

Travlr Getaways

# **CS 465 Project Software Design Document**

Version 3.0

## 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 | 03/14/2022 | Miranda Putnam | Milestone One |
| 2.0 | 04/02/2022 | Miranda Putnam | Milestone Two |
| 3.0 | 04/11/2022 | Miranda Putnam | Final Project |

## 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)

The client for this software is Travlr Getaways, who want us to create a travel booking website for customers to book travel packages. The client needs their customers to be able to create and use an account through the website, search for travel packages on the website by both location and price-point, and book reservations. The client also requires an admin-only site for Travlr Getaways administrators to maintain a customer base, trip packages, and pricing. The appropriate architecture of the web application for the client based on their software requirements is a client-server full-stack application that uses the MEAN stack. The MEAN stack uses MongoDB, Express, Angular, and Node.js in order to create the full-stack application effectively. The customer-facing side of the web application is built using the Express framework and Node.js. Express is an application framework from Node.js that is designed for building web applications. The server side of the web application uses MongoDB to store and use the database information when requested. MongoDB is a document-oriented database program that uses JSON documents. The administrator single-page application is built using the Angular Command Line Interface to build the necessary components and services for the client. Angular comes equipped with fully integrated tools in order to facilitate the build of the SPA. Overall, the MEAN stack applications will allow us to build a web application that meets the client’s needs for their Travlr Getaways site.

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

There are a few design constraints for developing the web-based Travlr Getaways application. These include the MEAN software specifications, a technical understanding of the software being used for the project, and time and schedule. The web application we will build for the client will utilize the MEAN stack. This constrains the development applications to include MongoDB, Express, Angular, and Node.js. Along with these software boundaries comes an understanding of them. The team will need to be familiar with each of these components of the MEAN stack in order to develop the web application properly. This includes an understanding of MongoDB commands to be able to build and maintain the database, fluency in JavaScript, and knowing how to utilize the Express framework through Node.js as well as Angular. Lastly, time and scheduling will constrain the project, as the client will want an up and running web application in a set amount of time. The development team will need to adhere to a schedule in order to meet this demand.

## [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).

The overall system architecture of the web application for the client Travlr Getaways is client/server with the use of a database. This system is represented using the component diagram above. The components within the client side of the system include the client session, the web browser, the traveler portfolio, and the graphic library. Each of these system components interacts with each other in different ways. The web browser is connected to the client session and traveler portfolio, which is denoted by a full circle, indicating that each of these components produces information which is used by the others. The half circle from the client session and web browser indicates each of these components requires information in order to perform their function. The traveler portfolio is also connected to the graphic library, which it uses to produce the correct images. The database portion of the system uses one component, the MongoDB database, which is connected to the traveler portfolio (client side) and the Mongoose ODM component within the server side of the system. MongoDB provides information for these components to use in order to perform their specific functions. The server side of the system contains the following components: Authentication server, server session, traveler database, and Mongoose ODM. The authentication server and the server session connect to the client session. The client session relies on the information provided by the server in order to function properly. The server session is also connected to the traveler database and Mongoose ODM. The server session relies on the information these components provide in order to function properly. Overall, each of these components and their interactions are vitally important to the function of the Travlr Getaways web application.

### Sequence Diagram

Timeline

Description automatically generated

The Travlr Getaways Sequence Diagram shows the interactions between the MEAN stack applications, as well as how the client, server, and database interact with each other to perform the desired outcome for the web application. The user first accesses the application through a route that is presented through a view such as a browser which will ask them to sign into their account. The user may then request trip information, view travel packages, makes reservations, and more which triggers an interaction with the client-side controller to begin to retrieve the desired information. For example, when the user presses the Travel tab on the webpage, the controller makes a call to the HTTP client to redirect the page to the new URL with the corresponding information. Next in the sequence, the client-side controller calls the HTTP client to retrieve the requested data, which in turn interacts with the server-side controller/model and HTTP client in order to retrieve the data. This data could be trip or account information. The server-side HTTP client then requests the information from the database, MongoDB, through Mongoose ODM. MongoDB then processes the information request, returns the information back to the server-side HTTP client, which transports the JSON file information to the client-side HTTP client. Back on the client side, the results are moved to the controller which assigns the results to scope and displays them in the browser view for the user to see.

## Class Diagram

Diagram

Description automatically generated

The Travlr Getaways Class Diagram depicts twelve different JavaScript classes consisting of information (attributes and operations) relevant to their function. Each class and their interactions with each other contribute to the functionality of the Travlr Getaways web application. The class Trip Info uses attributes for total price, total miles, and stopover. The Trip Info class is related to the Cruise Info, Flight Info, and Hotel Info classes, meaning Trip Info populates it’s information from the attributes of the aforementioned classes. Cruise Info contains the attributes name, cabin type, and price; Flight Info contains the attributes name, seat class, and price; Hotel Info contains the attributes name, star, location, rooms requested, and price. Each of these three classes also contribute to the Itinerary class, which inherits their information. The Itinerary class uses the attributes starting date, returning date, origin, and destination. The classes Hotel Booking, Flight Booking, and Cruise Booking are dependent on Hotel Info, Flight Info, and Cruise Info respectively. The booking classes each use a respective “get” operation for booking, which also uses the Traveler Info class information. The Traveler Info class consists of a single attribute, the companion number. Next, the Travel Agent class uses four different operations, book package, book flight, book hotel, and book cruise. These operations populate the Itinerary, Flight Info, Hotel Info, and Cruise Info. Lastly, the Member Account class consists of attributes member number, frequent airline, member status, and member club. The Member Account class is part of the Membership Admin class, which uses three different administrator operations: credit points, get points, and validate. Overall, each of these JavaScript classes and their interactions with each other inform how the Travlr Getaways web application functions.

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

| **Method** | **Purpose** | **URL** | **Notes** |
| --- | --- | --- | --- |
| **GET tripsList** | Retrieve list of all the trips. | /api/trips | Returns all active trips. |
| **GET tripsFindCode** | Retrieve a single trip by code. | /api/trips/{tripCode} | Returns a single trip instance, identified by the trip code passed on the request URL. |
| **POST**  **tripsAddTrip** | Adds a single trip to the database. | /api/add-trip | Adds a trip with specified information to the database from the add-trip page. |
| **PUT**  **tripsUpdateTrip** | Update a single trip’s information. | /api/edit-trip | Updates a trip’s information from the edit trip page and stores the new information in the database. |
| **POST Login** | Log an existing user into the system. | /api/login | Asks for the username and password of a registered user to sign into the system. |
| **POST Register** | Register a new user for system use. | /api/register | Registers a new name, username, and password for login. |

## The User Interface

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Updated Trip (Before and after):**

Graphical user interface, website

Description automatically generatedGraphical user interface, website

Description automatically generated

The Angular project structure differs from the Express HTML customer-facing page in that it is built specifically as a front-end framework for administrators (Dhaduk, 2022). The structure of the Angular project is a bit more complex than the Express framework. The Angular framework includes a source folder which holds most of the functionality of the Traveler Admin website. The Express framework is a lot more simplified, containing less base-line folders as well as less added folders in order to perform the desired functionality for the customer-facing page. Overall, Angular is a more complex front-end framework while Express is a more simplified, traditionally back-end framework (Dhaduk, 2022). Single Page Application functionality comes with both advantages and disadvantages. Some advantages include ease of use for users, simple production and deployment, and scalability. SPA’s improve user experience because they don’t require full page reloads as the user browses the website (Angular University, 2022). An SPA only renders the initial page load and then sends requests to the server based on the user (Angular University, 2022). This allows a better overall performance of the website and improves user experience by making it more efficient. SPA’s are also simple to produce and deploy, especially when using a service like Angular. The files are auto generated and just need to be built with the required data for the desired website. SPA’s are also very scalable and easily updated through use of minimal files and reusable code. Some disadvantages of SPA’s include a low ranking on search engines, reliance on browser resources, and security. Search engines populate web pages based on the number of pages they have, making SPA’s take a very low ranking (Kaur, 2021). SPA’s also rely heavily on browser performance, so they often require a specific browser that is updated to a most recent version in order to perform and display correctly (Kaur, 2021). Security is also a concern when it comes to SPA’s, as they are more prone to cross-site scripting attacks and data exposure (Kaur, 2021). Overall, SPA’s are a great choice for web applications if easy production and deployment and user experience are the main concern. In order to ensure an SPA is working with the API to GET and PUT data in a database there are a few different tests that can be done. First, testing on the website itself, both the admin site and the customer-facing site, can be done to ensure trips are displayed and added properly. Next, checking the database through a service such as Robo T3 can be done to ensure the new data displays in the database. Also, Postman can be used to test the API endpoints of the application, and can be used to register new users in order to test the login functionality on the website. Lastly, browser debugger tools also are helpful when testing an SPA, as they can shed light on any errors that may arise when browsing the website. Overall, there are many ways to test if the SPA is working with the API’s in order to GET and PUT data in the database.

**References**

Angular University. (2022, January 21). *Angular SPA: Why Single Page Applications?* Angular University. Retrieved April 6, 2022, from https://blog.angular-university.io/why-a-single-page-application-what-are-the-benefits-what-is-a-spa/

Dhaduk, H. (2022, February 28). *Express vs. angular: Key differentials to decide the right web app framework*. Insights on Latest Technologies - Simform Blog. Retrieved April 6, 2022, from https://www.simform.com/blog/express-vs-angular/

Kaur, A. (2021, May 14). *What is Single Page Application (SPA)? pros and cons with examples*. Insights - Web and Mobile Development Services and Solutions. Retrieved April 6, 2022, from https://www.netsolutions.com/insights/single-page-application/