



Travlr Getaways Web Application
CS 465 Project Software Design Document
Version 3.0

Table of Contents

CS 465 Project Software Design Document	1
Table of Contents	2
Document Revision History	2
Executive Summary	3
Design Constraints	3
System Architecture View	4
Component Diagram	4
Sequence Diagram	5
Class Diagram	6
API Endpoints	7
The User Interface	8

Document Revision History

Version	Date	Author	Comments
3.0	04/16/22	Luke Raghoo	Added User Interface segment and finished software design document

Executive Summary

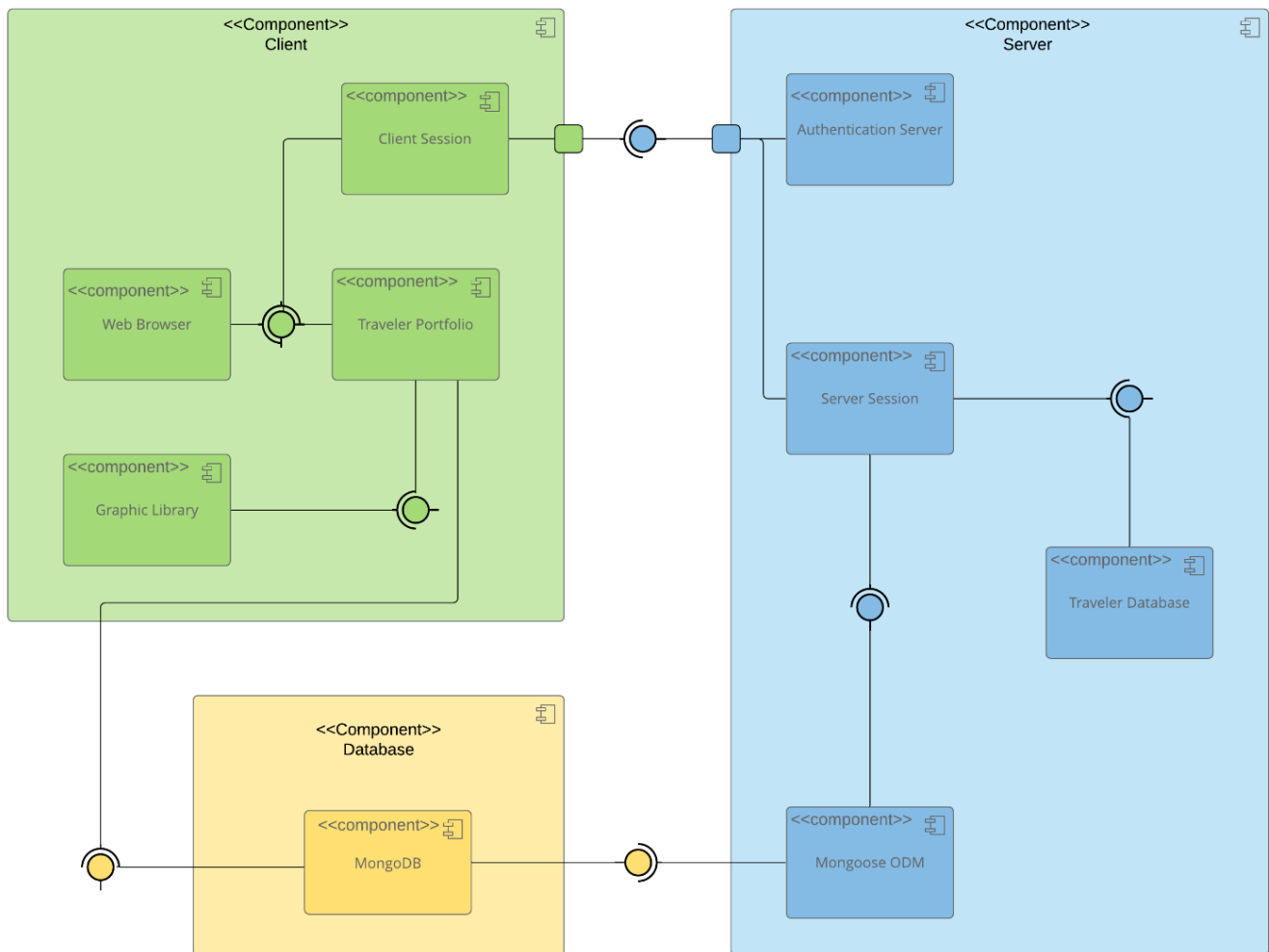
This web application utilizes something called the MEAN stack. Each letter in the acronym represents a software that is used: MongoDB, Express, Angular, and Node.js. All the software work together and scale up to support as many connections as possible in a short time frame. Travlr Getaways is trying to create a travel booking site for their customers to book travel packages. They need to be able to create an account, search for travel packages by location and price, and book reservations. Customers are supposed to be able to visit the website on a regular basis before their trip to view itineraries. This would be the customer-facing side of the application. The administrator single-page application is where Travlr Getaways administrators are able to maintain the customer base, available trip packages, and pricing for each item and package.

Design Constraints

One very common design constraint that these types of applications usually have would be the budget. Companies have a certain amount of money that they are willing to spend on a project. The budget is something that will influence the result of the project. Building the website is also done through the Node.js server and the Express framework. Depending on what the client wants to achieve this may or may not be the best way to go about doing it.

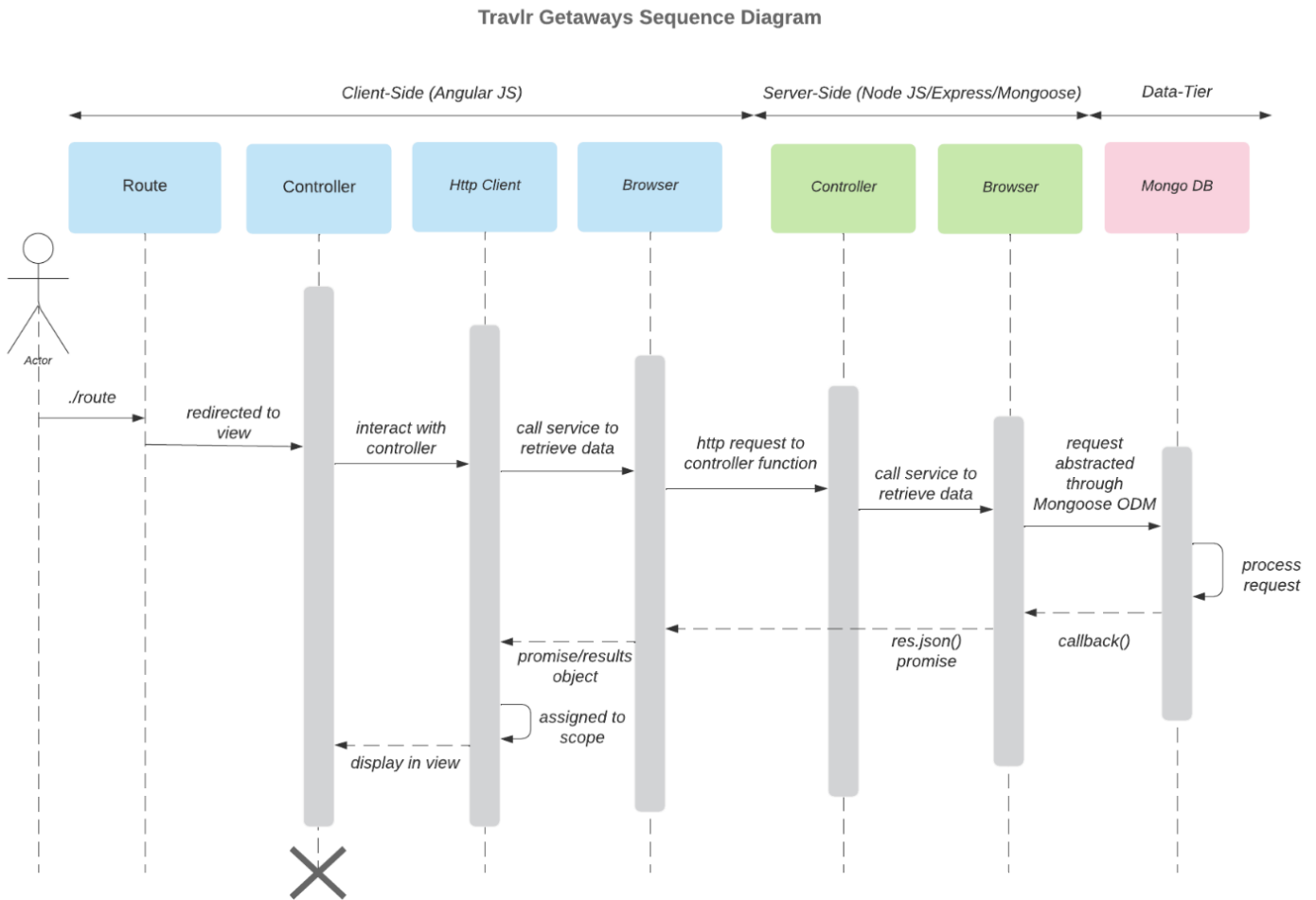
System Architecture View

Component Diagram



In the diagram above there are three component boxes. The first component box is the client component box in the top left corner. Within that box there are four components: the web browser, client session, traveler portfolio, and graphic library. The traveler portfolio is something that is stored in the database component. The database component (located in the bottom left) is also connected to Mongoose ODM in the server component box. Mongoose ODM is an object data modeling library for MongoDB and Node.js. It manages relationships between data and the representation of objects in MongoDB. The server component box is on the right side which includes three other components besides Mongoose ODM. It contains authentication server, server session, and traveler database.

Sequence Diagram

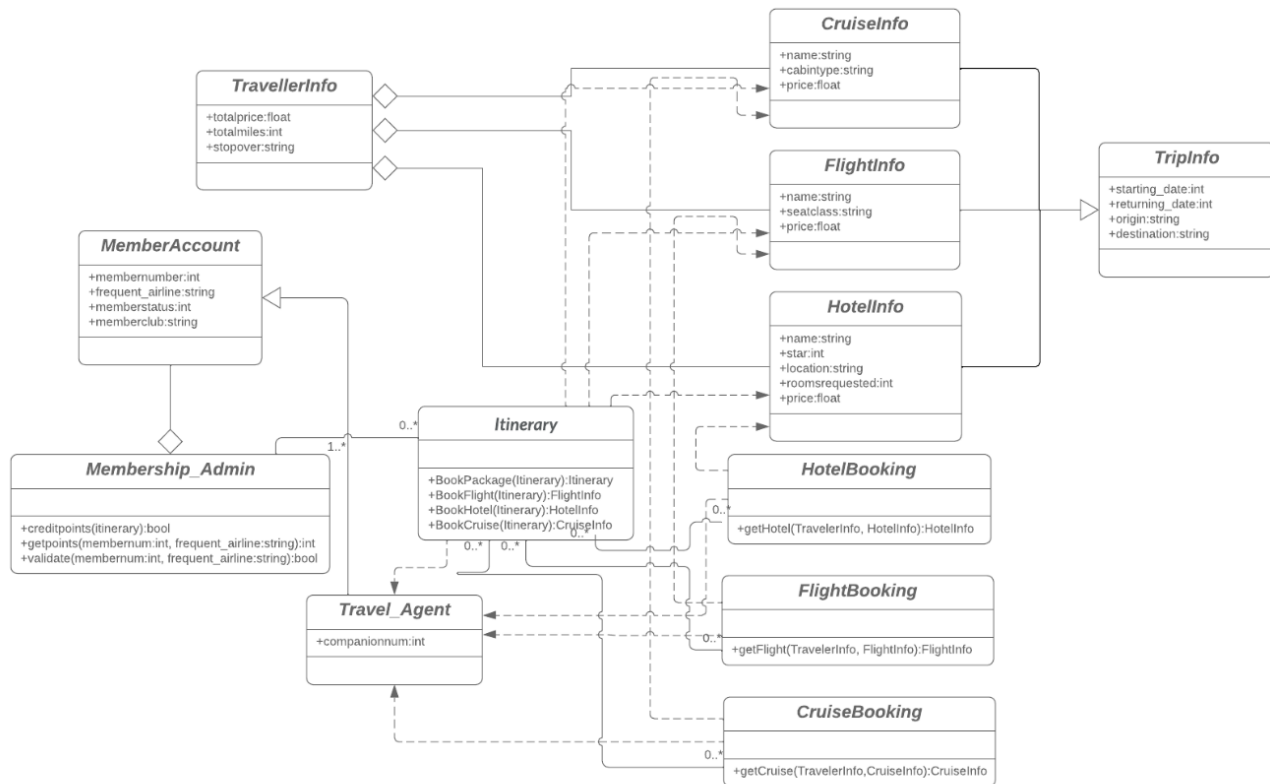


The first thing that appears on the sequence diagram is the Route. Routing refers to splitting an application's UI based on rules derived from the browser URL. For example, if you click a link and it changes from <https://test.com> to <https://test.com/contact> that is routing. Following routing is the controller. A controller is the part of the UI which coordinates multiple Views on the screen. It receives user input and sends the appropriate messages to its underlying Views. After interacting with the controller we arrive at the HTTP client. HTTP stands for Hypertext Transfer Protocol, and it is used for transmitting hypermedia documents. This client can send a request to and get a response from the server in HTTP format. The final part in the client side is the Browser which is the frontend.

Everything on the backend is the server-side. The first thing on the Server-Side is also a controller but it plays a different role compared to the one on the client side. The server-side controller is essentially an apex class which contains some specific data that you need to take care of. After the Server-side we can see the Data-Tier which contains Mongo DB. MongoDB is a database that is often used for web applications that require unstructured data to be stored.

Class Diagram

Travlr Getaways Class Diagram



There are a total of twelve JavaScript classes in this diagram. The first one that I would like to talk about is the Itinerary class. This is connected to every class except MemberAccount, TravellerInfo, and TripInfo. An Itinerary is essentially a travel guide so it contains information on the hotel, cruise, and flight. One of the most important classes is the MemberAccount class. This contains the members ID number, their airline, membership status, and member club. It is connected to the Membership_Admin class which is where the member can gain credit points. HotelBooking, FlightBooking, and CruiseBooking are all tied into HotelInfo, FlightInfo, and CruiseInfo. These classes contain information such as the price, name, location, etc. The HotelInfo, FlightInfo, and CruiseInfo classes are all connected to two other classes: TripInfo, and TravellerInfo. The TripInfo class contains the departure date, return date, destination etc. TravellerInfo contains the total price for the hotel, flight, and cruise as well as the total miles travelled.

API Endpoints

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
PUT	Update list of things	/api/things	Updates and replaces entire list of things. Previous data will be overwritten.
PUT	Update single thing	/api/things/:thingId	Updates and replaces a single thing. Previous data for the entry will be overwritten.
DELETE	Delete list of things	/api/things	Deletes the entire list of things
DELETE	Delete single thing	/api/things/:thingId	Deletes a single thing.
PATCH	Modifies list of things	/api/things	Unlike the PUT method this is only going for modifying resources through partial data. The entire list of things will be modified.
PATCH	Modify single thing	/api/things/:thingId	Unlike the PUT method this is only for modifying. Only a single thing will be modified.

The User Interface

Unique Trip

Consequat ipsum egestas nunc vel vestibulum orci gravida. Vestibulum sit amet porttitor odio. Nulla facilisi. Fusce at pretium felis. Sed consequat libero ut turpis venenatis ut aliquam risus semper. Etiam convallis mi vel risus pretium sodales. Etiam nunc lorem ullamcorper vitae laoreet.

Edit


perit et egestas nunc vel vestibulum orci gravida. Vestibulum sit amet porttitor odio. Nulla facilisi. Fusce at pretium felis. Sed consequat libero ut turpis venenatis ut aliquam risus semper. Etiam convallis mi vel risus pretium sodales. Etiam nunc lorem ullamcorper vitae laoreet.

Edit

quis semper arcu. Cras orci neque, euismod et accumsan ac, sagittis molestie lorem. Proin odio sapien, elementum at tempor non. Vulputate eget libero. In hac habitasse platea dictumst. Integer purus justo, egestas eu consectetur eu, cursus in tortor. Quisque nec nunc ac mi ultrices iaculis.

Edit

test



test

Edit

Edit Screen

[Traveler Getaways Admin](#)

Edit Trip

Code:

Name:

Length:

Start:

Resort:

PerPerson:

Image:

Description:

Save

Update Screen

Traveler Getaways Admin

Edit Trip

Code:

Name:

Length:

Start:

Resort:

PerPerson:

Image:

Description:

```
GET /api/trips/test 304 1.979 ms - -
OPTIONS /api/trips/test 200 0.228 ms - 12
{
  _id: '625266e1acf498144430ab6c',
  code: 'test',
  name: 'Test Village',
  length: '4 nights / 5 days',
  start: '2022-10-03T00:00:00.000Z',
  resort: 'Test Resort, 5 stars',
  perPerson: '$50.00',
  image: 'suite.jpg',
  description: '<p>This trip was added as a test by Luke Raghoo!</p>'
}
PUT /api/trips/test 200 3.493 ms - 284
GET /api/trips/ 200 3.122 ms - 2139
GET /api/trips/ 304 2.270 ms - -
```

```
pretium sodales. Etiam nunc lorem
ullamcorper vitae laoreet.
```

Edit


```
eu consectetur. In lorem dui,
elementum sit amet convallis ac,
tincidunt vel sapien.
```

Edit

```
purus justo, egestas eu
consectetur eu, cursus in tortor.
Quisque nec nunc ac mi ultrices
iaculis.
```

Edit


Test Village



Test Resort, 5 stars
4 nights / 5 days only per person
This trip was added as a test by
Luke Raghoo!

Edit

Testing Place



Testing Resort, 5 stars
3 nights / 4 days only \$100.00 per
person
This is a test by Luke Raghoo

Edit

The Angular project structure is different from that of the Express HTML project structure. One of the main differences I found was one was used more for backend purposes while the other was used more for frontend purposes. I had to use an admin folder alongside my usual travlr folder to get the angular project working. Some advantages of SPA include the speed load. Single-page applications normally load new information faster than regular sites. However, there still are a few disadvantages as well. Search engine optimization is the biggest known disadvantage. This is because SEO is dependent on HTML content. So, if there aren't that many unique URLs, keywords, etc. then the site will rank lower.

I did run into a few errors while working on this assignment. The first major error that I ran into was that nothing outside of the buttons was displaying. This happened due to me not connecting to the front end and back end properly. As I mentioned earlier, I had to use an admin folder for my front end and the travlr folder was used for the back end. Another issue that I ran into was the buttons would not add or update any content beyond what was displayed. This issue was happening because again I was not connected properly. Initially, I kept trying to run "ng serve" on the back end but I was supposed to do "npm start" instead. Once I did that I was connected to the database, and I ran "ng serve" from the front end. Doing this allowed everything to work fine.