Introduction:

The TravelMemory application has been developed using the MERN stack. Your challenge is to deploy this application on an Amazon EC2 instance. This will provide you with hands-on experience in deploying full-stack applications, working with cloud platforms, and ensuring scalable architecture.
Project Repository:
Access the complete codebase of the TravelMemory application from the provided GitHub link: https://github.com/UnpredictablePrashant/TravelMemory
Objective:
- 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.
Tasks:
1. Backend Configuration:
- Clone the repository and navigate to the backend directory.
- The backend runs on port 3000. Set up a reverse proxy using nginx to ensure smooth deployment on EC2.
- Update the *.env* file to incorporate database connection details and port information.
2. Frontend and Backend Connection:
- Navigate to the `urls.js` in the frontend directory.
- Update the file to ensure the front end communicates effectively with the backend.
3. Scaling the Application:
- Create multiple instances of both the frontend and backend servers.
- Add these instances to a load balancer to ensure efficient distribution of incoming traffic.
4. Domain Setup with Cloudflare:
- Connect your custom domain to the application using Cloudflare.
- Create a CNAME record pointing to the load balancer endpoint.
- Set up an A record with the IP address of the EC2 instance hosting the front end.

1. Backend Configuration:
- Clone the repository and navigate to the backend directory.
- The backend runs on port 3000. Set up a reverse proxy using nginx to ensure smooth deployment on EC2.
- Update the *.env* file to incorporate database connection details and port information.

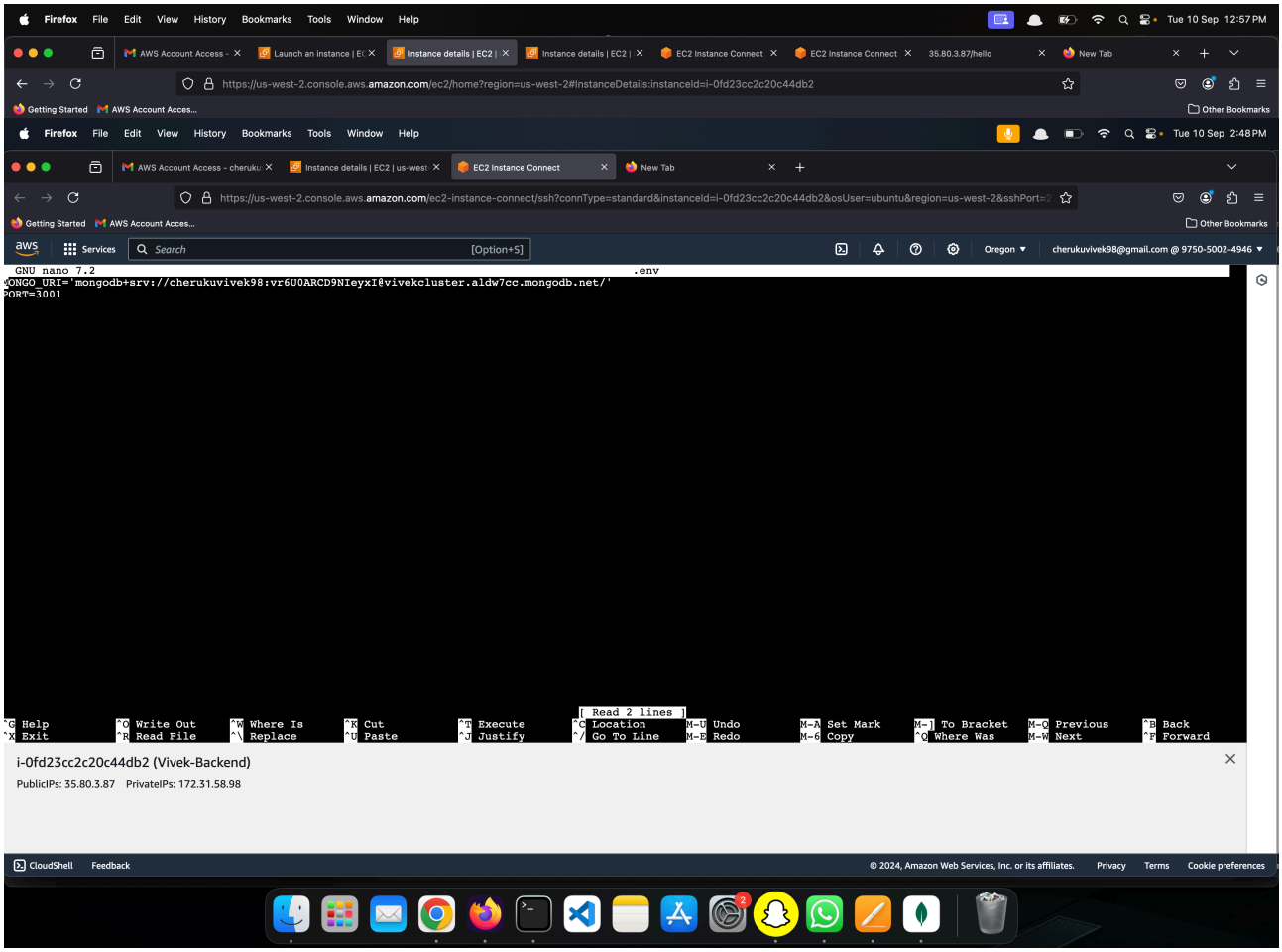
Created the EC2 Instances for both backend and frontend
In backend clone the repo using -> git clone https://github.com/UnpredictablePrashant/TravelMemory/
Navigate to the path for backend /home/ubuntu/Travelmemory/backend

Create a .env -> nano .env
Update the details of mongo BD cluster with
MONGO_URI='ENTER_YOUR_URL'
PORT=3001

Proceed with the sudo apt install npm command to install the nesscary packages.
For Reverse proxy Nginx
-> sudo apt install nginx
and proceed with the path

Sudo nano /etc/nginx/sites-available/default

Add the following configuration:



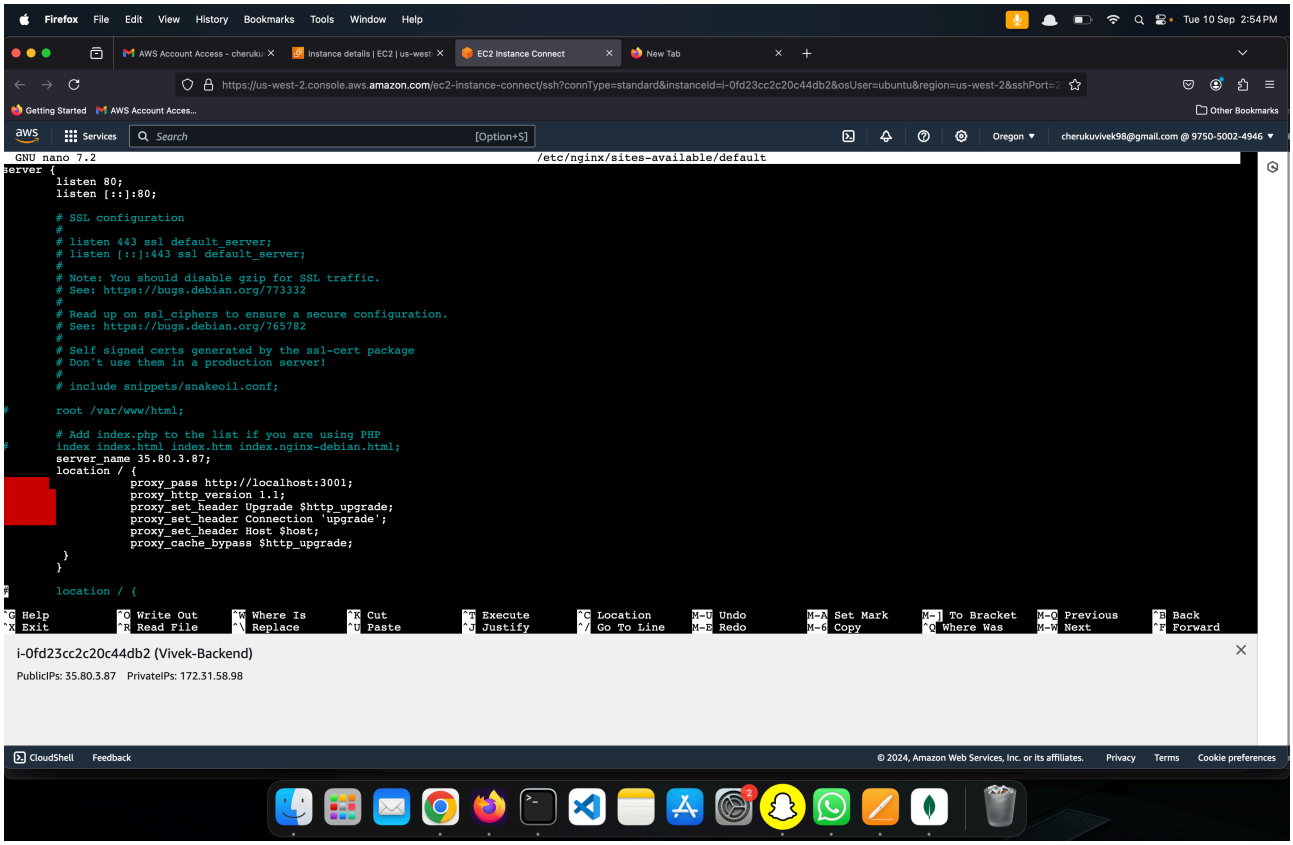

```
  server {
    listen 80;
    server_name “Backend_public_id”;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

check for the nginx test with -> sudo nginx -t if successful proceed to restart the nginx with sudo systemctl restart nginx

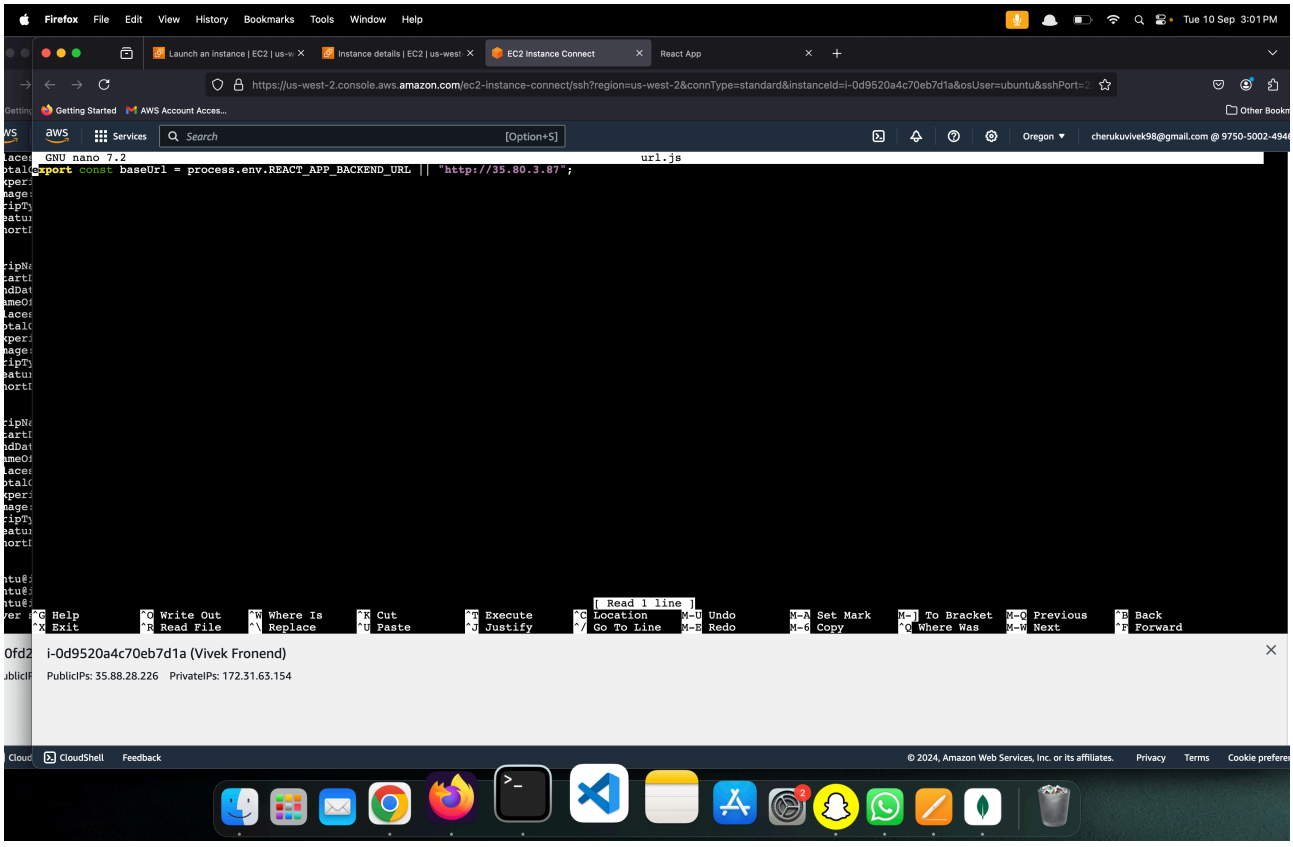
Once done proceed to backend folder and start the 3001 port via command node index.js

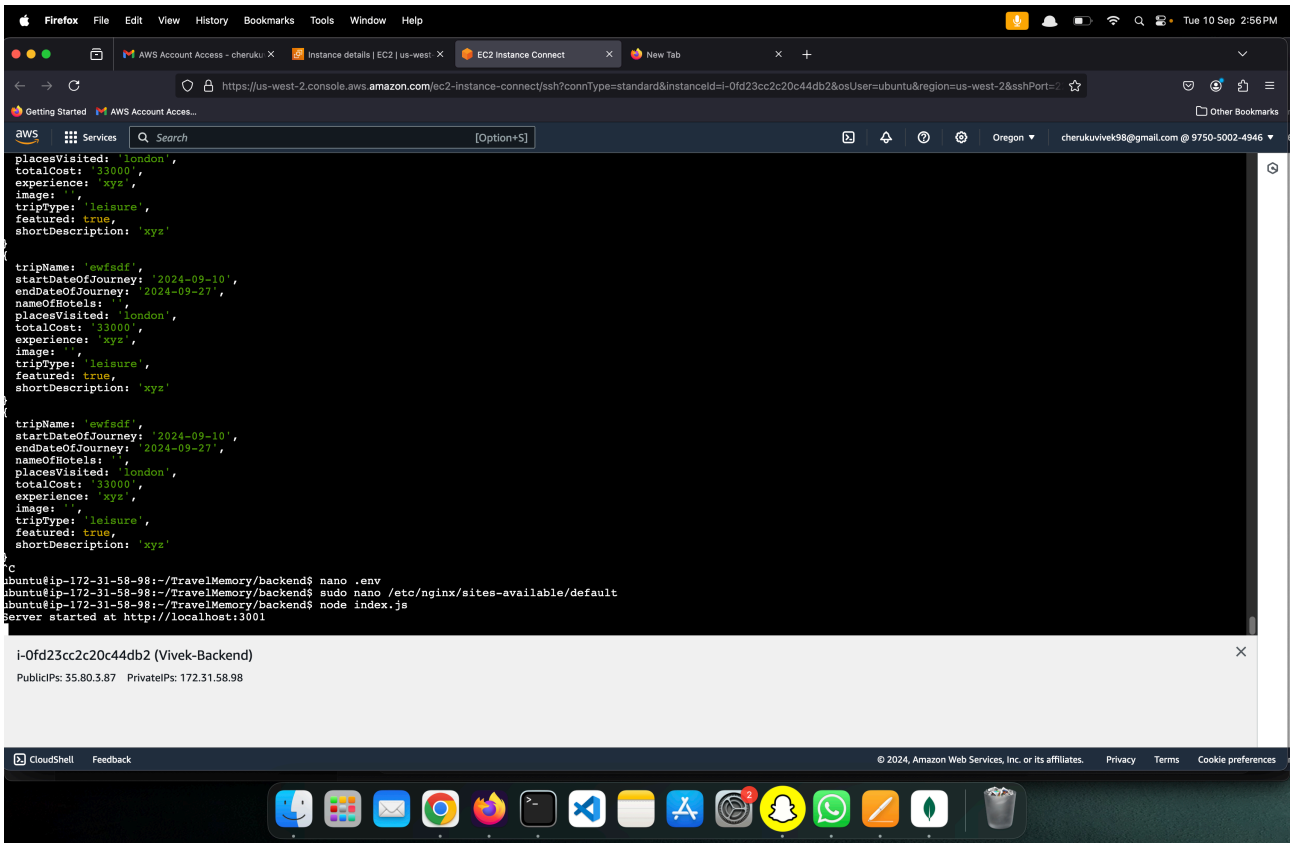
Back end would be started

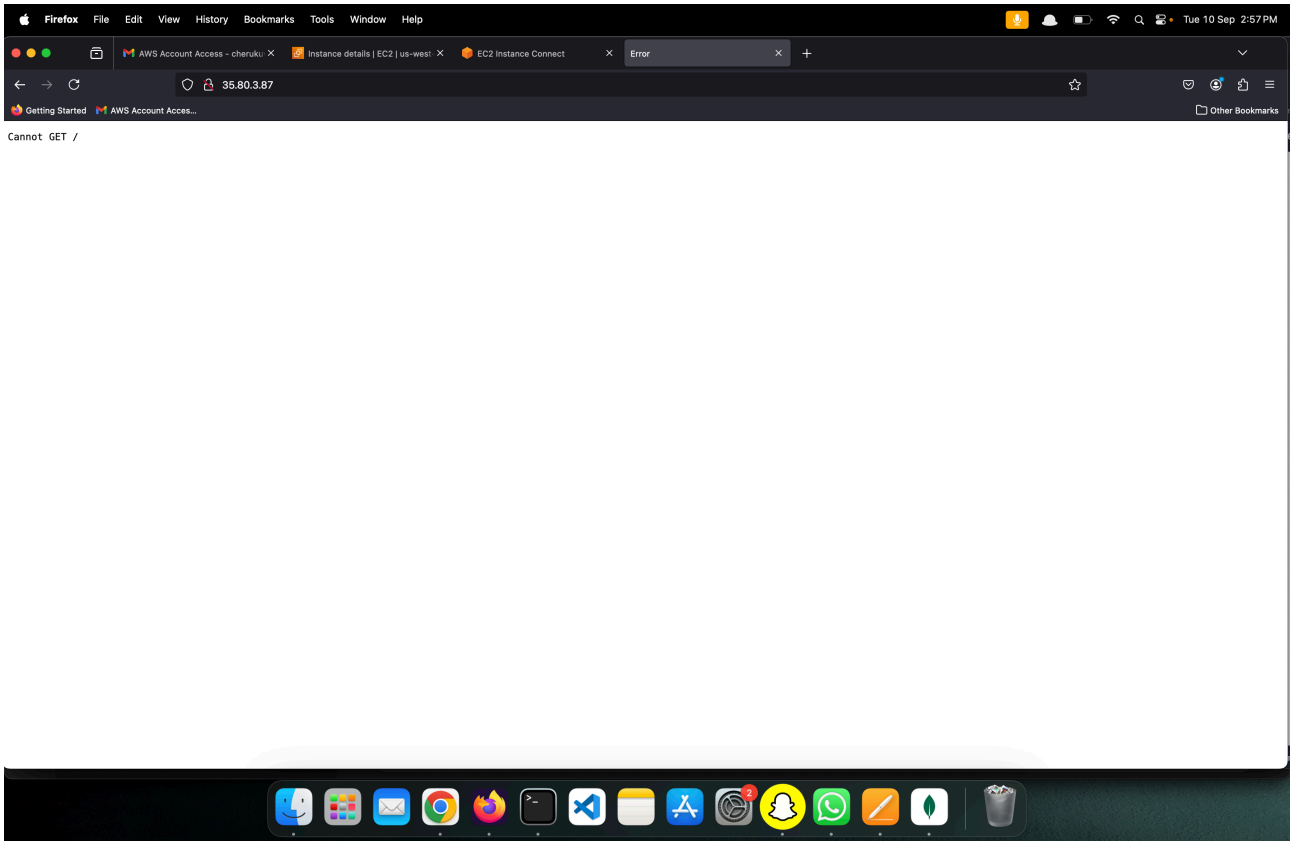
2. Frontend and Backend Connection:
- Navigate to the `urls.js` in the frontend directory.
- Update the file to ensure the front end communicates effectively with the backend.



In frontend EC2 instance proceed with the path /home/ubuntu/Travelmemory/frontend and proceed with the command sudo apt install npm for packages.
Go into the src package and update the url.js with the backend public id.







To ensure the front end communicated effectively with the backend.
Once done save and exit

Also reverse proxy for nginx

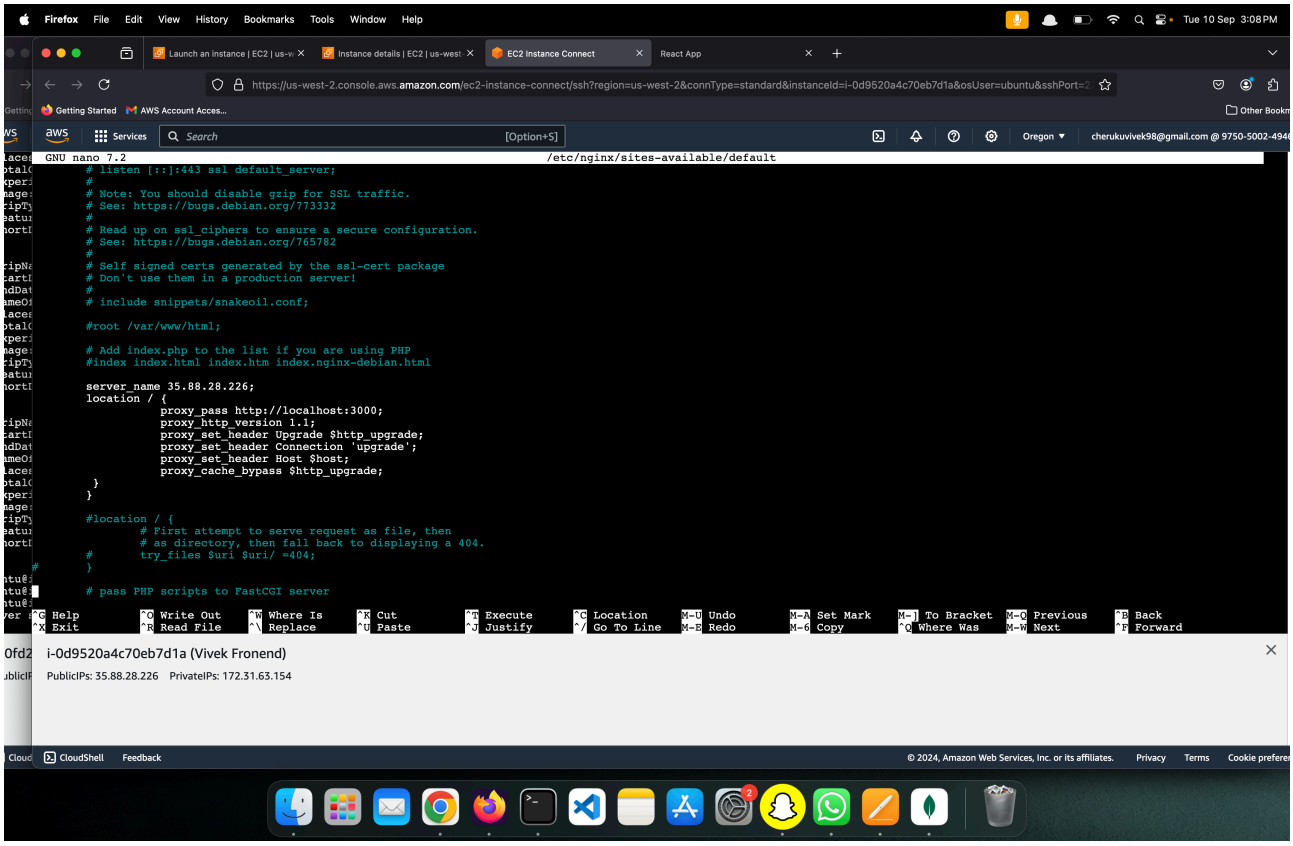
-> sudo apt install nginx
and proceed with the path

Sudo nano /etc/nginx/sites-available/default

Add the following configuration:

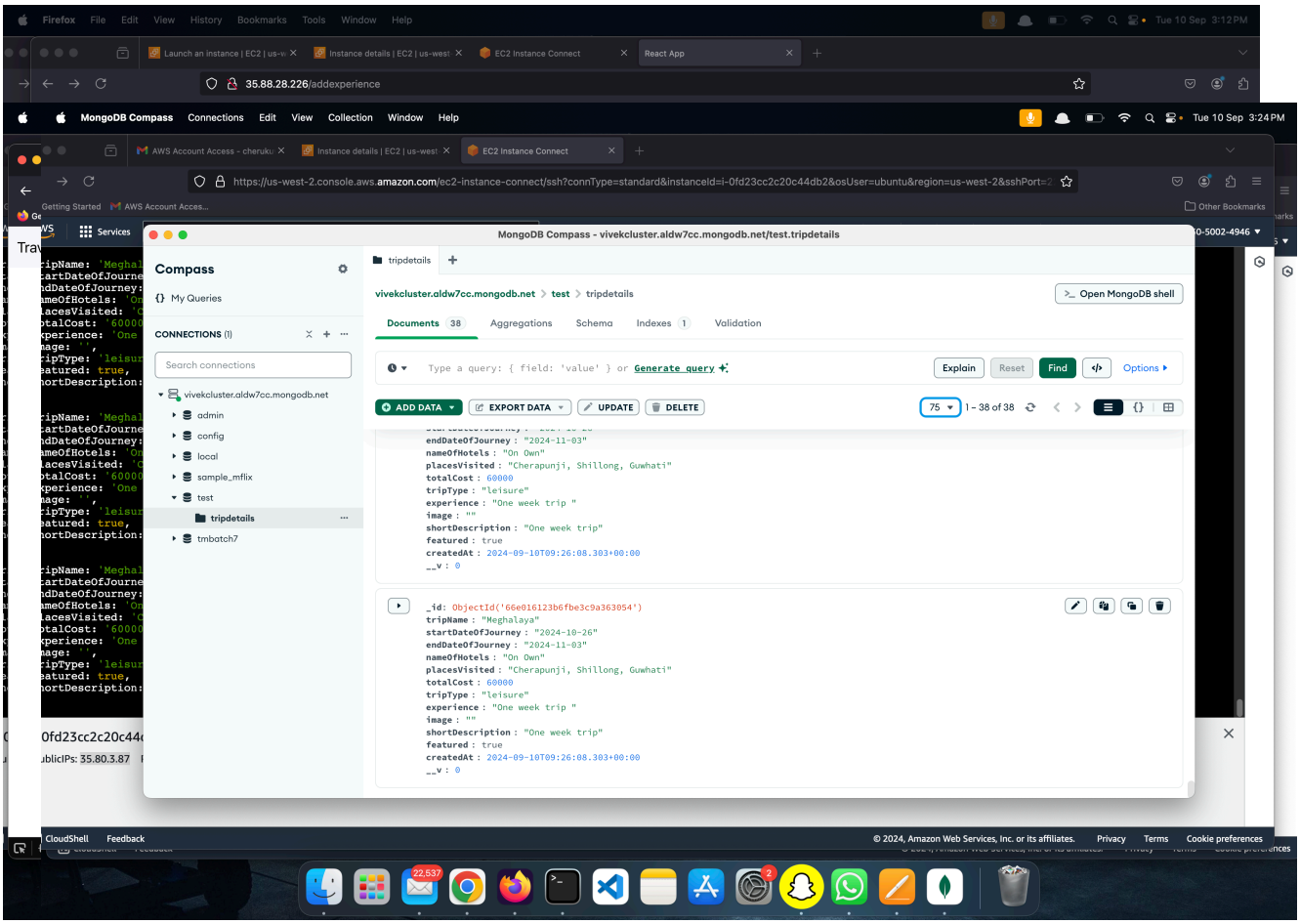
```
server {
  listen 80;
  server_name “front_end_Public ID” ;

  location / {
     proxy_pass http://localhost:3000;
     proxy_http_version 1.1;
     proxy_set_header Upgrade $http_upgrade;
     proxy_set_header Connection 'upgrade';
     proxy_set_header Host $host;
     proxy_cache_bypass $http_upgrade;
  }
```

check for the nginx test with -> sudo nginx -t if successful proceed to restart the nginx with sudo systemctl restart nginx



Come back with the frontend folder and proceed with the npm install command following with npm run build
Once done proceed with the npm start to start the front_end
The app is successfully deployed with the backend connected and the database as a mongobd
Enter the details and submit the form check that the details are saved in the data base

Once checking the EC2 and MongoDB

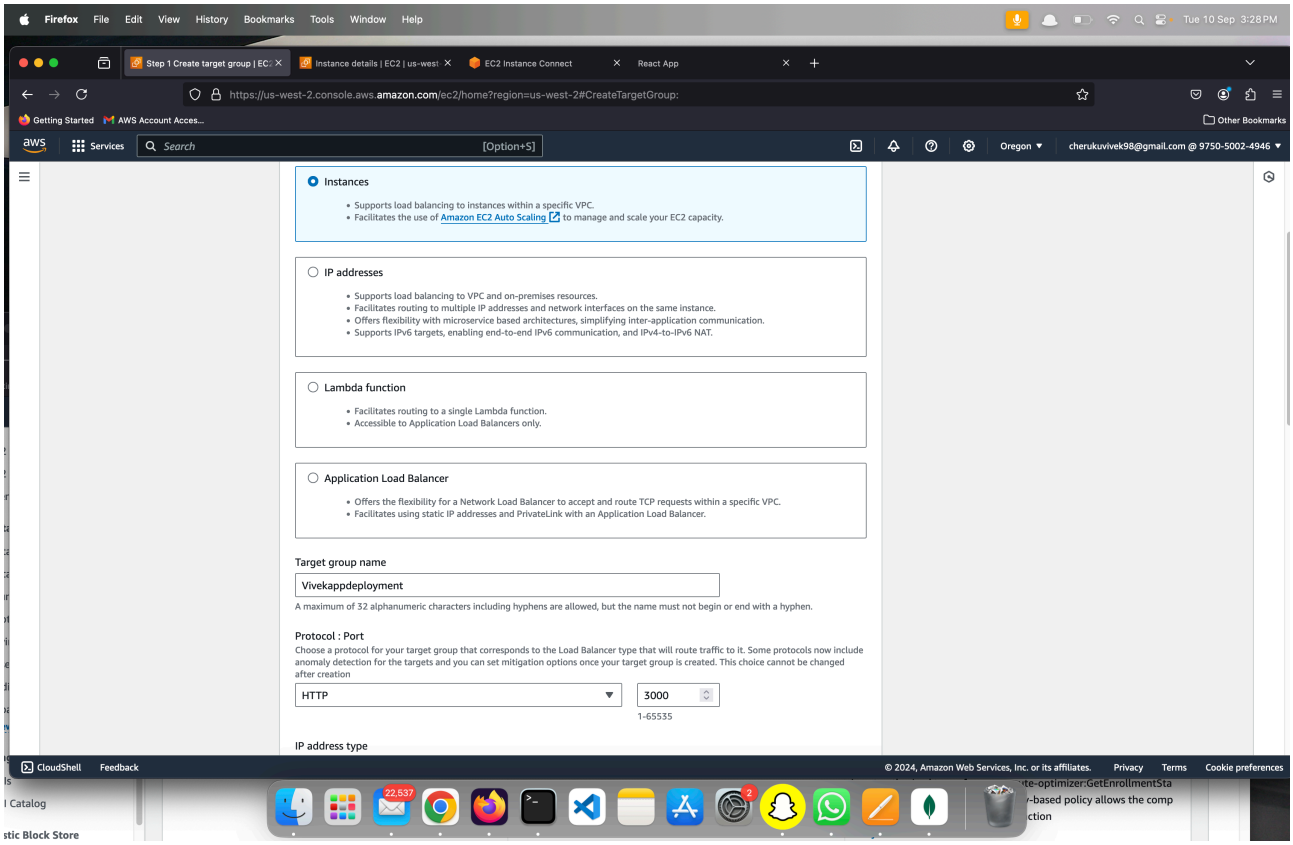


3. Scaling the Application:

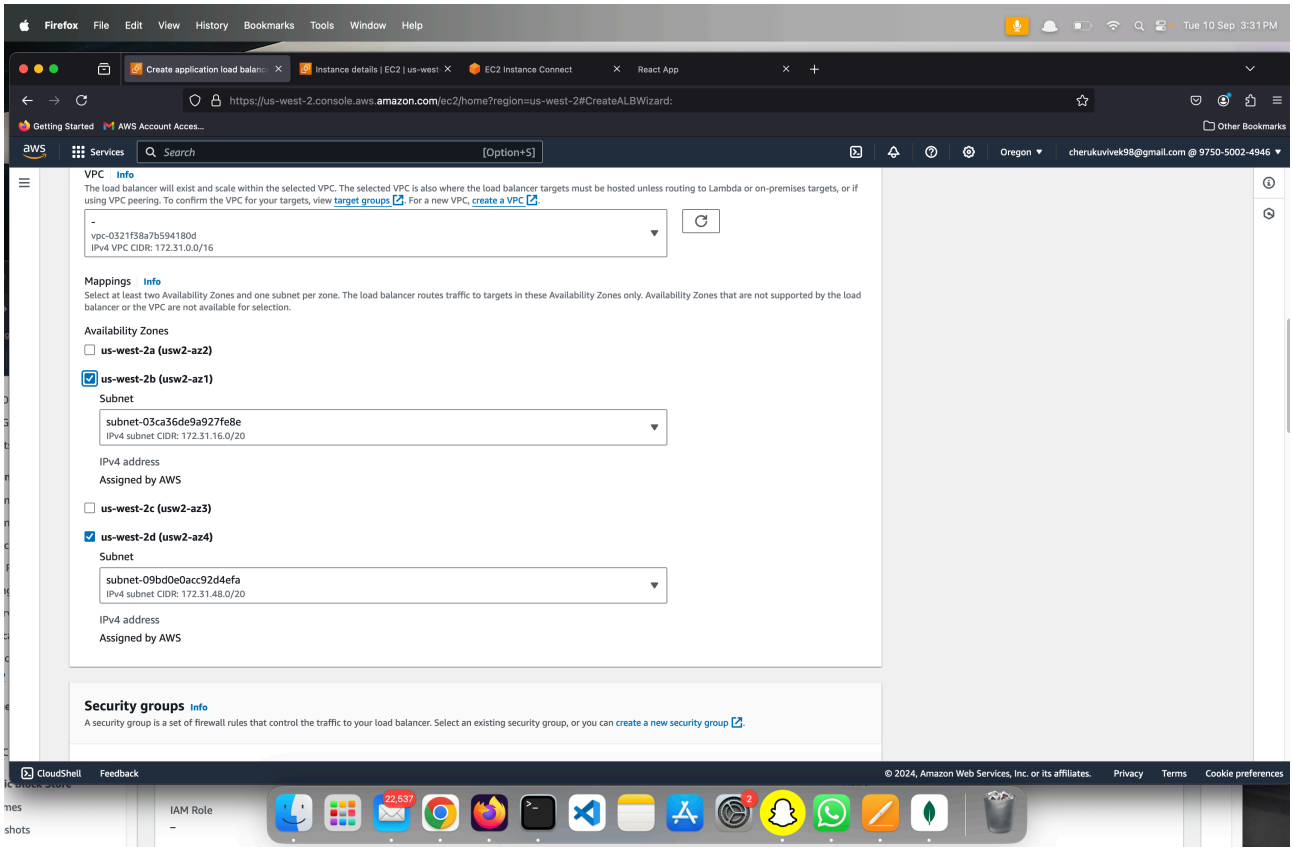
- Create multiple instances of both the frontend and backend servers.

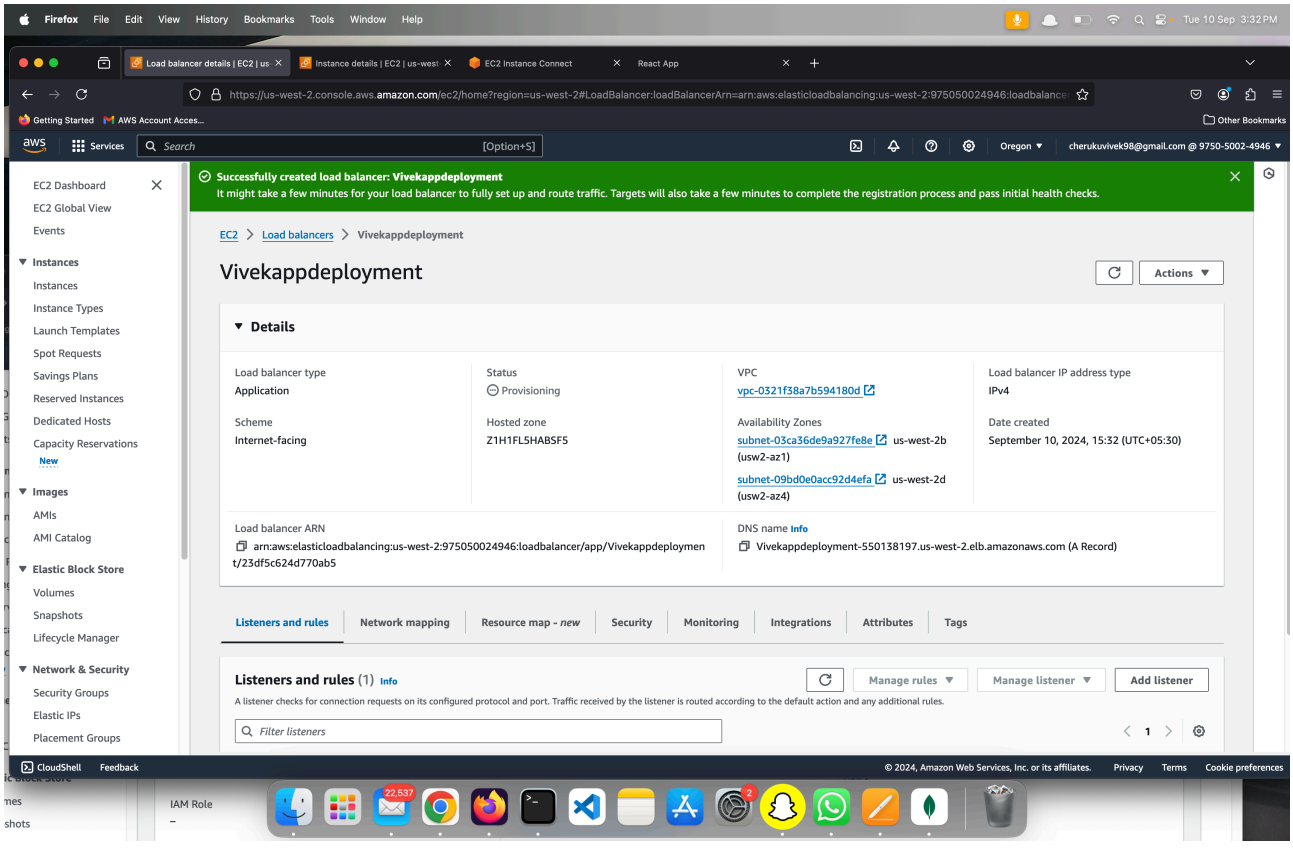
- Add these instances to a load balancer to ensure efficient distribution of incoming traffic.

Create a target group in AWS for frontend with the 3000 port HTTP

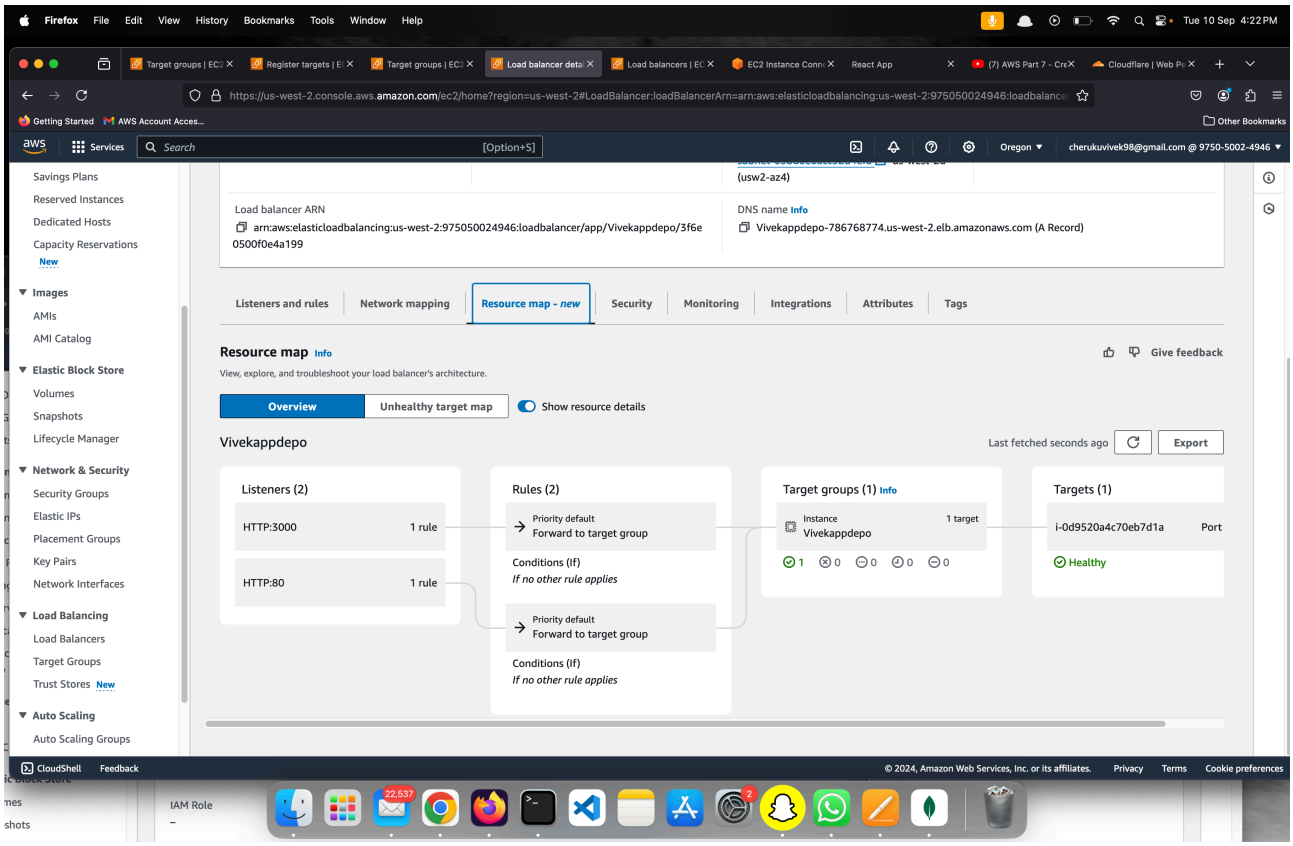


Add the available instance which is running the frontend
Now create the load balancer and add the target group which is created earlier with the instances created availability zones



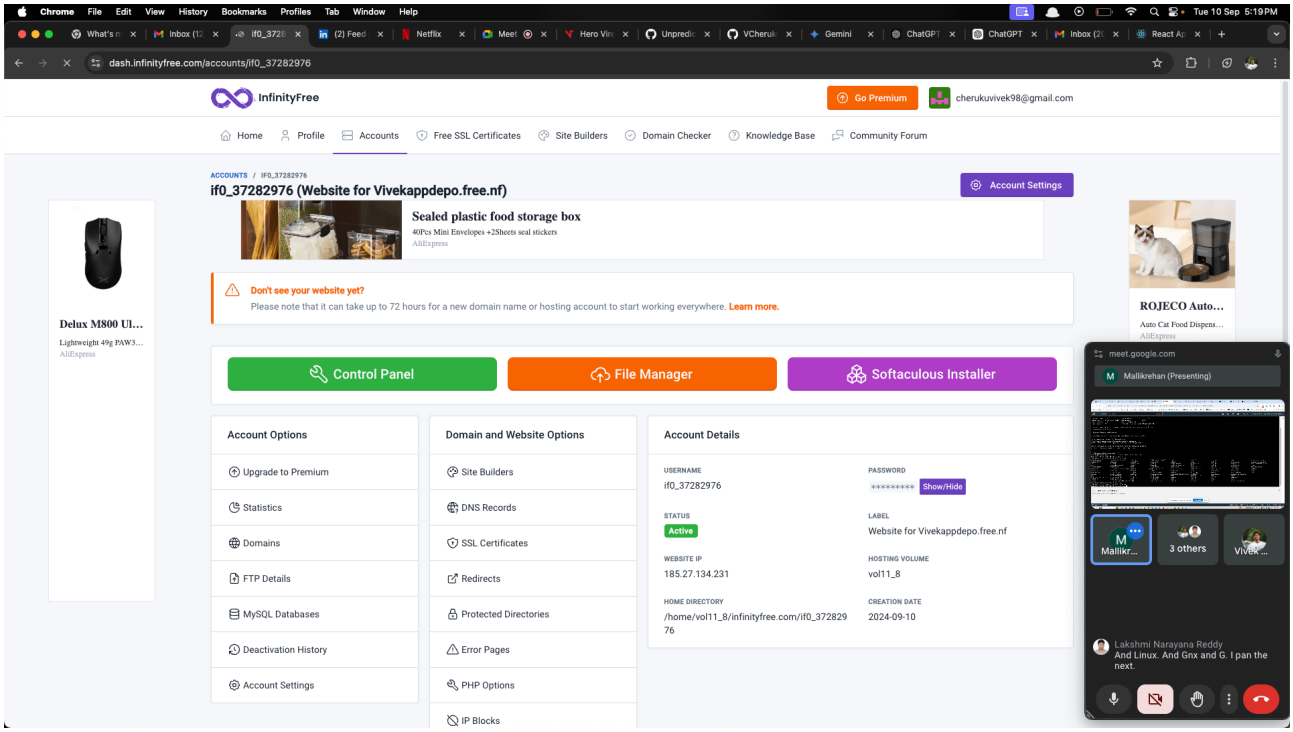


The load balance is created and the target is healthy

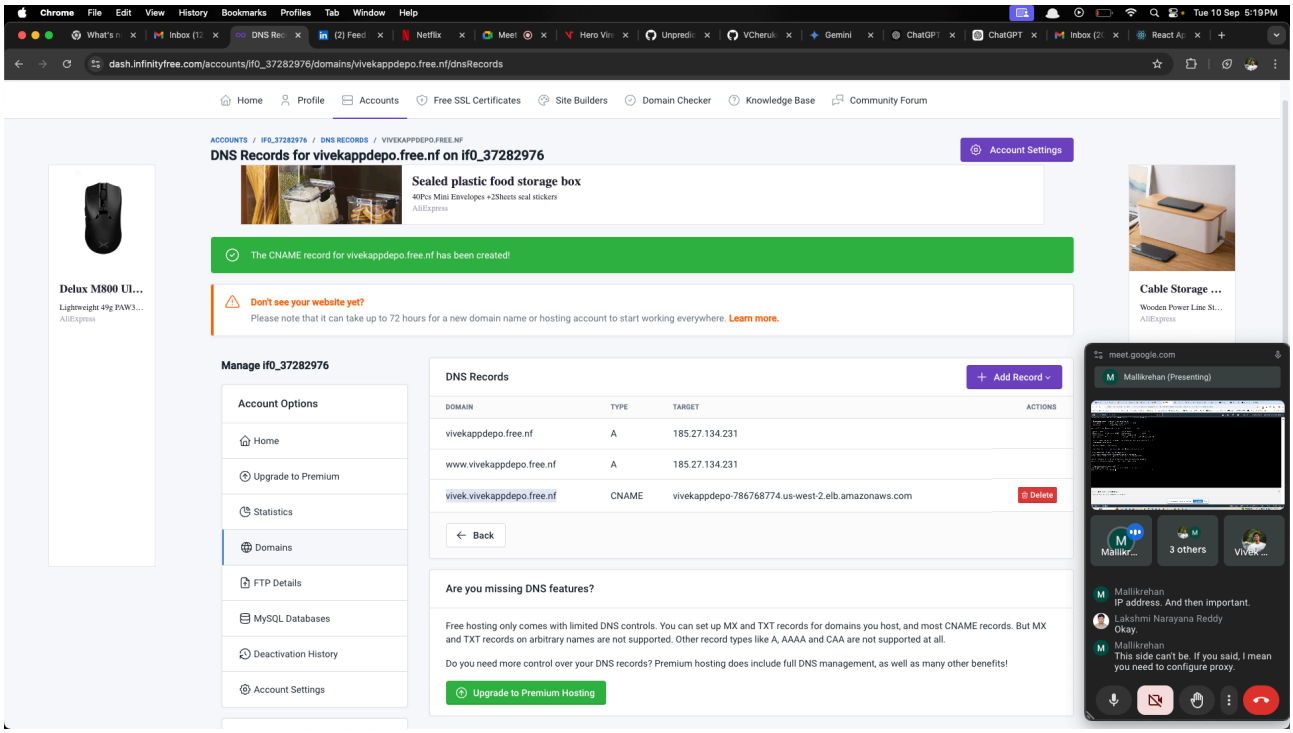


4. Domain Setup with Cloudflare:
- Connect your custom domain to the application using Cloudflare.
- Create a CNAME record pointing to the load balancer endpoint.
- Set up an A record with the IP address of the EC2 instance hosting the front end.

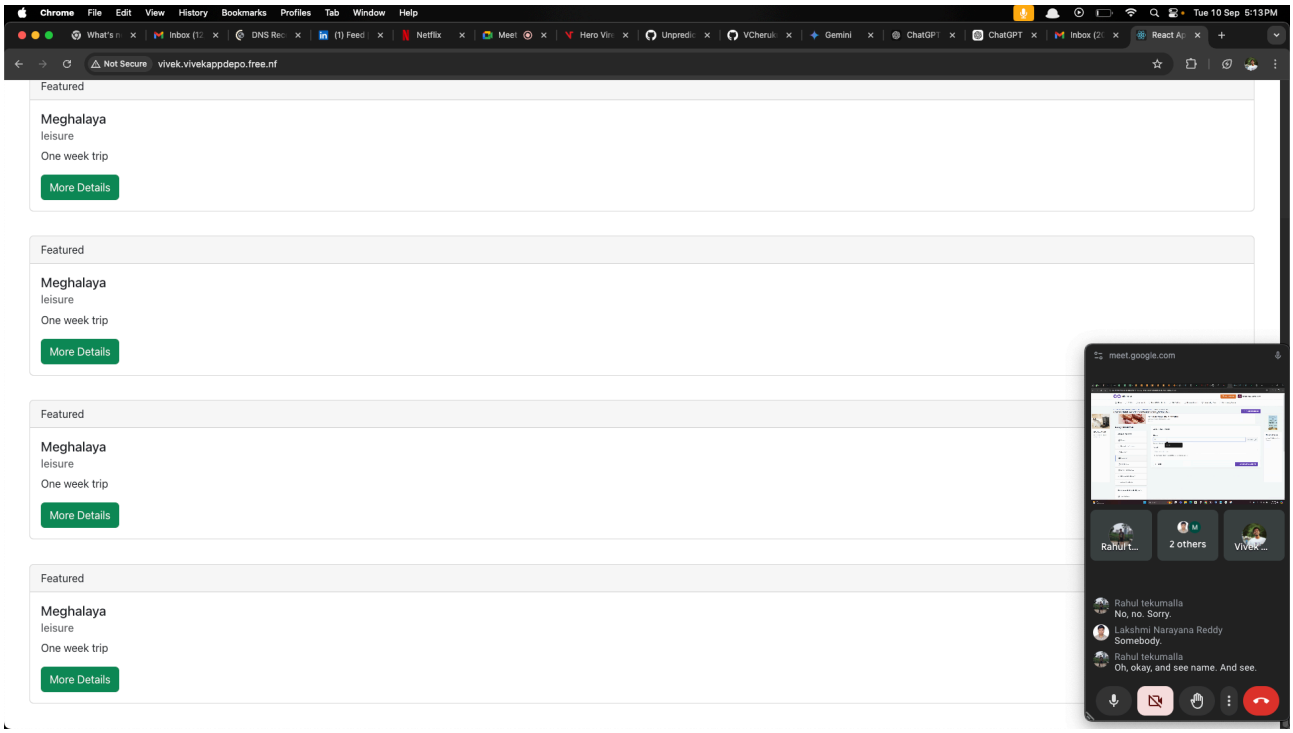
Setup a Domain. As the cloud flare is a paid one
Hence using https://dash.infinityfree.com/ to setup a domain



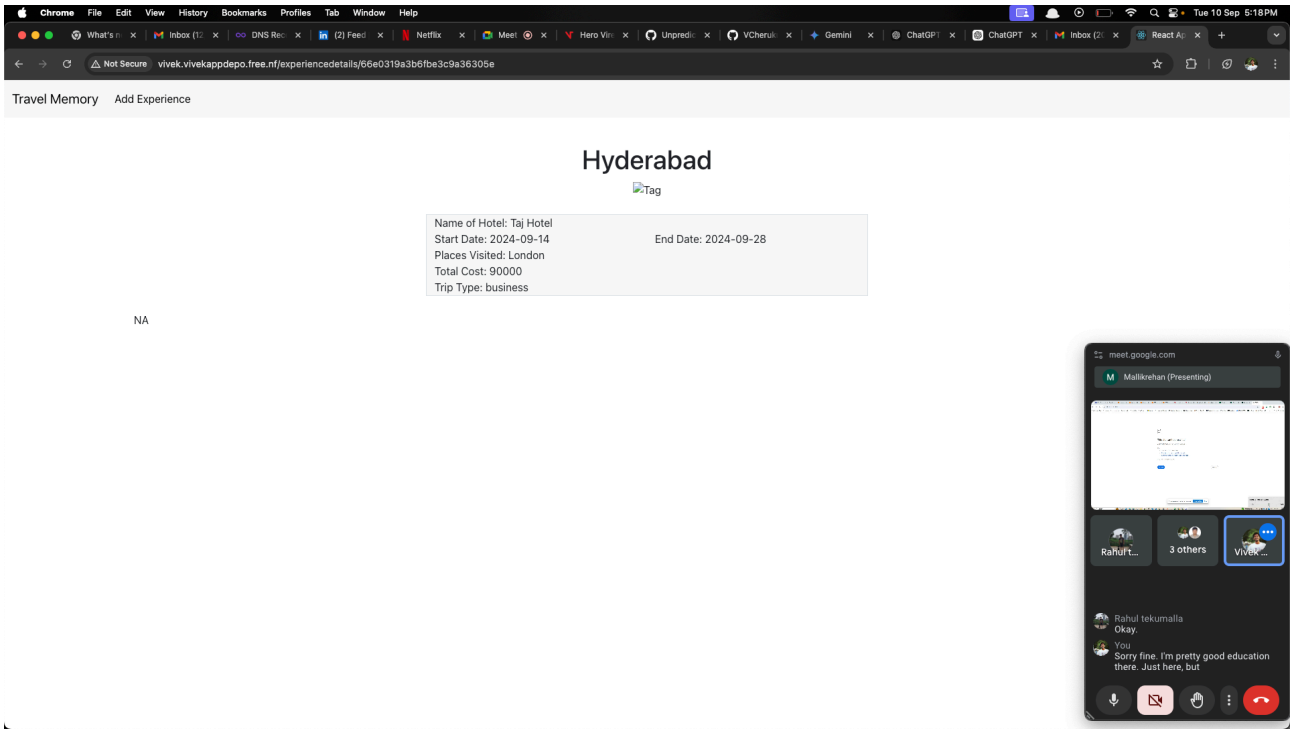
Now create a CNAME with and add the DNS name of the load balancer

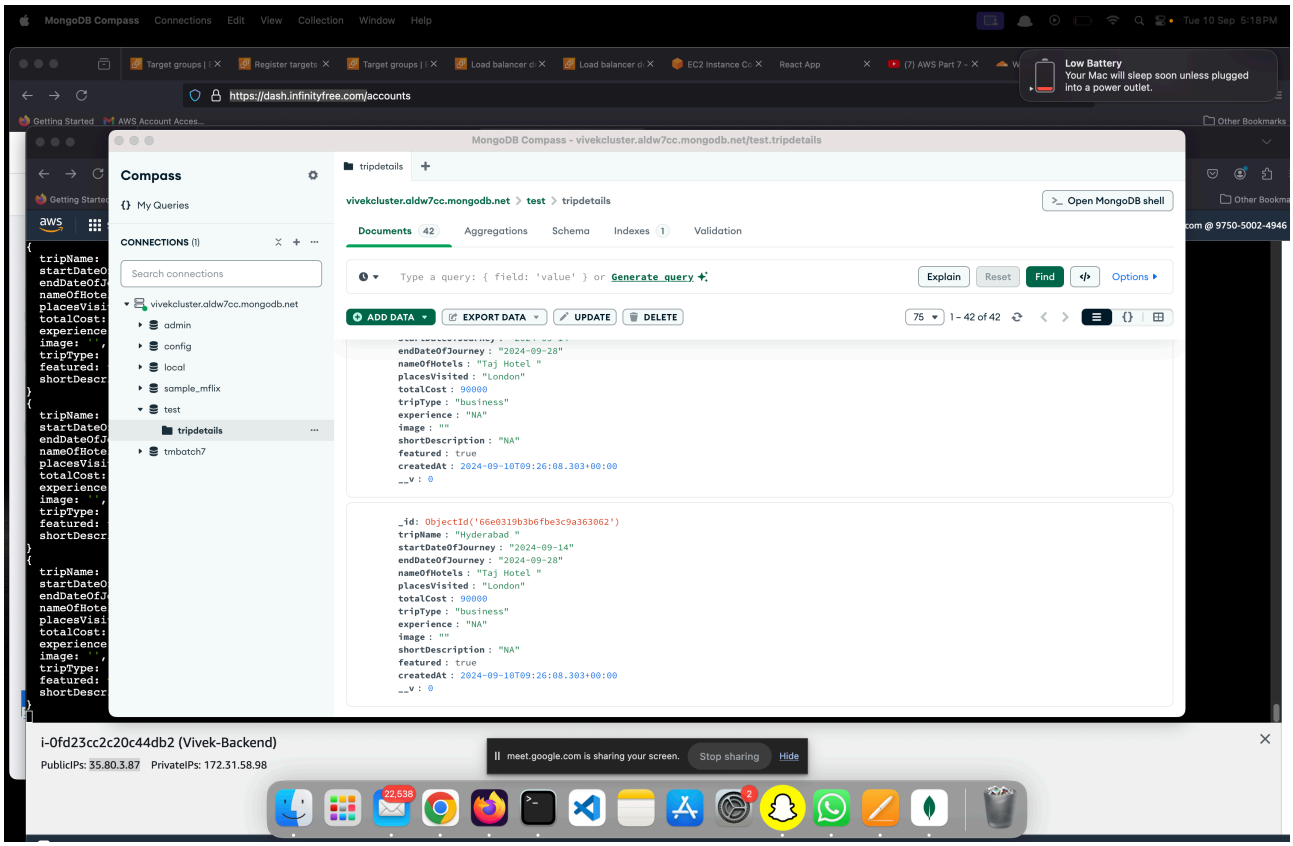


Using the DNS Records name the website will be opened



Check if the backend and frontend working fine by updating the details which should be reflected on to the frontend and mongo DB


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Not Secure vivek.vivekappdepo.free.nf/experiencedetails/66e0319a3b6fbe3c9a36305e
Travel Memory Add Experience
Hyderabad
Tag
Name of Hotel: Taj Hotel
Start Date: 2024-09-14
End Date: 2024-09-28
Places Visited: London
Total Cost: 90000
Trip Type: business
NA
meet.google.com
Mallikrehan (Presenting)
Rahul T...
3 others
Vivek ...
Rahul tekumalla
Okay.
You
Sorry fine. I'm pretty good education there. Just here, but


MongoDB Compass Connections Edit View Collection Window Help
Low Battery
Your Mac will sleep soon unless plugged into a power outlet.
https://dash.infinityfree.com/accounts
MongoDB Compass - vivekcluster.aldw7cc.mongodb.net/test.tripdetails
Compass
My Queries
CONNECTIONS (1)
Search connections
vivekcluster.aldw7cc.mongodb.net
admin
config
local
sample_mflix
test
tripdetails
tmbatch7
vivekcluster.aldw7cc.mongodb.net > test > tripdetails
Open MongoDB shell
Documents 42
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
1 - 42 of 42
endDateOfJourney : "2024-09-28"
nameOfHotels : "Taj Hotel "
placesVisited : "London"
totalCost : 90000
tripType : "business"
experience : "NA"
image : ""
shortDescription : "NA"
featured : true
createdAt : 2024-09-10T09:26:08.303+00:00
__v : 0
_id: ObjectId('66e0319b3b6fbe3c9a363062')
tripName : "Hyderabad "
startDateOfJourney : "2024-09-14"
endDateOfJourney : "2024-09-28"
nameOfHotels : "Taj Hotel "
placesVisited : "London"
totalCost : 90000
tripType : "business"
experience : "NA"
image : ""
shortDescription : "NA"
featured : true
createdAt : 2024-09-10T09:26:08.303+00:00
__v : 0
i-0fd23cc2c20c44db2 (Vivek-Backend)
PublicIPs: 35.80.3.87 PrivateIPs: 172.31.58.98
meet.google.com is sharing your screen.
Stop sharing
Hide