

# **Matt Knutson 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 | 09/22/2024 | Matt Knutson | Switching from a static HTML website to a dynamic website using a MEAN stack, MVC pattern, and a SPA web implementation. |

## [Executive Summary](#_heading=h.35nkun2)

The client Travlr Getaways hosts a website that allows its customers to search for vacation packages around the world. The company’s current webpage is built using static HTML. Travlr Getaways is looking to transition their site into a dynamically interactive environment so that each customer has a unique and customized experience that is targeted at their needs. In order to accomplish these tasks, the website will use the MEAN (MongoDB, Express, Angular, Node.js) programming stack and will utilize the MVC (Model, View, Controller) framework.

Switching the website to a dynamic structure will allow data to be manipulated as it’s passed through the code, facilitating a unique experience for each customer. Due to the nature of the business, this transition will be extremely important. Not only will trip package prices vary throughout the year, but different locations will change their offers to accommodate the seasons. Also, customers will want their location updated when they visit the site, and future vacation recommendations based on their purchasing and viewing history.

The website will use the SPA (Single Page Application) template, rather than an MPA (Multiple Page Application) template. This will provide a smoother transition for the user when they interact with and update a page or move back and forth between pages. Instead of an entirely new page being spawned for every action by the user, the SPA allows for partial page adjustments. This creates a much quicker and more fluid user experience.

The MEAN programming stack will be used to build the application, and it will be deployed in an MVC fashion. Using MEAN provides the opportunity to use JavaScript throughout the entire stack. This will allow for faster runtimes at the client’s end, but also makes it much easier on the programmers building and maintaining the application. Every component of the MEAN stack is also open-source software. This not only means it is much cheaper, or free, to use, but it is also maintained and updated by a large community of professional programmers. Separating the application into an MVC framework will break the code up into three main categories: the Model, View, and Controller. This, combined with using JavaScript, will eliminate a large portion of the code that is needed and make the programs structure easier to maneuver through when updating, or adding to, the code base. The MVC pattern also heightens security throughout the entire application. This is because the user only ever interacts directly with the Controller. This creates a middle-man dynamic between the user, Model, and View allowing for a centralized security strategy located in the Controller. The database is also isolated within the MVC framework, and only interacts with the Model, allowing an opportunity for more security layers.

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

This architecture provides many stable and efficient solutions, but there are several constraints ingrained in the design:

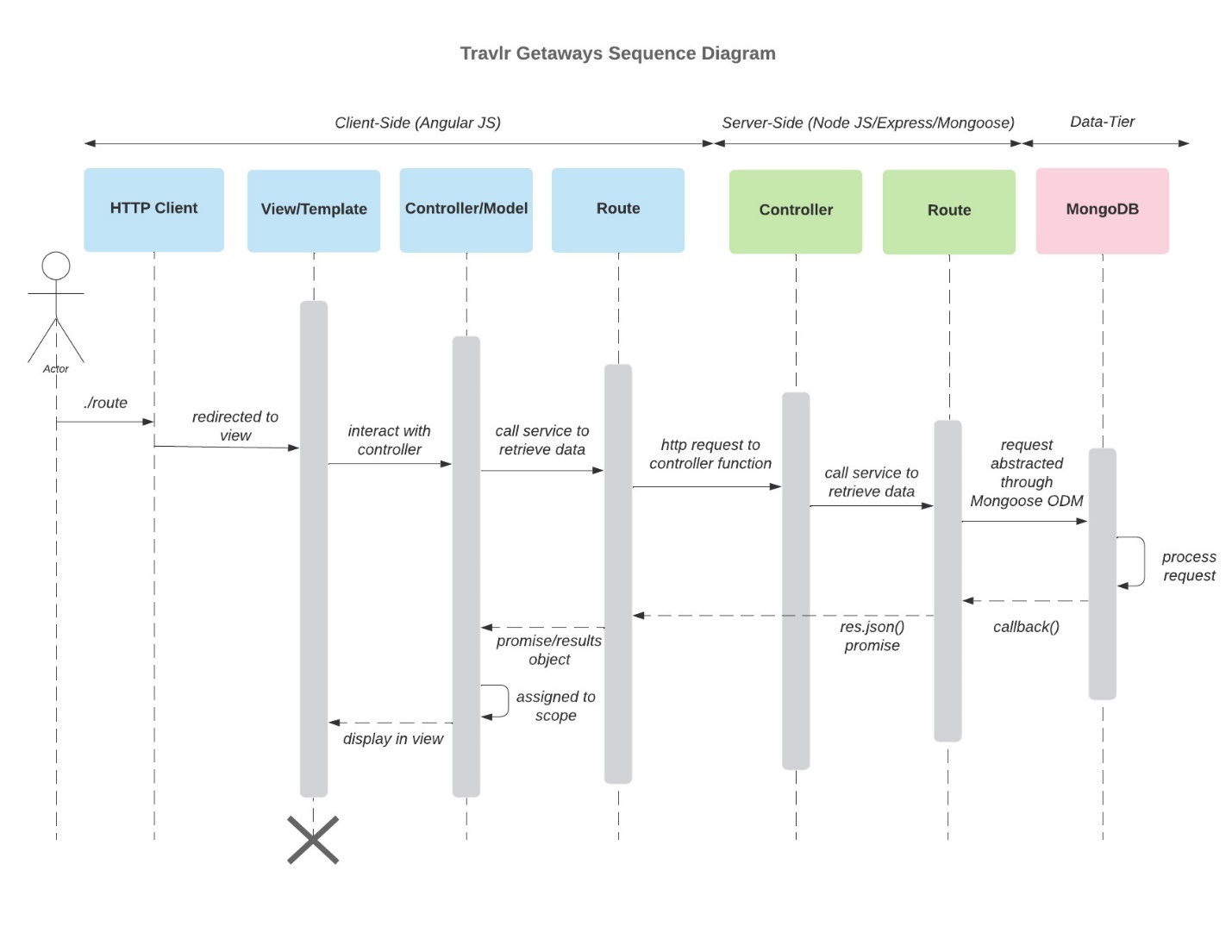
* Due to unexpected variables, a dynamic website could take longer to fully load than a static website.
* The MVC pattern will add a large amount of complexity to the project.
* Depending on the size of the website, the SPA could push a large number of resources to the client’s device.
* As the website grows, the MEAN stack can become less effeceint than other stack variants such as MERN, MEVN, or VIPER.

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

### Component DiagramPlease see the hyperlinked Word Document "CS 465 Full Stack Component Diagram Text Version" for alternative text.

From the diagram above, we can see that the running system is separated into three parts: the Client, the Server, and the MongoDB database. The client’s session is graphically facilitated by the Web Browser and the user is authenticated through the Authentication Server. After the user is verified, the Server Session gains access to the Traveler Database and can in turn feed that data to Mongoose. Mongoose securely interacts with the MongoDB database from the Server’s side, while the Traveler Portfolio sends and receives data to MongoDB from the Client’s side. The Graphics Library interacts with the data and then provides a User Experience to the Web Browser.

### Sequence Diagram



The Sequence Diagram helps visualize the flow of data throughout the system and shows the amount of time each component or process is active. The Travlr Sequence Diagram starts with a user putting in a request to a HTTP Client. The request is redirected to the websites View/Browser and the View interacts with the Model via the Controller. The Model then routes this information to the main Controller so that it can request the required data from the database. Mongoose then acts as a middleman between the main Controller and the database (MongoDB), sending the properly formatted request directly to the database. The database processes the request and then sends it back through the proper channels. The end result is the requested data being processed and sent back to the View where it is displayed as a webpage to the user.

## Class Diagram

**A diagram of a travel company

Description automatically generated**

## The Travlr Class Diagram shows the relationships between the objects, or Classes, in the system at a code level. Each object is labeled with its name and lists the variables it will use within the code. Lines and arrows are used to show how each object is associated with the other objects in the application. The diagram above shows that the TripInfo object gathers all its data from CruiseInfo, FlightInfo, & HotelInfo respectively. The Itinerary object is also built from data gathered from these three variables. The Travel\_Agent object has access to this information as well and assists members in booking trips and verifies the customers’ credentials and membership status.

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

| **Method** | **Purpose** | **URL** | **Notes** |
| --- | --- | --- | --- |
| **GET** | <Retrieve a list of trips> | </app\_api/controllers/trips> | <Returns all active trips> |
| **GET** | <Retrieve a single trip> | </app\_api/controllers/trips/:tripId> | <Returns a single trip instance, identified by the trip ID passed on the request URL> |
| **POST** | <Add a single trip> | </app\_api/controllers/trips/:tripInfo> | <Adds a single trip instance, identified by the trip info template passed on the request URL> |
| **PUT** | <Update an existing trip> | </app\_api/controllers/trips/:tripInfo> | <Updates a single trip instance, identified by the trip info template passed on the request URL> |

By implementing Angular into the project, the application is now using a structure that separates its tasks into Components. Each Component consists of an HTML, TypeScript, and CSS file, further separating the application’s logic into smaller segments. Using this framework and file system facilitates the integration of Buttons and Call Backs, creating a dynamic webpage that can be modified directly from the user interface. At the coding level, the Angular implementation is built in Administrator mode, adding an authorization element to the system as well.

Switching to a SPA framework allows the website to load directly onto the user’s device, rather than requesting a new HTML page from the server each time an action is performed. Since the website’s data and resources are already loaded on a local device, the transition between views and links happens much quicker and more fluidly. This arrangement also allows the user to access the website offline, and any locally changed data will be updated to the database as soon as online access is re-established. The downfall to using a SPA is that it can consume a lot of the user’s resources, since the entire website is loaded on the individual’s device. But, as time goes by, the standard sizes for RAM and storage grow exponentially, slowly eliminating this issue from the equation.

A program called Postman is used to test functionality between the Travlr website’s API and the MongoDB database. Within the Postman application, a developer can access the Travlr website using the GET window and the host’s address. From the GET window, the website is viewed and tested to guarantee that the database is synced to the site. Likewise, by switching from GET to PUT in the search bar, the developer can add data to the database from the UI  and observe how the program handles such actions.

**Citations**

Advantages And Disadvantages Of Static Website. Crystallize. (2024).

https://crystallize.com/answers/tech-dev/advantages-and-disadvantages-of-static

Angular Tutorial. Angular. (2024).

https://v17.angular.io/tutorial

Conceptual Structure. TIBCO Product Documentation. (2024).

https://docs.tibco.com/pub/tpm-rest/1.1.0/doc/tpm-rest/html/GUID-84C5C688-4620-444B-850B-4542697ED368.html

Eddie, J. (2022, January 7). Why MEAN Stack s Best or Web Development. Medium.

https://enlear.academy/why-mean-stack-is-best-for-web-development-55b7e4a00d72

Engineering. (2021, April 24). MEAN and MERN Stacks: Full Stack JavaScript Development Explained. AltexSoft.

https://www.altexsoft.com/blog/mean-mern-javascript-full-stack

Guilizzoni, P. (2024). What Are Wireframes And Why Are They Used? Balsamiq Wireframing Academy.

https://balsamiq.com/learn/articles/what-are-wireframes

Hernandez, R. D. (2021, April 19). The Model View Controller Pattern – MVC Architecture And Frameworks Explained. freeCodeCamp.org.

https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained

Juviler, J. (2022, March 30). Static vs. Dynamic Websites: Here’s The Difference. HubSpot Blog.

https://blog.hubspot.com/website/static-vs-dynamic-website

Kendal, R. (2019, March 22). How To Build A RESTful Node.js API Server Using JSON Files. Rob Kendal Articles RSS.

https://robkendal.co.uk/blog/how-to-build-a-restful-node-js-api-server-using-json-files

Kryzhanovska, A. (2022, March 23). Pros & Cons Of Building Single Page Applications In 2023. Gearheart.

https://gearheart.io/blog/pros-and-cons-building-single-page-applications-2019

Mongoose. Mongoose ODM v8.7.0. (2024).

https://mongoosejs.com

MozDevNet. (2024). Cross-Origin Resource Sharing (CORS) - HTTP: MDN. MDN Web Docs.

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Neskuba, A. (2024, August 16). Web Application Architecture: The Complete Guide 2024. Intellectsoft Blog.

https://www.intellectsoft.net/blog/web-application-architecture

Ramos, M. (2023, October 24). MEAN Stack Explained. Kinsta®.

https://kinsta.com/blog/mean-stack

Waheed, A. (2024, July 12). Step-by-step Postman Tutorial: Using Postman For API Testing. Apidog Blog.

https://apidog.com/blog/how-to-use-postman-for-api-testing

What Is MEAN Stack? Codeworks. (2024, June 21).

https://codeworks.me/blog/what-is-mean-stack

Williams, A. (2023, January 10). What Is The MEAN Stack? A Simple 2024 Guide. CareerFoundry.

https://careerfoundry.com/en/blog/web-development/mean-stack