

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 4](#_Toc36198469)

[Sequence Diagram 4](#_Toc36198470)

[Class Diagram 5](#_Toc36198471)

[API Endpoints 5](#_Toc36198472)

[The User Interface 5](#_Toc36198473)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/11/24 | Tiffany McDonnell | Completion of Executive summary, Design Constraints, and System Architecture View: Component Diagram. |

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

Travrl Getaways wants us to create a full stack web application to fulfill their needs. To create this full stack application we will be using MEAN stack for the developing process. This will include utilizing multiple software programs and computer languages. We will be utilizing handlebars to control duplicate code from being used based on html files.

MongoDB will be used to store and access data. Express will be used to communicate data between the front end and database. Node. js will be used as a framework for processing multiple tasks at the same time. AnglerJS will be utilized for most of the front-end development. It is essential to the int interaction communications of the client side of the web application. Angler also allows us to create Dynamic Single-Page Applications(SPAs). SPAs allow the clients to use the application easily without having to continually load a whole page for and minor update. This will in turn make things faster and give the client a better experience.

**Requirements:**

* Must have a customer facing website
* Must have a database
* Must have an administrative single-page application
* Everything must work together appropriately
* Login in interface for user

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

* Everything must work together seamlessly
* Must be able to handle thousands of connections in a short time frame.
* Website security for personal information
* Webpage must include scalability
* Code must be able to maintain manageability for updates

We must use these constraints to help maintain a proper working website. These constraints also guide us into completing the client’s requirements. When all constraints are met then the website should run the way the client expects for the finished product. This will allow us to develop and build upon in a smooth motion and make the project finish in an expected manner.

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

The diagram below shows the architecture of the application being developed. It shows a type of web application architecture. On the clients end there will be four different components. Starting with the client’s session the two components connecting to the session are the web browser and traveler portfolio. Both these components need each other for the client’s session on the web application to be successful. The main part would the traveler portfolio since that is what the client would like to access, but that can’t be viewed without some sort of browser to load the portfolio. The last component on the client’s end is the graphic library. The travel portfolio can access the graphic library to display any images the web application is designed to show the client.

On the sever side there are also four components. The first component is the authentication server. This component also affects the client side since the client will utilize this component within the clients session. It is also the connecting point from the client session to the server session component. The server session has connections to both the traveler database and the Mongoose ODM. Following the diagram there are also symbols used to show the relationships between the different components. The little squares are used to show the ports between major components. The circles in the relationship lines between components tell us possible two different things. If a structure of an outer circle is connected to a line to one component, that component is a required interface. If the other line(s) does not have the structure of an outer circle, then it is a provided interface.

### Component Diagram



### 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.>

**References**

Acharya, D. P. (2024, November 4). *What are Single Page Applications? examples, frameworks, and more*. Geekflare. https://geekflare.com/single-page-applications/

What is component diagram? (n.d.). https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/