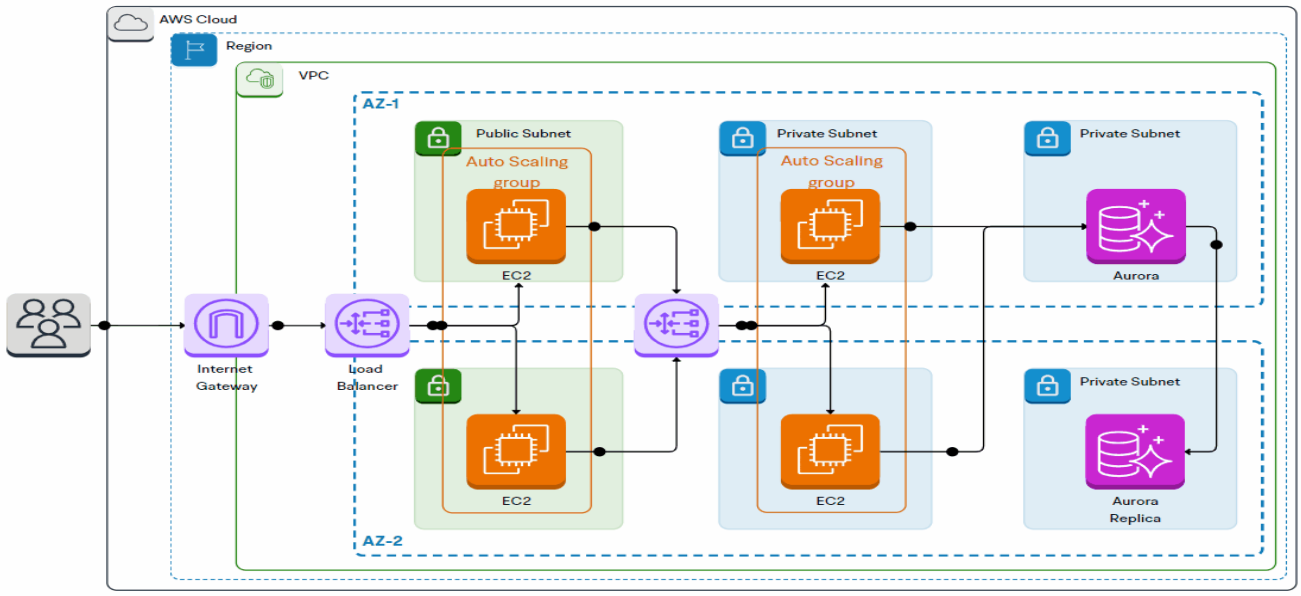
Nagireddy Hari Krishna Reddy

AWS Three Tier Web Architecture



An AWS three-tier architecture is a popular cloud architecture model that separates an application into three logical layers: the presentation tier, the application tier, and the data tier. This structure improves the scalability, reliability, and security of applications.

Architecture Overview

Presentation Tier

The front-end of the application that handles user interactions.

Application Tier

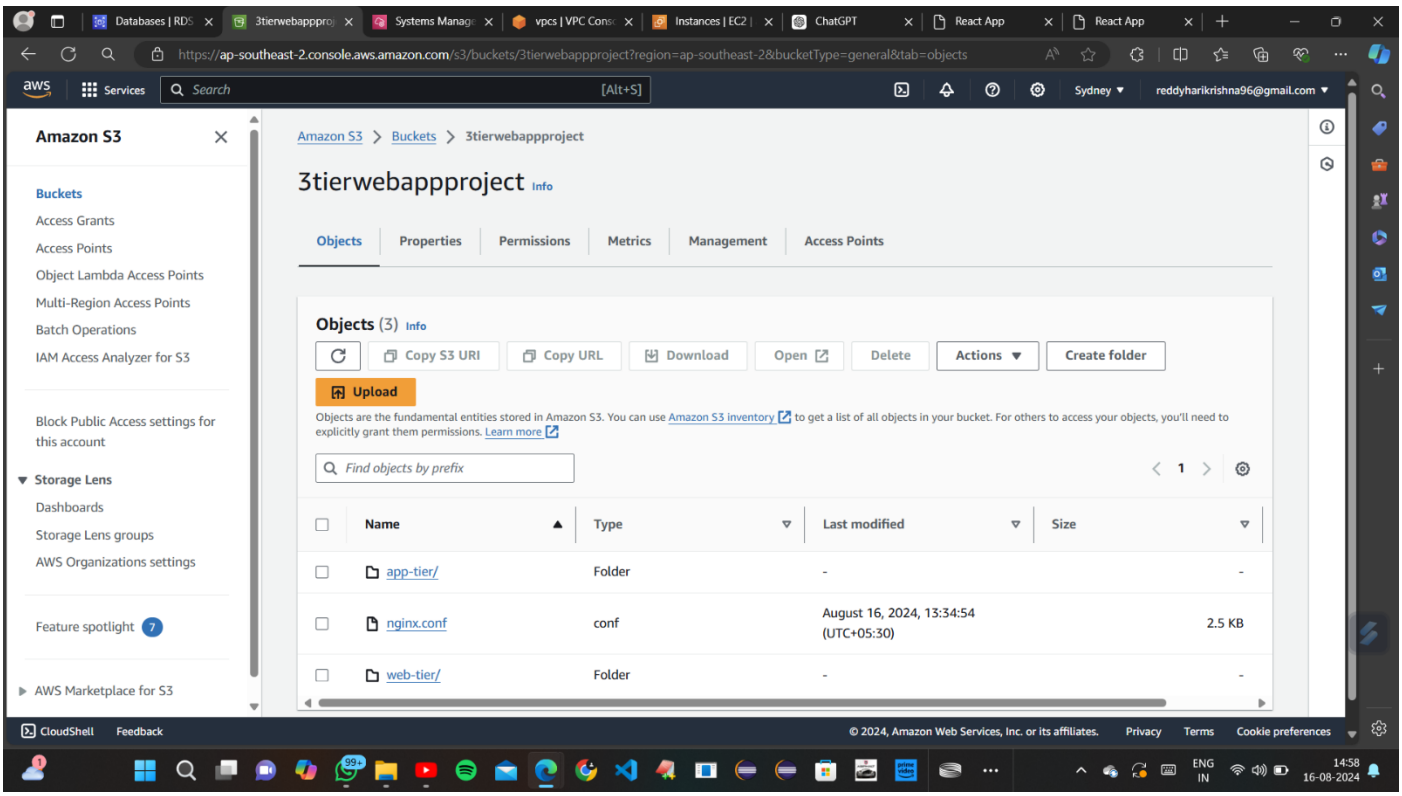
Also known as the business logic tier, this layer processes user requests and interacts with the data tier.

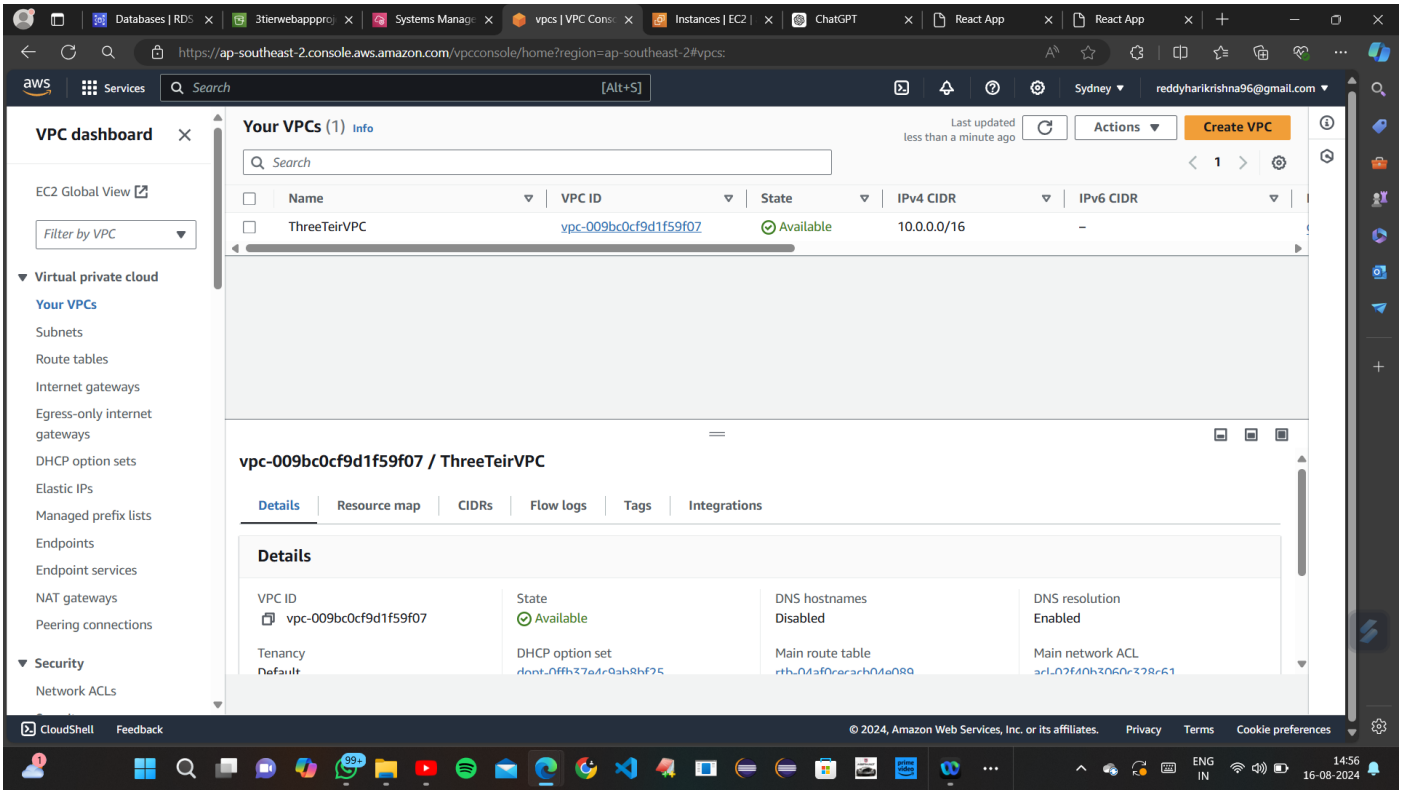
Data Tier

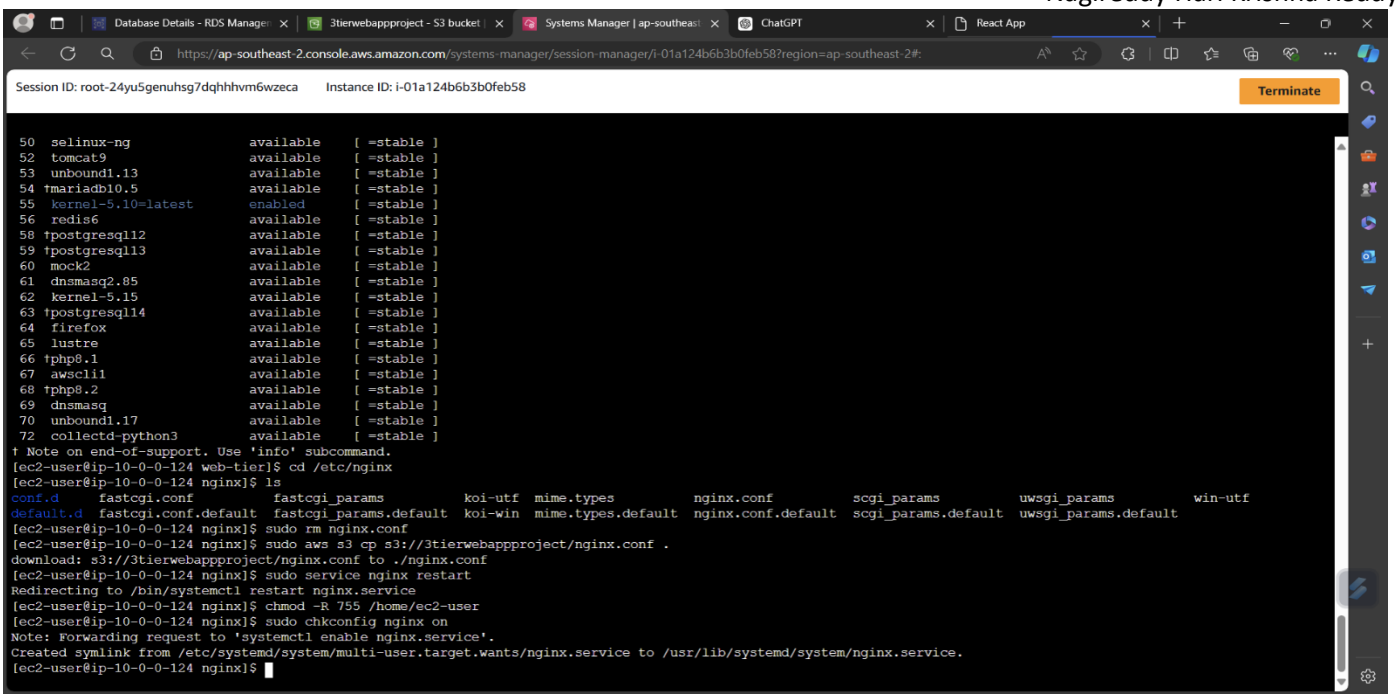
The backend of the application that stores and manages all critical information.

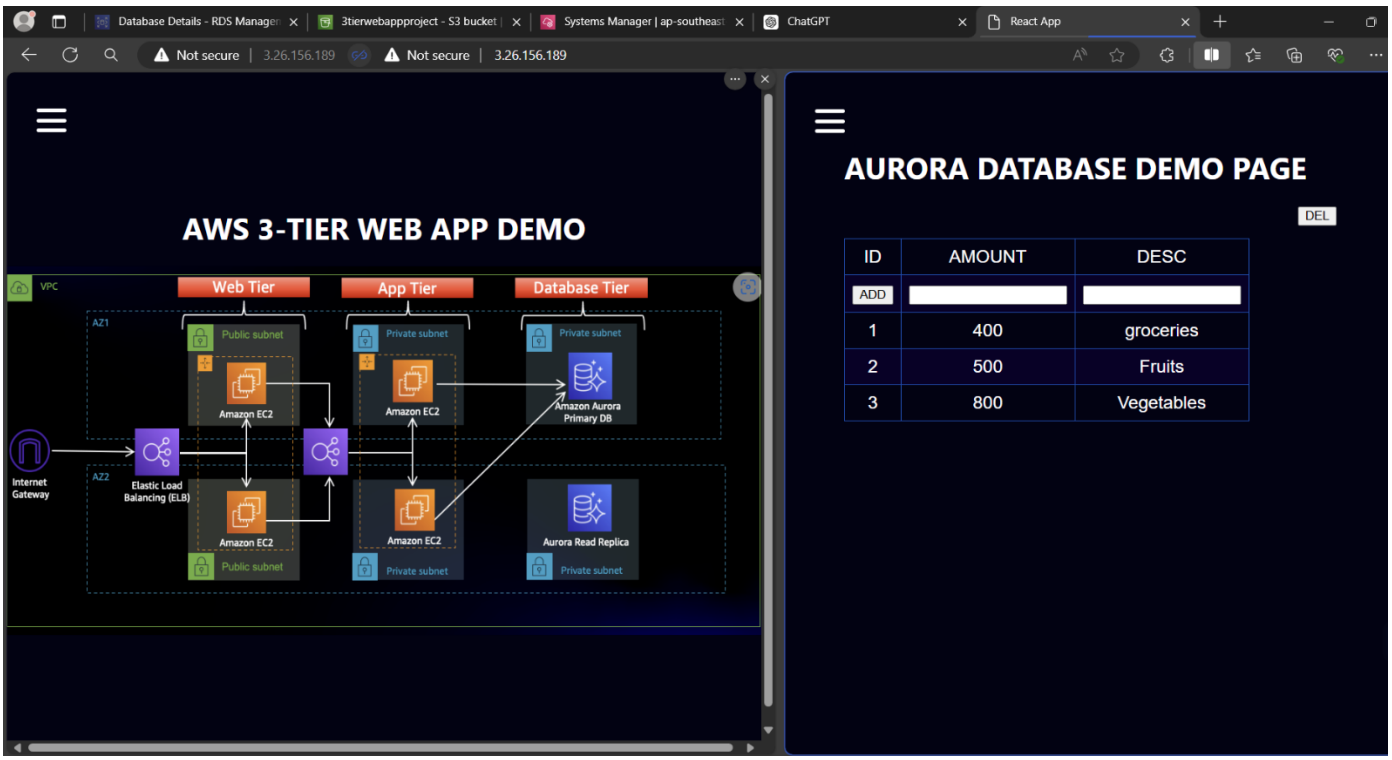
Nagireddy Hari Krishna Reddy

Steps for implementation processes






Database Details - RDS Manager
3tierwebappproject - S3 bucket
Systems Manager | ap-southeast
ChatGPT
React App
https://ap-southeast-2.console.aws.amazon.com/systems-manager/session-manager/i-01a124b6b3b0feb58?region=ap-southeast-2#:
Session ID: root-24yu5genuhsg7dqhhhvm6wzeca
Instance ID: i-01a124b6b3b0feb58
Terminate
50 selinux-ng available [=stable]
52 tomcat9 available [=stable]
53 unbound1.13 available [=stable]
54 †mariadb10.5 available [=stable]
55 kernel-5.10=latest enabled [=stable]
56 redis6 available [=stable]
58 †postgresql12 available [=stable]
59 †postgresql13 available [=stable]
60 mock2 available [=stable]
61 dnsmasq2.85 available [=stable]
62 kernel-5.15 available [=stable]
63 †postgresql14 available [=stable]
64 firefox available [=stable]
65 lustre available [=stable]
66 †php8.1 available [=stable]
67 awscli1 available [=stable]
68 †php8.2 available [=stable]
69 dnsmasq available [=stable]
70 unbound1.17 available [=stable]
72 collectd-python3 available [=stable]
† Note on end-of-support. Use 'info' subcommand.
[ec2-user@ip-10-0-0-124 web-tier]$ cd /etc/nginx
[ec2-user@ip-10-0-0-124 nginx]$ ls
conf.d fastcgi.conf fastcgi_params koi-utf mime.types nginx.conf scgi_params uwsgi_params win-utf
default.d fastcgi.conf.default fastcgi_params.default koi-win mime.types.default nginx.conf.default scgi_params.default uwsgi_params.default
[ec2-user@ip-10-0-0-124 nginx]$ sudo rm nginx.conf
[ec2-user@ip-10-0-0-124 nginx]$ sudo aws s3 cp s3://3tierwebappproject/nginx.conf .
download: s3://3tierwebappproject/nginx.conf to ./nginx.conf
[ec2-user@ip-10-0-0-124 nginx]$ sudo service nginx restart
Redirecting to /bin/systemctl restart nginx.service
[ec2-user@ip-10-0-0-124 nginx]$ chmod -R 755 /home/ec2-user
[ec2-user@ip-10-0-0-124 nginx]$ sudo chkconfig nginx on
Note: Forwarding request to 'systemctl enable nginx.service'.
Created symlink from /etc/systemd/system/multi-user.target.wants/nginx.service to /usr/lib/systemd/system/nginx.service.
[ec2-user@ip-10-0-0-124 nginx]$


Database Details - RDS Manager
3tierwebappproject - S3 bucket
Systems Manager | ap-southeast
ChatGPT
React App
Not secure | 3.26.156.189
Not secure | 3.26.156.189
AWS 3-TIER WEB APP DEMO
VPC
Web Tier
App Tier
Database Tier
AZ1
AZ2
Public subnet
Private subnet
Private subnet
Amazon EC2
Amazon EC2
Amazon Aurora Primary DB
Internet Gateway
Elastic Load Balancing (ELB)
Amazon EC2
Amazon EC2
Aurora Read Replica
Public subnet
Private subnet
Private subnet
AURORA DATABASE DEMO PAGE
DEL
ID
AMOUNT
DESC
ADD
1 400 groceries
2 500 Fruits
3 800 Vegetables

Nagireddy Hari Krishna Reddy

AWS Services Utilized

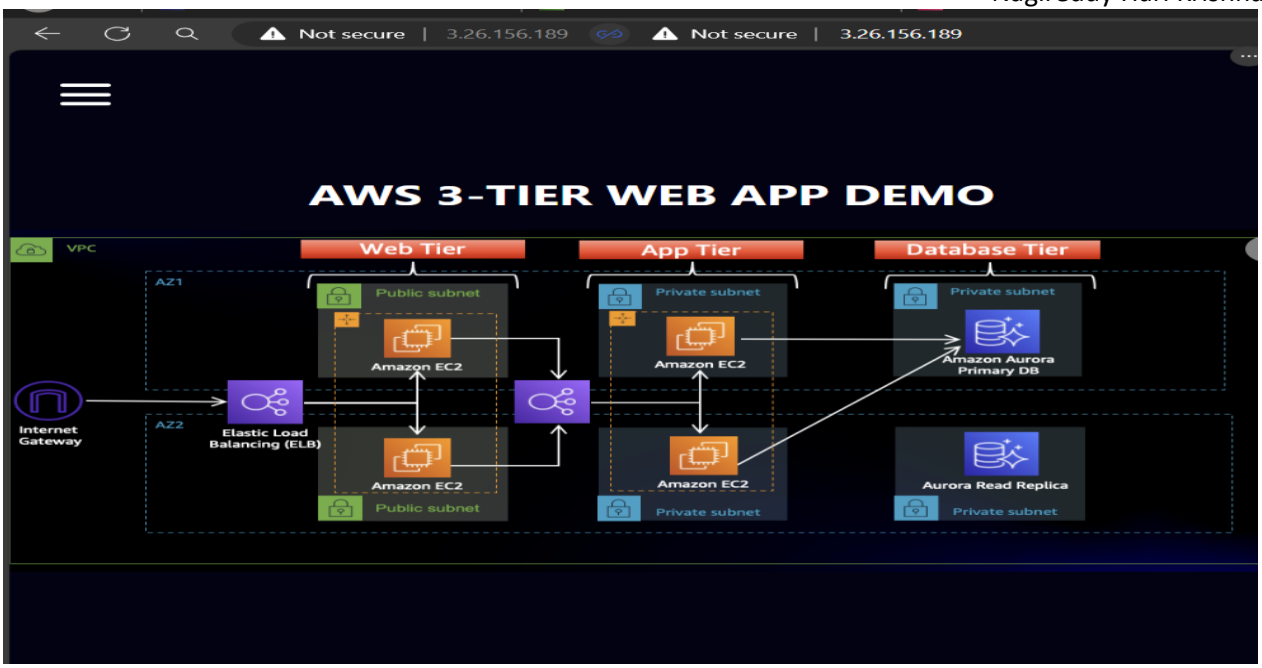
- EC2
- Amazon Aurora
- Amazon S3
- VPC
- Subnet
- Route Table
- NAT Gateway
- Security Group
- Elastic Load Balancing
- IAM

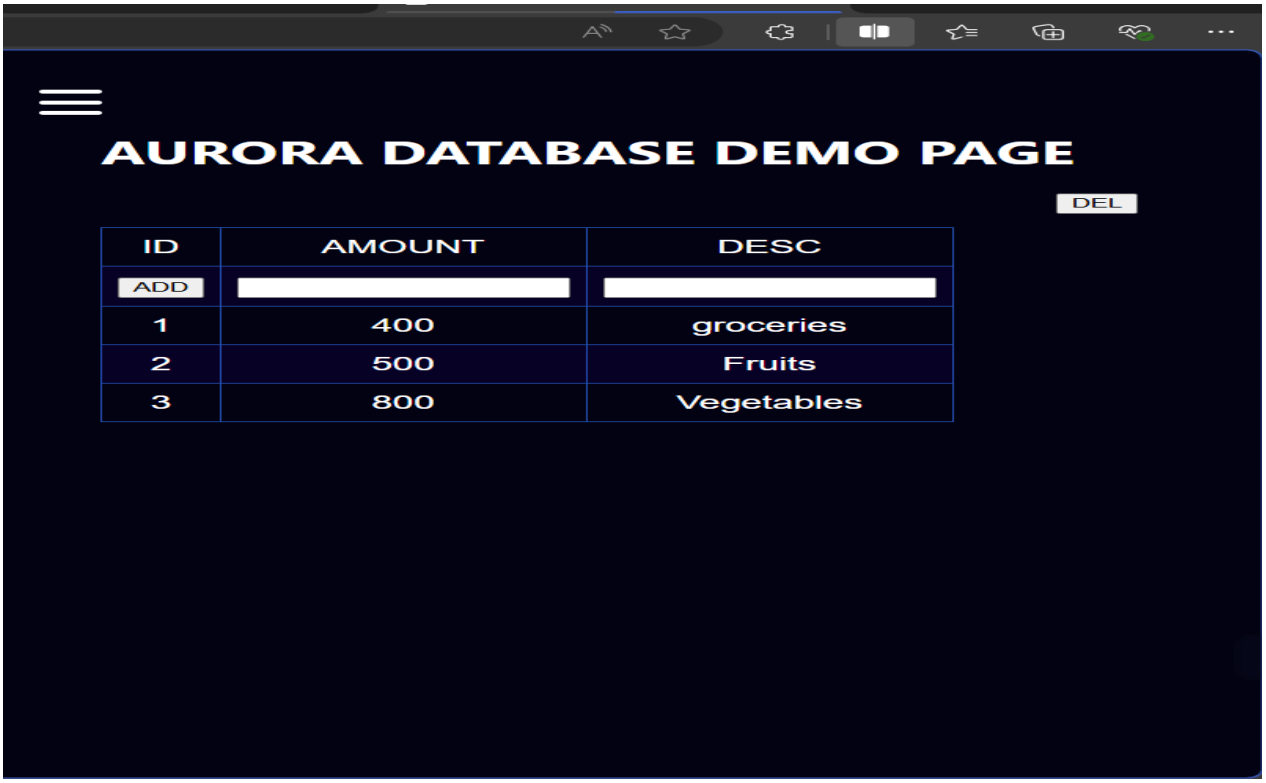
Highlights

- Set up a Virtual Private Cloud (VPC) with public and private subnets distributed across multiple Availability Zones.
- Configured NAT Gateways to ensure secure outbound internet access from private subnets.
- Deployed an Application Load Balancer (ALB) to distribute traffic efficiently to the web tier.
- Utilized Amazon RDS to manage the database instance within the data tier.
- Implemented Auto Scaling Groups (ASG) to automatically adjust the number of EC2 instances based on demand.
- Adhered to AWS Well-Architected Framework principles to create a resilient and high-performance architecture.

Desired Output

- use public IP from the web-instance and paste it in your web browser You can see the Application is getting served, we successfully created Three-tier web Architecture which is Scalable, highly available, fault-tolerant and secure.


Not secure | 3.26.156.189
Not secure | 3.26.156.189
AWS 3-TIER WEB APP DEMO
VPC
Web Tier
App Tier
Database Tier
AZ1
AZ2
Public subnet
Private subnet
Private subnet
Amazon EC2
Amazon EC2
Amazon Aurora Primary DB
Internet Gateway
Elastic Load Balancing (ELB)
Amazon EC2
Amazon EC2
Aurora Read Replica
Public subnet
Private subnet
Private subnet


AURORA DATABASE DEMO PAGE
DEL
ID
AMOUNT
DESC
ADD
1
400
groceries
2
500
Fruits
3
800
Vegetables

Key Learnings

- Implementing the Three-tier architecture improved security, scalability, and maintainability while providing development flexibility. The project achieved a robust, scalable, and highly available architecture with optimized resource usage and fault tolerance across multiple Availability Zones.

- I would like to extend my heartfelt thanks to Chandani Madam for her invaluable guidance and motivation throughout this project. Her support has been instrumental in my progress. I am eager to gain more hands-on experience with AWS and continue learning and growing in this area.