

Travlr Getaways Website

# **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 | 09/27/2024 | Madison Fleitas | Create and implement a customer-oriented front-end architecture that fulfills client software needs. Utilize the HBS templating engine to dynamically render web pages. Organize routes and views to align with the client’s software specifications using the MVC framework. Conduct tests to ensure that static HTML is effectively transformed into dynamic JSON data and displays correctly. |

## 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 web application for Travlr Getaways is designed using the MEAN stack (MongoDB, Express, Angular, Node.js), which provides a full JavaScript-based development environment for both the server-side and client-side components. This architecture allows for scalability, maintainability, and seamless user experience, meeting the specific requirements of both the customer-facing website and the administrator single-page application (SPA).

The customer-facing portion of the website will focus on providing a visually engaging and intuitive interface for customers to explore Travlr Getaways' offerings, make reservations, and interact with dynamic content such as travel packages. The use of Angular allows for efficient rendering of dynamic content, while the Express framework on the backend will handle server requests and facilitate communication between the front-end and the MongoDB database.

The admin SPA is designed to give Travlr Getaways' management team a powerful interface to manage content, user data, and bookings. Angular will also power the admin side, ensuring fast, real-time updates as administrators manage inventory and make changes to travel packages or customer information. This will allow for efficient data handling and streamlined management of the website's operations.

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

Developing the Travlr Getaways web application involves several design constraints that impact both the customer-facing website and the administrator single-page application (SPA). These constraints must be considered to ensure the final product is scalable, secure, and user-friendly.

Travlr Getaways has a limited development timeline and budget. These constraints restrict the use of complex and expensive technologies or third-party services, requiring the application to be built with open-source tools such as the MEAN stack.

The web application needs to support thousands of users simultaneously, particularly during peak travel seasons. To meet these demands, the application must be optimized for performance with efficient database queries, caching mechanisms, and minimal latency.

As the web application will handle sensitive customer data such as payment information and personal details, stringent security measures must be implemented. This includes encrypting data, ensuring secure authentication, and adhering to industry security standards.

The customer-facing website needs to be responsive and functional across various devices and browsers. Developing a responsive design that works seamlessly across desktops, tablets, and mobile devices while maintaining a consistent user experience adds to the development effort.

## [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 system architecture for the Travlr Getaways web application consists of three main layers: client, server, and database. On the client side, users interact through a web browser, which renders the traveler portfolio, a key feature for managing accounts and bookings. The client session component stores session data, ensuring that user-specific actions, such as browsing and booking, are handled smoothly. A graphic library enhances the visual aspects of the website, supporting dynamic elements like maps or interactive content.

On the server side, an authentication server handles user login and security, ensuring only authorized access. This server communicates with the server session, which tracks user interactions and ensures they are processed properly during actions like making bookings or account updates.

The MongoDB database stores all application data, including user information and travel packages. Mongoose ODM acts as an intermediary between the server and MongoDB, ensuring data is properly validated and managed. Together, these components create a robust, modular system that supports both the customer-facing website and the admin single-page application (SPA), enabling seamless user experiences and efficient data management.

### Sequence Diagram

<Illustrate the flow of logic in a web application by completing a sequence diagram. Insert an image of the sequence diagram here.>

<Describe the flow of logic in the web application based on the sequence diagram. Be sure to describe the interactions between the layers, or tiers, of the full stack application. It will be helpful to include significant processes such as Sign In, Trips, and Admin interactions when referring to the sequence diagram.>

## Class Diagram

<Illustrate the JavaScript classes of the web application by completing a class diagram for the web application. Insert an image of the class diagram here.>

<Describe the JavaScript classes of the web application based on the class diagram.>

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

<Exposing RESTful endpoints is a design approach to enable an application to participate in a larger ecosystem. Document each endpoint in the table below, including the HTTP method, purpose, URL, and notes.>

| **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> |

## The User Interface

<Insert screenshots from the development of the SPA development to show the following: (1) a unique trip, added by you, (2) the Edit screen, and (3) the Update screen.>

<Summarize the Angular project structure and how it compares to the Express project structure. Be sure to describe the rich functionality provided by the SPA compared to a simple web application interaction. Describe the process of testing to make sure the SPA is working with the API to GET and PUT data in the database.>