Order Creation API👾

by team FiveMusketeers

Team members👥: Elsa Doan, Hamsini Ganesan, Iris Zhang, Sanjay Bharath & Shreya Verma

Concept🔗

Our goal is to simplify the exchange of complex documents between buyers and sellers through an **affordable, user-friendly** solution that enables SMEs to connect and collaborate with other businesses, trade networks, and industry organisations online.

🎯Target audience & User needs

User Type	Description
Buyers🤑	SME retailers or procurement officers who regularly place bulk stock orders.
Suppliers💪	Distributors or wholesalers managing customer orders and inventory.
Business Admins👩🏻‍💻	Managers who want visibility over business operations and order history.

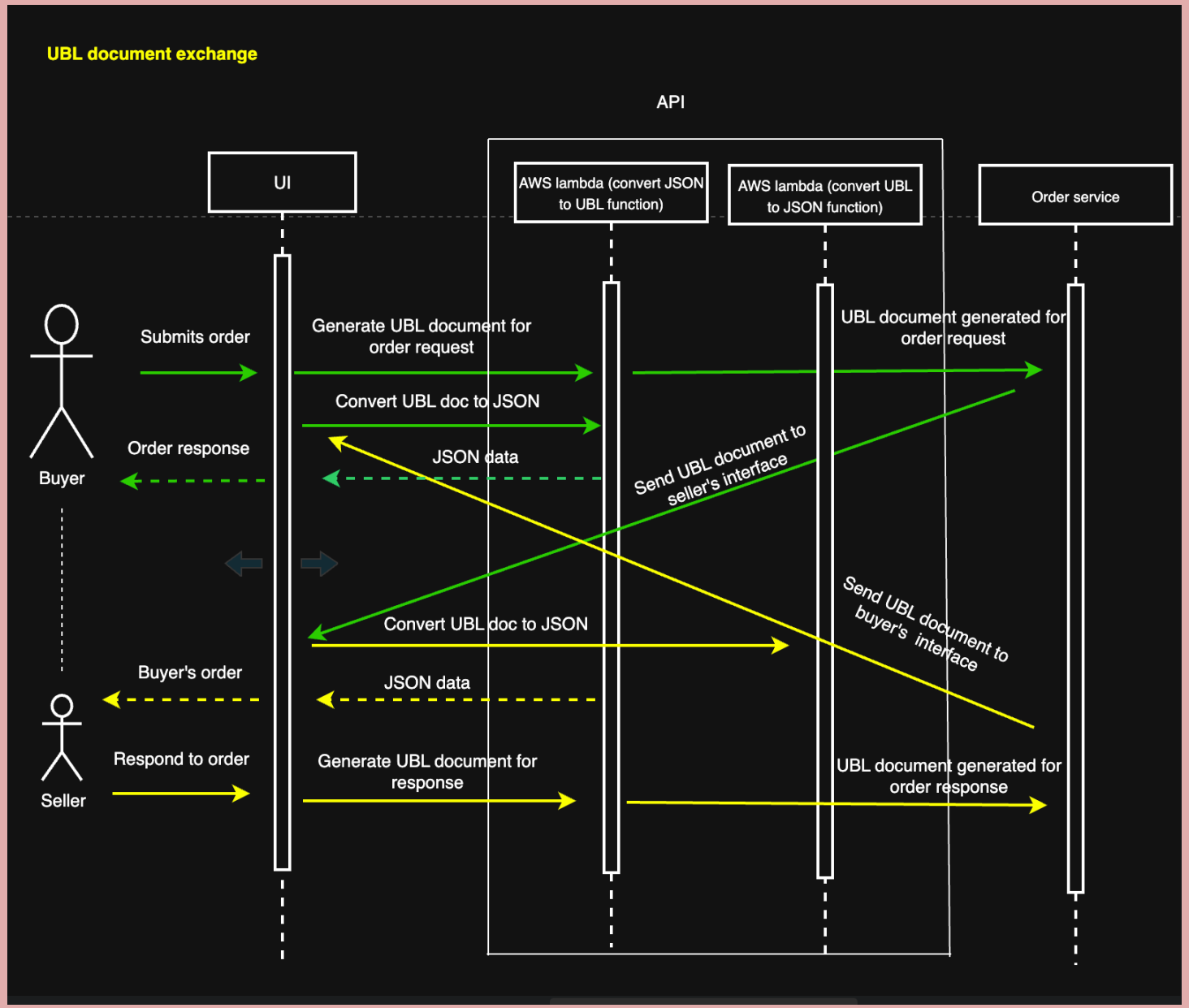
How does it provide business value to users? 💰

1. **Efficiency:** Streamlines the creation of procurement documents by automating processes, significantly reducing manual effort.
2. **Compliance:** Adheres to the **standardized UBL 2.1 format**, ensuring regulatory and industry compliance.
3. **Flexibility:** The API's modular design enables businesses to tailor workflows and seamlessly integrate with existing systems.
4. **Cost-Effective:** Offers a budget-friendly alternative to traditional, high-cost procurement platforms.

User Case 1 📝

Theory: As a SME procurement manager, I want to streamline communication between different enterprises by automatically generating standardised UBL documents, so that procurement processes are more efficient, accurate and, interoperable.

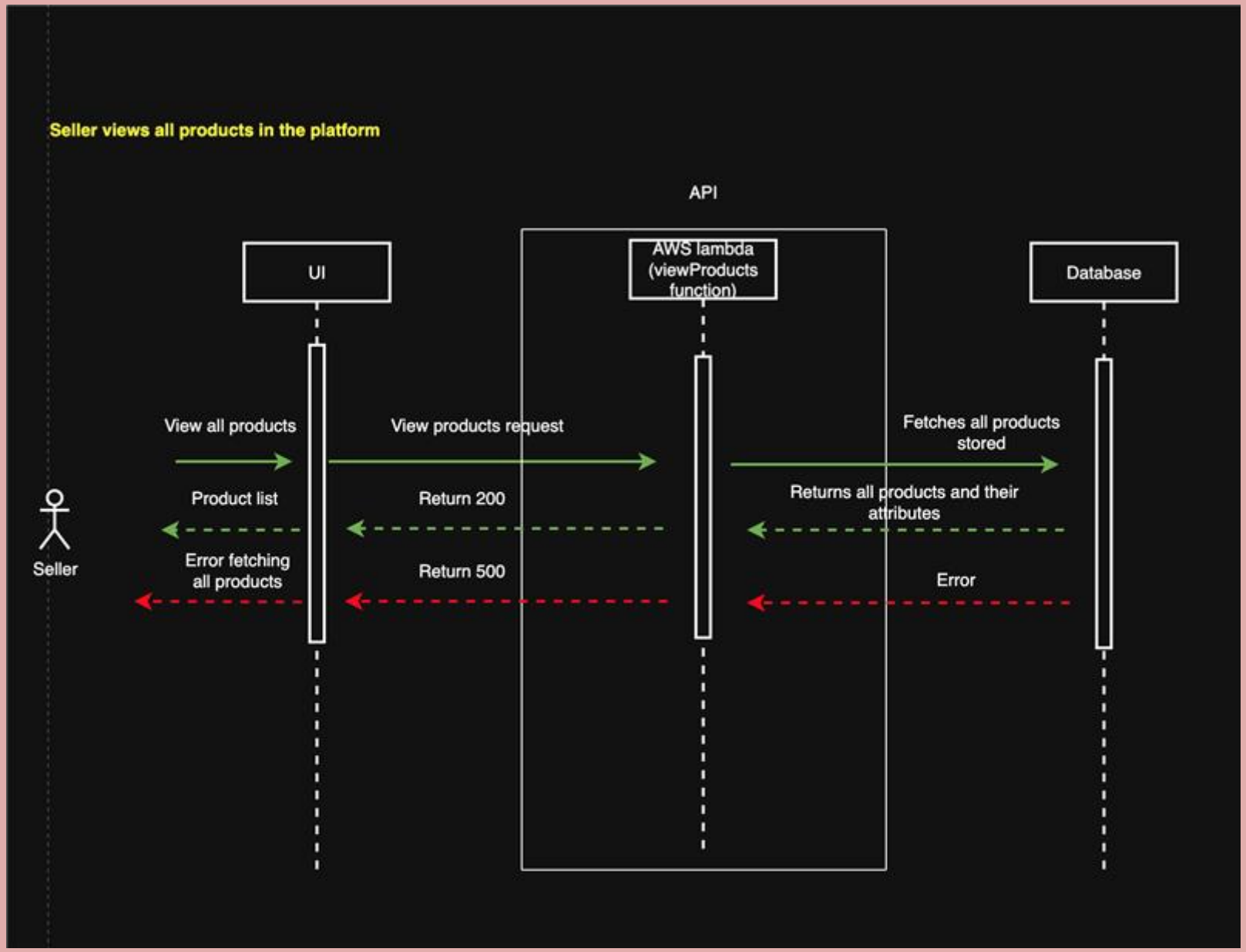
Sequence Diagram 📊:



User Case 2 

Theory: As a seller, I want to view a list of all products I’ve added to the platform so that I can manage inventory, check availability and update information if needed.

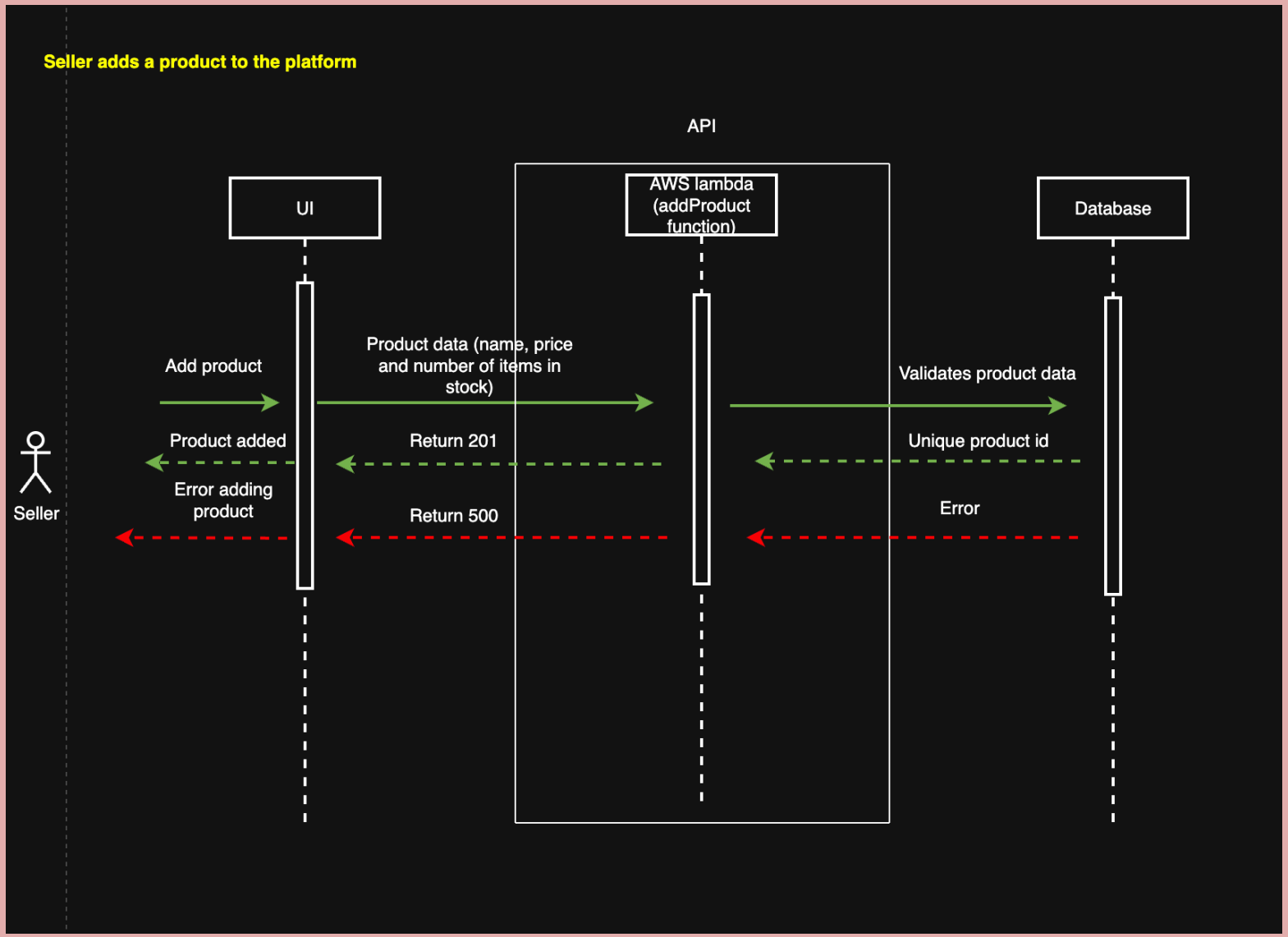
Sequence Diagram 📊:



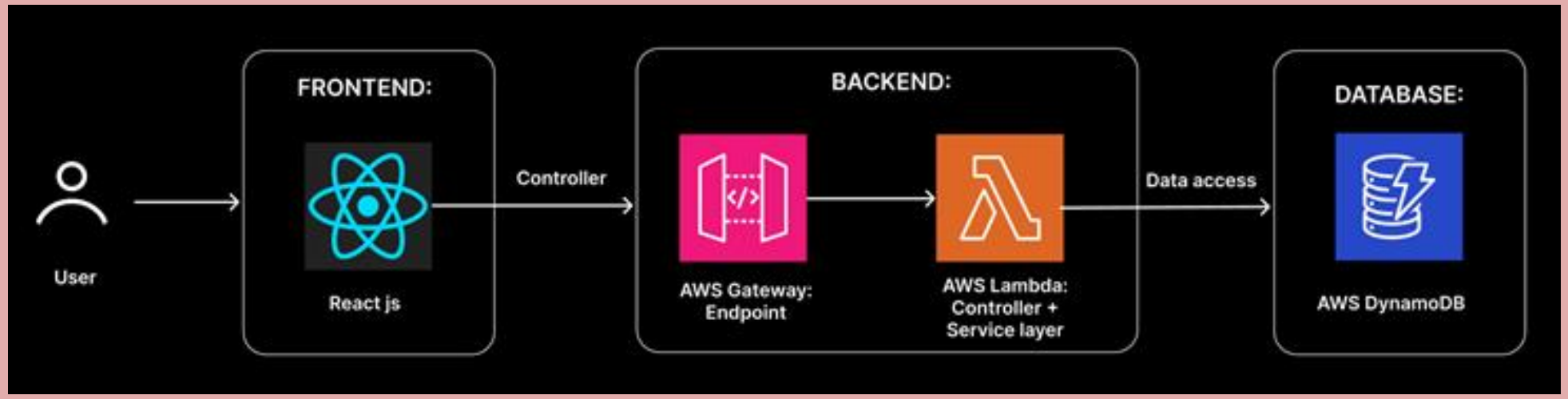
User Case 3 

Theory: As a seller I want to be able to add new products to the platform by submitting product information so they can become available for customers to purchase. This way the system can automatically generate order response according to the newly updated information about.

Sequence Diagram 📊:



Software Architecture Design📈

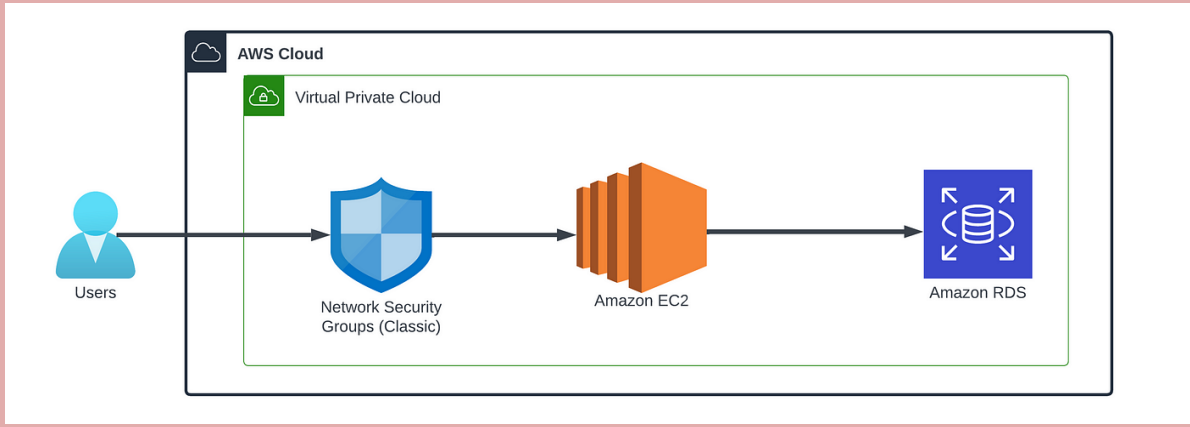


Live API Demonstration📲

Live Frontend Demonstration 

Technical challenges & constraints faced🔍

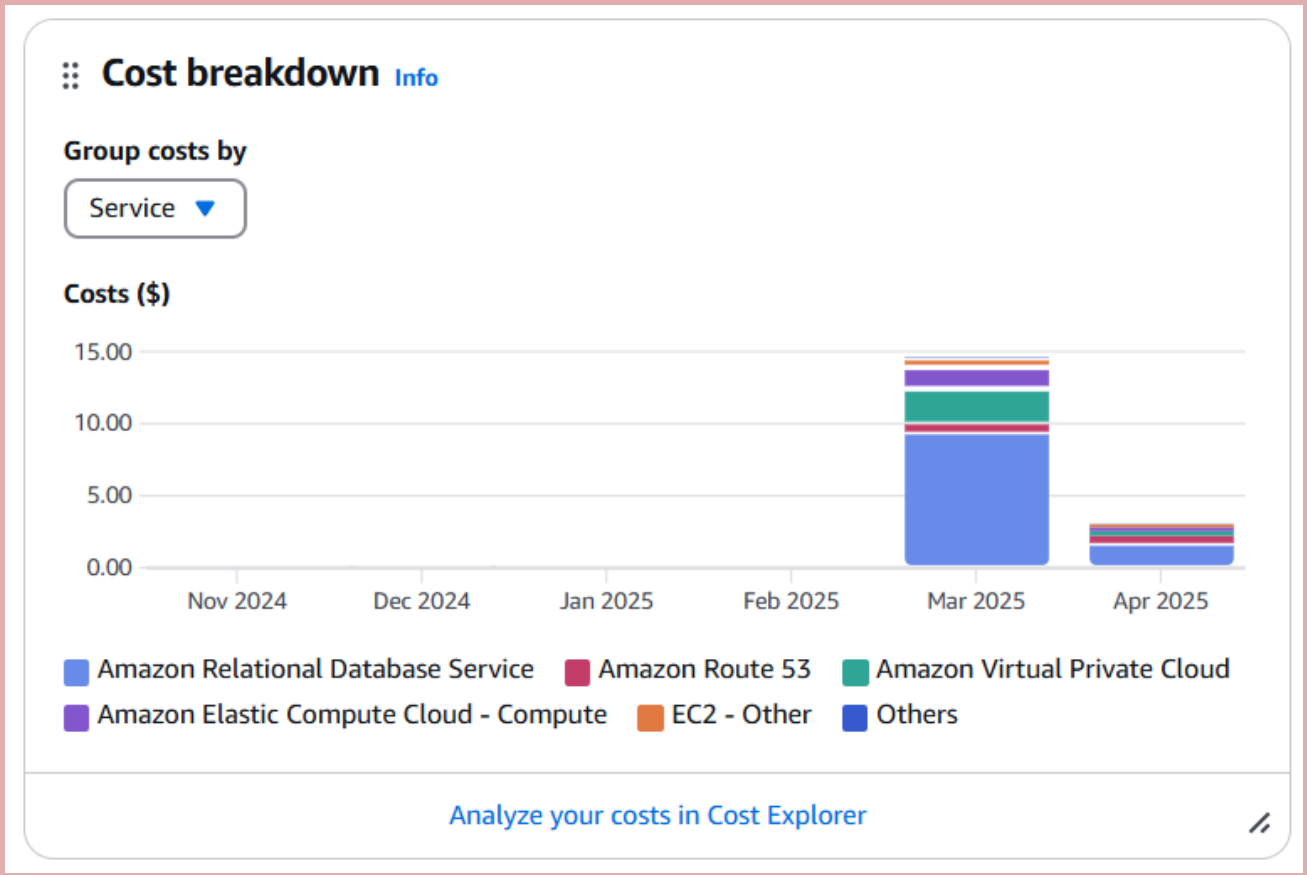
1. Connecting PostgreSQL to EC2 & testing.



Using the EC2 Instance.txt

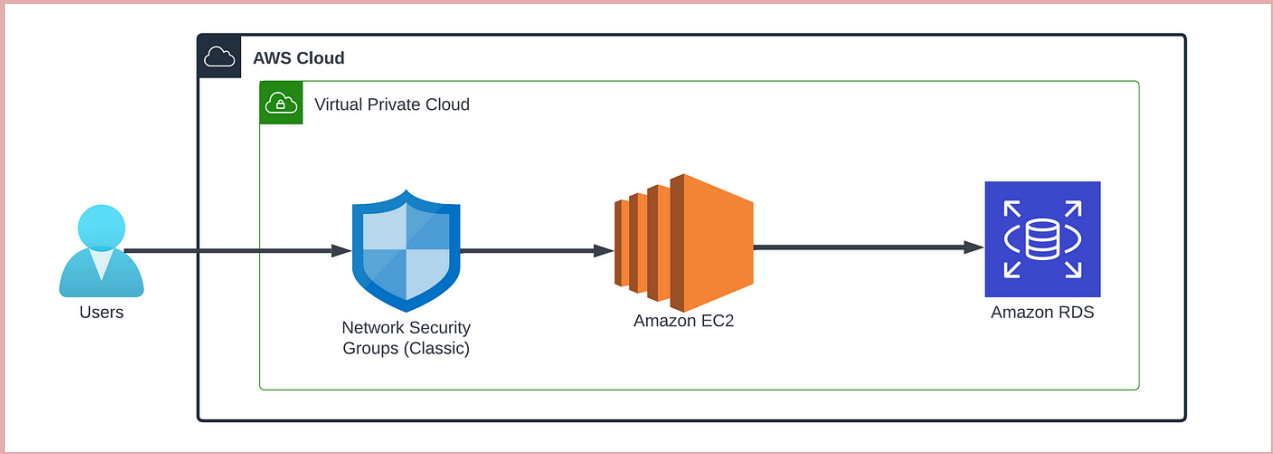
```
// to sync code into teh EC2 instance (may take aboy 5 seconds)
1. rsync -avz --exclude 'node_modules' --exclude '.git' --exclude '.env' -e "ssh -i ~/FiveMusketeers/.ssh/Musks.pem" .
ubuntu@ec2-34-201-134-164.compute-1.amazonaws.com:~/app
1.1 check the EC2 Instance is active
1.2 will need to alter (link --> ec2 instance --> Public IPv4 DNS)

// to connect to teh EC2 instance
2.-1 cd .ssh
2. ssh -i "Musks.pem" ubuntu@ec2-34-201-134-164.compute-1.amazonaws.com
2.1 will need to alter (link --> ec2 instance --> Public IPv4 DNS)
2.2 exit (exit)

// to start the server while connected to the database
3. DB_URL='postgresql://postgres:seng2021@musksdb.cw5opdtderna.us-east-1.rds.amazonaws.com:5432/order_creation' npm run dev
3.1 change URL IP to the current (EC2 Instance --> Public IPv4 address)
3.2 control + C (exit)

// to access the database and
4. psql --no-psqlrc -h musksdb.cw5opdtderna.us-east-1.rds.amazonaws.com -U postgres -d postgres
4.1 password: seng2021
4.2 \c order_creation (connect to order creartion database)
4.3 truncate customers, orderdetails, orderitems, products; (removes all data)
4.4 \q (quit)
```

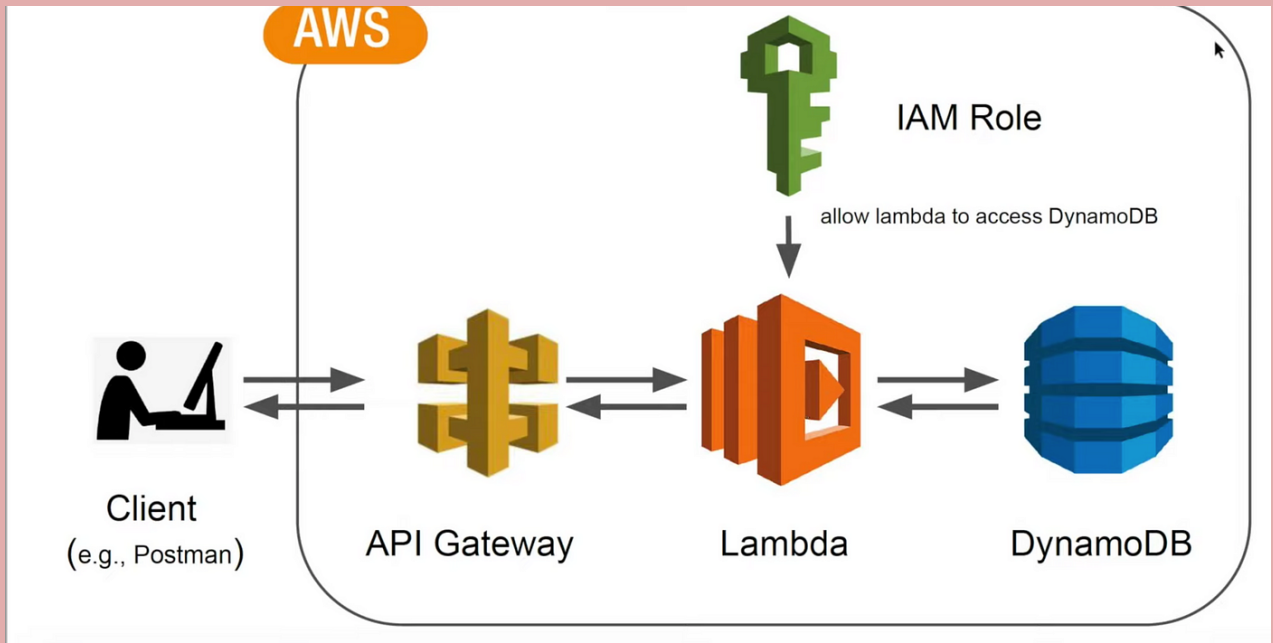
2. Deploying code on the EC2 instance.



3. Connecting the AWS lambda functions to AWS DynamoDB (the database).

4. EC2 instance to AWS lambda functions & API endpoints using AWS Gateway.





Going forward..💥

Thank you ✨future investors!✨