AWS Step Functions for Workflow Automation

A Course Project Report Submitted in partial fulfillment of the course requirements for the award of grades in the subject of

CLOUD BASED AIML SPECIALITY
(22SDCS07A)

by

K. Poojitha

2210030123

Under the esteemed guidance of

Ms. P. Sree Lakshmi
Assistant Professor,
Department of Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

K L Deemed to be UNIVERSITY

Aziznagar, Moinabad, Hyderabad,
Telangana, Pincode: 500075

April 2025

K L Deemed to be UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Certificate

This is Certified that the project entitled **"**AWS Step Functions for Workflow Automation Functions**"** which is a Experimental work carried out by K. Poojitha (2210030123), in partial fulfillment of the course requirements for the award of grades in the subject of **CLOUD BASED AIML SPECIALITY**, during the year **2024-2025**. The project has been approved as it satisfies the academic requirements.

Ms.P.Sree Lakshmi

Course Coordinator

Dr. Arpita Gupta

Head of the Department

Ms. P. Sree Lakshmi

Course Instructor

CONTENTS

1. INTRODUCTION

The AWS Step Functions for Workflow Automation project is a simple, scalable solution that uses AWS Step Functions to automate business workflows without the need for manual intervention. This makes processes faster, more accurate, and easier to manage. AWS Lambda is used to run the business logic, while Step Functions help control the flow of tasks, making sure everything works together smoothly.

The system is built to be reliable and flexible, so it can handle changes and scale easily as needed. It also provides visual workflows and error handling, helping developers track and manage tasks, even when the workflows get complex[5].

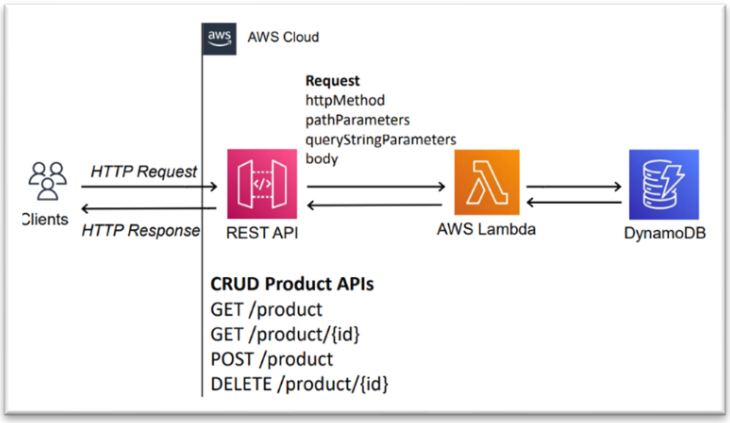
To manage data securely and efficiently, Amazon DynamoDB is used for storing and organizing information. IAM controls who can access the system, ensuring only authorized users can make changes or interact with the workflows. Additionally, CloudWatch is integrated for real-time monitoring and logging, providing visibility into the system's performance and enabling prompt detection and resolution of issues.

By using these AWS services, the system is easy to set up, reduces manual work, and scales automatically, making it perfect for companies that want to improve their operations using cloud technology.

2. AWS Services Used as part of the project

The project utilizes several AWS services to build a serverless architecture:

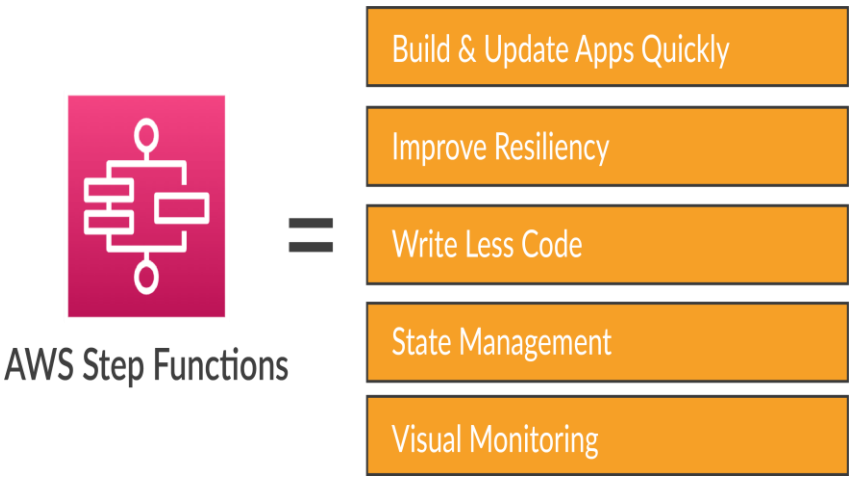
1. Amazon DynamoDB [3]:
    - A NoSQL database used to store task details such as task ID, status, and owner.
    - Provides scalable and low-latency data access.
    - Supports efficient querying and indexing for rapid results.
    - Ensures high availability and fault tolerance.



2. AWS Lambda [2]:
    - Executes business logic for each step.
    - Processes input/output and updates task states
    - Interacts with DynamoDB and other services.
    - Automatically scales to meet workload demands.

3. AWS Identity and Access Management (IAM) [4]:
    - Defines permissions and access roles for Lambda and Step Functions.
    - Ensures secure, role-based resource access.
    - Provides fine-grained access control for various services.
    - Logs API calls using AWS CloudTrail for auditing purposes.

4. AWS Step Functions [1]:
    - A serverless service that automates workflows by coordinating multiple AWS services.
    - Provides a visual interface to define and manage workflows.
    - Supports both sequential and parallel execution of tasks.
    - Automatically handles errors and retries to ensure reliability.
    - Integrates with AWS services like Lambda, DynamoDB, and others for efficient workflow management.



5. Amazon CloudWatch:
    - Monitors AWS services and application logs.
    - Sends alerts in case of anomalies or execution failures.
    - Helps in setting up metrics and dashboards for tracking.
    - Assists in debugging and system optimization.

3. Steps Involved in Solving the Project Problem Statement

This project uses AWS Step Functions and a serverless setup to automate task processing. The following steps were taken to implement the solution:

1. Create Lambda Functions:
    - In AWS Console, go to Lambda and create four Lambda functions: orderPlacement, paymentProcessing, shippingProcess, and orderConfirmation.
    - Write and upload the source code for each function based on its purpose.
    - Adjust runtime settings if needed and deploy all functions.
2. Create a table in DynamoDB:
    - A DynamoDB table was created with a primary key (like order_id) to store each order's details clearly and uniquely.
    - It helps in saving and retrieving data fast, making it easy to track order status during workflow execution.
3. Set Up IAM Roles:
    - Go to IAM > Roles.
    - Create the following roles with needed permissions:
        - StepFunctionLambdaRole – let Step Functions call Lambda.
        - LambdaExecutionRole – gives permissions to Lambda functions.
        - StepFunctionExecutionRole – allows Step Functions to run workflows.
    - Make sure each role has correct trust policies.
4. Create Step Functions State Machine:
    - Go to Step Functions > Create State Machine.
    - Define flow using Amazon States Language.
    - Use StepFunctionLambdaRole as execution role.
    - Deploy and test the workflow.
5. Run the Workflow:
    - Start execution with input as JSON.
    - Check real-time progress and transitions.
    - Make sure each step gives expected output.
    - Validate output results to ensure logic flows correctly.
6. Check Output and Logs:
    - Use CloudWatch to view logs for all functions.
    - Confirm all paths run correctly and edge cases work fine.
    - Check execution time and resource usage.

4. Stepwise Screenshots with Brief Description

Step 1: Lambda Function Creation

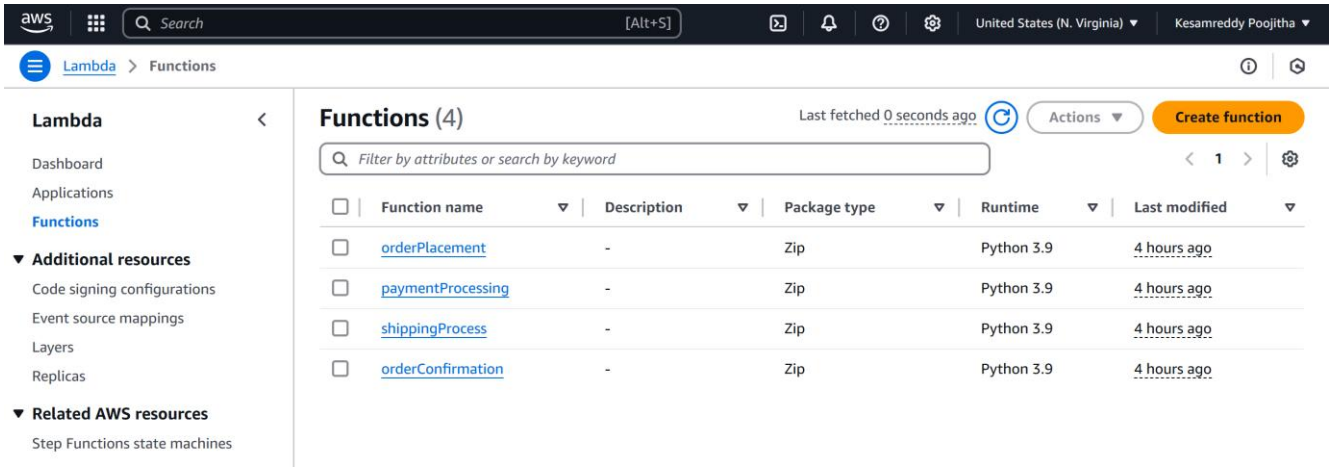


Fig: 4.1: Lambda Function Creation

The Lambda dashboard is used to create four separate functions: orderPlacement, paymentProcessing, shippingProcess, and orderConfirmation.

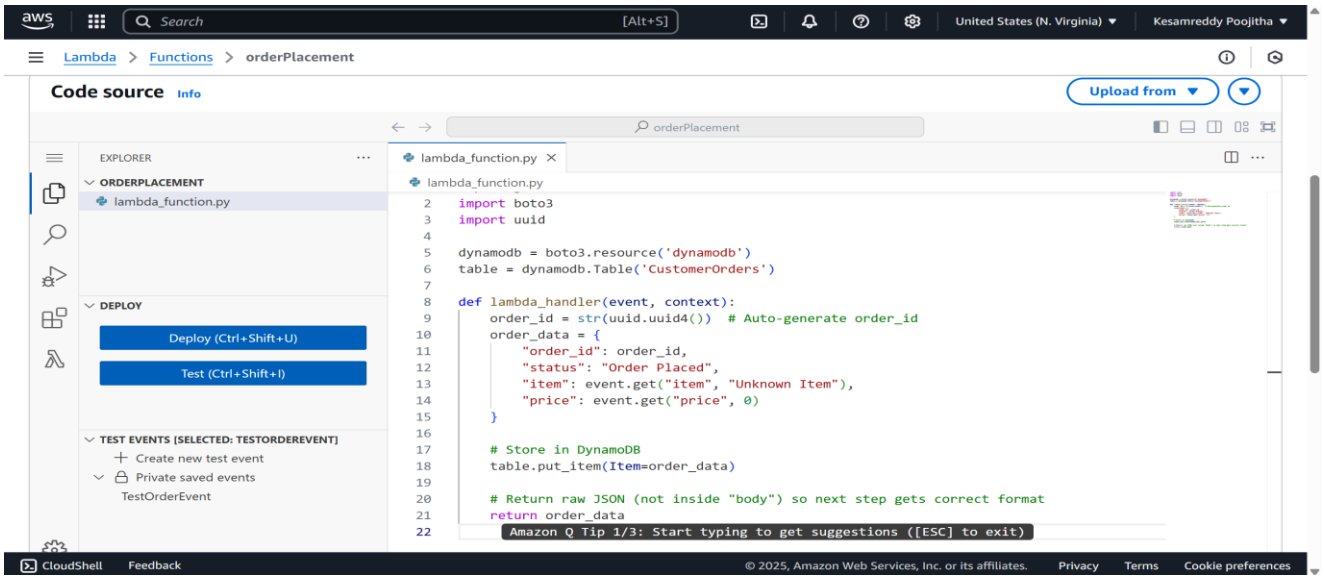


Fig 4.1.2 Source code in Lambda function

Each function, such as the above one for orderPlacement was uploaded with its respective logic, tested individually, and deployed for integration into the workflow.

Step 2: DynamoDB Table

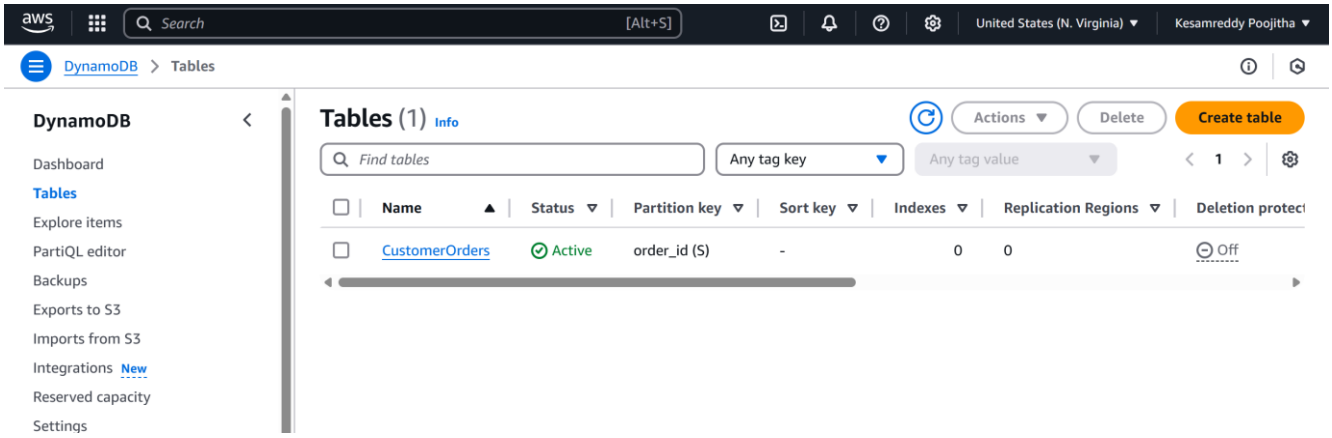


Fig: 4.2: DynamoDB Table

This screenshot shows the DynamoDB table which was created to save order details like order ID, status, and time. It helps store and get data quickly and easily during each step of the process.

Step 3: Attaching DynamoDB Permissions to IAM Role

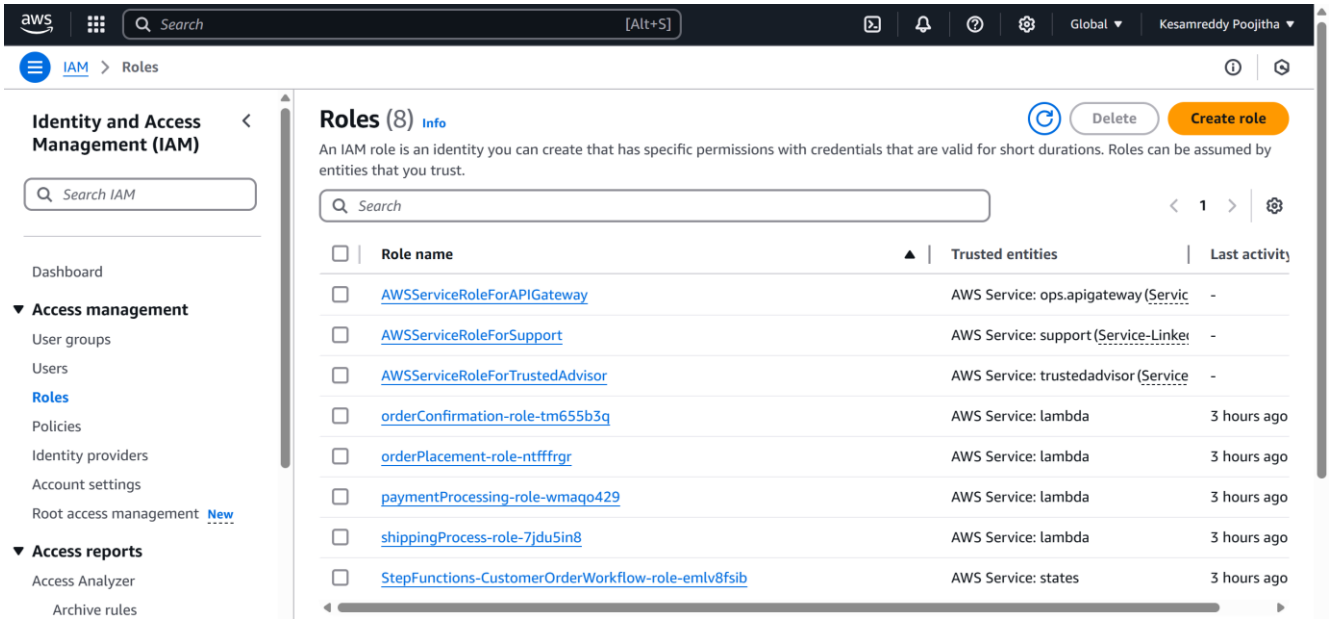


Fig: 4.3: Attaching DynamoDB Permissions to IAM Role

This screenshot shows the IAM role being given permissions to access DynamoDB. Actions like PutItem, GetItem, and UpdateItem are allowed so that Lambda functions can safely read from and write to the DynamoDB table during the workflow.

Step 4: Creating State Machine in Step Functions

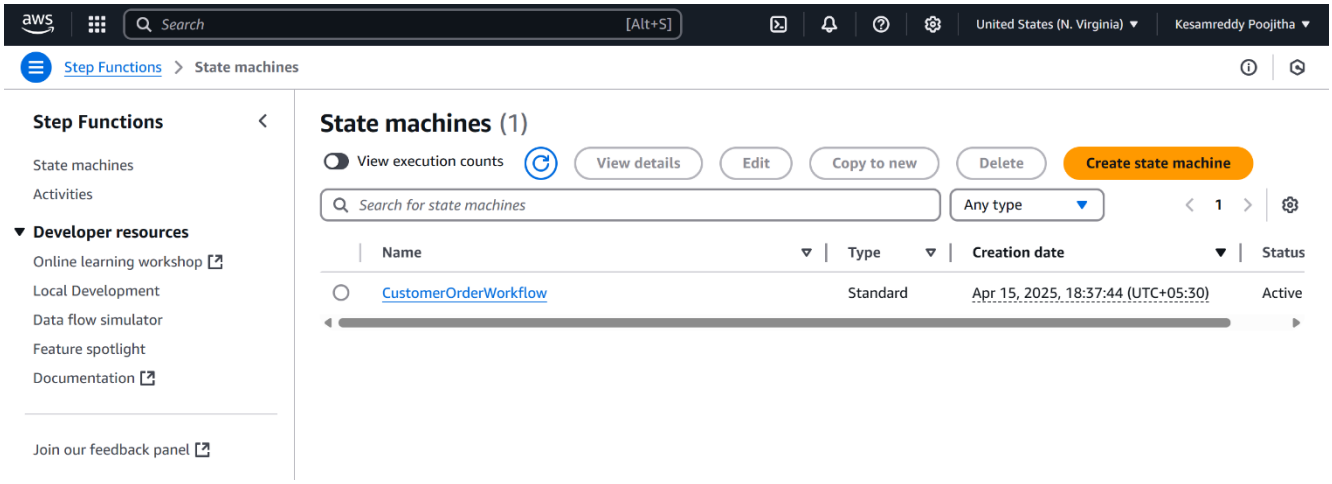


Fig: 4.4: Creating State Machine in Step Functions

This screenshot shows the process of creating a new State Machine in AWS Step Functions, where the workflow is set up and configured to automate the task sequence.

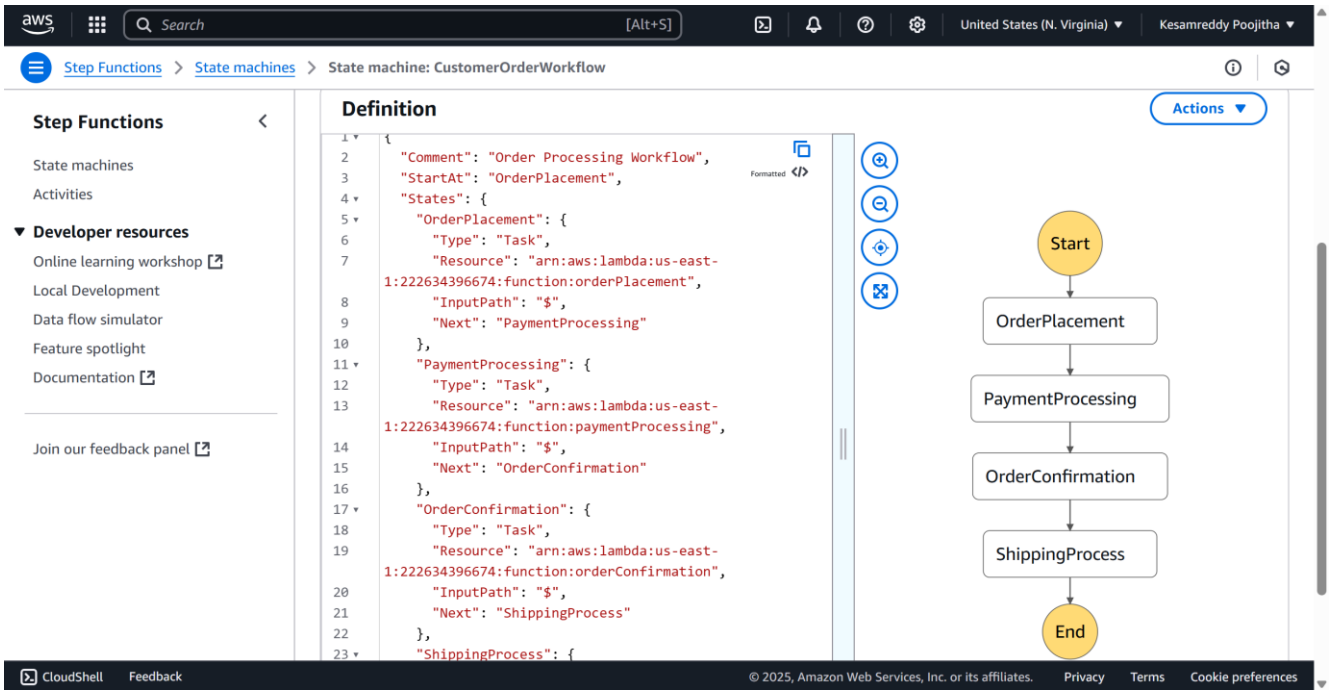


Fig: 4.4.1: Writing Code in Step Functions State Machine

This screenshot shows the creation of a State Machine where the workflow is defined using Amazon States Language (ASL). The Lambda functions (orderPlacement, paymentProcessing, shippingProcess, and orderConfirmation) are linked in a sequence to automate the task processing flow.

Step 5: Final output

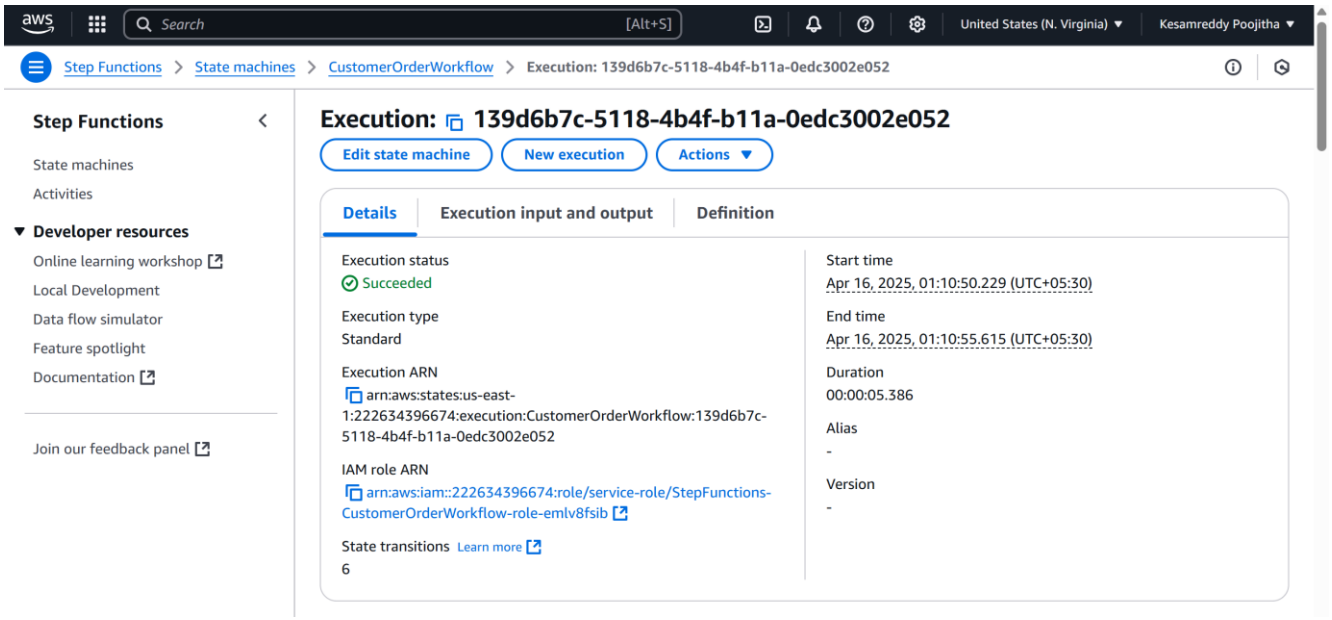


Fig: 4.5.1: Final output

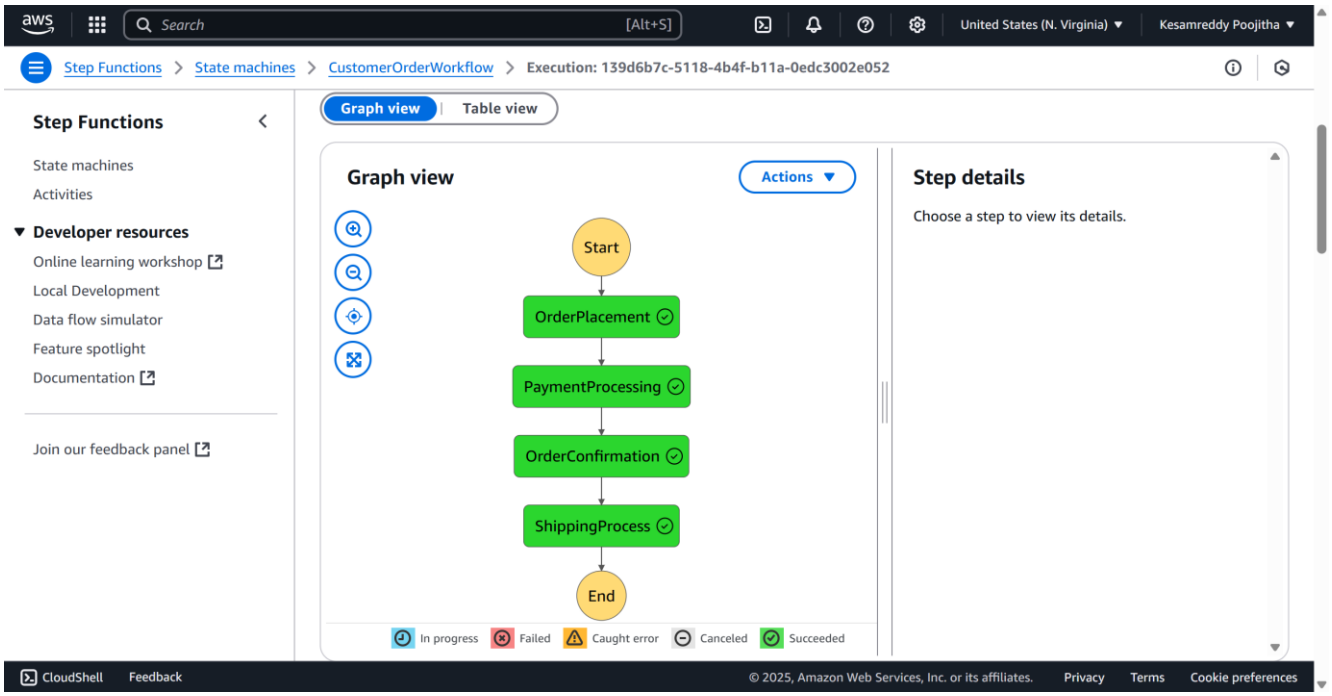


Fig: 4.5.2: Final output

This screenshot shows the workflow's final output displayed in a graph format, visualizing the sequence and outcomes of each function. The graph highlights the step-by-step execution of the integrated functions, offering a clear, visual understanding of the entire process.

5. Learning Outcomes

This project provided several key learning outcomes:

1. **Understanding Serverless Architecture**:
    - Learned to build a serverless application using AWS Step Functions and Lambda.
    - Understood the benefits such as cost efficiency and scalability in designing modern workflows.

2. **Working with AWS Services**:
    - Gained hands-on experience with AWS Step Functions, Lambda, IAM, and CloudWatch.
    - Understood how cloud services interact to build efficient applications.

3. **IAM Role Configuration**:
    - Developed skills in Creating and assigning policies to ensure proper role-based permissions for workflow execution.
    - Learned the importance of IAM roles in securing AWS resources.

4. **State Machine Design and Execution**:
    - Learned to design workflows using Amazon States Language (ASL) and monitor them during execution.
    - Also learned to validate correct task sequencing and error handling during state transitions.

5. **Best Practices in Development**:
    - Learned the importance of error handling, logging, and data serialization in serverless applications.
    - Gained experience in packaging and deploying Lambda functions with dependencies.

6. Conclusion

This project demonstrates the efficient use of AWS Step Functions to automate and manage a sequence of serverless tasks. Each Lambda function was developed with its own logic, thoroughly tested, and then integrated into a centralized workflow. By using Step Functions, the execution of tasks became more organized and easier to manage, reducing errors and improving overall reliability. This approach also enhances scalability, maintainability, and visibility of the system, making it well-suited for handling real-time business processes. Additionally, the graphical workflow provided by Step Functions simplifies debugging and monitoring. The modular design of individual functions allows for faster updates and easier troubleshooting. Overall, this implementation ensures a streamlined, efficient, and future-ready solution for complex workflows. The project showcases how cloud-native solutions can optimize backend processes with minimal overhead. It also highlights the importance of clear orchestration in improving system resilience. This model can be further extended to support more advanced use cases with minimal changes to the core logic.

7. References

[1] AWS Step Functions Documentation:
https://docs.aws.amazon.com/step-functions/latest/dg/welcome.html

[2] AWS Lambda Developer Guide:
https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

[3] Amazon DynamoDB Guide:
https://docs.aws.amazon.com/dynamodb/latest/developerguide/Introduction.html

[4] AWS IAM Documentation:
https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html

[5] AWS General Documentation:
https://docs.aws.amazon.com/