

Travlr website

# **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 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 | 11/17/2024 | David Vera | Executive summary, design constraints, and system architecture view |
| 1.1 | 12/1/2024 | David Vera | Class diagram and API endpoints |
| 1.2 | 12/15/2024 | David Vera | Expanded API endpoints and added User Interface |

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

The architecture for the web application will be created using the MEAN stack development process. The use of Angular will facilitate the creation of the front-end application that users or customers will interact with. Meanwhile, express is used to create and manage the back-end server logic. MongoDB is a powerful database application which is fast and easy to implement. Also, its NoSQL structure means it is easier for admins to add or remove properties to a database without a need to remake the entire database to match the parameters.

Management of the web application and databases will be implemented using the single-page application which will dynamically update to demonstrate the information on the databases and options for admins to add, update, or remove entries.

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

MongoDB is not designed for significantly large datasets and can experience a slowdown with large datasets. While this is a real thing to consider it is unlikely that the travlr web application will need to find a solution to this since a large dataset in this case would mean a database with millions of entries.

Once the transition to the MEAN stack form of the web application is completed it is difficult to backtrack to an older form of web application. Thus, it needs to be taken into consideration that any further development on the web application would need to follow the MEAN stack process.

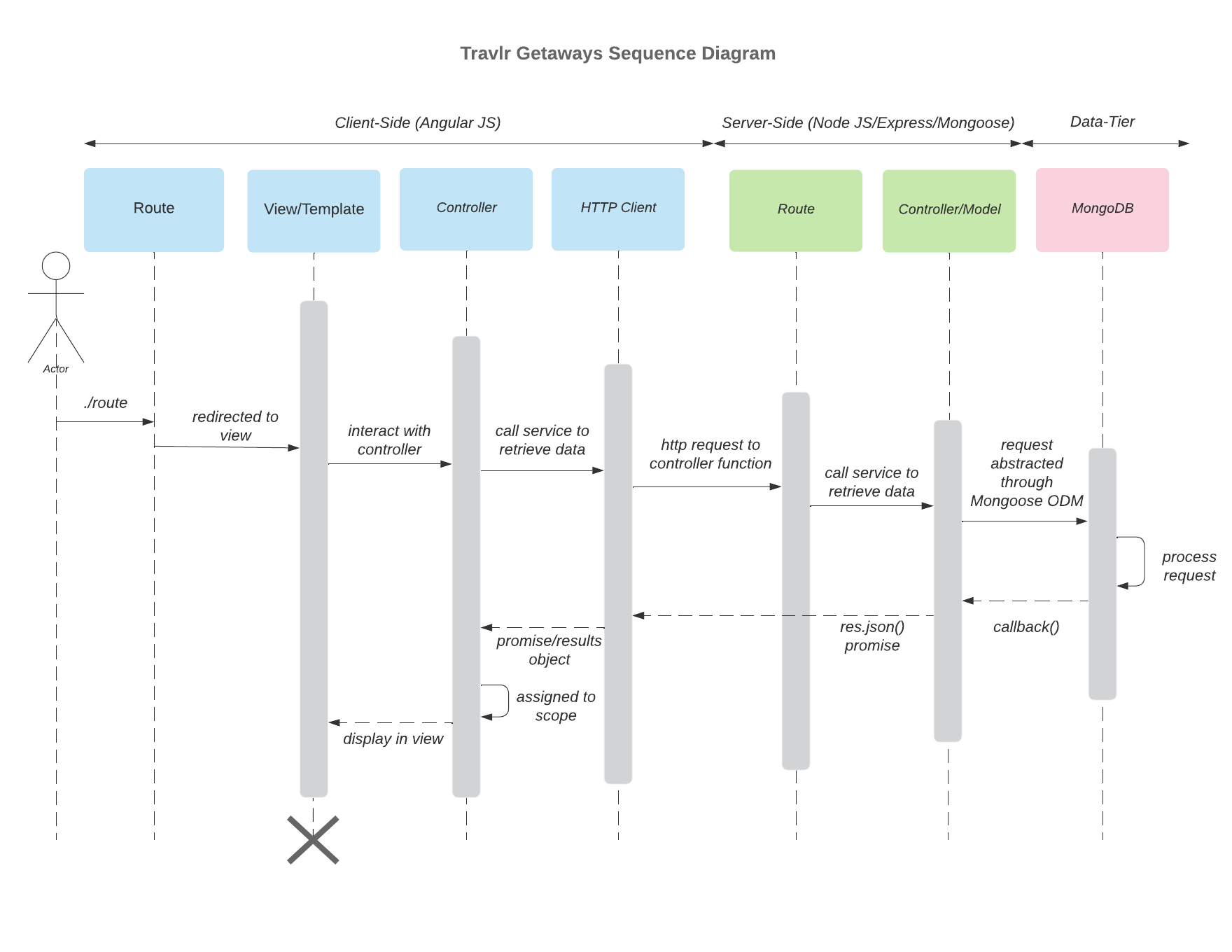
## [System Architecture View](#_heading=h.44sinio)

### Component Diagram



The client side of the web application is majorly handled by Angular. It is displayed through the users browser of choice and has the option for a user to login for preferences or past travel purchases. The Server side uses Express for communication between background structures, such as the MongoDB databases, and the client sided application. It also handles user authentication procedures to help protect sensitive data.

### Sequence Diagram



The actor (user) enters a specific route which is then redirected to the corresponding view. This view calls for the appropriate controller which renders the page that will be displayed while also making any necessary external data retrieval calls through the HTTP Client. The HTTP Client redirects the data requests to the corresponding API route which then calls the correct controller. This API controller makes the calls the MongoDB databases using mongoose and then the data returned goes through multiple returns till it once again reaches the controller and view stages of the client sided application.

## Class Diagram

A diagram of a travel application

Description automatically generated

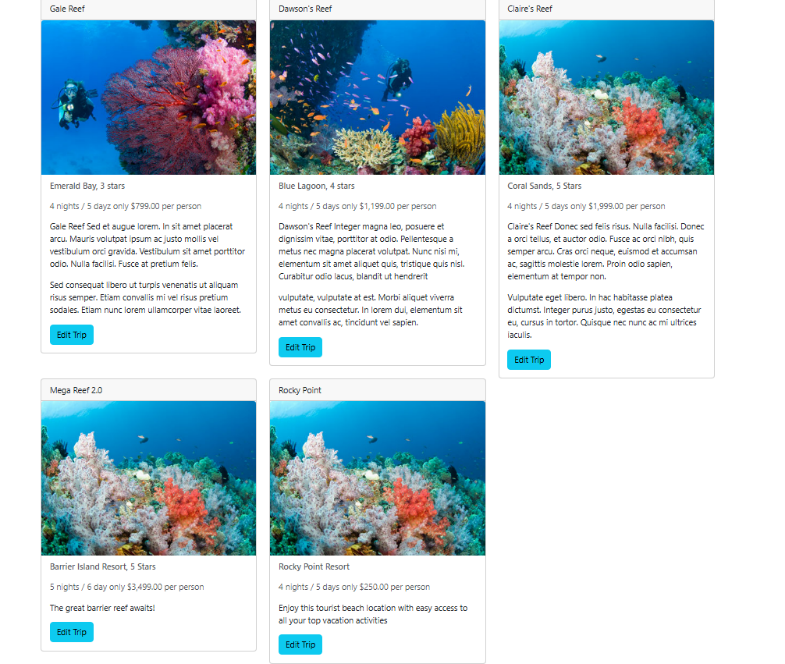
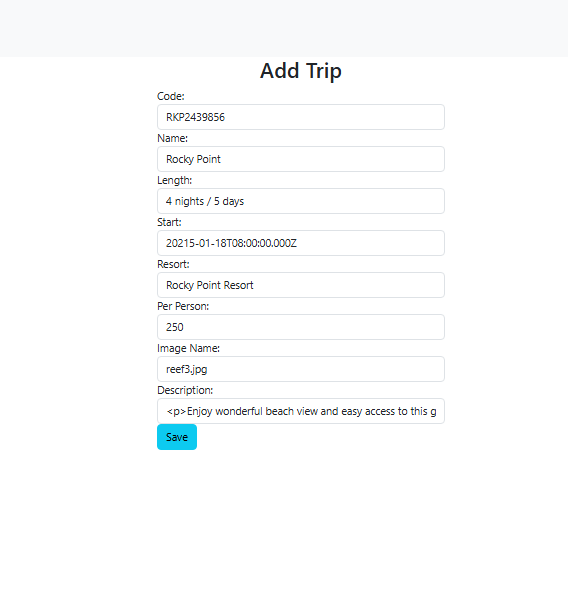
TravellerInfo inherits from the base MemberAccount class which itself contains a membership\_admin component. CruiseInfo, FlightInfo, and HotelInfo inherit multiple variables from TripInfo and each have itinerary components. Travel\_Agent has the methods to book a flight, hotel, cruide, or an entire package and has associations with the various hotel, flight, and cruise booking classes which each have the methods to return a corresponding cruise, flight, or hotel information class.

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

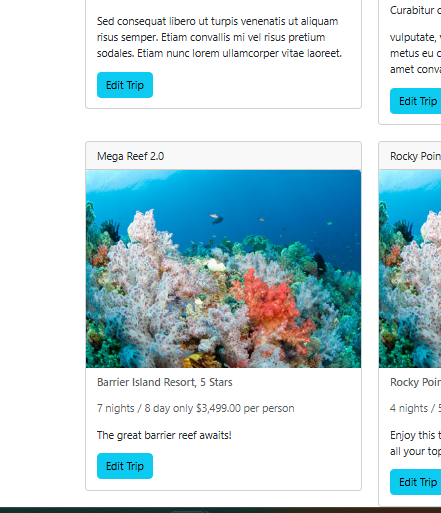
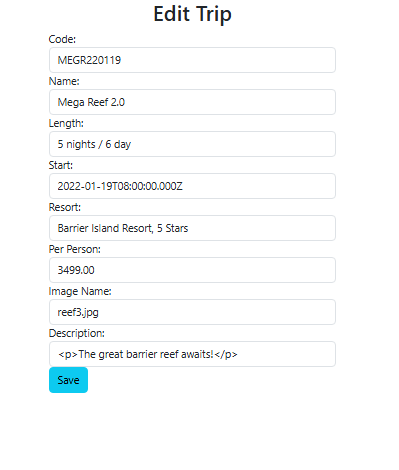
| **Method** | **Purpose** | **URL** | **Notes** |
| --- | --- | --- | --- |
| **GET** | Retrieve list of trips | /api/trips | Returns all trips |
| **GET** | Retrieve single trip | /api/trips/:tripCode | Returns a single trip |
| **POST** | Adds an admin account | /api/register |  |
| **POST** | Authenticates user credentials | /api/login |  |

## The User Interface

Add Trip



Edit and Update Trip



The angular project structure focuses on the implementation of individual components that are used in combination to produce the different sections of the SPA. The express project on the other hand separates into individual pages instead of components. While you can use partials to extract repeated code for easier maintainability it doesn’t create specialized functions like the components of angular. For the SPA testing is spread out to multiple locations. Api test software such as postman are used to verify that the API methods are working properly including checking that authentication is being used properly. Testing for the SPA itself is checked through the console logs of both the server and on the client to check for errors, warnings, and debug console messages.