CAMPSPOT

1 Application Detail

Campspot is a web application built using Angular that allows users to view available campsites, log in, and add new camps to the list. The app provides a user-friendly interface to browse through camp locations, prices, stay duration, and ratings. Users can also submit new camps with details like images, descriptions, and prices.

The application features forms for user authentication and camp submission, and utilizes Angular's powerful features like routing, observables, pipes, directives, and services to create a dynamic and interactive experience.

2 Workflow

2.1 Initial User Access

- **Step 1:** The user accesses the Campspot Web Application in their browser.
    - The Homepage loads, showing a list of available campsites.
    - An API call is made to fetch the list of campsites from the backend (e.g., camp details such as location, price, ratings, and stay duration).
    - The fetched camp data is displayed to the user on the homepage.

2.2 User Authentication (Login/Sign Up)

- **Step 2:** If the user is not logged in and clicks on Log In or Sign Up, they are directed to the Login/Sign-up Page.
    - The user enters their credentials (e.g., email, password).
    - Form validation ensures the credentials are correct (e.g., email format, password length).
- **Step 3:** The user submits their credentials to the Authentication API.
    - If authentication is successful, a session token (or session data) is saved in local storage.
    - The user is then redirected to the Homepage after a successful login.

- If authentication fails, an error message is displayed to the user.

2.3 Browsing Campsites

- **Step 4:** After logging in, the user is presented with the list of available campsites on the Homepage.
  - An API call is made to fetch the list of campsites (if not already stored).
  - The camp data is displayed dynamically using Angular's data-binding and directives (e.g., ngFor).
  - Users can filter and sort campsites by criteria such as price, rating, stay duration, etc.

2.4 Submitting a New Camp

- **Step 5:** A logged-in user can choose to Add a New Camp by clicking a button or navigating to a form.
  - The user fills out the New Camp Form with the following required details:
    - Camp name
    - Description
    - Location
    - Price and stay duration
    - Images (required – must upload at least one image)
  - Form validation ensures all fields are filled out correctly, including requiring at least one image file.
- **Step 6:** Once the form is completed, the user submits the data.
  - The image URLs (instead of actual file uploads) are validated to ensure they are correct and point to actual image resources.
  - An API call (POST request) is made to the backend to save the new camp with all its details, including the provided image URLs.
  - After the camp is successfully added, the user is redirected back to the Homepage, and the list of camps is updated to reflect the newly added camp.

2.5 User Logout

- **Step 7:** When the user decides to log out:

- The session data (user information and session token) is removed from local storage.
- The user is redirected to the Login Page.

2.6 Error Handling

- **Step 8:** If there's an error at any stage (e.g., failed login, invalid form data, failed API request):
  - An error message is displayed to notify the user (e.g., "Login failed" or "Please fill out all required fields").
  - Error messages typically appear briefly and are cleared once acknowledged.

2.7 Real-Time Data Updates

- **Step 9:** After the user adds a new campsite, the Homepage will automatically update to reflect the new data.
  - The campsite list is re-fetched or updated dynamically from the API (or local storage).
  - Users will see the newly added camp immediately without needing to refresh the page.

3 Angular Concepts

3.1 Components

- The project contains the following components:
  - AppComponent
  - HomeComponent
  - CampsComponent
  - NewCampComponent
  - NavBarComponent
  - LoginComponent
  - FooterComponent

3.2 Parent to Child Components

- Implemented parent-child relationship between components:
    - The HomeComponent acts as a parent and contains a list of camps. It passes the data to the CampsComponent, which renders individual camp details.

3.3 Structural and Attribute Directives

- **Structural Directives**:
    - Used in the NewCampComponent to conditionally display the success message when a new camp is successfully submitted. The directive dynamically adds or removes the success message from the DOM based on the form submission status.
    - ***ngFor**: Used in the HomeComponent to iterate over the list of camps and display individual camp cards dynamically.
- **Attribute Directives**:
    - Implemented in the NavBarComponent for dynamic routing, user interaction, and styling adjustments.
    - **[routerLink]**: Used to navigate between pages based on user clicks.
    - **[class.img-container]**: Dynamically applied to conditionally style the user icon container depending on the login state.
    - **[ngClass]**: Used to toggle CSS classes based on a condition to display or hide the popup menu.

3.4 @Input

- In the HomeComponent, data is passed to the CampsComponent using @Input(). The HomeComponent sends the list of camps, and CampsComponent receives and displays them.

3.5 Basic Routing

- The application includes basic routing to navigate between different views:
    - **HomeComponent**: Displays the homepage.
    - **LoginComponent**: Handles user login functionality.
    - **NewCampComponent**: Handles the form for adding a new campsite.

3.6 Services

- Implemented a service CampsService to interact with the backend API for fetching and adding camp details.

3.7 Observables

- Used Observables to handle asynchronous HTTP requests for fetching and submitting camp data through the CampsService.

3.8 API Calls

- The CampsService makes GET and POST requests to the backend API for retrieving and submitting camp data.

3.9 Template and Reactive Forms

- Template-driven form in the NewCampComponent is used for adding a new camp with validation.
- Reactive form in the LoginComponent is used for validating the login form, including username and password fields.

3.10 Pipes

- Custom pipes like customCurrency and star have been created and used in the CampsComponent for displaying prices in INR format and showing ratings with stars, respectively.