

Travlr Getaways

# **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 | 05/16/23 | Jaelyn Sloan | Version 1.0 |

## 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)

Based on the requirements of this project, MEAN will be utilized to create the architecture of this application. This JavaScript stack allows for creation and usage of a database (MongoDB), web framework (Express), front-end framework (Angular), and a web server (Node.js). By implementing these four main technologies, we can create a full stack program with all of the desired functionalities. The customer facing side of the program will be developed using Angular to create the framework using HTML, CSS, and JavaScript. This allows for designing the actual customer-facing website with interactive options for the user. The server side of the program will be developed using Express and Node.js to assist with the web framework and server by allowing URL routing and handling HTTP requests. The backend of the project will be based on MongoDB using NoSQL, where the database will store any data that is necessary for the application. The program also requires an administrator single-page application. For this, MEAN will still be implemented, but the way data is retrieved is slightly different. The program will use placeholder elements in the HTML files which include links to JavaScript files. Once the page is loaded, the server will fetch the content requested and update the HTML in the web browser.

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

The program is required to allow customers to create an account, search for travel packages by location and price point, and book reservations with the travel agency. In order to develop a program that meets all of these requirements, a database must be implemented properly to store and update data as necessary. Using Express and MVC routing, the program will be able to navigate through the site properly and smoothly. An admin-only page should also be developed to allow for administrators to maintain and update the website as needed. By implementing HTML templates with JSON, the admins will be able to use this SPA to make changes as necessary.

## [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).

In the above diagram, we can see there are three components: Client, Server, and Database. The client component is where the actual website will be viewed and interacted with. This is where a session will take place within the web browser. The user will have their own “portfolio” with all of their personal data which would be passed down into the database. The server component is where all of the behind-the-scenes functions happen, such as security measures and the web server data. The server component will provide information to the client component to allow the web page to be viewed and used as intended. The database component will receive data from both the client and server and store the data appropriately. When the client or server needs data, it will pull it from the database as well.

### Sequence Diagram

A picture containing text, diagram, screenshot, parallel

Description automatically generated

This diagram shows the flow of logic in the web application, from the client side to the server side. The first step in the process would be for the user to login. From there, the client side would pull up the webpage by connecting to the server side and pulling data from the MongoDB database. The user can then interact with the webpage by selecting different trips through the client side which would also pull data from the server side that would come from the database. Admin interactions would be performed in the same way by having to login to interact with the client side, which will interact with the server side to pull data from the database, then return a functioning webpage.

## 

## Class Diagram A picture containing text, diagram, plan, technical drawing Description automatically generated

The above diagram of the JavaScript classes of this web application show how each class connects and works with each other. The diagram begins with the TravlerInfo, which holds information relating to the traveler’s ideal vacation. From there, the user can view information for each: CruiseInfo, FlightInfo, and HotelInfo to show information about a travel location. These three options will each point to the TripInfo class, which will give details about the trip. Once the user is ready to book, they will be directed to HotelBooking, FlightBooking, and CruiseBooking which allows them to decide and choose a hotel, flight, or cruise option. These are all connected to the Itinerary class, which holds the data regarding the chosen package. From there, we have the Travel\_Agent class which holds data for the number of companions. Membership\_Admin and MemberAccount classes both hold data that refers to the account details of the user.

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

| **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 |
| **GET** | Retrieve a specific review | /api/things/:thingId/reviews/:reviewid | Return all things within a specific review |
| **POST** | Create new location | /api/things | Create a new location in a specified file |
| **POST** | Create new review | /api/things/:thingid/reviews | Create a new review in a specified file |
| **PUT** | Update specific location | /api/things/:thingid | Update data within a specific thing |
| **PUT** | Update specific review | /api/things/:thingid/reviews/:reviewid | Update data for all things within a review |
| **DELETE** | Delete a specific location | /api/things/:thingid | Delete a specific thing |
| **DELETE** | Delete a specific review | /api/things/:thingid/reviews/:reviewid | Delete a specific review |

## The User Interface

Angular uses a Model-View-Controller architecture that differentiates the application logic from the user interface. With Angular, the program will load the first page and continue to update the other pages within the program. The architecture of Angular allows for better organization of code components and simplifies the development structure by combining the MVC and two-way data binding. Express HTML does not require any specific architecture. However, it is recommended to use an MVC architecture type. Express HTML allows a web server to produce HTML pages only when requests are made by the users. The process of testing SPA can be performed using Unit Testing by injecting mock data into a controller. It would be necessary to test for credentialing, URL/login api results, and testing the functionality of the GET and PUT methods. Some errors that could result from this testing could be invalid credentials, network errors, or request errors.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated