

Travlr Getaways Application

# **CS 465 Project Software Design Document**

Version 3.0

## Table of Contents

[**CS 465 Project Software Design Document** 1](#_heading=h.gjdgxs)

[Table of Contents 2](#_heading=h.30j0zll)

[Document Revision History 2](#_heading=h.1fob9te)

[Instructions 2](#_heading=h.1fob9te)

[Executive Summary 3](#_heading=h.3znysh7)

[Design Constraints 3](#_heading=h.2et92p0)

[System Architecture View 3](#_heading=h.tyjcwt)

[Component Diagram 3](#_heading=h.3dy6vkm)

[Sequence Diagram 4](#_heading=h.4d34og8)

[Class Diagram 4](#_heading=h.2s8eyo1)

[API Endpoints 4](#_heading=h.17dp8vu)

[The User Interface 4](#_heading=h.4h5b3qv007h8)

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

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 1/22/23 | Matthew Muller | Executive Summary, Design Constraints, and System Architecture View completed |
| 2.0 | 2/2/23 | Matthew Muller | Sequence Diagram, Class Diagram, and API Endpoints completed |
| 3.0 | 2/19/23 | Matthew Muller | Document Completed |

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

The Travlr Getaways web application will function to provide users with information on travel to various locations utilizing a MEAN stack architecture. The MEAN (**M**ongoDB, **E**xpress.js, **A**ngular.js, **N**ode.js) framework will provide the application with its user interface, backend logic, and supporting database.

The customer-facing side of the application will be provided using Angular in the form of a single page application. Angular helps the application to run efficiently with low load times and a smooth flow of information between the front-end and back-end components of the system.

The back-end will be provided by Express ,a Node.js framework. Express is both minimalist and flexible, allowing for effective error handling without cluttering the application. It serves to route URLs and handle HTTP requests and responses via interaction with Angular. These components will all work together with a MongoDB database to store, access, and display all of the site’s travel information.

MongoDB is a NoSQL database that, through the use of its Node.js driver, will connect to the server-side of the application to store documents, as well as receive and retrieve them based on communication with the client-side. This communication comes in the form of JSON documents sent and received by the Angular front-end. MongoDB will benefit the application with its built in database security and easy cloud scalability.

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

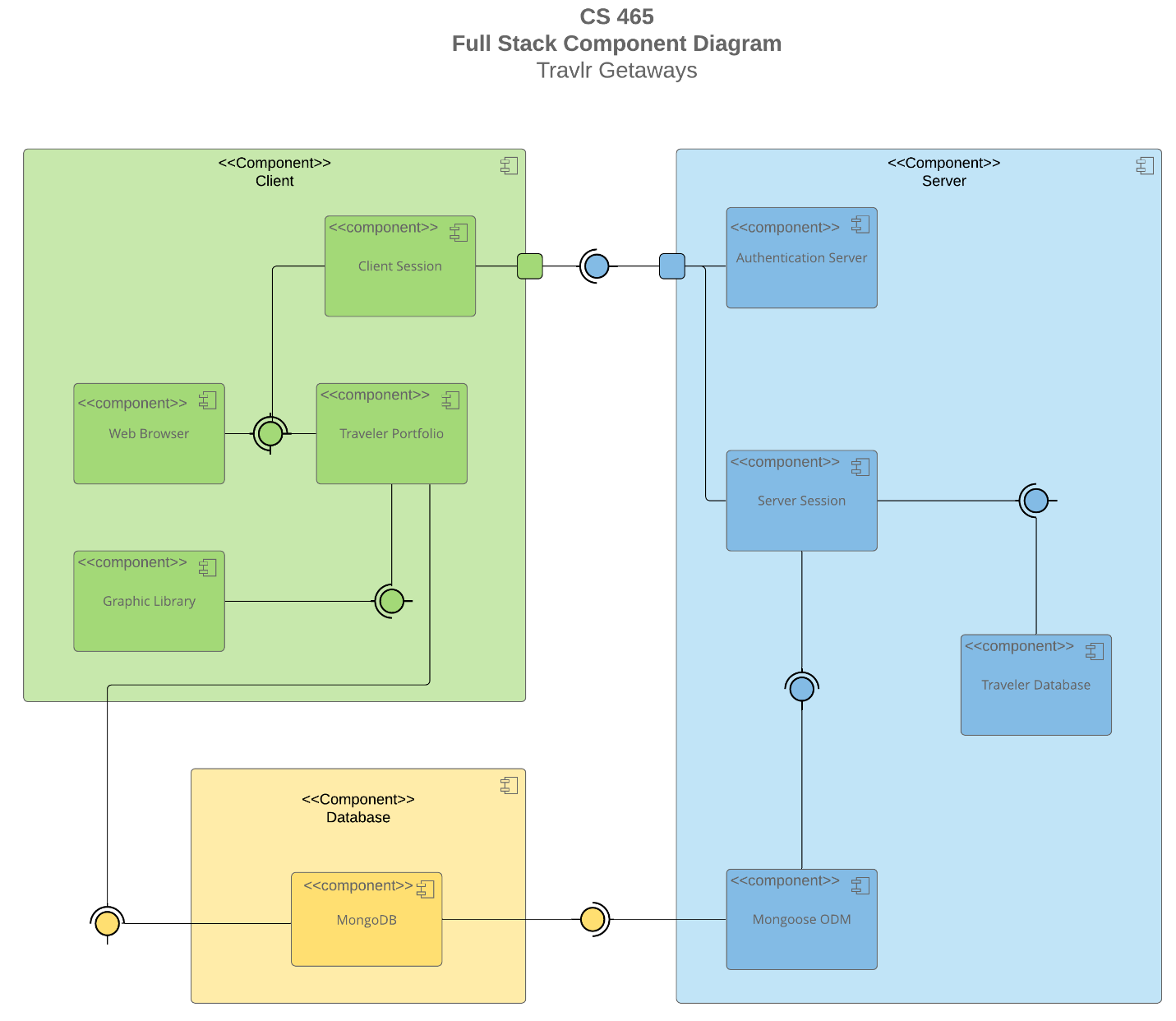
One of the main constraints for the development of the Travlr Getaways application is time. The project’s development schedule is set out such that a new component of the application is completed each week. It is important that development does not fall behind schedule as there will not be a gap in the application’s progression to allow for catching up by the project’s deadline.

Another design constraint is the style of the application. Travlr Getaways has provided style guidelines for their web application so it is important that they are followed throughout development. This will help to maintain consistency in the brand’s identity.

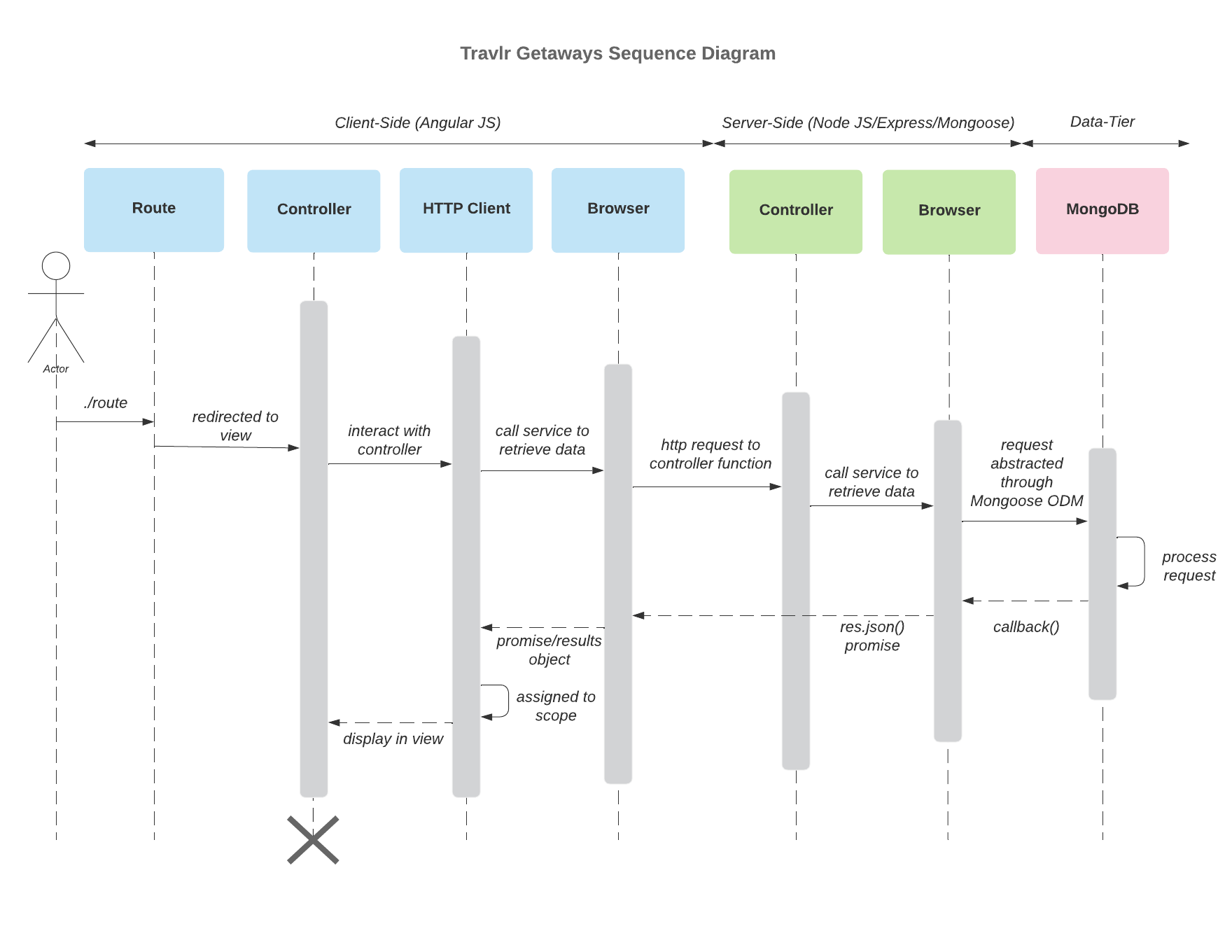
## [System Architecture View](#_heading=h.44sinio)

The Travlr Getaways web application’s system architecture will consist of a MEAN stack that provides the interactions between the Client, Server, and Database components of the application. The database component will be provided by MongoDB, which will connect to the server component of the application by communicating with Express through a Node.js driver. The database and server components will be connected to the Client via the Angular web framework.

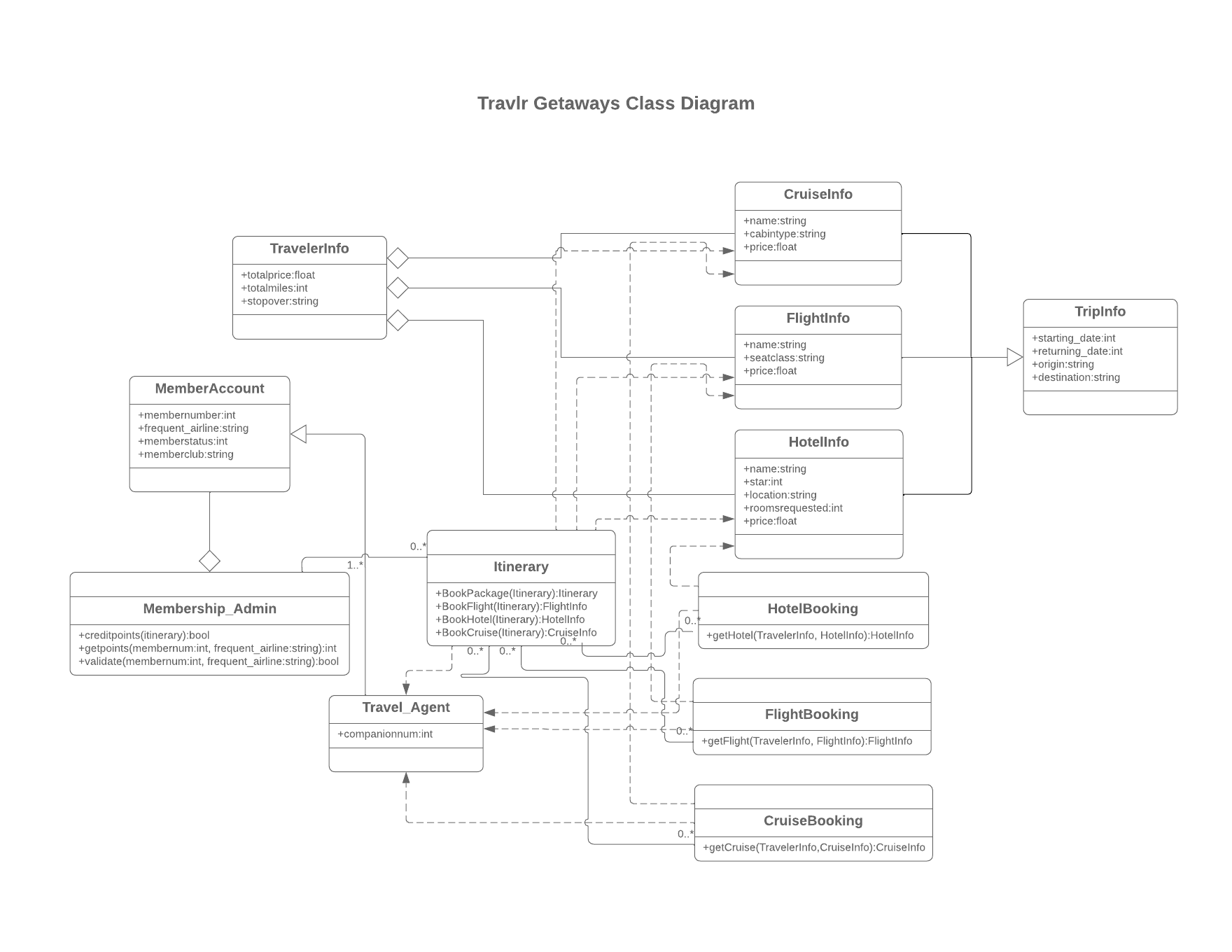
### Component Diagram



### Sequence Diagram



## Class Diagram



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

| **Method** | **Purpose** | **URL** | **Notes** |
| --- | --- | --- | --- |
| **GET** | Retrieve a list of trips | api/trips | Retrieves a list of all trips in the database |
| **GET** | Retrieve a single trip | api/trips/<id> | Retrieves a particular trip from the database |
| **DELETE** | Delete a trip | admin-api/delete-trip/<id> | Deletes a particular trip |
| **DELETE** | Delete a user | admin-api/delete-trvlr/<id> | Deletes a particular user |
| **POST** | Login | admin-api/index | Allow the admin to log in to the application |
| **POST** | Approve a trip request | admin-api/approve | Approves a trip request from a user |
| **PATCH** | Modify a trip’s details | api/update-trip/<id> | Allows user to modify details of trip with given id |
| **PATCH** | Modify all trips’ details | api/update-trip | Allows user to make changes to all trips at once |

## The User Interface

Angular is typically used as a frontend structure while Express generally provides backend capabilities. Angular is a typescript based JavaScript framework that allows for the creation of an easily scalable web application. It has built in tools and libraries that do not hinder the speed or size of an application. Express is a server-side framework for Node.js that provides plugins, template code, middleware packages, and routing functionality for fast and efficient web development. It delivers impressive speeds to Node.js applications that exceeds that of Angular for non-large-scale projects.

The single-page application architecture of this project provides many benefits over a simple web application. SPAs allow for very fast load times, as the website loads only a single page and then rewrites that page with new content fetched from a web server as the user interacts with it rather than loading an entirely new page for every interaction. Another benefit of SPAs is that they provide a better user experience (UX) as they navigate quickly and seamlessly and display a consistent user interface (UI).

When testing the functionality of the single-page application, there were numerous errors that could occur. Some of these included login errors, request errors, and network errors. These errors would usually occur when the application would attempt to GET or PUT data from the database. This included both user info and trip info, meaning that if something was not correct for either the login functionality or the trips’ CRUD capabilities, an error would result.