

<Name of Software Application>

# **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 |
| --- | --- | --- | --- |
| 1.2 | 4/21/2024 | Mitchell Fitzsimmons | Made recommended changes from the feedback on Brightspace, and I finished the User Interface aspect upon completing the Security 7-1 assignment |

## 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)

With the company Travlr we are tasked with making, refactoring, and optimizing the current web pages of theirs given to us. To do this we have decided to take the given static HTML pages and refactor it to instead use the MEAN stack which contains the use of Node, Angular, and Express JavaScript. We will also use MongoDB to finish completing the process. On the customer’s of Travlr’s side the website will have more capabilities, and user friendly functionality with quicker loading pages. The framework of the customer side is the JavaScript and MongoDB mentioned prior. The server is written in Node JavaScript.

The front end of the application is where the Angular Javascript is used, and it will give a rich functionality on the client side of the application. On the backend is where the Express Javascript is used and assists in communicating with MongoDB, and together these two sides communicate leading to high functioning capabilities. Between the front end and the back end of the application communicating with the Node.js server and MongoDB, this application will have unique functionality that end users will enjoy more than the previous versions.

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

There are many design constraints to keep in mind with the production of the application for Travlr. At the most general level there are always two things to keep in mind. First to begin with, are the outlined desired systems to be used to produce the application. The desired systems of Node, Angular, and Express JavaScript outlined how the server was going to be stood up, how the pages are loaded and more. Another thing to keep in mind is both keeping the project in line with the specified timeline, as well as any specific budget.

Another constraint and potential future issue is the use of MongoDB. MongoDB is a NoSQL database that is ideally used for IoT, and web-facing applications as it’s storage is document based and different than that of a relational database. So while for the current iteration of the project it is fine, as the database grows and potentially more databases and tables are needed for finances, and other things, keeping ACID principles will become complex. The one positive of MongoDB is that for the current iteration of the project is that it can ‘infinitely scale’ given the necessary memory and compute resources.

## [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 three connected diagrams above show the system as a whole. The smallest of the three diagrams is the database section, where MongoDB will be ran and store necessary information. This should only be able to be accessed by authorized users. The next main component is the client component. This is the customer facing side of the application where users will interact with the program and make potential trips. The last major component is the server component. This is where the communication from the client component to the database component will occur. That includes authentication, new database entries, and the server session.

There are a few sub components of major importance within each main component. The database component really only consists of the MongoDB database that was discussed prior. The client component consists of intuitive things like a graphics library to assist in the visual aspect of the website, the Travlr trip component where important information is displayed, and finally not so intuitive is the client session. The next set of subcomponents is under the server. The server will assist in major functionality components including the CRUD operations of the Travlr database, authenticating users, and giving them proper authorization to make changes to the Travlr database, and ensuring communication from the client session, to the server session, to the mongoose ODM, to the MongoDB previously discussed.

### Sequence Diagram

A diagram of a trade end

Description automatically generated

Controller/Model

Route

HTTP Client

Controller

Browser/View/Template

The sequence diagram follows the actions and flow of information that occurs when this system is in use. The end user begins at a specific webpage the home, or whatever page and can access a route. The route then processes the request, gives it to the view then it is passed to the controller, so the information can then be sent from the browser out of the client facing aspects of the project. The server components then bridges the gap between the end user and the database, MongoDB. The server passes information and parameters to MongoDB where it is stored through the same process of routes and controllers, and then MongoDB returns information to the server. Finally, given the information the display is given to the user via use of HTTP.

## Class Diagram

A diagram of a travel geoways class diagram

Description automatically generated

The class diagram above shows how information is stored, and how the different objects interact with one another. User information is stored in the base member account, and users can become admins when the proper change control and reasoning is in place. We can see this through the displayed inheritance and difference between MemberAccount, Membership Admin, and Travel Agent. These base users are then travel agents who then through the use of the system can book traveling trips and make itineraries for their customers. TripInfo is base class that gets more granular with aspects like Cruise, Flight, Hotel info and booking classes displayed to the left of it. The information regarding the specific trip, and planned details are then saved to the database which is accessible by the Travel Agent class, while the user has the ability to read their specific trips’ information.

## [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/trip> | <Returns all active things> |
| **GET** | <Retrieve single trip> | </api/trip/:tripId> | <Returns single thing instance, identified by the thing ID passed on the request URL> |
| **POST** | Create new single entry | </api/trip/:tripId> | Create new single entry |
| **DELETE** | Remove a single entry | </api/trip/:tripId> | Remove a single entry |
| **DELETE** | Remove a list of entries | </api/trip> | Remove a list of entries |
| **PUT** | Update or replace an entry | </api/trip/:tripId> | Update or replace an entry |

## The User Interface

The big difference between the angular framework and the express framework is that angular framework is primarily used for developing front end single page applications and dynamic web applications which are loaded from a set of files, while the express framework is used in the backend and processes requests that are then used to take the present webpage and process the processed request and return the users request pages.

The advantage to single page applications, SPAs, is that they are typically quicker and offer better functionality resulting in a better user experience. They work in the browser which makes them faster than their counterparts who have to load and communicate from and with the server.

The process of the system communicating and using the API to process and perform GET and PUT methods is vital. Checking credentials, permissions, input verification in the system ensures the least amount of errors. Typical errors that can be expected are when these things are missing, not being able to access the database, objects, or functional errors from wrong data types or blank fields can stop processes from occurring.