ReactDjangoApp

- By Likhith Seera

Link to Project: https://github.com/likhithseera/ReactDjangoApp

Overview

This project consists of a simple website built with React.js for the frontend and Django for the backend.

Frontend

- React.js: Displays a list of items fetched from the Django backend.

Backend

- Django: Provides two API endpoints to fetch and manage items.

Setup Instructions

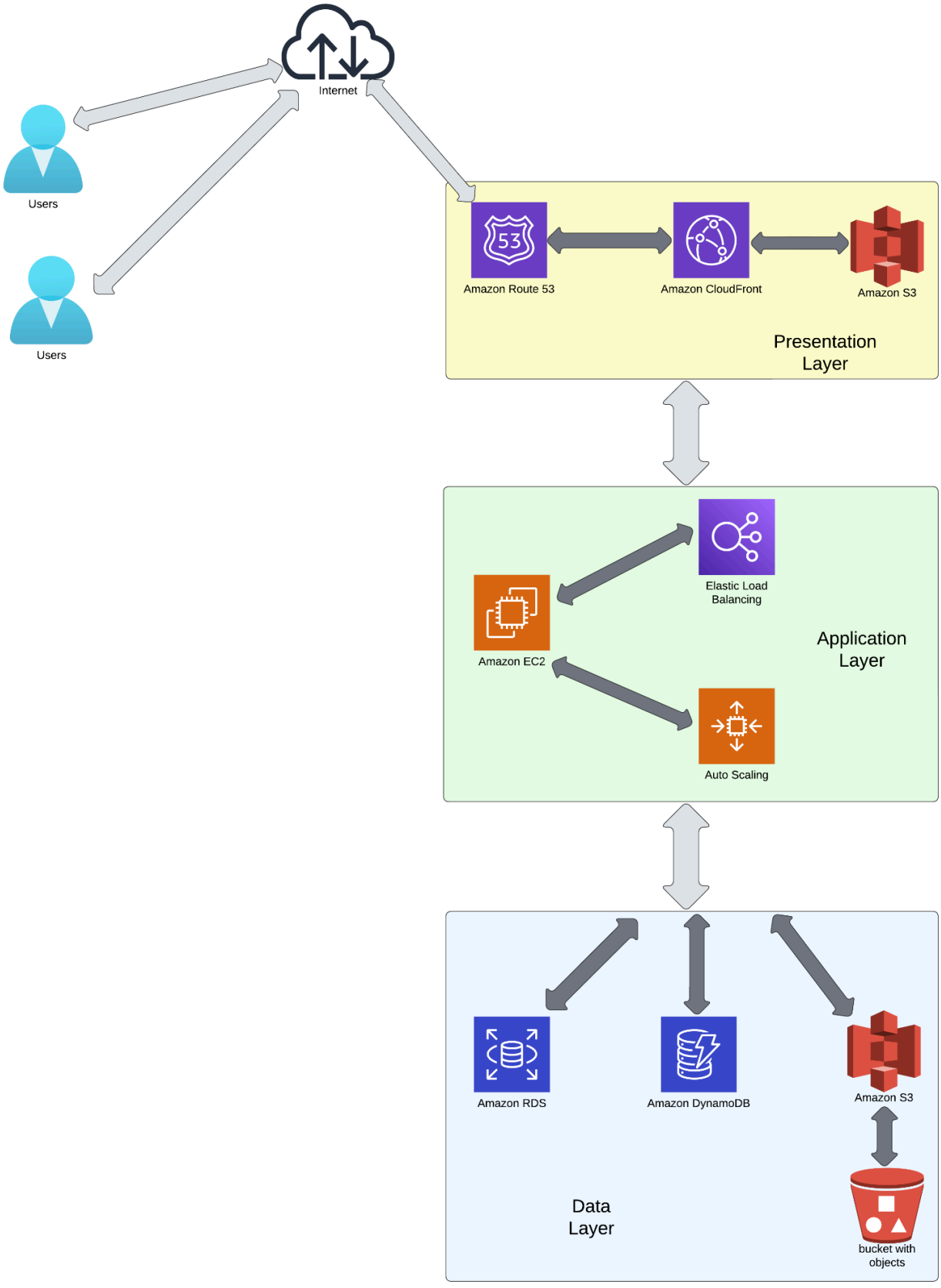
Frontend:

1. Navigate to the 'myfrontend' directory.
2. Run 'npm install' to install dependencies.
3. Run 'npm start' to start the development server.

Backend:

1. Navigate to the ‘myproject’ directory.
2. Run ‘pip install -r requirements.txt’ to install dependencies.
2. Run 'python manage.py runserver' to start the Django development server.

We need to be ensure that the backend server is running on 'http://localhost:8000' for the frontend to properly fetch data which runs on 'http://localhost:3000'.



3-tier AWS architecture diagram based on the website

Description

The **ReactDjangoApp** uses a three-tier AWS architecture with Presentation, Application, and Data Layers. Users interact with the website by sending and receiving HTTP requests and responses over the internet.

In the **Presentation Layer**, there are three main components: **Amazon Route 53, Amazon CloudFront, and Amazon S3.** Amazon Route 53 manages DNS records to direct traffic to Amazon CloudFront, which caches and delivers content closer to users, reducing latency. Amazon S3 hosts the static website and stores assets like images, scripts, and stylesheets.

The **Application layer** consists of three key components: **Amazon EC2, Amazon ELB, and Amazon Auto Scaling**. Amazon EC2 hosts the Django application, managing the core functions and servers of the website. Amazon ELB distributes incoming traffic across multiple EC2 instances to ensure the site remains reliable and available. Amazon Auto Scaling adjusts the number of EC2 instances based on traffic demands, allowing the application to handle varying levels of visitors efficiently.

The **Data layer** generally consists of databases for storing data. Here, **Amazon RDS** is used for managing relational database services, while **Amazon DynamoDB** is used for handling NoSQL database services, whereas **Amazon S3** stores backups, logs, and large amounts of unstructured data.