

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 | 11/11/24 | Veronica Guzman | Added to Executive Summary. Design Constraints, and System Architecture View |

## [Executive Summary](#_heading=h.35nkun2)

The web application for Travlr Getaways will be designed using the MEAN stack, ensuring a modern, scalable, and robust solution for the client. The architecture will comprise a customer-facing website, which allows users to explore and book getaways, and an administrator single-page application (SPA) for managing the site’s content and user data. The MEAN stack (MongoDB, Express, Angular, Node.js) will be used for its flexibility, performance, and strong community support, enabling both the front-end and back-end to communicate seamlessly. The customer-facing site will be a dynamic interface allowing easy navigation, while the admin SPA will enable efficient management of content with a modern user experience.

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

The development of the Travlr Getaways application is subject to certain constraints. First, the application must handle potentially high traffic, especially during peak booking periods. This requires the system to be scalable, ensuring it can handle thousands of simultaneous users. Another constraint is ensuring that data security measures are in place to protect sensitive user information, including personal data and payment details. The user interface (UI) should be simple and intuitive, with accessibility in mind. These constraints will guide the development of the web application and its components, ensuring it is efficient, secure, and user-friendly.

## [System Architecture View](#_heading=h.44sinio)

### Component Diagram



The system will be composed of several key components:

* Frontend (Angular SPA): The customer-facing website and the administrator interface will be built using Angular. Angular will communicate with the server-side Node.js API to retrieve and send data.
* Backend (Node.js and Express): The server will handle API requests, ensuring smooth communication between the front-end and the database.
* Database (MongoDB): MongoDB will store user data, booking information, and other application data, providing a flexible and scalable database solution.
* Third-party APIs: These will be integrated for payment processing and possibly other external data (e.g., maps, travel information).

The system architecture follows a three-tier model:

1. Presentation Layer (Frontend): Angular SPA that interacts with users.
2. Business Logic Layer (Backend): Node.js and Express to handle the API logic.
3. Data Layer (Database): MongoDB to store and manage data.

### 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 trips | /api/trips | Returns all active trips in the system |
| **GET** | Retrieve single trip | /api/trips/ | Returns a single trip by ID |

## 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.>