**Internal Student Health & Productivity Tracker**

# **Overview**

This project creates a private web application for NeoCloud students to track their productivity, provide feedback, and report blockers. The system uses serverless AWS components managed through Terraform.

# **Architecture Diagram**

```

```

## **Terraform Implementation**

### **cognito.tf**

```

resource "aws\_cognito\_user\_pool" "student\_pool" {

name = "neo-cloud-student-pool"

username\_attributes = ["email"]

auto\_verified\_attributes = ["email"]

password\_policy {

minimum\_length = 8

require\_lowercase = true

require\_numbers = true

require\_symbols = true

require\_uppercase = true

}

schema {

name = "email"

attribute\_data\_type = "String"

required = true

mutable = true

}

}

resource "aws\_cognito\_user\_pool\_client" "client" {

name = "student-tracker-client"

user\_pool\_id = aws\_cognito\_user\_pool.student\_pool.id

explicit\_auth\_flows = ["ALLOW\_USER\_PASSWORD\_AUTH", "ALLOW\_REFRESH\_TOKEN\_AUTH"]

prevent\_user\_existence\_errors = "ENABLED"

}

```

### **dynamodb.tf**

```

resource "aws\_dynamodb\_table" "productivity\_logs" {

name = "StudentProductivityLogs"

billing\_mode = "PAY\_PER\_REQUEST"

hash\_key = "userId"

range\_key = "logDate"

attribute {

name = "userId"

type = "S"

}

attribute {

name = "logDate"

type = "S"

}

attribute {

name = "entryType"

type = "S"

}

global\_secondary\_index {

name = "EntryTypeIndex"

hash\_key = "userId"

range\_key = "entryType"

projection\_type = "ALL"

}

}

```

### **gateway.tf**

```

resource "aws\_api\_gateway\_rest\_api" "tracker\_api" {

name = "StudentTrackerAPI"

}

resource "aws\_api\_gateway\_resource" "logs" {

rest\_api\_id = aws\_api\_gateway\_rest\_api.tracker\_api.id

parent\_id = aws\_api\_gateway\_rest\_api.tracker\_api.root\_resource\_id

path\_part = "logs"

}

resource "aws\_api\_gateway\_method" "logs\_method" {

rest\_api\_id = aws\_api\_gateway\_rest\_api.tracker\_api.id

resource\_id = aws\_api\_gateway\_resource.logs.id

http\_method = "POST"

authorization = "COGNITO\_USER\_POOLS"

authorizer\_id = aws\_api\_gateway\_authorizer.cognito.id

}

resource "aws\_api\_gateway\_integration" "logs\_integration" {

rest\_api\_id = aws\_api\_gateway\_rest\_api.tracker\_api.id

resource\_id = aws\_api\_gateway\_resource.logs.id

http\_method = aws\_api\_gateway\_method.logs\_method.http\_method

integration\_http\_method = "POST"

type = "AWS\_PROXY"

uri = aws\_lambda\_function.tracker\_lambda.invoke\_arn

}

resource "aws\_lambda\_permission" "api\_gw" {

statement\_id = "AllowExecutionFromAPIGateway"

action = "lambda:InvokeFunction"

function\_name = aws\_lambda\_function.tracker\_lambda.function\_name

principal = "apigateway.amazonaws.com"

source\_arn = "${aws\_api\_gateway\_rest\_api.tracker\_api.execution\_arn}/\*/\*"

}

```

### **Lambda Function (lambda.py)**

```

import json

import boto3

from datetime import datetime

dynamodb = boto3.resource('dynamodb')

table = dynamodb.Table('StudentProductivityLogs')

def lambda\_handler(event, context):

# Parse request

body = json.loads(event['body'])

user\_id = event['requestContext']['authorizer']['claims']['sub']

# Prepare log item

log\_item = {

'userId': user\_id,

'logDate': datetime.now().isoformat(),

'entryType': body.get('type', 'log'), # log/feedback/blocker

'content': body['content'],

'details': body.get('details', {})

}

# Store in DynamoDB

table.put\_item(Item=log\_item)

return {

'statusCode': 200,

'body': json.dumps({'message': 'Log saved successfully'})

}

```

# 

# **Frontend Structure (React App)**

```

frontend/

├── public/

│ ├── index.html

│ └── assets/

├── src/

│ ├── components/

│ │ ├── Login.js

│ │ ├── Dashboard.js

│ │ ├── LogEntry.js

│ │ └── Insights.js

│ ├── services/

│ │ ├── auth.js

│ │ └── api.js

│ ├── App.js

│ └── index.js

├── package.json

└── README.md

```

# **Deployment Process**

### **Initialize Terraform:**

```

cd terraform

terraform init

```

### **Apply infrastructure:**

```

terraform apply

```

### **Deploy frontend to S3:**

```

aws s3 sync ../frontend/build s3://student-tracker-ui

```

# **Monitoring Setup**

CloudWatch will automatically track:

- API Gateway request counts and latencies

- Lambda invocation metrics and errors

- DynamoDB read/write capacity usage

## **Security Considerations**

1. All API routes require Cognito authentication

2. DynamoDB items are partitioned by userId

3. Least-privilege IAM roles for Lambda

4. HTTPS enforced via API Gateway

5. Sensitive data encrypted at rest

## **Cost Optimization**

1. Pay-per-request DynamoDB billing

2. Serverless components scale to zero

3. CloudFront caching for static assets

4. Lambda configured with appropriate memory/timeout

This implementation provides a complete, secure, and scalable solution for the student productivity tracking system while keeping costs minimal for low-traffic internal use.