TITLE: Voting Website Using AWS services

By KOLAMUDI PRANEETHA- 2200031748

CLOUD & SERVERLESS COMPUTING – 22SCEC3305A

Section 31

Under the guidance of

Dr. K.V.RAVI TEJA

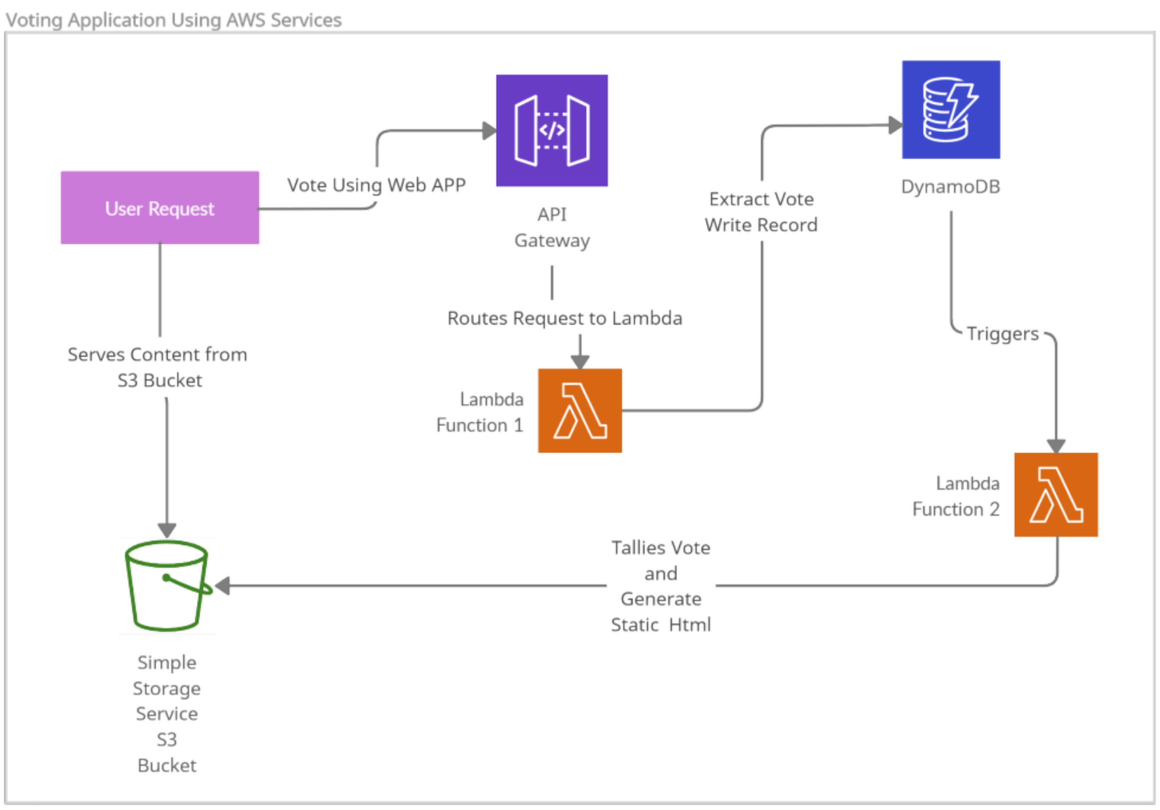
ABSTRACT

The Serverless Voting Application is designed to provide a highly scalable, secure, and cost-effective online voting system using AWS serverless technologies. The architecture eliminates traditional server dependencies by utilizing AWS Lambda for backend logic, Amazon API Gateway for handling HTTP requests, and Amazon DynamoDB for efficient and reliable vote storage. Amazon Cognito is integrated for secure user authentication and access management, while AWS Step Functions orchestrate workflows for validating and processing votes.

This serverless approach ensures automatic scaling, high availability, and fault tolerance, allowing the system to efficiently handle varying voting loads. Additionally, the pay-as-you-go pricing model reduces operational costs by charging only for actual usage. The event-driven execution model enables real-time vote processing, making the application highly responsive and adaptable for large-scale elections, surveys, and polling events.

By leveraging AWS-managed services, the Serverless Voting Application minimizes infrastructure overhead while ensuring a secure, reliable, and efficient voting process, making it an ideal solution for modern online voting requirements.

ARCHITECTURE


Voting Application Using AWS Services
User Request
Vote Using Web APP
API Gateway
Routes Request to Lambda
Lambda Function 1
Extract Vote Write Record
DynamoDB
Triggers
Lambda Function 2
Tallies Vote and Generate Static Html
Serves Content from S3 Bucket
Simple Storage Service S3 Bucket

MODULES

User Authentication & Authorization:

Implements Amazon Cognito for user registration, login, and role-based access control (voter, admin).

Voting Management

Allows users to cast votes via API Gateway.

AWS Lambda processes vote submissions.

Stores votes securely in Amazon DynamoDB.

Real-time Results Processing

Fetches live voting results from DynamoDB.

Uses AWS Lambda and API Gateway for result retrieval.

Dashboard

Enables admins to manage elections, add candidates, and view statistics.

We can also see the count/number of votes are there per option.

SERVICES

Lambda: AWS Lambda is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you. These events may include changes in state or an update, such as a user placing an item in a shopping cart on an ecommerce website.

DynamoDB: Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. You can use Amazon DynamoDB to create a database table that can store and retrieve any amount of data, and serve any level of request traffic.

Amazon Cognito: Amazon Cognito is a service provided by AWS that is used **for user authentication, authorization, and user management** in web and mobile applications.

API Gateway: API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other.

AWS IAM (Identity & Access Management for Security): Controls permissions for Lambda, API Gateway, and DynamoDB. Ensures role based access for different user types.

<u>Lambda Function Codes</u>

Code for storing vote: Python code

```
import boto3 import json from

decimal import Decimal

dynamodb = boto3.resource('dynamodb') table

= dynamodb.Table('Votes')

# Helper function to convert Decimal values def

decimal_to_int(obj):      if isinstance(obj,

Decimal):

        return int(obj)  # Convert Decimal to int
    raise TypeError  # Other types remain unchanged

def lambda_handler(event, context):      http_method =

event.get('httpMethod')

    if not http_method:

        return {

            'statusCode': 400,

            'headers': {'Access-Control-Allow-Origin': '*'},
            'body': json.dumps({'message': 'Error: Missing httpMethod in
request'})
        }
    # Handle CORS preflight request     if
http_method == 'OPTIONS':
        return {
            'statusCode': 200,
            'headers': {
                'Access-Control-Allow-Origin': '*',
              'Access-Control-Allow-Methods': 'POST, OPTIONS',
     'Access-Control-Allow-Headers': 'Content-Type'
 },
              'body': json.dumps({'message': 'CORS preflight response'})
          }
      if http_method == 'POST':
          try:
              body = json.loads(event['body']) if isinstance(event['body'],
  str) else event['body']
   option = body.get('option')
               if not option:
  return {
```

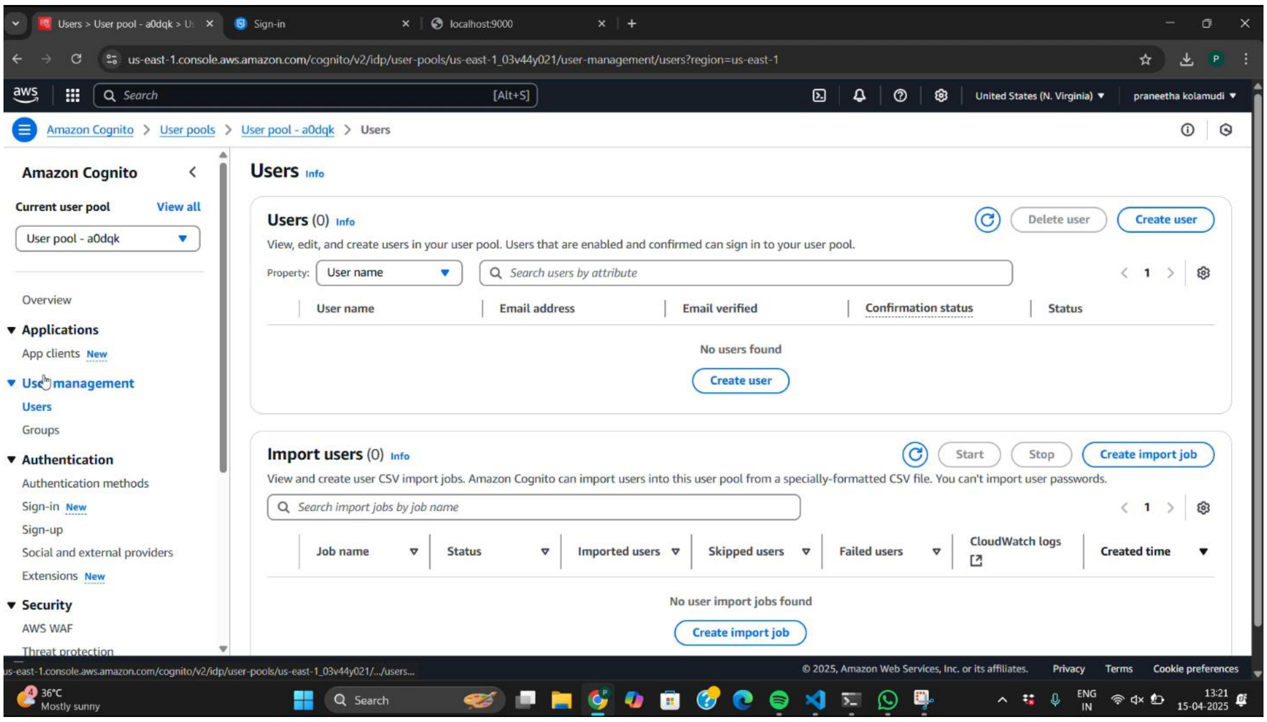
```
                    'statusCode': 400,

                    'headers': {'Access-Control-Allow-Origin': '*'},

                    'body': json.dumps({'message': 'Missing vote option'})

                }


            key_name = table.key_schema[0]['AttributeName']


            response = table.update_item(

              Key={key_name: option},

              UpdateExpression="ADD #cnt :inc",

      ExpressionAttributeNames={"#cnt": "vote_count"},

             ExpressionAttributeValues={':inc': Decimal(1)},  # Ensure it's
Decimal-compatible

             ReturnValues="UPDATED_NEW"

           )

           # Convert response values to avoid Decimal error

updated_response = json.dumps(response, default=decimal_to_int)

           return

{

             'statusCode': 200,
                    'headers': {
                        'Access-Control-Allow-Origin': '*',
                        'Access-Control-Allow-Methods': 'OPTIONS, POST,  GET',
                        'Access-Control-Allow-Headers': 'Content-Type'
                    },
                    'body': updated_response  # Ensuring JSON-safe format
                }
        except Exception as e:
return {
'statusCode': 500,
                    'headers': {'Access-Control-Allow-Origin': '*'},
                    'body': json.dumps({'message': 'Internal Server Error', 'error':
        str(e)})
                }
        {
          "httpMethod": "POST",
          "body": "{\"option\": \"Candidate_A\"}"
        }
```

Code implemented for get user:

```
import boto3 import json from

decimal import Decimal


dynamodb = boto3.resource('dynamodb') table

= dynamodb.Table('Votes')


def convert_item(item):

   """ Recursively convert Decimal values to int/float for JSON

serialization. """     if isinstance(item, Decimal):          return

int(item) if item % 1 == 0 else float(item)

elif isinstance(item, list):

return [convert_item(i) for i in item]

 elif isinstance(item, dict):          return {k:

 convert_item(v) for k, v in item.items()}     return item


 def lambda_handler(event, context):

     try:

        print("Received Event:", json.dumps(event, indent=2))
        # Handle CORS Preflight Requests
        if event.get("httpMethod") == "OPTIONS":
            return {
                'statusCode': 200,
                'headers': {
                    'Content-Type': 'application/json',
                    'Access-Control-Allow-Origin': '*',  # Allow all origins
                    'Access-Control-Allow-Methods': 'GET, OPTIONS',
                    'Access-Control-Allow-Headers': 'Content-Type'
                },
                'body': json.dumps({'message': 'CORS Preflight Success'})
            }
       # Scan the table          response =
table.scan()                           items   =
response.get('Items', [])

       # Convert data to JSON serializable format          vote_results =
convert_item(items)
        return {
          'statusCode': 200,
            'headers': {

                'Content-Type': 'application/json',

                'Access-Control-Allow-Origin': '*',  # Ensure CORS works
'Access-Control-Allow-Methods': 'GET, OPTIONS',

                'Access-Control-Allow-Headers': 'Content-Type'

            },

            'body': json.dumps(vote_results)

        }

    except Exception as e:         print("Error:", str(e))
 return
 {

           'statusCode': 500, #Return a 500 error code.

            'headers': {

                'Content-Type': 'application/json',

                'Access-Control-Allow-Origin': '*',

                'Access-Control-Allow-Methods': 'GET, OPTIONS',

                'Access-Control-Allow-Headers': 'Content-Type'

            },

            'body': json.dumps({'error': str(e)}) #return the error message in
the body.

        }
```

MODULES IMPLEMENTATION

FIG 1-5 : To create the login and signup system using Amazon Cognito, we first logged into the AWS Management Console and navigated to the Cognito service. We created a new User Pool to manage user accounts, enabling email as the primary sign-in method and configuring password policies for security. In the user pool settings, we enabled the self-registration option, allowing users to sign up with a verification email. Then, we created an App Client without a client secret to connect our frontend to the user pool. We enabled Hosted UI in Cognito to auto-generate a login/signup page, which includes built-in forms and flows for user authentication. The generated Hosted UI URL was embedded into our HTML login page as a button or redirect link. After a successful login, Cognito returns a token that can be used to manage user sessions securely. This simplified the authentication process by eliminating the need to code the login logic manually and provided a fully managed, secure user authentication system.


us-east-1.console.aws.amazon.com/cognito/v2/idp/user-pools/us-east-1_03v44y021/user-management/users?region=us-east-1

Amazon Cognito > User pools > User pool - a0dqk > Users

Amazon Cognito
Current user pool: User pool - a0dqk

Users

Users (0)
View, edit, and create users in your user pool. Users that are enabled and confirmed can sign in to your user pool.
Property: User name

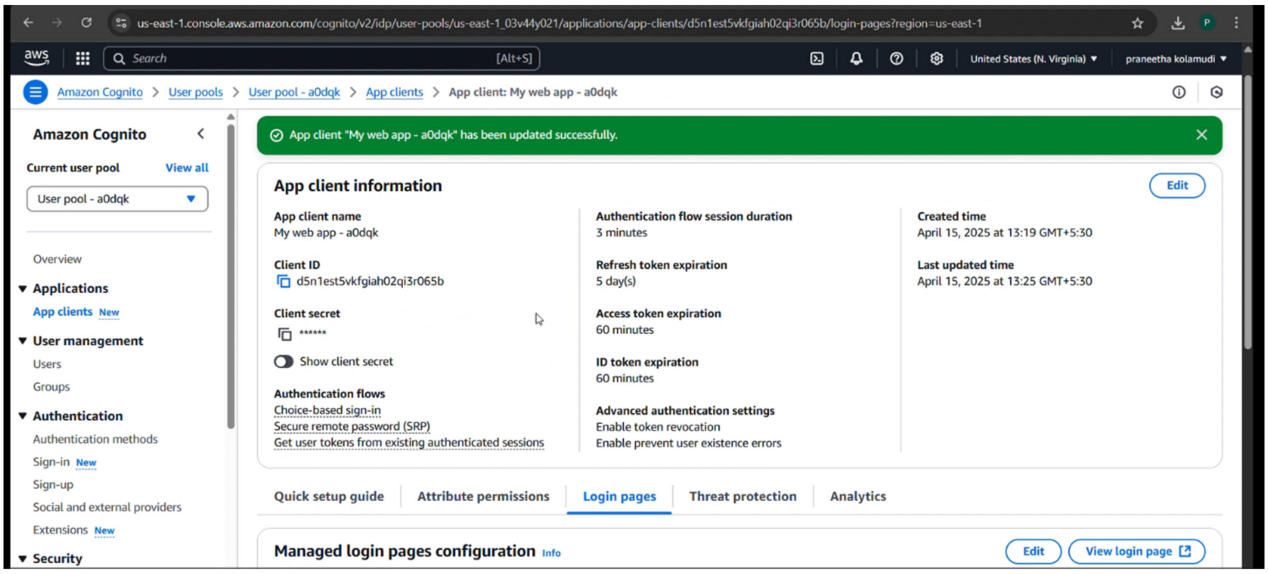
User name	Email address	Email verified	Confirmation status	Status

No users found

Import users (0)
View and create user CSV import jobs. Amazon Cognito can import users into this user pool from a specially-formatted CSV file. You can't import user passwords.

Job name	Status	Imported users	Skipped users	Failed users	CloudWatch logs	Created time

No user import jobs found



us-east-1.console.aws.amazon.com/cognito/v2/idp/user-pools/us-east-1_03v44y021/applications/app-clients/d5n1est5vkfgiah02qi3r065b/login-pages?region=us-east-1

Search [Alt+S] United States (N. Virginia) praneetha kolamudi

Amazon Cognito > User pools > User pool - a0dqk > App clients > App client: My web app - a0dqk

Amazon Cognito

Current user pool View all

User pool - a0dqk

Overview

Applications
- App clients New

User management
- Users
- Groups

Authentication
- Authentication methods
- Sign-in New
- Sign-up
- Social and external providers
- Extensions New

Security

App client "My web app - a0dqk" has been updated successfully.

App client information Edit

App client name
My web app - a0dqk

Client ID
d5n1est5vkfgiah02qi3r065b

Client secret
******

Show client secret

Authentication flows
Choice-based sign-in
Secure remote password (SRP)
Get user tokens from existing authenticated sessions

Authentication flow session duration
3 minutes

Refresh token expiration
5 day(s)

Access token expiration
60 minutes

ID token expiration
60 minutes

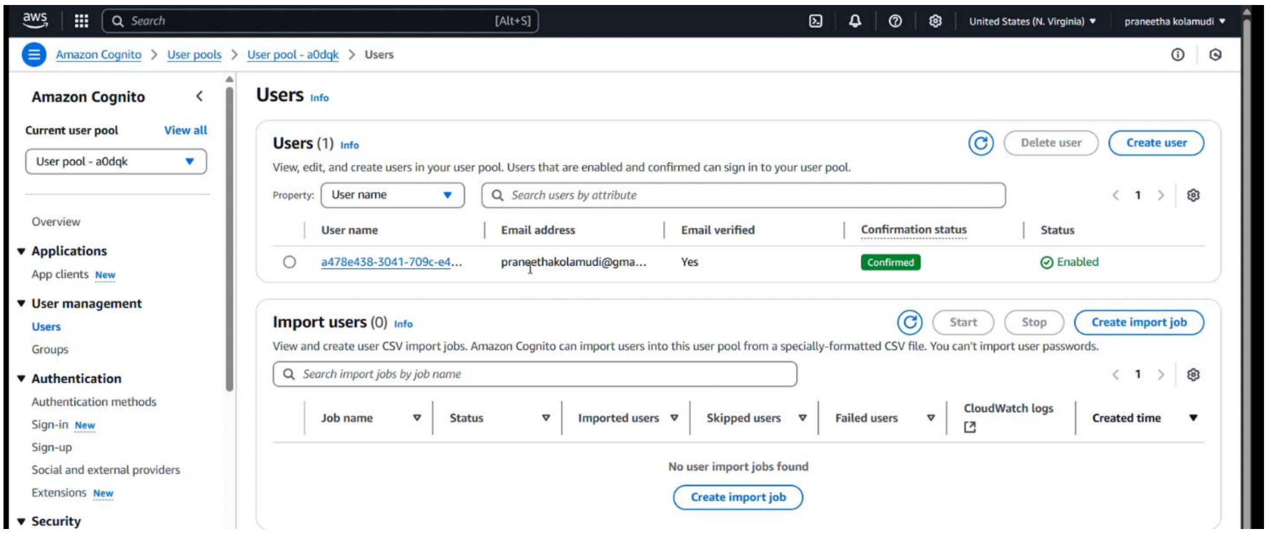
Advanced authentication settings
Enable token revocation
Enable prevent user existence errors

Created time
April 15, 2025 at 13:19 GMT+5:30

Last updated time
April 15, 2025 at 13:25 GMT+5:30

Quick setup guide | Attribute permissions | Login pages | Threat protection | Analytics

Managed login pages configuration Info Edit View login page



Search [Alt+S] United States (N. Virginia) praneetha kolamudi

Amazon Cognito > User pools > User pool - a0dqk > Users

Amazon Cognito

Current user pool View all

User pool - a0dqk

Overview

Applications
- App clients New

User management
- Users
- Groups

Authentication
- Authentication methods
- Sign-in New
- Sign-up
- Social and external providers
- Extensions New

Security

Users Info

Users (1) Info Delete user Create user

View, edit, and create users in your user pool. Users that are enabled and confirmed can sign in to your user pool.

Property: User name Search users by attribute

	User name	Email address	Email verified	Confirmation status	Status
○	a478e438-3041-709c-e4...	pranqethakolamudi@gma...	Yes	Confirmed	Enabled

Import users (0) Info Start Stop Create import job

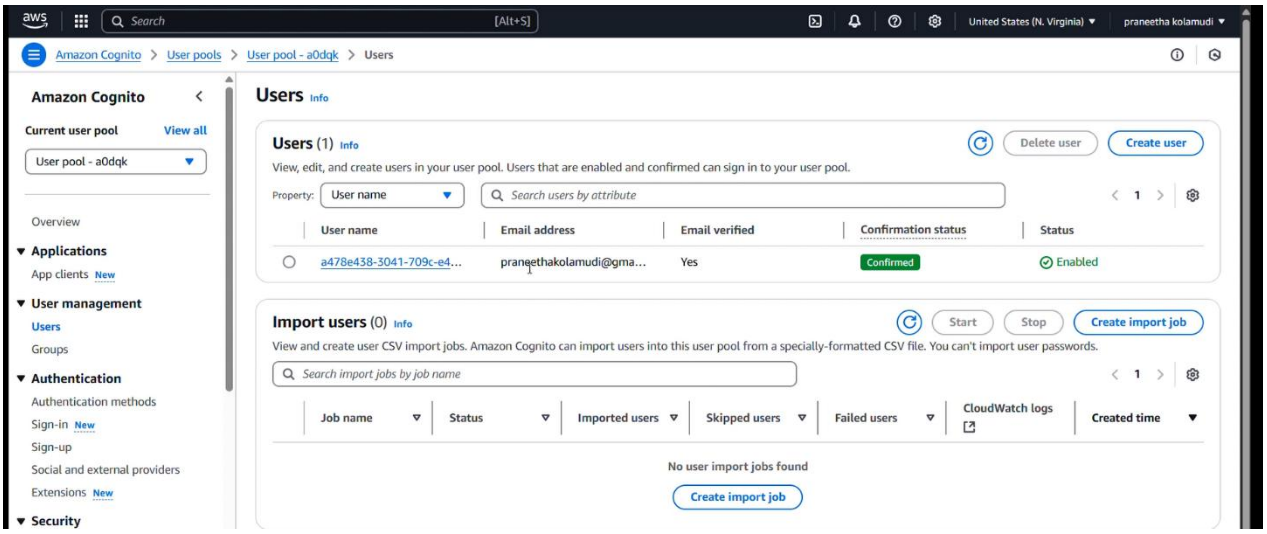
View and create user CSV import jobs. Amazon Cognito can import users into this user pool from a specially-formatted CSV file. You can't import user passwords.

Search import jobs by job name

Job name	Status	Imported users	Skipped users	Failed users	CloudWatch logs	Created time

No user import jobs found

Create import job



Users Info

Users (1) Info

Delete user | Create user

View, edit, and create users in your user pool. Users that are enabled and confirmed can sign in to your user pool.

Property: User name | Search users by attribute

User name	Email address	Email verified	Confirmation status	Status
a478e438-3041-709c-e4…	praneethakolamudi@gma…	Yes	Confirmed	Enabled

Import users (0) Info

Start | Stop | Create import job

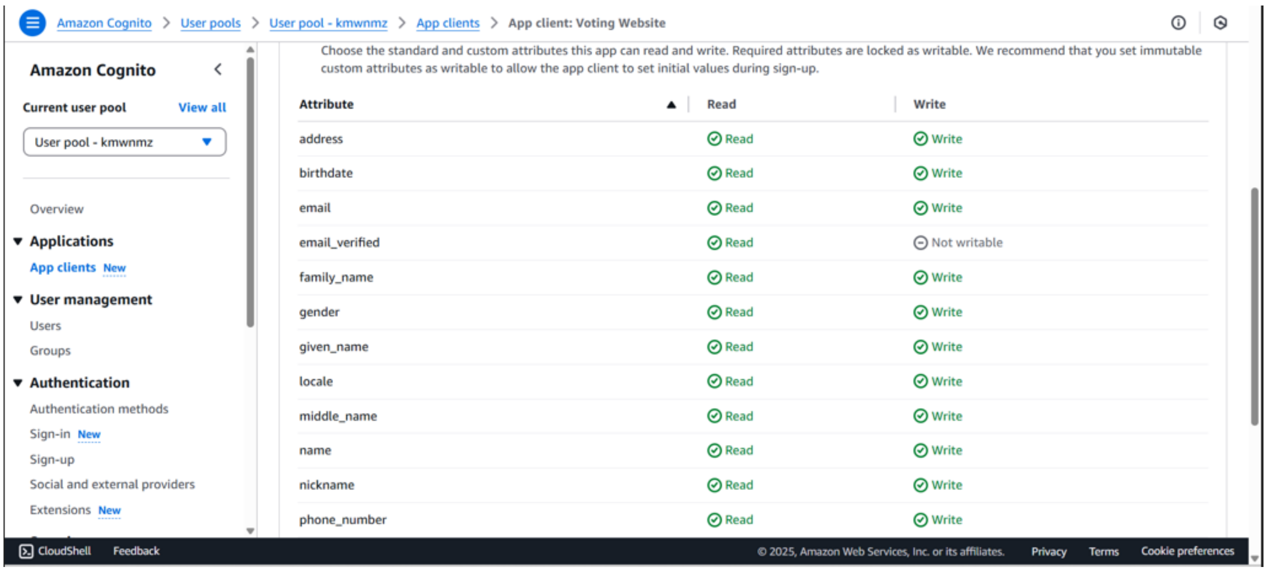
View and create user CSV import jobs. Amazon Cognito can import users into this user pool from a specially-formatted CSV file. You can't import user passwords.

Search import jobs by job name

Job name	Status	Imported users	Skipped users	Failed users	CloudWatch logs	Created time

No user import jobs found

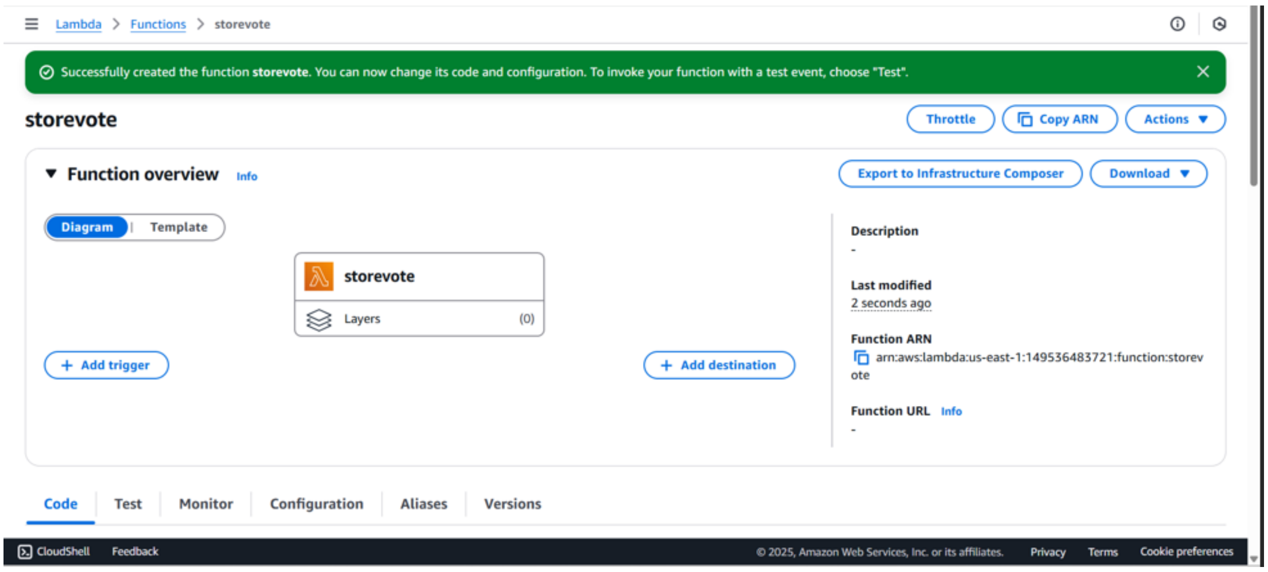
Create import job



Choose the standard and custom attributes this app can read and write. Required attributes are locked as writable. We recommend that you set immutable custom attributes as writable to allow the app client to set initial values during sign-up.

Attribute	Read	Write
address	Read	Write
birthdate	Read	Write
email	Read	Write
email_verified	Read	Not writable
family_name	Read	Write
gender	Read	Write
given_name	Read	Write
locale	Read	Write
middle_name	Read	Write
name	Read	Write
nickname	Read	Write
phone_number	Read	Write

FIG 6-12 : The first function, storeVote, accepts user votes through an API Gateway endpoint, processes the input, and stores the data in a DynamoDB table named Votes. Input validation and error handling were included to ensure data integrity. The second function, getResult, scans the DynamoDB table and aggregates vote counts in real-time. Both functions were written in Python (can be Java/Node.js) and tested using test events and API Gateway integrations. IAM roles were configured to allow secure access between Lambda and DynamoDB. We deployed both Lambda functions behind RESTful API endpoints using Amazon API Gateway, which were then called from the frontend using JavaScript. This setup enabled a fully serverless, scalable, and real-time voting system.



Lambda > Functions > storevote

Successfully created the function **storevote**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

storevote

Function overview

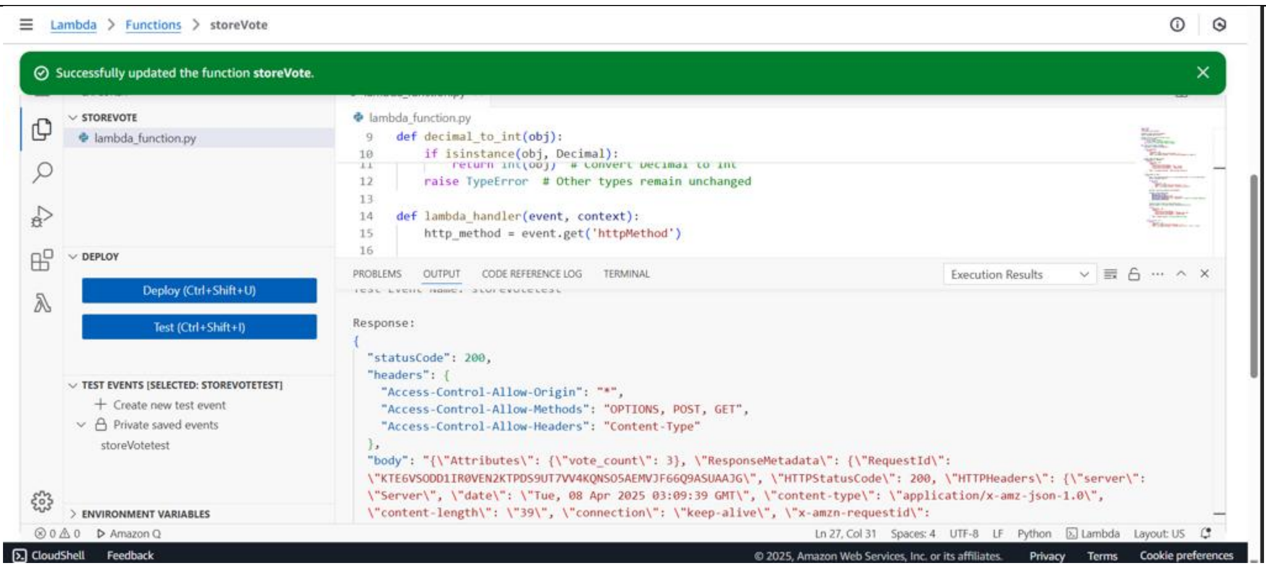
Description
-

Last modified
2 seconds ago

Function ARN
arn:aws:lambda:us-east-1:149536483721:function:storevote

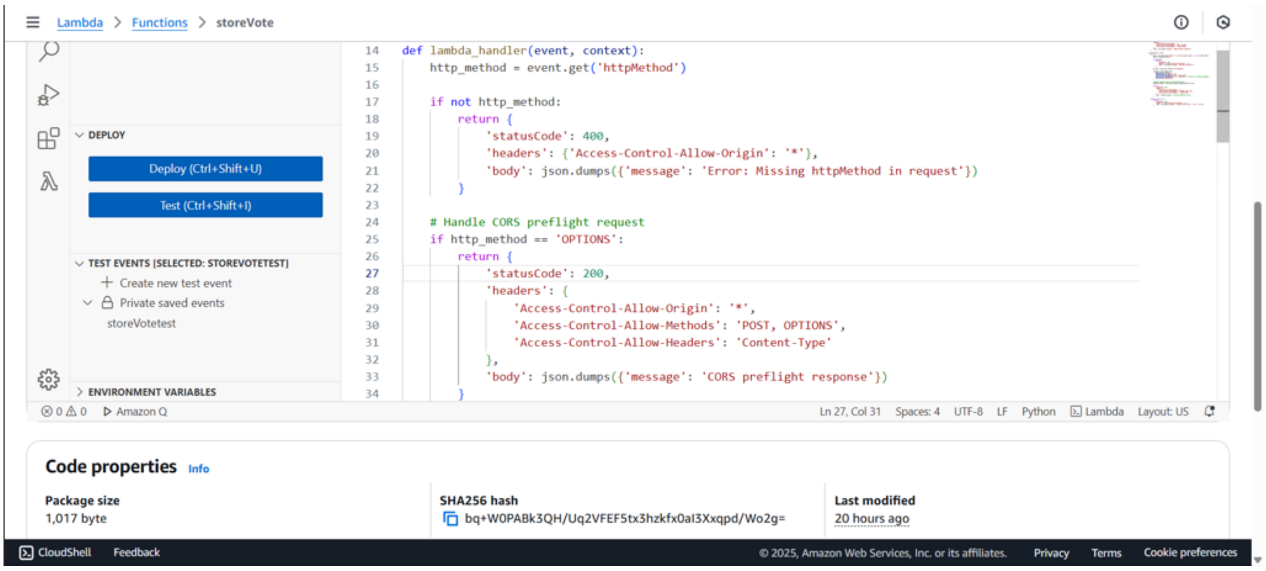
Function URL
-

© 2025, Amazon Web Services, Inc. or its affiliates.



```python
def decimal_to_int(obj):
    if isinstance(obj, Decimal):
        return int(obj)  # Convert Decimal to int
    raise TypeError  # Other types remain unchanged

def lambda_handler(event, context):
    http_method = event.get('httpMethod')

```

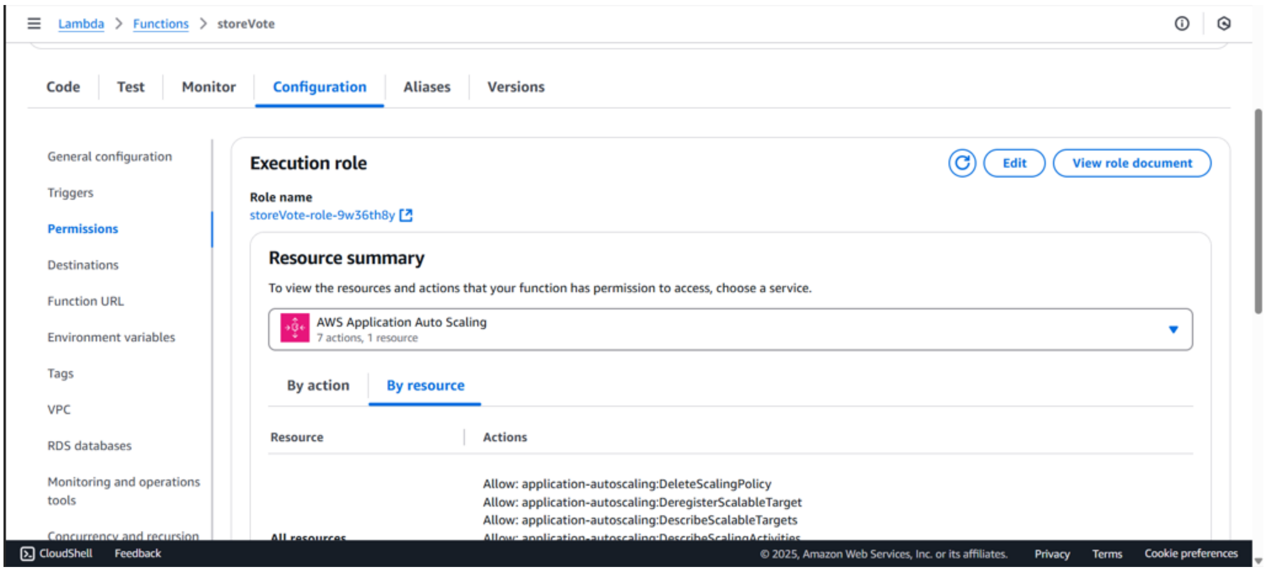
```
Response:
{
  "statusCode": 200,
  "headers": {
    "Access-Control-Allow-Origin": "*",
    "Access-Control-Allow-Methods": "OPTIONS, POST, GET",
    "Access-Control-Allow-Headers": "Content-Type"
  },
  "body": "{\"Attributes\": {\"vote_count\": 3}, \"ResponseMetadata\": {\"RequestId\":
  \"KTE6VSODD1IR0VEN2KTPDS9UT7VV4KQNSO5AEMVJF66Q9ASUAAJG\", \"HTTPStatusCode\": 200, \"HTTPHeaders\": {\"server\":
  \"Server\", \"date\": \"Tue, 08 Apr 2025 03:09:39 GMT\", \"content-type\": \"application/x-amz-json-1.0\",
  \"content-length\": \"39\", \"connection\": \"keep-alive\", \"x-amzn-requestid\":
```


Lambda > Functions > storeVote
DEPLOY
Deploy (Ctrl+Shift+U)
Test (Ctrl+Shift+I)
TEST EVENTS [SELECTED: STOREVOTETEST]
Create new test event
Private saved events
storeVotetest
ENVIRONMENT VARIABLES
Ln 27, Col 31 Spaces: 4 UTF-8 LF Python Lambda Layout: US
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

```python
def lambda_handler(event, context):
    http_method = event.get('httpMethod')

    if not http_method:
        return {
            'statusCode': 400,
            'headers': {'Access-Control-Allow-Origin': '*'},
            'body': json.dumps({'message': 'Error: Missing httpMethod in request'})
        }

    # Handle CORS preflight request
    if http_method == 'OPTIONS':
        return {
            'statusCode': 200,
            'headers': {
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': 'POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type'
            },
            'body': json.dumps({'message': 'CORS preflight response'})
        }
```

Code properties Info

Package size	SHA256 hash	Last modified
1,017 byte	bq+W0PABk3QH/Uq2VFEF5tx3hzkfx0aI3Xxqpd/Wo2g=	20 hours ago



Execution role

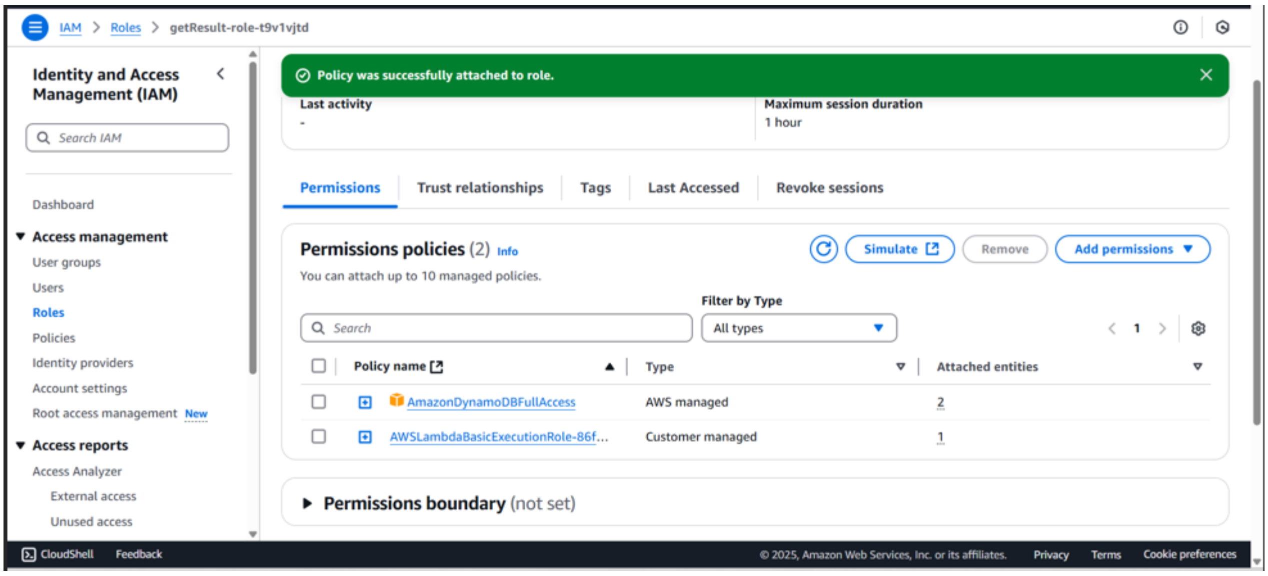
Role name
storeVote-role-9w36th8y

Resource summary

To view the resources and actions that your function has permission to access, choose a service.

AWS Application Auto Scaling
7 actions, 1 resource

Resource	Actions
All resources	Allow: application-autoscaling:DeleteScalingPolicy Allow: application-autoscaling:DeregisterScalableTarget Allow: application-autoscaling:DescribeScalableTargets Allow: application-autoscaling:DescribeScalingActivities



Policy was successfully attached to role.

Last activity
-

Maximum session duration
1 hour

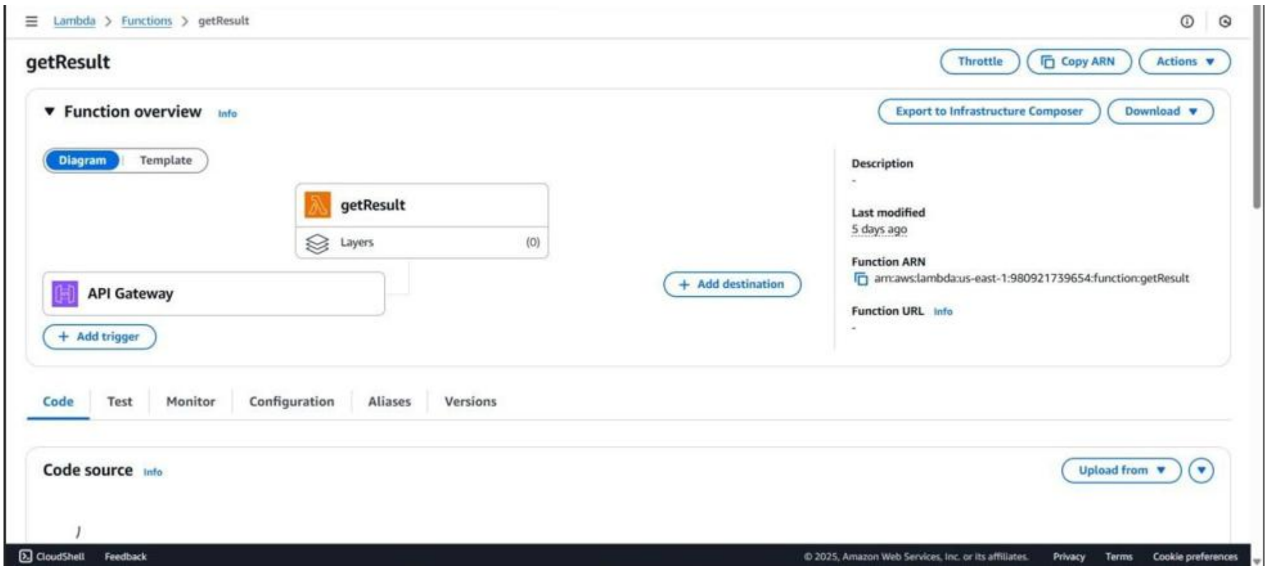
Permissions policies (2)

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AmazonDynamoDBFullAccess	AWS managed	2
AWSLambdaBasicExecutionRole-86f...	Customer managed	1

Permissions boundary (not set)

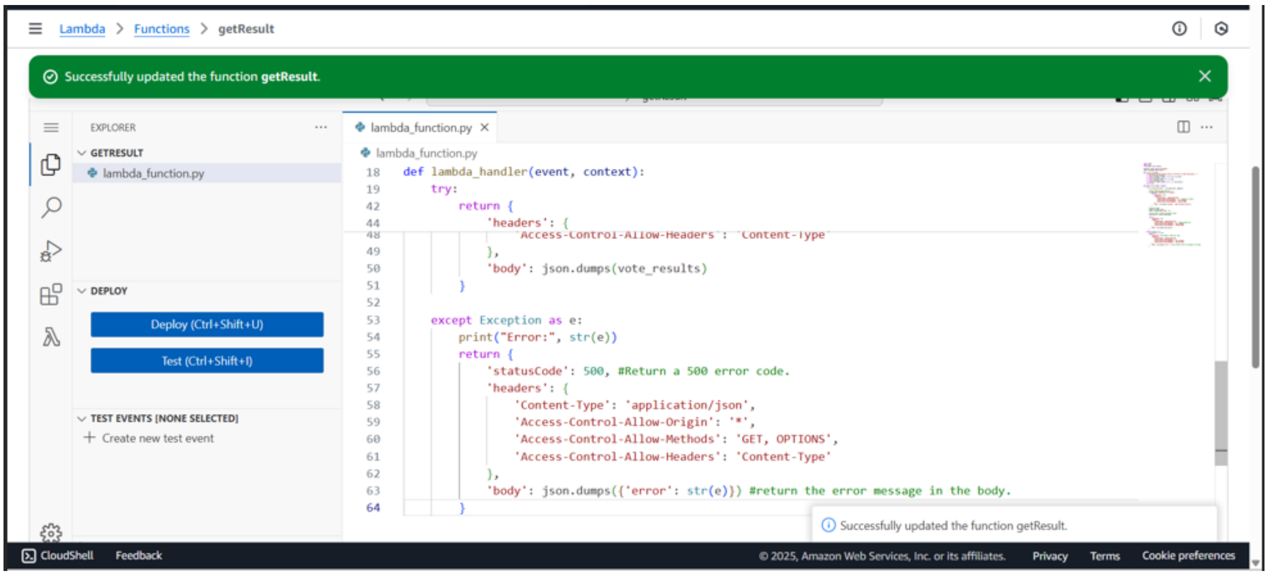


Description
-

Last modified
5 days ago

Function ARN
arn:aws:lambda:us-east-1:980921739654:function:getResult

Function URL Info
-

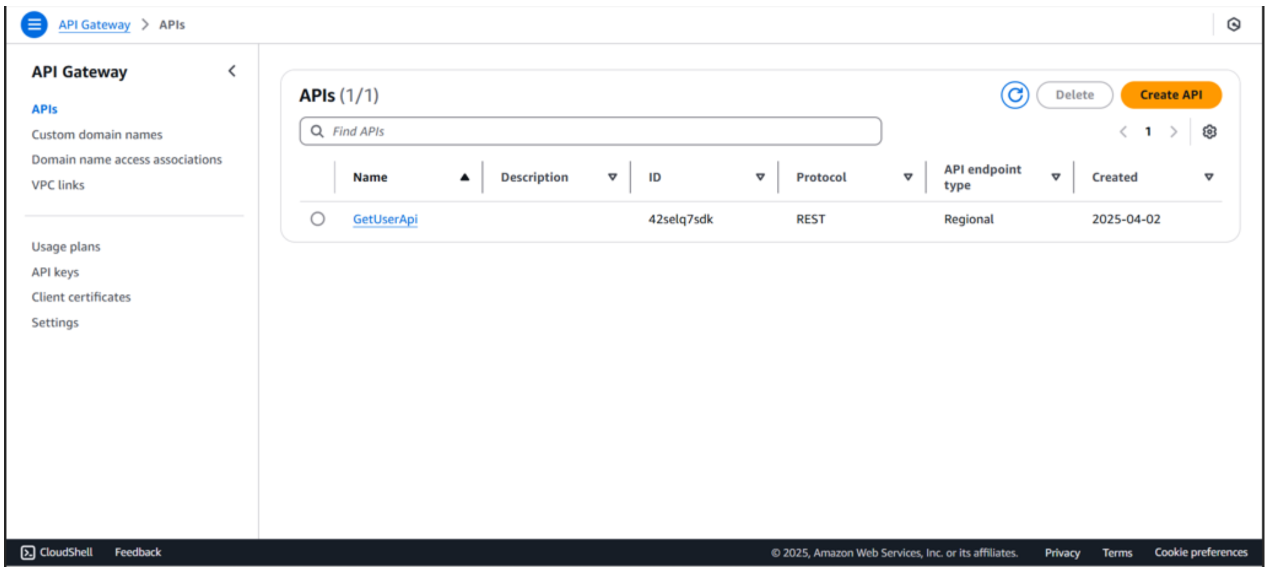


Successfully updated the function **getResult**.

```
def lambda_handler(event, context):
    try:
        return {
            'headers': {
                'Access-Control-Allow-Headers': 'Content-Type'
            },
            'body': json.dumps(vote_results)
        }

    except Exception as e:
        print("Error:", str(e))
        return {
            'statusCode': 500, #Return a 500 error code.
            'headers': {
                'Content-Type': 'application/json',
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': 'GET, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type'
            },
            'body': json.dumps({'error': str(e)}) #return the error message in the body.
        }
```

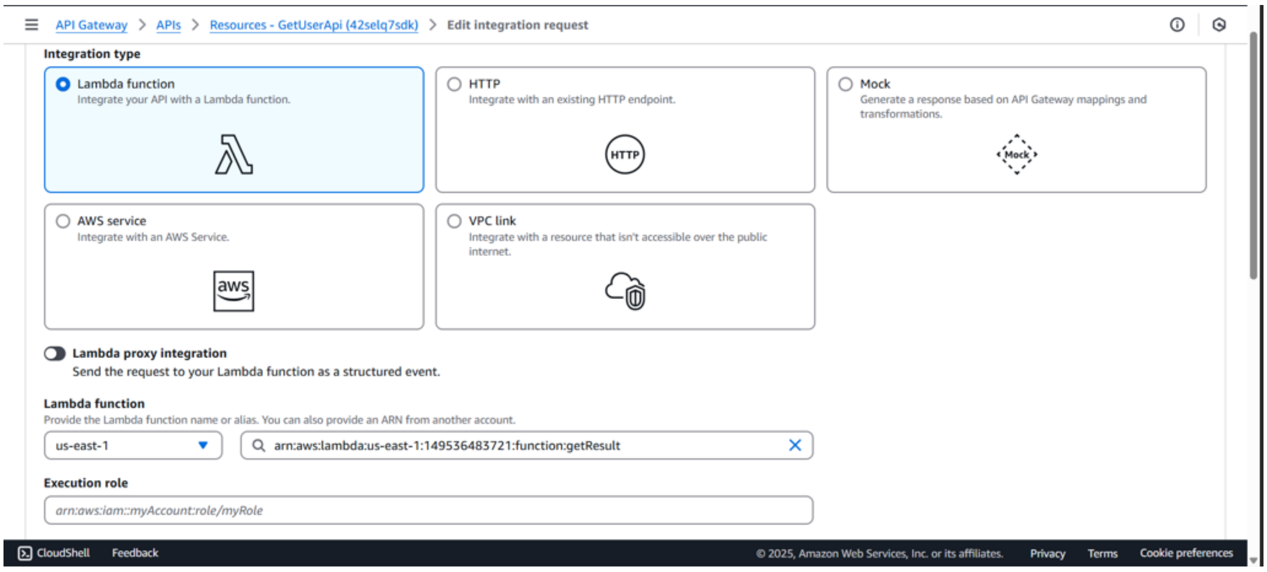
Successfully updated the function getResult.

FIG 13-26 : created a REST API using Amazon API Gateway to expose backend Lambda functions to the frontend. Two primary resources were added: /vote and /results. The /vote endpoint uses the POST method to allow users to cast votes, and the /results endpoint uses the GET method to fetch vote results. These methods were integrated with the corresponding AWS Lambda functions (storeVote and getResult) by choosing the Lambda integration type during method creation. CORS was enabled using the OPTIONS method for cross-origin access from the frontend. To allow API Gateway to invoke Lambda, we granted execution permissions by attaching the AWS-managed policy AWSLambdaRole or a custom role with the lambda:InvokeFunction action. Additionally, the Lambda functions were granted read/write access to DynamoDB using the AmazonDynamoDBFullAccess or a scoped-down policy. Finally, the API was deployed to a stage and tested using the browser and frontend code.



APIs (1/1)

Name	Description	ID	Protocol	API endpoint type	Created
GetUserApi		42selq7sdk	REST	Regional	2025-04-02


API Gateway > APIs > Resources - GetUserApi (42selq7sdk) > Edit integration request

Integration type

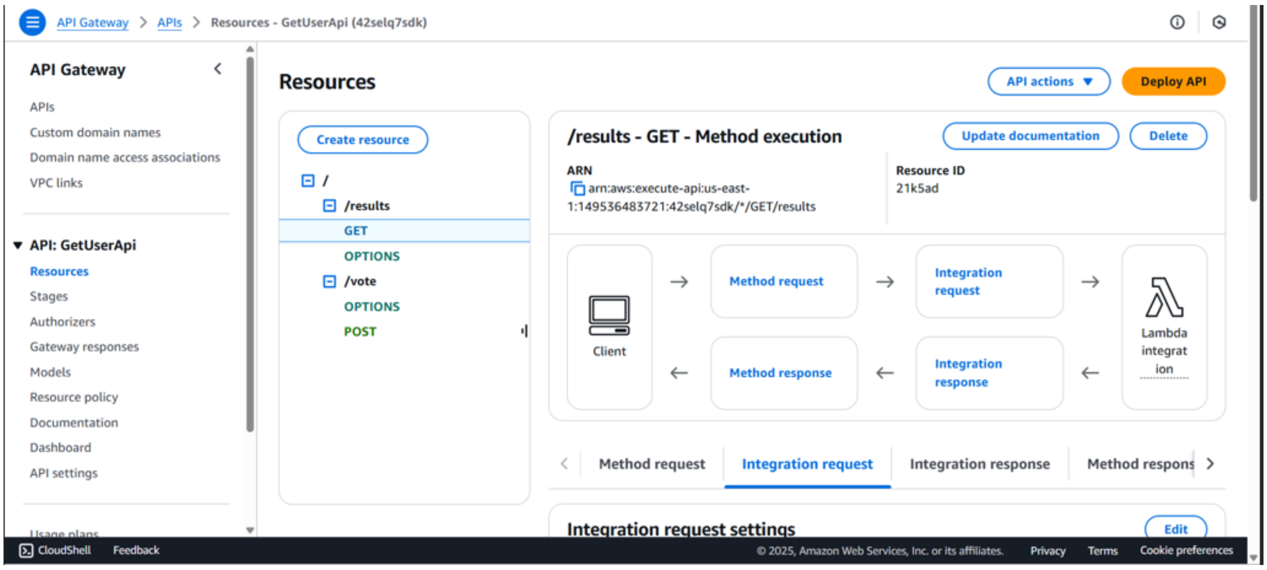
- [✓ Lambda function] Integrate your API with a Lambda function.
- HTTP Integrate with an existing HTTP endpoint.
- Mock Generate a response based on API Gateway mappings and transformations.
- AWS service Integrate with an AWS Service.
- VPC link Integrate with a resource that isn't accessible over the public internet.

Lambda proxy integration
Send the request to your Lambda function as a structured event.

Lambda function
Provide the Lambda function name or alias. You can also provide an ARN from another account.
us-east-1 | arn:aws:lambda:us-east-1:149536483721:function:getResult

Execution role
arn:aws:iam::myAccount:role/myRole

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences


API Gateway > APIs > Resources - GetUserApi (42selq7sdk)

API Gateway: APIs, Custom domain names, Domain name access associations, VPC links

API: GetUserApi: Resources, Stages, Authorizers, Gateway responses, Models, Resource policy, Documentation, Dashboard, API settings, Usage plans

Resources — API actions, Deploy API

Create resource

```
/
  /results
    GET
    OPTIONS
  /vote
    OPTIONS
    POST
```

/results - GET - Method execution — Update documentation, Delete

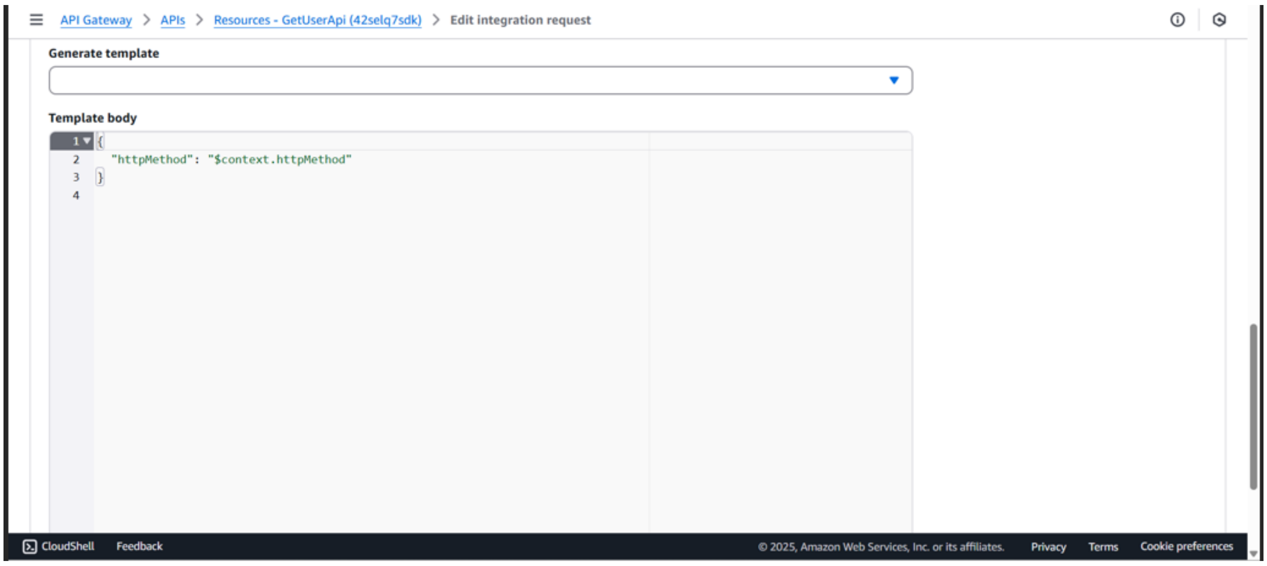
ARN
arn:aws:execute-api:us-east-1:149536483721:42selq7sdk/*/GET/results

Resource ID
21k5ad

Method request | Integration request | Integration response | Method respons

Integration request settings — Edit

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

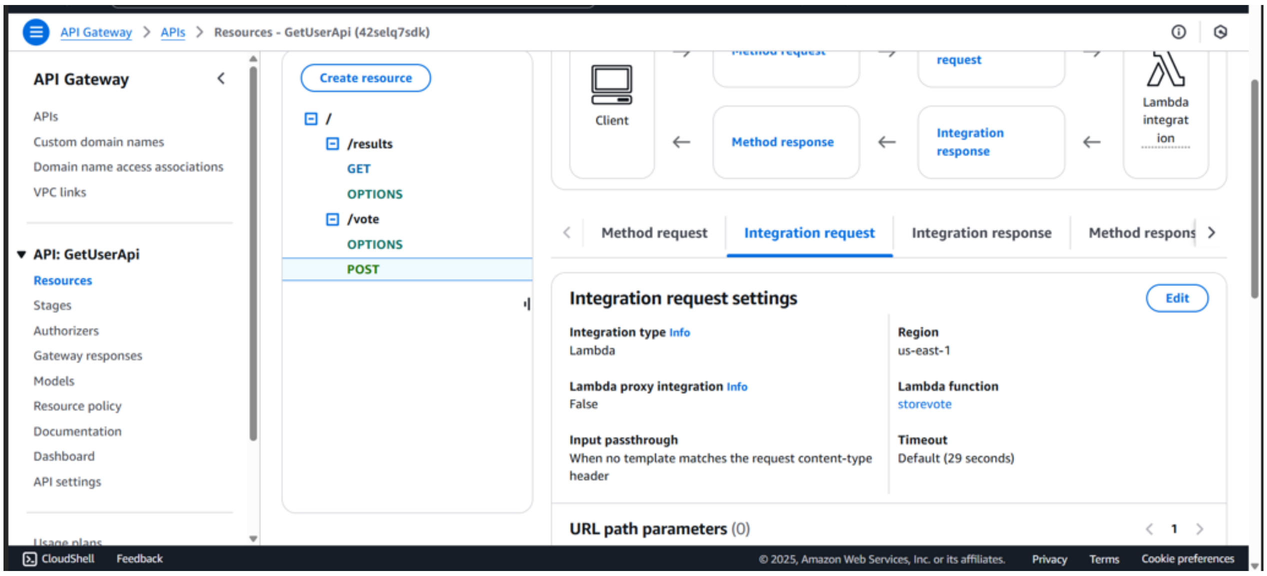

API Gateway > APIs > Resources - GetUserApi (42selq7sdk) > Edit integration request

Generate template

Template body

```
{
  "httpMethod": "$context.httpMethod"
}

```

CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



Integration request settings

Edit

Integration type Info
Lambda

Region
us-east-1

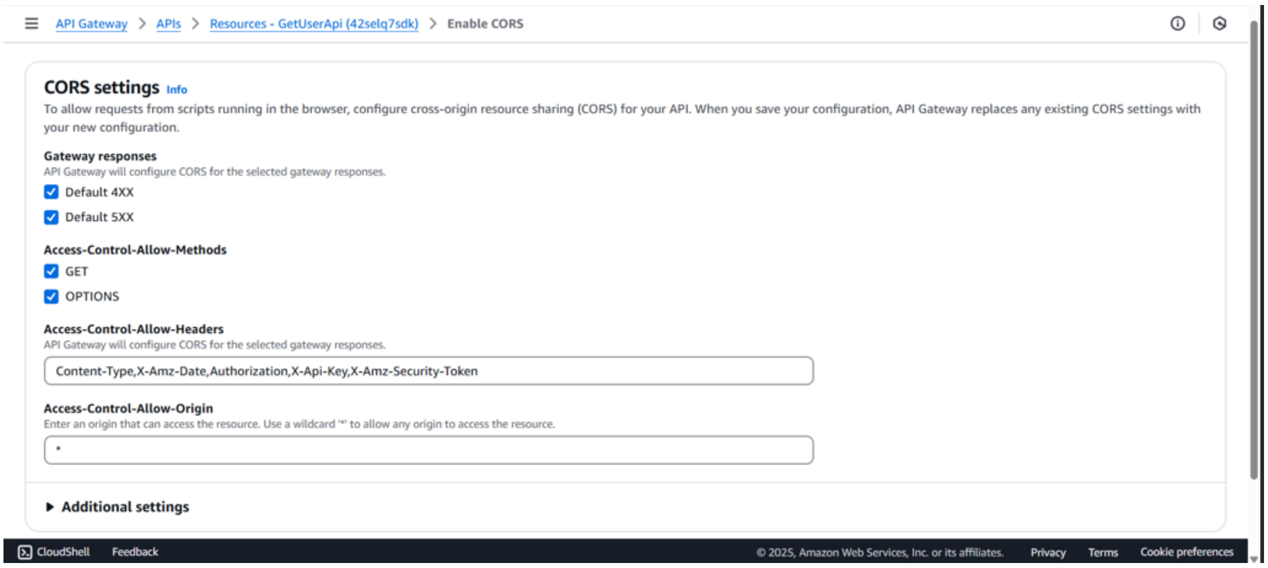
Lambda proxy integration Info
False

Lambda function
storevote

Input passthrough
When no template matches the request content-type header

Timeout
Default (29 seconds)

URL path parameters (0)



API Gateway > APIs > Resources - GetUserApi (42selq7sdk) > Enable CORS

CORS settings Info

To allow requests from scripts running in the browser, configure cross-origin resource sharing (CORS) for your API. When you save your configuration, API Gateway replaces any existing CORS settings with your new configuration.

Gateway responses
API Gateway will configure CORS for the selected gateway responses.
- [✓] Default 4XX
- [✓] Default 5XX

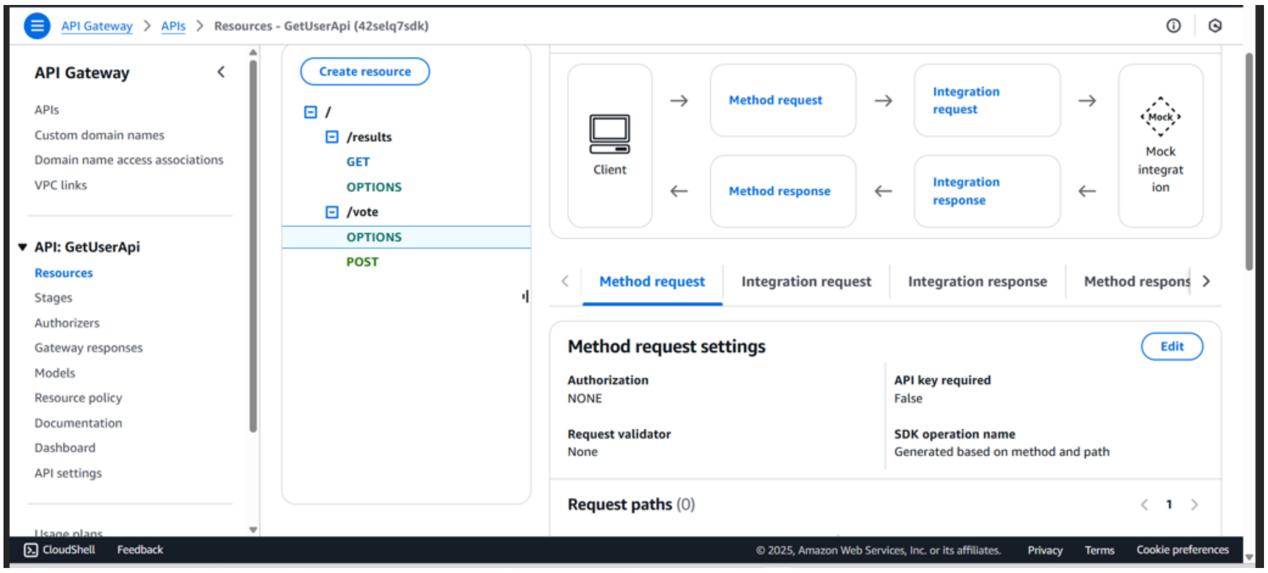
Access-Control-Allow-Methods
- [✓] GET
- [✓] OPTIONS

Access-Control-Allow-Headers
API Gateway will configure CORS for the selected gateway responses.
Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token

Access-Control-Allow-Origin
Enter an origin that can access the resource. Use a wildcard '*' to allow any origin to access the resource.
*

▸ Additional settings

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



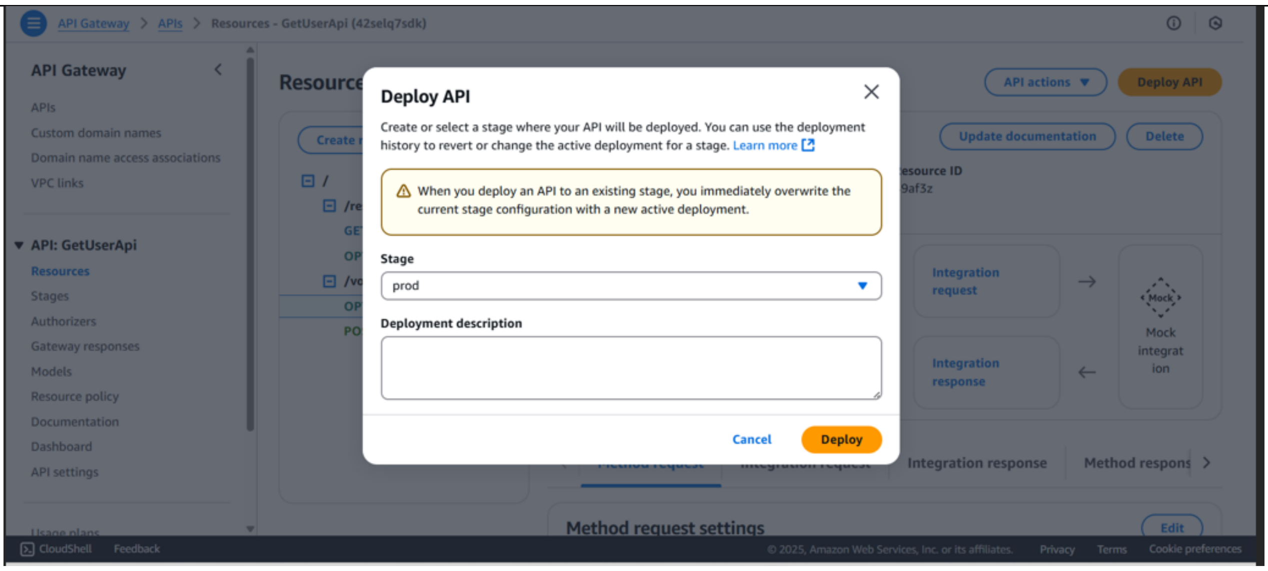
API Gateway > APIs > Resources - GetUserApi (42selq7sdk)

Method request settings

Authorization NONE	**API key required** False
Request validator None	**SDK operation name** Generated based on method and path

Request paths (0)

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences


API Gateway > APIs > Resources - GetUserApi (42selq7sdk)

Deploy API

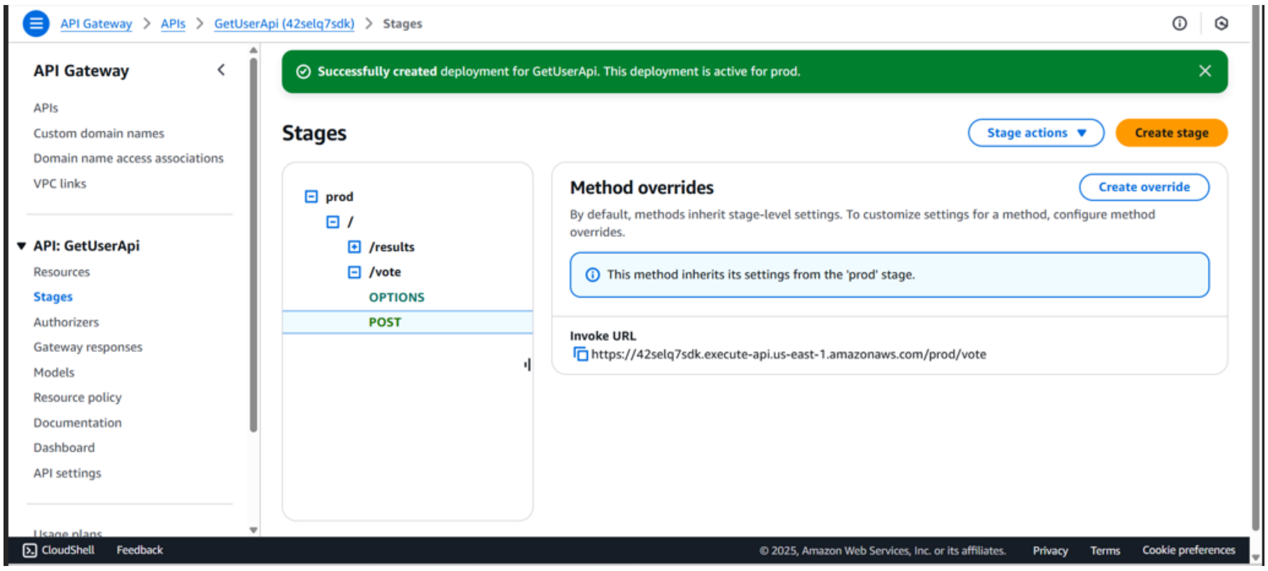
Create or select a stage where your API will be deployed. You can use the deployment history to revert or change the active deployment for a stage. Learn more

When you deploy an API to an existing stage, you immediately overwrite the current stage configuration with a new active deployment.

Stage: prod

Deployment description:

Cancel | Deploy


API Gateway > APIs > GetUserApi (42selq7sdk) > Stages

Successfully created deployment for GetUserApi. This deployment is active for prod.

Stages

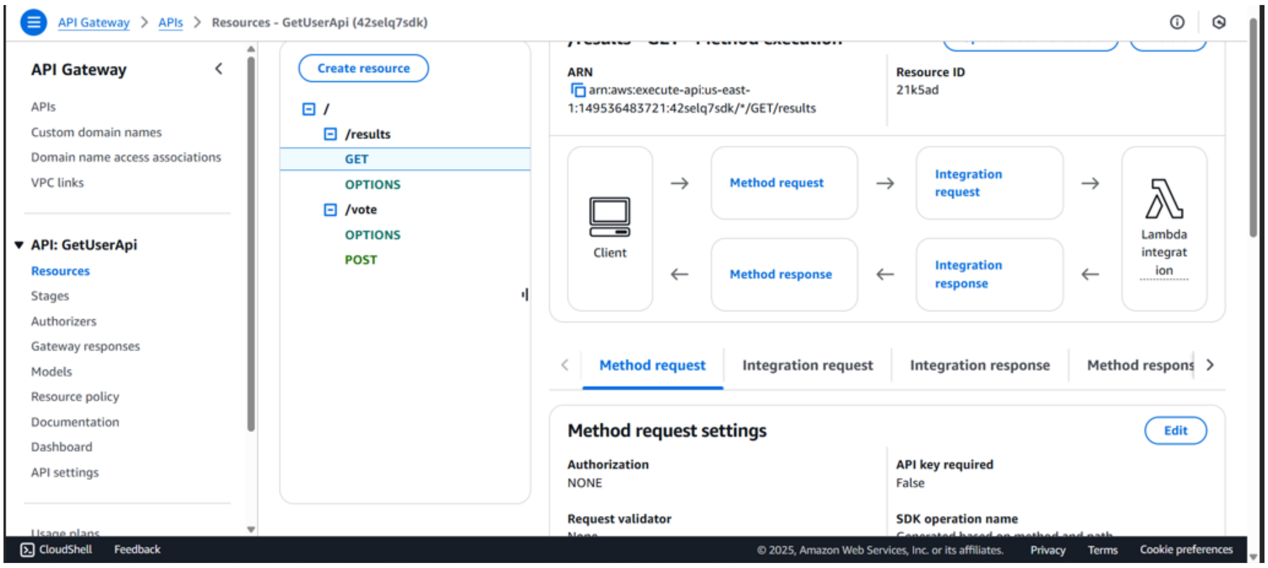
```
prod
  /
    /results
    /vote
      OPTIONS
      POST
```

Method overrides

By default, methods inherit stage-level settings. To customize settings for a method, configure method overrides.

This method inherits its settings from the 'prod' stage.

Invoke URL
https://42selq7sdk.execute-api.us-east-1.amazonaws.com/prod/vote

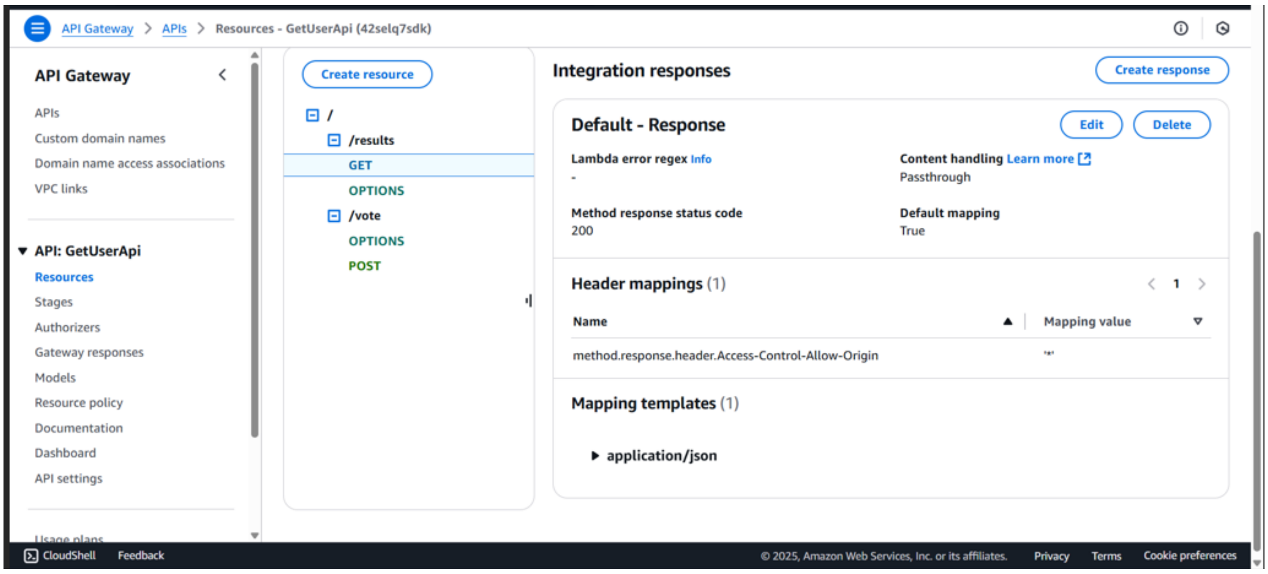


ARN
arn:aws:execute-api:us-east-1:149536483721:42selq7sdk/*/GET/results

Resource ID
21k5ad

Method request settings

Authorization	API key required
NONE	False
Request validator	SDK operation name



Integration responses

Default - Response

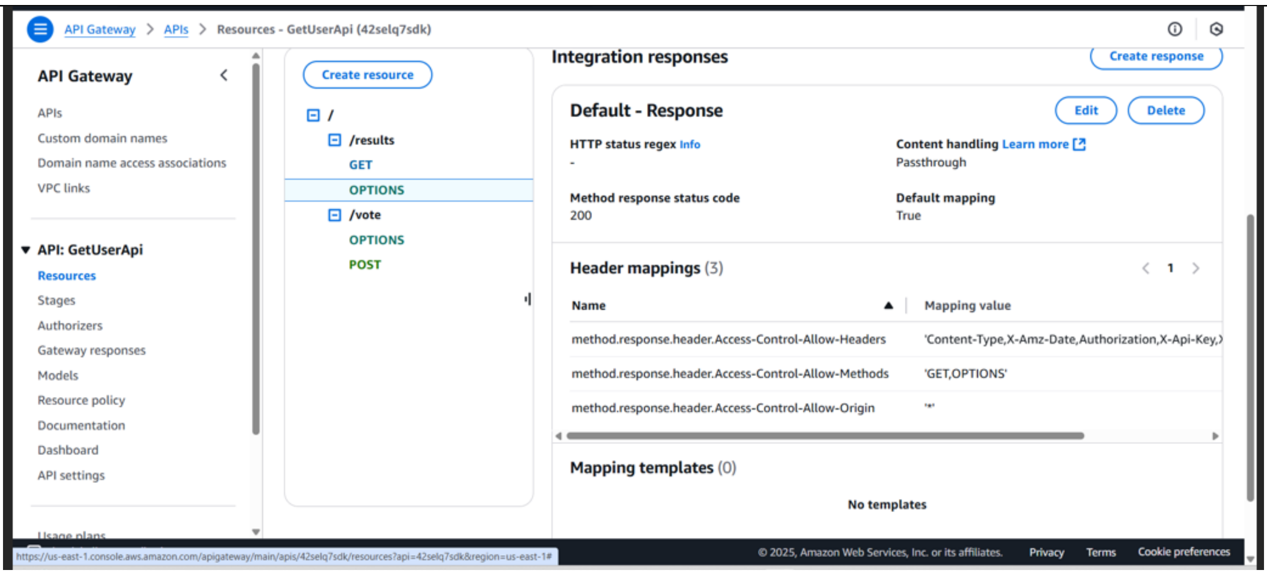
Lambda error regex Info	Content handling Learn more
-	Passthrough
Method response status code	Default mapping
200	True

Header mappings (1)

Name	Mapping value
method.response.header.Access-Control-Allow-Origin	'*'

Mapping templates (1)

▸ application/json



Integration responses

Default - Response

HTTP status regex Info
-

Content handling Learn more
Passthrough

Method response status code
200

Default mapping
True

Header mappings (3)

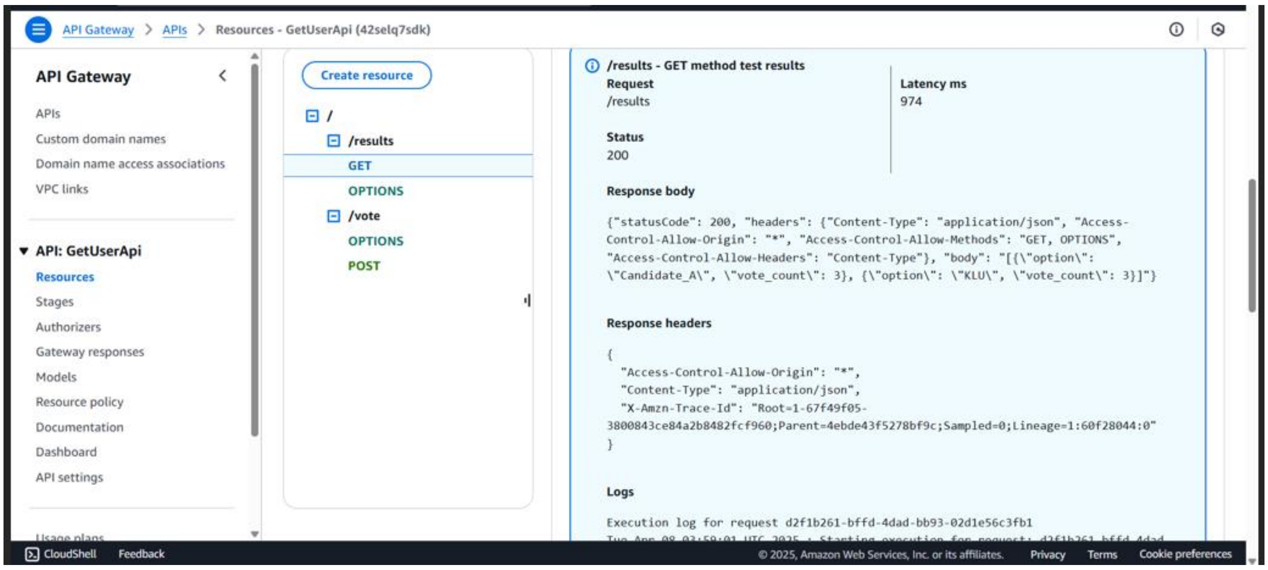
Name	Mapping value
method.response.header.Access-Control-Allow-Headers	'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X
method.response.header.Access-Control-Allow-Methods	'GET,OPTIONS'
method.response.header.Access-Control-Allow-Origin	'*'

Mapping templates (0)

No templates

https://us-east-1.console.aws.amazon.com/apigateway/main/apis/42selq7sdk/resources?api=42selq7sdk&region=us-east-1#

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



/results - GET method test results

Request
/results

Latency ms
974

Status
200

Response body

```
{"statusCode": 200, "headers": {"Content-Type": "application/json", "Access-
Control-Allow-Origin": "*", "Access-Control-Allow-Methods": "GET, OPTIONS",
"Access-Control-Allow-Headers": "Content-Type"}, "body": "[{\"option\":
\"Candidate_A\", \"vote_count\": 3}, {\"option\": \"KLU\", \"vote_count\": 3}]"}
```

Response headers

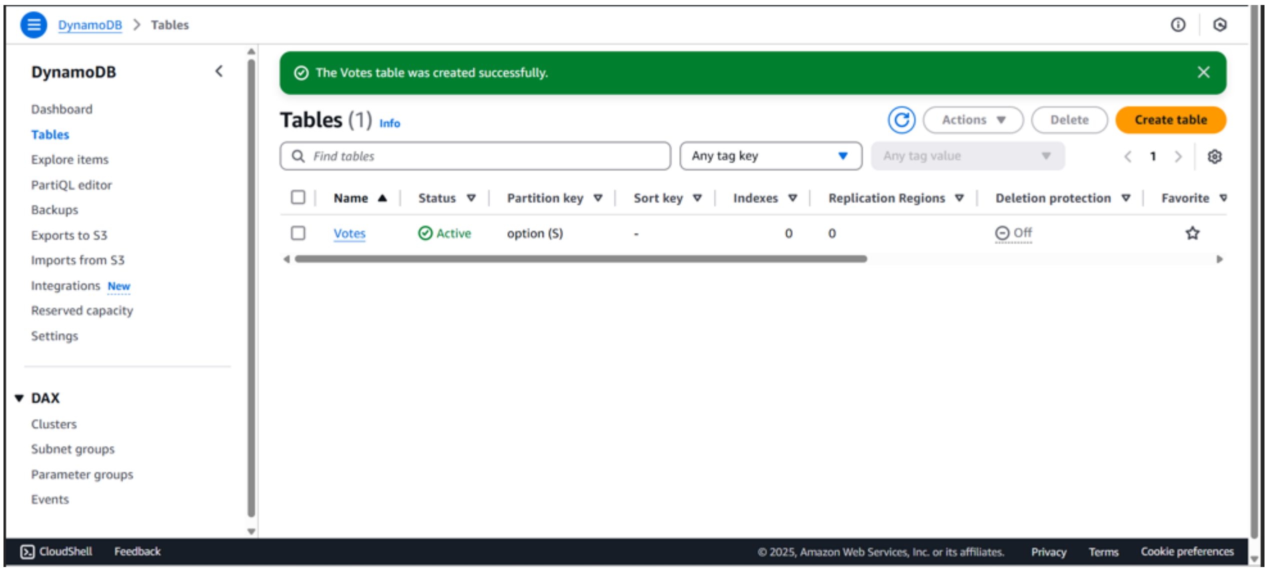
```
{
  "Access-Control-Allow-Origin": "*",
  "Content-Type": "application/json",
  "X-Amzn-Trace-Id": "Root=1-67f49f05-
3800843ce84a2b8482fcf960;Parent=4ebde43f5278bf9c;Sampled=0;Lineage=1:60f28044:0"
}
```

Logs

```
Execution log for request d2f1b261-bffd-4dad-bb93-02d1e56c3fb1
```

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

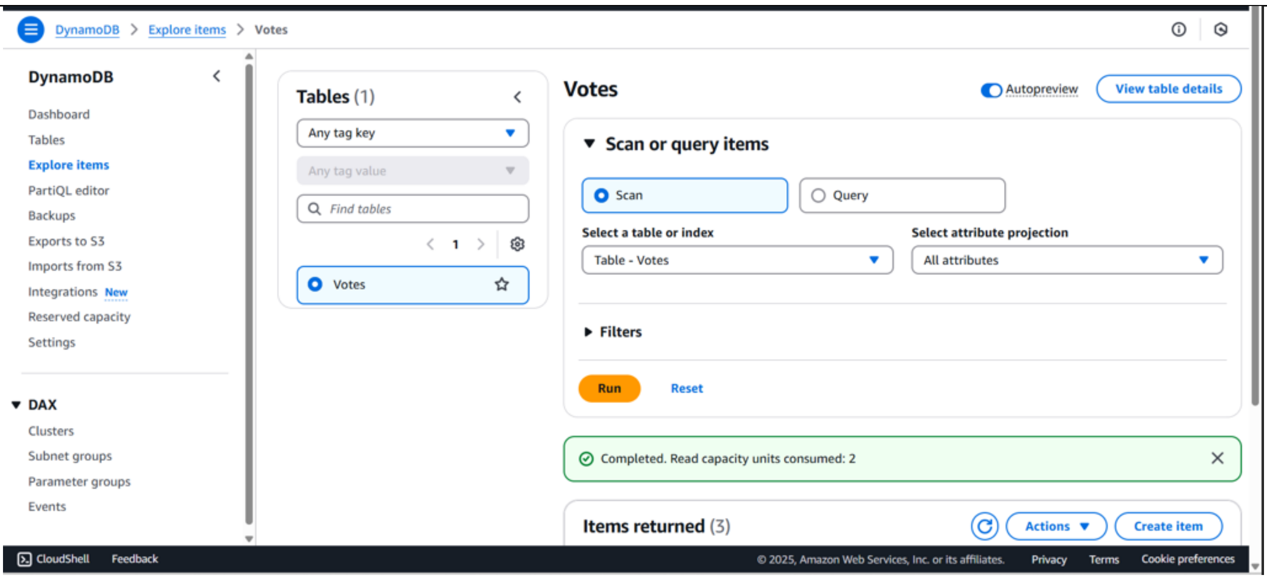
FIG 27: We used Amazon DynamoDB as the backend database to store vote data in the Cognition application. A table named Votes was created with attributes option (university name) and vote_count (number of votes). The storeVote Lambda function updates this table when a vote is cast, and the getResult function reads the data to display results. This serverless NoSQL database ensures fast performance, scalability, and easy integration with AWS Lambda.



The Votes table was created successfully.

Tables (1) Info

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite
Votes	Active	option (S)	-	0	0	Off	☆


DynamoDB > Explore items > Votes

Votes

Scan or query items

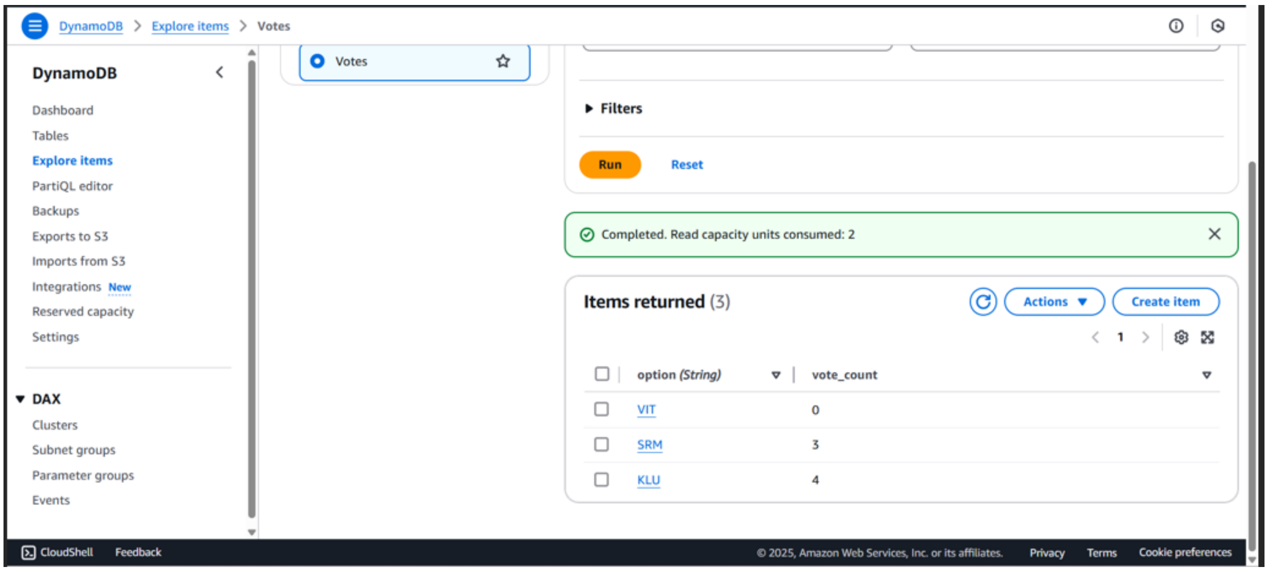
[✓ Scan] Query

Select a table or index: Table - Votes

Select attribute projection: All attributes

Completed. Read capacity units consumed: 2

Items returned (3)


DynamoDB > Explore items > Votes

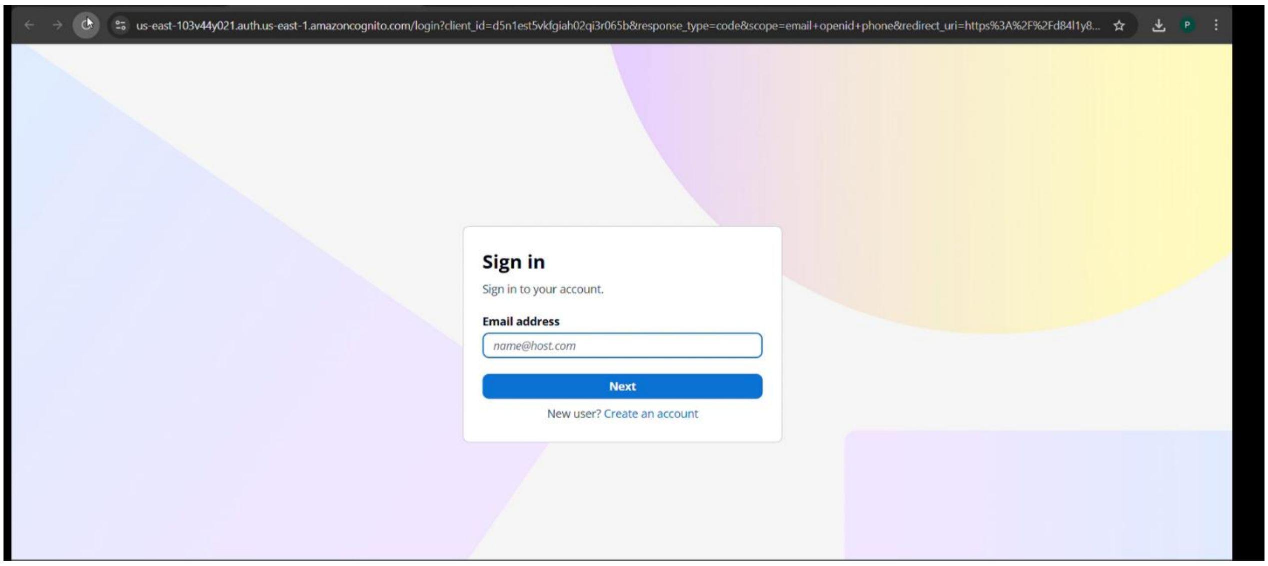
Completed. Read capacity units consumed: 2

Items returned (3)

option (String)	vote_count
VIT	0
SRM	3
KLU	4

OUTPUT

<u>Signing with new user:</u>

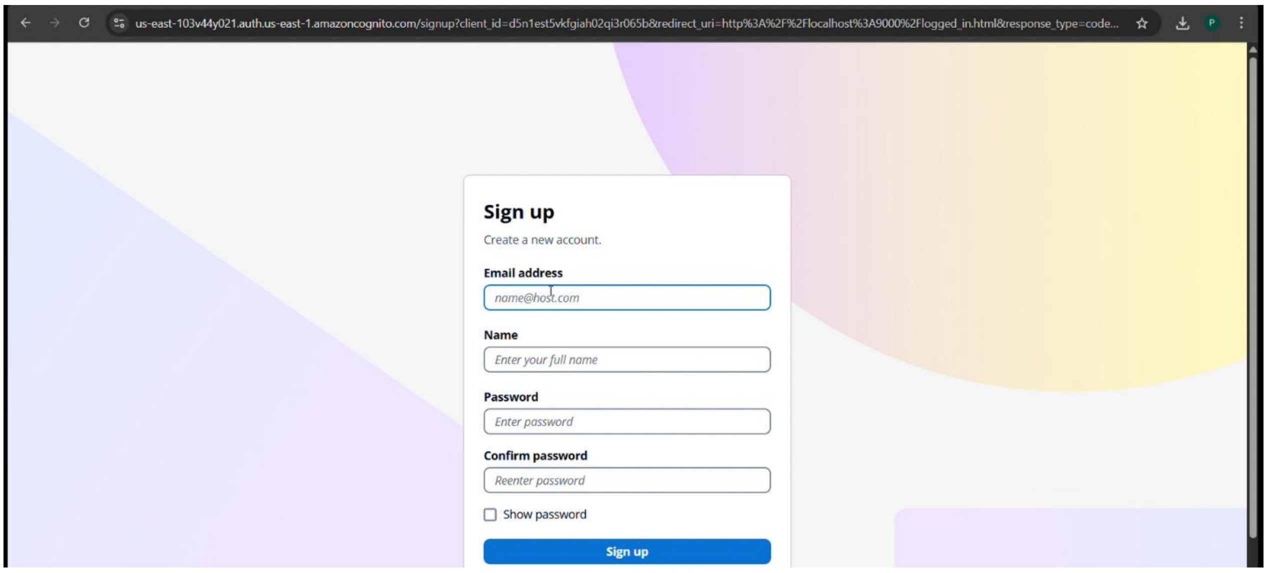

us-east-103v44y021.auth.us-east-1.amazoncognito.com/login?client_id=d5n1est5vkfgiah02qi3r065b&response_type=code&scope=email+openid+phone&redirect_uri=https%3A%2F%2Fd84l1y8...

Sign in

Sign in to your account.

Email address

name@host.com


us-east-103v44y021.auth.us-east-1.amazoncognito.com/signup?client_id=d5n1est5vkfgiah02qi3r065b&redirect_uri=http%3A%2F%2Flocalhost%3A9000%2Flogged_in.html&response_type=code...

Sign up

Create a new account.

Email address

name@host.com

Name

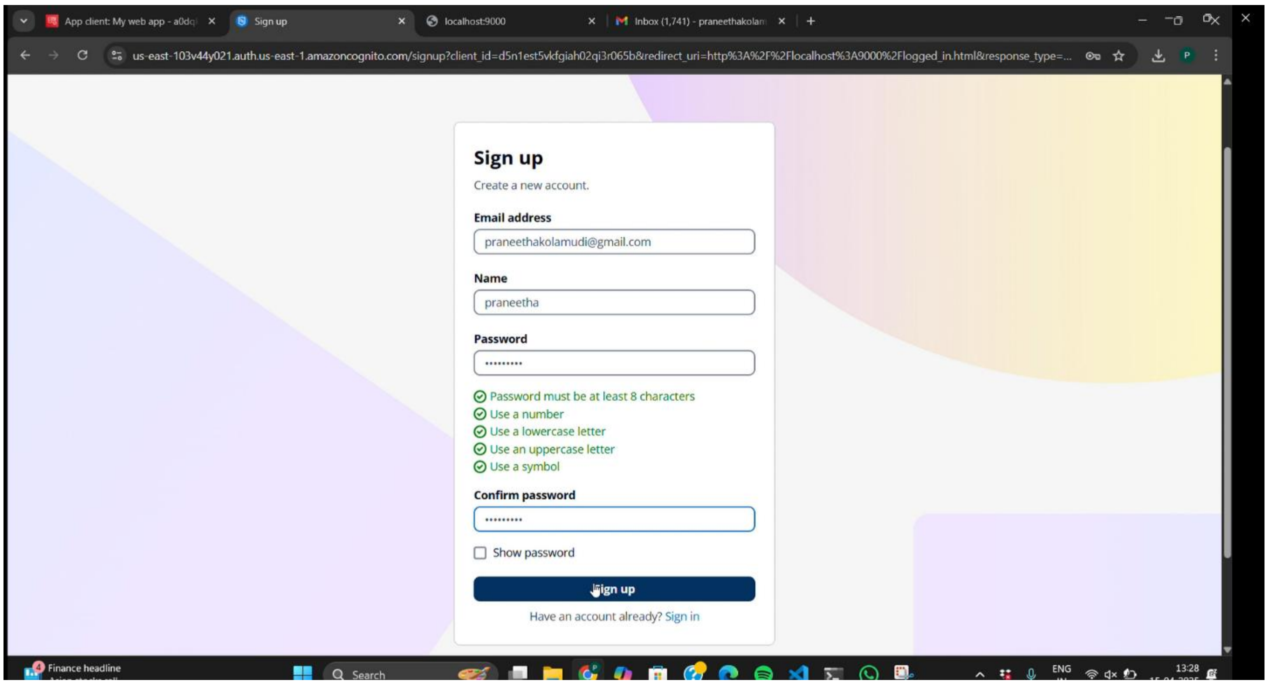
Enter your full name

Password

Enter password

Confirm password

Reenter password


App client: My web app - a0dq
Sign up
localhost:9000
Inbox (1,741) - praneethakolam
us-east-103v44y021.auth.us-east-1.amazoncognito.com/signup?client_id=d5n1est5vkfgiah02qi3r065b&redirect_uri=http%3A%2F%2Flocalhost%3A9000%2Flogged_in.html&response_type=...

Sign up

Create a new account.

Email address: praneethakolamudi@gmail.com

Name: praneetha

Password: ••••••••••

- ✓ Password must be at least 8 characters
- ✓ Use a number
- ✓ Use a lowercase letter
- ✓ Use an uppercase letter
- ✓ Use a symbol

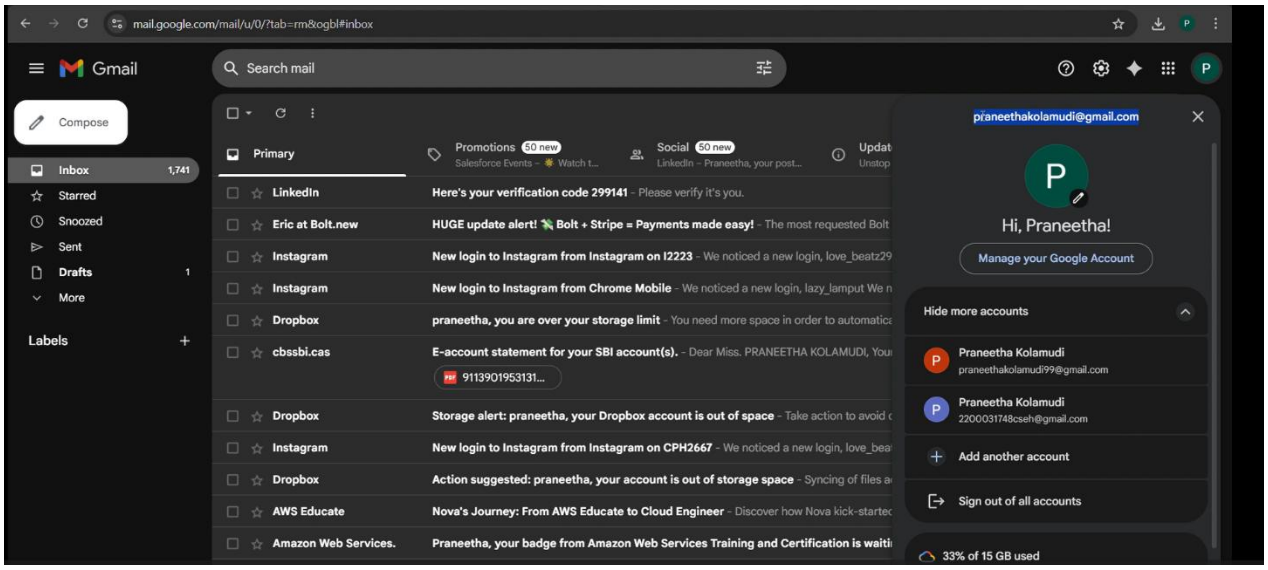
Confirm password: ••••••••••

☐ Show password

Sign up

Have an account already? Sign in

Finance headline
Search
ENG
13:28



LinkedIn	Here's your verification code 299141 - Please verify it's you.
Eric at Bolt.new	HUGE update alert! Bolt + Stripe = Payments made easy! - The most requested Bolt
Instagram	New login to Instagram from Instagram on I2223 - We noticed a new login, love_beatz29
Instagram	New login to Instagram from Chrome Mobile - We noticed a new login, lazy_lamput We n
Dropbox	praneetha, you are over your storage limit - You need more space in order to automatica
cbssbi.cas	E-account statement for your SBI account(s). - Dear Miss. PRANEETHA KOLAMUDI, Your 9113901953131...
Dropbox	Storage alert: praneetha, your Dropbox account is out of space - Take action to avoid c
Instagram	New login to Instagram from Instagram on CPH2667 - We noticed a new login, love_beat
Dropbox	Action suggested: praneetha, your account is out of storage space - Syncing of files a
AWS Educate	Nova's Journey: From AWS Educate to Cloud Engineer - Discover how Nova kick-startec
Amazon Web Services.	Praneetha, your badge from Amazon Web Services Training and Certification is waiti

praneethakolamudi@gmail.com

Hi, Praneetha!

Manage your Google Account

Hide more accounts

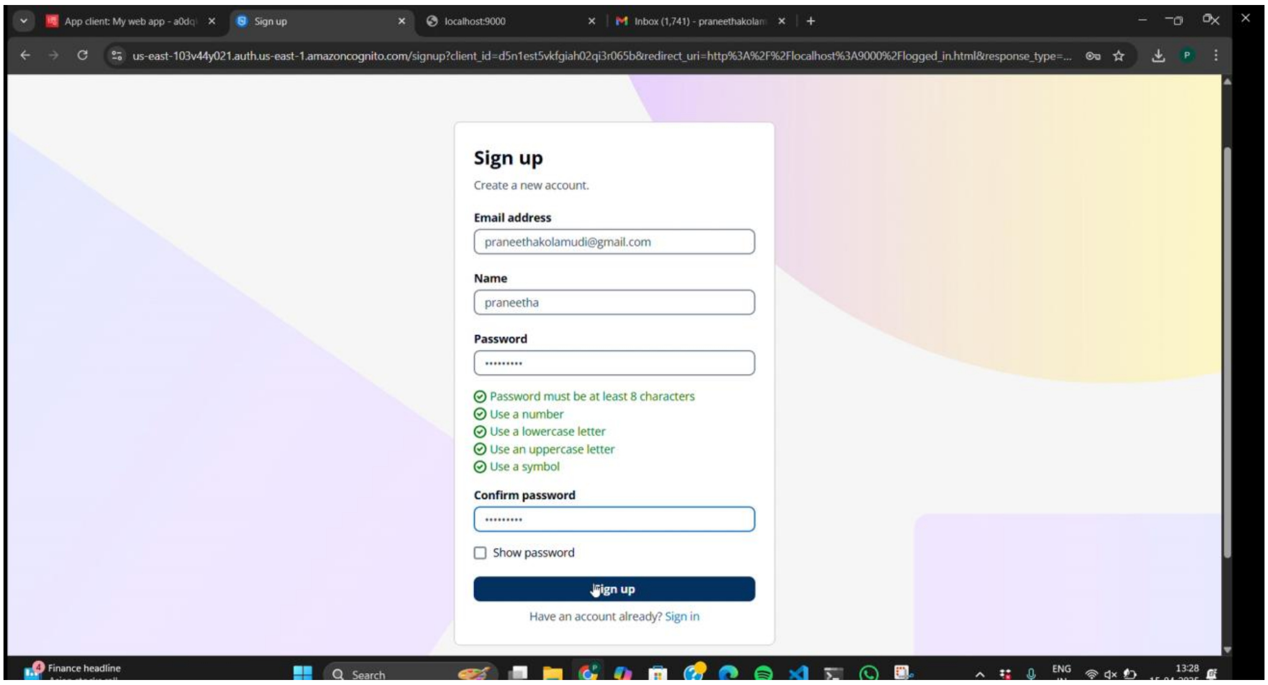
Praneetha Kolamudi praneethakolamudi99@gmail.com

Praneetha Kolamudi 2200031748cseh@gmail.com

Add another account

Sign out of all accounts

33% of 15 GB used



us-east-103v44y021.auth.us-east-1.amazoncognito.com/signup?client_id=d5n1est5vkfgiah02qi3r065b&redirect_uri=http%3A%2F%2Flocalhost%3A9000%2Flogged_in.html&response_type=...

Sign up

Create a new account.

Email address

praneethakolamudi@gmail.com

Name

praneetha

Password

•••••••••

- ✓ Password must be at least 8 characters
- ✓ Use a number
- ✓ Use a lowercase letter
- ✓ Use an uppercase letter
- ✓ Use a symbol

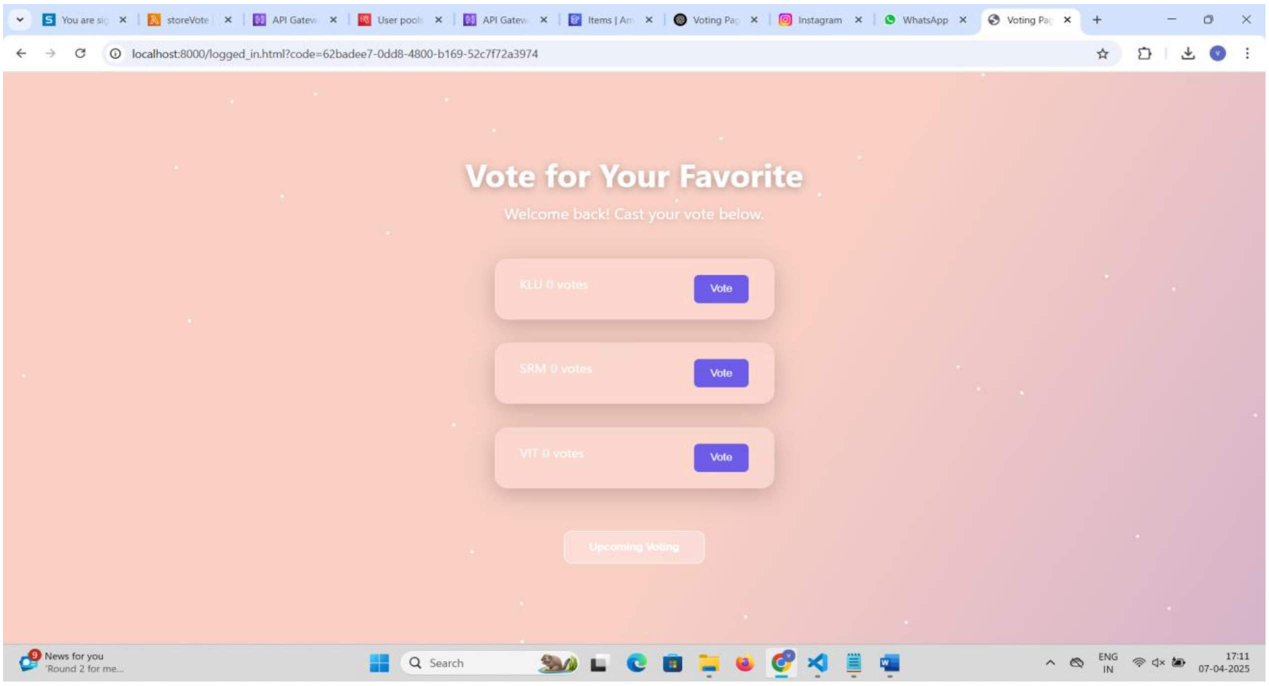
Confirm password

•••••••••

☐ Show password

Sign up

Have an account already? Sign in



localhost:8000/logged_in.html?code=62badee7-0dd8-4800-b169-52c7f72a3974

Vote for Your Favorite

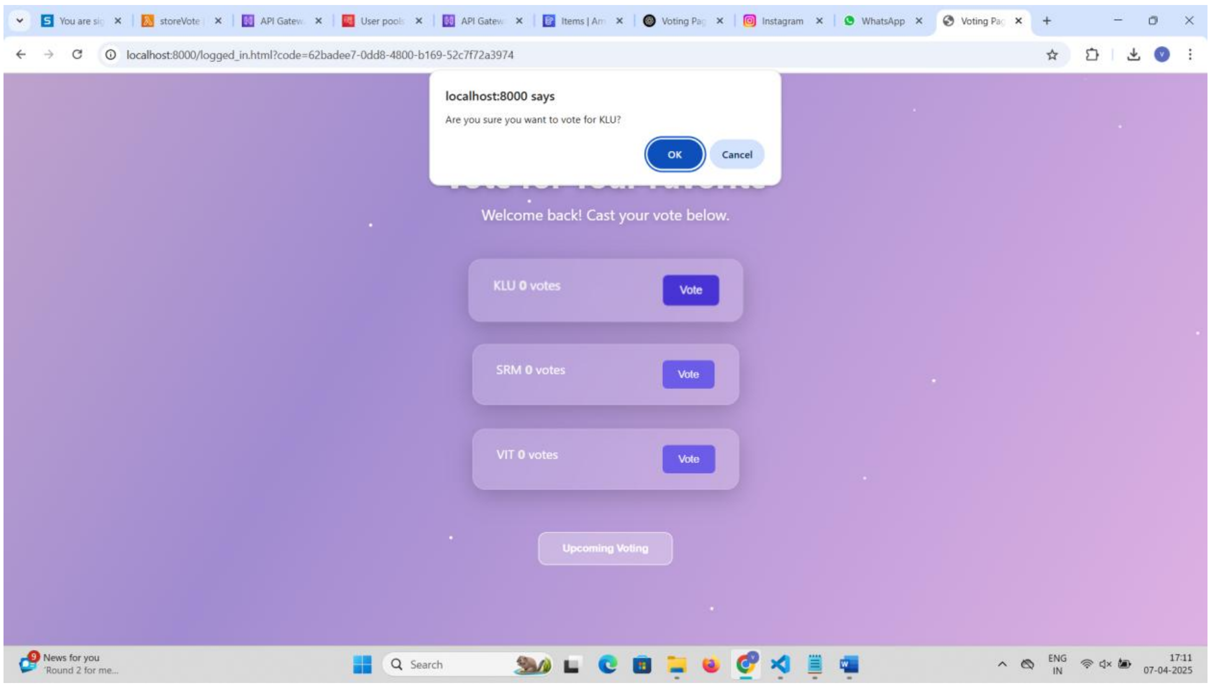
Welcome back! Cast your vote below.

KLU 0 votes — Vote

SRM 0 votes — Vote

VIT 0 votes — Vote

Upcoming Voting


localhost:8000/logged_in.html?code=62badee7-0dd8-4800-b169-52c7f72a3974

localhost:8000 says

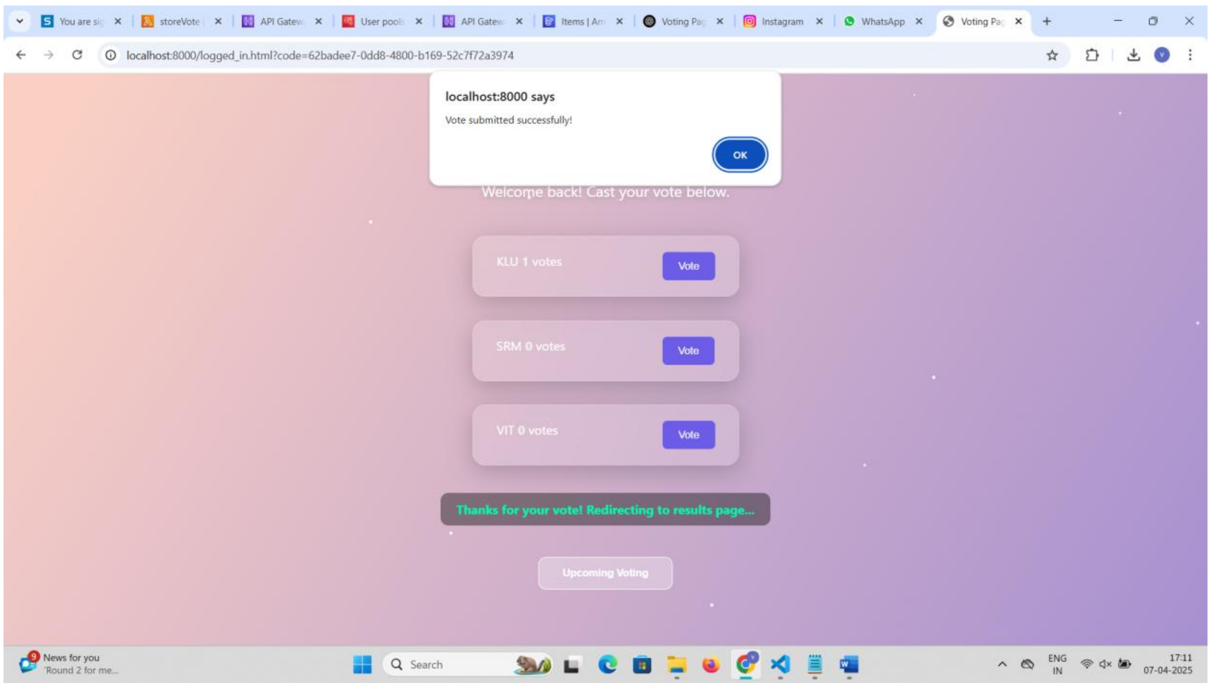
Are you sure you want to vote for KLU?

Welcome back! Cast your vote below.

KLU **0** votes

SRM **0** votes

VIT **0** votes


localhost:8000/logged_in.html?code=62badee7-0dd8-4800-b169-52c7f72a3974

localhost:8000 says

Vote submitted successfully!

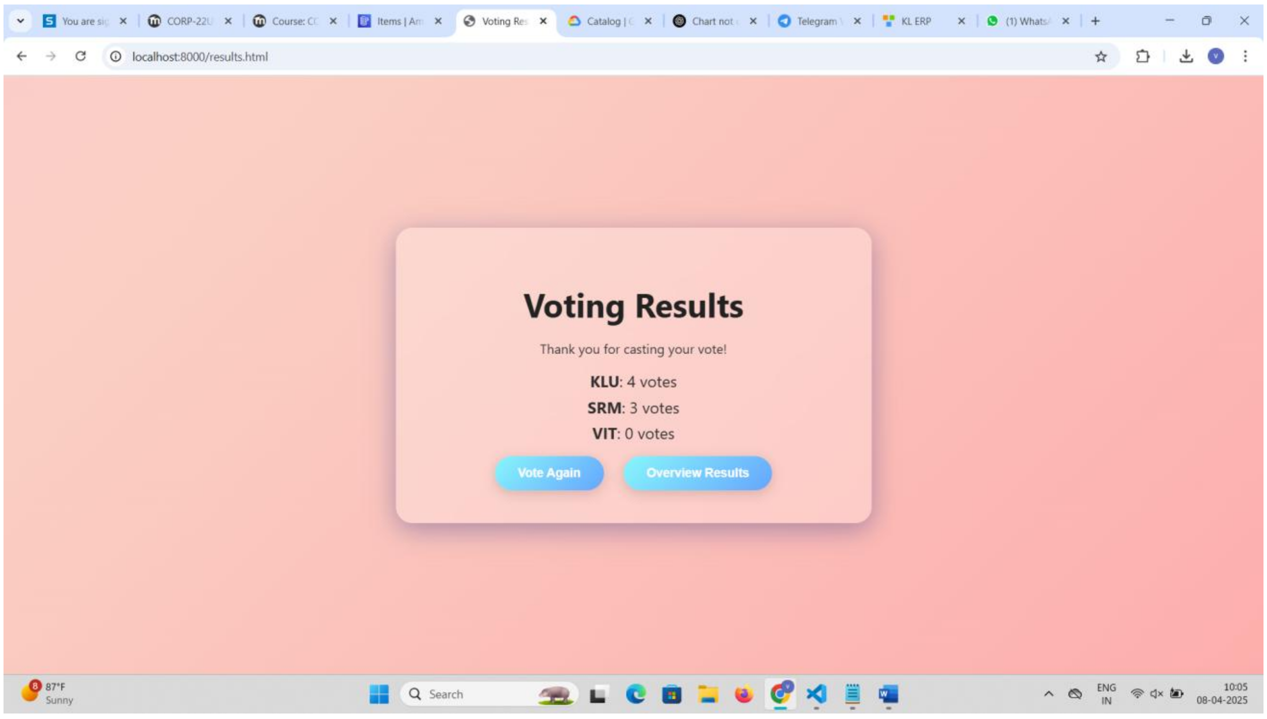
Welcome back! Cast your vote below.

KLU **1** votes

SRM **0** votes

VIT **0** votes

Thanks for your vote! Redirecting to results page...



localhost:8000/results.html

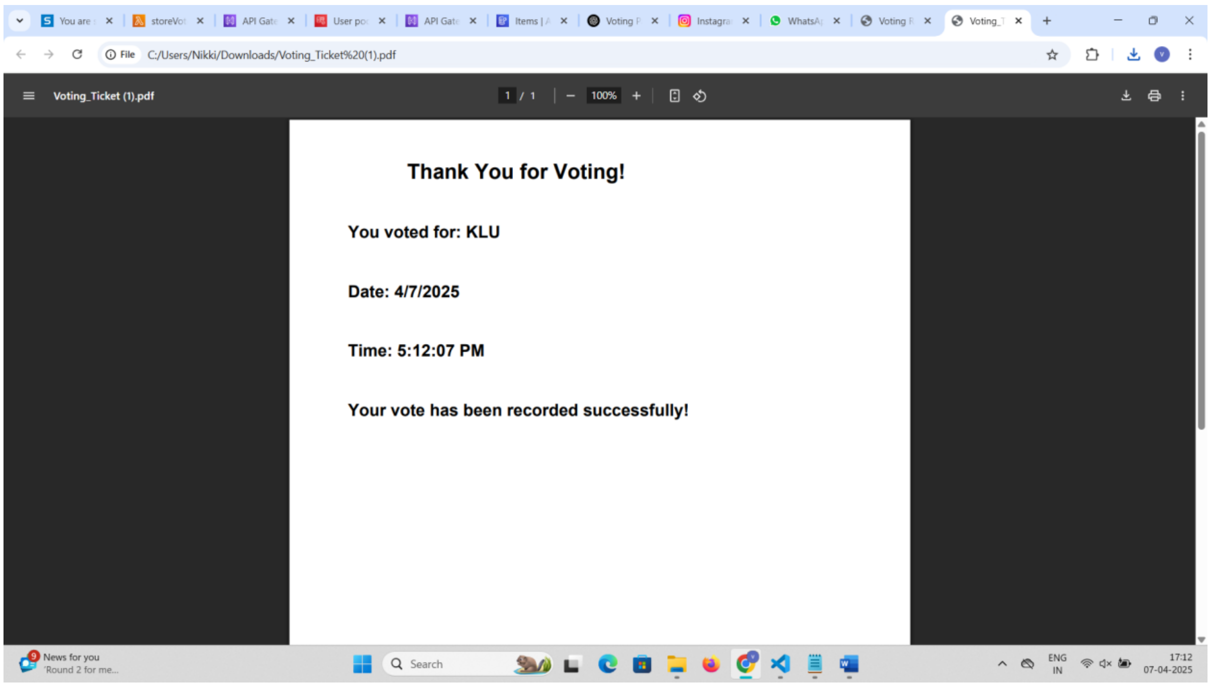
Voting Results

Thank you for casting your vote!

KLU: 4 votes

SRM: 3 votes

VIT: 0 votes



C:/Users/Nikki/Downloads/Voting_Ticket%20(1).pdf

Voting_Ticket (1).pdf

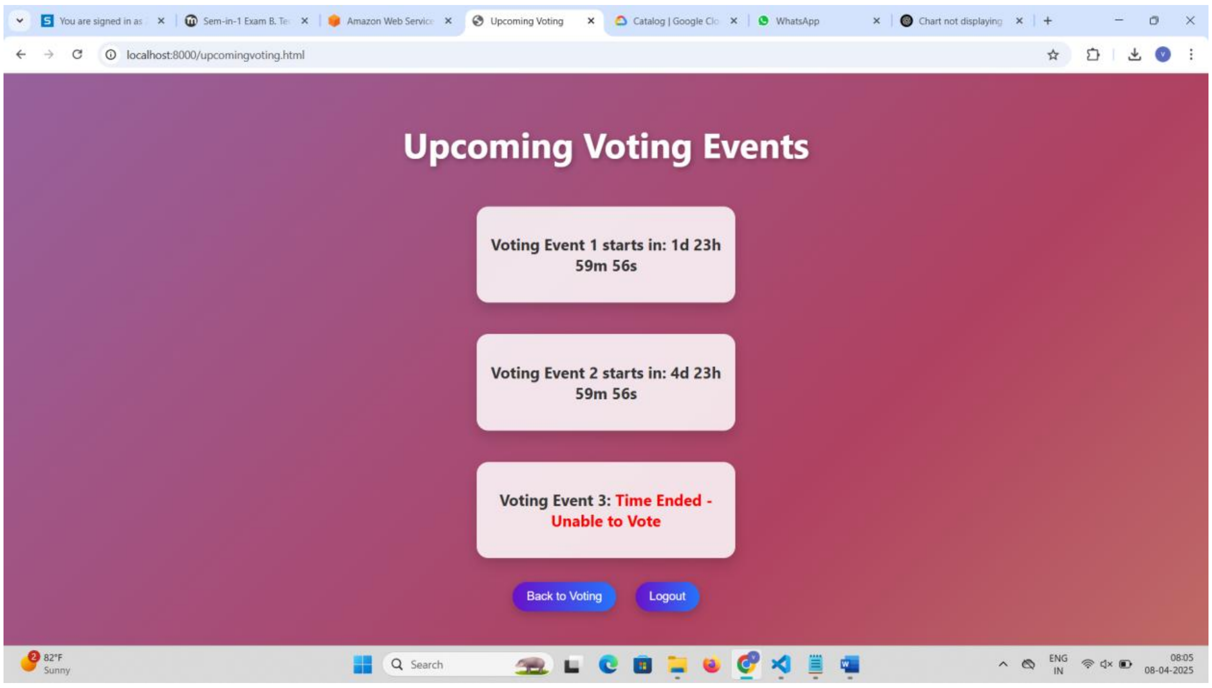
Thank You for Voting!

You voted for: KLU

Date: 4/7/2025

Time: 5:12:07 PM

Your vote has been recorded successfully!

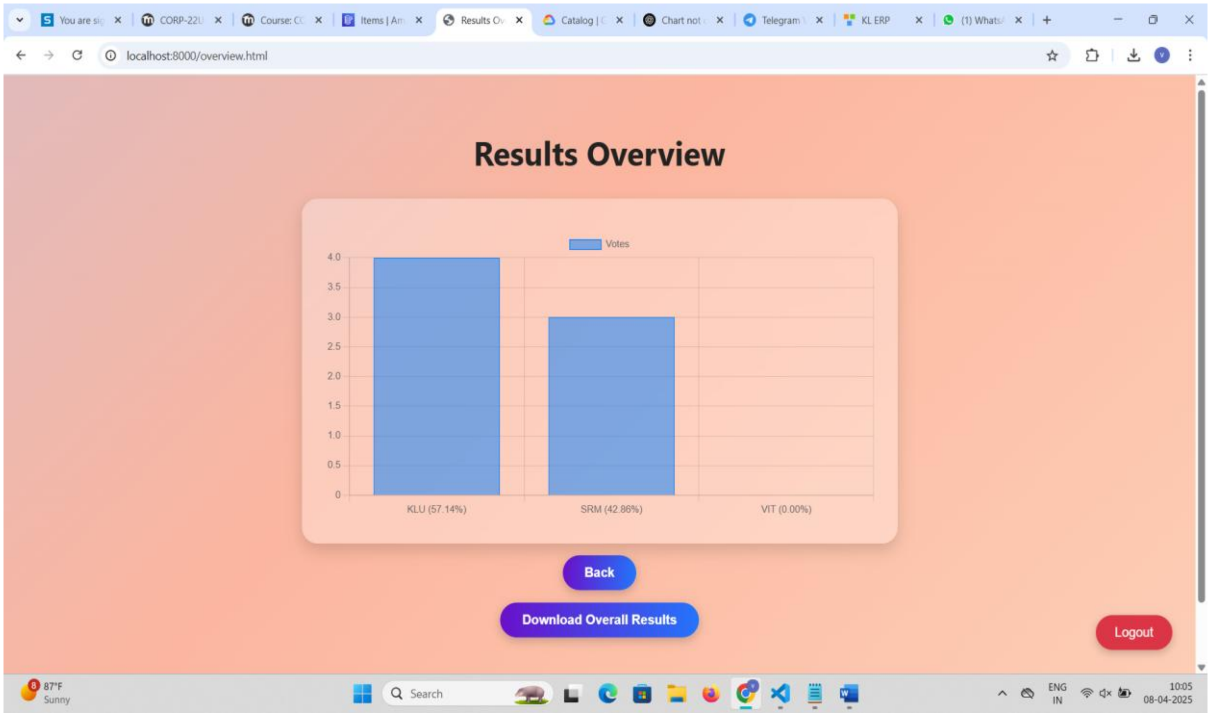

localhost:8000/upcomingvoting.html

Upcoming Voting Events

Voting Event 1 starts in: 1d 23h 59m 56s

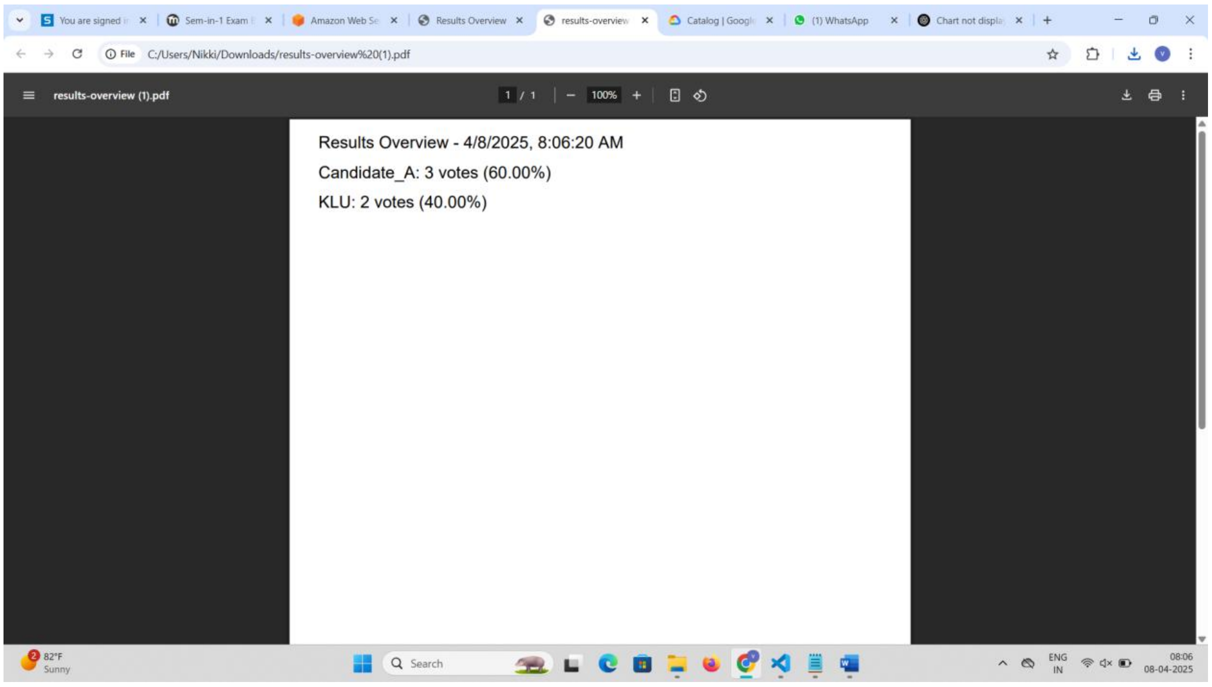
Voting Event 2 starts in: 4d 23h 59m 56s

Voting Event 3: Time Ended - Unable to Vote


localhost:8000/overview.html

Results Overview

Total result:

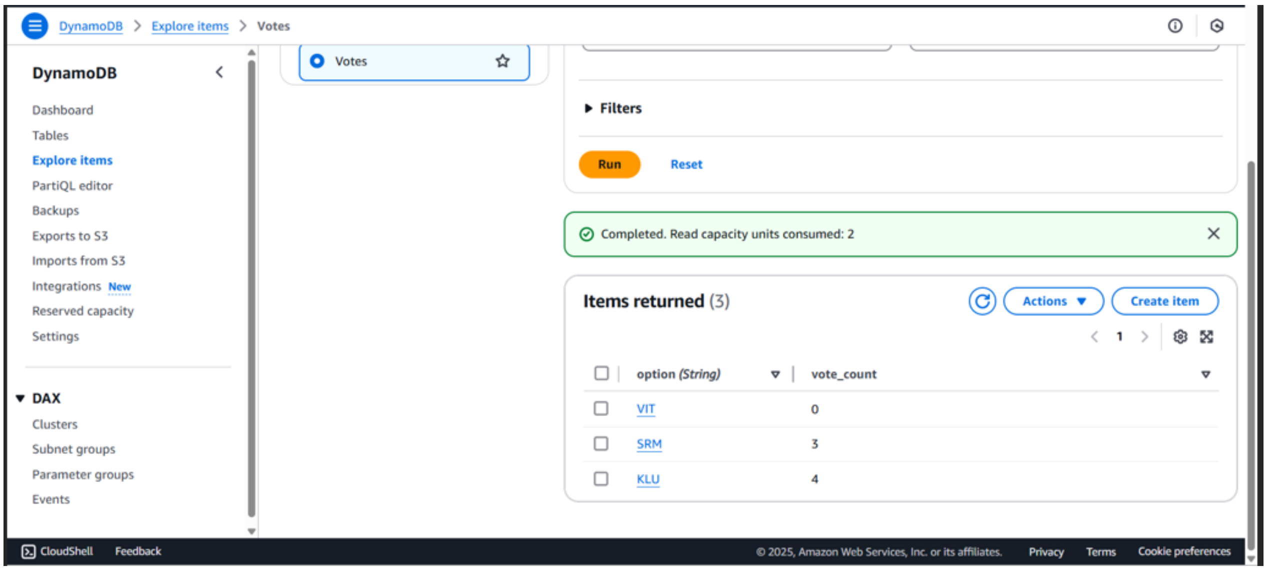

Results Overview - 4/8/2025, 8:06:20 AM

Candidate_A: 3 votes (60.00%)

KLU: 2 votes (40.00%)

Updating in dynamo db :

After voting to KLU


Completed. Read capacity units consumed: 2

Items returned (3)

option (String)	vote_count
VIT	0
SRM	3
KLU	4

Conclusion

The Voting Application project provided a rich, hands-on opportunity to apply and sharpen my cloud computing skills, particularly in the area of serverless architecture and AWS services. This project not only fulfilled the functional objective of creating a real-time voting system, but also gave me in-depth experience in designing, deploying, and managing scalable cloud-native applications.

One of the key skills acquired was user authentication and access control through Amazon Cognito, where I learned to configure user pools, app clients, and integrate secure sign-up/sign-in workflows. Additionally, working with AWS Lambda functions taught me to handle backend logic without provisioning servers, manage environment variables, and ensure fault-tolerant code execution. By integrating API Gateway, I built and exposed RESTful APIs, defined methods (GET/POST), managed endpoint security, and enabled CORS for frontend communication.

A critical takeaway was learning to configure and grant IAM permissions, which is vital for secure interactions between AWS services like Lambda and DynamoDB. I created and tested Lambda functions to store and fetch vote data from the DynamoDB table Votes, strengthening my understanding of event-driven data flow and NoSQL schema design.