Roulettech Inc.

RECIPE MANAGER - APPLICATION

SIMRN GUPTA - JUNIOR SOFTWARE ENGINEER APPLICATION

Introduction

I built the Recipe Manager application to demonstrate my ability to develop and deploy a full-stack web application using React.js for the frontend and Django for the backend. The application allows users to add, view, and delete recipes. The frontend is hosted on **AWS S3**, and the backend is deployed on an **EC2** instance within a custom **VPC**. **AWS CloudFront** is used as a **CDN** for the frontend, and a **load balancer** ensures secure and efficient communication between the frontend and backend inside the VPC.

Tech Stack and Architecture

Tech Stack

- Frontend: React.js
- Backend: Django
- Database: SQLite
- Hosting:
    - Frontend: AWS S3 with AWS CloudFront for CDN
    - Backend: AWS EC2
- Infrastructure: Custom VPC with private and public subnets, Application Load Balancer
- Other Tools: SSH, OpenSSL

Architecture Diagram:

```
Browser
   |---> CloudFront (HTTPS)
         |---> S3 (React.js frontend)
         |---> ALB (HTTPS)
             |---> EC2 (Django backend in private subnet)
```

Deliverable:

Web application CDN Url : https://dqv20utk8ylp7.cloudfront.net

Website hosted on S3-bucket :
http://recipe-manage-bucket.s3-website.us-east-2.amazonaws.com

Screenshots attached at the end of the document - Go to screenshots

Github repo - recipe-manager

Frontend Implementation

Description:

I built the frontend of the application using React.js. It includes components to add, view, and delete recipes. The application uses functional components and maintains state using React hooks.

Key Components:

RecipeList: Displays a list of recipes.

RecipeForm: Form to add recipes.

RecipeDetail: Displays details of a single recipe.

Deployment:

I deployed the frontend code to AWS S3 and configured AWS CloudFront to provide a CDN for faster and more secure access.

Folder Structure:

```
src
   |---> components
         |---> AddRecipe.js
         |---> RecipeForm.js
         |---> RecipeDetail.js
         |---> {sub components}
    |---> services
         |---> api.js
    |---> styles
         |---> css files for all components
```

Backend Implementation

Description:

The backend of the application is built using Django, providing API endpoints for managing recipes. It includes basic CRUD operations (Create, Read, Update, Delete).

API Endpoints:

- **GET /api/recipes/**: Retrieves a list of all recipes.
- **POST /api/recipes/**: Adds a new recipe.
- **GET /api/recipes/<id>/**: Retrieves a specific recipe.
- **DELETE /api/recipes/<id>/**: Deletes a specific recipe.

Deployment:

The backend is deployed on an EC2 instance within a private subnet of a custom VPC. The EC2 instance runs the Django server. I used OpenSSL to generate SSL certificates for the backend EC2 instance and the load balancer to ensure secure communication with the CDN URL that is publicly available.

Folder Structure:

```
backend
    |---> backend
            |---> settings.py
            |---> urls.py
            |---> wsgi.py
    |---> recipes
            |---> apps.py
            |---> models.py
            |---> serializers.py
            |---> urls.py
            |---> views.py
```

Deployment Process

Frontend Deployment to AWS S3:

Build React Application:

- Created the build files for the react application using npm run build.

Deploy to S3:

- Upload static build files to the S3 bucket.
- Enabled static website hosting on S3 bucket.

Set up CloudFront Distribution:

- Created a CloudFront distribution pointing to the S3 bucket.
- Configured the distribution to serve the React application.

Backend Deployment to AWS EC2:

Set Up a Custom VPC:

- I created a VPC with both public and private subnets.
- I configured an internet gateway and route tables.

Launch an EC2 Instance:

- I launched an instance in the private subnet for the Django backend.

Set Up the Bastion Host:

- I launched an instance in the public subnet to act as a bastion host for SSH access to the private instance. SSH into the public instance using key-pair.pem key.

Configure Security Groups:

- I created security groups to allow necessary inbound and outbound traffic.

Deploy the Django Application:

- I SSHed into the private instance through the Bastion host and set up the Django application.

Bonus Requirements

Use AWS CloudFront for CDN:

- I configured CloudFront to distribute the frontend assets stored in S3, providing faster access and better security.

Create a Custom VPC with One Private Subnet:

- I created a custom VPC with both public and private subnets.
- I deployed the backend server in the private subnet and set up a bastion host for secure access.

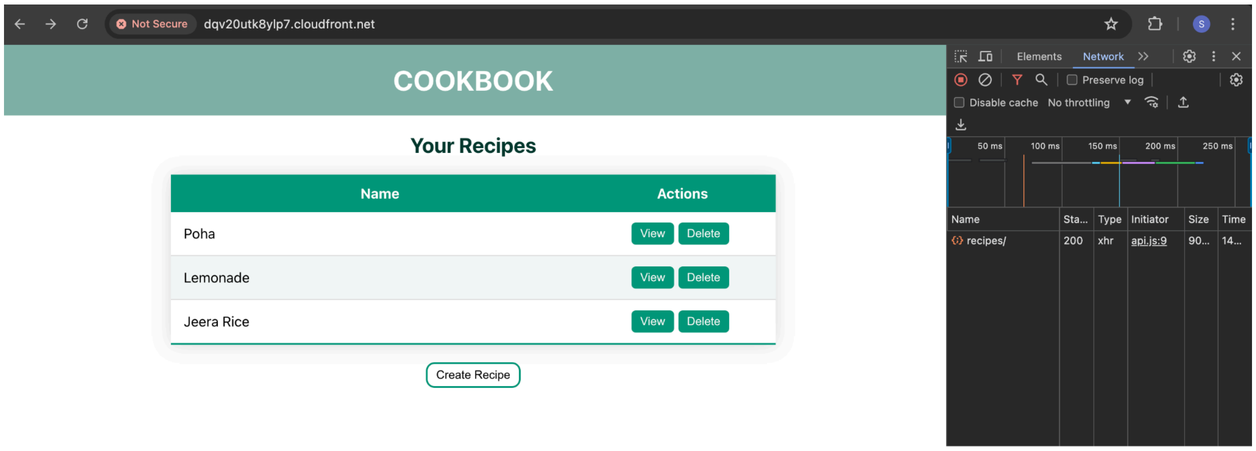
AWS Application Load Balancer:

To ensure high availability and secure communication between the frontend and backend, I used an AWS Application Load Balancer (ALB).

The ALB sits in the public subnet and handles incoming requests from the internet or CloudFront on port 443 (HTTPS).

I configured the ALB to route traffic to the target group that included my backend EC2 instance in the private subnet over HTTPS (port 443).

Screenshots of application with requests



Not Secure dqv20utk8ylp7.cloudfront.net

COOKBOOK

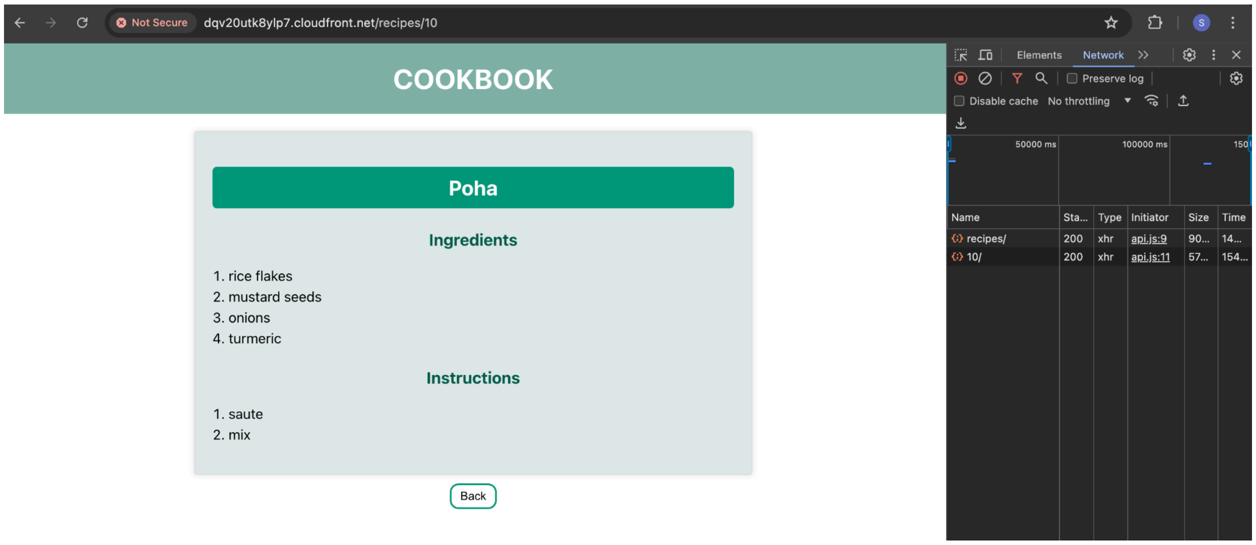
Your Recipes

Name	Actions
Poha	View Delete
Lemonade	View Delete
Jeera Rice	View Delete

Create Recipe

Name	Sta...	Type	Initiator	Size	Time
recipes/	200	xhr	api.js:9	90...	14...

View Recipe


dqv20utk8ylp7.cloudfront.net/recipes/10

COOKBOOK

Poha

Ingredients

1. rice flakes
2. mustard seeds
3. onions
4. turmeric

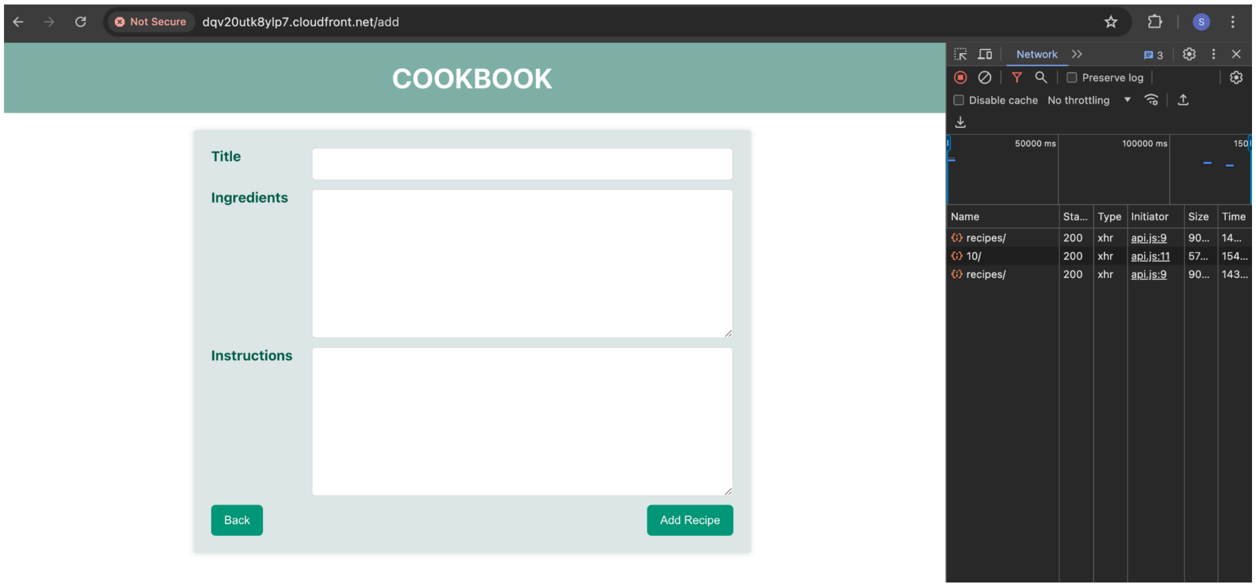
Instructions

1. saute
2. mix

Back

Name	Sta...	Type	Initiator	Size	Time
recipes/	200	xhr	api.js:9	90...	14...
10/	200	xhr	api.js:11	57...	154...

Add Recipe


dqv20utk8ylp7.cloudfront.net/add

COOKBOOK

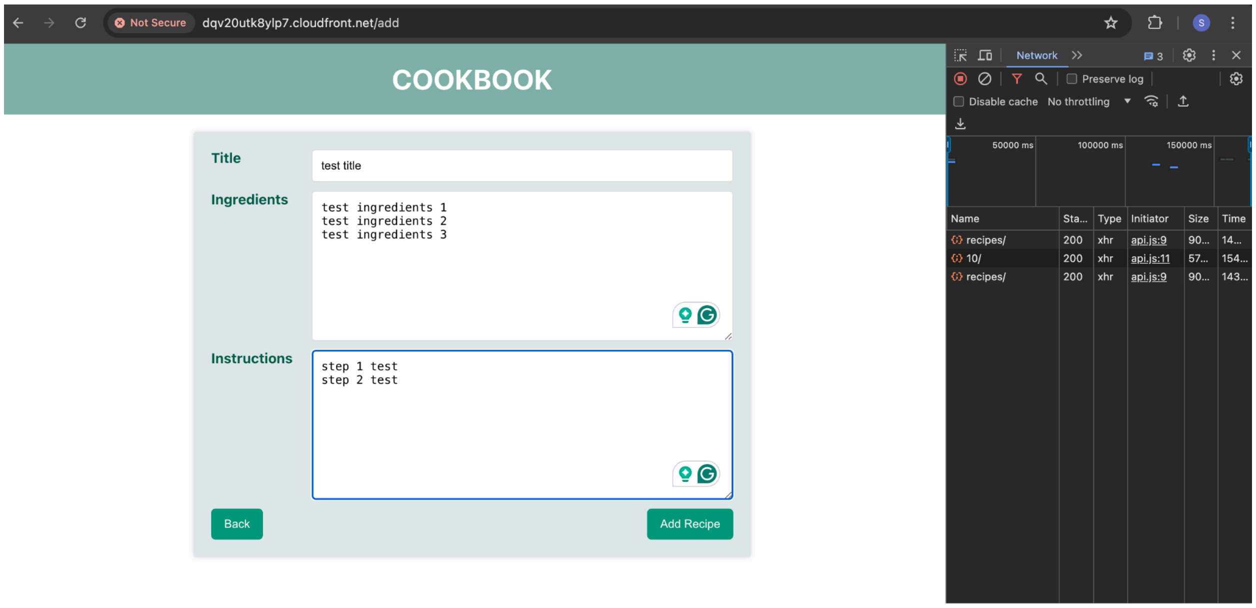
Title

Ingredients

Instructions

Back Add Recipe

Name	Sta...	Type	Initiator	Size	Time
recipes/	200	xhr	api.js:9	90...	14...
10/	200	xhr	api.js:11	57...	154...
recipes/	200	xhr	api.js:9	90...	143...


Not Secure dqv20utk8ylp7.cloudfront.net/add

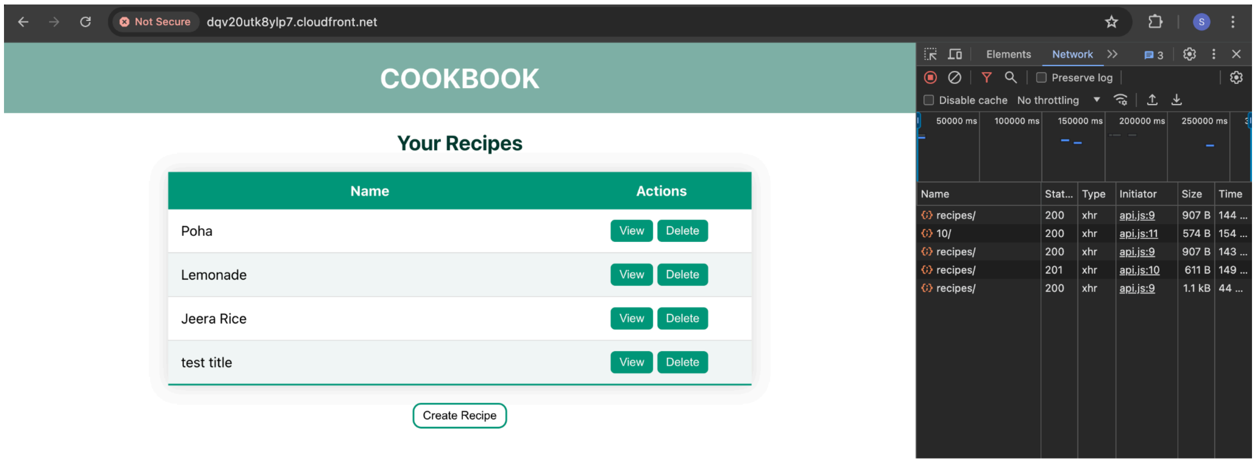
COOKBOOK

Title: test title

Ingredients:
```
test ingredients 1
test ingredients 2
test ingredients 3
```

Instructions:
```
step 1 test
step 2 test
```

Name	Sta...	Type	Initiator	Size	Time
recipes/	200	xhr	api.js:9	90...	14...
10/	200	xhr	api.js:11	57...	154...
recipes/	200	xhr	api.js:9	90...	143...


Not Secure dqv20utk8ylp7.cloudfront.net

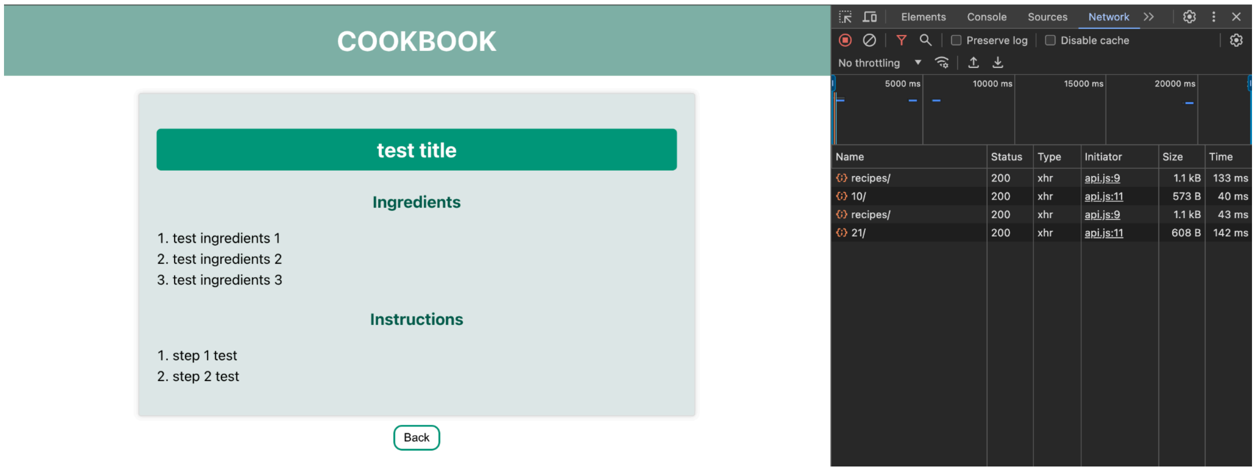
COOKBOOK

Your Recipes

Name	Actions
Poha	View Delete
Lemonade	View Delete
Jeera Rice	View Delete
test title	View Delete

Name	Stat...	Type	Initiator	Size	Time
recipes/	200	xhr	api.js:9	907 B	144 ...
10/	200	xhr	api.js:11	574 B	154 ...
recipes/	200	xhr	api.js:9	907 B	143 ...
recipes/	201	xhr	api.js:10	611 B	149 ...
recipes/	200	xhr	api.js:9	1.1 kB	44 ...

View recipe


COOKBOOK

test title

Ingredients

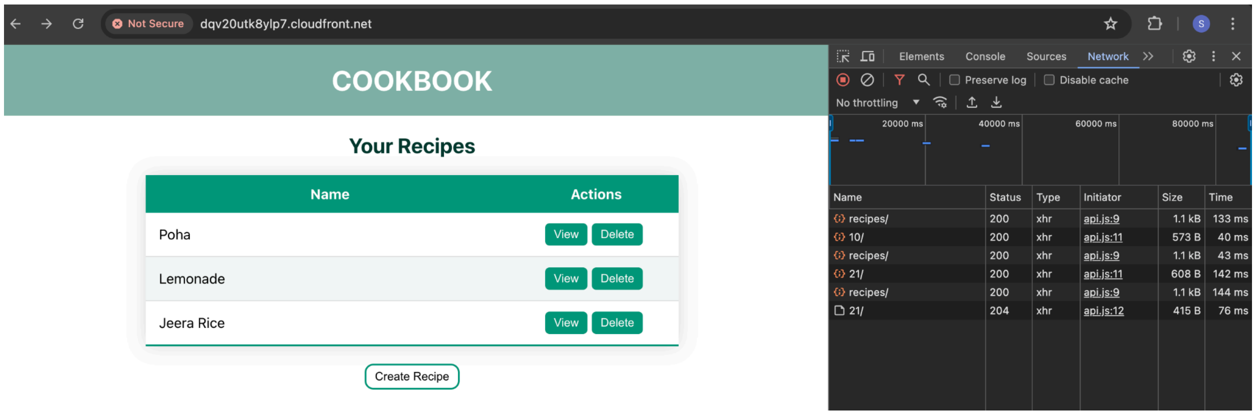
1. test ingredients 1
2. test ingredients 2
3. test ingredients 3

Instructions

1. step 1 test
2. step 2 test

Name	Status	Type	Initiator	Size	Time
recipes/	200	xhr	api.js:9	1.1 kB	133 ms
10/	200	xhr	api.js:11	573 B	40 ms
recipes/	200	xhr	api.js:9	1.1 kB	43 ms
21/	200	xhr	api.js:11	608 B	142 ms

Delete recipe


Not Secure dqv20utk8ylp7.cloudfront.net

COOKBOOK

Your Recipes

Name	Actions
Poha	View Delete
Lemonade	View Delete
Jeera Rice	View Delete

Create Recipe

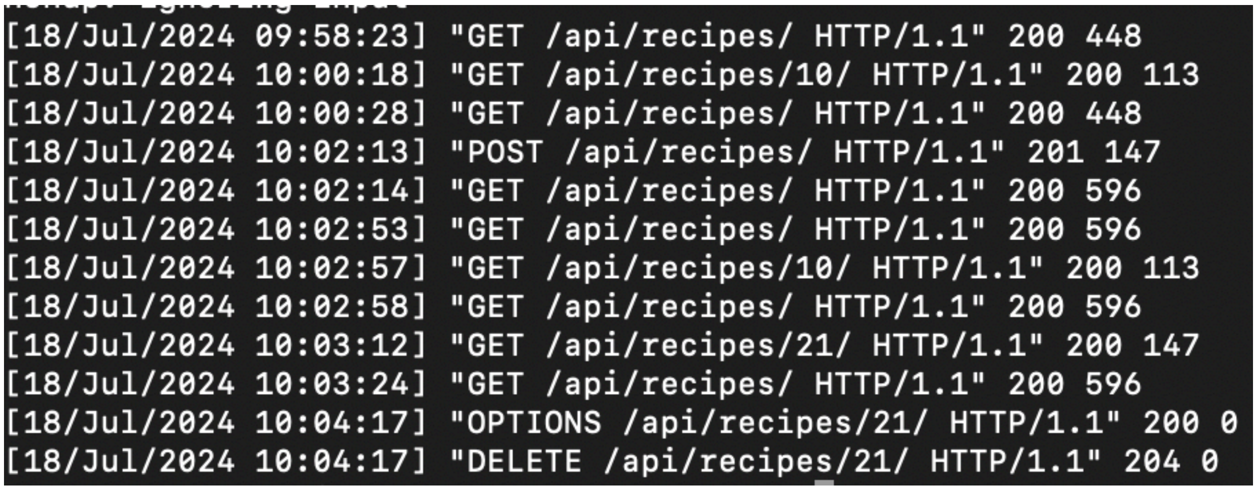
Elements Console Sources Network

Preserve log Disable cache

No throttling

Name	Status	Type	Initiator	Size	Time
recipes/	200	xhr	api.js:9	1.1 kB	133 ms
10/	200	xhr	api.js:11	573 B	40 ms
recipes/	200	xhr	api.js:9	1.1 kB	43 ms
21/	200	xhr	api.js:11	608 B	142 ms
recipes/	200	xhr	api.js:9	1.1 kB	144 ms
21/	204	xhr	api.js:12	415 B	76 ms

Server log for this session


```
[18/Jul/2024 09:58:23] "GET /api/recipes/ HTTP/1.1" 200 448
[18/Jul/2024 10:00:18] "GET /api/recipes/10/ HTTP/1.1" 200 113
[18/Jul/2024 10:00:28] "GET /api/recipes/ HTTP/1.1" 200 448
[18/Jul/2024 10:02:13] "POST /api/recipes/ HTTP/1.1" 201 147
[18/Jul/2024 10:02:14] "GET /api/recipes/ HTTP/1.1" 200 596
[18/Jul/2024 10:02:53] "GET /api/recipes/ HTTP/1.1" 200 596
[18/Jul/2024 10:02:57] "GET /api/recipes/10/ HTTP/1.1" 200 113
[18/Jul/2024 10:02:58] "GET /api/recipes/ HTTP/1.1" 200 596
[18/Jul/2024 10:03:12] "GET /api/recipes/21/ HTTP/1.1" 200 147
[18/Jul/2024 10:03:24] "GET /api/recipes/ HTTP/1.1" 200 596
[18/Jul/2024 10:04:17] "OPTIONS /api/recipes/21/ HTTP/1.1" 200 0
[18/Jul/2024 10:04:17] "DELETE /api/recipes/21/ HTTP/1.1" 204 0
```

For any questions, you can reach me at :

Linkedin - https://www.linkedin.com/in/simrn-gupta/