## **Travel Memory Application Deployment**

The Travel Memory Application is used to create and keep a record of the memories of your most beautiful travel experinces.

The TravelMemory application has been developed using the MERN stack. The application consists of frontend & backend which are developed using JavaScript programming language. In this task, our responsibility is to deply the application onto the serves, so that it is accessible to the users to share their travel experiences.

**Objectives :**

* Set up the backend running on Node.js.
* Configure the front end designed with React.
* Ensure efficient communication between the front end and back end.
* Deploy the full application on an EC2 instance.
* Facilitate load balancing by creating multiple instances of the application.
* Connect a custom domain through Cloudflare.

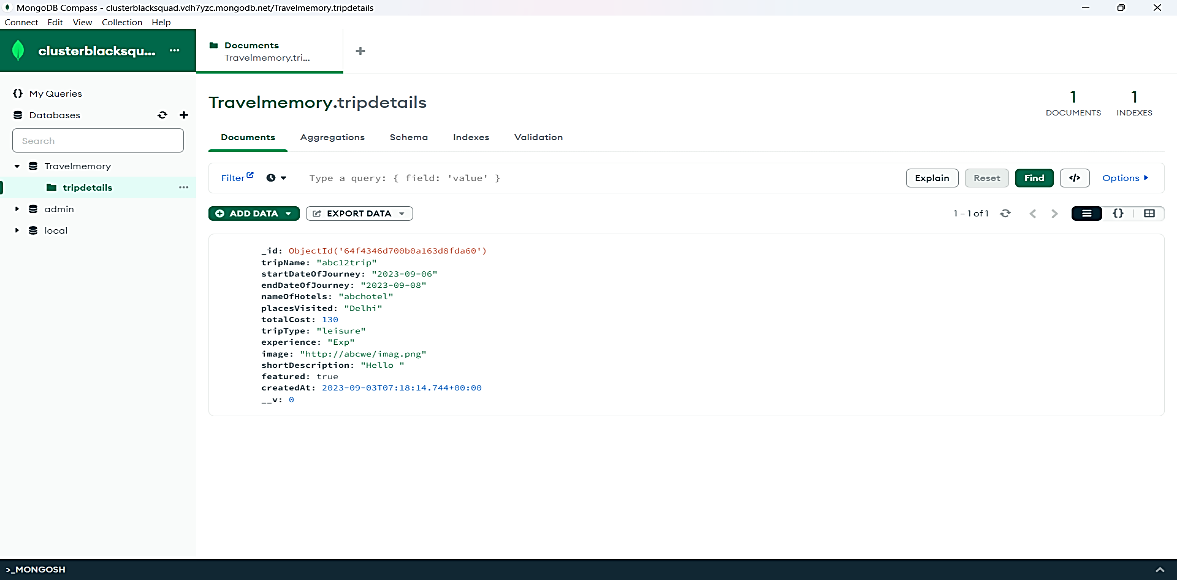
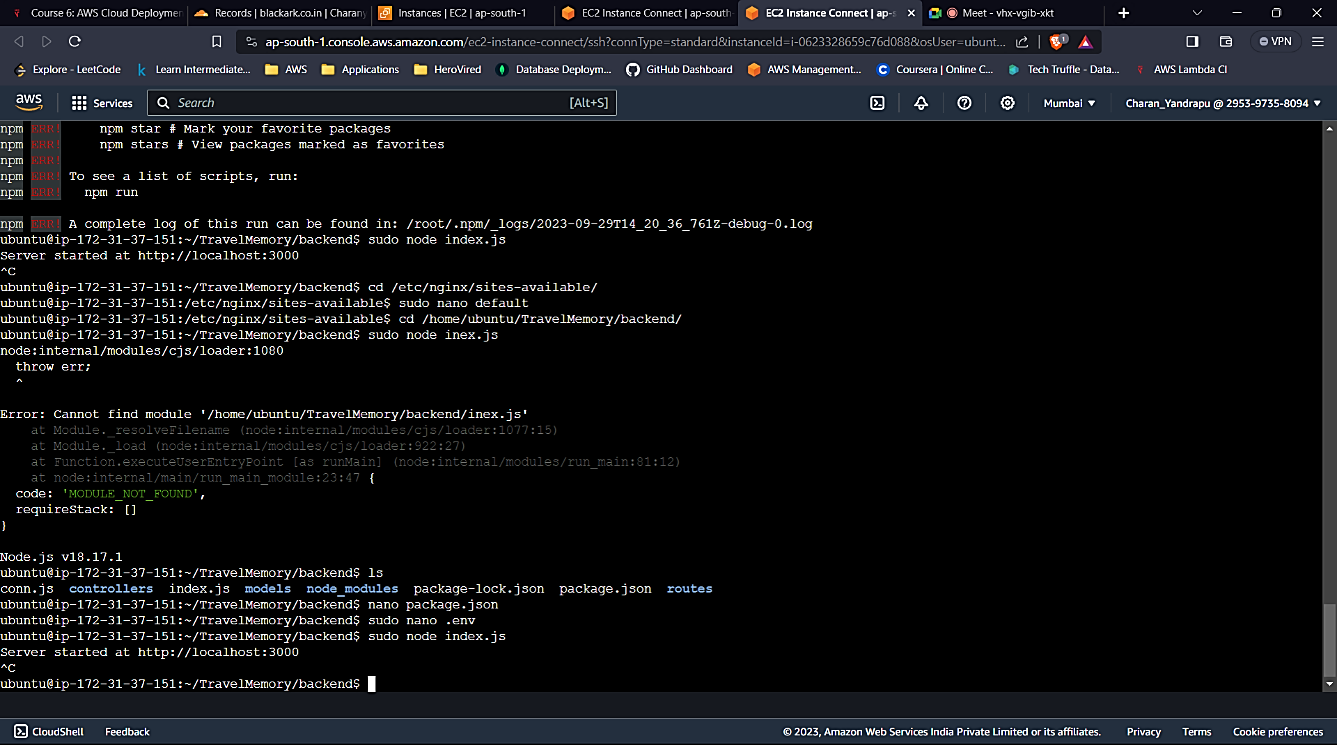
**Architecture :**

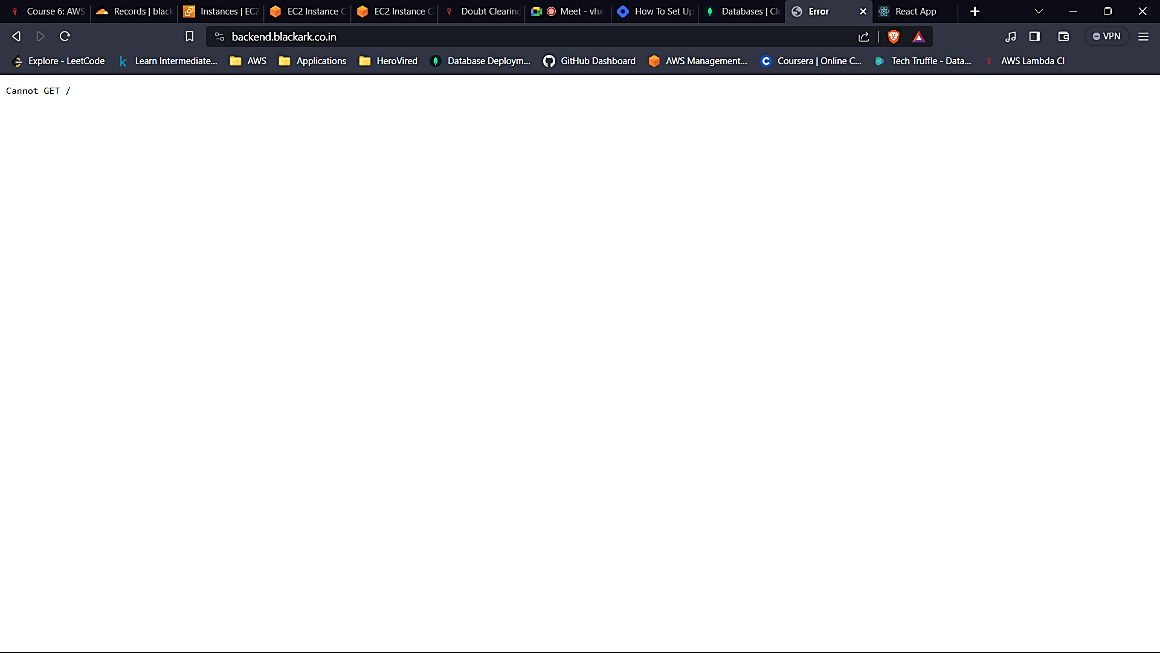
You can find the architecture at the end of this document.

**Task 1 : Backend Configuration**

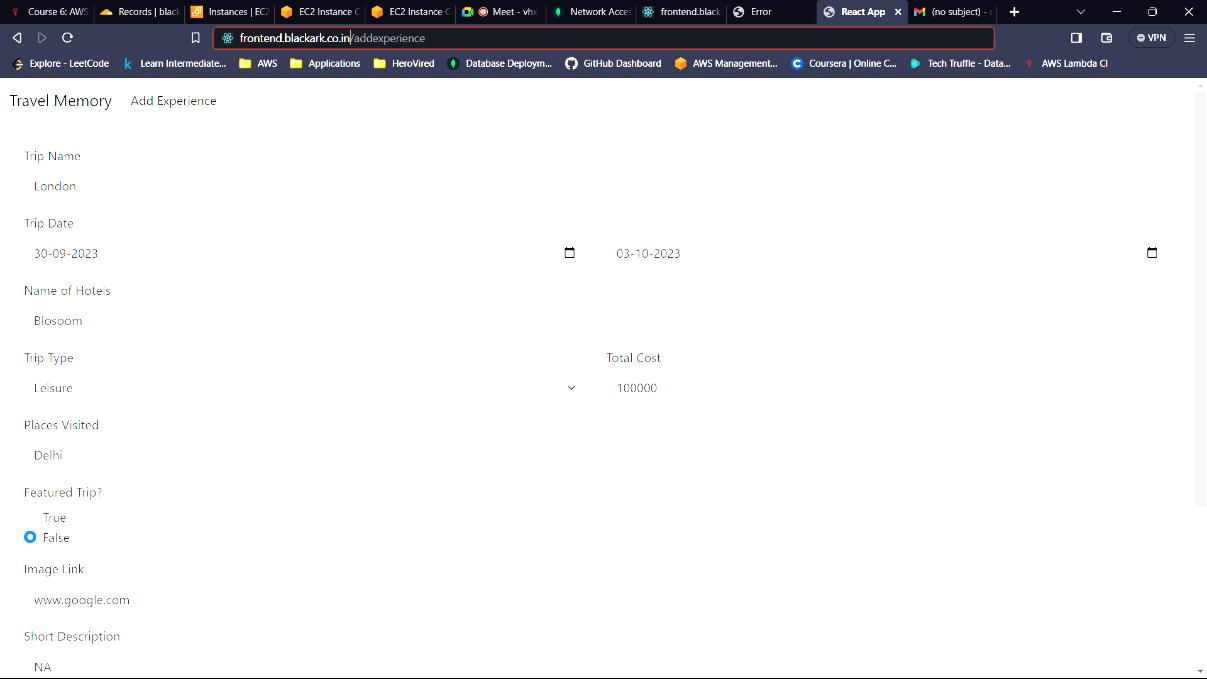
* Access the complete codebase of TravelMemory Application from the GitHub link.

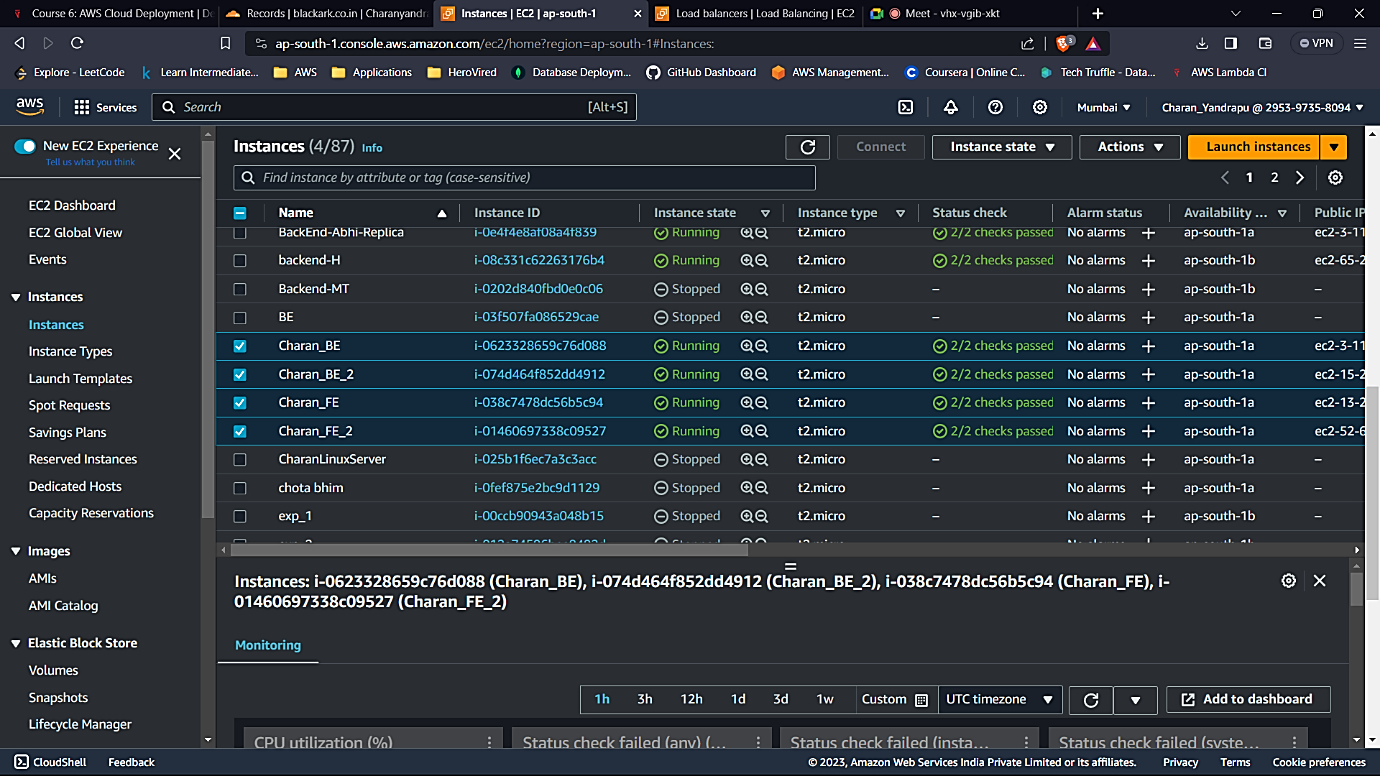
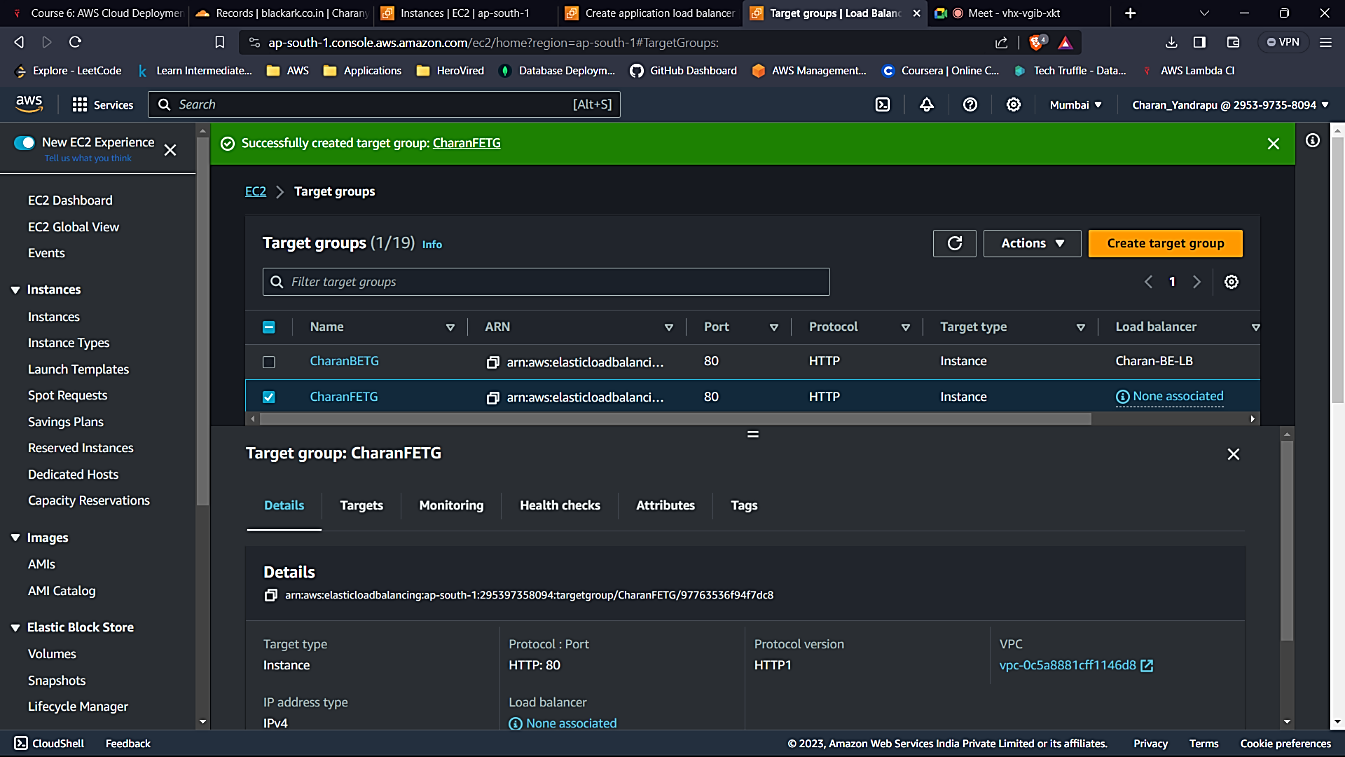
<https://github.com/UnpredictablePrashant/TravelMemory>

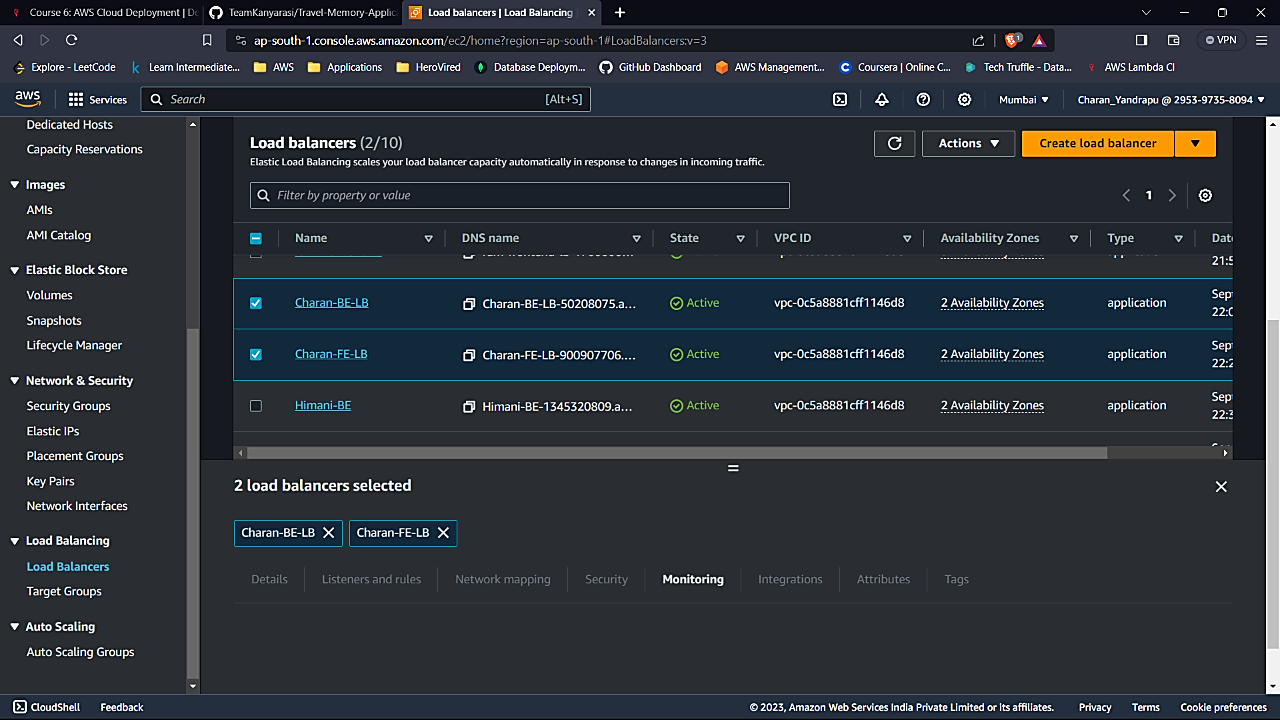
* Clone the complete repository and navigate to the backend directory of the TravelMemory application.
* Make the necessary changes into the codebase as mentioned in the README.md file in the repository.
* Update the .env file in the backend directory of the application to incorporate with the database connection details and port information.
* To connect the backend with the database utilize Mongo DB compass and enter the Mongo DB URL and the port number in the .env file.
* Configure the backend to run on PORT=3000 and set up the reverse proxy by installing Nginx and run it on PORT=80.
* To set up the reverse proxy make the necessary changes in the /etc/nginx/sites-available/default file with the details of the subdomain and the proxy\_pass = <http://webserver>.
* If the reverse proxy setup is done successfully, this is how the backend application with the subdomain looks like.

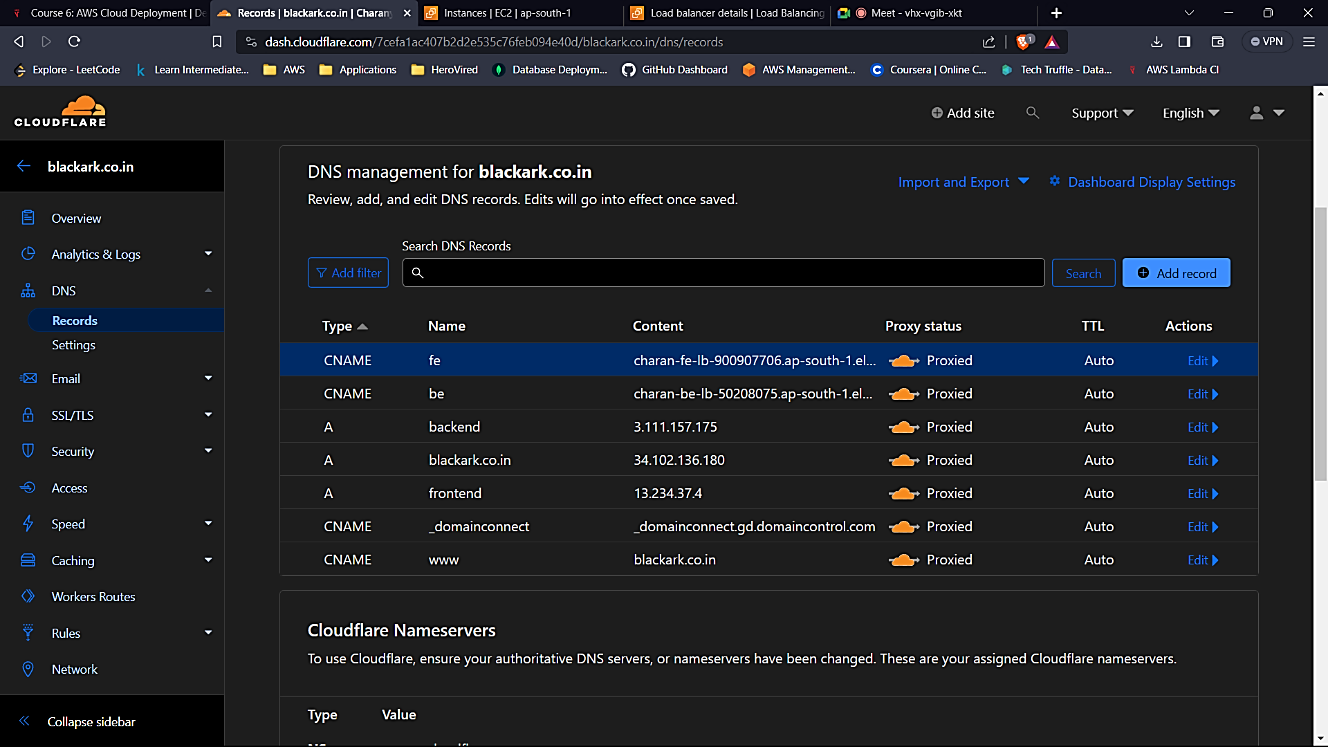
**TASK 2 : Frontend And Backend Connection**

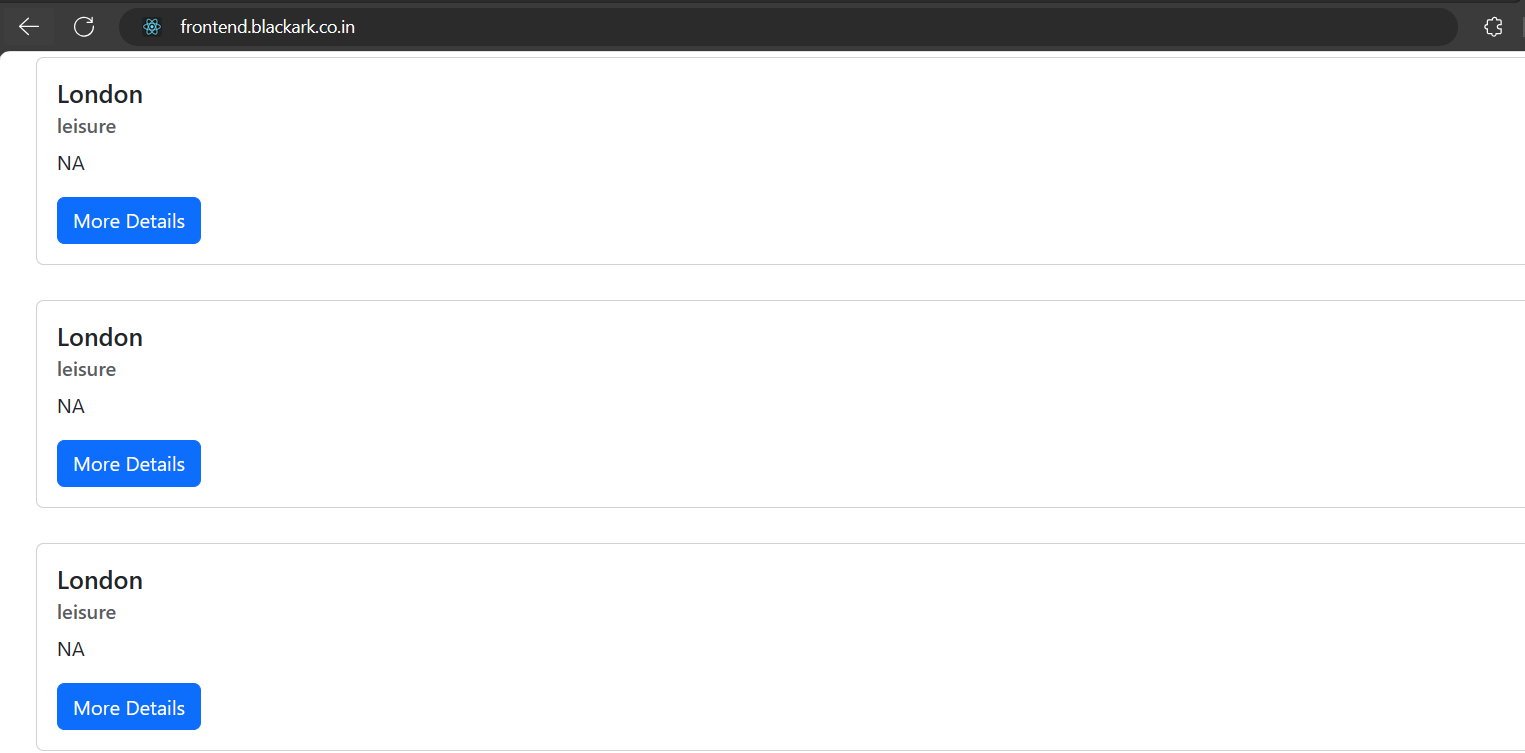
* Navigate to the frontend directory of the application and in the url.js file update it with the subdomain of the backend application.
* Also make the necessary changes in the package.json file, so that the frontend application runs on the preferred port number when the servers are up.
* Follow the same steps for the reverse proxy setup as done in the backend application.
* If the reverse proxy setup for the frontend is done successfully, this is how the frontend application with the subdomain looks.

**TASK 3 : Scaling the Application**

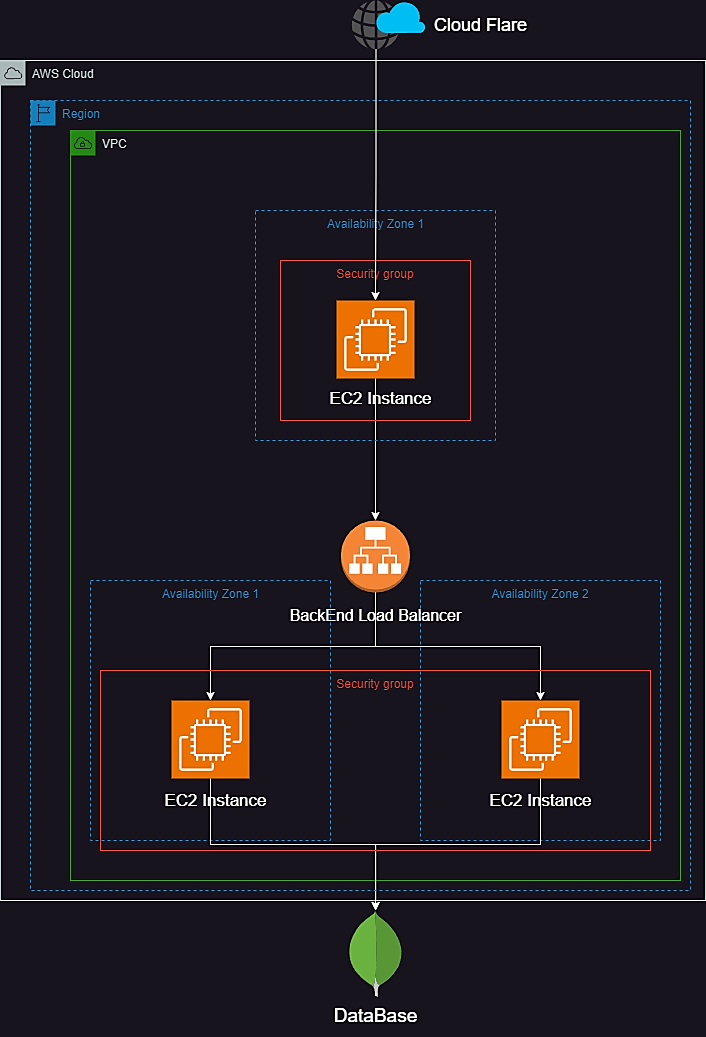
* To scale the application, we need to create multiple instances with the same configurations by creating the snapshots of the frontend and backend application.
* We need to create the AMI and launch new instances using the AMI.
* We need to create a load balancer to ensure the efficiency of incoming traffic.
* Create the application load balancers by specifying the instances into their respective target groups.

**TASK 4 : Domain Setup with CloudFlare**

* Connect your custom domain to the application using Cloudflare.
* Create a CNAME record pointing to the load balancer endpoint of the backend.
* Create an A record with the IP address of the EC2 instance hosting the frontend.

**TASK 5 : Successful Deployment**

**ARCHITECTURE ( According to the Assignment ) :**



**ARCHITECTURE FOR HIGH AVAILABILITY, SECURITY & PERFORMANCE :**

