

Travlr Getaways Web Application

# **CS 465 Project Software Design Document**

Version 1.2

## 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 | 03/24/2024 | Laura McAroy | Initial Revisions and Writeup |
| 1.1 | 04/07/2024 | Laura McAroy | Sequence Diagram, Class Diagram, API Endpoints to date |
| 1.2 | 04/20/2024 | Laura McAroy | Final Version |

## 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)

To create a web application that is user-friendly and meets Travlr Getaways vision, a MEAN stack development process is the most ideal. MEAN stack includes the use of MongoDB as the database to store all website and user data, Express.js as a web framework to work with Node.js, a JavaScript runtime environment, and Angular, which will be utilized to create all of the client-side interaction features. Express and Node work together to connect the front end of the application (client side) to the back end (Travlr database).

Administrators will be able to take advantage of a single-page application, or SPA, which will provide a simple interface that will enable them to easily make changes to the database. This SPA will be able to read all the HTML, CSS, and JavaScript code to make dynamic updates.

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

While MEAN stack development is the best choice for developing the Travlr Getaways web application, there are a couple things to take into consideration throughout the development process. MongoDB was not designed for large, heavy workloads and has the potential to slow down or even lose records if the database becomes too large. Fortunately, it is unlikely that the data contained in this web application will reach MongoDB’s level of capacity, so this is not likely to be an issue for the purpose of this project.

Since the technologies used in MEAN stack development are still relatively new, there is a potential that they can be a target for hackers. Extra security measures should be implemented to ensure the safety of customer data.

## [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).

Angular is used to create the website that clients see when they open their web browser. Users will be able to login and access their profile, once the credentials are validated and authenticated through the server. The server side consists of Express.js running on a Node.js server, and enables all of the communication between the database, MongoDB, and the web page the client views on their browser.

These components work seamlessly together to create a fast, efficient, and user-friendly web application.

### Sequence Diagram

A diagram of a process

Description automatically generated

The user enters a page address into the browser which routes them to their desired browser page. On the browser, the user interacts with items on the page by using controllers which call on the HTTP client to retrieve data from the web server. This leads to a repeat in processes on the server side, where the http request is routed to the appropriate controller, which searches for the requested information in MongoDB, and passes the information back through the server to the client side where it is displayed on the web browser.

## Class Diagram

**A diagram of a travel geoways class diagram

Description automatically generated**

Starting at the right of the diagram, TripInfo contains the starting and end date of the trip, as well as the origin and destination. It contains three subclasses, FlightInfo, HotelInfo, and CruiseInfo, that will inherit these attributes. These classes are dependent on their respective booking classes, as well as the Travel\_Agent class which contains the functions to book the flights, hotels, or cruises. The FlightInfo, CruiseInfo, and HotelInfo classes are all parts that make up Itinerary class, which determines the total price, total miles, and stopover for the trip.

The TravellerInfo class on the lower left provides necessary information about the number of companions to the HotelBooking, FlightBooking, and CruiseBooking classes, as well as the Travel\_Agent. It is a subclass of MemberAccount, which contains the member’s number, club, status, and frequent airline. The Membership\_Admin class uses the MemberAccount class to determine points and validate information, which is then used by the Travel\_Agent class.

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

| **Method** | **Purpose** | **URL** | **Notes** |
| --- | --- | --- | --- |
| **GET** | Retrieves list of all trips in database | /api/trips | Returns all trips listed in database |
| **GET** | Retrieve single trip | /api/trips/:tripcode | Returns single trip from database, using tripcode as reference. |
| **POST** | Add a trip to the database | /api/trips | Adds a trip to the database |
| **POST** | Register a new user | /api/register | Takes a name, email, and password and registers a user by placing them in the database |
| **POST** | Login a user | /api/login | Uses email and password to login a user, returns a jwt |

## The User Interface

A screenshot of a computer

Description automatically generated

User Login Screen

A screenshot of a computer

Description automatically generated

Home page displaying current trips in database

A screenshot of a computer

Description automatically generated

Add Trip button adds a new trip to the database

A screenshot of a computer

Description automatically generated

Updated home screen displaying new trip

A screenshot of a computer

Description automatically generated

Edit Trip function to edit existing trips

**SPA Analysis:**

The backend of this project structure utilizes Angular, node.js, and MongoDB to access and display all current trip information on a single page application for administrative users. This SPA provides a more fluid, dynamic user experience compared to traditional web applications because it allows instant content updates without full page reloads with every request. The Angular SPA can manage multiple components through a single service, accessing information quicker than the Express framework, which requires each page to have its own route and html.

To ensure that the Angular structure was working correctly, Postman was used to test API routing by checking HTTP requests. After the Postman testing was completed, developer tools were utilized while running the browser on Chrome.