**AWS Assignment Documentation**

**Project Overview**

This project implements a serverless solution using AWS services to store and retrieve user data. The infrastructure is provisioned using AWS CDK.

**Architecture & AWS Services Used**

1. **Amazon DynamoDB**: Stores user data with schema designed for efficient querying of the latest 10 records.
2. **Amazon API Gateway**: Exposes an endpoint for adding users.
3. **AWS Lambda**:
   * Handles user addition via API.
   * Runs a scheduled function every 1 hours to fetch the latest 10 users and send an email.
4. **Amazon Simple Email Service (SES)**: Sends an email with the latest user records.
5. **AWS CDK**: Manages and deploys the infrastructure as code.

**Project Structure**A screenshot of a computer

AI-generated content may be incorrect.

**DynamoDB Table Schema**

| **Attribute** | **Type** | **Description** |
| --- | --- | --- |
| userId | S | Unique identifier (Partition Key) |
| userName | S | Name of the user |
| createdAt | N | Timestamp for sorting |

The **createdAt** field allows querying the latest 10 users efficiently.

**Deployment Commands**

**1. Install AWS CDK & Dependencies**

npm install -g aws-cdk

python -m venv .venv # Create a virtual environment

source .venv/bin/activate # Activate the virtual environment

pip install -r requirements.txt # Install dependencies

**2. Bootstrap the CDK App**

cdk bootstrap

**3. Deploy the Infrastructure**

cdk synth # Synthesize the CloudFormation template

cdk deploy # Deploy the stack

**4. Destroy the Stack (Optional)**

cdk destroy

**API Endpoints**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | / users | Adds a new user to DynamoDB |

**Example Request (POST /** users curl**)**

curl --location 'https://of2lspjx2c.execute-api.ap-south-1.amazonaws.com/prod/users' \

--header 'Content-Type: application/json' \

--data-raw '[

{

"userName": "sandesha",

"email": "sandesha@example.com"

},

{

"userName": "Kiran",

"email": "Kiran@example.com"

},

{

"userName": "Kavya",

"email": "Kavya@example.com"

},

{

"userName": "Raju",

"email": "Raju@example.com"

},

{

"userName": "Brij",

"email": "brij@example.com"

},

{

"userName": "Vivek",

"email": "vivek@example.com"

},

{

"userName": "Neelansh",

"email": "neelansh@example.com"

},

{

"userName": "Vidya",

"email": "vidya@example.com"

},

{

"userName": "Vicky",

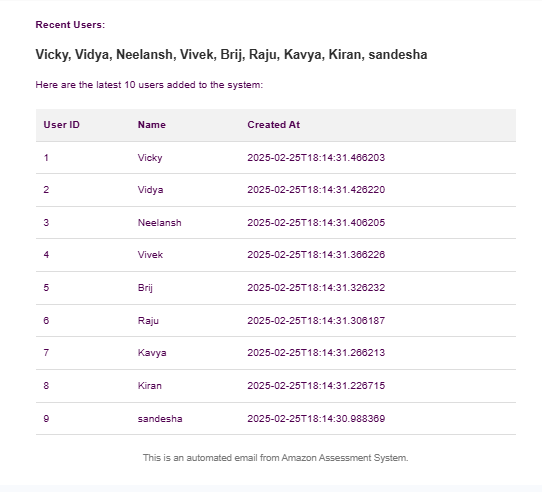
"email": "vicky@example.com"

}

]'

**Scheduled Lambda Function**

* Runs every **1 hours** (configured via AWS EventBridge).
* Fetches the latest 10 records from DynamoDB.
* Sends an email via AWS SES.

**Email Template Sent via SES** **Email Received From SES :**A screenshot of a phone

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

**Conclusion**

This project demonstrates the use of AWS serverless technologies to implement a scalable user management system with automated notifications. All infrastructure is managed using AWS CDK for easy deployment and maintenance.

**Next Steps**

* Implement authentication with AWS Cognito.
* Enhance monitoring with AWS CloudWatch Logs & Metrics.
* Add error handling and retries in Lambda functions.

Proof of DB,Lambda in AWS  
  
Scheduler Running in AWS  
A screenshot of a computer

AI-generated content may be incorrect.

Data in DB (added through API)  
A screenshot of a computer

AI-generated content may be incorrect.

Dynamo DB instance  
  
A screenshot of a computer

AI-generated content may be incorrect.  
A screenshot of a computer

AI-generated content may be incorrect.