Travlr Getaways

CS 465 Project Software Design Document

Version 1.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.1	6/12/2022	Ayden Hooke	Milestone One

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

This application is built using MEAN stack, which is a combination of express.js, angular,js, node.js, and MongoDB. The framework belongs to node.js, while the client-side part belongs to angular.js and the server side belongs to node.js. MongoDB wraps everything together as the database. The customer-facing side of the application is the front-end of the code where the customer can interact with the server and database through an interface. The single page administrator application is quicker than client side applications and it is easier to maintain, but losing internet can lead to losing data, which would not be good.

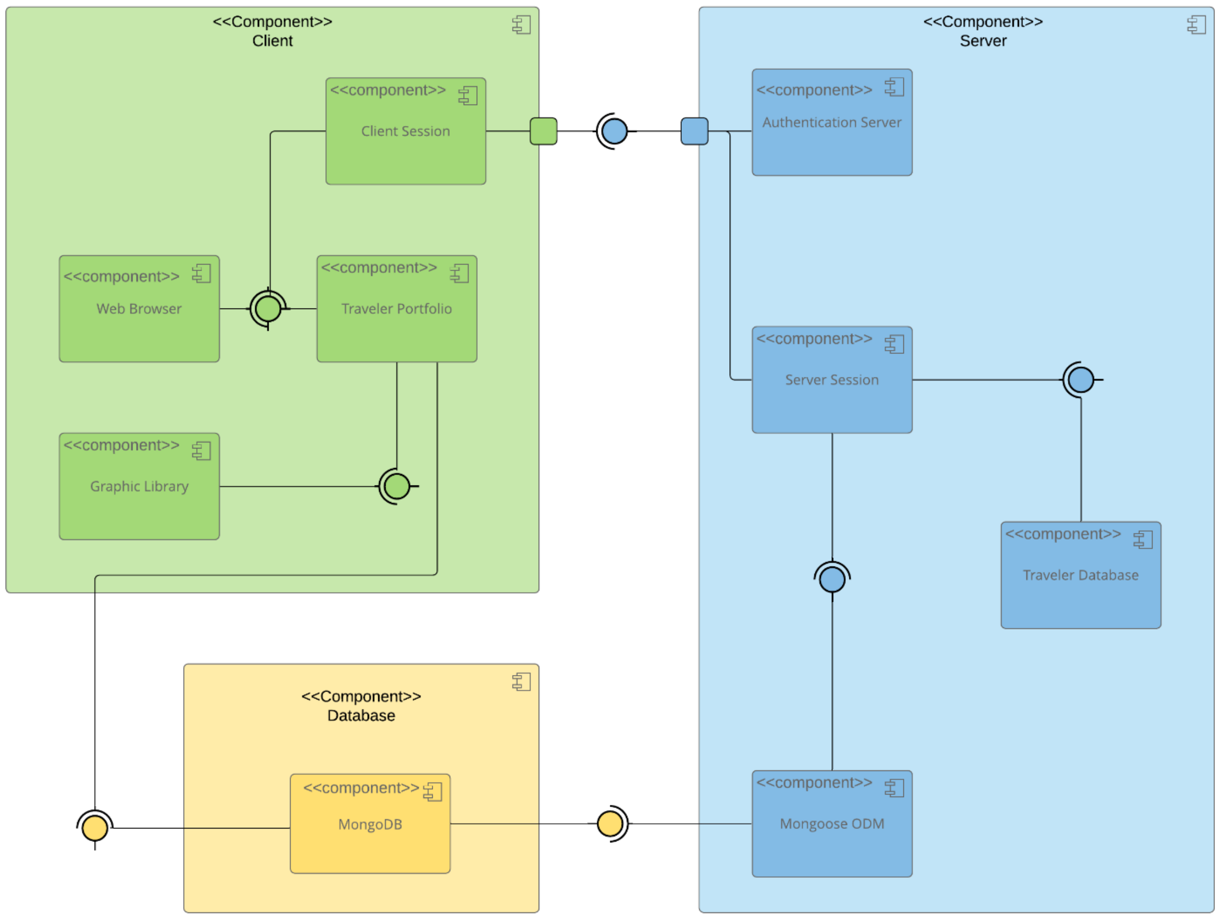
MongoDB transfers data between the client and the server, and it supports multi-dimensional data types, allowing for higher performance, more availability, and a much easier time scaling. Express.js is the framework of everything and plays an important role in the back-end. It tends to handle monotonous tasks, and it helps all sorts of web applications. Angular.js is another framework that is used in this project. It builds applications and is the front-end of the MEAN stack. It removes code so that other components can work with one another more efficiently. It is good for a dynamic application. The last of the components is Node.js, which runs the server. It uses an event driven and an on-blocking I/O model.

Design Constraints

The design constraints of any project essentially amount to three interorrelating things: budget, manpower, and deadlines -- all from which this project can suffer from. Budget is the main thing that holds any project up. Being able to stay under budget is something that would cause issues with this application development. If the project needed to be done quickly with a short deadline, then you would run into the issue of not having enough manpower to finish it in time, which would require a higher budget. Otherwise, you would have to push out the deadline in order to use less manpower in order to meet the budget. These three things go hand-in-hand with one another and form the main design constraints of this project.

System Architecture View

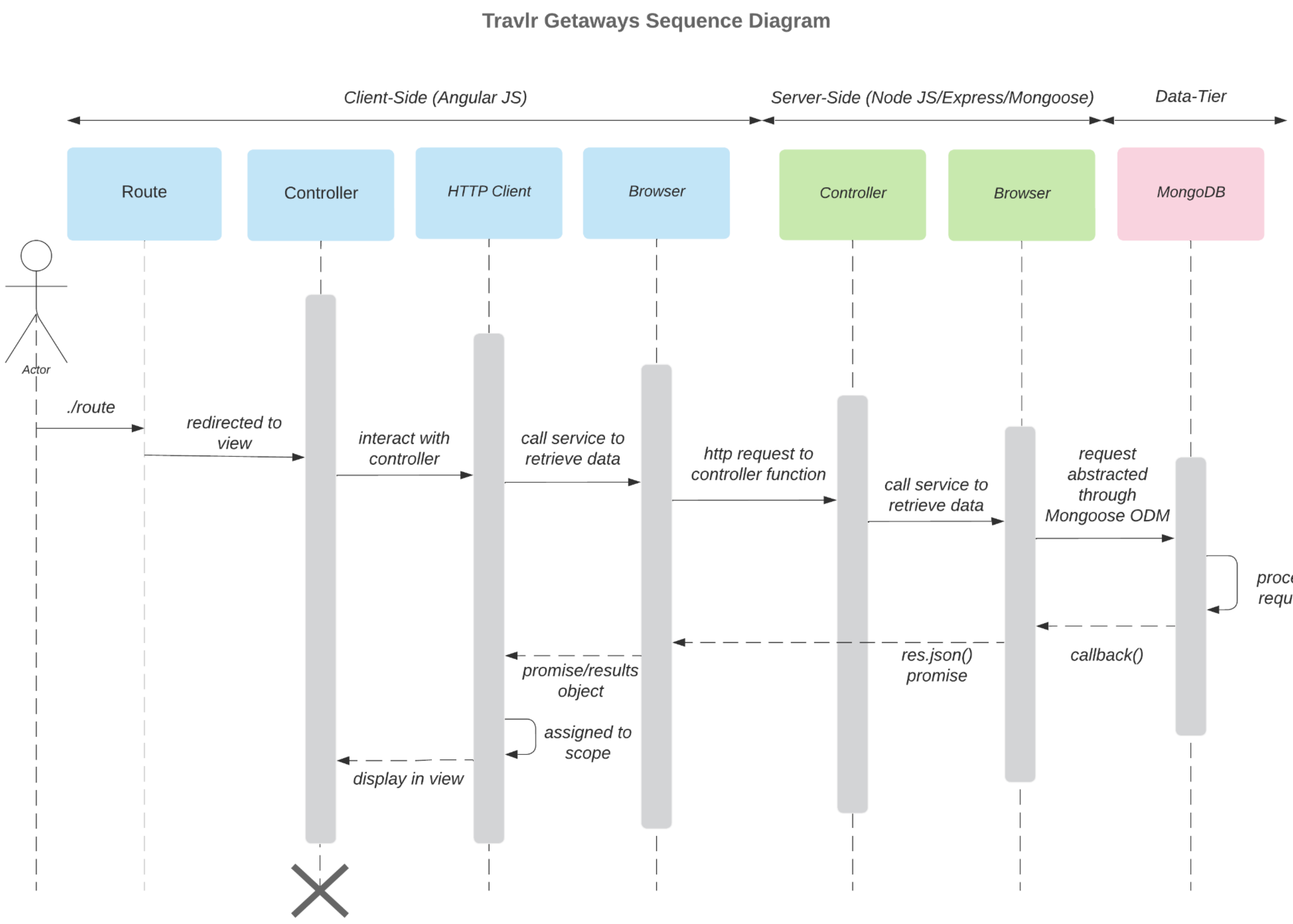
Component Diagram



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

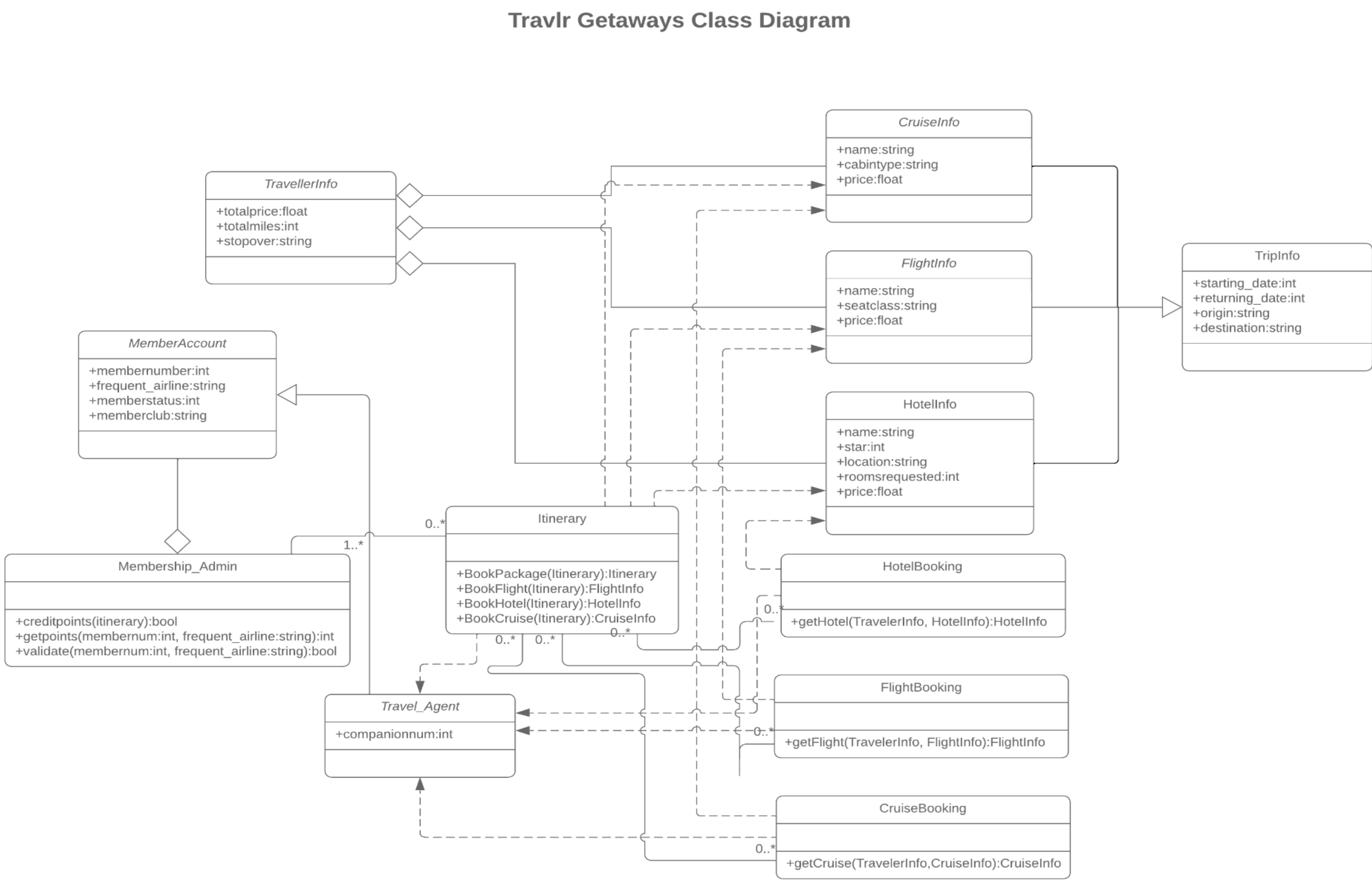
The significant components in the component diagram above are the Client, the Server, and the Database. The client and the server are connected together through the use of ports. The server gets information from the database. The information can then be passed through to the client. The server is also responsible for authenticating the client. The database is responsible for being able to store information and being able to supply it to whatever component requests it.

Sequence Diagram



This process begins with the actor's computer. The route that is entered by them redirects them to the view. When he interacts with the website, the controller interacts with the client, which calls to the server for the appropriate response to his action. The controller on the server side passes the call to MongoDB, which processes the request of the user. It then runs callback() and delivers the data to the HTTP client. At that point, it is assigned to scope and it is displayed in the view. Here, the actor can see it.

Class Diagram



This class diagram shows the relationship between different classes. For example, each member account has the possibility of being an admin, which would grant roles that supersede that of a normal account. Every normal account has a travel agent assigned to them, who can interact with the itinerary, the hotel, flight, and cruise bookings, as well as the info on all of the bookings. There is a class called TravellerInfo that stores the total data of CruiseInfo, FlightInfo, and HotelInfo. The FlightInfo gives information to TripInfo, which holds the traveler’s starting date, returning date, and where they are coming from/going to.

API Endpoints

Method	**Purpose**	**URL**	**Notes**
GET	Retrieves a list of things	</api/things>	Returns all active things
GET	Retrieves a single thing	</api/things/:thingId>	Returns single thing instance, identified by the thing ID passed on the request URL
POST	Creates a list of things	</api/things>	Creates a new list of things
POST	Creates a single thing	</api/things/:thingId>	Creates a single thing by the ID passed on the request URL
PUT	Updates a list of things	</api/things>	Updates and replaces a list of things, overriding everything and updating the records
PUT	Updates a single thing	</api/things/:thingId>	Updates and replaces a single thing by the ID passed on the request URL, overriding everything and updating the records
PATCH	Modifies a list of things	</api/things>	Updates and replaces a list of things, updating the records with only the values passed to it
PATCH	Modifies a single thing	</api/things/:thingId>	Updates and replaces a single thing by the ID passed on the request URL, updating the records with only the values passed to it
DELETE	Deletes a list of things	</api/things>	Deletes a list of things
DELETE	Deletes a single thing	</api/things/:thingId>	Deletes a single thing 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.>
// Given that I was not able to overcome the errors of my code that arose at the end, I cannot complete this section. However, the add/edit/update functions of the code should be functional, even though the server cannot launch for it to be confirmed in testing.

Angular's project structure is different from Express's HTML customer-facing page. Angular uses HTML to make the app's interface, while Express procedurally defines it in JavaScript. HTML is more easy to manipulate and change than JavaScript. With Express, there is much more involvement with the flow of the program and how things should be loaded, while with Angular, the specifics are handled automatically.

Some of the advantages of SPA functionality is loading times, the user experience, and the caching capabilities, but there are security issues and it needs more resources from the browser. It also only has a single URL, which causes loads of issues for sharing anything. SPA is also more difficult to upgrade as well.

Some additional functionality of SPA is that it loads everything in a single page, so it is faster. It is also simple and effective since it remains on the one page. Everything is in the browser, so it is faster because of that too, and it can be used separately from a server in certain use cases for periods of time.

To test that SPA is working, use the login information, check that it is responsive, notice the details of the components, and check that the API works by verifying information. Some errors that can occur are login errors, network errors, request errors, and fatal errors.