Subject: Back-end API Design

Date: October 26, 2020

The problem given is pretty straightforward. The approach I've thought of involves the data to be stored in a database, in this case I've used Postgres. Then it establishes a connection to the database via the back-end to access the data. After this, I've carved up the entire task into two parts though so much is there to implement apart from these two I'd say these two are vital. One is authentication and authorization. As the entire thing stays online, and the clients are allowed to interact with the data authentication plays a vital role in maintaining the security and integrity of the data. The second task involves writing the core API to provide access to the client for interacting with the data. Using the proper HTTP methods along with the authentication method that's already written, this task can be implemented in a pretty decent manner.

So, the first step is to implement proper authentication and authorization methodology. For achieving this, I thought of using JSON Web Tokens (JWT). JWT is very easy to understand and implement as it requires no additional server to store the information. Inbuilt JWT.sign() method is used for taking the user credentials, and by using those and by using a secret-key a user token can be generated with a specified time for expiry and pass this token wherever it's required. Then, the middleware is required to verify the token via the jwt.verify() method. This middleware can be used wherever required i.e, instead of just writing a query directly to create new user data the user token can be verified first and once it's matched then the users are allowed to add a new user, and similarly, the same can be implemented wherever needed. JWT is pretty basic and once this entire thing is made into an app we can add stuff like Oauth to make user authentication and authorization more elegant and secure.

After this, as said earlier, the main task is to implement the core API i.e, allowing users to access the data. I've written the API showcasing the main HTTP methods allowing users to interact with the data and perform operations based on the requirements. The API I've written can be improved very much to make it more adaptable to the client's needs. The main idea is instead of throwing the entire data from the CSV to a single table, several tables can be created based on the requirements like a table for entire user-related stuff(id, userID, vehicleModelID, etc), and another one involving the locations, and so on. This way more complex queries can be written like getting the users who made the booking via an online site or mobile site, and though this can be done with the single table also, making several separate tables keeps everything clean and makes the fetching of data a little smooth. I've written the most common HTTP codes for letting the client know whether the data has been posted or if the data he wants exists like sending a 200 if everything's fine or sending a 409 for conflict like if the client is trying to update or create a user that already exists, and so on. These status codes can be made more explicit for helping the client understand any fragile situation. The idea is to write the status codes along with a clear context so that the client can know what's happening. Instead of sending a status code like "400 Bad Request", a clearer response is sent that includes information like the server used, whether the connection is closed or open and a clean error message stating what's wrong like "400 Bad Request: Your request is missing ¡some¿ parameters. So verify it and resubmit".