**FeedbackFlow**

**Personal Growth**

This project taught me how to approach unfamiliar problems with patience and persistence. I learned to read error messages critically, trace issues across services, and adapt solutions to my development environment. More importantly, I gained the confidence to build, debug, and deploy real-world cloud applications independently.

**Overview:**

Built a full-stack serverless web app that collects user feedback via a React frontend and stores it in DynamoDB using AWS Lambda and API Gateway. Demonstrated real-world deployment, CORS handling, and cloud integration.

**Tech Stack:**

* **Frontend**: React (JavaScript)
* **Backend**: AWS Lambda (Node.js)
* **API Gateway**: REST API
* **Database**: DynamoDB
* **Deployment**: AWS Console (manual setup)
* **Environment**: Windows 11

**Key Implementation Steps:**

1. **Frontend Setup**

* Initialized React app using npx create-react-app
* Built a feedback form with input fields and submit button
* Used fetch() to send POST requests to API Gateway

**2. Lambda Function**

* Created a Node.js Lambda function to receive feedback
* Parsed JSON input and stored it in DynamoDB
* Packaged dependencies and custom logic into a Lambda layer (on Windows)

3. **DynamoDB Integration**

* Created a table with primary key feedbackId
* Used AWS SDK (aws-sdk) to insert items from Lambda

4. **API Gateway**

* Created a REST API with POST method
* Integrated with Lambda using proxy integration
* Enabled CORS manually to allow frontend requests

5. **CORS Troubleshooting**

* Added Access-Control-Allow-Origin headers in Lambda response
* Verified CORS settings in API Gateway method and stage

6. **Deployment & Testing**

* Deployed Lambda and API Gateway via AWS Console
* Connected React frontend to live API endpoint
* Tested end-to-end flow: form → API → Lambda → DynamoDB

7. **Tagging & Cleanup**

* Tagged resources with Project: FeedbackFlow
* Used AWS Tag Editor to locate and clean up resources

**Skills Demonstrated:**

* Full-stack cloud integration
* Serverless architecture
* AWS Lambda packaging and layers (on Windows)
* CORS debugging and API Gateway setup
* DynamoDB data modeling
* React-to-API communication
* Resource tagging and cleanup

**FeedbackFlow: Serverless Architecture**

(React Frontend + AWS Backend)

A diagram of a software development process

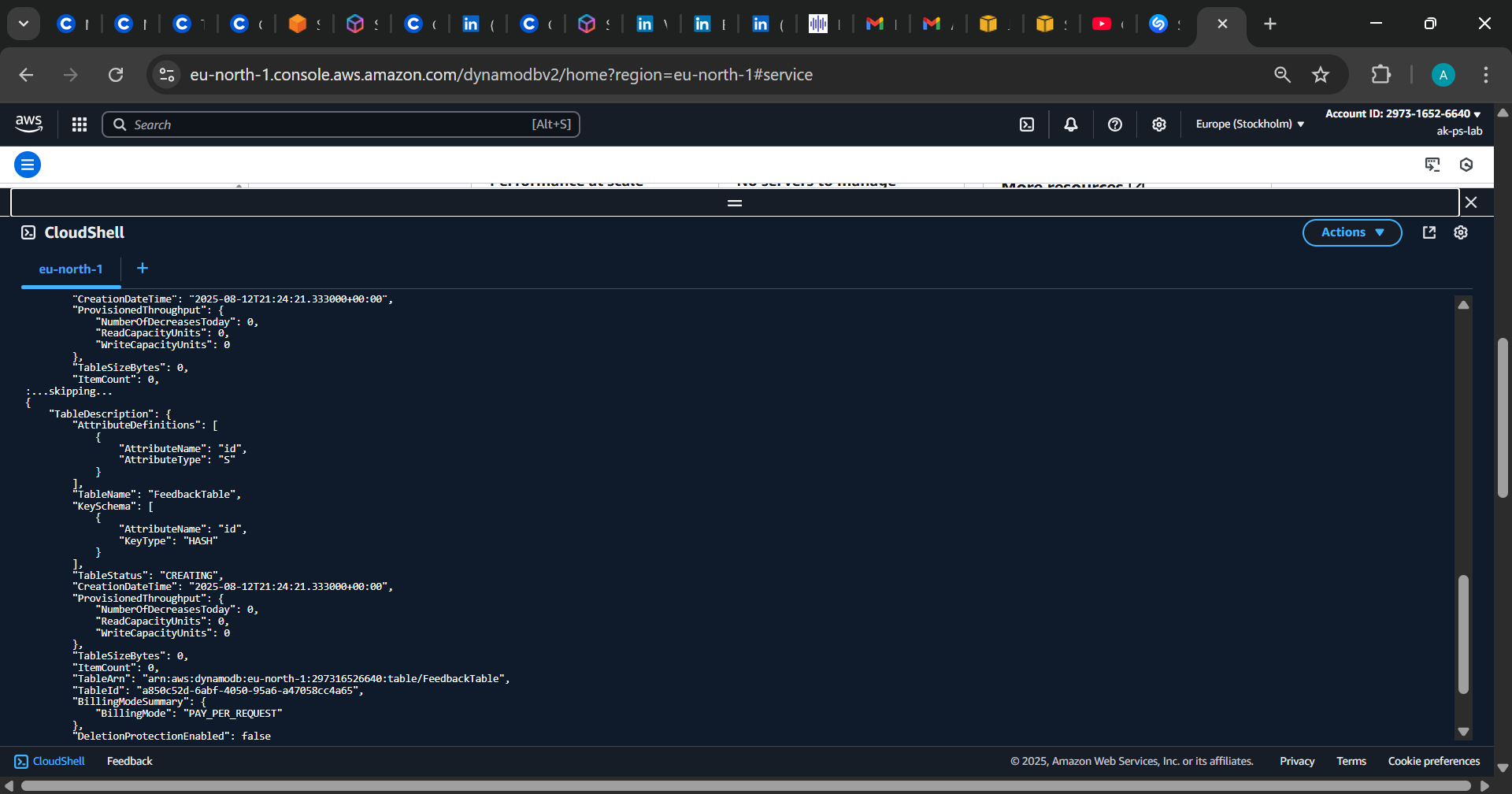
AI-generated content may be incorrect.

Serverless Feedback App Architecture Diagram

(React + AWS Lambda + DynamoDB)

**Screenshots**

1.A



1.B

A screenshot of a computer

AI-generated content may be incorrect.

2.

A screenshot of a computer

AI-generated content may be incorrect.

2B.

A screenshot of a computer

AI-generated content may be incorrect.

3.A

A screenshot of a computer

AI-generated content may be incorrect.

4.AA screenshot of a computer

AI-generated content may be incorrect.

5.A

A screenshot of a computer

AI-generated content may be incorrect.

6.A

A screenshot of a computer

AI-generated content may be incorrect.

7.A CURL check

A screenshot of a computer

AI-generated content may be incorrect.

8.A ERROR ENCOUNTERED

A screenshot of a computer

AI-generated content may be incorrect.

9.A ERROR DEBUG CORS FIX

A screenshot of a computer

AI-generated content may be incorrect.

10.A STAGES Properties Page

A screenshot of a computer

AI-generated content may be incorrect.

11.A SUCCESS Submission Screenshot

A screenshot of a computer

AI-generated content may be incorrect.

12.A DYNAMO DB ENTRY PRESENT

A screenshot of a computer

AI-generated content may be incorrect.

13.A NEW FRESH ENTRY

A screenshot of a computer

AI-generated content may be incorrect.

13.B NEW FRESH ENTRY UPDATED IN DYNAMO DB TABLE

A screenshot of a computer

AI-generated content may be incorrect.

Key Challenges & Resolutions

* CORS Policy Errors
* Encountered browser-level restrictions when connecting React to API Gateway. Resolved by configuring CORS headers in both Lambda responses and API Gateway settings, gaining a clear understanding of cross-origin request handling.
* Lambda Packaging on Windows
* Faced deployment issues due to incorrect file structures and missing dependencies. Learned to zip Lambda code and properly, and created Lambda Layers to manage shared libraries efficiently.
* API Gateway Integration Confusion
* Initial requests failed due to non-proxy integration. Switched to proxy integration and adapted Lambda code to parse , improving my grasp of how API Gateway transforms HTTP requests.
* IAM Role Permissions
* Lambda was unable to write to DynamoDB due to insufficient permissions. Resolved by attaching a custom IAM role with precise access policies, reinforcing my understanding of AWS security and role-based access control.
* Resource Cleanup & Tagging
* Discovered that deleting an IAM user doesn’t automatically remove associated resources. Used AWS Tag Editor to locate and clean up all tagged resources, learning best practices for cost management and environment hygiene.

Technical Skills Strengthened

* Full-stack integration using React and AWS services
* Serverless architecture design and deployment
* Lambda Layers and dependency management on Windows
* RESTful API creation and CORS troubleshooting
* DynamoDB schema design and data operations
* IAM role configuration and permission debugging
* Resource tagging and cleanup using AWS Tag Editor