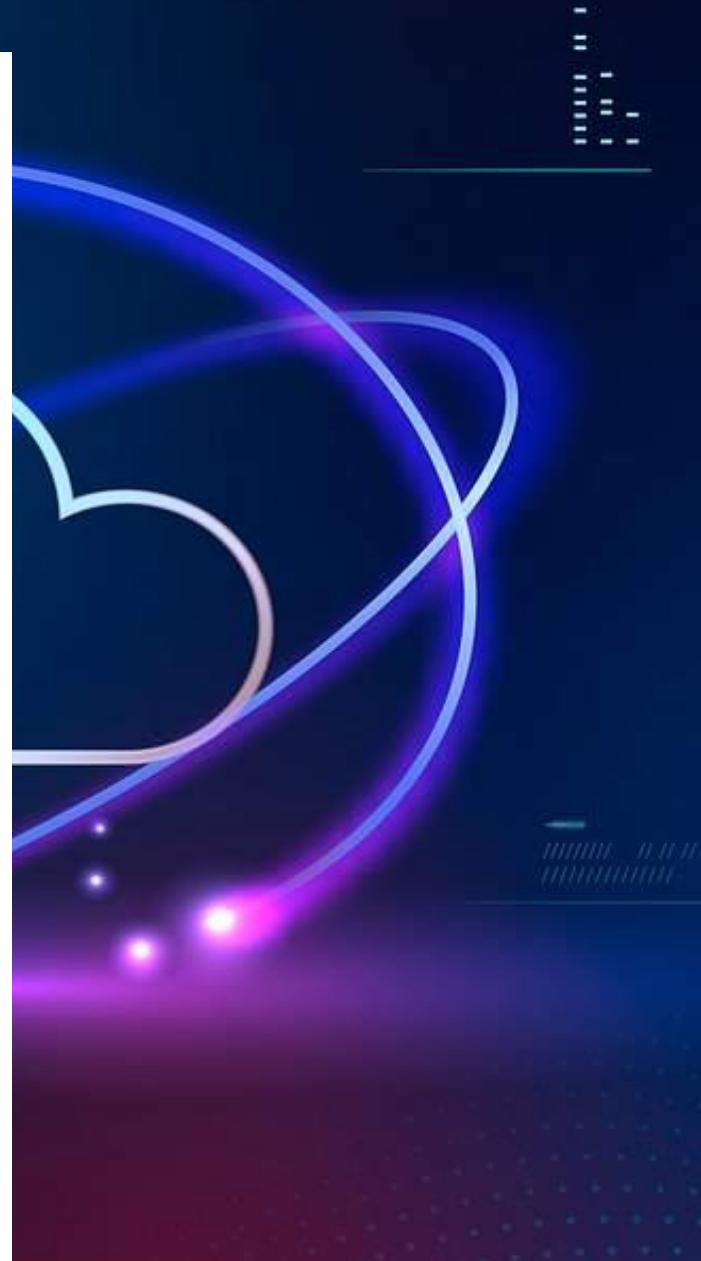


INTEGRATING APPLICATIONS HOSTED ON EC2 WITH RDS USING THIRD- PARTY TOOL



AUGUST 24

Devamadhav Kattimuttathu



INTEGRATING APPLICATIONS HOSTED ON EC2 WITH RDS USING THIRD-PARTY TOOL

Problem Statement

The goal is to deploy a scalable, cloud-based solution for a car manufacturing and dealership ecosystem using Amazon Web Services (AWS). The solution involves three interconnected applications:

- App1 - Car Manufacturing: This application manages car manufacturing processes. It requires deployment on an AWS EC2 instance with a database hosted on Amazon RDS.
- App2 - Car Dealership: This application handles car dealership operations and needs to be integrated with App1 to fetch manufacturing data. It will also be deployed on an AWS EC2 instance with a database on Amazon RDS.
- App3 - Car Details: This application provides detailed information about the cars manufactured and available for sale. It connects with App2 to fetch dealership data and with App1 for manufacturing data. App3 does not require a database but will be deployed using an EC2 container on AWS.

Observations and Screenshots

APP 1

Initially, we created an RDS database named app1 and selected MySQL as the database engine, choosing the free tier. We then set up an inbound rule in the security group to allow MySQL/Aurora connections. Following that, we launched an EC2 instance running Windows.

After setting up the EC2 instance, we connected it to the RDS database. We accessed the instance using the key pair generated during its creation. Once logged in, we installed SQL Workbench and XAMPP on the EC2 instance and started the XAMPP services.

Next, we configured SQL Workbench to connect to the RDS database using the provided endpoint. We created a new schema in the database and imported the necessary file into it. Additionally, we copied the contents of the app1 folder into the htdocs directory of the EC2 instance.

When we accessed the website via localhost/app1, it prompted us for a username and password and successfully logging into the website. We added a car brand named "BMW," and it appeared correctly in the car brand list.

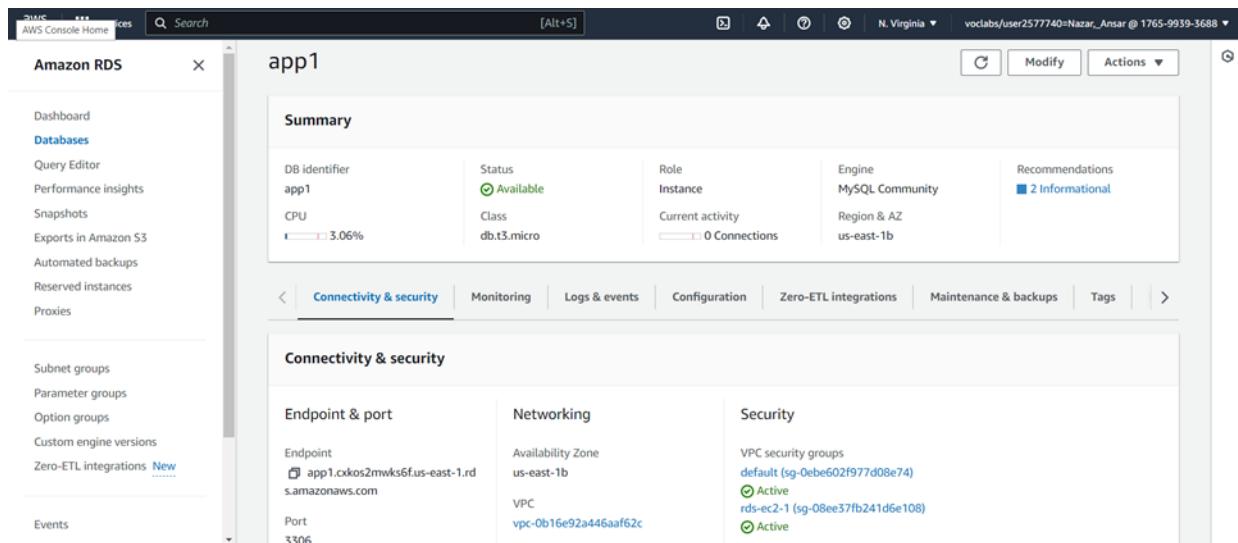


Figure 1 RDS database with name app1 created.

The screenshot shows the AWS RDS Security Groups interface. The URL is [EC2 > Security Groups > sg-0ebe602f977d08e74 - default > Edit inbound rules](#). The title is "Edit inbound rules". A note says "Inbound rules control the incoming traffic that's allowed to reach the instance." Below is a table of inbound rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-08dc495367840ff8	MySQL/Aurora	TCP	3306	Custom	0.0.0.0/0
sgr-0bfc87c6ab2b4af72	All traffic	All	All	Custom	sg-0ebe602f977d08e74

Buttons at the bottom include "Add rule", "Cancel", "Preview changes", and "Save rules".

Figure 2 RDS database security group rule.

The screenshot shows the AWS EC2 Instances interface. The URL is [EC2 > Instances > i-0d35de5b6421281ba](#). The title is "Instance summary for i-0d35de5b6421281ba (ids-finalproject)". The instance is running with a public IP of 52.200.216.181 and a private IP of ip-172-31-29-3.ec2.internal. It has an Auto-assigned IP address of 52.200.216.181 [Public IP]. The VPC ID is vpc-0b16e92a446aaf62c. The instance ARN is arn:aws:ec2:us-east-1:176599393688:instance/i-0d35de5b6421281ba.

Figure 3 An EC2 instance with the name *ids-finalproject* created.

The screenshot shows the AWS RDS service page. On the left, there's a sidebar with options like Dashboard, Databases, Query Editor, Performance insights, Snapshots, Exports in Amazon S3, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, and Custom engine versions. The main area is titled "Connected compute resources (1) Info". It lists one EC2 instance: "i-0d35de5b6421281ba" (Resource identifier), "EC2 instance" (Type), "us-east-1a" (Availability Zone), "rds-ec2-1" (VPC security group), and "ec2-rds-1" (Compute resource security group). Below this is a section for "Proxies (0)", which is currently empty. There are "Actions" buttons for both sections.

Figure 4 RDS database connected to the instance.

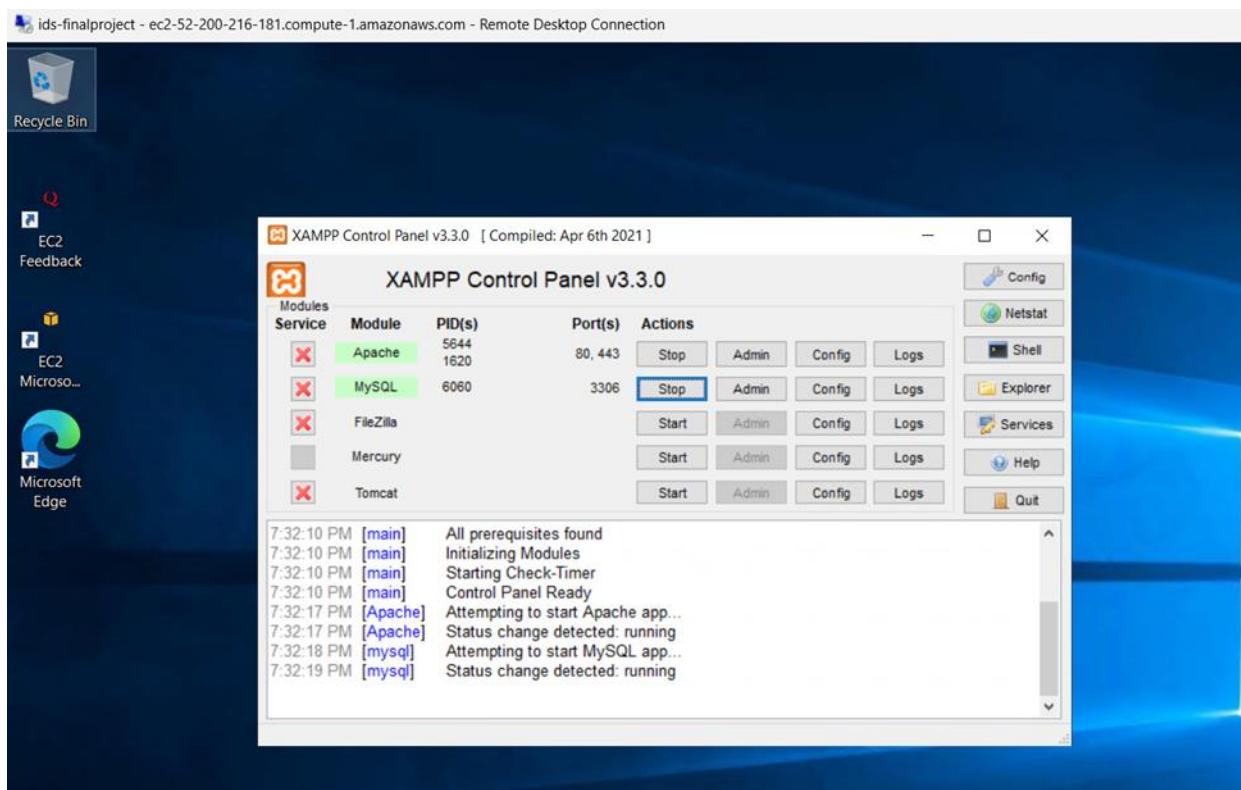


Figure 5 XAMPP installed and started the services.

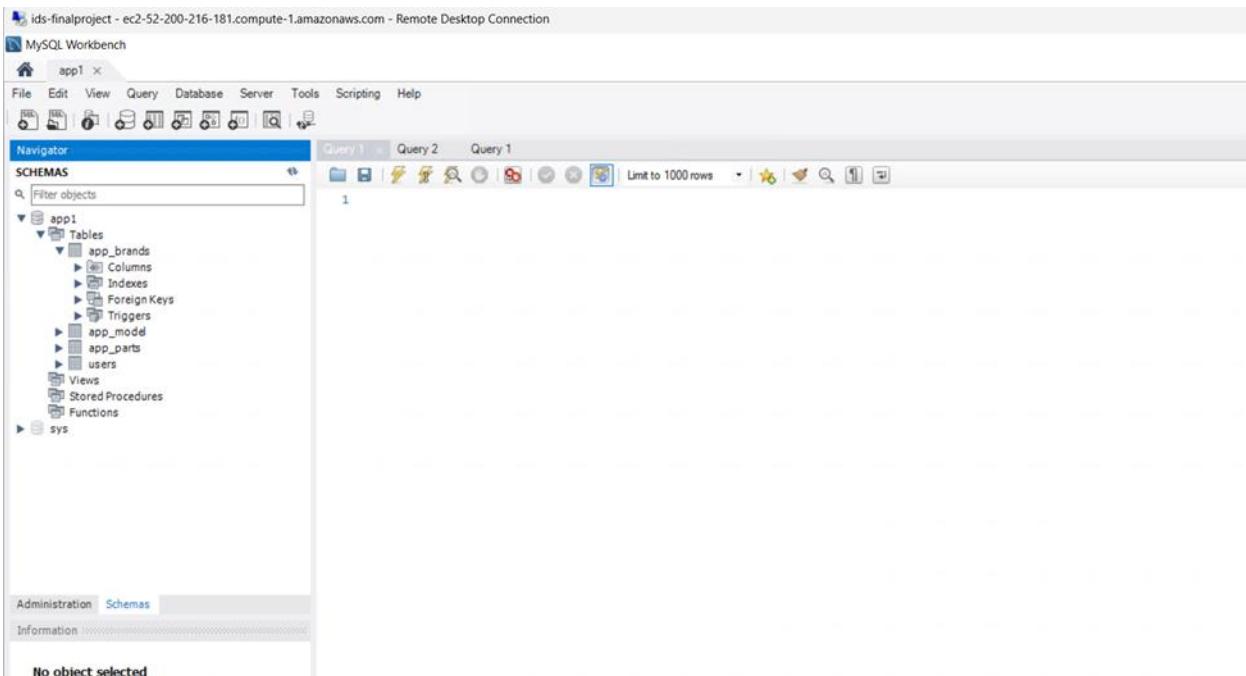


Figure 6 Workbench connected to the database and a schema named app1 created and imported the SQL.

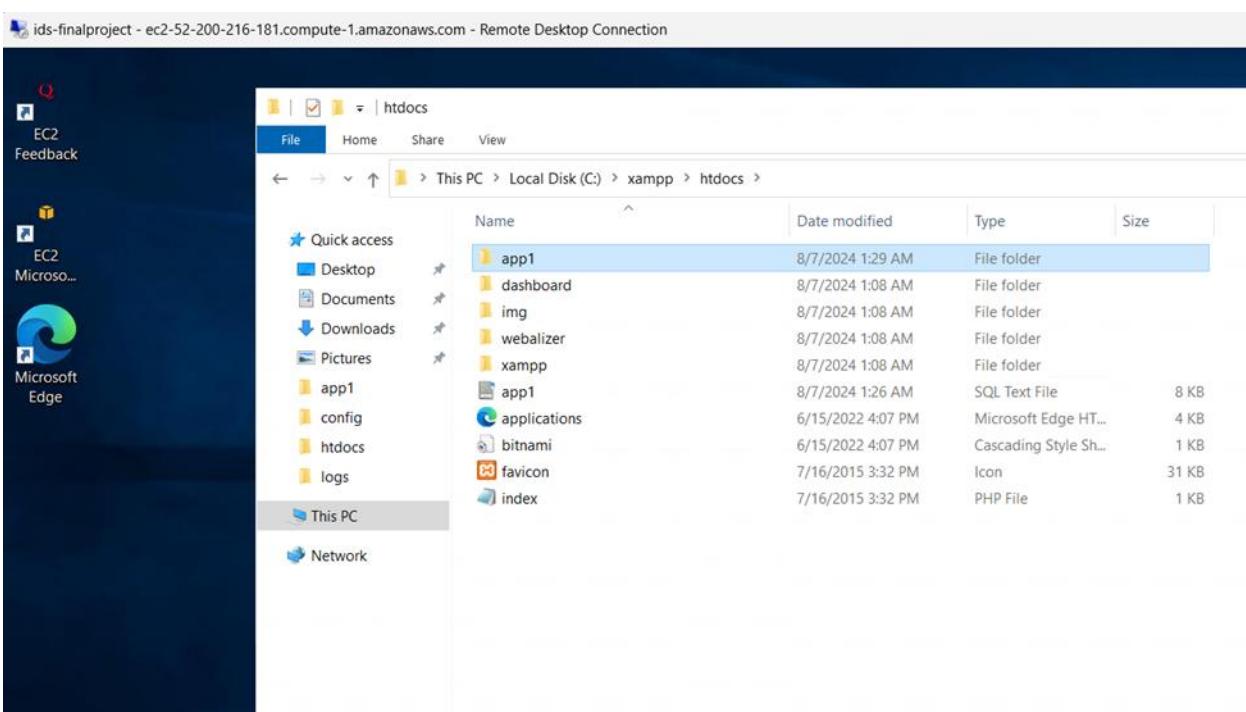


Figure 7 app1 data copied to the htdocs folder in the EC2 instance.

ids-finalproject - ec2-52-200-216-181.compute-1.amazonaws.com - Remote Desktop Connection

.env - Notepad

File Edit Format View Help

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:2pWfo6IIIs6XLyT4Un59G0qBILhF4fa2nLpIHAP21BvA=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=app1.cxkos2mwks6f.us-east-1.rds.amazonaws.com
DB_PORT=3306
DB_DATABASE=app1
DB_USERNAME=admin
DB_PASSWORD=Secret55

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1
```

Figure 8 Configuration details of the .env file.

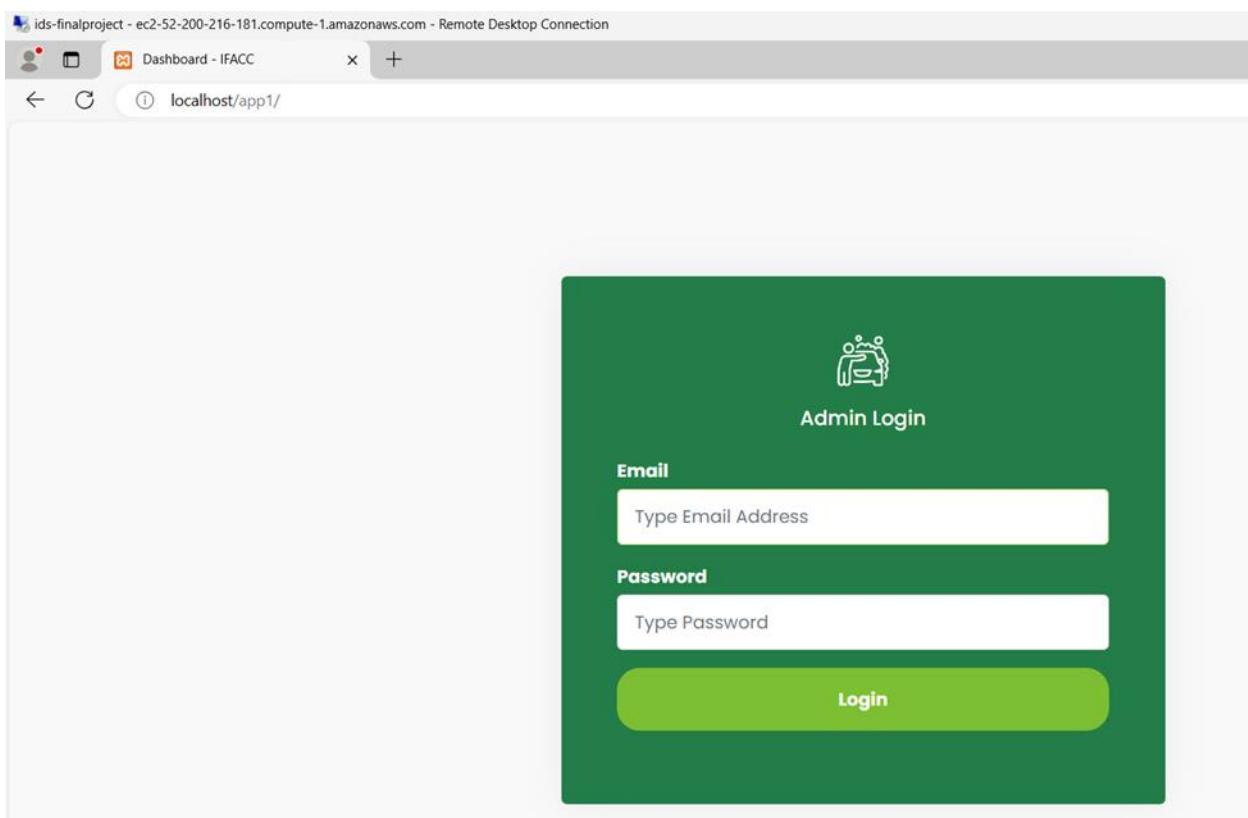


Figure 9 app1 login page accessed.

A screenshot of a web browser window titled "ids-finalproject - ec2-52-200-216-181.compute-1.amazonaws.com - Remote Desktop Connection". The address bar shows "localhost/app1/dashboard". The main content is a dashboard titled "Dashboard". On the left is a sidebar with icons for Dashboard (selected), Car Manufacturing (with a dropdown arrow), and Logout. The main area shows a "Brand List" table with four rows. The table has columns for "Brand Name", "date", and "Actions". Each row contains a "delete" button. The data is as follows:

Figure 10 Successfully logged into the app1.

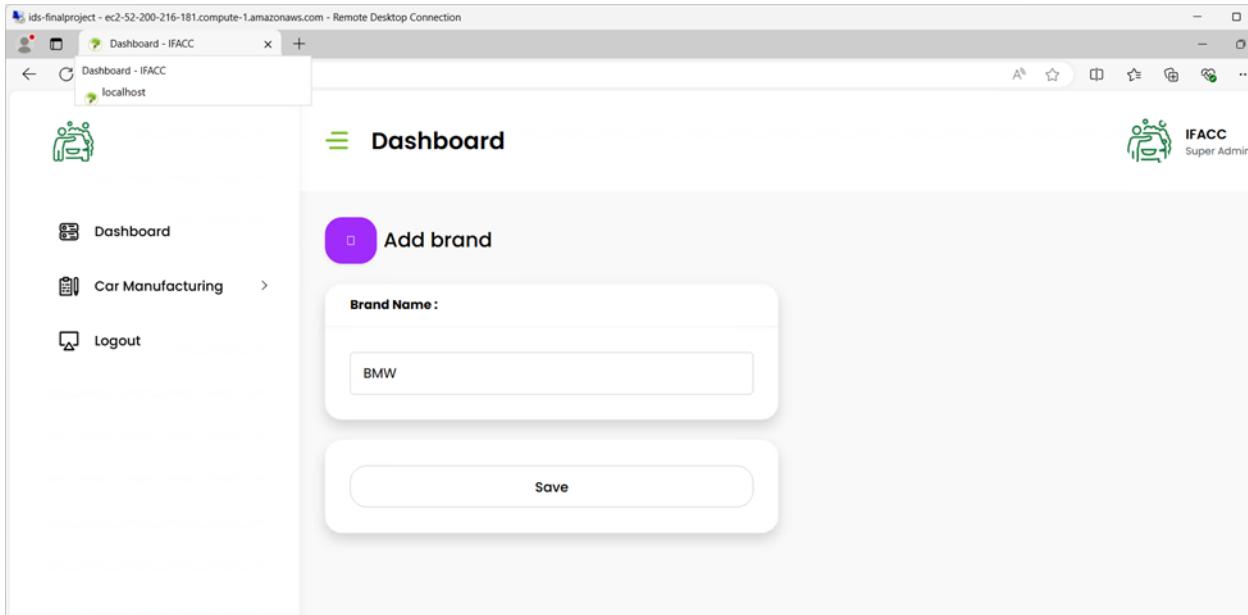


Figure 11 Adding a new car brand BMW.

Brand Name	Date	Action
Honda	2022-03-11 04:26:13	<button>delete</button>
Maruti Suzuki	2022-03-11 04:26:20	<button>delete</button>
Toyota	2022-03-11 04:26:24	<button>delete</button>
Ford	2022-03-18 15:41:49	<button>delete</button>
BMW	2024-08-07 21:13:39	<button>delete</button>

Figure 12 Added car brand is shown in the list.

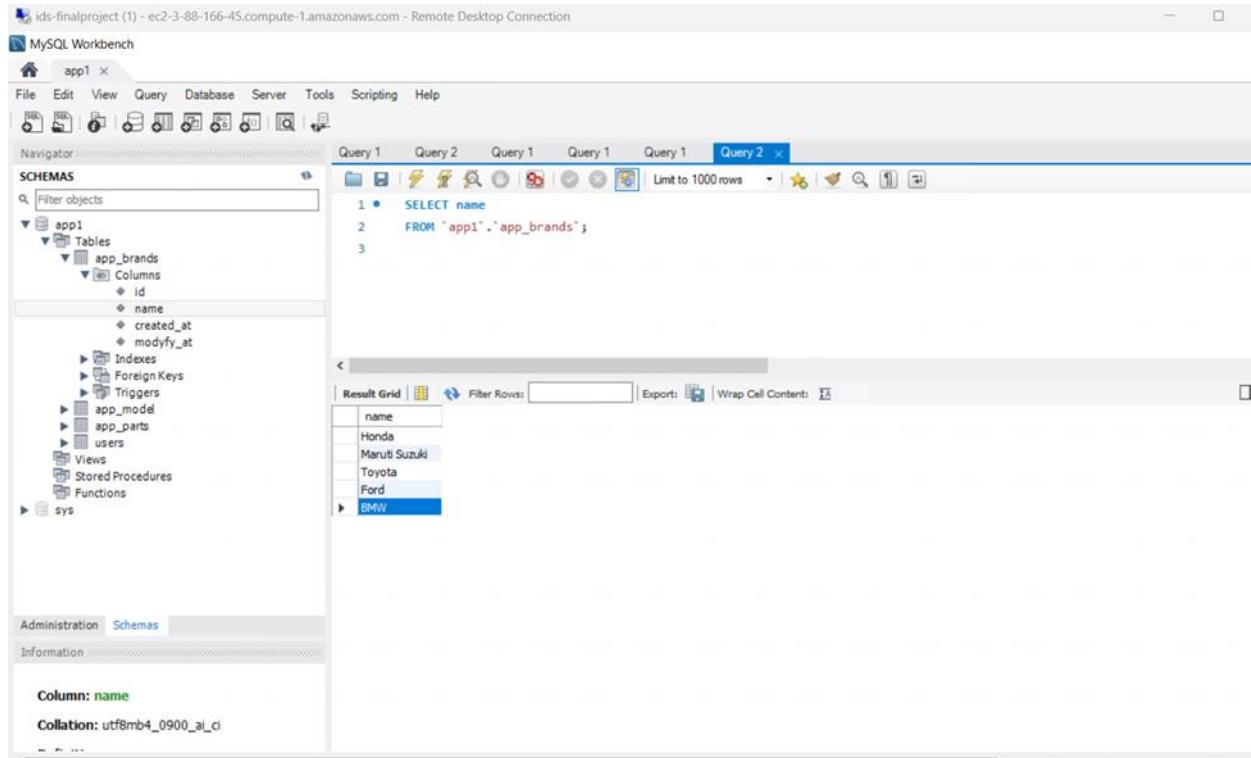


Figure 13 added brand is showing in the workbench.

APP 2

We started the lab by creating an EC2 instance for app2. We enabled HTTP, RDS, HTTP traffic for the EC2. Afterward, we created an RDS database in our AWS named app2. We configured the security groups to allow traffic from the ec2 and the mysql port and ensured the dp is publically accessible. Moreover, the ec2 instance is successfully connected to the RDS. Thereafter we accessed the ec2 instance using RDP and installed XAMPP and workbench to proceed with further configurations. XAMPP was installed, and we started the Apache and MySQL server successfully. Afterward, we created a connection with our RDS database in the workbench using the endpoint from the RDS instance we created. The connection with RDS was successfully established, and we created a new schema named app2. Afterward, we imported all the tables to the schema. The app2 file is copied to the htdocs folder in XAMPP. We then navigated to the .env file and configured the host, db_name, port, username, and password correspondingly. All the connections are configured, and we loaded the applications locally and accessed the web application successfully. The modifications made in app1 are reflected in app2. We verified the integration by adding a new brand BMW in app1, and it was listed in the brands in app2.

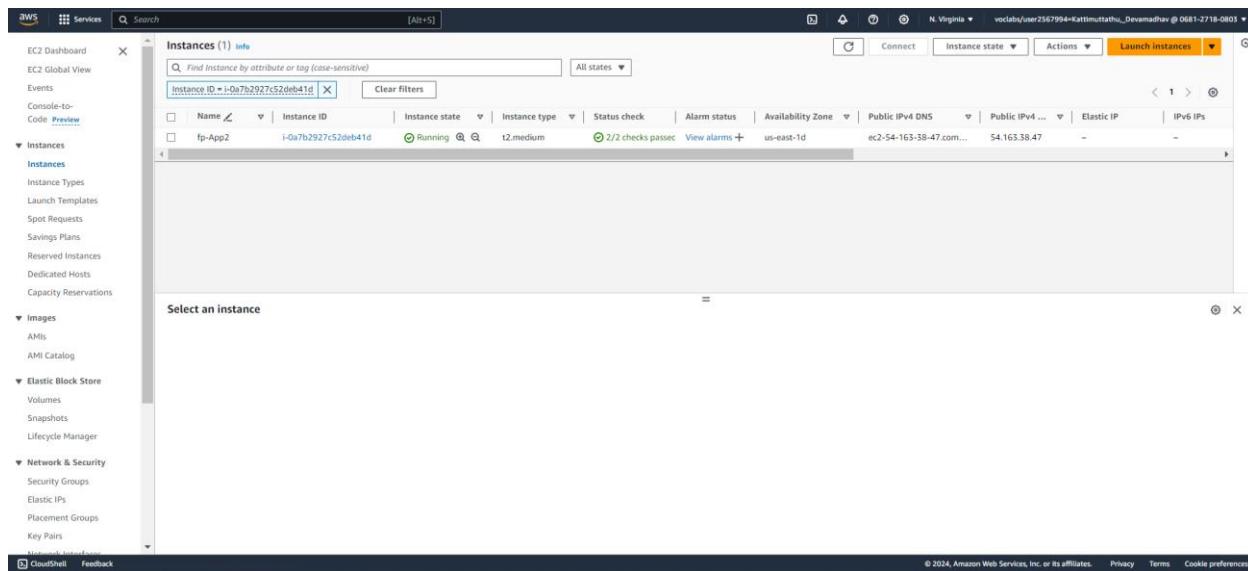


Figure 1 - Created an instance for app2

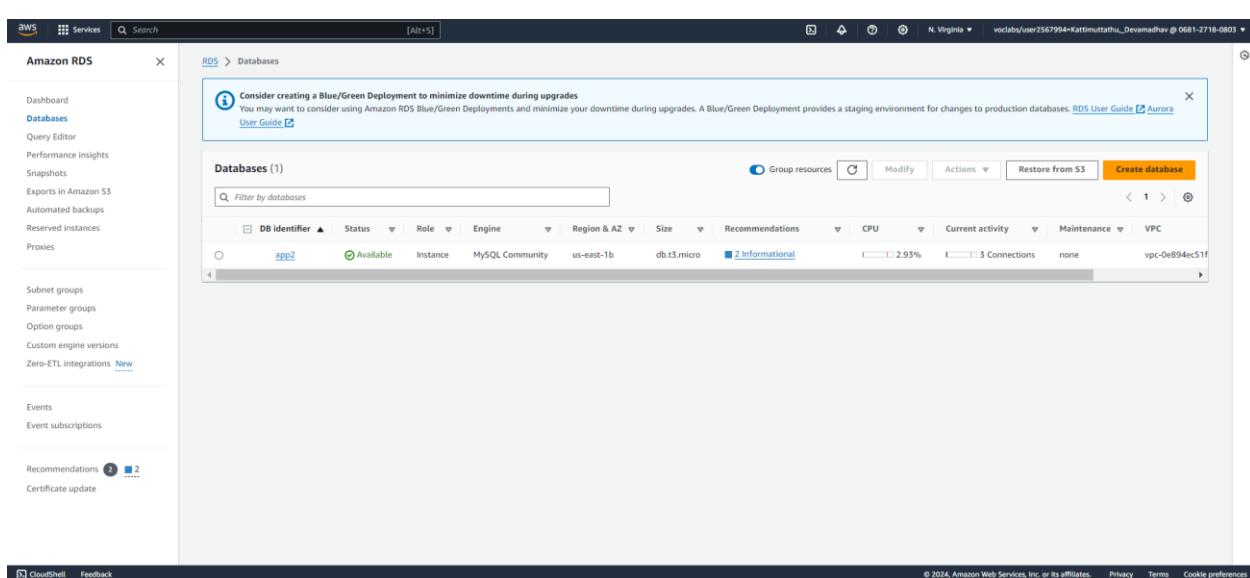


Figure 2- Created an RDS in app2

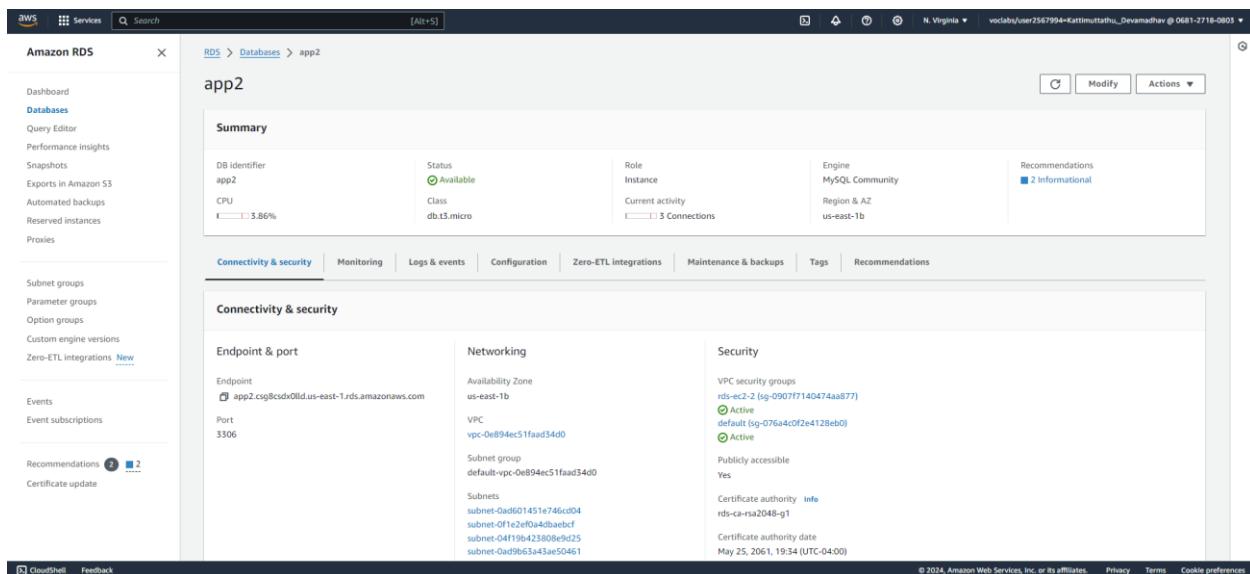


Figure 3 -Displaying the details of RDS

The screenshot shows the AWS EC2 Dashboard with the 'Security Groups' section selected. There are two security groups listed:

Security group ID	Security group name	VPC ID	Description
sg-0213ead52725a0b38	ec2-rds-2	vpc-0e894ec51faad34d0	Security group attached to instances to securely connect to fp-app2-db. Modification could lead to connection loss.
sg-0907f7140474aa877	rds-ec2-2	vpc-0e894ec51faad34d0	Security group attached to fp-app2-db to allow EC2 instances with specific security groups attached to connect to the database. Modification could lead to connection loss.

Figure 4 - Security groups of RDS

The screenshot shows the 'Edit outbound rules' screen for the security group sg-0213ead52725a0b38. It displays one rule:

Outbound rule ID	Type	Protocol	Port range	Destination	Description
sgr-0d1ba58cb96c02dd7	MySQL/Aurora	TCP	3306	Custom	Rule to allow connections to fp-app2-db from any instances this security group is attached to

Figure 5 - Outbound rule set for the ec2 instance

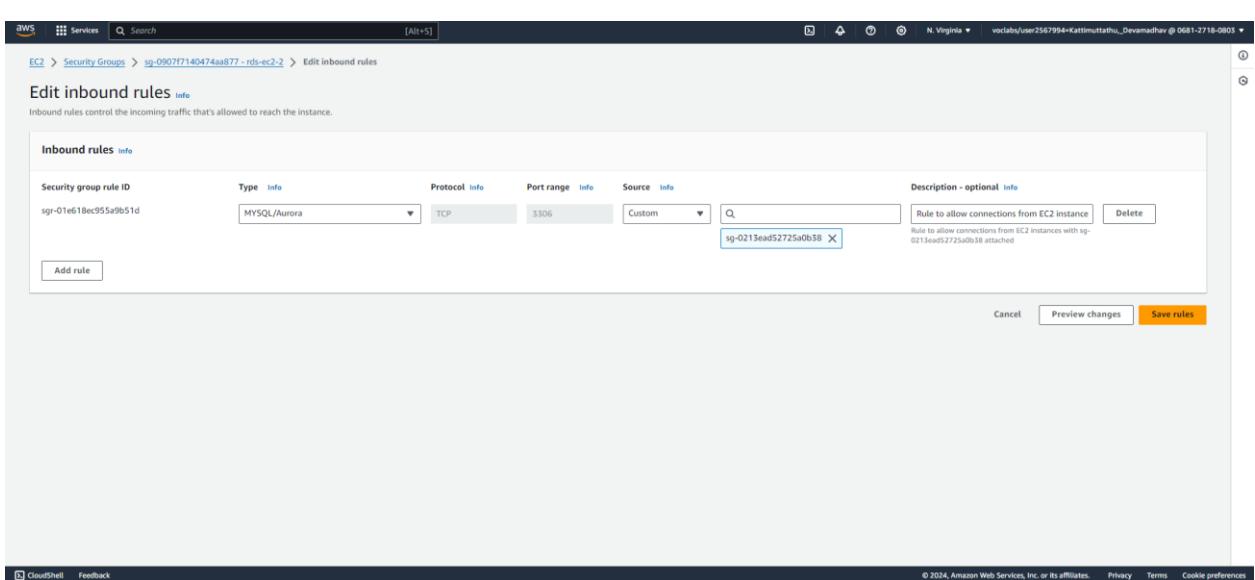


Figure 6 - Inbound set rules for RDS

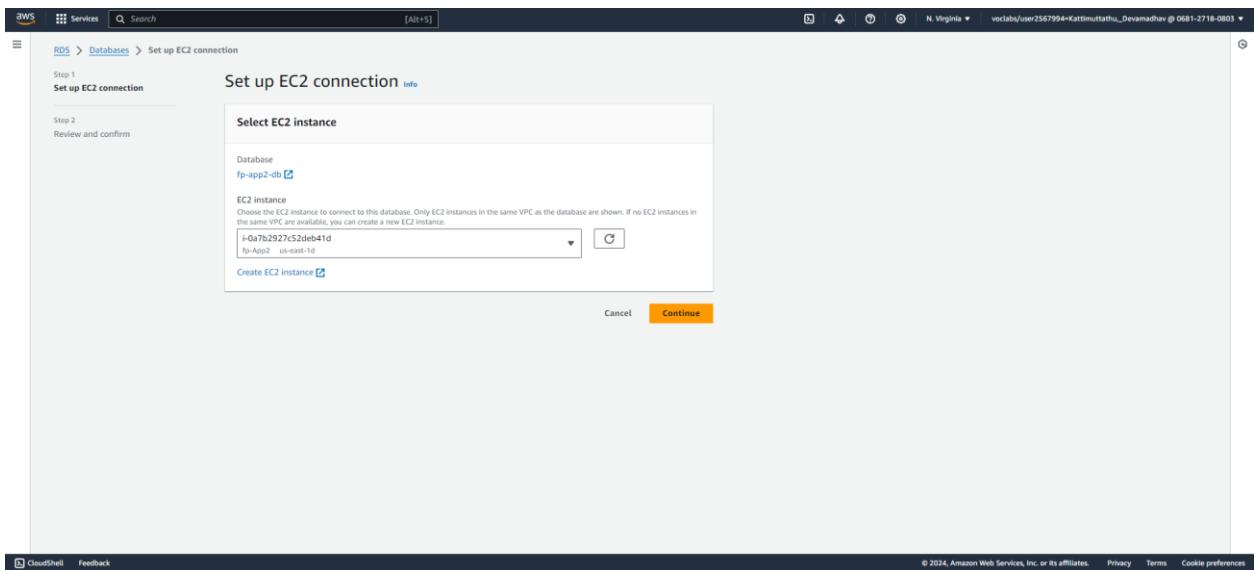


Figure 7 - Connecting Ec2 with RDS

The screenshot shows the AWS RDS Subnet group configuration page. The subnet group is named 'default-vpc-0e894ec51faad34d0'. It lists several subnets and their details. The 'Network type' is set to 'IPv4'. On the right, it indicates the subnet is 'Publicly accessible' (Yes), has a 'Certificate authority' of 'rds-ca-ca2048-g1', and a 'Certificate authority date' of 'May 25, 2021, 19:34 (UTC-04:00)'. The 'DB instance certificate expiration date' is 'August 06, 2025, 17:19 (UTC-04:00)'. Below this, the 'Connected compute resources' section shows one EC2 instance connected, with its identifier 'i-0x7a29277c52ebe1d1' highlighted with a red box. The EC2 instance is associated with the VPC security group 'rds-ec2-2' and the compute resource security group 'pr2-rds-2'. The 'Proxies' section shows no proxies currently assigned.

Figure 8 - Successfully connected the EC2 with RDS

The screenshot shows the MySQL Workbench interface. A 'Setup New Connection' dialog box is open. The 'Connection Name' is set to 'fp_connection', 'Connection Method' is 'Standard (TCP/IP)', and 'Parameters' are set to 'SSL'. The 'Hostname' is '8csdvlid.us-east-1.rds.amazonaws.com', 'Port' is '3306', 'Username' is 'admin', and 'Default Schema' is left blank. A sub-dialog box titled 'Please enter password for the following service:' is displayed, asking for the password for the service 'Mysql@fp-app2-db.cspxdsvlilid.us-east-1.rds.amazonaws.com:3306' with user 'admin'. The password field contains '*****'. At the bottom of the main dialog, there are buttons for 'Configure Server Management...', 'Test Connection', 'Cancel', and 'OK'.

Figure 9 - Creating a connection in workbench

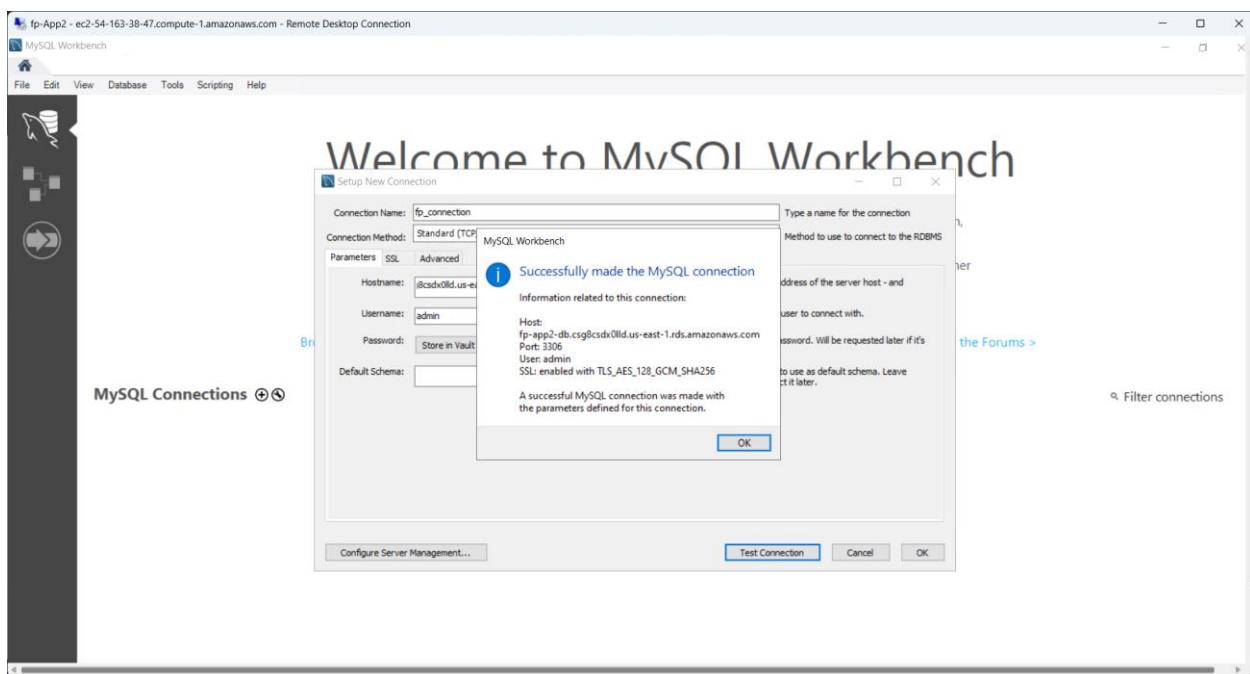


Figure 10 - Successfully connected to the RDS

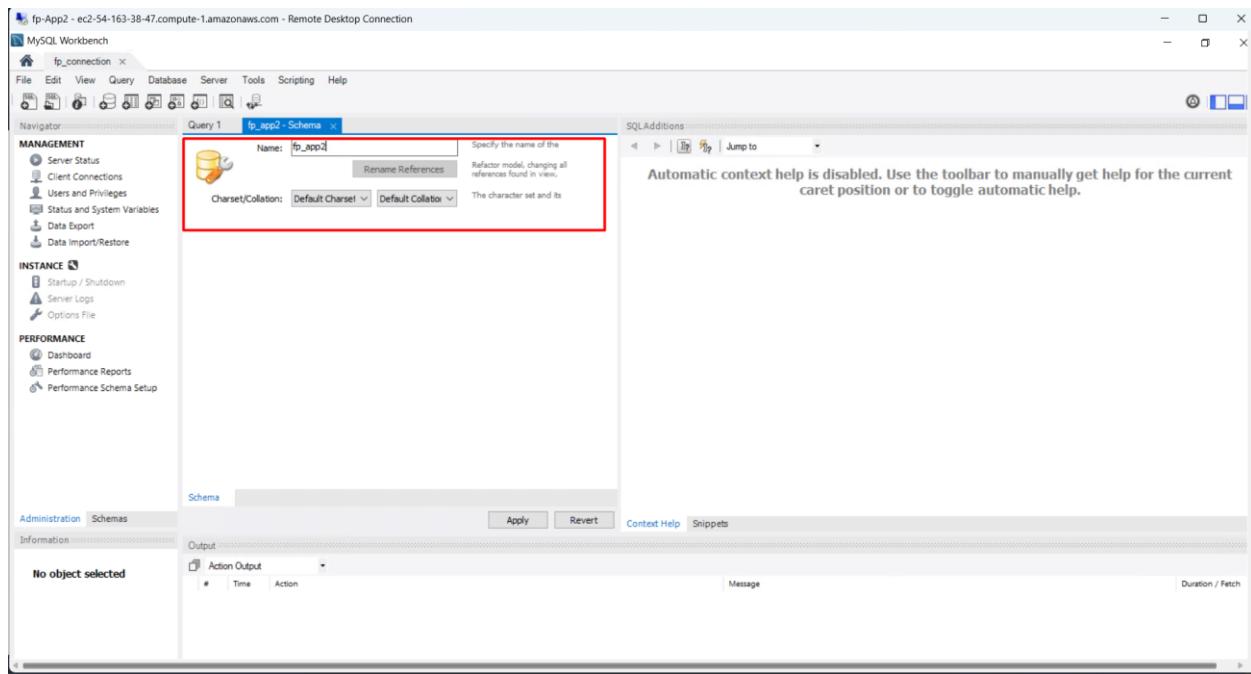


Figure 11 - Creating a schema inside the workbench

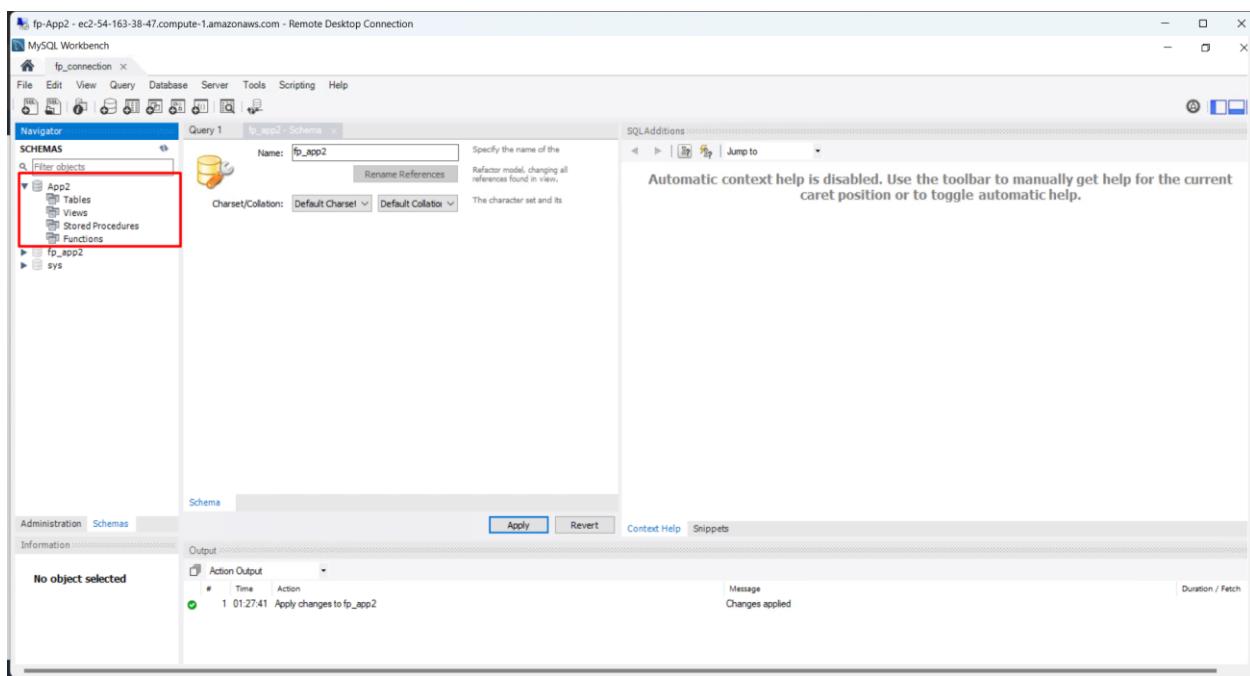


Figure 12 -Successfully created the schema

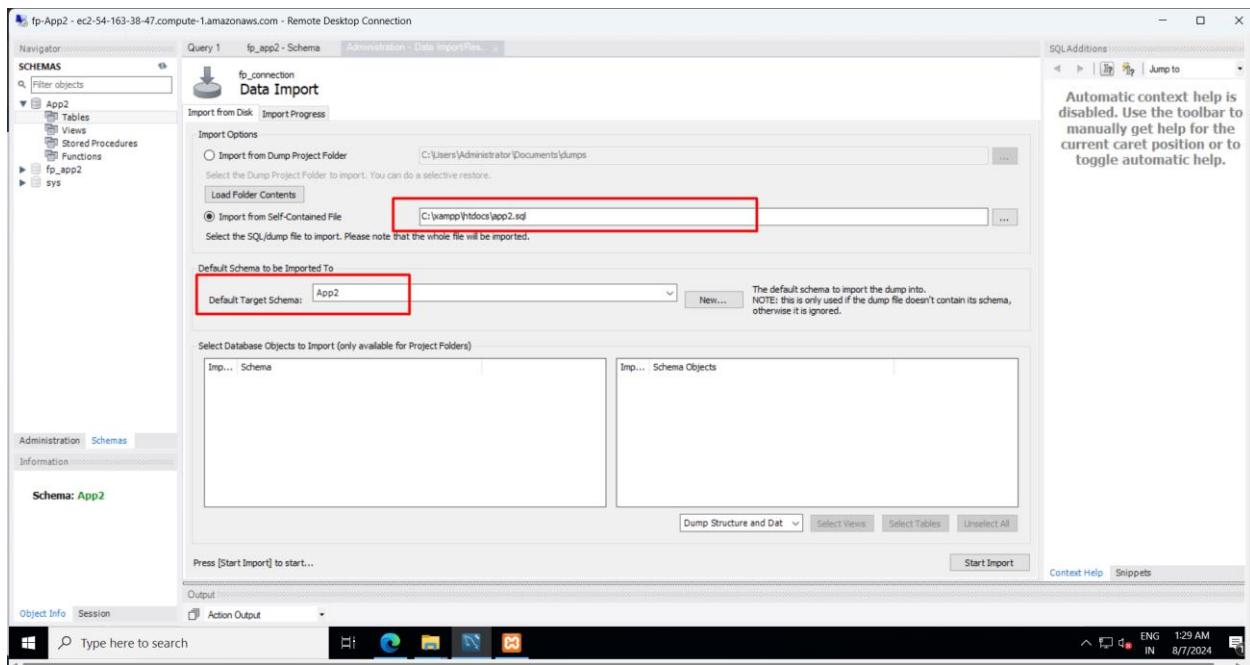


Figure 13 _ Importing the tables

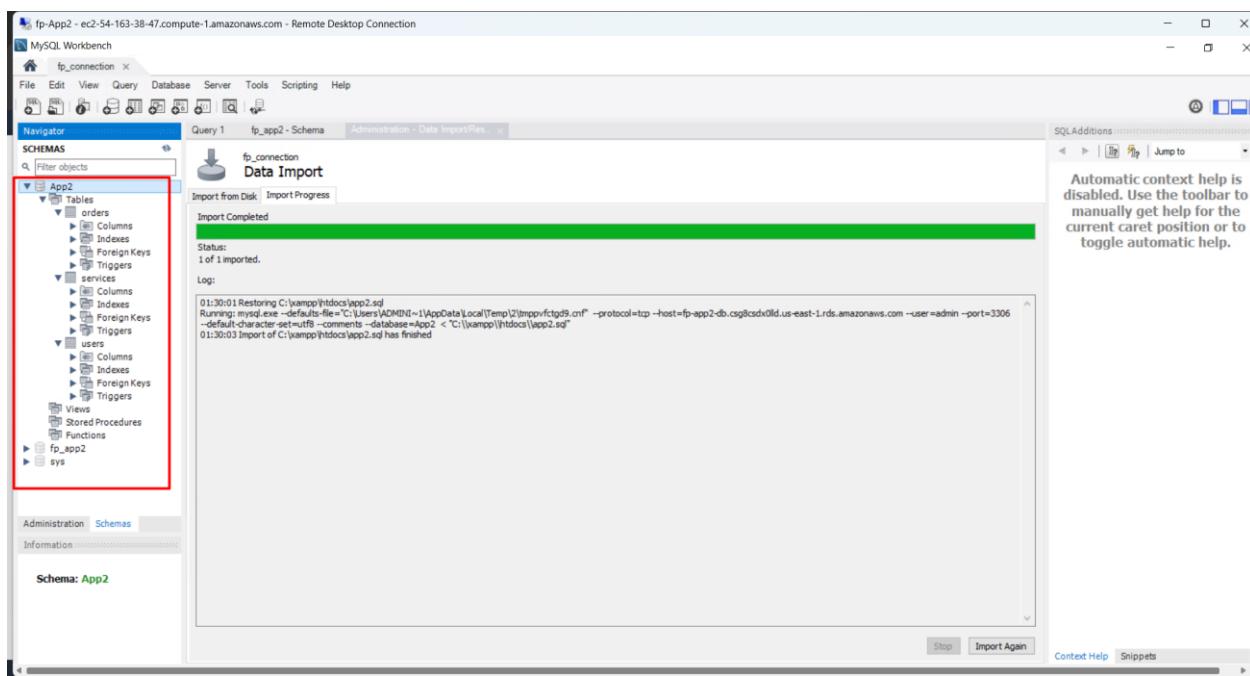


Figure 14 - Successfully imported the data

```

Recycle Bin
File Edit Format View Help
APP_ENV=local
APP_KEY=base64:2pWfo6IIis6XlyT4UUn59G0qB1lhF4fa2nLpIHAP21BVA=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=app2.csg8csdx0lld.us-east-1.rds.amazonaws.com
DB_PORT=3306
DB_DATABASE=app2
DB_USERNAME=admin
DB_PASSWORD=Secret55

DB_CONNECTION_SECOND=mysql
DB_HOST_SECOND=app1.cxxos2mwks6f.us-east-1.rds.amazonaws.com
DB_PORT_SECOND=3306
DB_DATABASE_SECOND=app1
DB_USERNAME_SECOND=admin
DB_PASSWORD_SECOND=Secret55

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=

```

Figure 15 - Modified the .env file

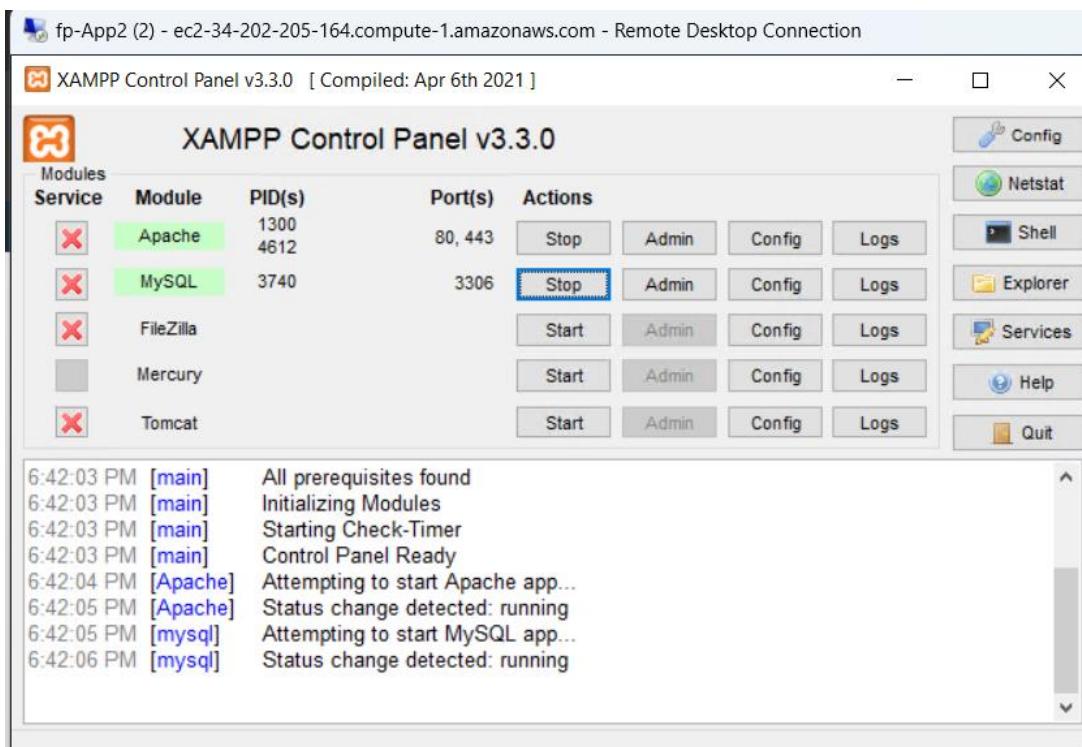


Figure 16 -Displaying the status of xampp

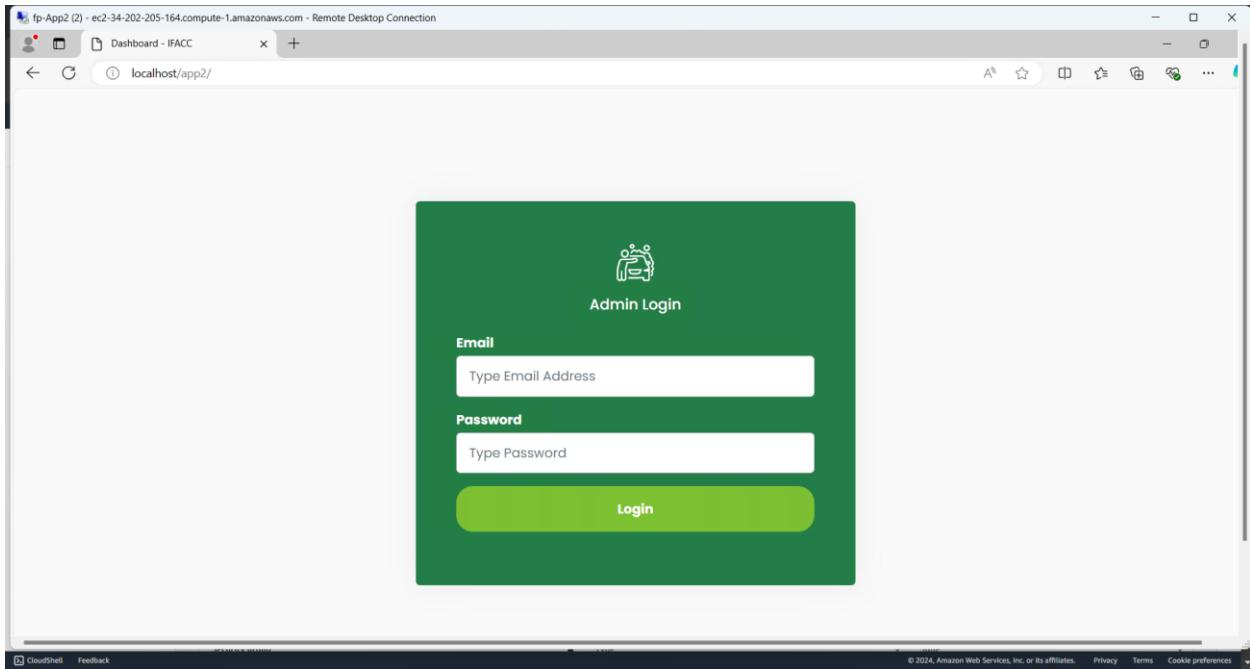


Figure 17 - Successfully accessed the homepage

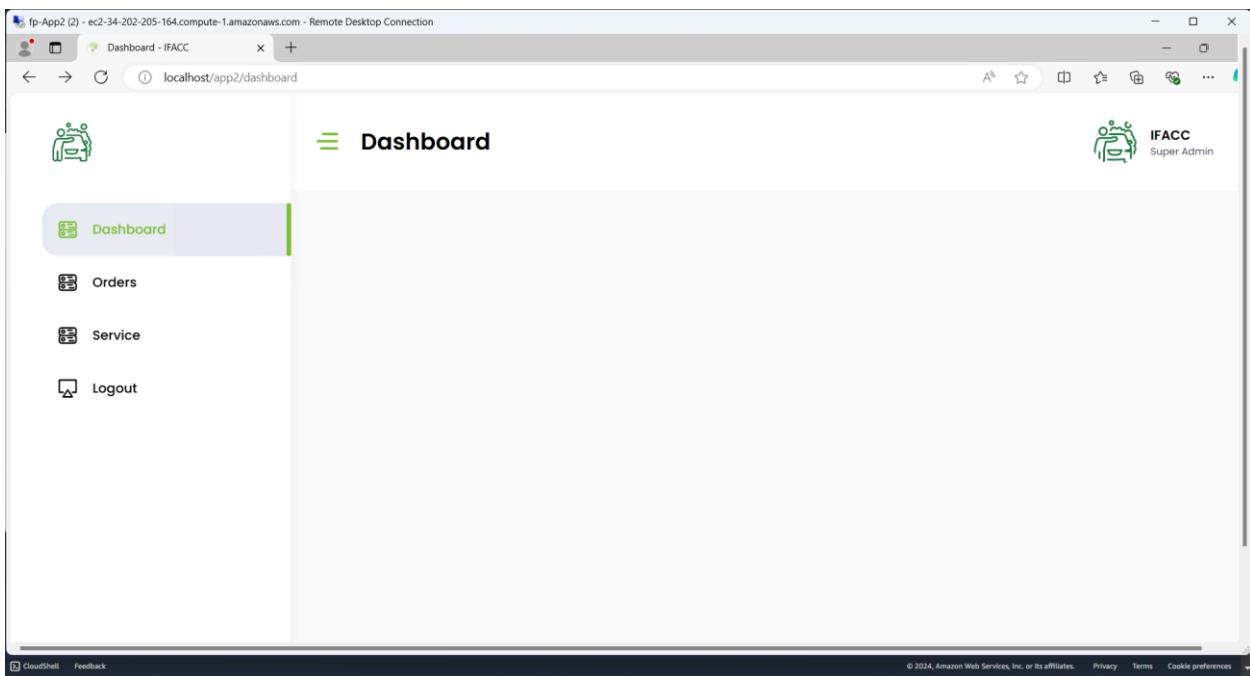


Figure 18 - Accessed the dashboard of our app2

The screenshot shows a web browser window titled 'fp-App2 (2) - ec2-34-202-205-164.compute-1.amazonaws.com - Remote Desktop Connection'. The URL is 'localhost/app2/orders'. The page has a sidebar on the left with icons for Dashboard, Orders (selected), Service, and Logout. The main area is titled 'Dashboard' and shows a table of 'Orders'. The table has columns: Customer Name, Address, Telephone, Brand, Model, Year, Price, DOC, and Actions. There are two rows of data:

Customer Name	Address	Telephone	Brand	Model	Year	Price	DOC	Actions
test	india	356356	Ford	Mustang	2013	300	18 March, 2022	<button>delete</button>
test	test dr, waterloo	9998889999	Ford	Mustang	2012	4000	18 March, 2022	<button>delete</button>

Figure 19 - Displaying the order tab

The screenshot shows a web browser window titled 'fp-App2 (2) - ec2-34-202-205-164.compute-1.amazonaws.com - Remote Desktop Connection'. The URL is 'localhost/app2/orders/create'. The page has a sidebar on the left with icons for Dashboard, Orders, Service, and Logout. The main area is titled 'Dashboard' and shows a form titled 'Add brand'. The form consists of several input fields:

- Brands:** A dropdown menu showing 'Honda'.
- Model:** A dropdown menu showing 'City'.
- Year:** An input field showing '2024'.
- Price:** An input field showing '30000'.
- Car VIN:** An input field containing '12345678qwerty'.
- Customer Name:** An input field labeled 'Customer Name'.

Figure 20 - Adding a new order

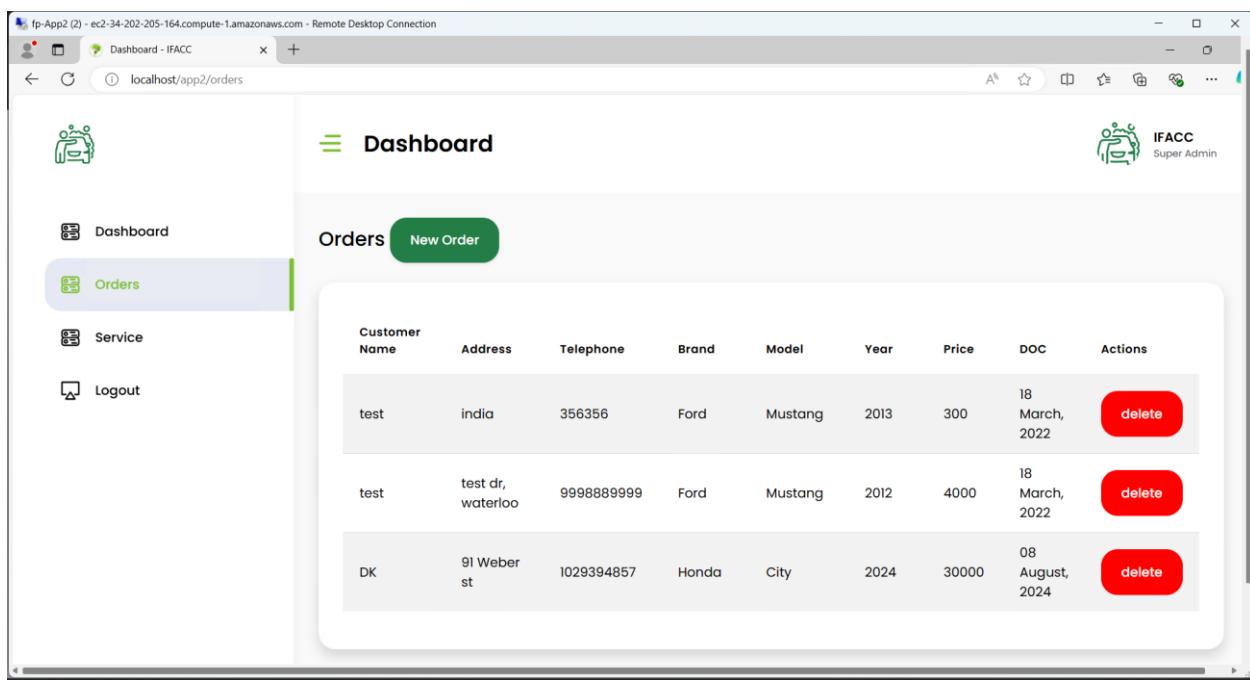


Figure 21 - Successfully added a new order

	id	vin	customer_name	address	telephone	brand_name	model_name	year	price	created_at	modify_at
1	3	78656450dfg	test	india	356356	Ford	Mustang	2013	300	2022-03-18 15:44:01	2022-03-18 15:44:01
2	4	mcfdfk3453	test	test dr, waterloo	9998889999	Ford	Mustang	2012	4000	2022-03-18 15:45:01	2022-03-18 15:45:01
3	5	123456789query	DK	91 Weber st	1029394857	Honda	City	2024	30000	2024-08-08 00:18:30	2024-08-08 00:18:30

Figure 22 - Displaying the new order updated on our db

Dashboard

Add Service

VIN:	Customer Name:
123456dkaz	Dk
Mechanic Name:	Service Type:
ansar	xxxx
Date of service:	
08/01/2024	

Figure 23 - Adding a new service in app2

Dashboard

Orders **New Order**

Customer Name	Date of service	vin	Mechanic Name	Service Type	DOC	Actions
TEST	2022-03-21	7865645gdfg	TESTMEC	oil change	18 March, 2022	delete
Dk	2024-08-01	123456dkaz	ansar	xxxx	08 August, 2024	delete

Figure 24 - Successfully added a new service log

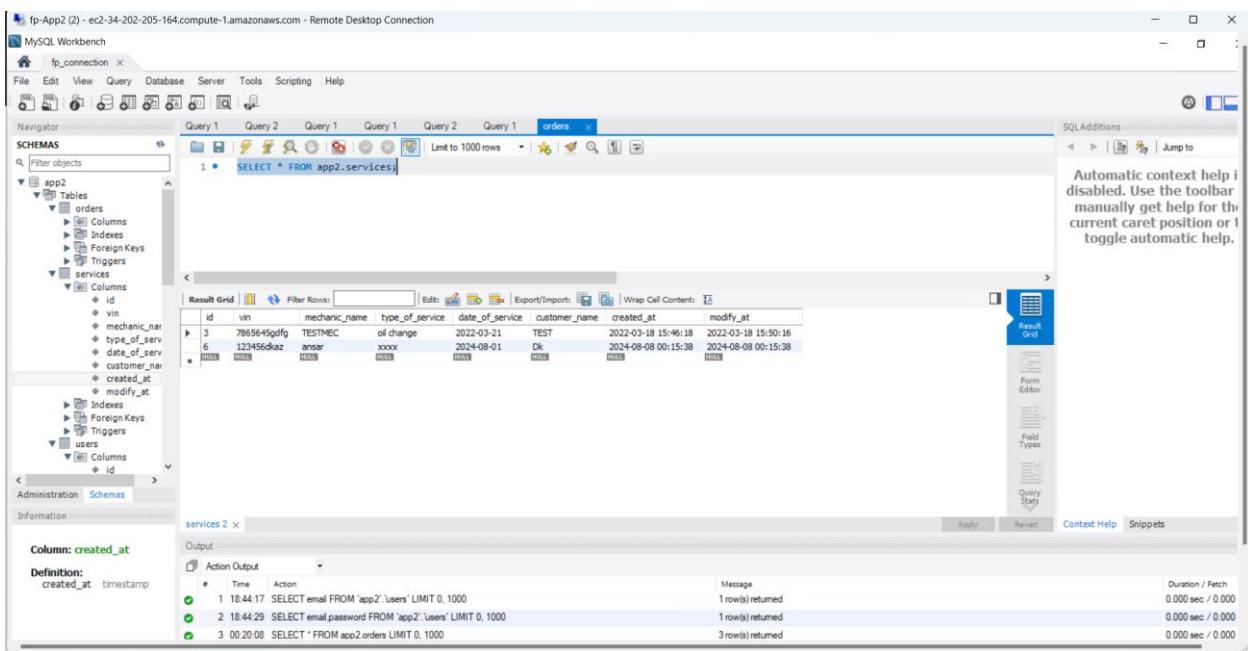


Figure 25 - The service is successfully added on the db

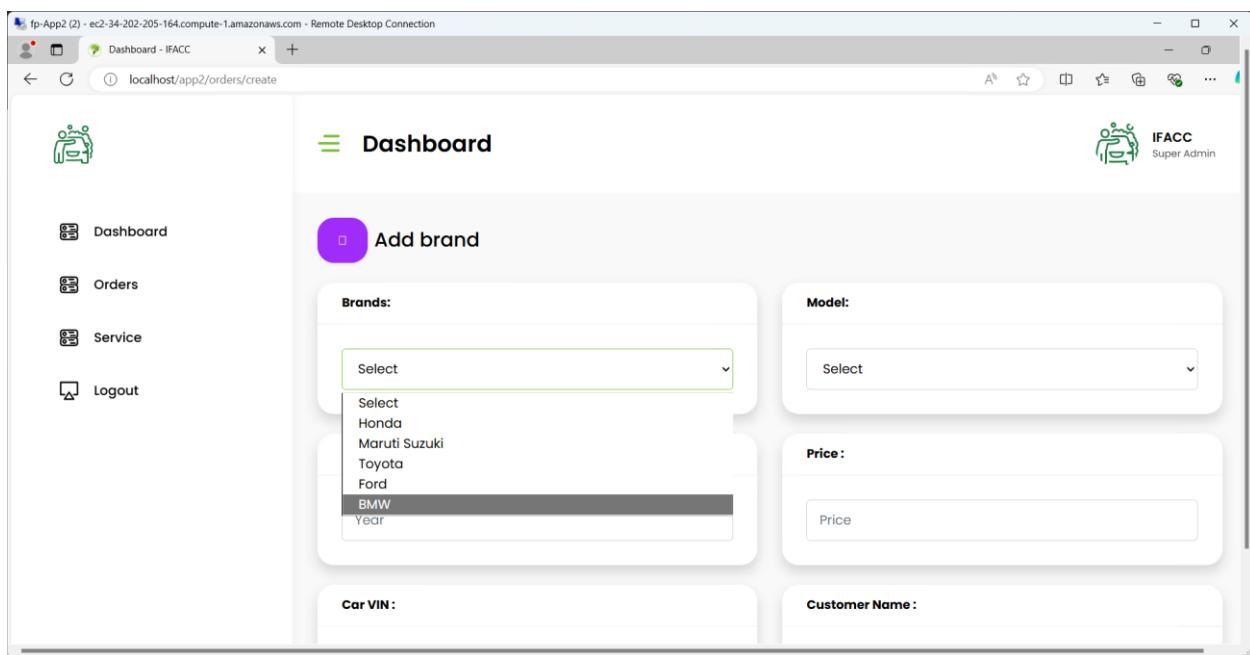


Figure 26 - A new brand BMW added on app1 is reflected on the app2

APP 3

Created the EC2 instance for app3. Added the outbound and inbound rules for EC2. Downloaded and installed XAMPP. Then we started the Apache and Mysql in XAMPP. Edited the env file added the db host details of app 2. Then launched the app3 in localhost. Successfully accessed the web application. I entered the vin ID and got the details of the corresponding ID.

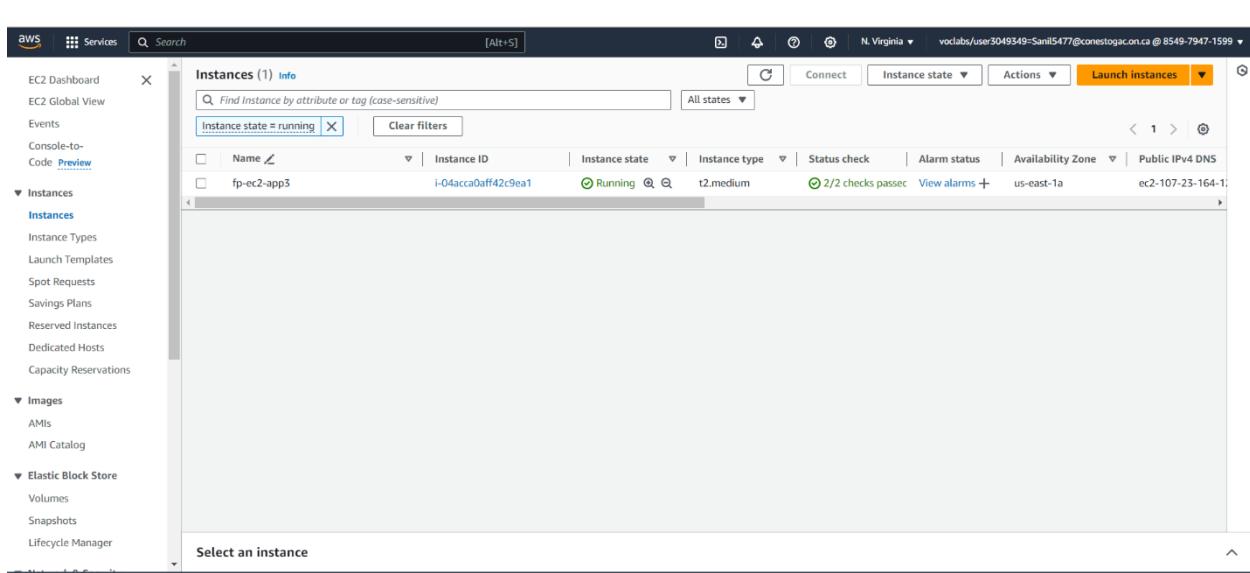


Figure 27 - Created an instance for app3

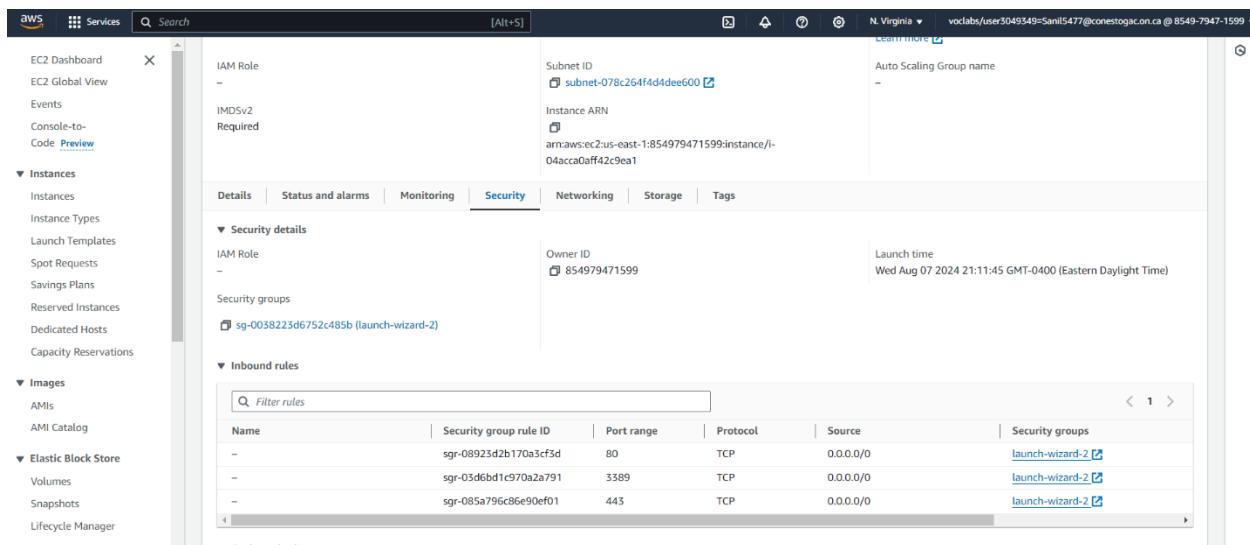


Figure 6 - Inbound rule set for the ec2 instance

```

.env - Notepad
File Edit Format View Help
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:2pWfo6IIis6XlyT4Un59G0qB1hF4fa2nLpIHAP21BvA=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=app2.csg8csdx0lld.us-east-1.rds.amazonaws.com
DB_PORT=3306
DB_DATABASE=app2
DB_USERNAME=admin
DB_PASSWORD=Secret55

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

<

```

20 items 1 item selected 746 bytes

Ln 1, Col 1 100% Unix (LF) UTF-8

Hostname: EC2A
Instance ID: i-04a
Private IPv4 address: 172.31.16.2.r
Public IPv4 address: 40.116.160.160

Figure 12 - Modified the .env file

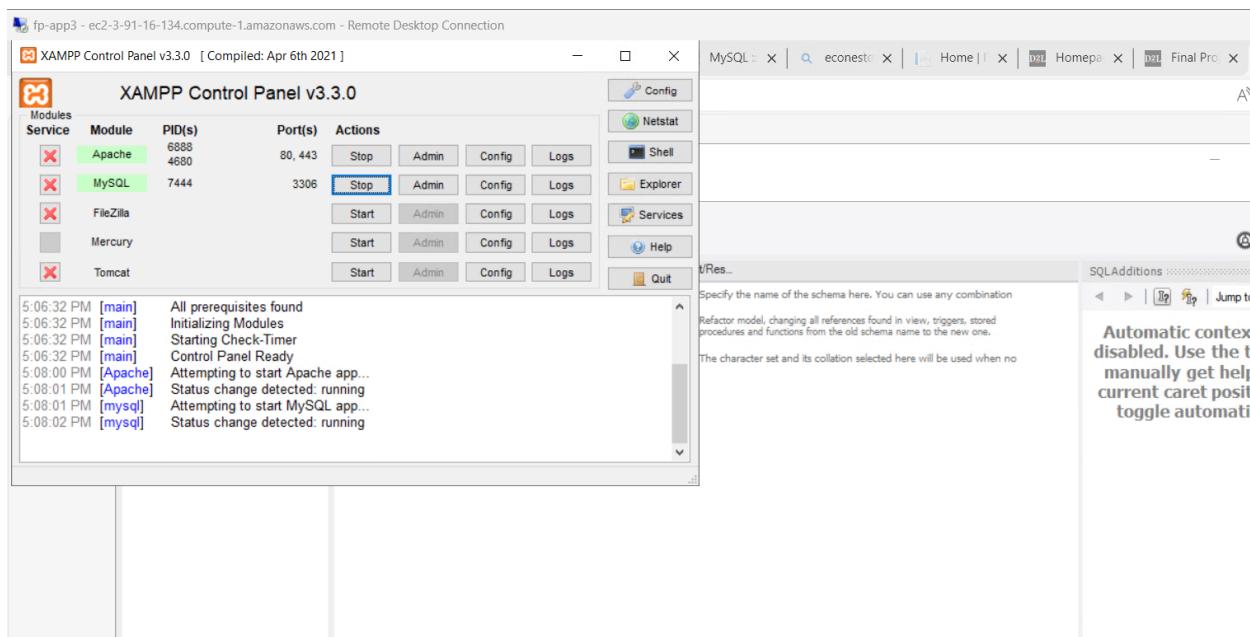


Figure 13 -Displaying the status of xampp

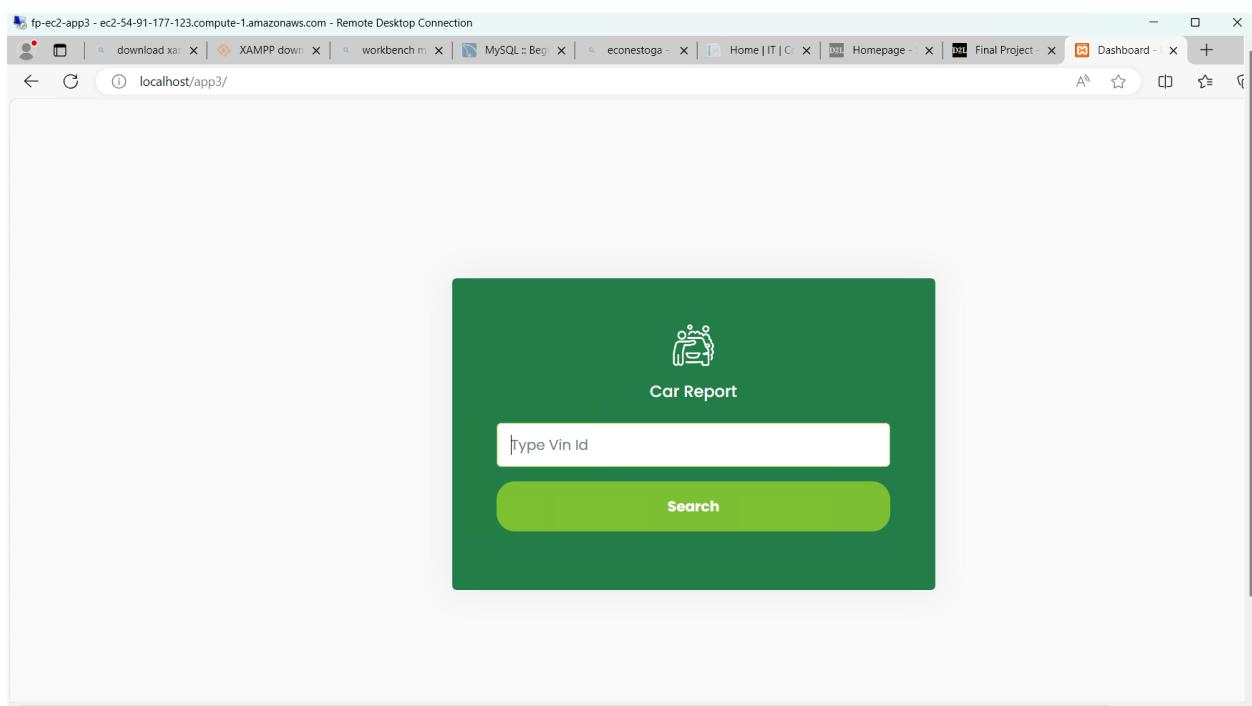


Figure 14 -Displaying the localhost/app3

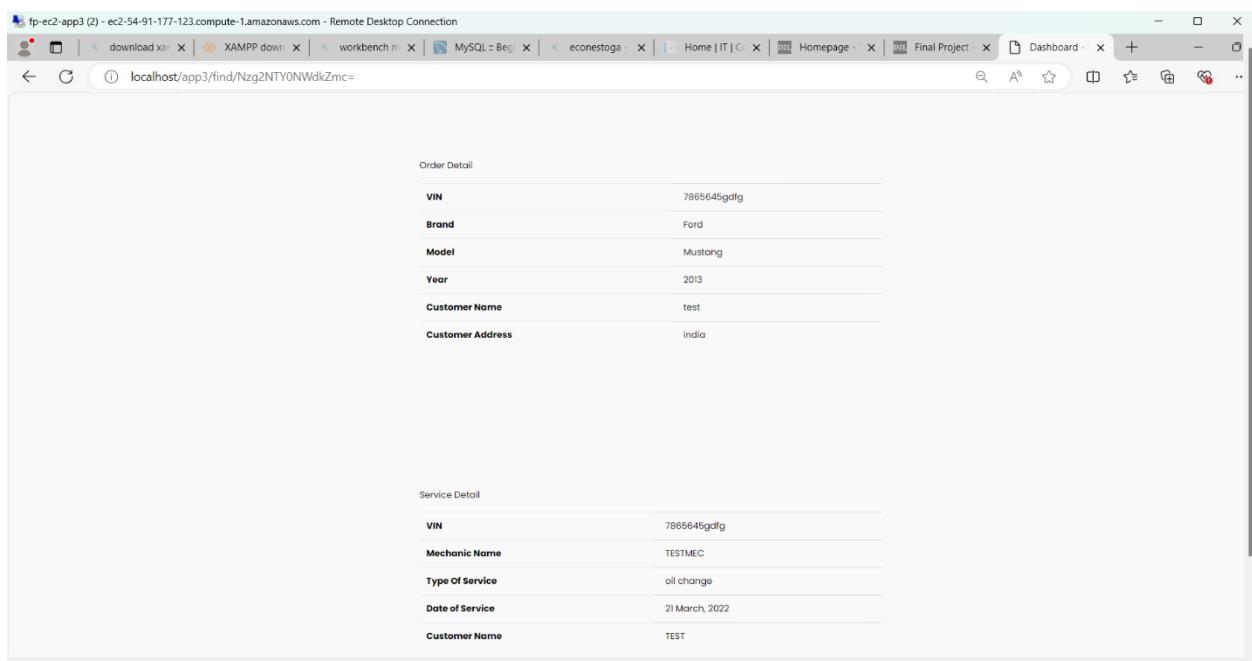


Figure 15 – Entered the vin id and got the order details.

Reflections

Based on the instructions provided, three integrated applications on the AWS are ready. This process includes creating EC2 instances, creating environments, migrating applications, and configuring RDS databases. Every application was hosted on another AWS account to have a distributed and scalable structure.

Key Achievements:

App1: We used first AWS account for implementing the App1. Car Manufacturing was established on an EC2 instance that came as a package with an RDS database, meaning that the root for storing data was created.

App2: We used second AWS account for implementing the App2. Car Dealership was launched on another EC2 instance having reference to its database from RDS and was also successfully integrated with App1 to fetch data from it.

App3: we used third account to show the integration part. App3 known as Car Details was of web application nature and was installed with Docker at an EC2 instance, intended for pulling and presenting information from both App1 and App2.

By correct arrangements of security groups, environment, and network settings, each application now communicates, which forms an extremely reliable system that mimics real-world car manufacturing, dealers, and detailed car data collection. This deployment proves the usage of AWS services in the case of the distributed applications and the understanding of the management of complex integrations in the above-mentioned cloud environment.