Travel Planner

Objects

1. **Constructing a Travel Planner**
    - Create an object `TravelPlanner`. This object should contain an empty array called `locations`. Each element of the `locations` array is an object which contains the following information about the location: `name`, `description`, `bestTimeToVisit`, and `localCuisine`.
2. **Adding Locations**
    - Implement a method `addLocation` in `TravelPlanner` which receives a location name and an object containing information about the location. Before adding a new location object to the `locations` array, the method should check if a location with the same name already exists in the array. If so, it should log an error message and should not add the duplicate location.
3. **Deleting Locations**
    - Implement a method `deleteLocation` that takes a location name as input and removes it from the `locations` array. If no location matches the given name, the method should log an appropriate message.
4. **Searching Locations**
    - Implement a method `findLocation` that takes a location name as input and returns the location object. If no location matches the input, the method should return `null`.
5. **Sorting Locations**
    - Implement a method `sortLocations` that sorts the locations alphabetically based on their names.
6. **Edit Location**
    - Implement a method `editLocation` that takes a location name and an updated information object as inputs and updates the corresponding location in the

`locations` array. If no location matches the given name, the method should log an appropriate message.

7. **Show All Locations**
    - Implement a method `showAll` that logs all location names and their corresponding details to the console.
8. **Filtering Locations**
    - Implement a method `filterLocations` that takes a string as an argument and returns all locations whose description contains that string. The method should be case-insensitive.
9. **Implementing Ratings**
    - Extend the `TravelPlanner` to also support ratings for locations. Each location should now also have a `ratings` property, which is an array of numbers, each between 1 and 5.
    - Implement a method `rateLocation` that takes a location name and a rating as input and adds the rating to the corresponding location's `ratings` array. The method should handle invalid inputs appropriately.
    - Implement a method `getAverageRating` that takes a location name as input and returns the average of all ratings for that location. If no ratings exist for the location, the method should return `null`.
10. **Top Rated Location**
    - Implement a method `getTopRatedLocation` that finds and returns the location with the highest average rating. If multiple locations share the highest average rating, the method should return all of them.