

<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 | <9/20/25> | <Reginald True> | <Travler project |

## 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 web app was built using the MEAN stack, which means it uses MongoDB, Express, Angular, and Node.js. This app has two main parts. The first part is for customers, where people can look at different travel trips and read about destinations. The second part is for administrators, who can log in to a special screen called a single-page application (SPA) to add, edit, or delete trips. The front end (Angular) connects to the back end (Express and Node.js) through special web addresses called APIs. The data for all the trips and users is stored in MongoDB. This setup helps the app run fast and makes it easy to update and use.

The Travlr Getaways website was built with a set of tools called the MEAN stack. This stands for MongoDB, Express, Angular, and Node.js. These tools work together to make the site fast and easy to use.

The customer side of the website lets people look at trips, see details, and contact the company. It uses Express and Handlebars to show pages from the server.

The admin side is a Single-Page App made with Angular. This means admins can see all their tools on one page without it reloading each time. Admins can log in, add new trips, edit old ones, and delete trips. The SPA talks to the Express server through the internet to send and get data from the MongoDB database.

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

There were a few limits, or constraints, when building the Travlr Getaways app. First, the app had to be made using the MEAN stack, so we used JavaScript for everything. That made it easier to build but also meant we had to follow JavaScript’s rules. The app also needed two sides: one for customers and one for admins. This meant we had to make sure the admin side was safe with logins and passwords. Another limit was that the app had to work well on different devices, like phones and computers. Because of these limits, we focused on making the app simple, secure, and easy to use everywhere.

When making the Travlr Getaways site, there were some rules and limits we had to follow:

* It had to use the MEAN stack, which means only JavaScript and TypeScript were allowed.
* The app needed to be safe so only admins could change data.
* It had to be fast and work on phones, tablets, and computers.
* It needed to run on a Node.js server and use MongoDB for all the data.

These limits helped make the site secure and reliable, but also meant we had to plan carefully so everything worked smoothly together.

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

The Travlr Getaways app has three main parts. The first is the front end, which is the part users see and click on, made with Angular. The second is the server, built with Express and Node.js, which takes care of requests from users and talks to the database. The third part is the database, made with MongoDB, which stores all the trip and user data. The user’s browser talks to the Angular front end, which sends messages to the Express server. The server then gets the right data from MongoDB and sends it back. The admin part uses the same system but lets the user make changes, like adding or updating trips.

The Travlr Getaways app has three main parts:

Front End :

Regular visitors use the Express pages made with Handlebars.

Admins use the Angular SPA to log in and manage trips.

Middle Layer :

The Node.js and Express server handles routes and logic.

It sends and receives data from the database.

Back End :

The MongoDB database keeps all trip and user information safe.

Mongoose is used to connect the server to the database.

Each layer works together. The front end sends requests, the server processes them, and the database stores the information.

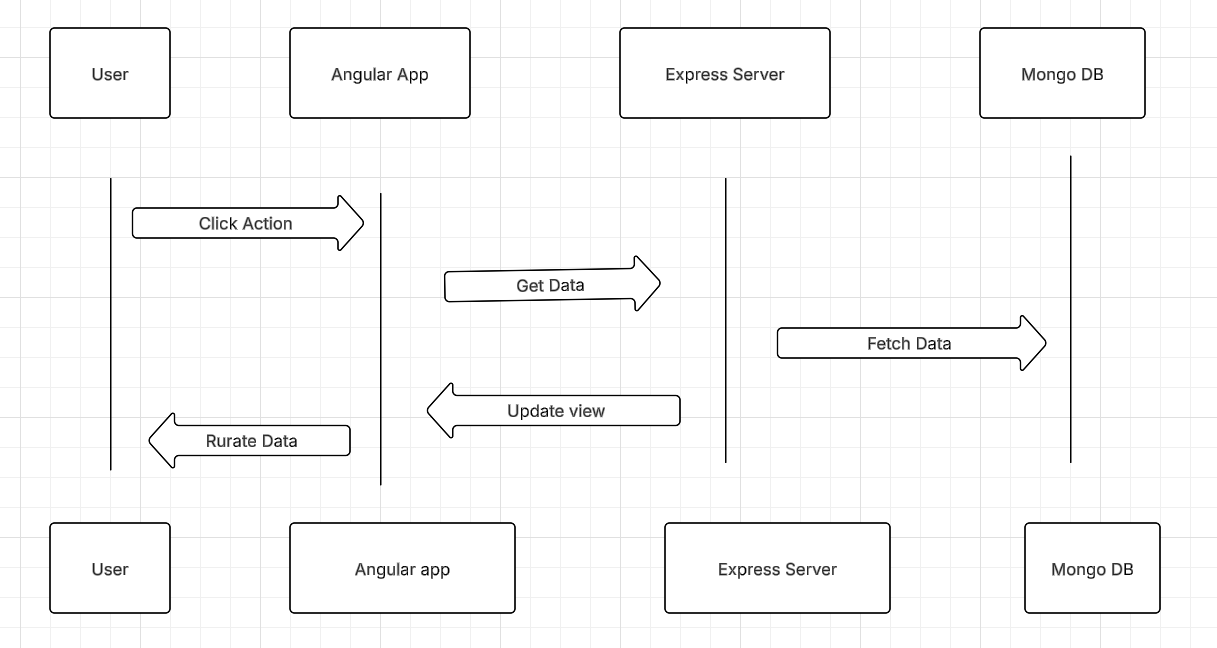
### Component Diagram



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).

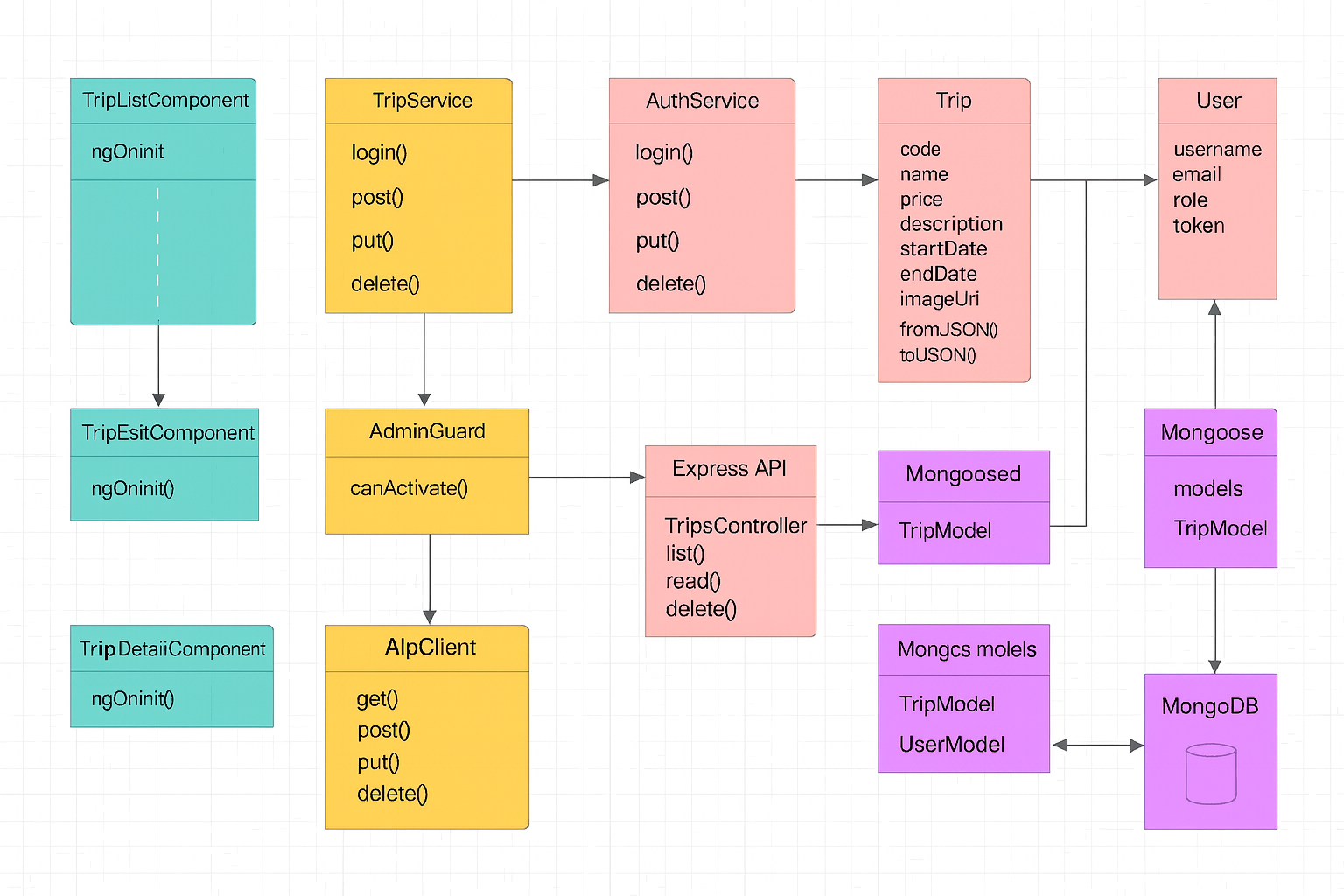
The **Angular project structure** organizes components, services, and routing files for clear separation of features. Each component controls a specific part of the interface, such as the trip list or edit page. The **Express project structure**, by contrast, focuses on routes, controllers, and models that define how the server handles API requests. The Angular SPA is more interactive and responsive than a basic website because it updates data dynamically without reloading. Testing involved confirming that the Angular front end correctly sends **GET** and **PUT** requests to the Express API and that the data changes in **MongoDB** appear instantly in the web interface. >

### Sequence Diagram



When someone uses the app, the steps are simple. The user clicks on something in the browser. The Angular app sends a message to the Express server to get the data it needs. The server asks the MongoDB database for that data. Once MongoDB sends it back, the server gives it to Angular, and the user sees it on their screen. For admins, when they change or add a trip, the same steps happen, but the server also updates the database before showing the new information. This makes the app work quickly without needing to reload the page.

## Class Diagram



In the Travlr Getaways app, there are a few main classes that help organize the code. The **Trip class** holds details about each travel trip, such as the trip name, code, cost, and description. The **User class** stores information about admins and their logins. The **Angular components and services** work together to display trips on the screen and send data between the front end and the back end. These classes make it easy to manage information and connect everything inside the app.

## [API](#_heading=h.2jxsxqh) Endpoints

| **Method** | **Purpose** | **URL** | **Notes** |
| --- | --- | --- | --- |
| **GET** | <Retrieve list of things> | </api/things> | <Returns all active things> |
| **GET**  **Post**  **Put**  **Delete** | <Retrieve single thing>  Add a new trip  Update a trip  Delete a trip | </api/things/:thingId>  /api/trips  /api/trips/:tripCode  /api/trips/:tripCode | <Returns single thing instance, identified by the thing ID passed on the request URL> |

## The User Interface

The Angular project structure organizes components, services, and routing files for clear separation of features. Each component controls a specific part of the interface, such as the trip list or edit page. The Express project structure, by contrast, focuses on routes, controllers, and models that define how the server handles API requests. The Angular SPA is more interactive and responsive than a basic website because it updates data dynamically without reloading. Testing involved confirming that the Angular front end correctly sends GET and PUT requests to the Express API and that the data changes in MongoDB appear instantly in the web interface.

The Angular SPA has separate parts :

trip-list.component shows all trips.

trip-detail.component shows one trip.

trip-edit.component lets you change a trip.

trip-add.component lets you make a new trip.

The Express app is simpler. It has folders for routes, controllers, and views.

The SPA is better for admin work because it can change data without reloading the whole page. It uses HttpClient to send GET and PUT requests to the Express API. Testing means making sure the data updates correctly in the database and that both the website and the SPA show the same trip details.