

Travlr Getaways Booking

# **CS 465 Project Software Design Document**

Version 1.0

## Table of Contents

[**CS 465 Project Software Design Document** 1](#_Toc172380687)

[Table of Contents 2](#_Toc172380688)

[Document Revision History 2](#_Toc172380689)

[Instructions 2](#_Toc172380690)

[Executive Summary 3](#_Toc172380691)

[Design Constraints 4](#_Toc172380692)

[System Architecture View 5](#_Toc172380693)

[Component Diagram 5](#_Toc172380694)

[Sequence Diagram 6](#_Toc172380695)

[Class Diagram 6](#_Toc172380696)

[API Endpoints 7](#_Toc172380697)

[The User Interface 7](#_Toc172380698)

[References 8](#_Toc172380699)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/21/04 | Justin Starr | Updated Executive Summary, Design Constraints, and System Architecture View |

## 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 architecture of the web application that will be developed for the client Travelr Getaways is a full-stack web application consisting of a customer-facing website, a database, and an administrative single-page application (SPA) that will all work together. To achieve this, a combination of technologies, called a MEAN stack will be implemented to create a database and web server that will operate on the back end, contain application logic and control in the middle of the stack, and will go all the way to the front end with a user interface (Harber & Holmes, 2019).

The MEAN stack is a complete JavaScript stack that is comprised of four main technologies, allowing for the creation and deployment of dynamic web applications. The M in MEAN stands for MongoDB, which is the database, the E stands for Express, which is the web framework, the A stands for Angular, which is used for the front-end framework, and the N stands for Node.js, which is used for the web server.

MongoDB is a NoSQL database that allows the storing of data in JSON-like documents, making it more flexible, and it provides easy scalability for web applications as well as high performance (Eddie, 2022).

Express is the server framework that can serve static HTML content and support using JavaScript to code the necessary logic and behaviors required to meet the client requirements. It is a lightweight framework that provides an easy-to-use interface, which connects NodeJS with MongoDB, allowing you to use RESTful routes that directly connect HTML content to data in the application's database (Eddie, 2022).

AngularJS is used to develop single-page applications, which in the case of this application will be used for the administrative page. Typically, a MEAN stack application’s architecture includes a representational state transfer (REST) API that feeds the single-page application (Harber & Holmes, 2019). It essentially creates a stateless interface to the database that enables applications, such as Angular SPA to work with the data, and it uses JSON throughout the stack, including in the database (Harber & Holmes, 2019).

NodeJS is a platform that is event-driven and it is a non-blocking input/output model, which keeps it lightweight and efficient (Eddie, 2022). This means that it can handle multiple requests at the same time without blocking other code execution (Ramos, 2023).

Express and Node.js work in conjunction with each other to deliver HTML content directly from the server to the client-facing side of the application, which can also be done using templates, such as handlebars. The administrative single-page application will be implemented with Angular on the front end. Using a REST API that is built using Express and Node.js, content can be routed to both the client-facing side of the application and the Angular single-page application. Changes made from the administrative side can be seen on the customer-facing side almost immediately after making any changes.

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

It is essential to consider that while developing the Travelr Getaways Booking application there are design constraints. For example, JavaScript applications are difficult to crawl and index, and search engines, while they look at HTML content on a page, they do not execute or download much JavaScript (Harber & Holmes, 2019). Therefore, it can not be determined precisely how much of the content will be indexed if all of the content is served via a JavaScript application (Harber & Holmes, 2019). A similar issue exists is that automatic previews from social-sharing sites do not work well, similarly because they look at the HTML of the page you are trying to link, where the social-sharing sites attempt to extract some relevant text and images. Just like the search engine problem, they do not run JavaScript on the page, and therefore any JavaScript content that is served up will not be seen (Harber & Holmes, 2019). There are workarounds for these problems, however, they require a large amount of effort, and maintaining them can become problematic (Harber & Holmes, 2019). These concerns are significant because the type of application that is being built would benefit from the exposure that can be gained from social media sharing and search engine traffic. The more people that can either find the web application or hear about it from other people, the better the business will likely grow.

Additionally, another consideration is that SPAs have a slower first-page load than server-based applications. (Harber & Holmes, 2019). This is because SPAs have to bring down the framework and application code before the required view can be rendered in the browser as HTML, whereas server-based applications only have to push the required HTML (Harber & Holmes, 2019). However, based on the size of the client’s application that is being built, this for the time being, should not be much of a problem.

Another consideration of the design is the database itself. Earlier versions of MongoDB do not support ACID transactions, which could result in the loss of data if while processing multiple requests, the application becomes unstable (Eddie, 2022). However, MongoDB version 4.0 began supporting ACID transactions which help to ensure database transactions are processed in a reliable way.

Lastly, implementing a MEAN stack architecture can have drawbacks such as performance issues. Node.js executes JavaScript code one task at a time, unlike how it handles other tasks. While this single-threaded approach does have its advantages, it could potentially lead to bottlenecking during CPU-intensive tasks (Ramos, 2023).

## [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).

As seen in the above component diagram, we have three main components, which are the Client, Server, and Database components. Within these three components contain sub-components that are responsible for one clear aim, and on a need-to-know basis, only interact with other essential elements (Visual Paradigm, 2024). Each component can be recognized by the component icon located in the top right corner of every component rectangle.

The colored circles with half circles around them represent interfaces that indicate that they are either required by one component or provided by a component. The direction for which that half circle around the interface indicates that it is pointing to the required interface, and the other component that the interface connects to is the provided component. Therefore, we can see that the Client component requires the Server component, and therefore, the Server component is provided to the Client component. Also, we see that both the Client and Server components require the Database component, and the Database component is provided to both the Client and Server components.

Both the Client and Server components contain a port, represented using a square placed on the side or edge of both components. They are used to “help expose required and provided interfaces of a component” (Visual Paradigm, 2024).

Lines are used to connect components to interfaces that then connect to another component. Because these types of lines do not have arrows on either end of them, this means that they demonstrate an association between connected interfaces/components. By understanding what components are associated with other components, and knowing whether or not a component is required of or provided to another component, we gain a better understanding of the system and the relationship between components.

Beginning with the Client component, there is a Traveler Portfolio component that is required by the Web Browser component, which is the user interface. Both of these components are required by the Client Session component. The Graphics Library component requires the Traveler Portfolio component, and the Traveler Portfolio component requires the Database component.

The Client Component requires the Server component, however it is not the case that the Server component requires the Client component. A user is able to be directed to the Authentication Server component, where if authenticated, a Server Session is established.

The Server Session Component allows the authenticated user to manage the Traveler Database and update the Database component. The Server Session component requires the Traveler Database and Mongoose ODM Component, and the Mongoose ODM component requires the MongoDB component. The Mongoose ODM (Object Data Modeling) is a library “for MongoDB and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB” (Hossen, 2022).

### 

### Sequence Diagram

<Illustrate the flow of logic in a web application by completing a sequence diagram. Insert an image of the sequence diagram here.>

<Describe the flow of logic in the web application based on the sequence diagram. Be sure to describe the interactions between the layers, or tiers, of the full stack application. It will be helpful to include significant processes such as Sign In, Trips, and Admin interactions when referring to the sequence diagram.>

## Class Diagram

<Illustrate the JavaScript classes of the web application by completing a class diagram for the web application. Insert an image of the class diagram here.>

<Describe the JavaScript classes of the web application based on the class diagram.>

## [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> |

## 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.>

<Summarize the Angular project structure and how it compares to the Express project structure. Be sure to describe the rich functionality provided by the SPA compared to a simple web application interaction. Describe the process of testing to make sure the SPA is working with the API to GET and PUT data in the database.>

## References

Eddie, J. (2022, January 7). *Why MEAN Stack is Best for Web*

*Development?* Medium. <https://enlear.academy/why-mean-stack-is-best-for-web-development-55b7e4a00d72>

Harber, C., & Holmes, S. (2019). *Getting MEAN with Mongo, Express, Angular, and Node, Second*

*Edition* (2nd ed.). Manning Publications.

<https://learning.oreilly.com/library/view/getting-mean-with/9781617294754/>

Hossen, I. (2022, March 4). *Mongoose*. DEV Community.

<https://dev.to/iftakher_hossen/introduction-to-mongoose-409g>

Ramos, M. (2023, September 11). *MEAN Stack Explained*. Kinsta.

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

Visual Paradigm. (2024). *What is Component Diagram?* [https://www.visual-](https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/)

[paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/](https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/)