

Travlr Getaways

# **CS 465 Project Software Design Document**

Version 2.0

## 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 |
| --- | --- | --- | --- |
| 2.0 | 12/8/2023 | Vince Taliaferro | Adding SPA User Interface |

## 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 was developed using the MEAN stack, incorporating Angular.js, Express.js, Node.js, and MongoDB. Node.js serves as the framework on the server side, while Angular.js handles the client-side responsibilities. MongoDB is employed as the database, encapsulating all components. The customer-facing front end enables users to interact with the database and server via the interface. To mitigate data loss in case of internet disruptions, a single-page administrator application was implemented for enhanced speed compared to client-side applications. MongoDB facilitates data transfer between the client and server, supporting multi-dimensional data for increased availability and performance. Express.js is utilized comprehensively in the back end, managing various aspects of the application. Angular.js, functioning as the front end of the MEAN stack, constructs the dynamic application. Node.js powers the server, utilizing an event-driven I/O model for efficient operation.

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

The project faces three critical design constraints: deadline, manpower, and budget. Of these, budget holds significant influence in potentially causing project delays. Adherence to the allocated budget is imperative for the project's success. When confronted with a tight timeframe for project completion, there is a likelihood of encountering manpower challenges, potentially leading to delays. An increased need for manpower, while essential for meeting tight deadlines, also translates to a higher budget. These three constraints are intricately connected, and a setback in one area can have cascading effects on the others. The interdependence of deadline, manpower, and budget underscores the necessity for a balanced and well-coordinated project management approach.

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

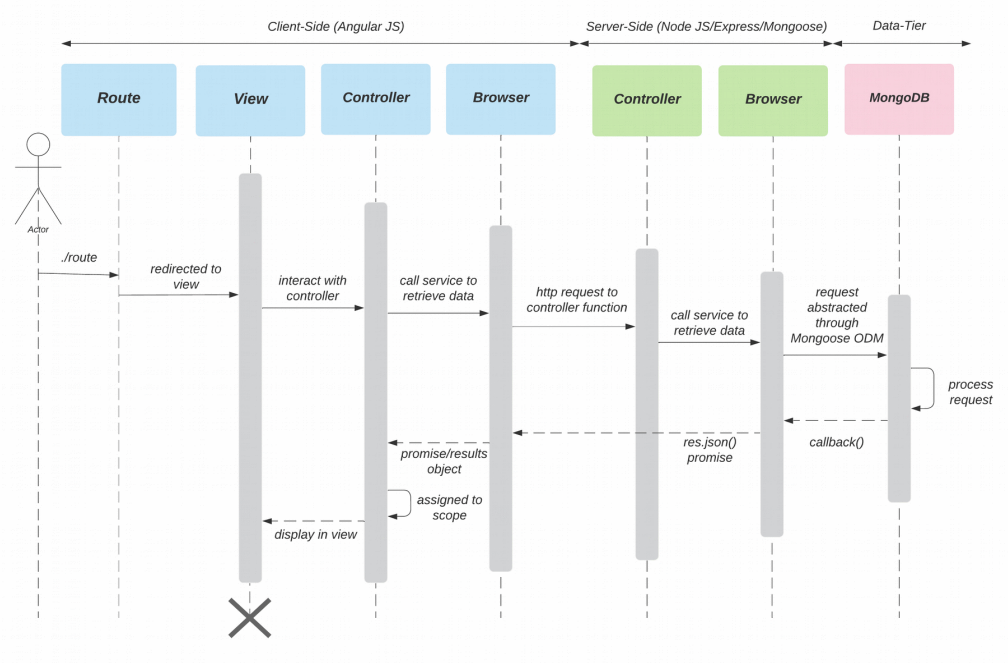
### Component Diagram



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).

In this diagram, the key components are the server, database, and client. These components form a vital interaction network where the server and client are connected through ports. The database plays a crucial role by supplying information to the server, which can then be relayed to the client. The server assumes the critical responsibility of authenticating the client, ensuring secure and authorized access. Additionally, the database plays a pivotal role in storing information and furnishing it upon request, underscoring its significance in supporting the overall functionality of the system. The collaborative interaction of these components is essential for the seamless flow of data and the effective operation of the system.

### Sequence Diagram

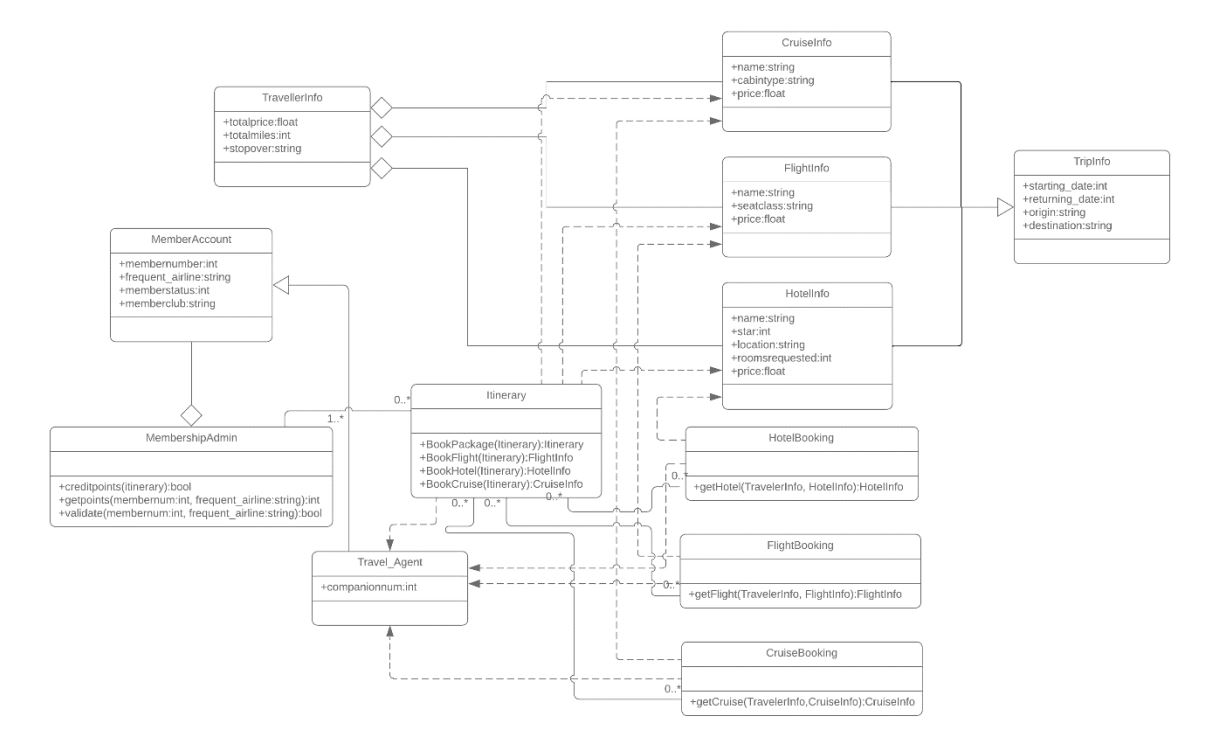


The diagram comprises three primary components: the client-side, server-side, and data-tier. When a user initiates a sign-in process on the web application, the system directs them to a specific view via a designated route. Subsequently, this view initiates a request for information from the controller. The controller, in turn, establishes communication with the server-side controller through the browser.

The server-side controller, responsible for managing business logic, then triggers a request for information from the data-tier, typically a MongoDB database. This request is transmitted through the browser connection. Subsequently, the data-tier processes the request, retrieves the necessary data from the MongoDB database, and sends it back through the browser connection to the server-side controller.

Upon receipt of the data, the server-side controller forwards it to the client-side controller. The client-side controller, in turn, communicates with the view to seamlessly present the retrieved information. In essence, the user's sign-in action sets in motion a series of interactions involving views, controllers, and the data-tier, orchestrating the retrieval, processing, and display of pertinent data on the web application.

## Class Diagram



The diagram shows how different parts of the Travlr Getaways web application work together. The TripInfo class holds all the trip details, including CruiseInfo, FlightInfo, and HotelInfo. The Itinerary class handles booking different parts of the trip, like packages, hotels, cruises, and flights. The TravelAgent class is like a manager, organizing and keeping track of all the trip info for the member.

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

A white rectangular object with text

Description automatically generated with medium confidence

## The User Interface

The key difference between traditional Express HTML and Angular project structures lies in how they load and handle content. Express HTML loads only the specific part of the application the user is currently viewing, fetching the necessary HTML from the server when they switch. In contrast with Angular Single Page Applications (SPAs) load the entire application when the user opens it, allowing seamless transitions between sections without constant server communication.

To illustrate, think of an Express HTML app as a one-room building that gets disassembled and reassembled around you when you switch rooms. An SPA is like a multi-room building where you can instantly teleport to a different room.

SPAs offer advantages such as faster loading times, smoother user experience, and reduced bandwidth usage. They have drawbacks such as potential issues with search engines and increased browser resource usage.

Testing SPAs typically involves checking API integrations and employing end-to-end testing. One common question is whether SPAs can be integrated alongside multi-page applications.