

Travel Booking

# **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 | 05/21/25 | Brandon Myers | -Created an app server folder  -Moved the routes and views folder to the  app\_server folder  -Created folder for controllers and routes  - Created a handlebar view.  - Created header and footer hbs |

## 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 application is being created with MEAN stack. Mean stands for MongoDB, Express.js, AngularJS, and Node.js. The user side of the application allows the user to book vacation rentals. The administration side of the application lets the owners of the property edit or delete the posting.

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

One of the main design constraints is there has to make sure the application is compatible with the different versions of web browsers. The browsers that it needs to be compatible with is Internet explorer, foxfire, google chrome, safari, and opera. The application also has to be compatible with different screen sizes. Another constraint is the security, the security has to be strong enough to protect the customers personal information.

## [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 diagram shows the Travlr Getaways program. The diagram shows the clint side, the database, and the server side. The client side deals with the user interface and displays the users data. The database side stores the information. The server side is responsible for connecting the database and the API. The diagram shows the flow of communication.

### Sequence Diagram

A screenshot of a computer

AI-generated content may be incorrect.

The process starts with the actor. The actor uses the browser to make a request. The route takes the request on the server side and compares the URL path. The route guides the parts to the controller. From there the system communicates with the model to complete the requested action. The controller sets up the request. Next the HTTP client sends a request to the server side. The server side takes the request and communicates with the MongoDB and the data is read or modified.

A screenshot of a computer

AI-generated content may be incorrect.

The Itinerary represents the travel plan. The MemberAccount has the members account information like account number, and frequent flyer status. The TripInfo box has information like packages, flights, hotels, and cruises. CruiseInfo has the name, cabin type, and price. The FlightInfo has the name, seat class, and price. The HotelInfo has the name, star rating, location, room requests and price. Itinerary has starting and return dates as well as the origin and the destination. TravelerInfo has information about the traveler

## [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> |
| **POST** | Create a new list of records | /api/things | Create a new list of records |
| **POST** | Create a single records | /api/things/:thingId | Create a new record |
| **PUT** | Update the list of records |  | Update and replace the list of records |
| **PUT** | Update a single record |  | Update and replace a single record |
| **PATCH** | Update the full list of records |  | Update the list of records |
| **PATCH** | Update a single thing |  | Update a single record |
| **DELETE** | Delete all of the list of records |  | Delete all of the records |
| **Delete** | Delete a single record |  | Delete a single record |

## 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.>

Angular project structure are organized into modules. The modules group the components, different functionalities, and services together. The components define the behavior of the parts of the application. The service handles the business logic, data handling, and API calls. The routing manages the navigation within the application. The assets is a folder that has the images, and stylesheets.

The express project structure has routes which define the API endpoints. There is also the controllers which are functions that handle the request logic. The model is a representation of the data structure. The middleware has the security functions, the log in access, and authentication. There is also the static files which has the resources like the images and the CSS.

SPA like Angular load singular HTML files which help with eliminating page reloads. The benefit is a smoother user interaction. Angular does the routing on the user side which gives a better user experience. The express structure is easier to set up but it doesn’t have the same interactivity and efficiency as Angular.