Travlr Getaways

The Travel Guide

CS 465 Project Software Design Document

Version 1.2

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
Class Diagram........9
API Endpoints........11
The User Interface........12

Document Revision History

Version	Date	Author	Comments
1.0	01/21/2024	Joshua Wozny	Draft Project Outline
1.1	02/11/2024	Joshua Wozny	Added Sequence and Class Diagrams, and API Endpoints table
1.2	03/03/2024	Joshua Wozny	Final draft

Executive Summary

Project Overview: Travlr Getaways aims to develop a sophisticated travel booking site. This platform will allow customers to create accounts, search, and book travel packages, and manage their itineraries. Additionally, an admin-only segment will enable effective management of customers, trip packages, and pricing.

Architecture and Technology: Leveraging the MEAN stack (MongoDB, Express.js, Angular, and Node.js), the application will follow a Model-View-Controller (MVC) architecture. This robust stack ensures a seamless, full-stack JavaScript environment, conducive to agile development and effective data handling.

Customer-Facing Website: The customer interface, built using Angular, will offer a dynamic and responsive user experience. Key features include account creation, package searching by location and price, and itinerary management. The Express framework will manage routing, enhancing the application's scalability and maintainability.

Admin SPA (Single-Page Application): An admin-only SPA, also built using Angular, will provide a seamless and efficient interface for managing customer data, travel packages, and pricing. This SPA will interact with a RESTful API, ensuring real-time data updates and maintenance.

Data Management and Security: The NoSQL database, MongoDB, will store application data, managed via Mongoose for schema definition and data validation. Security measures will include secure endpoints and authenticated access, safeguarding both customer and administrative interactions.

Dynamic Content Rendering: The application will transition from static HTML to dynamic content using Handlebars, a templating engine, to render JSON data. This approach ensures a more interactive and personalized user experience.

Single-Page Application (SPA) Features: The SPA for administrators will provide comprehensive management capabilities. Features will include user-friendly interfaces for updating trip details, managing bookings, and adjusting pricing, all interconnected through a RESTful API.

Security Enhancements: Security is paramount; hence, the application will implement secure login forms and best practices in endpoint security to prevent unauthorized access.

Conclusion: This project proposes a robust, user-friendly, and secure web application for Travlr Getaways, employing the MEAN stack to meet the diverse needs of customers and administrators. The design focuses on dynamic content rendering, efficient data management, and stringent security measures, ensuring a high-quality user experience and streamlined administrative processes.

Design Constraints

Technology Stack: The exclusive use of the MEAN stack (MongoDB, Express.js, Angular, Node.js) poses a constraint, limiting the application to technologies compatible with JavaScript and JSON. This affects third-party integrations and requires developers to be proficient in these technologies.

Dynamic Content Rendering: Implementing dynamic content via Handlebars requires specific approaches to UI design and may restrict how data is displayed and updated in real-time.

Database: Using a NoSQL database like MongoDB dictates the data modeling approach, impacting how data relationships are managed and queried. This could limit complex relational data handling.

Security: The need for robust security controls, including secure login and endpoint protection, requires additional development effort and expertise, potentially impacting deployment timelines and ongoing maintenance.

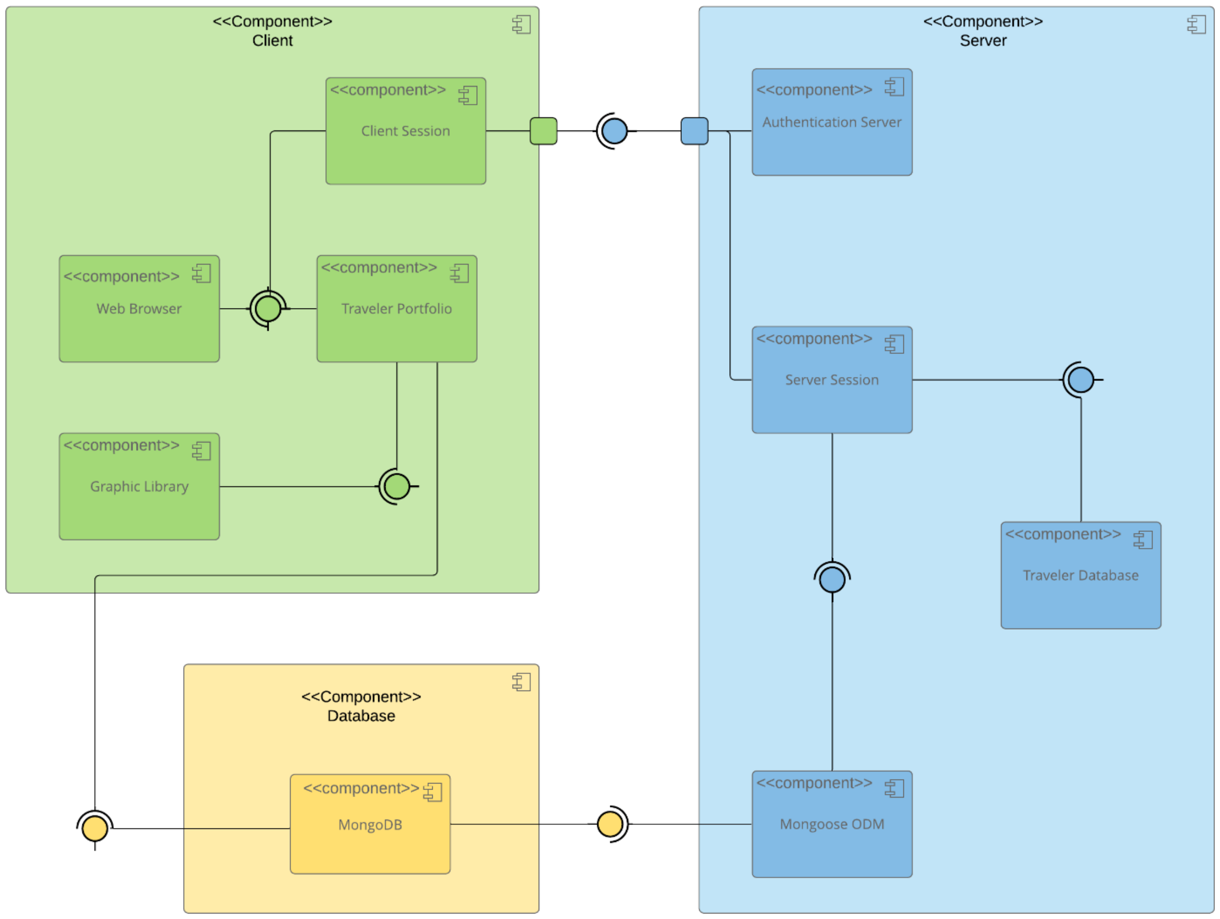
Single-Page Applications (SPA): Developing SPAs for both customer and admin interfaces demands a deep focus on frontend performance and state management, influencing the application's overall architecture and user experience.

RESTful API Integration: The application’s dependence on RESTful APIs for data interaction means that backend development needs to be aligned precisely with frontend requirements, affecting the development workflow and testing.

These constraints guide the application's design and development, influencing decisions related to technology choices, data management, user experience, and security. Each constraint must be carefully managed to ensure a balanced and effective solution that meets the project’s objectives.

System Architecture View

Component Diagram



<<Component>> Client
<<component>> Client Session
<<component>> Web Browser
<<component>> Traveler Portfolio
<<component>> Graphic Library
<<Component>> Server
<<component>> Authentication Server
<<component>> Server Session
<<component>> Traveler Database
<<component>> Mongoose ODM
<<Component>> Database
<<component>> MongoDB

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

The system architecture of the Travlr Getaways web application, as depicted in the component diagram, consists of three primary components: Client, Server, and Database.

Client Component:
Contains a Web Browser, Client Session, Traveler Portfolio, and Graphic Library.
The Web Browser interacts with both the Client Session and the Traveler Portfolio.
The Traveler Portfolio is linked to the Graphic Library and MongoDB in the Database component.

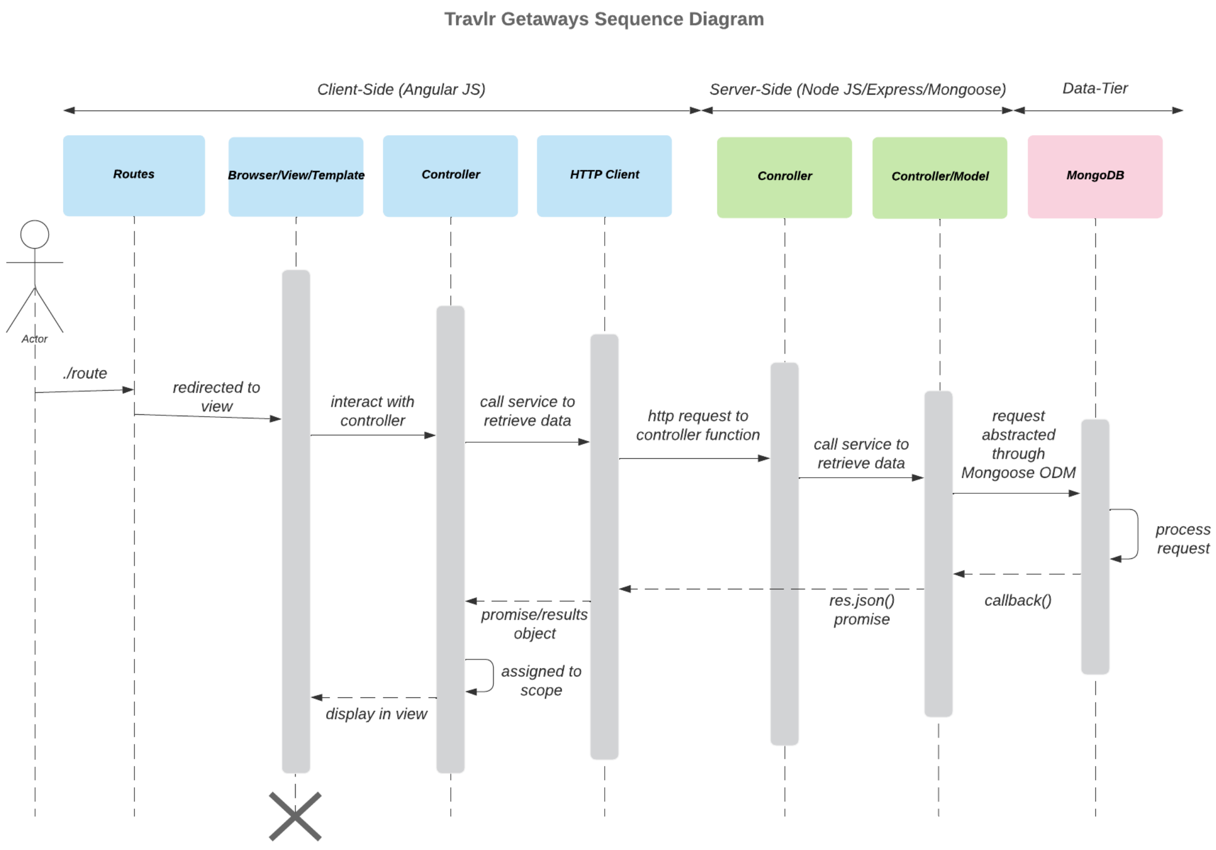
Server Component:
Comprises an Authentication Server, Server Session, Traveler Database, and Mongoose ODM.
The Mongoose ODM interfaces with MongoDB and is linked to the Server Session.
The Server Session connects with both the Traveler Database and the Client Session.

Database Component:

Includes MongoDB, which is interconnected with the Traveler Portfolio and Mongoose ODM.

The diagram indicates that the Traveler Portfolio is a pivotal component, interfacing with the client-side Web Browser and the server-side database, MongoDB. This setup suggests a design where user interactions in the browser are processed and reflected in real-time within the database, facilitated by the Mongoose ODM for object data modeling. The Client and Server Sessions indicate a secure, authenticated interaction between the user interface and the server. The Authentication Server's link with the Client Session underscores the importance of security in user authentication processes.

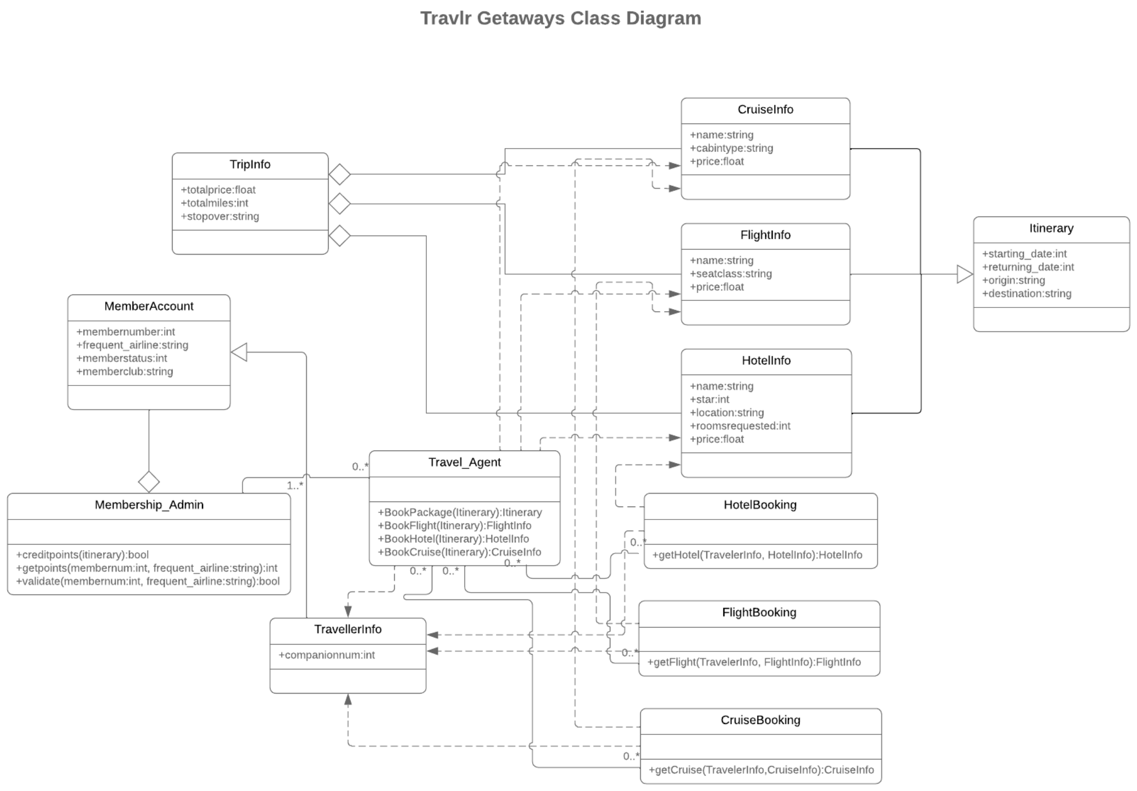
Sequence Diagram



- Route: The server receives the HTTP request and routes it to the appropriate controller action. Browser/View/Template: This represents the view displayed in the browser and where the user initiates an action, like clicking a button or loading a page.
- Controller: Once the request is routed correctly, the corresponding controller action takes over.
- HTTP Client: After the user initiates an action, the HTTP Client sends an HTTP request to the server.
- Controller: This controller interacts with the database.
- Controller/Model: The controller interacts with the model using Mongoose to process data from the database.
- MongoDB: The database is queried or updated based on the controller's request. This interaction involves reading from or writing to the database, depending on the operation required.
- Controller: After receiving the response from MongoDB, the controller processes the data, transforms it, and prepares a response to be sent back to the client.
- Route: The response is sent back through the route to the HTTP Client.
- HTTP Client: The HTTP Client receives the response data and uses it to update the View, typically through AngularJS's data binding and templating features.

- Browser/View/Template: Finally, the view in the browser is updated to reflect the new or changed data, completing the cycle of interaction.

Class Diagram



Classes: The diagram shows several classes, each representing a different component of the system.

- TripInfo: Contains attributes like totalprice, totalmiles, and stopover.
- MemberAccount: Has attributes such as membernumber, frequent_airline, memberstatus, and memberclub.
- Membership_Admin: This seems to be a specialized class related to administering memberships, with methods like creditpoints, getpoints, and validate.
- TravelerInfo: Includes an attribute companionnum.
- Itinerary: Holds travel details with attributes like starting_date, returning_date, origin, and destination.
- Travel_Agent: This class includes methods for booking various aspects of travel, such as BookPackage, BookFlight, BookHotel, and BookCruise.

Related Classes: The diagram also includes classes for specific travel components, each with attributes relevant to their context.

- CruiseInfo: With attributes name, cabin_type, and price.
- FlightInfo: Contains name, seating, class, and price.

- HotelInfo: Has name, location, star, and roomsrequested.

Association Classes: These are specialized classes that handle booking functionalities.

- HotelBooking: Has a method getHotel which connects TravelerInfo and HotelInfo.
- FlightBooking: Contains a method getFlight linking TravelerInfo and FlightInfo.
- CruiseBooking: Includes a method getCruise connecting TravelerInfo and CruiseInfo.

Relationships:

- MemberAccount to Membership_Admin: Each MemberAccount is associated with a single Membership_Admin, and vice versa.
- Travel_Agent to Itinerary: One Travel_Agent can handle multiple Itineraries.
- Travel_Agent to TravelerInfo: A single Travel_Agent can be associated with many TravelerInfo instances.
- Travel_Agent to {HotelBooking, FlightBooking, CruiseBooking}: Travel_Agent can make zero or many bookings of each type of Booking.
- TravelerInfo to {HotelBooking, FlightBooking, CruiseBooking}: Each booking class (HotelBooking, FlightBooking, CruiseBooking) has a method that takes TravelerInfo and their respective information class (HotelInfo, FlightInfo, CruiseInfo) as parameters. These are used to create a booking for a traveler.
- {CruiseInfo, FlightInfo, HotelInfo} to Itinerary: The Itinerary class depends on these classes.
- {CruiseInfo, FlightInfo, HotelInfo} to {CruiseBooking, FlightBooking, HotelBooking}: A booking cannot exist without the respective {CruiseInfo, FlightInfo, HotelInfo}

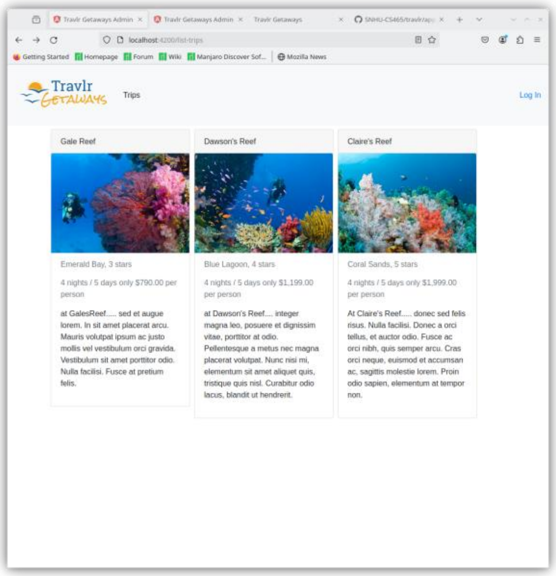
API Endpoints

Below are the Endpoints available via the Travlr Getaways web site.

Method	**Purpose**	**URL**	**Notes**
GET	Retrieve all trips	/api/trips	Returns a full list of trips currently available through the Travlrs website.
POST	Add a trip	/api/trips	Add a new trip and confirm by returning trip details.
GET	Retrieve a single trip	/api/trips/:tripCode	Returns a single trip with the code passed in {tripCode} which is a unique identifier for a trip on the Travlr website.
PUT	Update a single trip	/api/trips/:tripCode	Update trip by tripCode, confirm by returning trip detail.
DELETE	Delete a single trip	/api/trips/:tripCode	Delete trip by tripCode.
POST	Register new user	/api/register	Create a new user with provided name, email, and password.
POST	Login user	/api/login	Login a user with provided email and password. Security token is returned.

The User Interface

FIGURE 1



localhost:4200/list-trips

Gale Reef

Emerald Bay, 3 stars

4 nights / 5 days only $790.00 per person

at GalesReef..... sed et augue lorem. In sit amet placerat arcu. Mauris volutpat ipsum ac justo mollis vel vestibulum orci gravida. Vestibulum sit amet porttitor odio. Nulla facilisi. Fusce at pretium felis.

Dawson's Reef

Blue Lagoon, 4 stars

4 nights / 5 days only $1,199.00 per person

at Dawson's Reef.... integer magna leo, posuere et dignissim vitae, porttitor at odio. Pellentesque a metus nec magna placerat volutpat. Nunc nisi mi, elementum sit amet aliquet quis, tristique quis nisl. Curabitur odio lacus, blandit ut hendrerit.

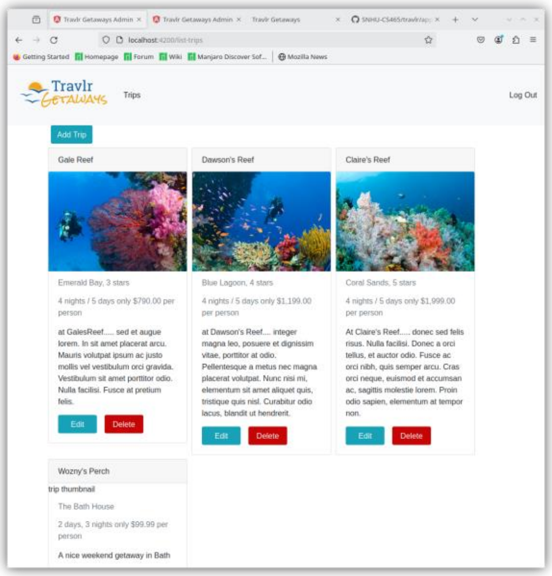
Claire's Reef

Coral Sands, 5 stars

4 nights / 5 days only $1,999.00 per person

At Claire's Reef..... donec sed felis risus. Nulla facilisi. Donec a orci tellus, et auctor odio. Fusce ac orci nibh, quis semper arcu. Cras orci neque, euismod et accumsan ac, sagittis molestie lorem. Proin odio sapien, elementum at tempor non.

FIGURE 2



localhost:4200/list-trips

Gale Reef

Emerald Bay, 3 stars

4 nights / 5 days only $790.00 per person

at GalesReef..... sed et augue lorem. In sit amet placerat arcu. Mauris volutpat ipsum ac justo mollis vel vestibulum orci gravida. Vestibulum sit amet porttitor odio. Nulla facilisi. Fusce at pretium felis.

Dawson's Reef

Blue Lagoon, 4 stars

4 nights / 5 days only $1,199.00 per person

at Dawson's Reef.... integer magna leo, posuere et dignissim vitae, porttitor at odio. Pellentesque a metus nec magna placerat volutpat. Nunc nisi mi, elementum sit amet aliquet quis, tristique quis nisl. Curabitur odio lacus, blandit ut hendrerit.

Claire's Reef

Coral Sands, 5 stars

4 nights / 5 days only $1,999.00 per person

At Claire's Reef..... donec sed felis risus. Nulla facilisi. Donec a orci tellus, et auctor odio. Fusce ac orci nibh, quis semper arcu. Cras orci neque, euismod et accumsan ac, sagittis molestie lorem. Proin odio sapien, elementum at tempor non.

Wozny's Perch

trip thumbnail

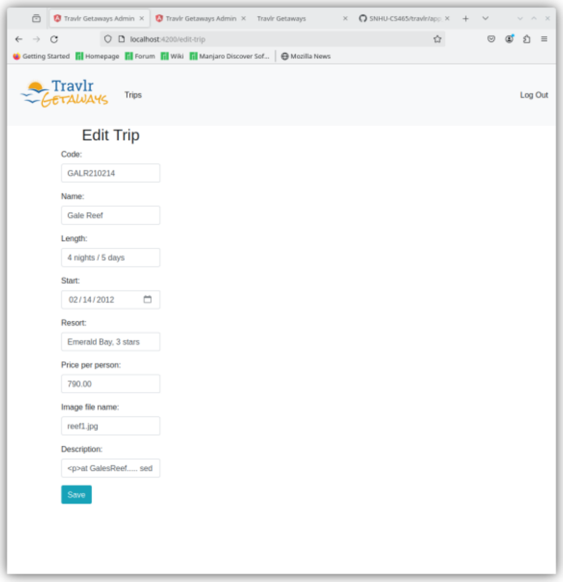
The Bath House

2 days, 3 nights only $99.99 per person

A nice weekend getaway in Bath

FIGURE 3

FIGURE 4



localhost:4200/edit-trip

Edit Trip

Code: GALR210214

Name: Gale Reef

Length: 4 nights / 5 days

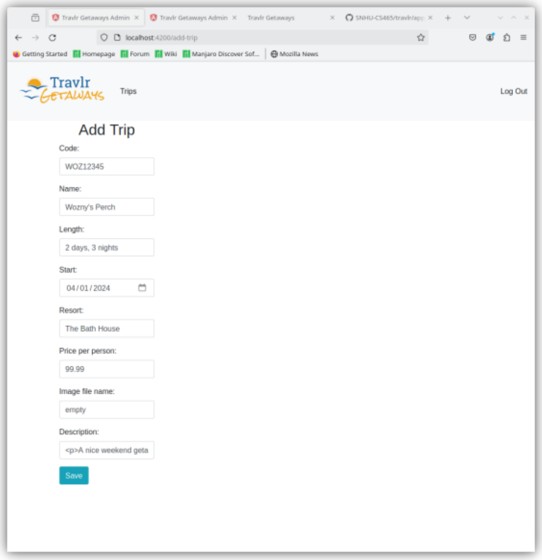
Start: 02 / 14 / 2012

Resort: Emerald Bay, 3 stars

Price per person: 790.00

Image file name: reef1.jpg

Description: <p>at GalesReef..... sed



localhost:4200/add-trip

Add Trip

Code: WOZ12345

Name: Wozny's Perch

Length: 2 days, 3 nights

Start: 04 / 01 / 2024

Resort: The Bath House

Price per person: 99.99

Image file name: empty

Description: <p>A nice weekend geta

FIGURE 1: Trip list before logging in. Note: buttons are not shown because only authorized users may add, edit, or delete trips.
FIGURE 2: Trip list after logging in. Note: buttons are now active since a user is logged in. Also note the added Trip in the second row
FIGURE 3: Edit trip form
FIGURE 4: Add trip form

Angular Project Structure

- We used a modular structure where components, services, modules, and other artifacts are organized into separate directories.
- The `src` directory is the main directory containing all the source code for the Angular application.
- Inside `src`, you will find directories like `app` (containing the main application components), `assets` (for static assets like images and fonts), and `environments` (for environment-specific configurations).
- Angular also encourages the use of modules to organize related components, and services. Which are also located in the app directory
- The `angular.json` file defines project configurations like build settings, assets, and environments.

Express Project Structure

- The Express project follows a more flexible structure, including directories for `routes` (to define API endpoints), 'database' amd `models` (for database models), `controllers` (to handle request/response logic), and `config` (for middleware configuration, like passport).
- The `package.json` file manages dependencies and scripts for running the application.

Comparing Angular and Express

- Angular is a front-end framework for building Single Page Applications (SPAs), while Express is a back-end framework for building server-side applications, including RESTful APIs.
- Angular provides rich client-side functionality with features like data binding, dependency injection, routing, and component-based architecture, leading to a more interactive and dynamic user experience.
- Express mainly focuses on handling HTTP requests, routing, middleware management, and interfacing with databases or other services. It serves as the backend API for Angular or other front-end frameworks.
- SPAs built with Angular offer seamless user interactions without page reloads, providing a smoother and more responsive user experience compared to traditional web applications.

Testing SPA with API Interaction

1. Integration Testing: Test the integration between Angular components and services with a mocked API layer. Verify that components interact correctly with the services and handle responses appropriately.
2. API Testing: Test the API separately using Postman endpoint testing. Verify that API endpoints respond correctly to GET and PUT requests, and database operations are performed as expected