

<Name of Software Application>

# **CS 465 Project Software Design Document**

Version 1.0

## Table of Contents

[**CS 465 Project Software Design Document** 1](#_heading=h.gjdgxs)

[Table of Contents 2](#_heading=h.30j0zll)

[Document Revision History 2](#_heading=h.1fob9te)

[Instructions 2](#_heading=h.1fob9te)

[Executive Summary 3](#_heading=h.3znysh7)

[Design Constraints 3](#_heading=h.2et92p0)

[System Architecture View 3](#_heading=h.tyjcwt)

[Component Diagram 3](#_heading=h.3dy6vkm)

[Sequence Diagram 4](#_heading=h.1t3h5sf)

[Class Diagram 4](#_heading=h.4d34og8)

[API Endpoints 4](#_heading=h.4d34og8)

[The User Interface 4](#_heading=h.4h5b3qv007h8)

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

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/24/24 | Chuke Rupert | Initial design proposal |

## 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 client, Travlr Getaways, requires a web-based system for booking vacation packages. Such a system requires three primary components: the customer-facing front-end application, a customer profile database, and a back-end application server. This system will be constructed using the MEAN stack for its development, meaning a JavaScript stack relying on four supporting technologies. These include MongoDB for database management, Express to implement the web framework, an Angular front-end framework, and Node.js for the web server.

The front-end (or customer-facing) application is what users will interact with to preview the travel destinations, get information about trip options, and review their travel information and selections. The back-end SPA (single page application) is what the client will use to securely review customer information and access trip packages.

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

Functional constraints:

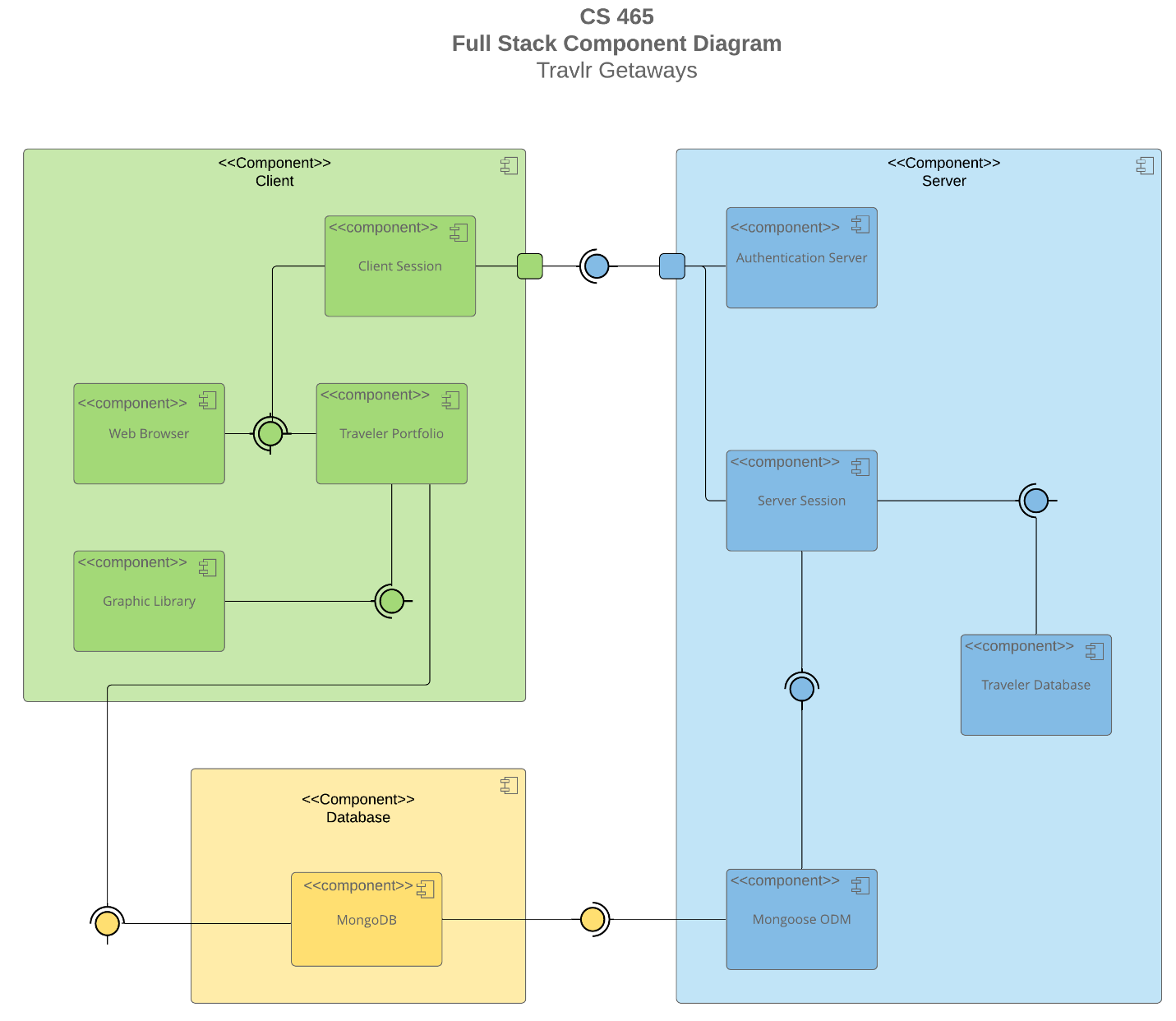
* Customers must be able to navigate between pages from the header and footer banners
* Customers must be able to submit a contact/feedback card from a “Contact us” page
* Client must be able to log-in to secure client-side SPA
* Client must be able to generate and review customer struct object data associated with [0..many] trip objects
* Client must be able to generate and review trip object data and associate it with exactly one customer object
* No requirements indicated at this time for updating travel/food/room option offerings at this time. {For future consideration}

Non-Functional constraints:

* Aunthentication, authorization, and accounting security features for client-facing SPA
* Full stack design including customer-facing front end web application, back end server side single page application, and storage database.
* Pages to include a home landing page, trips/travel options, rooming options, meal options, a news feed page, about the client page, and a contact page.
* Currently no requirements indicated from client regarding scalability, maintainability, or performance

## [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 will be composed of three major components: A customer-facing web application (highlighted in green), a client-side server, and a database storage component. Customer and travel data is stored on the database for access in later sessions. Users interface with this data using the web application via a browser. Meanwhile, the server requires an interface from the database as well in order to present and interact with the 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 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.>