Capstone Project

Cohort 11 - Group 4

Documentation

Project Overview

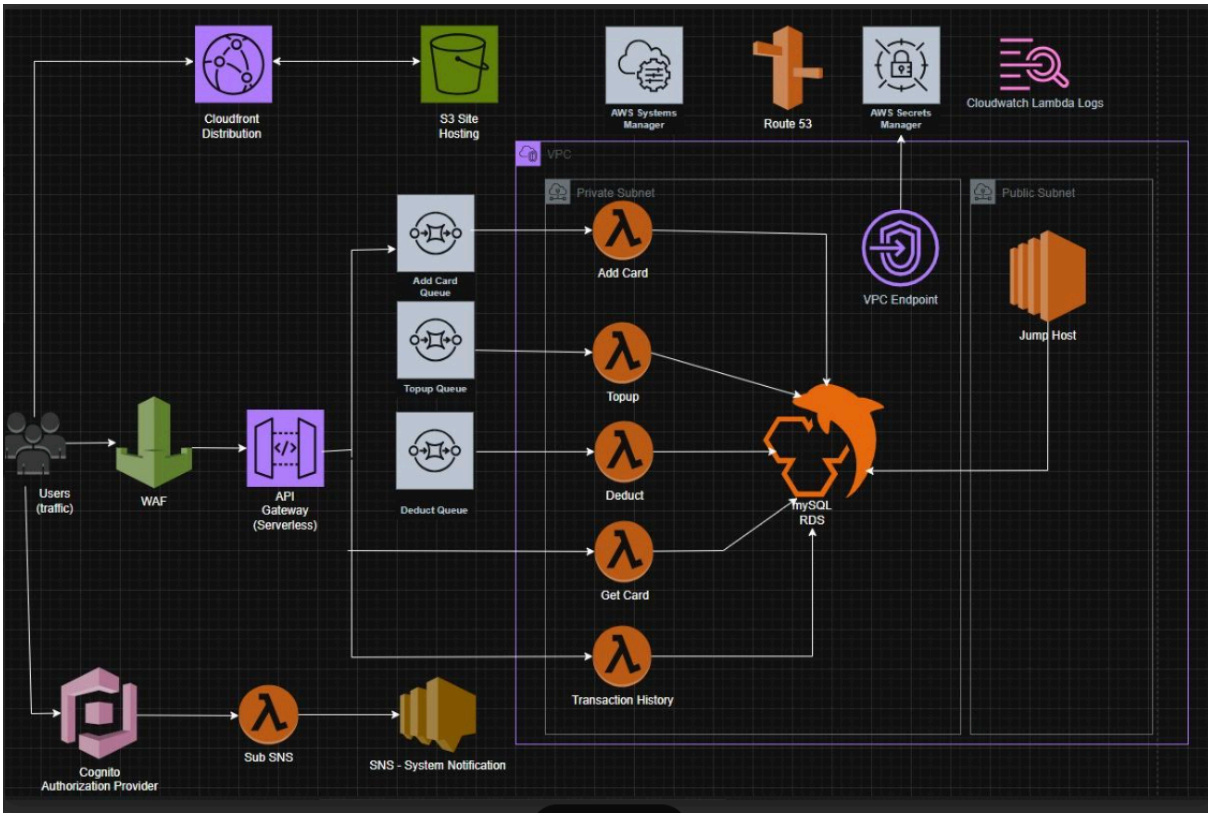
Project Name: Card-Based Payment System

This project aims to design and implement a **Continuous Integration and Continuous Deployment (CI/CD) pipeline** for a cloud-native application**.** The application is a cloud-based Card-Based Payment System built on Amazon Web Services (AWS), featuring a serverless backend, a frontend that uses the Angular framework, and a secure infrastructure managed via Terraform.

The current features of the application are:

- User sign-up / login (consumers)
- Top-up wallet of the card
- Payment/deduction using the card
- Display of Transaction History
- Audit and Monitoring

Architecture Overview



High-Level Architecture and Components

1. Infrastructure:

- VPC with public/private subnets
- EC2 Jump Host in public subnet for DB management
- All infrastructure provisioned via Terraform

2. Frontend

- S3 Bucket for Website Hosting
- Hosted on S3 with CloudFront for CDN (Content Delivery Network) and HTTPS)
- Route 53: DNS management and domain routing
- CORS (Cross-Origin Resource Sharing) is configured at the API Gateway level
- A user interface (UI) layer of a web application built using Angular
- Cognito User Pool for authentication

3. Backend:

- API Gateway (Serverless): REST API endpoints for client communication
- AWS Lambda: Serverless compute for business logic
- SQS queues for asynchronous processing, reliability, scalability, and fault tolerance
- Cloudwatch Logs

4. Database

- Aurora MySQL RDS for better performance, availability, and scalability than standard MySQL, and without managing database infrastructure
- Managed in a private subnet for security, isolation, and controlled access

- Aurora MySQL credentials are securely stored by AWS Secrets Manager

5. CI/CD

- CI: Create AWS Resources using Terraform
- CD: GitHub Actions -> deploy to production via Workflow Dispatch

6. Security Considerations

- VPC with private/public subnets
- RDS in private subnet
- Secrets Manager for credentials
- Cognito for authentication
- API Gateway authorization
- WAF for CloudFront

Github Repository Setup

1. Secrets (GitHub Secrets):

- AWS_ACCESS_KEY_ID_NTU: your-access-key-id
- AWS_SECRET_ACCESS_KEY_NTU: your-secret-access-key
- SNYK_TOKEN: your-snyk-token
- AWS_REGION: us-east-1
- DEV_ALLOWED_ORIGIN: *
- PROD_ALLOWED_ORIGIN: *

2. Variables (GitHub Variables):

- S3_BUCKET_FOR_TF_STATE_FILE: ce11-capstone-group4
- AZS: ["us-east-1a","us-east-1b"]

Database Schema

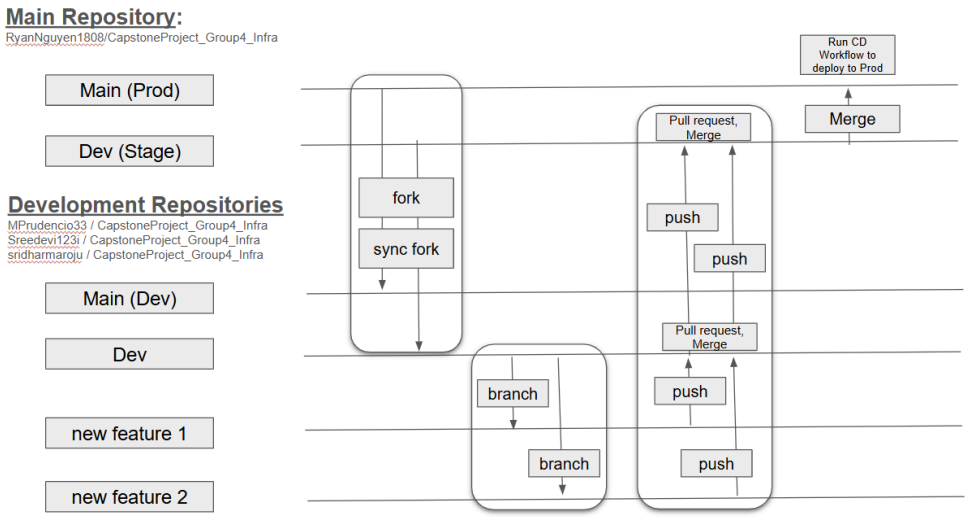
```
MySQL [ce11capstonegroup4AppDB]> desc TRANSACTIONS
    -> ;
+----------------------+-------------+------+-----+---------+----------------+
| Field                | Type        | Null | Key | Default | Extra          |
+----------------------+-------------+------+-----+---------+----------------+
| TRANSACTION_ID       | bigint      | NO   | PRI | NULL    | auto_increment |
| CARD_ID              | varchar(36) | NO   | MUL | NULL    |                |
| TYPE                 | smallint    | NO   |     | NULL    |                |
| AMOUNT               | double      | NO   |     | NULL    |                |
| TRANSACTION_DATETIME | datetime    | NO   |     | NULL    |                |
+----------------------+-------------+------+-----+---------+----------------+
5 rows in set (0.017 sec)

MySQL [ce11capstonegroup4AppDB]> desc CARDS;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| CARD_ID | varchar(36) | NO   | PRI | NULL    |       |
| USER_ID | varchar(36) | NO   |     | NULL    |       |
| BALANCE | double      | NO   |     | 0       |       |
+---------+-------------+------+-----+---------+-------+
3 rows in set (0.001 sec)

MySQL [ce11capstonegroup4AppDB]>
```

Key Tables:

- USERS - User profiles and authentication
- CARDS - Payment card information
- TRANSACTIONS - Add Card, Topup, Deductions

Development Setup & Deployment

1. Github Repositories and CI/CD Workflow



Initial Setup:

- Fork/clone the main repository
- Add all required Secrets and Variables to GitHub
- Run CI/CD pipeline to provision infrastructure

CI/CD Workflow:

- 2 branches (long live) for GitHub Repo
  - Main branch for Production Environment
  - Dev branch for Development Environment
  - Using Terraform Workspace to manage the environments
- Main and Dev are protected
- Changes introduced by pull requests
- CI runs on push and pull requests
- CD runs on *"workflow_dispatch"*
- DESTROY runs on *"workflow_dispatch"*

Illustration of deployment from dev to prod

New Feature: Include a column 'Balance' in the display of Transaction History

Steps:
In the development environment / repository (created by forking from the main (upstream) repository):

1. Create a new branch from the dev branch in the dev ("origin") repo
   $ git checkout -b feature/balance
2. Push the feature/balance to the origin repo
   $ git add .
   $ git commit -m "add feature/balance branch"
   $ git push origin feature/balance
3. In the development repo in Github, run the CD workflow for the feature/balance branch. This will create the resources (for the development environment) in AWS.
4. Using the AWS Console, open the lambda function ce11-capstone-group4-get-TransHist-api-sandbox-mprudencio33
5. In the Code Source, update the SQL statement to get the Balance from CARDS table by joining to the TRANSACTIONS table using the CARD_ID

Steps cont'd:

6. Push the feature/balance branch to the origin repository..
7. Create a Pull Request for the dev branch in the origin repository, review the changes and then do a merge of the dev branch to the main branch.
8. Run the CD workflow for the Main branch.
9. Go the application portal and go to the Transaction History menu/tab and enter a valid card id to see if the Balance value is now displayed for each of the transactions.

In the production environment / repository:

10. Create a pull request for the dev branch, review the changes and then do a merge of the dev branch to the main branch
12. Run the CD workflow for the 'main' branch to deploy the new feature in production environment.

2. Database Access

- After deployment, check AWS Secrets Manager for:
  - MySQL username/password
  - Database name
  - RDS endpoint
- Create a t3.micro EC2 instance in the public subnet
- Install MySQL client:
  ```bash
  sudo apt update
  sudo apt install mysql-client
  ```
- Connect to RDS:
  ```bash
  mysql -h [RDS_ENDPOINT] -u [USERNAME] -p
  ```

3. API Testing

- Find API Gateway Invoke URL in the AWS Console under API Gateway → Stages.
- Use Postman or curl to test endpoints:
  `POST /add-card`
  `GET /transactions`
  `POST /topup` (via SQS → Lambda flow)
  `POST /deduct` (via SQS → Lambda flow)

4. VPC & Security Notes

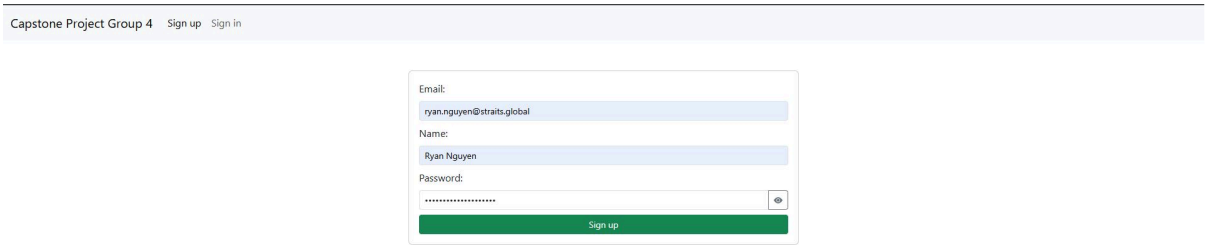
- Lambdas and RDS are in a private subnet.
- Use the Jump Host in the public subnet for SSH and database access.
- Ensure Security Groups allow traffic between the Jump Host and RDS.
- Cognito is used for authentication; configure user pools for frontend login

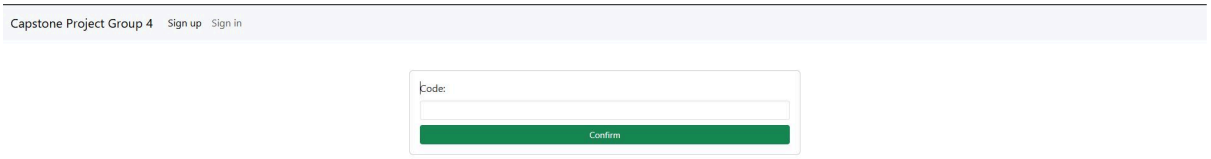
5. UI PORTAL

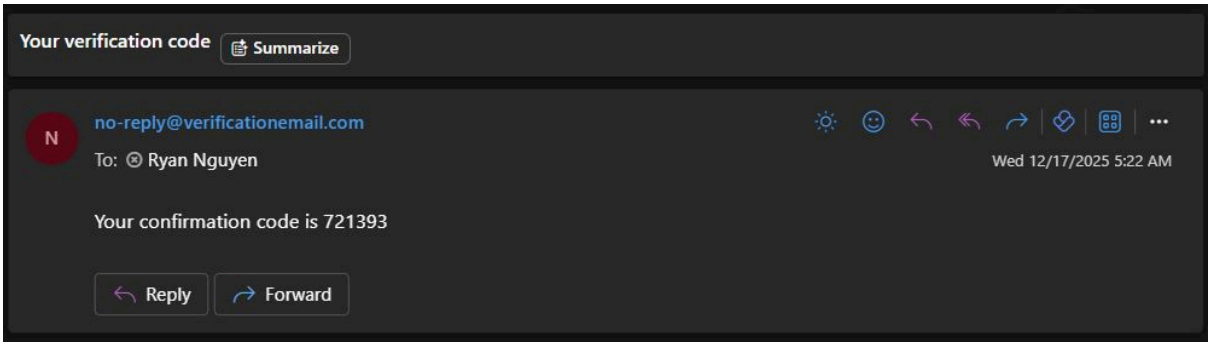
- **UI Portal is at** https://github.com/RyanNguyen1808/CapstoneProject_Group4_Frontend
- **Follow the Instructions in the Frontend Repo to deploy the UI PORTAL**

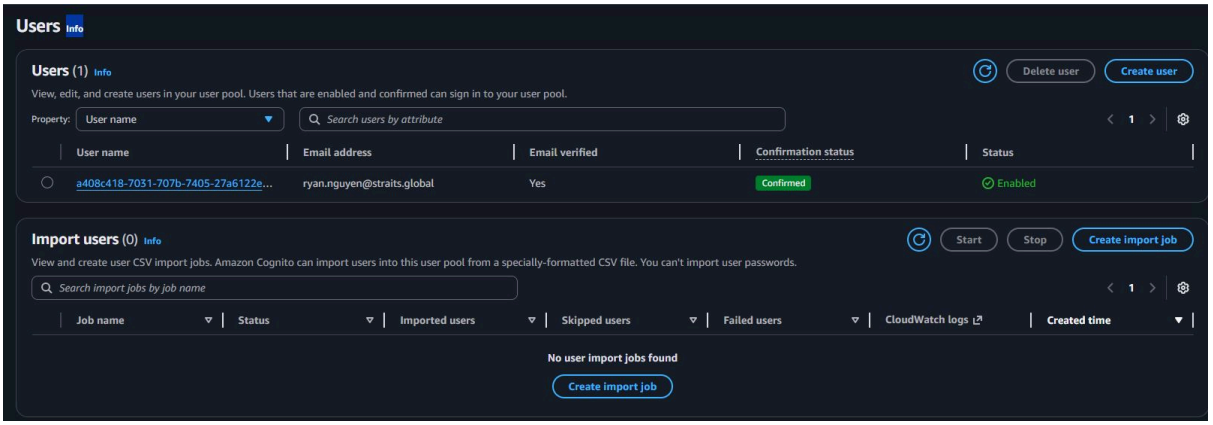
User Interface Walkthrough

Sign Up Process

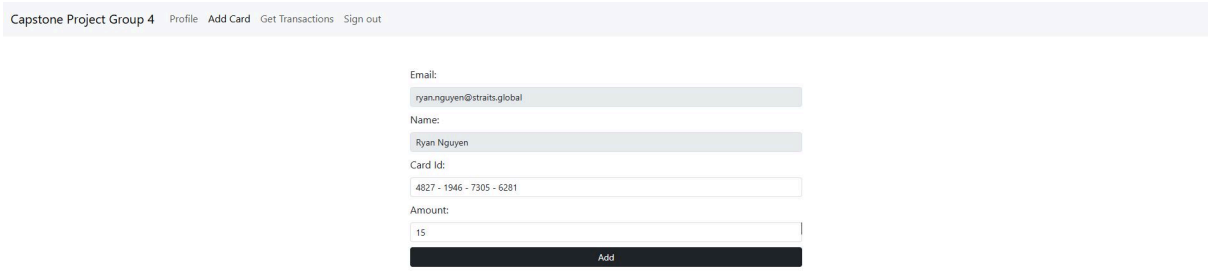


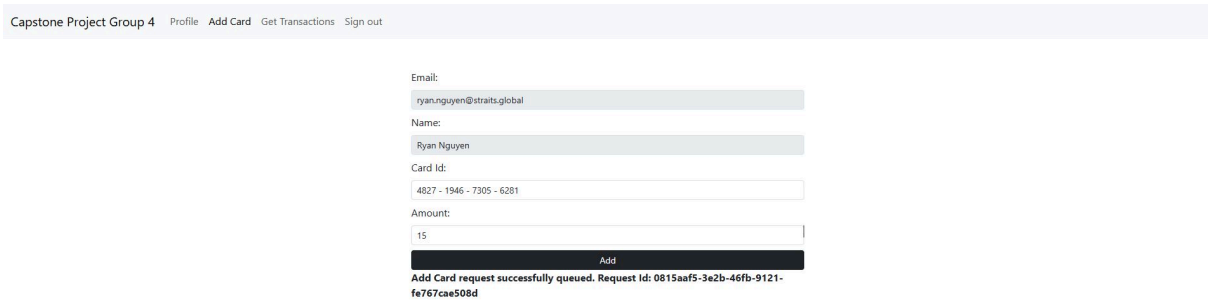




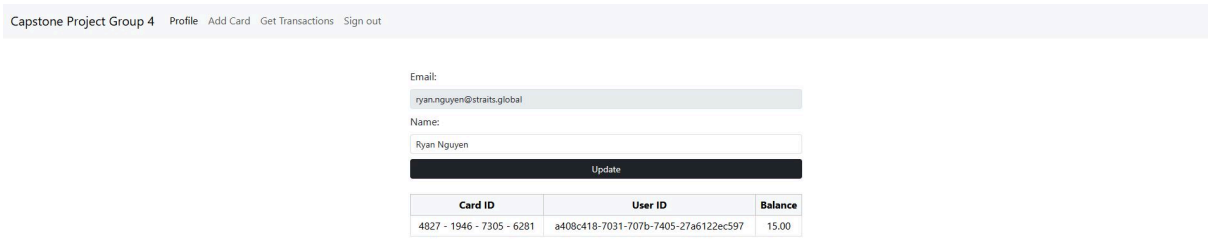


Add Card

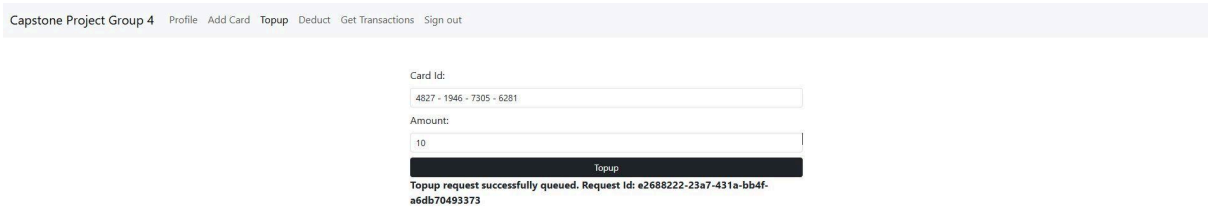


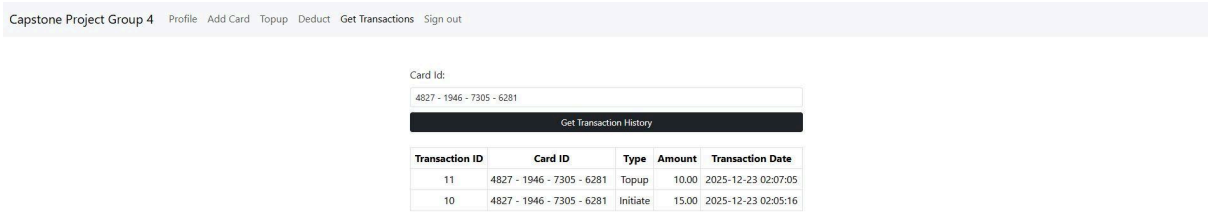



```
MySQL [ce11capstonegroup4AppDB]> select * from CARDS; select * from TRANSACTIONS;
+--------------------------+--------------------------------------+---------+
| CARD_ID                  | USER_ID                              | BALANCE |
+--------------------------+--------------------------------------+---------+
| 4827 - 1946 - 7305 - 6281 | a408c418-7031-707b-7405-27a6122ec597 |      15 |
+--------------------------+--------------------------------------+---------+
1 row in set (0.004 sec)

+----------------+--------------------------+------+--------+---------------------+
| TRANSACTION_ID | CARD_ID                  | TYPE | AMOUNT | TRANSACTION_DATETIME |
+----------------+--------------------------+------+--------+---------------------+
|              9 | 4827 - 1946 - 7305 - 6281 |    1 |     15 | 2025-12-17 05:29:25  |
+----------------+--------------------------+------+--------+---------------------+
1 row in set (0.080 sec)

MySQL [ce11capstonegroup4AppDB]>
```

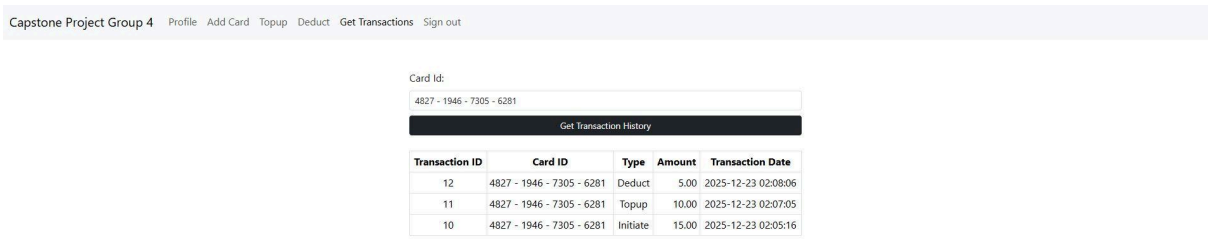


Topup

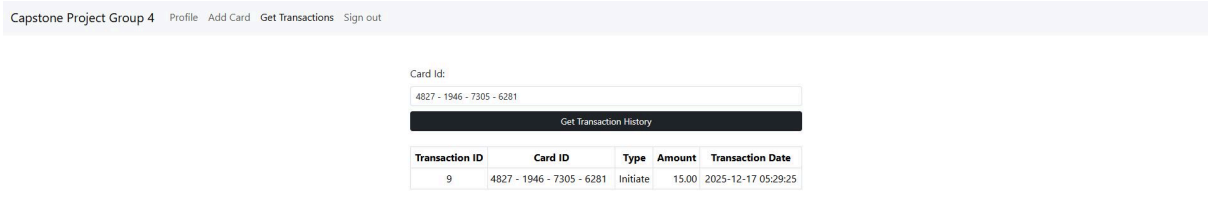




Deduct



View Transactions



Troubleshooting

Common Issues:

1. CD Pipeline Fails:

- Verify all secrets/variables are set
- Check IAM permissions for the service account

2. Cannot Connect to RDS:

- Verify security groups allow traffic from the jump host
- Check Secrets Manager for correct credentials

3. API Returns 403:

- Verify Cognito tokens are valid
- Check API Gateway stage deployment

4. Troubleshooting CORS Issues:

- "No 'Access-Control-Allow-Origin' header"
  - Check API Gateway CORS configuration in Terraform
  - Verify the origin matches your CloudFront/S3 URL
  - Add custom headers to `allow_headers` in API Gateway CORS
- Preflight (OPTIONS) request fails
  - Ensure the OPTIONS method is allowed in the API Gateway
  - Check API Gateway integration responses

Project Engineers:

Group: CapstoneProject - Cohort 11 Group4
Course: Cloud Infrastructure Engineering
Presentation Date: 27 December 2025

Team Members:

Nguyen Tien Thinh (Ryan)
Marlon Francisco Prudencio (Marlon)
Maroju Sridhar (Sridhar)
Voruganti Sreedevi (Sreedevi)