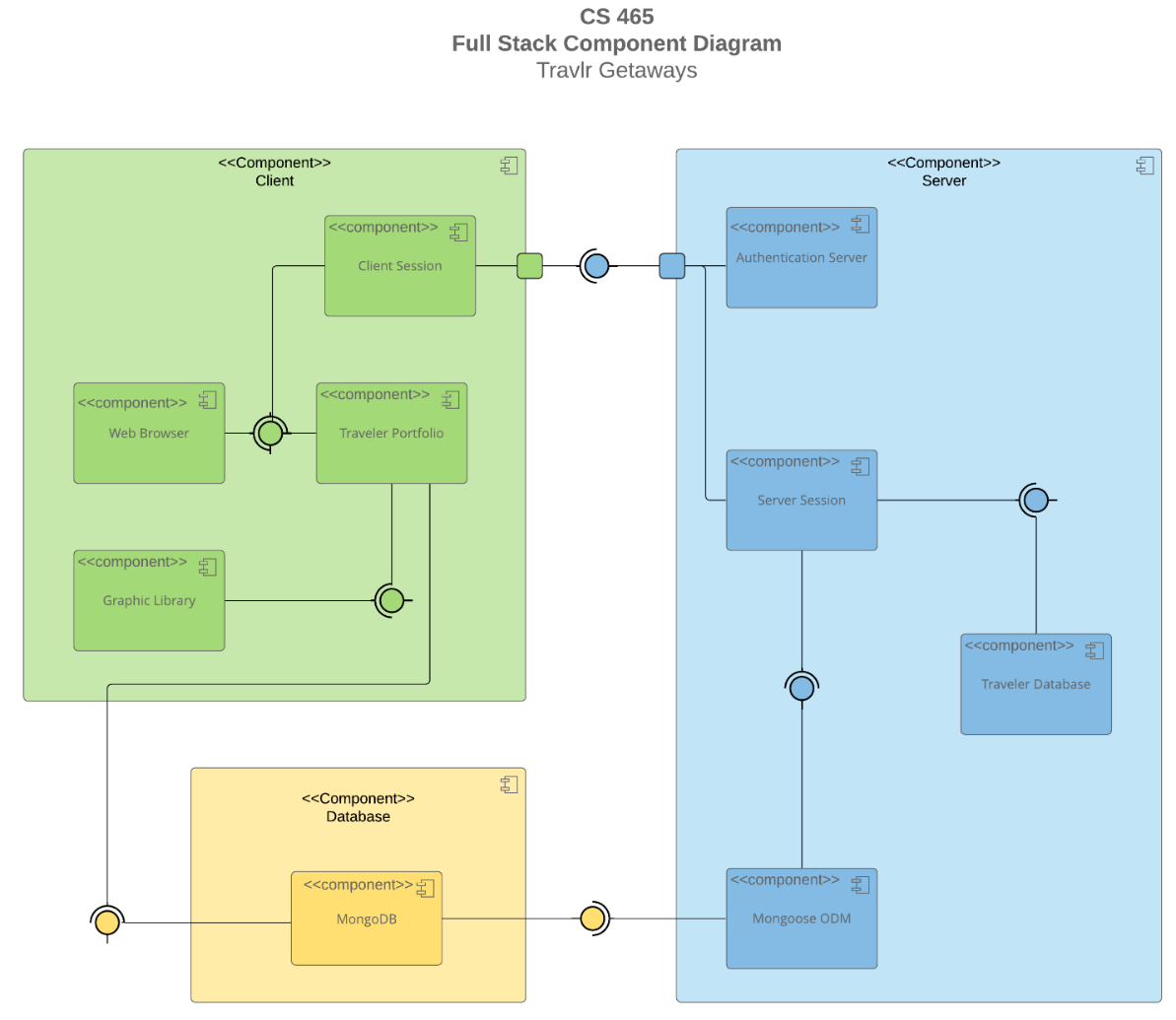
Within the backend side of the application, we will be utilizing Node.js servers and express frameworks to make this application a reliable and maintainable system. With this base, the company will be able to process user and travel data, edit and maintain the database, and manage user input/inquiries. MongoDB use will be important for data retrieval and storage, complex information inquiries and is super scalable as the company advances.

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

For the Travlr Getaway web-based application, there’s a few constraints needed to deliver the best product outcome for the clients. Security controls are needed when dealing with user data and potential payment information. The application will need the be scalable in order to handle large amounts of user data, requiring a database to store the data and API routing for the incoming data. The application needs to be able to perform well for a friendly user experience based on API response to the data load. To maintain the user friendly experience, both the customer and admin frontend sites will need a strong Single Page Application interface. With a strong SPA, this will allow both parties to have a frieldy interface that allows users to have a responsive application.

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

### Component Diagram



In this diagram we can see the client server, and database side of the application. It shows us how the data will move inside of the system. The client side has the Client Session where users are managed in the browser, they have the User Interface in the web browser, the Traveler Portfolio where user information is kept and the Graphical Library for graphics. In the database server side, the NoSQL database will be used to store user data and send it to both the customer and the admins. The Server side contains the information to authenticate the users, manage user sessions and the information of data, and the traveler database that has the users travel info stored.

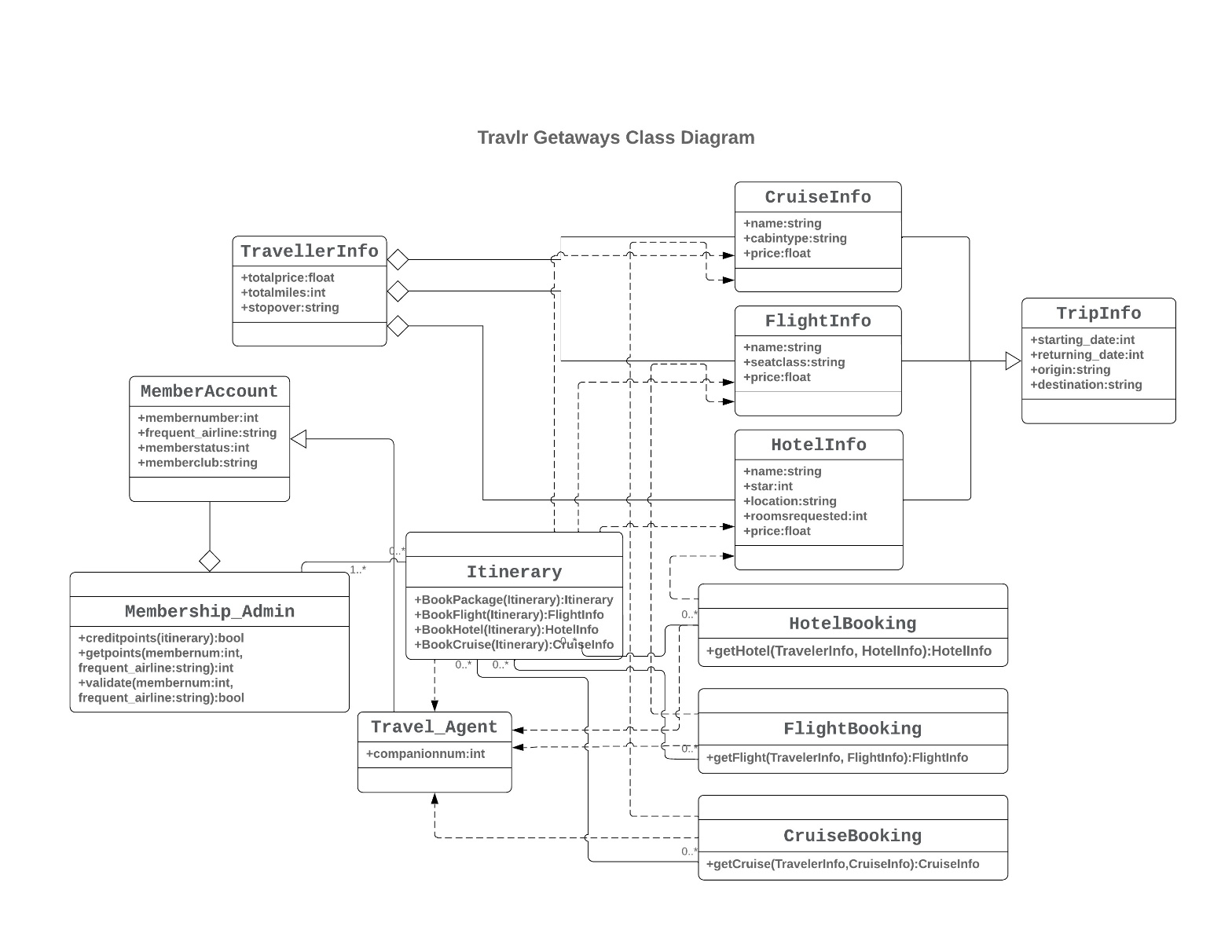
### Sequence Diagram

A diagram of a server

Description automatically generated

The diagram shows three key components, the client, the server and the database. It starts with the user side of the computer, then moving to the view. The user computer would log in and access. The server side then calls the website using MongoDB and connects to the Travlr website. From there a scope and a view is assigned and displayed. The data then delivers the HTTP to the user computer.

## Class Diagram



The diagram shows the different relationships between relating classes. Each member starts with a basic account and the ability to obtain different permissions. The data in CruiseInfo, FlightInfo and HotelInfo’s is stored in TravellerInfo. The data will be moved within the site, and it would show the site’s info to the members when the user interacts on the website.

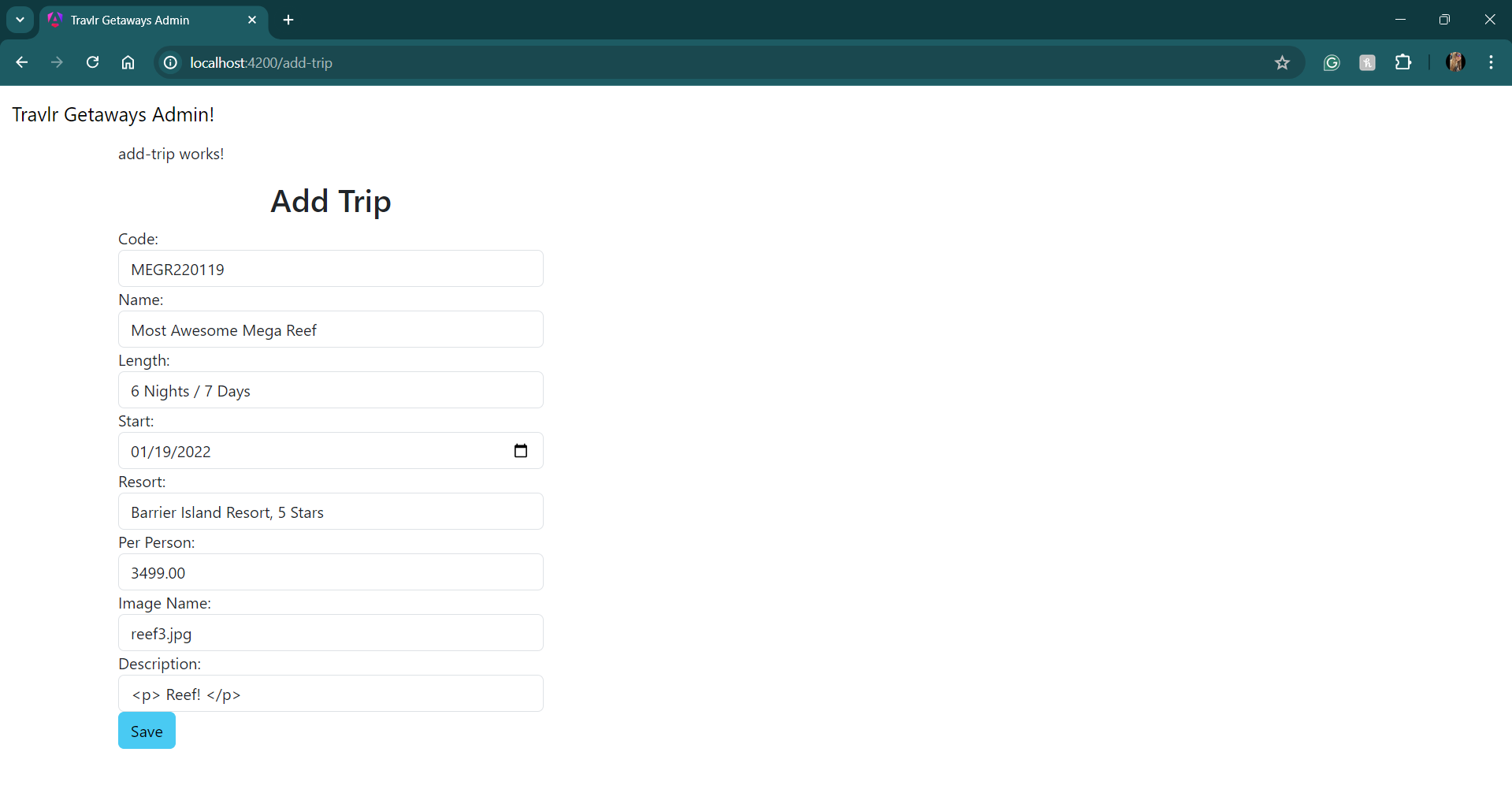
## [API](#_heading=h.2jxsxqh) Endpoints

Exposing RESTful endpoints is a design approach to enable an application to participate in a larger ecosystem. Document each endpoint in the table below, including the HTTP method, purpose, URL, and notes.

| **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** | <Create new list of things> | </api/things> | <Creates new list ofthings> |
| **POST** | <Create a single thing> | </api/things/:thingId> | <Creates single thing instance, identified by the thing ID passed on the request URL> |
| **PUT** | <Update and replaces list of all things> | </api/things> | <Updates and replaces list of all things> |
| **PUT** | <Update single thing> | </api/things/:thingId> | <Updates and replaces single thing instance, identified by the thingID passed on the quest URL> |
| **PATCH** | <Update list of all things> | </api/things> | <Updates and modifies list of all things> |
| **PATCH** | <Update and modify single thing> | </api/things/:thingId> | <Updates and modifies single thing instance, identified by the thingID passed on the quest URL> |
| **DELETE** | <Delete list of all things> | </api/things> | <Deletes full list of  things>  <Deletes list of all things> |
| **DELETE** | <Delete single thing> | </api/things/:thingId> | <Deletes single thing instance, identified by the thing ID passed on the request URL> |

## The User Interface

(1) a unique trip, added by you



(2) the Edit screen

A screen shot of a computer

Description automatically generated

the Update screen

A screen shot of a computer

Description automatically generated

Angular and Express are both frameworks related in JavaScript, each has a unique role and has a distinct structure for different tasks inside of the project. Angular is a front-end framework, that has a modular design and separates different processes within the application. Angular projects may include several important directories for application code, for static files, and other features. The app directory divides the processes that encapsulate each individual functionality. Angular uses TypeScript and follows a strict system structure that allows for maintainability and scalability throughout the development process. Express is a simple back-end framework for Node. Its structure is flexible and less structured compared to Angular's. An Express projects directories include mapping routes, database schemas, controller logic, and rendering. Express relies on JavaScript, providing more freedom in organizing the code but requiring more to maintain structure in larger applications. Overall, Angular's is more structured and standardized, suitable for extensive front-end application development, while Express offers a more flexible back-end development, allowing developers to design their systems architecture according to the application's needs.

In testing this application, a few different methods were used. One of which was Unit Testing. On the front end, we used HttpClientTestingModule in Angular to set up mock requests. On the backend, we used mongo to show the data being moved and loaded during different actions. Another method of testing used was integration testing. Again, using HttpClientTestingModule in Angular we were able to create calls to the API to see that the frontend responded correctly. Tests were written to trigger GET and PUT requests using the buttons formulated during the App\_Admin process. There were a few times when using Postman and MongoDBCompass to create mock data and interactions to confirm that the application features were working properly.

References

*Express vs. Angular: Key Differentials to Decide the Right Web App Framework*. (2021, May 3). Insights on Latest Technologies - Simform Blog. <https://www.simform.com/blog/express-vs-angular/>

*Single Page Applications: Everything You Need to Know*. (n.d.). Www.magnolia-Cms.com. Retrieved June 30, 2024, from https://www.magnolia-cms.com/blog/all-about-single-page-applications.html

‌ *What Is a Single-Page Application? Architecture, Benefits, and Challenges |*. (n.d.). <https://www.spiceworks.com/tech/devops/articles/what-is-single-page-application/>

*What is Single Page Application? Understanding SPA*. (n.d.). Www.outsystems.com. Retrieved June 30, 2024, from https://www.outsystems.com/tech-hub/app-dev/what-is-single-page-application/

‌

‌