Travlr Getaways

CS 465 Project Software Design Document

Version 1.0

Table of Contents

Executive Summary

Travlr Getaways will need to meet certain requirements based on the client's needs. The client needs a software that has a robust, flexible, and dynamic design; all of which can be made possible with the implementation of the MEAN stack development. By using the MEAN stack development, Travlr Getaways will be dynamic and interactive on the customer-facing side of the application as well as dynamic and efficiently manageable for the administrative side of the application.

The architecture of the full stack application will have a representational state transfer API (RESTful API) that feeds a single-page application (SPA). The API is built with MongoDB, Node.js and Express, while the SPA is built with Angular. Both the customer and administrator interfaces interact with the back end via RESTful APIs, but the administrator interface will be separate from the customer interface to manage system functions. The RESTful API setup makes for a fast and responsive API that efficiently creates a collection of structured URLs containing usable data for applications.

The three components that make up the Travlr Getaways application include MongoDB (Database), Node.js and Express (Application Server) and Angular (Front End). MongoDB fits into the stack well because of how fast and scalable it is, as well as the number of useful features that it offers for building web applications. If used correctly, Node.js and Express work well for full stack development by working together to make setting up a web server extremely fast and the ability for the tools to make the most out of system resources. Lastly, Angular is appropriate and efficient because it allows for some of the data processing and logic to be handled by the browser by helping Node.js and Express; consequently, lessening the workload of the web server even more.

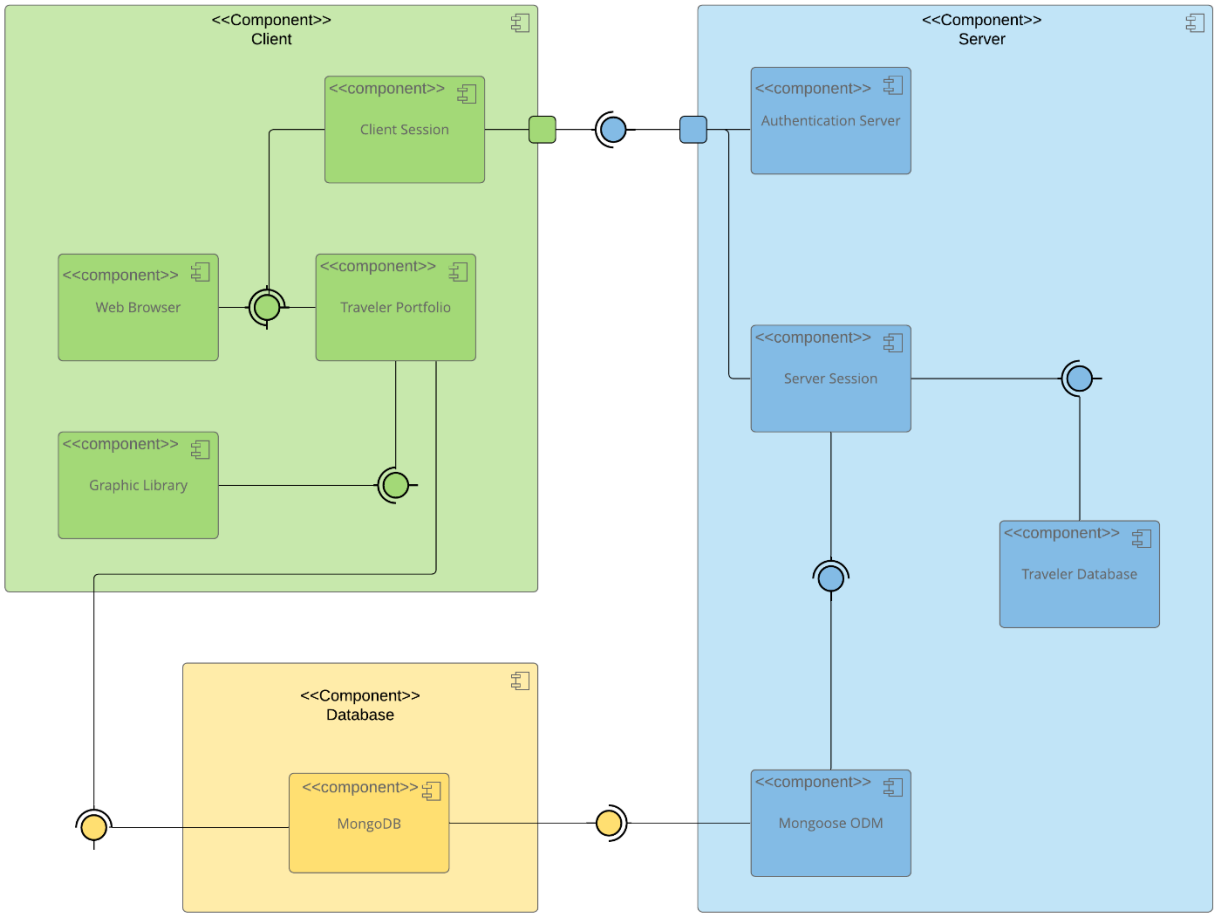
Design Constraints

Travlr Getaways will be a considerably simple website that works well with the MEAN stack development. Although design constraints exist with MEAN stack development, the type of build that Travlr Getaways has and the potential of what it could be, do not bring up many of these possible design constraints in the future.

Although, the reliance on JavaScript can be and issue when it comes to browser support and search engines. JavaScript applications can be hard for search engines to crawl and index. The reason being that search engines index through JavaScript-created content less efficiently than content that is delivered by the server. There are methods that can help imitate a crawlable site, but the methods can be tedious by requiring lots of extra energy and effort, require extra maintenance, increase risk of site penalization, and can lower search engine optimization.

System Architecture View

Component Diagram



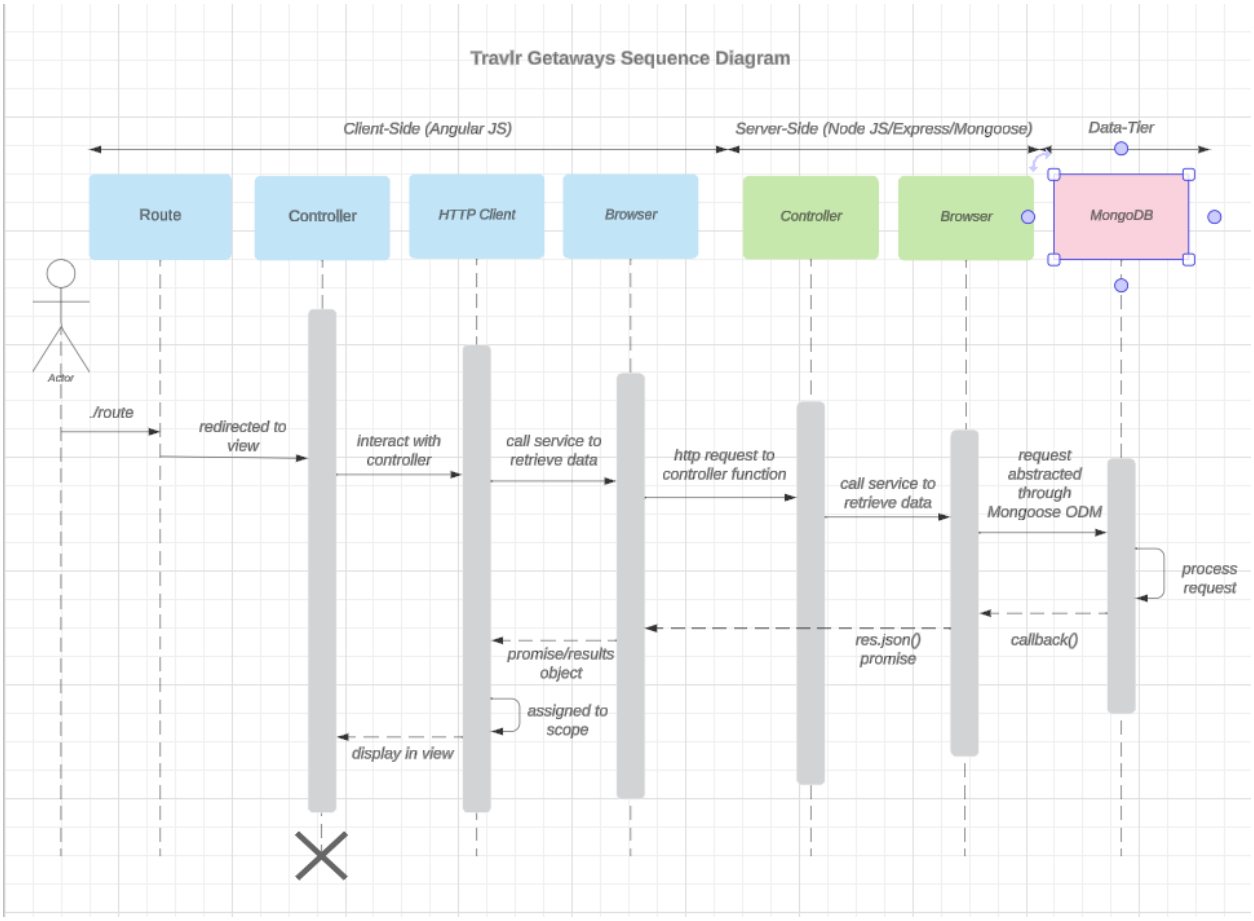
A text version of the component diagram is available: [CS 465 Full Stack Component Diagram Text Version].

The components above describe the overall system architecture of the web application. All the components can be categorized as either the client components, server components, or the database components. All the components work together cohesively to create a functioning web application. The components withing the client components are the graphic library, traveler portfolio, web browser, and client session. The traveler portfolio is connected to the MongoDB database in the database component, where all the traveler's information is stored. When the data is collected from the database it is rendered by the graphic library and the web browser. The client session component represents the push and pull of data between what is being stored and extracted from the database and what is being rendered per traveler's request.

The server component consists of components authentication server, server session, traveler database and Mongoose ODM. The authentication server component handles the data that is going to be sent to the client component by authentication of privacy and security rights. The server component is the back end of the web application and is where the hard data that is stored per traveler and the data the is being accessed per traveler session will be passed to the client component. All the data in the server component is received through the Mongoose ODM, which gets the data from MongoDB.

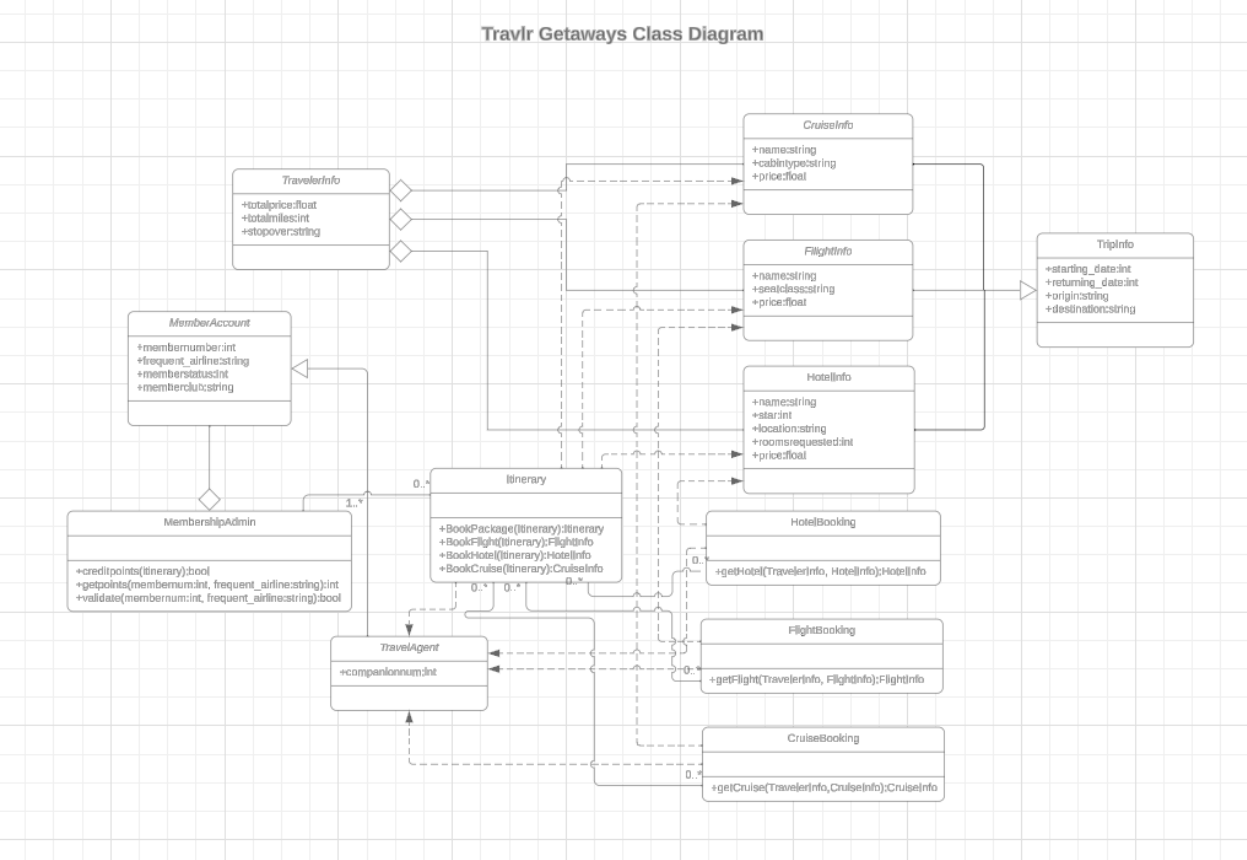
Lastly, the database component consists of the simple MongoDB component, which stores all the data.

Sequence Diagram



The user begins by accessing the specific webpage via a web browser. Upon loading the webpage, the browser fetches a client-side view/template, presenting the user with the webpage's content. When the user interacts with a clickable element with an anchor tag, a controller responsible for page navigation initiates an HTTP request via an HTTP client. The client serves as an intermediary between the client-side and server-side components. The HTTP client forwards the request for information to the appropriate API controller for the requested page. The relevant page data is processed and retrieved from a MongoDB database. The retrieved data is then sent back to the user's browser and displayed for viewing.

Class Diagram



JavaScript classes of the web application based on the class diagram:

-Membership_Admin is an aggregation of MemberAccount.
-MemberAccount is inherited from TravelAgent and aggregated by Membership_Admin.
-Travel_Agent inherits MemberAcount and is used by Itinerary, HotelBooking, FlightBooking and CruiseBooking.
-Itinerary has associations with Travel_Agent, Membership_Admin, CruiseInfo, FlightInfo, HotelInfo, HotelBooking, FlightBooking, and CruiseBooking.
-CruiseBooking is associated with Travel_Agent and CruiseInfo.
-FlightBooking is associated with Travel_Agent and FlightInfo.
-HotelBooking is associated with Travel_Agent and HotelInfo.
-CruiseInfo is aggregated by TravelerInfo, it inherits TripInfo and is associated with Itinerary and CruiseBooking.
- FlightInfo is aggregated by TravelerInfo, it inherits TripInfo and is associated with Itinerary and FlightBooking.
-HotelInfo is aggregated by TravelerInfo, it inherits TripInfo and is associated with Itinerary and HotelBooking.
-TripInfo is inherited from CruiseInfo, FlightInfo and HotelInfo.
-TravelerInfo aggregates CruiseInfo, FlightInfo and HotelInfo.