

Dreamscape Destinations

A journey from your dreams to reality

Group Name: SE PROJECT TEAM 02

Group Members:

- Veeresh Sakali
- Susmith Meesa
- Harshath Budida
- Harshitha Thokala
- Gopi Krishna Kummari
- Pooja Sree Poka
- Sashidhar Chary Viswanathula
- Balaji Valeti

Deliverable 3 – Project Phase 1

CSCE 5430 (Spring 2024)

Implementation Phase 1:

In this phase our target was pretty much big to meet the overall project margin while maintaining a buffer zone and we got successful in this case as we have achieved pretty much all the requirements with some changes in the coding frame work which we will describe in its respective description. For now, we have created the basic user experience with the front on React JS. Let's dig into this in more details

Requirements:

1. Change of plan

Initially we thought of making the front end for phase 1, back end for the phase 2 and optimization for the phase 3, but after our requirements submission we got peer feedback and teacher feedback which lead us to alter our plan a little bit. So, now our target is to make front end along with the back end and database of functional requirements and that complete the remaining set of requirements in the phase 2 and phase 3 is still allocated for the web optimization, code optimization and SEO. The 2nd biggest change is requirement is the change of our database. In initial plan we planned to use the SQLite3 database but looking at our back end which is purely in the Nodes.JS, we find out that the best fit database with the Nodes is MongoDB. We have also conduct a thorough search on the database compatibility and found that, as Nodes.JS is a browser language and MongoDB is also a browser-based database so they best fit with each other (Vohra, D. (2015). Using MongoDB with Node.js. In: Pro MongoDB™ Development. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-1598-2_5). So, we have seted up a MongoDB account and created a cluster for our project.

2. Front end

● Home

The home is consisting of pretty much our branding, as this is page which will decide traffic to our website. So, we made this with very care and after a long back and forth discussions. The Navbar component, intelligently engineered, ensures seamless navigation by adapting its layout effortlessly, catering to various screen sizes without the need for backend integration. Meanwhile, the Hero section captivates users with its visually engaging video background, promoting user engagement and retention.

Further down the page, the Search component stands out as a beacon of convenience. Designed to provide effortless exploration, it enables users to discover nearby attractions intuitively, offering a glimpse into the platform's future potential once backend functionalities are integrated. Similarly, the Book section, although currently operating without dynamic data fetching capabilities, showcases a user-friendly interface primed for streamlined trip planning, laying the groundwork for future enhancements.

The Discover section adds an element of intrigue with its visually appealing hover effects, enticing users to explore curated destinations with ease. Finally, the About section rounds off the homepage by offering insights into the platform's vision and mission, establishing a connection with users beyond mere functionality. Together, these features represent the foundation upon which Dream Escape will continue to evolve, promising an enriching and immersive travel experience for all users.

- **Login Page**

Our login page is designed to seamlessly connect users to their accounts, unlocking a treasure trove of travel possibilities. With a user-friendly interface and secure authentication powered by MongoDB, logging in becomes a breeze. Users simply enter their email address and password, and they're ready to dive into a world of unforgettable experiences. Our streamlined form ensures smooth navigation, allowing users to focus on what truly matters – their next adventure.

But wait, there's more! Don't have an account yet? No problem! Our login page also offers a convenient signup option, inviting users to join our vibrant community of travelers. With just a few clicks, they'll gain access to exclusive deals, personalized recommendations, and so much more.

- **Sign up Page**

Designed with simplicity and user-friendliness in mind, our Sign-Up process makes it easy for users to create their accounts and start planning their next adventure. Begin by entering your desired username, email address, and password – the keys to unlocking a world of travel possibilities. Share a bit about yourself by providing your location and a brief description. Whether users are seasoned globetrotters or first-time travelers, our platform welcomes everyone with open arms.

Once the form is completed, simply hit the "Create Account" button, and users are ready to go! Already a member? No problem! Users can easily log in using their credentials and pick up where they left off. At Dream Escape, we're committed to providing users with a seamless and enjoyable experience from sign-up to check-in.

- **Profile Page**

The profile page of our project is designed to provide users with a comprehensive view of their profile information and interactions within the platform. Upon visiting the profile page, users can see a personalized feed of their own posts and interactions, allowing them to easily engage with content. Additionally, users have the ability to update their profile information, including their avatar, location, and bio, through a user-friendly form interface.

Furthermore, the profile page is designed to be intuitive and responsive, providing a seamless user experience across devices. Whether users are accessing the platform from a desktop computer or a mobile device, they can easily navigate their profile, view their posts, and interact with other users. Overall, the profile page serves as a central hub for users to manage their profile information, engage with content, and connect with other members of the community, fostering a vibrant and active user experience.

- **Search**

The search feature of our project provides users with a powerful tool to discover and explore various attractions, restaurants, and points of interest in their desired location. Upon accessing the search page, users are greeted with an intuitive interface that allows them to enter their location or use their current geolocation to search for nearby places of interest.

Utilizing the Google Places API, the search feature dynamically fetches relevant place data based on the user's input, displaying a list of attractions along with corresponding details such as ratings, reviews, and categories. Users have the flexibility to filter search results by type (e.g., attractions, restaurants) and rating, allowing for personalized and tailored search experiences.

The search page is designed to be responsive and user-friendly, providing a seamless experience across devices and screen sizes. Whether users are exploring the platform on a desktop computer, tablet, or mobile device, they can easily navigate the search interface, view search results, and interact with individual place listings.

Furthermore, the integration of Google Maps enhances the search experience by providing users with a visual representation of search results plotted on an interactive map. Users can interact with map markers to view additional details about specific places and seamlessly switch between list and map views to explore search results in their preferred format.

- **Booking**

The booking feature of our platform offers users a convenient way to search for and book flights to their desired destinations, helping them spend less and travel more. The booking page presents users with a user-friendly interface where they can input their travel details and explore enticing deals on flights.

Upon accessing the booking page, users are greeted with options to select their preferred trip type—whether it's a round-trip or one-way journey. The interface dynamically adjusts based on the selected trip type, providing users with the necessary input fields to specify their origin and destination cities or airports, choose the number of passengers, and select their departure and return dates.

For round-trip bookings, users can easily toggle the visibility of the return date input field to streamline their booking experience. Additionally, users have the flexibility to customize their search parameters, such as the number of passengers and travel dates, to find the best flight options for their needs.

As users explore available flight options, they are presented with enticing deals on flights to various destinations. Each deal is accompanied by an image representing the destination city, along with details such as the city name, price per person, and special deal offers. Users can view more details about each deal and proceed to book their flights with just a few clicks, making the booking process seamless and hassle-free.

- **Blog/Feedback Page**

The blog page of our project serves as a dynamic platform where users can share their travel experiences, insights, and tips with the community. Upon accessing the blog page, users are greeted with a curated feed of blog posts contributed by fellow travelers, providing inspiration and valuable information for planning their own adventures.

The layout of the blog page is designed to be visually engaging and user-friendly, with posts arranged in a grid format for easy browsing. Users have the ability to interact with blog posts by liking, commenting, and sharing their favorite content, fostering a sense of community and collaboration within the platform.

One notable feature of the blog page is the inclusion of a post form, allowing logged-in users to contribute their own blog posts to the platform. This feature empowers users to share their unique travel stories and insights, enriching the overall content of the blog and enhancing the community experience. Furthermore, the blog page is optimized for responsiveness and accessibility, ensuring a seamless user experience across devices and screen sizes. Whether users are accessing the platform from a desktop computer, tablet, or mobile device, they can easily explore the blog, engage with content, and contribute their own posts.

3. Back end

● User Model

The user model defined in our backend application serves as the blueprint for storing user-related data in our MongoDB database. This schema incorporates various fields to capture essential user information, including their username, email, hashed password for security, profile picture, location, description, and saved places. Each user can also have associated posts and friends, which are stored as references to other documents in the database.

To ensure data integrity and security, the user schema includes validation rules such as required fields, unique constraints for usernames and emails, and minimum password length. Additionally, passwords are securely hashed using bcrypt before being stored in the database, enhancing protection against unauthorized access. Furthermore, the schema defines virtual properties like ``placeCount``, which dynamically calculate the number of saved places associated with each user, respectively. These virtuals provide convenient access to aggregated data without storing redundant information in the database.

Pre-save middleware is implemented to automatically hash passwords before saving new user documents or updating existing ones. This ensures that user passwords remain secure and protected against potential security breaches.

● Post/item Model

The post model in our backend application defines the structure for storing and managing post-related data in our MongoDB database. This schema captures essential details such as the post title, text content, creation timestamp, author username, associated comments, likes, and the author's unique identifier. Each post requires a title and text content, ensuring that users provide meaningful information when creating posts. The creation timestamp is automatically set to the current date and time using the ``Date.now``

function, with a custom getter to display the relative time since creation using the ``moment`` library.

The post schema includes an array of comments and likes, allowing users to engage with posts by adding comments or expressing appreciation through likes. Comments are stored as subdocuments using a separate comment schema, promoting modularity and code organization. Similarly, likes are represented as objects containing the username of the user who liked the post and the timestamp of the action. The schema also maintains a reference to the author of the post using their unique identifier, ensuring data integrity and establishing relationships between users and posts. This reference is immutable to prevent accidental modification and maintain consistency within the database.

Additionally, virtual properties such as ``commentCount`` and ``likeCount`` are defined to dynamically calculate the number of comments and likes associated with each post, respectively. These virtuals provide convenient access to aggregated data without storing redundant information in the database, optimizing performance and resource utilization.

● **Place Model**

The ``placeSchema`` is an essential component of our MongoDB database, defining the blueprint for storing information about various places. Each document adhering to this schema encapsulates details such as the unique ``placeId``, descriptive ``placeName``, succinct ``description``, indicative ``rating``, relevant ``tags``, and a link to the ``thumbnail_url`` image. These attributes collectively furnish a comprehensive profile for each place, facilitating efficient data organization and retrieval. By adhering to this structured schema, our application ensures consistency in place data representation, streamlining processes such as querying, filtering, and displaying places to users. This schema's versatility and adherence to MongoDB conventions empower our application to seamlessly manage diverse place information, enhancing user experiences and promoting effective exploration of different destinations.

● **Comment Model**

The ``commentSchema`` is an integral part of our MongoDB database structure, designed to encapsulate essential information about user comments on posts. This schema defines the blueprint for storing details such as the ``commentBody``, representing the textual content of the comment, the ``username`` of the commenter, and the ``createdAt`` timestamp indicating when the comment was created. Additionally, it includes a reference to the ``author``, ensuring that each comment is associated with the user who created it. By leveraging MongoDB's schema-based approach, this schema enables consistent and organized storage of comment data, facilitating efficient retrieval and manipulation within our application. Furthermore, the inclusion of `moment.js` allows for the automatic generation of human-readable timestamps, enhancing the user experience by providing contextually relevant time information. This schema's versatility and adherence to best practices in database design contribute to the overall functionality and usability of our application's commenting system.

- **GraphQL queries**

The GraphQL schema defines the structure of our API, specifying the types of data that can be queried and manipulated. In our schema, we have defined several types, including `User`, `Post`, `Comment`, `Like`, and `Place`, each representing different entities within our application. The `Query` type defines the available queries that can be made to retrieve data, such as fetching the currently authenticated user (`me`), fetching all users (`users`), fetching a specific user by their username (`user`), fetching posts authored by a specific user (`posts`), and fetching a single post by its ID (`post`).

The `Mutation` type defines the operations that can be performed to modify data, including logging in (`login`), adding a new user (`addUser`), updating user information (`updateUser`), adding a new post (`addPost`), deleting a post (`deletePost`), adding a comment to a post (`addComment`), deleting a comment from a post (`deleteComment`), liking a post (`likePost`), adding a friend (`addFriend`), removing a friend (`removeFriend`), saving a place (`savePlace`), and removing a saved place (`removePlace`).

Additionally, input types such as `PlaceInput` and `UpdateUserInput` are defined to specify the shape of data expected for certain mutation operations.

- **Resolver**

The resolver functions described in the provided code are integral parts of a backend server application built using Node.js and Express.js. These resolver functions handle incoming requests from clients and execute corresponding operations to interact with the MongoDB database. The resolver functions are organized based on the types defined in the GraphQL schema. For instance, query resolvers handle requests for fetching data, such as retrieving user profiles, posts, or comments. Mutation resolvers handle requests that modify data, including adding new users, creating posts, liking posts, adding comments, and managing friendships between users.

Authentication logic is also implemented within the resolver functions to ensure that only authenticated users can perform certain operations. This helps in securing sensitive functionalities and preventing unauthorized access to data.

Error handling is another crucial aspect of these resolver functions. They are equipped to handle potential errors that may occur during data retrieval or manipulation processes, ensuring robustness and reliability of the server application.

4. Database

The MongoDB database connection established using Mongoose serves as the persistent data storage solution for our project. MongoDB is a NoSQL database that offers flexibility and scalability, making it well-suited for storing unstructured or semi-structured data. In our project, MongoDB allows us to efficiently manage various types of data, such as user profiles, blog posts, travel destinations, and booking information.

We opted for MongoDB due to its schema-less nature, which accommodates the evolving nature of our project's data requirements. This flexibility enables us to store and retrieve data without strict schema constraints, facilitating rapid development and iteration. Additionally, MongoDB's support for nested documents and arrays aligns well with the

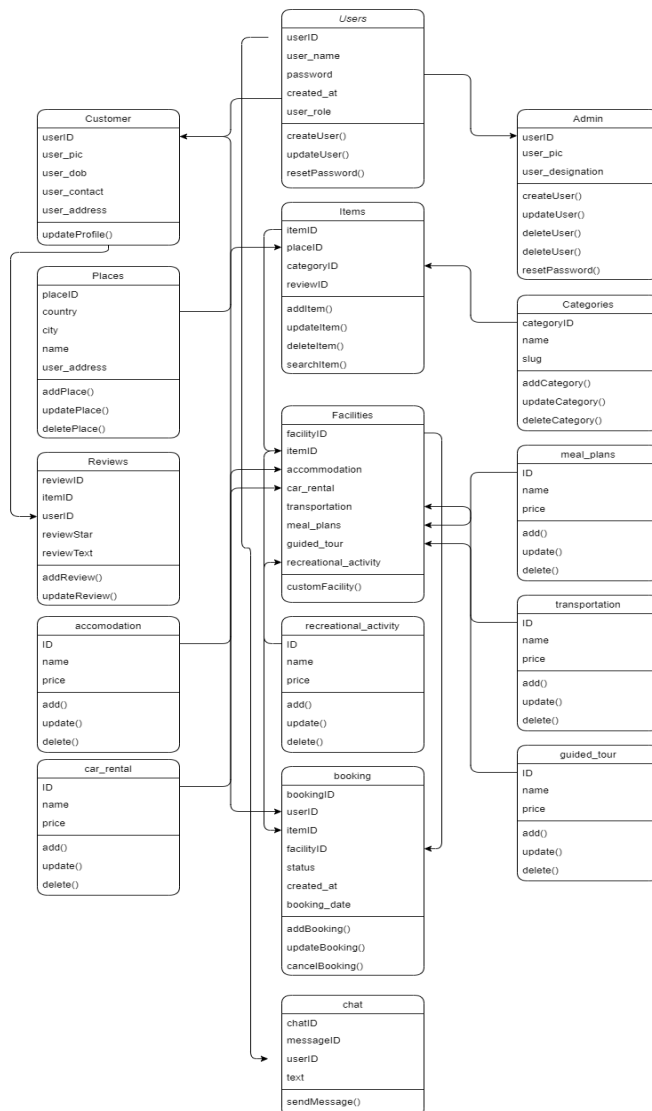
hierarchical structure of our project's data, such as user profiles with associated posts or booking details.

By using Mongoose, a MongoDB object modeling tool, we enhance our development workflow by providing schema validation, data mapping, and other utilities that streamline database interactions within our Node.js application. Through Mongoose, we can define schemas that enforce data consistency and integrity while leveraging MongoDB's scalability and performance benefits. Overall, MongoDB, coupled with Mongoose, empowers our project with a robust and scalable database solution tailored to our dynamic data storage needs.

UML Diagrams:

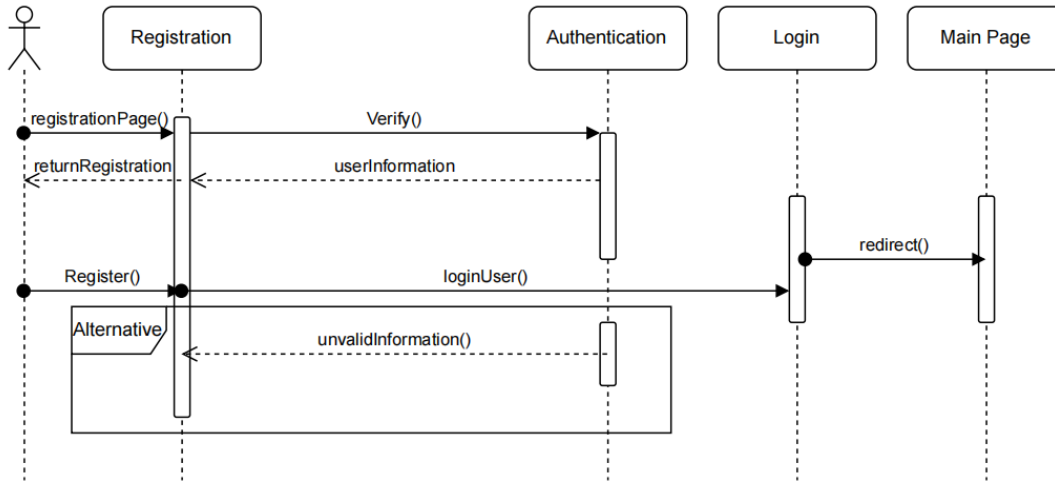
This phase most of the part was on front end but we have created whole UML:

1. Class Diagram:

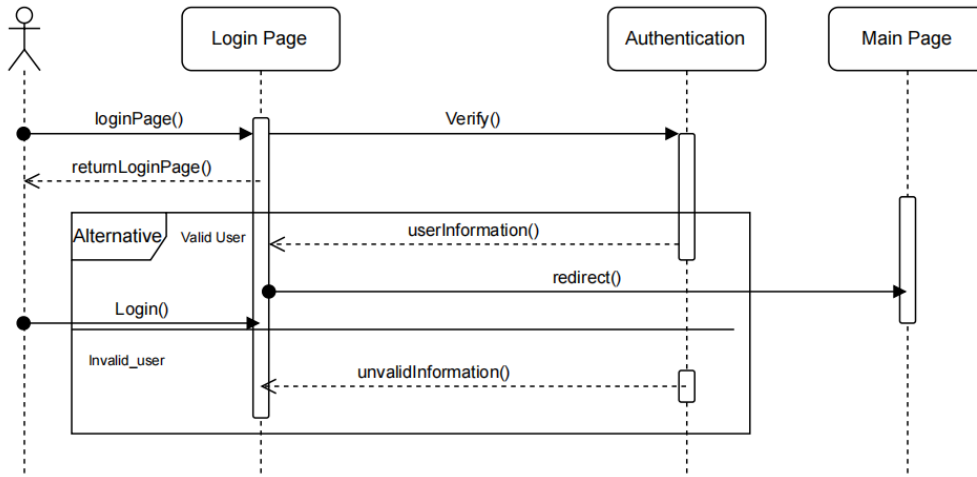


2. Sequence Diagram:

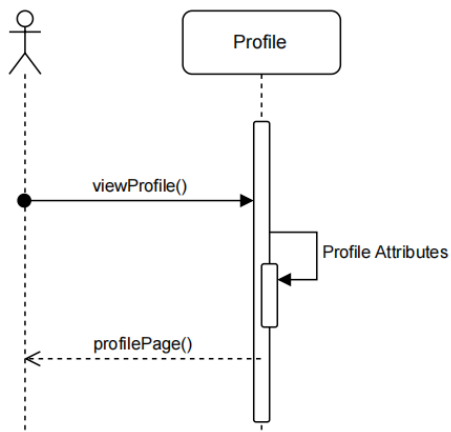
● Sign Up



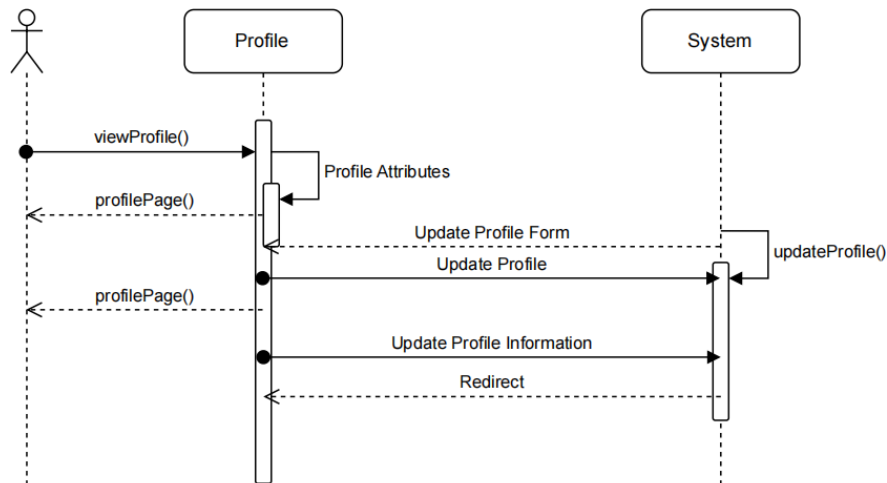
● Log in



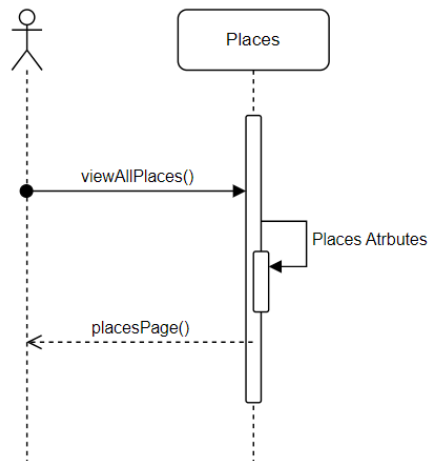
● Profile



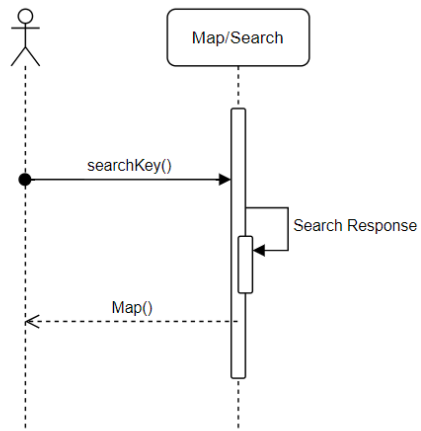
- Update Profile



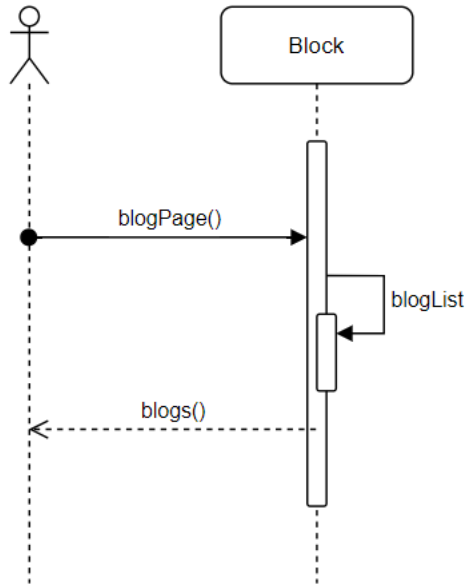
- Places



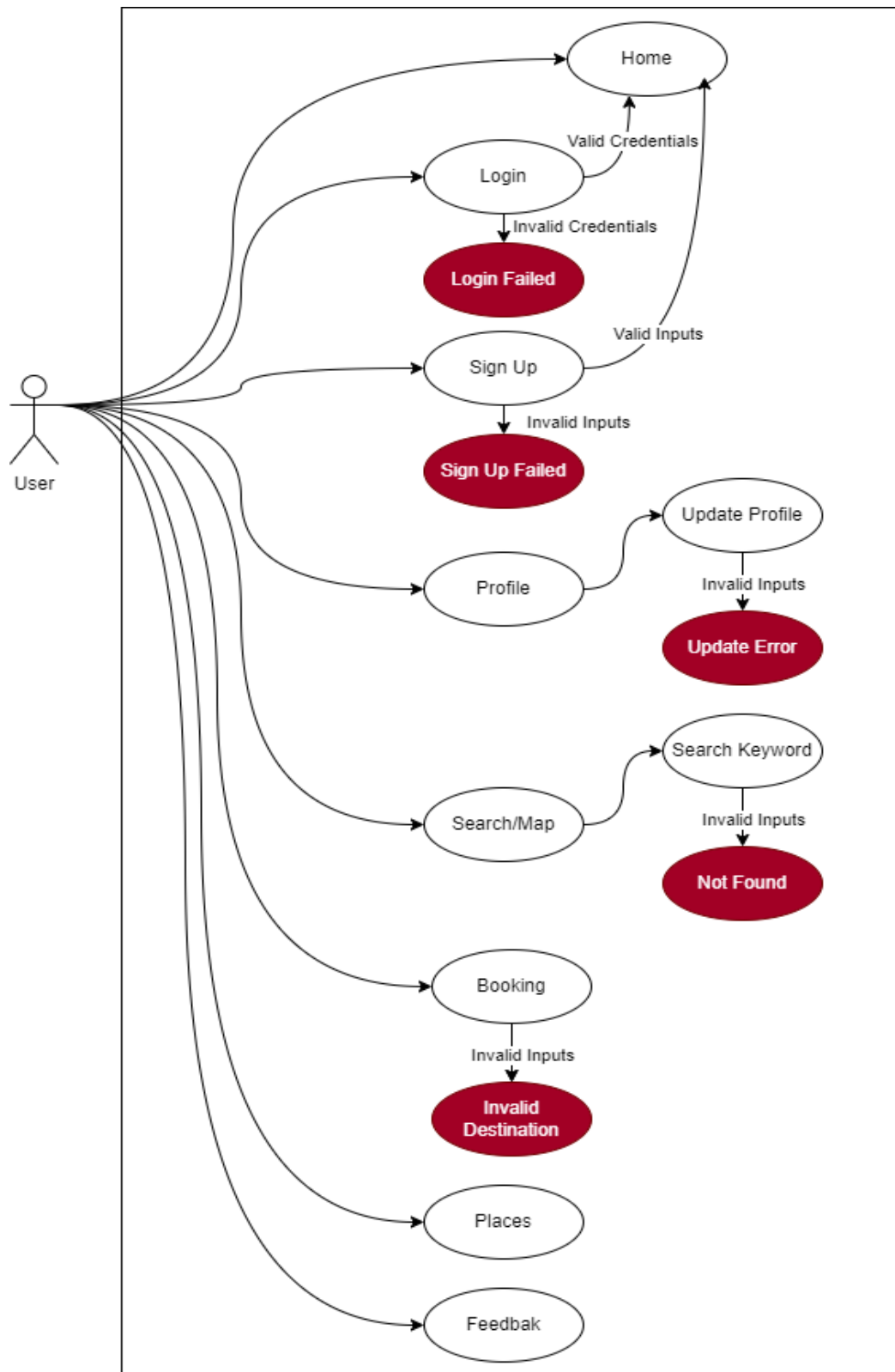
- Map



- Blog



3. Use Case Diagram:



Test Cases:

This phase mostly consists of the front end and navigation so, as a test case we have tested all the pages, their front end functions and also their navigation. Moreover, we have also tested the dynamic nature of our front end so it can be a better fit for all type of devices.

1. Login

DREAM ESCAPE #destinations #cities #travel tips #resources Blog Login Signup

WELCOME BACK
THERE'S A WHOLE WORLD OUT THERE

Log In

Don't have an account? [Sign up →](#)

DREAM ESCAPE
Privacy & Terms Policy Help Center Partnership Contact

© 2024 Dream Escape. All rights reserved. Made with ❤ by Dream Escape Team.

2. Sign up

DREAM ESCAPE #destinations #cities #travel tips #resources Blog Login Signup

SIGN UP
JOIN THE TRAVEL COMMUNITY

Create Account

Already a member? [Login →](#)

DREAM ESCAPE
Privacy & Terms Policy Help Center Partnership Contact

© 2024 Dream Escape. All rights reserved. Made with ❤ by Dream Escape Team.

3. Booking Front End

SPEND LESS. TRAVEL MORE.

ROUND-TRIP

ONE-WAY

FROM

Origin city or airport

TO

Destination city or airport

PASSENGERS

1 Passenger

DEPARTURE DATE


mm/dd/yyyy

RETURN DATE

mm/dd/yyyy

Find Your Trip

DON'T MISS THESE DEALS




Reykjavik, Iceland

3 nights hotel + Round-trip flight

\$556

per person

View Deal




Amalfi, Italy

3 nights hotel + Round-trip flight

\$749

per person

View Deal



Paris, France

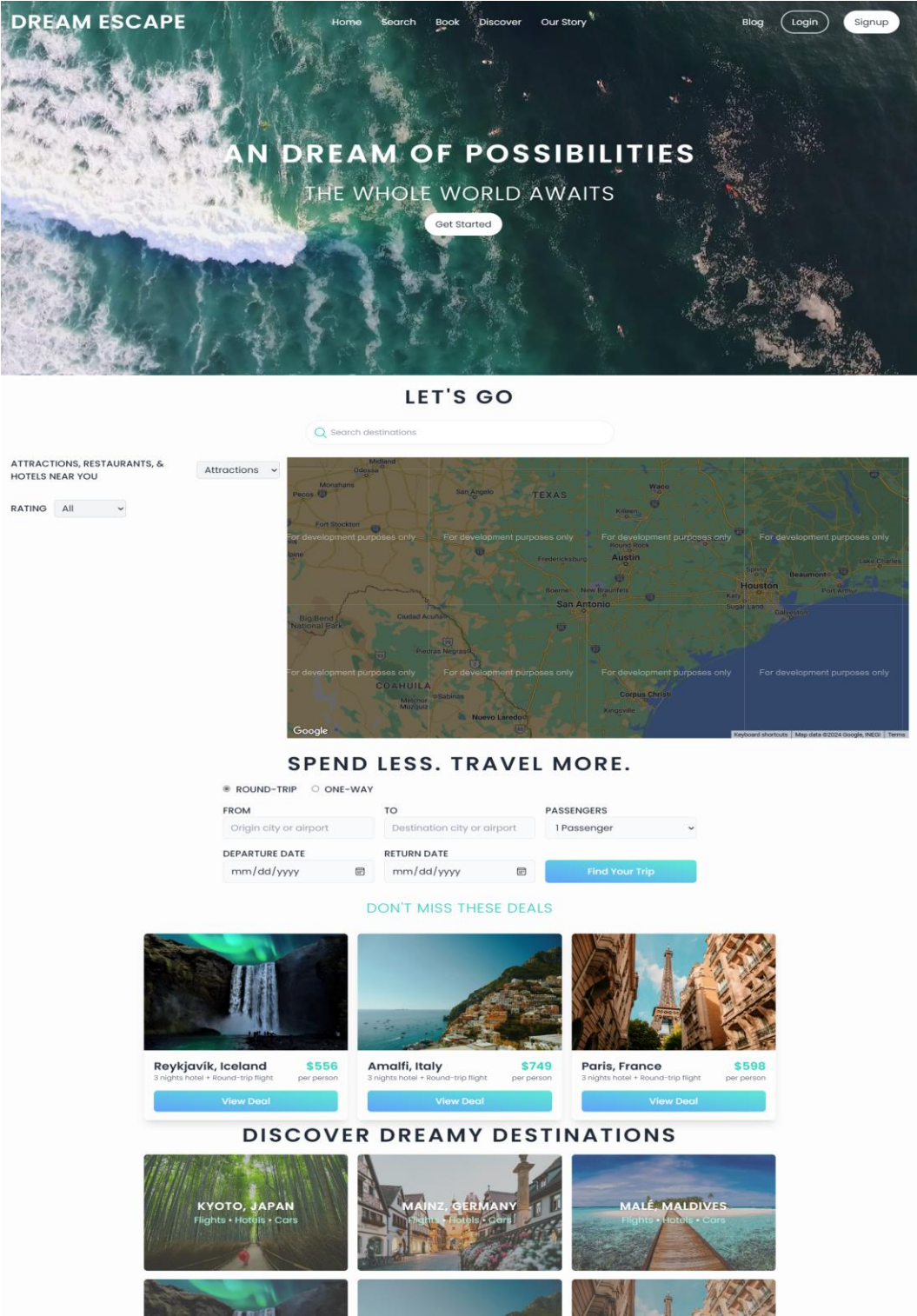
3 nights hotel + Round-trip flight

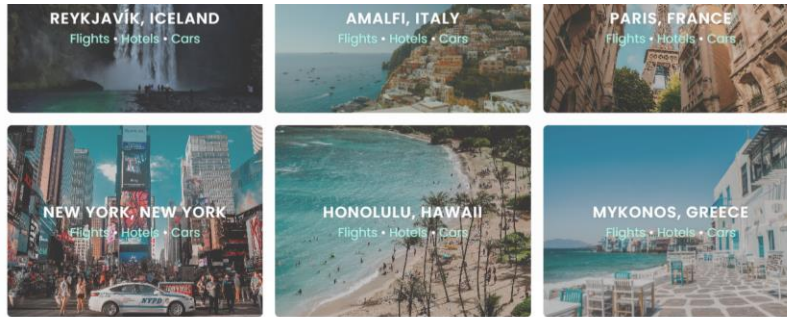
\$598

per person

View Deal

4. Home Page





BRINGING THE DREAMS TO THE WORLD

Dream Escape was founded in 2024 by a Dream Escape Team

At Dream Escape, we're more than just a booking platform – we're your partners in exploration, dedicated to unlocking the world's wonders one journey at a time. Founded on the belief that travel has the power to enrich lives and broaden horizons, we strive to connect travelers with authentic experiences that leave a lasting impression. With a commitment to excellence and a passion for adventure, we've earned a reputation as a trusted authority in the travel industry, helping countless explorers turn their wanderlust into reality.

Driven by a relentless pursuit of excellence, our team of travel enthusiasts is dedicated to crafting seamless and unforgettable experiences for every traveler. Whether you're embarking on a solo adventure, planning a romantic getaway, or organizing a group expedition, we're here to make your travel dreams come true. From meticulously curated itineraries to personalized recommendations, we're with you every step of the way, ensuring your journey is nothing short of extraordinary. Join us on a journey of discovery and let's create memories that will last a lifetime.



5. Map

LET'S GO

Search destinations

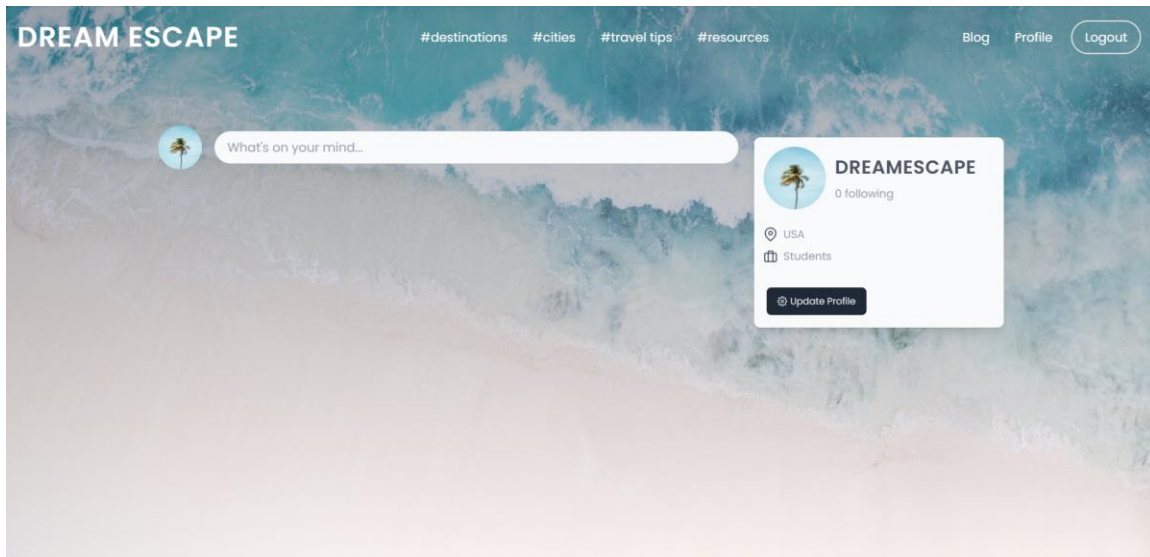
Attractions, Restaurants, & Hotels Near You

Attractions

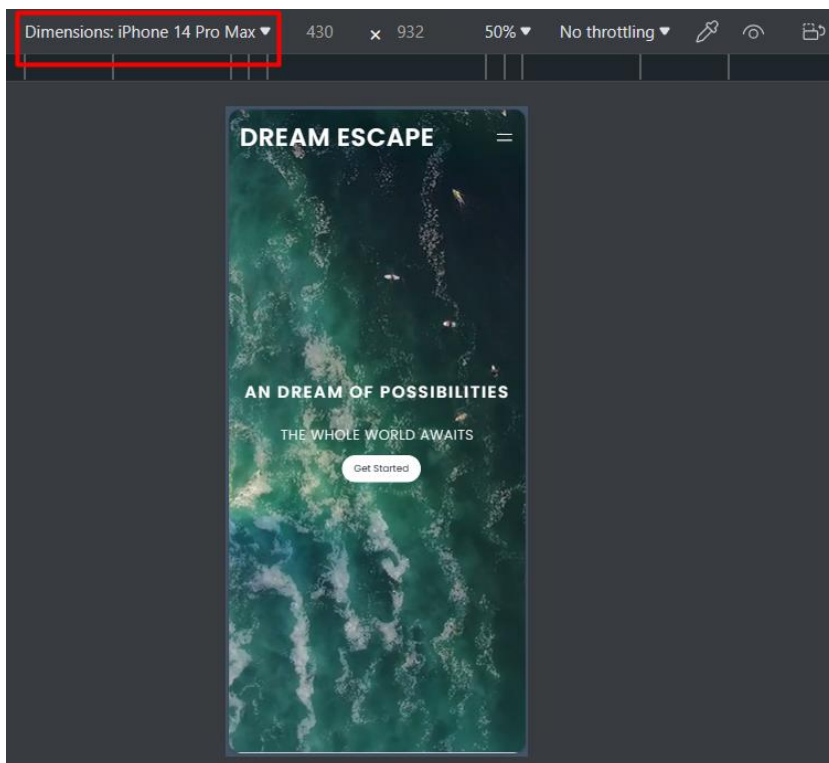
RATING All

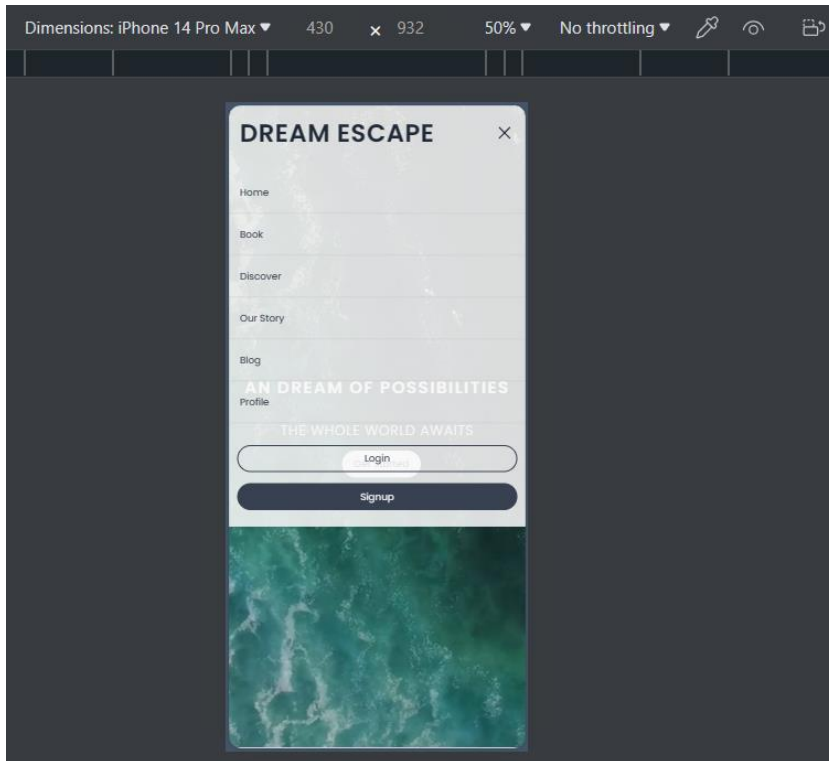
A map of Texas and surrounding regions, including parts of Oklahoma, New Mexico, and Louisiana. Major cities like San Antonio, Austin, Houston, and Corpus Christi are labeled. The map shows highways, water bodies, and various points of interest. A search bar at the top allows for destination searches. On the left, there are filters for 'Attractions, Restaurants, & Hotels Near You' and a 'RATING' dropdown set to 'All'. The map is sourced from Google Maps, with data from 2024.

6. Profile



7. Dynamic Page adjustment





Installation User Manual

- Prerequisites

User must need to install Nodes.JS from the official website <https://nodejs.org/en/download>. Install as per the user operating system either it is Windows, Mac, or Linux. After downloading and installing the nodes, please make sure the NPM which is actually nodes package manager, is also installed correctly and to check this please run this command

```
npm list -g
```

- Installation

Once nodes is configured, you must need to install some modules that are being used in this project. To do this, open the client folder in the CMD and run following command

```
npm install
```

Once all modules are installed successfully now it's time to open the server folder inside CMD and repeat the process again. By this all required modules for the front end and back end will be installed.

Program Run User Manual

To run the program, user need to open both client and server folders in two separate CMD. After that in both cmd please write this command

```
npm start
```

This will start two ports. On port 3001 the server is running which is actually back end of our project and on port 3000 the front end will start running and you can access this by this url in your local browser

<http://localhost:3000/>

Peer Review

Initially our target was to just create the front end as the phase 1 of the development but with the peer review and teacher review we have to change our plan as they prescribed the best practice will be to make both front end and back end along side and this will save a lot of end hustle as this can potential reduce development bugs. We have welcomed these reviews and made changes in our development phases requirements and as you can see in this phase we have created all 3 pillars of website: front end, back end and database.

Collaboration

Member name	Contribution description	Overall Contribution (%)
Veeresh Sakali	I have mainly worked on the front-end part especially on Home page, Login page and signup page.	12.5%
Susmith Meesa	I have worked on class diagram which acts as a blue print to understand the relation between classes and attributes, I also have assisted in back-end part.	12.5%
Harshath Budida	I have worked on Back-end part mainly on User model, item model, graphql queries and resolver.	12.5%
Harshitha Thokala	I have worked on Search feature and booking page which comes under front-end part and was a part of documentation making.	12.5%
Gopi Krishna Kummari	I have worked on front-end part Profile page and feedback page and helped in documentation process.	12.5%
Pooja Sree Poka	I have worked on back-end part mainly on place model, comment model and was a helping hand in documentation.	12.5%
Sashidhar Chary Viswanathula	I have worked on sequence diagram which is useful for understanding the flow control and communication.	12.5%
Balaji Valeti	I have worked on use case diagram which is a successful method for defining and arranging functional needs and documented the test cases.	12.5%