

<Name of Software Application>

# **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 | <01/28/2024> | Nickolas Askey | Updated the Executive Summary, Design Constraints, and System Architecture View |
| 1.0 | <02/11/2024> | Nickolas Askey | Updated the System Architecture View: Sequence Diagram, Class Diagram, and the API Endpoints table |
| 1.0 | <02/25/2024> | Nickolas Askey | Updated the User Interface section as well as fixing the API Endpoints table |

## 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)

For this program, Travlr Getaways tasked me with creating a travel booking site for customers to book travel packages. To achieve this, customers should be able to create an account, search for different travel packages based on location or price, and book reservations with the travel agency. There should also be an admin-only site where administrators can maintain available trip packages and pricing for each item and package, as well as keep track of the customer profiles and information. The website should be easy to navigate and there should be plenty of images and text to help new users find what they are looking for. We utilized the MEAN stack to accomplish this goal, since we could utilize Angular on the front-end and Node.js on the backend with MongoDB to help organize and handle complex data.

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

The application should be able to perform well even with high traffic, and provide very fast response times for the user. MEAN stack can help achieve this by utilizing MongoDB to quickly and efficiently store data, while Angular can provide dynamic and interactive front-end development. The app should also provide a high level of user security, since we would be dealing with user payment details, as well as personal information to reserve different packages. To do this, the MEAN stack can help achieve this since Express.js can create a secure web server, and MongoDB can be used to encrypt and hash sensitive 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).

The first component is the Client Component. This component is the front-end of the application. It utilizes a graphic library to render the user interface. This component communicates with the server component through the client session which deals with authentication. The next component is the Server Component. This component is the back-end of the application and runs on Node.js. This component uses Express.js as a framework to handle requests from the client component. This also connects to the Database Component. This component uses an authentication server to verify a user’s identity. The last component is the Database Component. This is the final component that it used to store data related to the application, such as profiles, itineraries, user information, payment info, etc. This component also utilizes MongoDB as a database for flexible data storage and retrieval.

### Sequence Diagram

A diagram of a diagram

Description automatically generated

The /route is connected to the Browser/View/Template. This then interacts with the controller, which then uses a call service to retrieve data from the HTTP Client. This then uses an http request to the controller function, and then makes another call to retrieve data from the HTTP client. Then the request is abstracted through Mongoose ODM and uses MondoDB to process the request and then return the callback() to the HTTP client. The res.json() promise is then returned through the Controller back to the HTTP client, the primes/results object is sent to the controller/model, and then it is assigned to scope. This is finally returned to be displayed in the view on the Browser/View/Template.

## Class Diagram

A diagram of a travel company

Description automatically generated

TripInfo keeps track of the price, miles, and stopover information. This leads into the TravellerInfo subclasses, CruiseInfo, FlightInfo, and HotelInfo. This information is used in the Itinerary to collect data for the trip. The HotelInfo keeps track of the price of the room, location, and number of rooms requested. The FlightInfo keeps track of the seatclass and the price of the flight. The CruiseInfo keeps track of the cabin type and price of the cruise. This info feeds the Travel\_Agent class, and this updates the memberAccount details, as well as collecting data from the membership\_Admin class to verify the user. This all works to complete the Travlr Getaways web application to organize, book, and travel quickly and easily.

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

| **Method** | **Purpose** | **URL** | **Notes** |
| --- | --- | --- | --- |
| **GET** | Retrieve a list of trips | /app\_admin/src/app/trip-card | Returns all active trip instances |
| **GET** | Retrieve a single trip | /app\_admin/src/app/trip-listing | Returns a single trip instance |
| **POST** | Add a new trip | /app\_admin/src/app/add-trip | Adds a single trip instance to the existing list of trips |
| **PUT** | Update an existing trip | /app\_admin/src/app/edit-trip | GET’s an existing trip, and updates the trip with new information |

## The User Interface

There are many big differences between both Angular and Express project structures. First of all, Angular is a front-end framework that creates and utilizes SPAs (single-page applications), while Express is a back-end framework that uses JavaScript to create web servers and APIs (application programming interfaces). These two structures utilize entirely different structures and can perform tasks in different ways. Angular’s structure is module-based, meaning that each module contains related components that help to formulate the framework’s SPAs, while Express is a routing and middleware-based framework, meaning that it is essentially a series of middleware function calls that execute code, make changes to the request and response objects, and then call the next middleware function. In the Angular framework, SPAs are essential to the overall functionality of the program; they are web applications that load a single HTML page and then will dynamically update the page as the user interacts with it. This form of application can function faster and smoother than other application frameworks, they can cache locally, so you don’t need a strong internet connection, and they can integrate with device features, like cameras, push notifications, and even geolocation.

It is also important to ensure that the SPA is working with the API to GET and PUT data in the database; to make sure that everything is working properly, and the different applications can work together, we must do testing on the functionality. One way to do this is with unit testing, this is the process of testing different units of the code; this might include classes, methods, or functions in isolation. Another form of testing would be performance testing, which is the process of testing how the SPA and API perform under different conditions like latency, load, or stress.

**Citations**

Acharya, D. P. (2023, November 7). *What are Single Page Applications? examples, frameworks, and*

*more*. Geekflare. https://geekflare.com/single-page-applications/

Dhaduk, H. (2022, November 11). *Express vs. angular: Key differentials to decide the right web app*

*framework*. Simform. <https://www.simform.com/blog/express-vs-angular/>

Angular. (n.d.). What is Angular? [Webpage]. Retrieved February 24, 2024, from

https://angular.io/guide/what-is-angular