Service Rating Website

API wrapper for restaurants, movies, and hotels

(Sarthak & Prathamesh)

Everyone must have used apps online to order food, electronics, etc. Even if not, at least you must know about it. There are also websites to book hotels for a stay. The plain old Google Maps also displays ratings of restaurants, hotels, and shops. Have you ever thought about how these applications work or how they break down into components that are then connected? Choose this project to find out and design one yourself!

Framework

You need to design a **web-API wrapper** that can query external APIs for information about services like hotels and restaurants based on the user's parameters. The requirements mentioned here are the minimum ones. You are free to add other features and implementations to your wrapper. The parts and sub-parts you need to make are listed to help you divide the work among teammates. **We recommend (note - this does not mean you have to use it) using Django or Django-REST for the back end and ReactJS or Angular for the front end.**

Logistics

Each section/feature has some allocated marks. You are free to add more features, but make sure you are prepared to explain them during the viva. We might have an objective (auto-graded) evaluation for some project sections.

Requirements

- **Signup/Login Page**
- **Logout Page**
- **Profile Page**
- **Dashboard**
- **Display**
- **Search with Filters**
- **Reviews**

Details

Signup/Login

Here it would be best if you implemented authentication so that users can sign up/log in to the wrapper website using a username and password. You should also ensure no duplicate users (same username). Once signed up, the user should be able to log in using the same credentials.

Logout

Logout will log the user out but should preserve the credentials provided.

Profile

Display user information along with a list of reviews this user had submitted to the website. You can add an option to update profile information.

Dashboard

This page should show the option to search across hotels and restaurants and a separate search for movies. Base the search on location (primarily) and any other parameters you wish to search using. You can assume that searching would be using strings. There should be separate redirects to pages for hotels, movies, and restaurants, respectively.

Display

These are separate pages, one each for hotels, movies, and restaurants. There should be a list of ‘suggested’ hotels (or movies or restaurants) on this page, sorted based on ratings and reviews or whatever parameters you see fit.

Search

This search tool will allow users to browse through available options, depending on the parameter they want to search for. The results displayed should be sorted as you see fit. The data displayed can be updated dynamically or periodically.

Filters

This section involves filtering the results based on price range, rating range, etc., which can be chosen using either a checkbox or a two-pointer slider. Filter and display the results accordingly. Try to make filters as per the properties in API.

Reviews

When the user selects some result from the display, they should have the option to add their review (via a form object that displays fields) when they click a button. The database should accordingly update the data. This submitted review should also appear on the user's profile page when they visit that.

Beginning

To start, think about the database and the components/objects you might need to store. What kind of tables will you need? Consider all things, like users, hotels, restaurants, types of queries, what connections/mappings you need between components, etc.

Also, look out for available external APIs that can suit your needs. A good starting point to browse for APIs is RapidAPI Hub.

It would be best if you hosted the database and back-end on a local server. Your API wrapper should query this server to get the data. Make sure users only see information that is relevant to them. You can use Xampp as a local server. A recycler view might be better than a simple list view.

Report

It would help if you documented an explanation for your project in a report (for example, a README of the GitHub repository that you may use to collaborate). Mention the framework and functionalities so that any third person can understand what the project implements just by reading through the README.

Bonus

Designing the user interface to appear clean and user-friendly, adding additional features, adding more service APIs, making the wrapper as robust as possible, etc., all count as bonuses. We always welcome creativity.