

TRAVLR Getaways – Merrik Wright

# **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 | 01/25/2025 | Merrik Wright | Initial version with Executive Summary, Design Constraints, and System Architecture sections. |

## 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 Travlr Getaways website will be developed using the MEAN stack, an acronym for MongoDB, Express.js, Angular, and Node.js. This technology stack enables seamless development by ensuring efficient handling of JSON data, which enhances communication between the front-end, back-end, and database. The MEAN stack is known for its efficiency, scalability, and ability to streamline development workflows.

* The web interface will be managed using Angular, providing a responsive and visually appealing user experience.
* The server-side logic will be handled by Node.js, utilizing the Express.js framework for efficient request handling.
* The database will be powered by MongoDB, allowing for fast retrieval and storage of trip-related data, ensuring optimal website performance.

In terms of data flow, Angular requests information from Node.js, which processes the request through Express.js. Node.js then interacts with MongoDB to retrieve the necessary data, which is then sent back to Angular for display. This structured data flow ensures a smooth and responsive user experience.

Customer-Facing Web Application:

The customer-facing side of the application will allow users to browse available trips, make bookings, and manage their reservations. The interface will be designed to offer a seamless and enjoyable experience, incorporating features such as dark mode to accommodate users who are sensitive to bright screens.

The application follows an MPA (Multi-Page Application) structure, built using Express.js and Handlebars. While MPAs can sometimes have slightly longer load times between pages, the overall website performance will remain fast and efficient.

Administrator Single-Page Application (SPA):

The admin panel will be developed as a Single-Page Application (SPA) using Angular. This approach ensures that all necessary data is preloaded and dynamically updated, minimizing page reloads and enhancing responsiveness.

Administrators will have the ability to add, edit, and delete trip listings and manage other website content in real-time. Any changes made through the admin panel will be instantly reflected for all users, ensuring efficient content management and seamless updates.

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

When developing the Travlr Getaways web application, several technical and architectural constraints must be considered to ensure scalability, security, and performance. Since the application is built using the MEAN stack (MongoDB, Express.js, Angular, and Node.js), these constraints influence how each component interacts and operates within the system.

Scalability & Performance

* The application must support thousands of simultaneous users, requiring efficient load balancing and optimized database queries.
* MongoDB's BSON document size is limited to 16MB, preventing excessive memory usage. Additionally, nested document depth is capped at 100 levels, ensuring structured and manageable data storage.
* The front-end (Angular) and back-end (Node.js, Express.js) must be optimized for fast load times, potentially incorporating caching mechanisms like Redis to enhance responsiveness.

Security & Data Storage

* Authentication must be implemented using JWT (JSON Web Tokens) or OAuth to ensure secure user access and prevent unauthorized actions.
* MongoDB enforces unique field names and follows case-insensitive database naming conventions, which should be adhered to for consistency and error prevention.
* Data indexing and efficient schema design are critical for quick retrieval of trip details, user profiles, and bookings without performance degradation.

Cross-Platform Compatibility & Deployment

* The application must function seamlessly across desktop and mobile devices, ensuring a responsive design for an optimal user experience.
* Hosted on cloud services such as AWS, Firebase, or Heroku, with proper scaling strategies to handle fluctuating traffic.
* Express.js is used to render dynamic content efficiently, and Angular with Handlebars allows for flexible and user-friendly UI interactions.

By adhering to these design constraints, Travlr Getaways will have a better chance to maintain a secure, scalable, and high-performance system, providing a smooth experience for both customers and administrators.

## [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 Travlr Getaways web application is built on the MEAN stack architecture, consisting of three main components: Client, Database, and Server. Each component interacts seamlessly to ensure a secure, efficient, and scalable user experience. This can be seen in the graph provided above.

Client Component:

The Client is where user interactions begin and consists of four key sub-components:

* Client Session: Manages active user sessions and sends authentication requests to the server.
* Graphic Library: Ensures smooth rendering of the interface and enhances visual elements.
* Traveler Portfolio: Displays trip details and allows users to interact with the website.
* Web Browser: Serves as the user's gateway to access the application.

When a user launches the web application in their browser, the Client Session is initiated. This session sends a request to the Authentication Server to verify whether the user is already logged in. Once authenticated, the Traveler Portfolio loads, displaying available trips and personalized user data. During this process, the Graphic Library initializes to enhance the interface and ensure smooth rendering.

Database Component:

The Database layer consists of a single but crucial sub-component:

* MongoDB: The central database where all trip data, user profiles, and bookings are stored and managed.

When a user interacts with the website through the Traveler Portfolio, MongoDB dynamically updates to reflect any changes. This ensures that trip availability, bookings, and user preferences are stored and retrieved in real time.

Server Component:

The Server is responsible for handling user authentication, managing sessions, and ensuring smooth data flow between the Client and Database. It consists of four key sub-components:

* Authentication Server: Verifies user credentials and ensures secure access.
* Mongoose ODM: Acts as an Object-Document Mapper, enabling efficient schema management and database interactions.
* Server Session: Manages user session states and validates login credentials.
* Traveler Database: Handles trip-related data and user-specific information.

When a request is made from the Client, the Server Session checks the user’s credentials against the MongoDB database. This is facilitated through Mongoose ODM, which streamlines schema-based operations and speeds up data retrieval. The Authentication Server then confirms the user’s legitimacy and relays the authentication status back to the Client Session, allowing continued interaction with the application.

Overall, this interconnected system ensures that the **Travlr Getaways** web application remains **responsive, scalable, and secure**. By leveraging **MongoDB, Express.js, Angular, and Node.js**, the application provides a seamless experience for both travelers and administrators. Additionally, **Handlebars** is utilized to dynamically render content, reinforcing the efficiency of the MEAN stack architecture.

### 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.>