

MATHEMATICAL CALCULATION

Project Report submitted to Bengaluru North University, Kolar in
partial fulfillment for the award of the degree of

BACHELOUR OF COMPUTER APPLICATION

Submitted by

CHANDAN A N

U19GZ21S0091

**Under the guidance of
MADHU N**

Assistant Professor
Department of Computer Science
Government First Grade College
KR Puram, Bengaluru– 560036.



**DEPARTMENT OF COMPUTER SCIENCE
GOVERNMENT FIRST GRADE COLLEGE
K R Puram, Bengaluru-560036**

2023-2024



**DEPARTMENT OF COMPUTER SCIENCE
GOVERNMENT FIRST GRADE COLLEGE
K R Puram, Bengaluru-560036**

Certificate

This is to certify that the Project titled “**MATHEMATICAL CALCULATION**” Is an original work of Mr. **CHANDAN A N** [U19GZ21S0091] and is being submitted in partial fulfillment for the award of the Bachelor of computer application of Bengaluru North University, Government First Grade College, K R Puram, Bengaluru- 560036. This report or part of this report has not been submitted earlier either to this University or to any other University / Institution for the fulfillment of the requirement of a course of study or published / presented for any other purpose.

Signature of the Guide

Signature of HOD

Submitted for the Viva-Voce held on..... at Government First Grade College, KR Puram, Bengaluru.

External Examiners

1.

2.

DECLARATION

We **Mr. CHANDAN A N [U19GZ21S0091]** hereby declare that the project entitled **MATHEMATICAL CALCULATION** submitted to the Bengaluru North University, Kolar in partial fulfillment for the award of the Degree of **BACHELOUR OF COMPUTER APPLICATION** and that the project has not previously formed the basis for the award of any other degree, Diploma, Associateship, Fellowship or other title.

Signature of the Candidate

Place: KR Puram, Bengaluru

Date:

ACKNOWLEDGEMENT

We are thankful to our Principal, **Prof. Prathibha Parswanath**, Government First Grade, KR Puram, Bengaluru for the permission to perform the project on stage.

We express our sincere thanks to **Mrs. Nalini R**, HOD, Department of Computer Science, Government First Grade, KR Puram, Bengaluru for his valuable guidance and timely suggestion at every stage of this project.

We express our special thanks to **Mr. MADHU N**, Guest Lecturer, Department of Computer Science, Government First Grade, KR Puram, Bengaluru for his/her valuable guidance and timely suggestion at every stage of this project.

We express our sincere thanks to all Faculties, Department of Computer Science, Government First Grade, KR Puram, Bengaluru, for valuable guidance and timely suggestion at every stage of this project.

This acknowledgment would be incomplete without expressing gratitude to our parents, friends who motivated and inspired us throughout this project.

CONTENTS

<u>CHAPTER NO</u>	<u>Particulars</u>	<u>Page. No</u>
1.	ABSTRACT	1
2.	INTRODUCTION	2-8
3.	SOFTWARE AND HARDWARE REQUIRMENTS	9-11
4.	SYSTEM DESIGN	12-15
5.	DATABASE DESIGN	16-20
6.	CONNECTIVITY	21-28
7.	SOURCE CODE	29-61
8.	TESTING	62-69
9.	FORMS	70-75
10.	CONCLUSION	76-78
11.	FUTURE ENHANCEMENT	79-82
12.	BIBLIOGRAPHY	83-85

CHAPTER 1

ABSTRACT

This project report explores the implementation of mathematical calculations using Amazon Web Services (AWS), specifically focusing on calculating the power of a number, trigonometric functions, and logarithmic functions. The integration of these mathematical operations with AWS services demonstrates the potential of cloud computing for performing complex computations in a scalable and efficient manner.

The project leverages AWS Lambda and AWS API Gateway to create serverless functions that handle the mathematical calculations. AWS Lambda allows the execution of code in response to specific events, while AWS API Gateway facilitates the exposure of these functions as RESTful APIs, enabling easy access and integration.

The report provides an in-depth overview of AWS services, detailed implementation steps, and Python code examples for the mathematical calculations. Additionally, it includes testing procedures and sample results to validate the functionality of the implemented solutions. By utilizing AWS's powerful cloud infrastructure, this project showcases the benefits of serverless computing for performing essential mathematical operations, offering a scalable, cost-effective, and efficient solution for various computational needs.

CHAPTER 2

INTRODUCTION



Amazon Web Services offers a broad set of global cloud-based products including compute, storage, databases, analytics, networking, mobile, developer tools, management tools, IoT, security, and enterprise applications: on-demand, available in seconds, with pay-as-you-go pricing. From data warehousing to deployment tools, directories to content delivery, over 200 AWS services are available.

In 2006, Amazon Web Services (AWS) began offering IT infrastructure services to businesses as web services—now commonly known as cloud computing. One of the key benefits of cloud computing is the opportunity to replace upfront capital infrastructure expenses with low variable costs that scale with your business. With the cloud, businesses no longer need to plan for and procure servers and other IT infrastructure weeks or months in advance. Instead, they can instantly spin up hundreds or thousands of servers in minutes and deliver results faster. Today, AWS provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that powers hundreds of thousands of businesses in 190 countries around the world.

The advent of cloud computing has revolutionized the way computational tasks are performed, offering unparalleled scalability, flexibility, and cost efficiency. Among the leading providers of cloud services is Amazon Web Services (AWS), a platform that provides a vast array of services to cater to diverse computing needs. This project explores the application of AWS for performing mathematical calculations, specifically focusing on the power of a number, trigonometric functions, and logarithmic functions.

Mathematical calculations are fundamental to numerous scientific, engineering, and business applications. Performing these calculations efficiently and accurately is crucial for various domains such as data analysis, simulations, and financial modelling. Traditionally, these computations were handled by on-premises hardware or local software, which often posed limitations in terms of scalability, maintenance, and cost. The integration of cloud computing platforms like AWS introduces a new paradigm where these limitations are significantly reduced, providing a robust and scalable environment for complex calculations.

This project utilizes AWS Lambda and AWS API Gateway to implement serverless functions that perform the required mathematical calculations. AWS Lambda is a compute service that allows you to run code without provisioning or managing servers. It automatically scales the execution in response to incoming requests, ensuring high availability and efficient resource utilization. AWS API Gateway, on the other hand, is a fully managed service that enables developers to create, publish, maintain, monitor, and secure APIs at any scale. By combining these services, we can create a

seamless and efficient system for performing mathematical calculations via RESTful APIs.

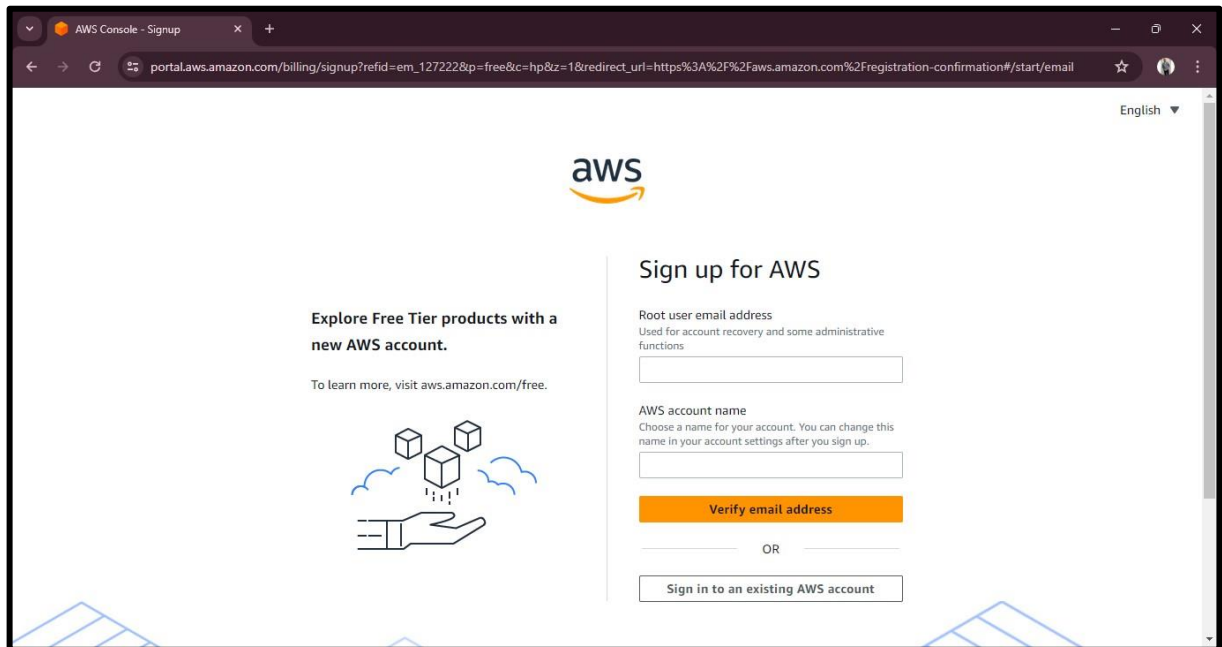
The primary objectives of this project are:

1. To demonstrate the implementation of basic mathematical calculations using AWS Lambda and AWS API Gateway.
2. To provide a detailed guide on setting up and configuring these AWS services for computational tasks.
3. To offer Python code examples for performing power calculations, trigonometric functions, and logarithmic functions.
4. To validate the implemented solutions through testing and presenting sample results.

By achieving these objectives, this project aims to showcase the potential of cloud computing for handling mathematical computations, highlighting the benefits of scalability, efficiency, and cost-effectiveness provided by AWS. This introduction sets the stage for the detailed exploration of the project's components, implementation steps, and the resulting benefits of using AWS for mathematical calculations.

To Start this project, you need to start from creating AWS Management Console Account.

CREATION OF AWS FREE TIER ACCOUNT



Explore Free Tier products with a new AWS account.

To learn more, visit aws.amazon.com/free.

Sign up for AWS

Root user email address
Used for account recovery and some administrative functions

AWS account name
Choose a name for your account. You can change this name in your account settings after you sign up.

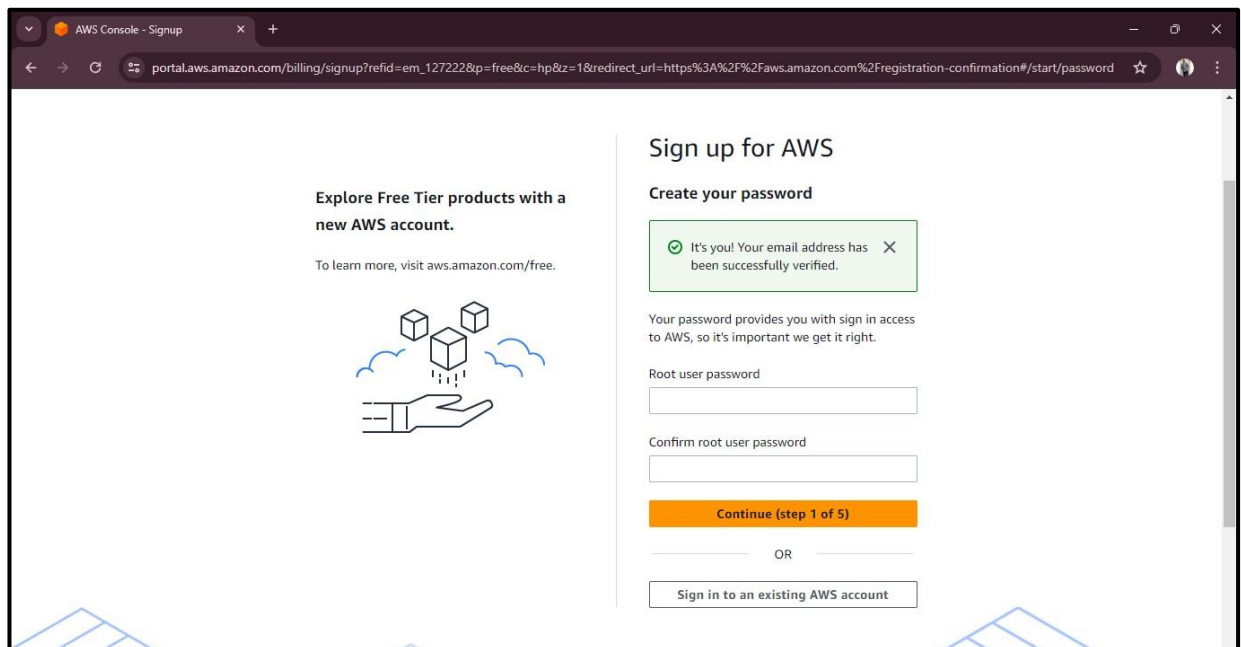
Verify email address

OR

[Sign in to an existing AWS account](#)

Step1: Create account in AWS Console.

Step2: Use Email id for Sign up(Verification code will be sent to the registered Email id)



Explore Free Tier products with a new AWS account.

To learn more, visit aws.amazon.com/free.

Sign up for AWS

Create your password

It's you! Your email address has been successfully verified.

Your password provides you with sign in access to AWS, so it's important we get it right.

Root user password

Confirm root user password

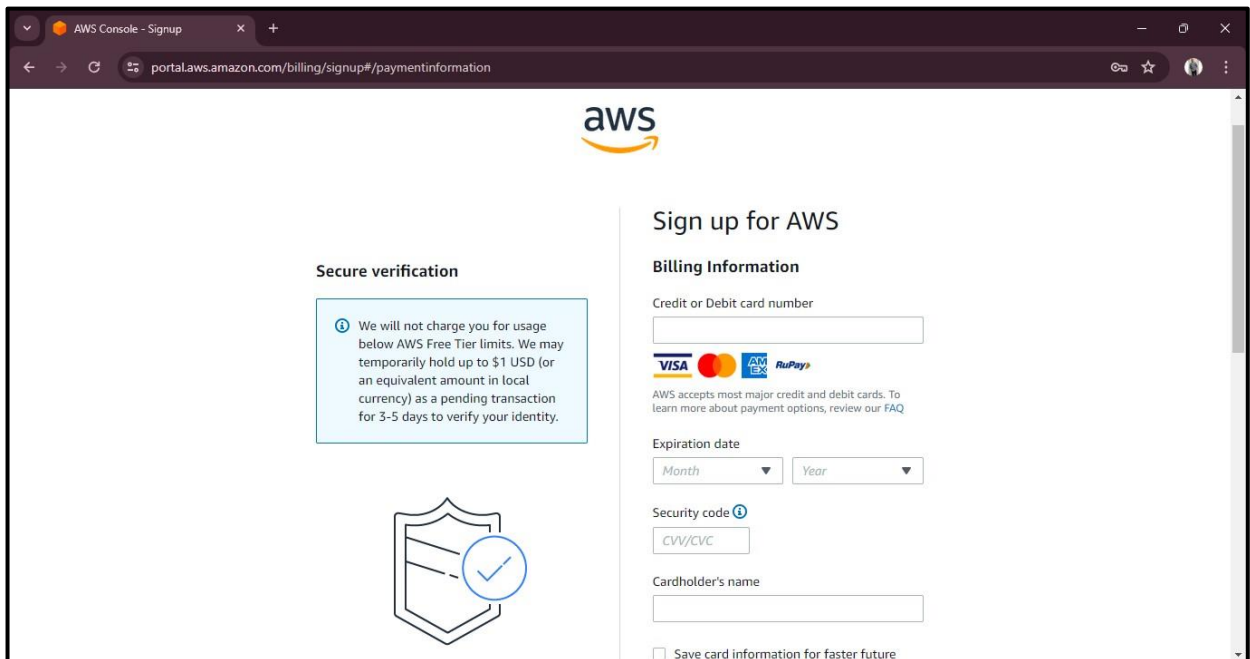
Continue (step 1 of 5)

OR

[Sign in to an existing AWS account](#)

Step3: Create Root user password (That should contain 1 Upper Case Alphabet,1 Lower Case Alphabet, Digits and Special Characters).

Step4: Enter VISA Debit CARD, RuPay Credit card, Master Card for Payment Details (AWS Charges Pay-as-you-go pricing model So the amount will be debited after the 12Months (Free Tier)).



The screenshot shows the AWS Sign up for AWS Billing Information page. The page is titled "Sign up for AWS" and "Billing Information". It includes a "Secure verification" section with a note: "We will not charge you for usage below AWS Free Tier limits. We may temporarily hold up to \$1 USD (or an equivalent amount in local currency) as a pending transaction for 3-5 days to verify your identity." Below this is a shield icon with a checkmark. The "Billing Information" section includes fields for "Credit or Debit card number", "Expiration date" (Month and Year dropdowns), "Security code" (CVV/CVC), and "Cardholder's name". There is also a checkbox for "Save card information for faster future".

Step5: Select the Support Plan.



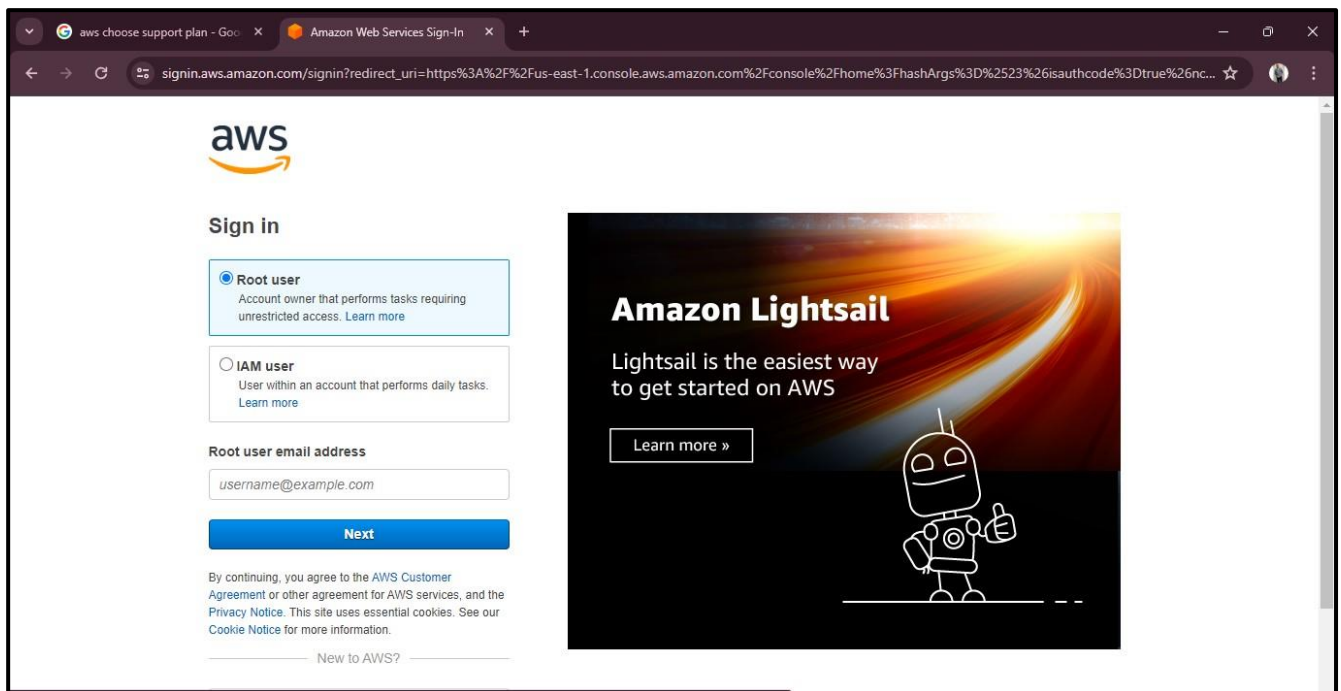
The screenshot shows the AWS Sign up for AWS Select a support plan page. The page is titled "Sign up for AWS" and "Select a support plan". It includes a note: "Choose a support plan for your business or personal account. Compare plans and pricing examples. You can change your plan anytime in the AWS Management Console." Below this are three support plan options:

- Basic support - Free**
 - Recommended for new users just getting started with AWS
 - 24x7 self-service access to AWS resources
 - For account and billing issues only
 - Access to Personal Health Dashboard & Trusted Advisor
- Developer support - From \$29/month**
 - Recommended for developers experimenting with AWS
 - Email access to AWS Support during business hours
 - 12 (business)-hour response times
- Business support - From \$100/month**
 - Recommended for running production workloads on AWS
 - 24x7 tech support via email, phone, and chat
 - 1-hour response times
 - Full set of Trusted Advisor best-practice recommendations

Below these options is a section for "Need Enterprise level support?" with a note: "From \$15,000 a month you will receive 15-minute response times and concierge-style experience with an assigned Technical Account Manager. Learn more". At the bottom is a "Complete sign up" button.

Step6: Ready to use the AWS Management Console Account using Your Credentials.

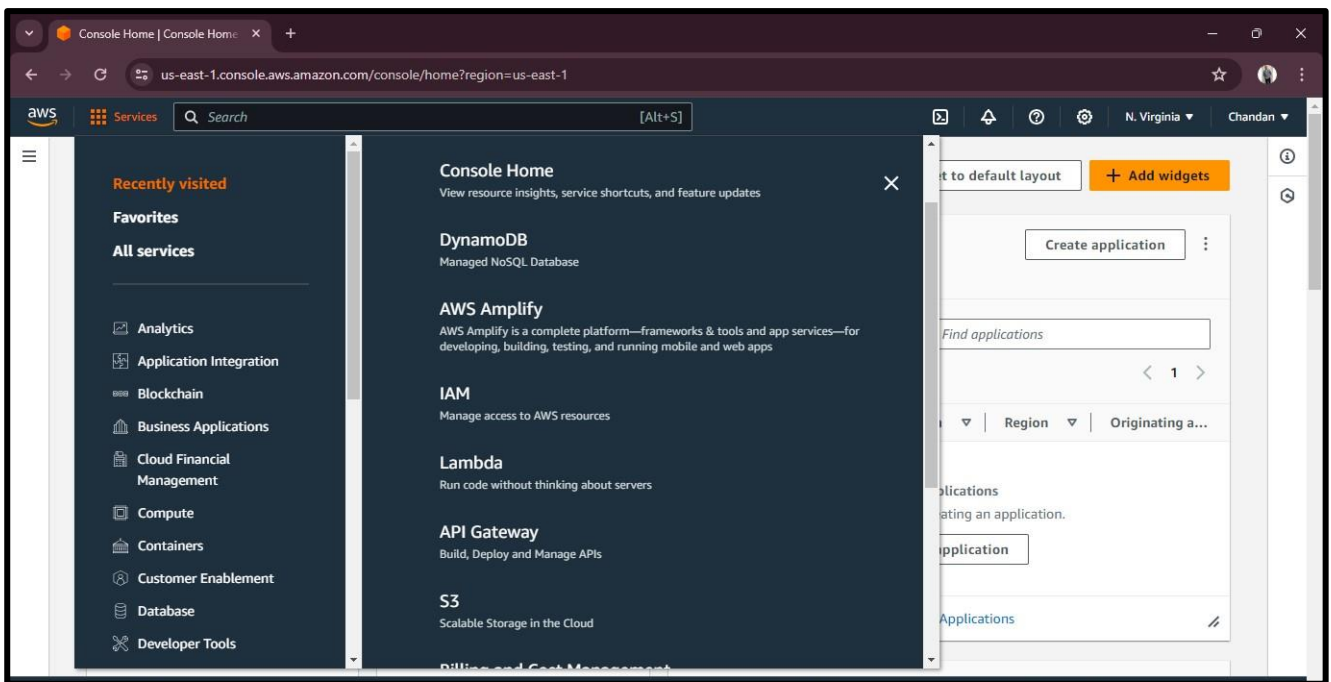
Step7: Login as Root User.



EXPLORING AWS MANAGEMENT CONSOLE

- Using Login Credentials I logged into AWS Management Console account.
- Explored AWS Services Like AWS Lambda, AWS Amplify, AWS IAM etc
- These Services are Categorized based on the service they provide like Compute, Database, Storage, Containers, Developer Tools, Front-end & Mobile, App Development etc.

- By Exploring these service we can do project, host our own static and Dynamic Websites, Creating Cryptographic Models etc.



CHAPTER 3

SOFTWARE AND HARDWARE REQUIREMENTS

SOFTWARE REQUIREMENTS

1. AWS Account
 - An active AWS account with appropriate permissions to create and manage AWS Lambda and AWS API Gateway resources.
2. AWS CLI (Command Line Interface)
 - AWS CLI installed and configured on your local machine for managing AWS resources via command line.
3. Python
 - Python 3.x installed on your local machine for developing and testing the Lambda functions.
4. Integrated Development Environment (IDE)
 - An IDE like Visual Studio Code, PyCharm, or any text editor for writing and editing Python code.
5. Postman or Curl
 - Tools like Postman or Curl for testing the APIs exposed via AWS API Gateway.
6. Boto3
 - The AWS SDK for Python (Boto3) if additional AWS interactions are needed beyond Lambda and API Gateway.
7. JSON Library
 - A library for handling JSON data in Python, which is typically included in the Python standard library.

Hardware Requirements

1. Local Development Machine

- A computer with a modern operating system (Windows, macOS, or Linux) capable of running Python and AWS CLI.
- Minimum 4 GB RAM, but 8 GB or more is recommended for smoother development and testing.

2. Internet Connection

- A stable internet connection is essential for accessing AWS services and deploying Lambda functions and API Gateway configurations.

3. Optional: External Storage or Backup

- External storage or cloud-based backup solutions for saving your project files and code securely.

Detailed Setup

Setting Up the AWS Environment

1. Create an AWS Account

- Sign up for an AWS account if you don't have one already. Visit the AWS Signup page to create an account.

2. Install AWS CLI

- Download and install the AWS CLI from the official AWS CLI installation guide.
- Configure AWS CLI with your credentials using AWS configure.

3. Set Up IAM Roles and Permissions

- Create an IAM role with the necessary permissions for Lambda and API Gateway.

Development Environment Setup

1. Install Python

- Download and install Python 3.x from the official Python website.

2. Set Up IDE

- Install an IDE like Visual Studio Code or PyCharm for writing and testing your Python code.
- Configure your IDE to work with Python and AWS.

CHAPTER 4

SYSTEM DESIGN

OVERVIEW

The Mathematical Calculation Project is designed to offer users the ability to perform various mathematical operations through a web interface. The system includes user authentication (registration and login) and interfaces with serverless APIs for performing calculations.

Components

1. **Frontend (Client-Side)**
2. **Backend (Server-Side)**
3. **APIs for Mathematical Calculations**

1. Frontend (Client-Side)

Technologies

- HTML, CSS, JavaScript

Components

- **Login and Registration Forms**: Interfaces for users to sign up and log in.
- **Calculation Interface**: Sections to perform power functions, trigonometric functions, and logarithm functions.
- **JavaScript Functions**: Handle user actions, form submissions, and API calls.

Data Flow

1. **User Registration**:
 - User inputs name, email, and password.
 - Data is temporarily stored in an in-memory array for the session.

2. User Login:

- User inputs email and password.
- Credentials are validated against the in-memory data.
- On success, user is directed to the calculation interface.

3. Mathematical Calculations:

- User selects a calculation type and inputs required data.
- JavaScript sends data to the backend API and displays the result.

2. Backend (Server-Side)

Technologies

- AWS API Gateway
- AWS Lambda
- Dynamo DB

Components

- API Gateway: Routes requests to appropriate Lambda functions.
- Lambda Functions: Handle computation for power, trigonometric, and logarithm functions.
- Dynamo DB: Stores data related to power functions, trigonometric functions, and logarithm functions

Data Flow

- API Requests:
 - The frontend sends a POST request to API endpoints with user input data.

- **Lambda Processing:**

- Each Lambda function processes the input data to perform the respective calculation.
- Results are sent back to the frontend in the API response.

- **DynamoDB Tables:**

- **Power Functions Table:** Stores data related to power function calculations.
- **Trigonometric Functions Table:** Stores data related to trigonometric function calculations.
- **Logarithm Functions Table:** Stores data related to logarithm function calculations.

3. APIs for Mathematical Calculations

Endpoints

1. Power Functions API

- **Endpoint:** /power
- **Functionality:** Calculates the result of a base raised to an exponent.

2. Trigonometric Functions API

- **Endpoint:** /trig
- **Functionality:** Computes sine, cosine, tangent, and their inverse functions.

3. Logarithm Functions API

- **Endpoint:** /log
- **Functionality:** Computes the logarithm of a number with a specified base.

Data Flow

1. Power API:

- Input: { "base": number, "exponent": number }
- Output: { "result": number }

2. Trig API:

- Input: { "Sin": number, "Cos": number, "Tan": number, "aSin": number, "aCos": number, "aTan": number }
- Output: { "Sin": result, "Cos": result, "Tan": result, "aSin": result, "aCos": result, "aTan": result }

3. Log API:

- Input: { "base": number, "number": number }
- Output: { "result": number }

SECURITY

- **HTTPS:** Ensure all communication between frontend and backend is over HTTPS.
- **Input Validation:** Validate all inputs on the client-side before sending to the backend to prevent injection attacks.

SCALABILITY AND PERFORMANCE

- **Serverless Architecture:** Leverages AWS Lambda for automatic scaling based on the number of requests.
- **API Gateway:** Efficiently manages API traffic with features like rate limiting and throttling.

CHAPTER 5

DATABASE DESIGN

Database Design for Mathematical Calculation Project Using AWS DynamoDB

1. User Table

This table will store user information, including name, email, and password.

Table Name: Users

Primary Key:

- UserID (String, unique identifier for each user, can be generated using UUID)

Attributes:

- UserID (String) - Unique identifier for the user.
- Name (String) - User's name.
- Email (String) - User's email (unique).
- Password (String) - User's password (hashed for security).
- CreatedAt (String) - Timestamp of when the user was created.

Example Table Entry:

UserID	Name	Email	Password	CreatedAt
user1	John Doe	john@example.com	hashedpassword	2024-08-01T12:00:00Z

2. Functions Table

This table stores different mathematical functions.

Table Name: Functions

Primary Key:

- FunctionID (String, unique identifier for each function)

Attributes:

- FunctionID (String) - Unique identifier for the function.
- FunctionType (String) - Type of the function (e.g., Power, Trigonometric, Logarithm).
- FunctionName (String) - Name of the function (e.g., sin, cos, log, exp).
- FunctionExpression (String) - Mathematical expression or formula.
- CreatedAt (String) - Timestamp of when the function was added.
- UpdatedAt (String) - Timestamp of the last update to the function.

Example Table Entry:

Function ID	FunctionType	FunctionName	FunctionExpression	CreatedAt	UpdatedAt
func1	Power	exp	exp(x)	2024-08-01T12:00:00Z	2024-08-01T12:00:00Z
func2	Trigonometric	sin	sin(x)	2024-08-01T12:05:00Z	2024-08-01T12:05:00Z
func3	Logarithm	log	log(x)	2024-08-01T12:10:00Z	2024-08-01T12:10:00Z

3. Calculations Table

This table stores calculation history.

Table Name: Calculations

Primary Key:

- CalculationID (String, unique identifier for each calculation)

Attributes:

- CalculationID (String) - Unique identifier for the calculation.
- UserID (String) - Identifier for the user who performed the calculation.
- FunctionID (String) - Identifier for the function used.
- InputValues (Map) - Input parameters for the calculation.

- Result (String) - Result of the calculation.
- CalculatedAt (String) - Timestamp of when the calculation was performed.

Example Table Entry:

CalculationID	UserID	FunctionID	InputValues	Result	CalculatedAt
Power	Power	Powerfunction	{"base": "2", "exponent": "3"}	8.0	2024-08-01T12:15:00Z
Trigonometric	Trigonometry	Trigfunction	{"x": "0.5"}	0.479426	2024-08-01T12:20:00Z
Logarithm	Logarithm	Logfunction	{"base": "10", "number": "100"}	2.0	2024-08-01T12:25:00Z

Example AWS Infrastructure

1. Amazon DynamoDB:

- Tables: Users, Functions, Calculations
- Global Secondary Indexes (GSIs): If needed for specific query patterns, such as querying calculations by UserID.

2. AWS Lambda:

- For handling user registration, login, and mathematical calculations by interacting with DynamoDB.

3. Amazon API Gateway:

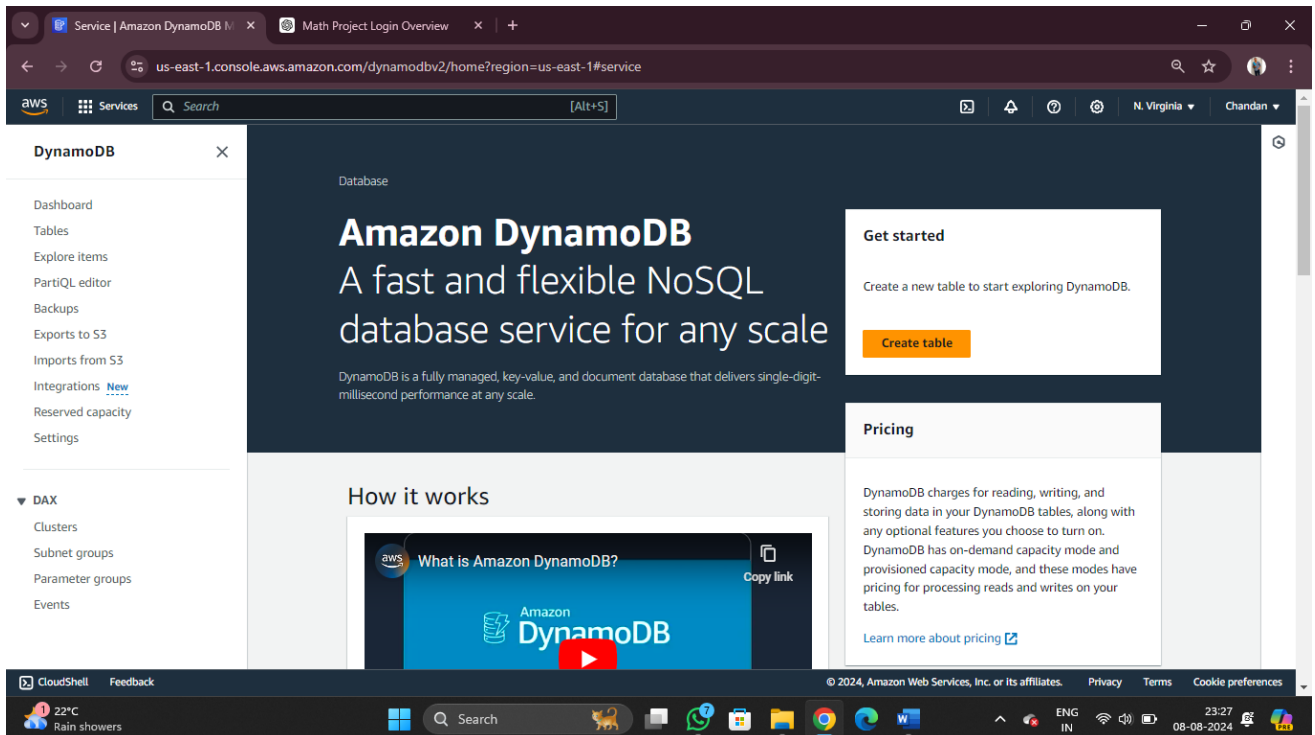
- For exposing APIs to interact with your system, connecting frontend with backend Lambda functions.

4. AWS IAM:

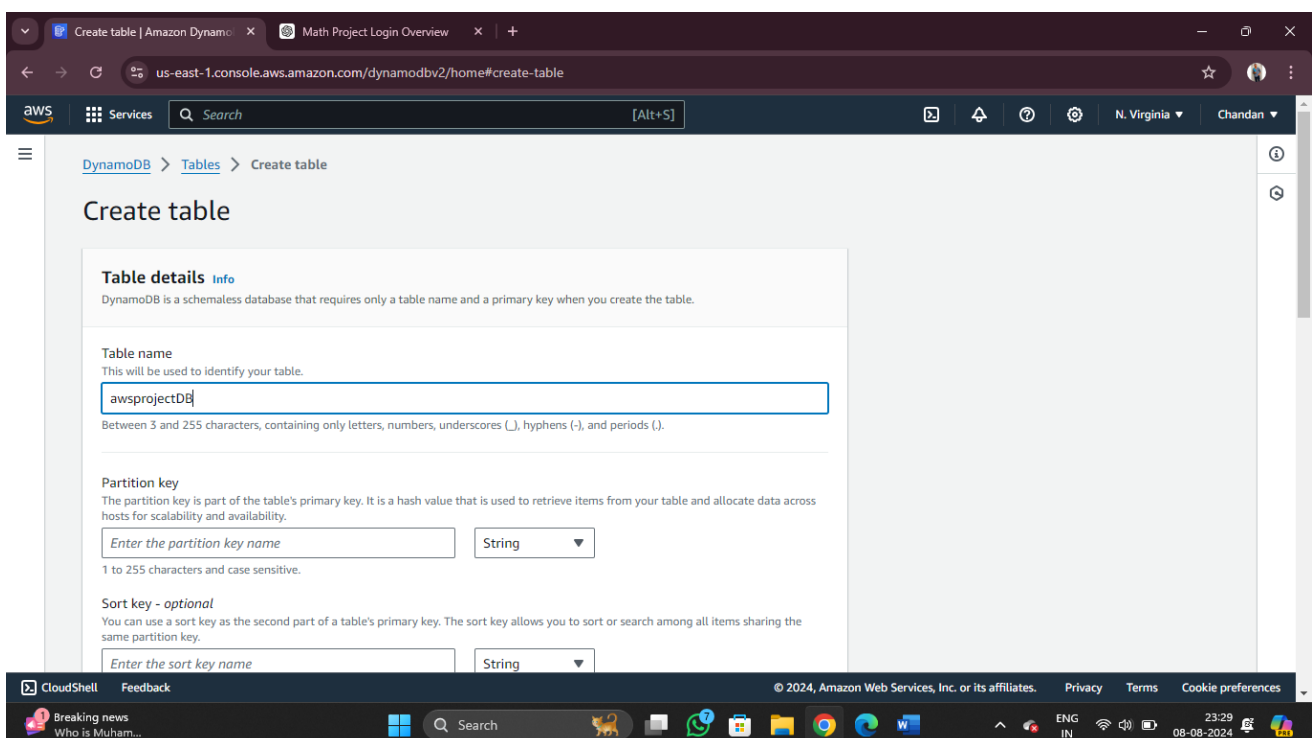
- For defining roles and policies to control access to DynamoDB tables and other resources.

DATABASE CREATION IN AWS MANAGEMENT CONSOLE

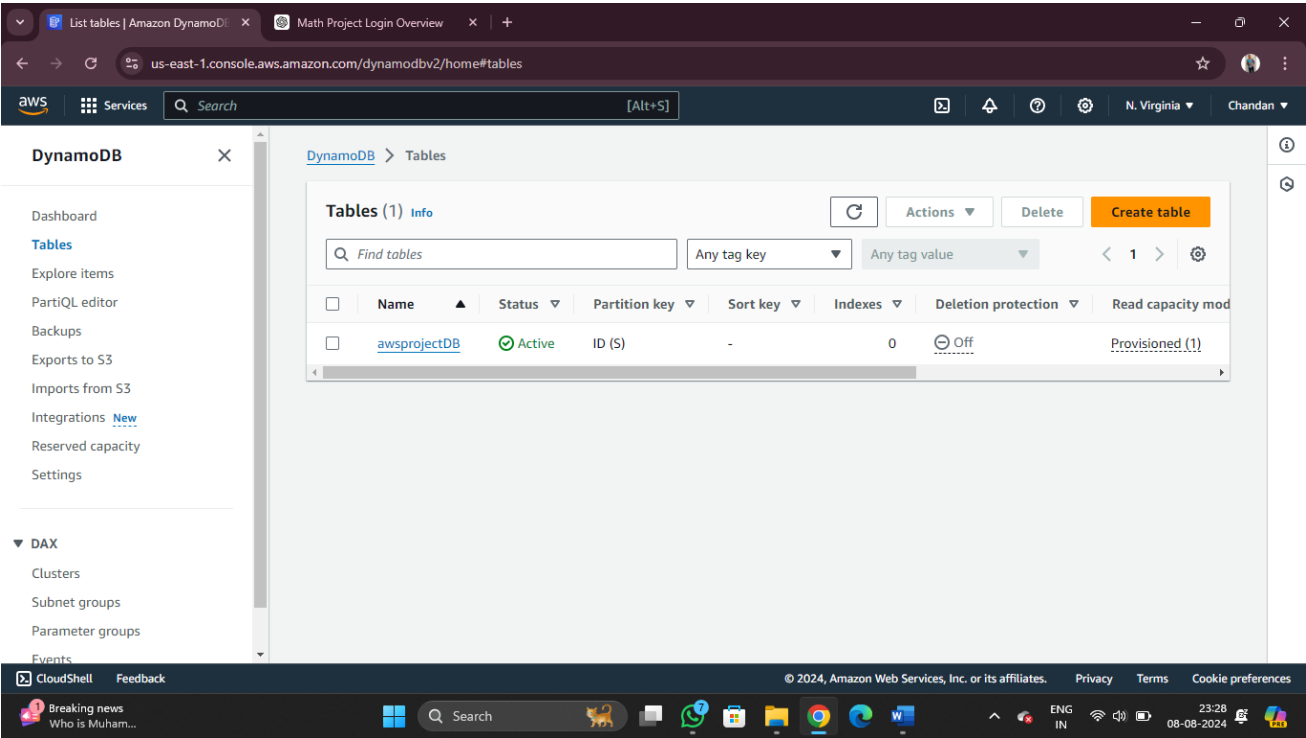
Step1: Selecting Dynamo DB in AWS Services.



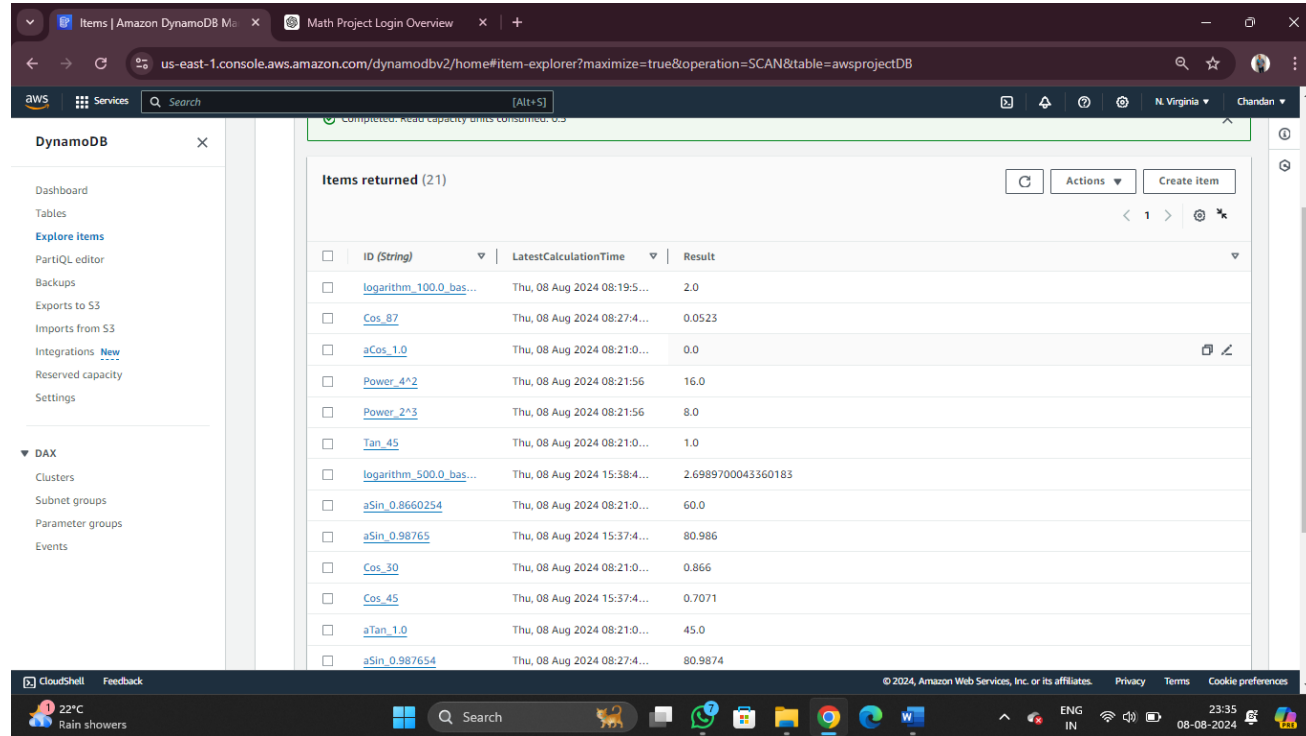
Step2: Creating a New Table.



Step 3: Exploring the Table:



Step 4: Exploring Table Items From the Created Database:



CHAPTER 6

CONNECTIVITY

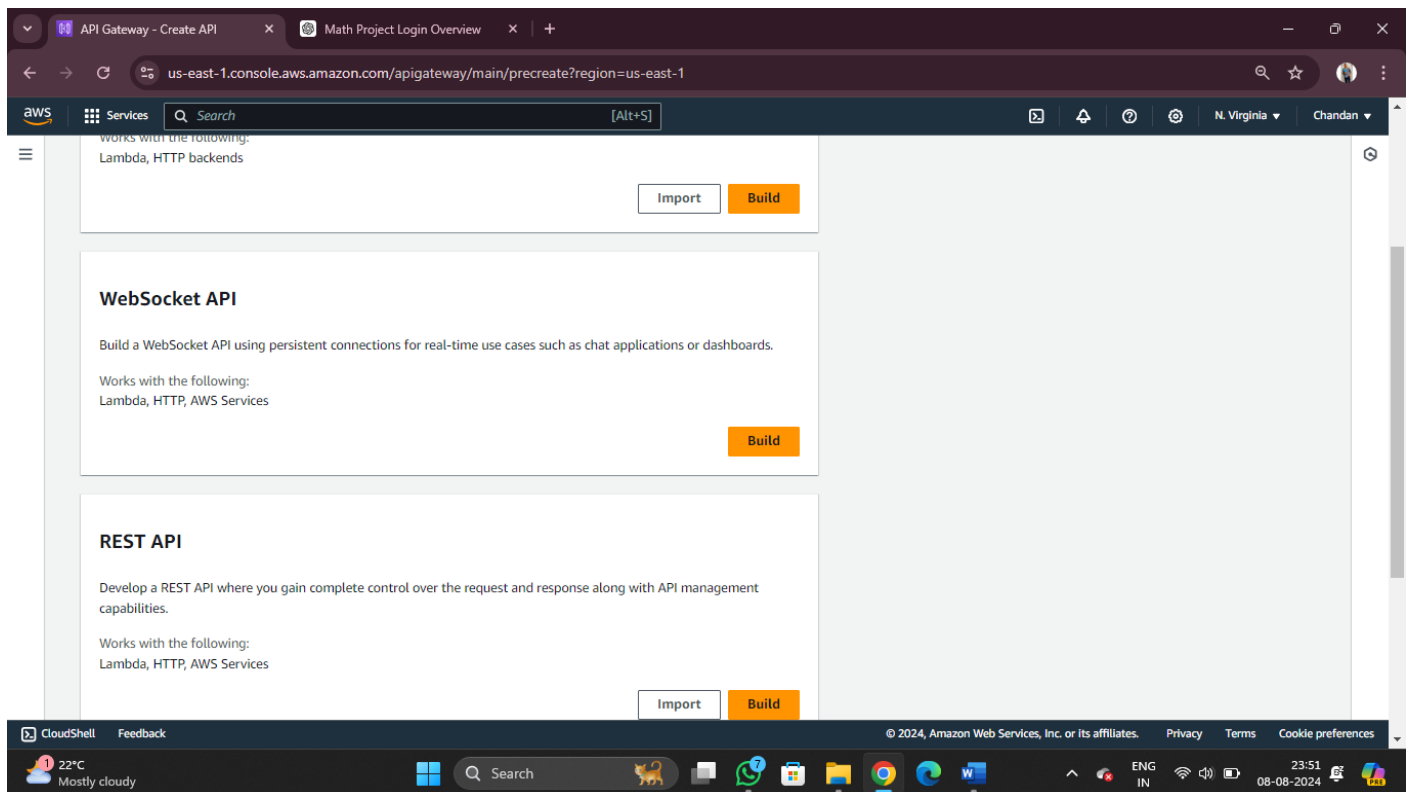
Connectivity

Frontend and API Connectivity:

- The frontend interface interacts with the backend services through RESTful API calls. The API endpoints for mathematical calculations are hosted on AWS and are invoked using the fetch API in JavaScript.
- For example, the Power Functions form triggers a POST request to the /power endpoint of the API Gateway. The request payload includes the base and exponent values entered by the user.

To connect Lambda(Backend) to Frontend we use API GATEWAY these are the following steps:

Step 1:Select type of API as REST and Click on Build.



Step 2:Give a name for the API and Click on Create

The screenshot shows the AWS Management Console for creating a new REST API. The browser address bar shows the URL: `us-east-1.console.aws.amazon.com/apigateway/main/create-rest?experience=rest®ion=us-east-1`. The page title is "Create REST API".

API details

- ☒ **New API**
Create a new REST API.
- ☐ Clone existing API
Create a copy of an API in this AWS account.
- ☐ Import API
Import an API from an OpenAPI definition.
- ☐ Example API
Learn about API Gateway with an example API.

API name:

Description - optional:

API endpoint type:
Regional APIs are deployed in the current AWS Region. Edge-optimized APIs route requests to the nearest CloudFront Point of Presence. Private APIs are only accessible from VPCs.

Buttons: Cancel, Create API

Step 3:Create Resource by using specific name:

The screenshot shows the AWS Management Console for creating a new resource. The browser address bar shows the URL: `us-east-1.console.aws.amazon.com/apigateway/main/apis/5ov3hm0id2/resources/nc1pcn/create-resource?api=5ov3hm0id2®ion=us-east-1`. The page title is "Create resource".

Resource details

- ☐ Proxy resource [Info](#)
Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example (proxy+).

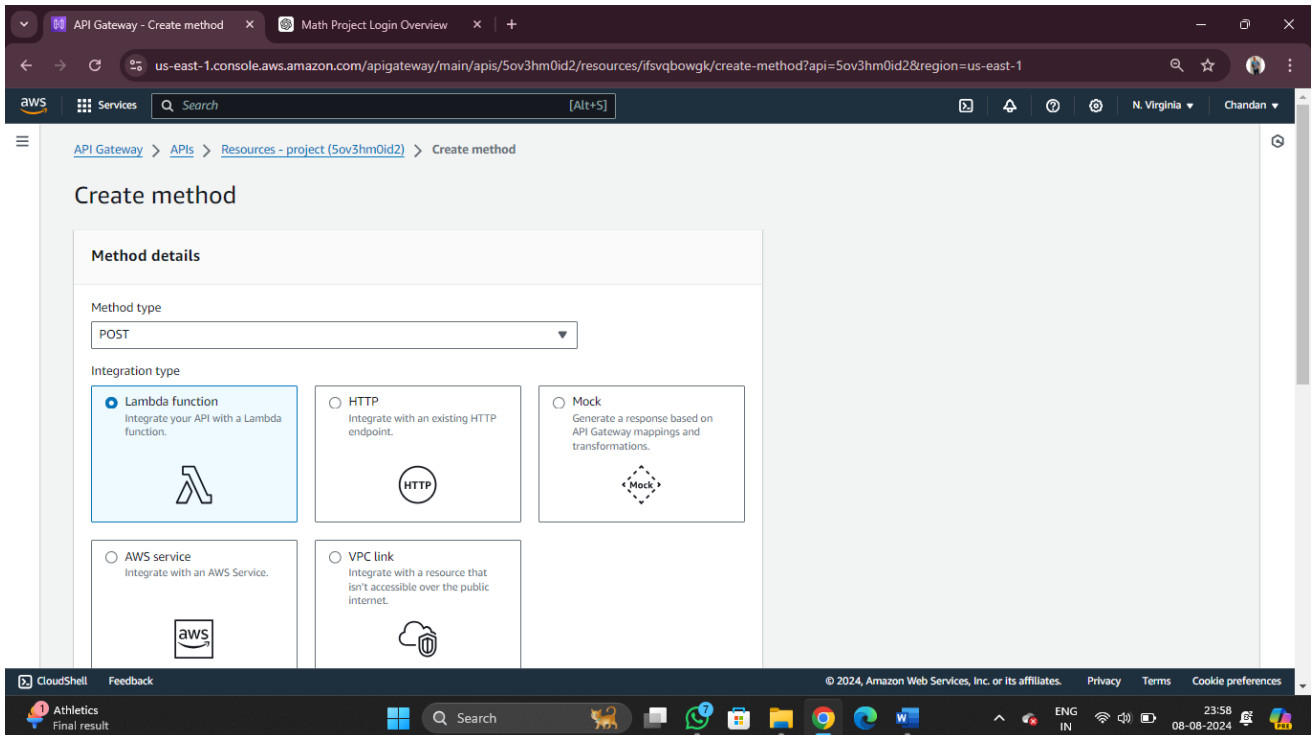
Resource path:

Resource name:

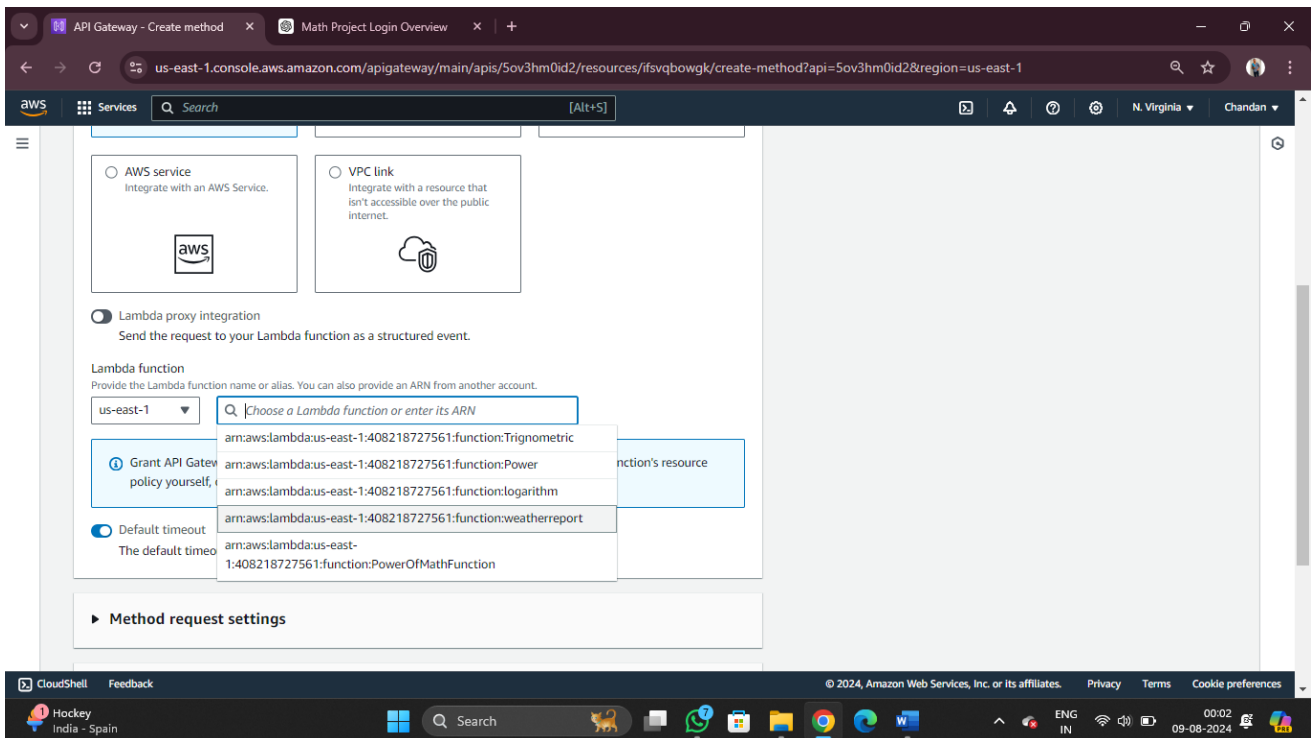
- ☐ CORS (Cross Origin Resource Sharing) [Info](#)
Create an OPTIONS method that allows all origins, all methods, and several common headers.

Buttons: Cancel, Create resource

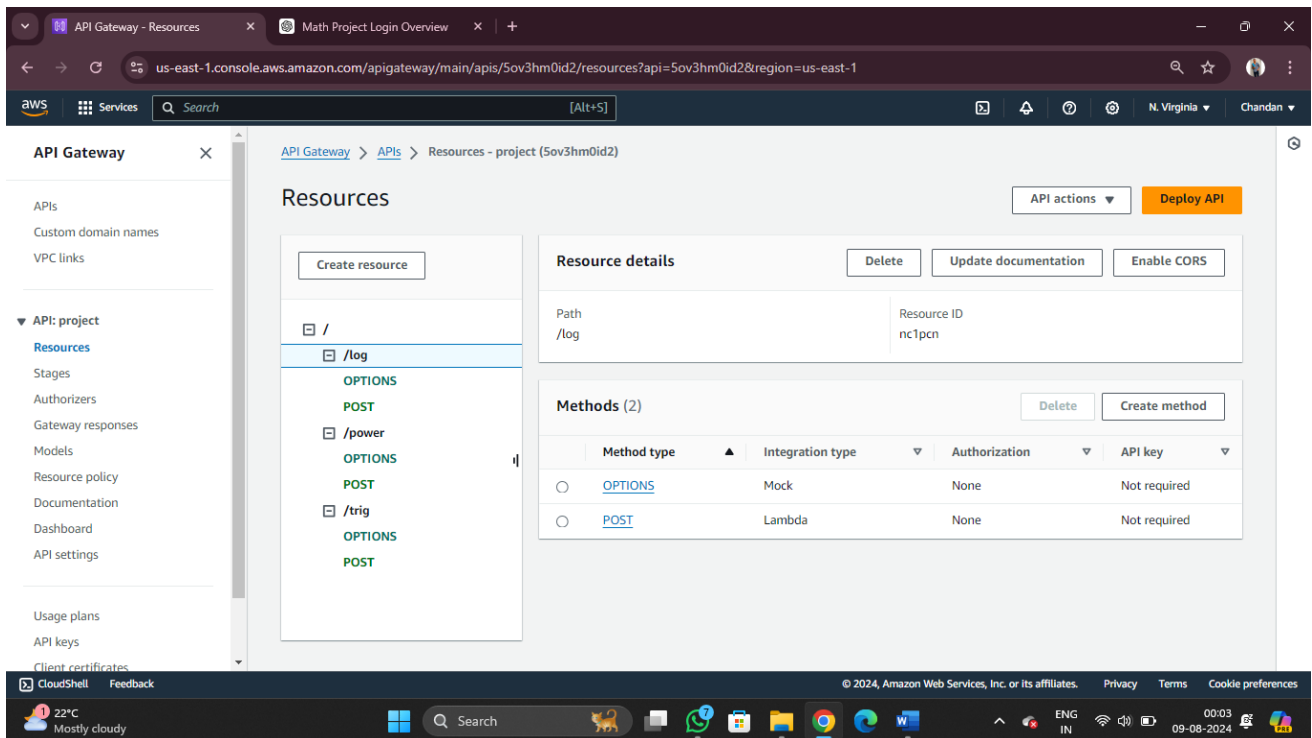
Step 4: Once Creation of resource is done, Create Methods using post method and select LAMBDA has Integration type for the specific resource using “/” has Extension.



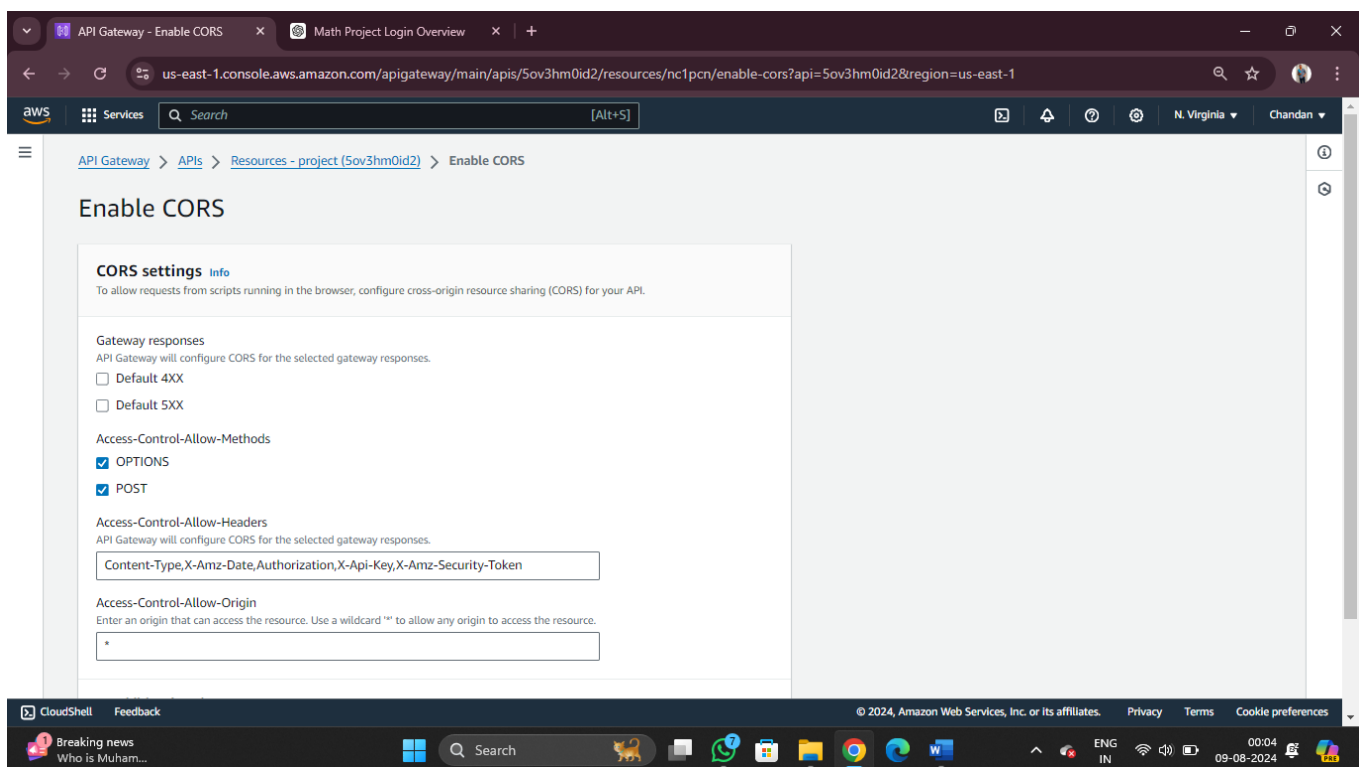
Step 5: Select the Lambda function to create a methods.



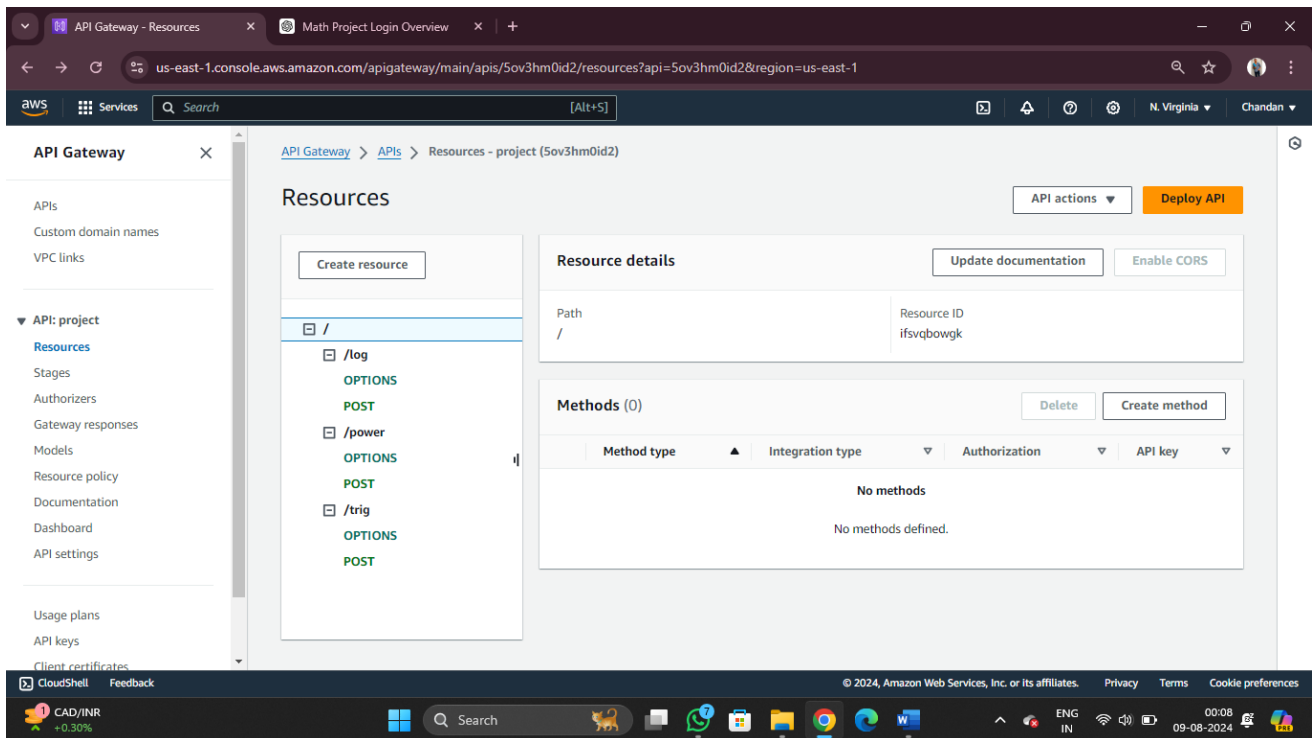
Step 6: Once Creation of Method is Done we need to enable CORS for the Created resource



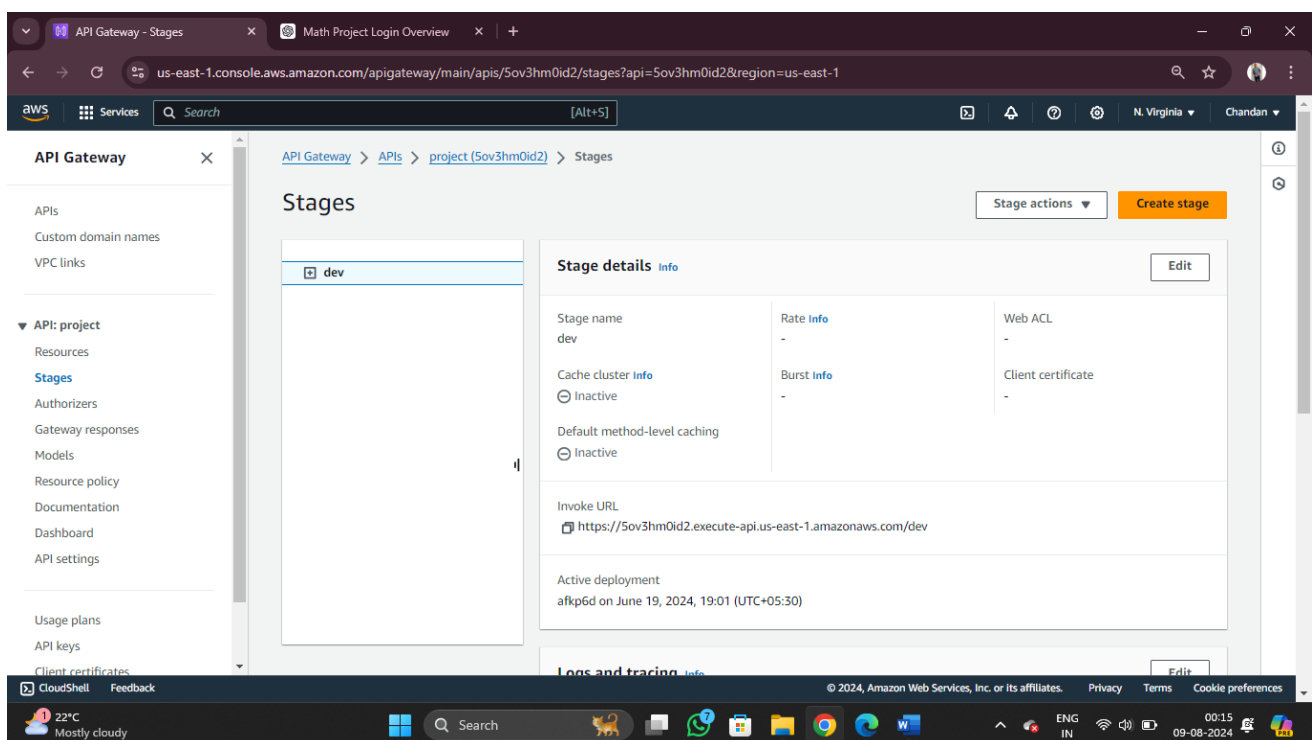
Step 7: Select both Access Control Allow Method i.e- OPTIONS and POST



Step 8: Once Creation of methods, resource and enabling of CORS is done
Deploy Your API.



Step 9: Select Stages in API GATEWAY Column and Copy the INVOKE URL To trigger Backend to Frontend.

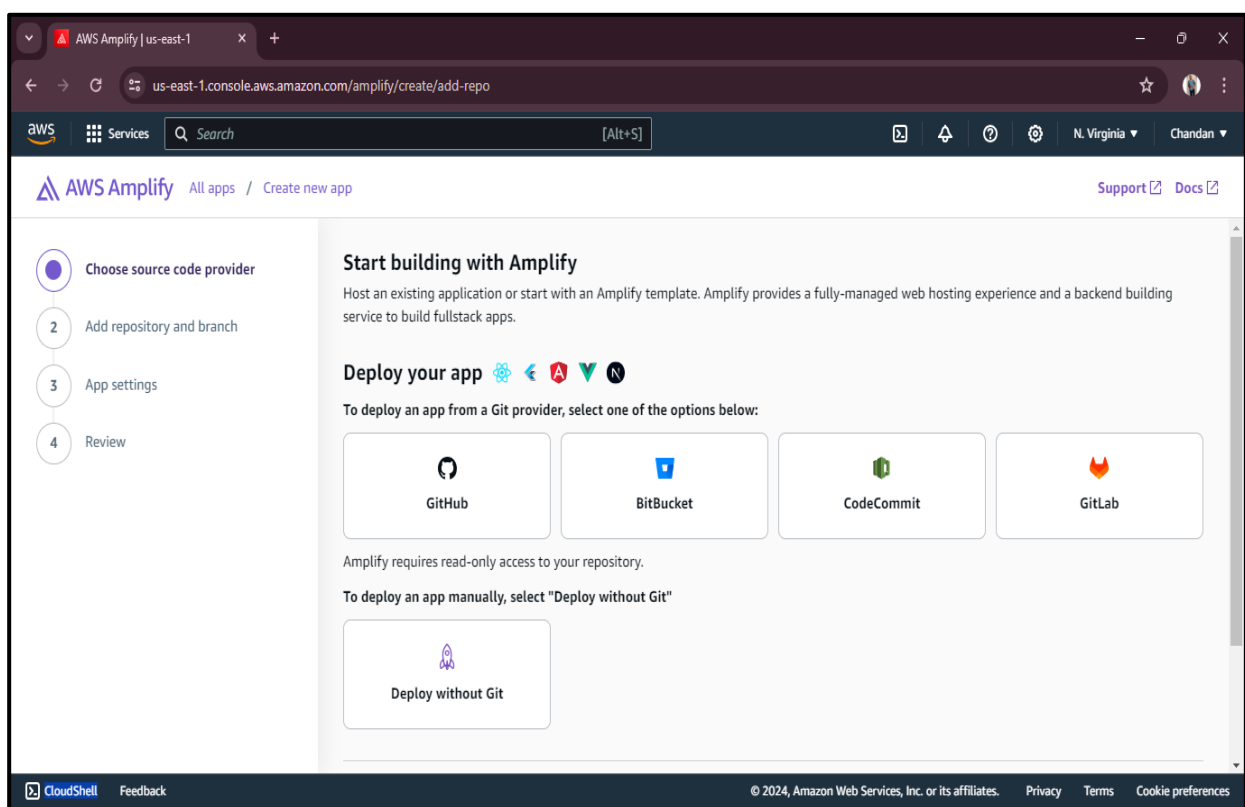


Once All the Connectivity Deploy the frontend Code by Creating it into zip file and upload it to AWS AMPLIFY.

AWS Amplify: AWS Amplify is a JavaScript library for frontend and mobile developers building cloud-enabled applications. The library is a declarative interface across different categories of operations in order to make common tasks easier to add into your application. The default implementation works with Amazon Web Services (AWS) resources but is designed to be open and pluggable for usage with other cloud services that wish to provide an implementation or custom backends.

Before Deploying in AWS Amplify, we need to ZIP the folder or file where the registration form code is present.

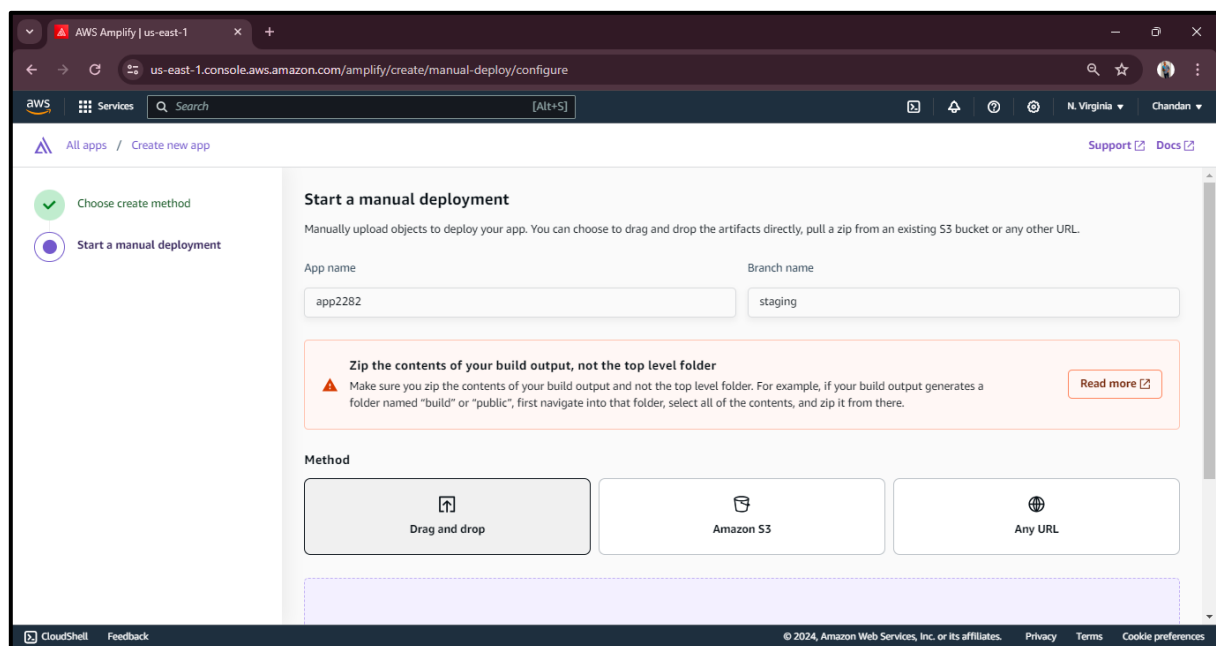
The page looks like:



AWS Amplify we can see that there are lot of ways to Deploy your app(e.g. GitHub, Bitbucket, Code Commit, GitLab)

If we want our website or code should available in another Git provider, we can select the following Git Provider:

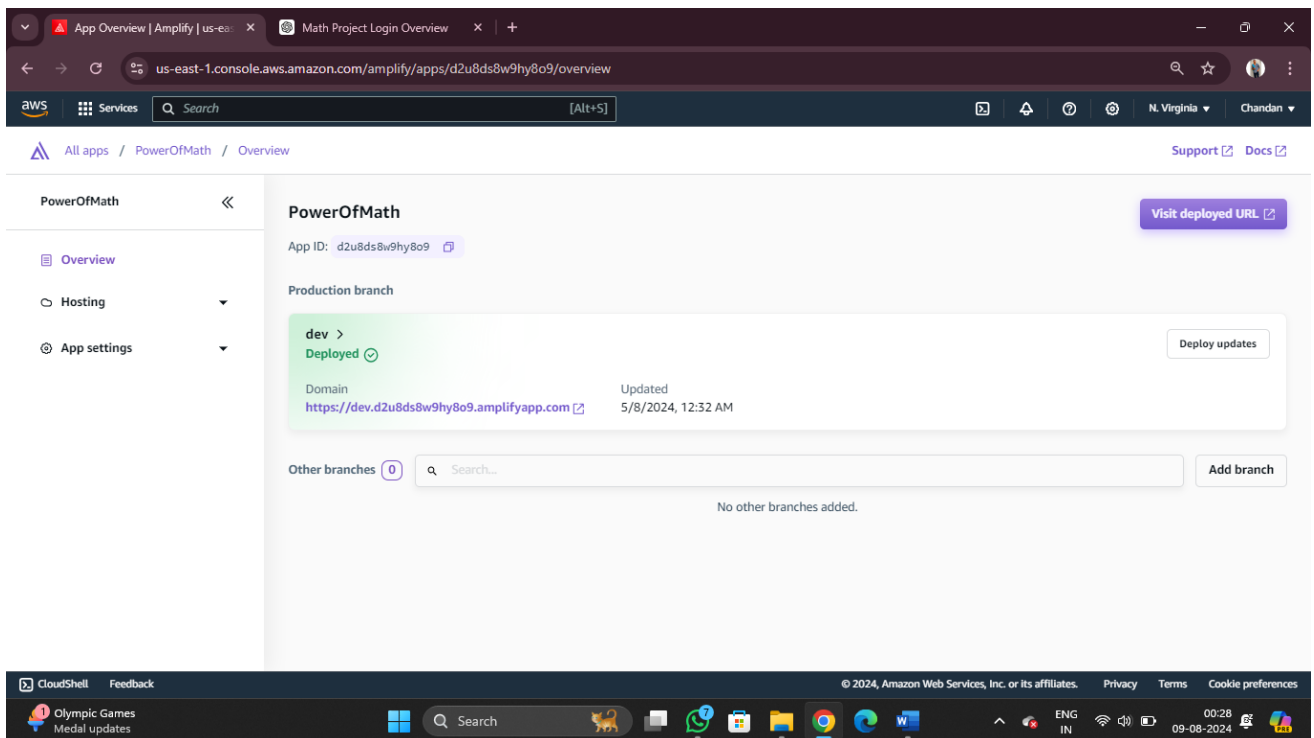
- **GitHub**: GitHub is a web-based version control and collaboration platform for software developers. Git is used to store the source code for a project and track the complete history of all changes to that code. It lets developers collaborate on a project more effectively by providing tools for managing possibly conflicting changes from multiple developers.
- **Bitbucket**: Bitbucket is a Git-based source code repository hosting service owned by Atlassian. Bitbucket offers both commercial plans and free accounts with an unlimited number of private repositories.
- **Code Commit**: AWS Code Commit is a secure, highly scalable, fully managed source control service that hosts private Git repositories.
- **GitLab**: GitLab is an open-core company that operates GitLab, a DevOps software package that can develop, secure, and operate software. The open-source software project was created by Ukrainian developer Dmytro Zaporozhets and Dutch developer Sytse



Sijbrandij. In 2018, GitLab Inc. was considered to be the first partly-Ukrainian unicorn.

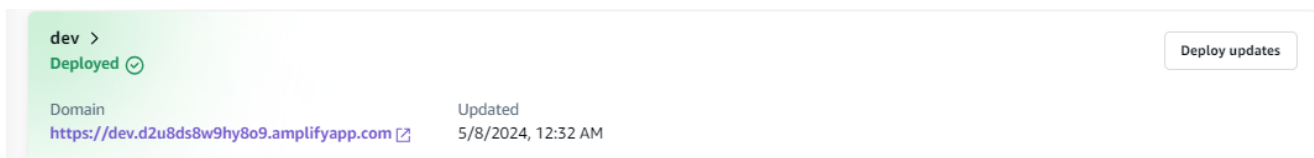
If we Choose the above Git Providers it may charge more so its better to deploy our Application without Git.

If we need to deploy our application without git the file or the folder should be compressed to ZIP folder.



The given domain link should Redirect to MATHEMATICAL CALCULATION form In a new tab.

The domain link will look like:



CHAPTER 6

SOURCE CODE

SOURCE CODE FOR MATHEMATICAL CALCULATION PROJECT

|— **Backend/**

| |— **Lambda functions/**

| | |— **Power.py**

| | | |— Power.py With Database Connection

| | | |— Power.py With Database Connection

| | |— **Trignomteric.py**

| | | |— Trignometric.py With Database Connection

| | | |— Trignometric.py With Database Connection

| | |— **logarithm.py**

| | | |— logarithm.py With Database Connection

| | | |— logarithm.py With Database Connection

|— **Frontend/**

| |— index.html

```
|— Backend/  
| |— Lambda functions/  
| | |— Power.py
```

Source Code without Database Connection

```
# import the JSON utility package  
import json  
  
# import the Python math library  
import math  
  
# import the AWS SDK (for Python the package name is boto3)  
import boto3  
  
# define the handler function that the Lambda service will use as an entry  
point  
def lambda_handler(event, context):  
    # extract the two numbers from the Lambda service's event object  
    base = int(event['base'])  
    exponent = int(event['exponent'])  
    math_result = math.pow(base, exponent)  
    # write result and time to the DynamoDB table using the object we  
    instantiated and save response in a variable  
    response = table.put_item(  
        Item={  
            'ID': f'Power_{base}^{exponent}',  
            'Result': str(round(math_result, 4)),  
            'LatestCalculationTime': now  
        }  
    )
```

```
# return a properly formatted JSON object
return {
    'statusCode': 200,
    'body': json.dumps({'result': round(math_result, 4)})
}
```

Source Code with Database Connection

```
# import the JSON utility package
import json

# import the Python math library
import math

# import the AWS SDK (for Python the package name is boto3)
import boto3

# import the time package to help with dates and date formatting
from time import gmtime, strftime

# create a DynamoDB object using the AWS SDK
dynamodb = boto3.resource('dynamodb')

# use the DynamoDB object to select our table
table = dynamodb.Table('awsprojectDB')

# store the current time in a human-readable format in a variable
now = strftime('%a, %d %b %Y %H:%M:%S', gmtime())

# define the handler function that the Lambda service will use as an entry
point
def lambda_handler(event, context):
    # extract the two numbers from the Lambda service's event object
    base = int(event['base'])
    exponent = int(event['exponent'])
    math_result = math.pow(base, exponent)
```

write result and time to the DynamoDB table using the object we instantiated and save response in a variable

```
response = table.put_item(
    Item={
        'ID': f'Power_{base}^{exponent}',
        'Result': str(round(math_result, 4)),
        'LatestCalculationTime': now
    }
)
```

return a properly formatted JSON object

```
return {
    'statusCode': 200,
    'body': json.dumps({'result': round(math_result, 4)})
}
```

| | | — Trigonometric.py

Source Code without Database Connection

```
import json
```

```
import math
```

Define the handler function that the Lambda service will use as an entry point

```
def lambda_handler(event, context):
```

```
    trig_results = {}
```

Helper function to store results in DynamoDB and prepare response

```
def store_and_prepare_response(calculation_type, value, result):
```

```
    table.put_item(
```

```
        Item={
```

```

        'ID': f'{calculation_type}_{value}',
        'Result': str(result),
        'LatestCalculationTime': now
    }
)
return {
    'float': result,
    'int': int(result)
}

```

try:

Sin calculation

if 'Sin' in event:

sin_result = round(math.sin(math.radians(float(event['Sin']))), 4)

trig_results['Sin']=store_and_prepare_response('Sin',event['Sin'],

sin_result)

Cos calculation

if 'Cos' in event:

cos_result = round(math.cos(math.radians(float(event['Cos']))), 4)

trig_results['Cos']=store_and_prepare_response('Cos',event['Cos'],

cos_result)

Tan calculation

if 'Tan' in event:

tan_result = round(math.tan(math.radians(float(event['Tan']))), 4)

trig_results['Tan']=store_and_prepare_response('Tan',event['Tan'],

tan_result)

aSin calculation

```

    if 'aSin' in event:
        aSin_value = float(event['aSin'])
        if -1 <= aSin_value <= 1:
            asin_result = round(math.degrees(math.asin(aSin_value)), 4)
trig_results['aSin']=store_and_prepare_response('aSin',event['aSin'],
asin_result)
        else:
            trig_results['aSin'] = "Value out of range for asin calculation"
# aCos calculation
    if 'aCos' in event:
        aCos_value = float(event['aCos'])
        if -1 <= aCos_value <= 1:
            acos_result = round(math.degrees(math.acos(aCos_value)), 4)
trig_results['aCos']=store_and_prepare_response('aCos',event['aCos'],
acos_result)
        else:
            trig_results['aCos'] = "Value out of range for acos calculation"
# aTan calculation
    if 'aTan' in event:
        atan_result
=
round(math.degrees(math.atan(float(event['aTan']))), 4)
trig_results['aTan']=store_and_prepare_response('aTan',event['aTan'],
atan_result)
    except ValueError as e:
        return {
            'statusCode': 400,
            'body': json.dumps({

```

```
        'errorMessage': str(e)
    })
}
```

Return a properly formatted JSON object

```
return {
    'statusCode': 200,
    'body': json.dumps(trig_results)
}
```

Source Code with Database Connection

```
import json
import math
import boto3
from time import gmtime, strftime
# Create a DynamoDB object using the AWS SDK
dynamodb = boto3.resource('dynamodb')
# Use the DynamoDB object to select our table
table = dynamodb.Table('awsprojectDB')
# Store the current time in a human-readable format in a variable
now = strftime('%a, %d %b %Y %H:%M:%S +0000', gmtime())
# Define the handler function that the Lambda service will use as an entry
point
def lambda_handler(event, context):
    trig_results = {}
    # Helper function to store results in DynamoDB and prepare response
    def store_and_prepare_response(calculation_type, value, result):
        table.put_item(
```



```

    Item={
        'ID': f'{calculation_type}_{value}',
        'Result': str(result),
        'LatestCalculationTime': now
    }
)
return {
    'float': result,
    'int': int(result)
}
try:
    # Sin calculation
    if 'Sin' in event:
        sin_result = round(math.sin(math.radians(float(event['Sin']))), 4)
        trig_results['Sin']=store_and_prepare_response('Sin',event['Sin'],
sin_result)
    # Cos calculation
    if 'Cos' in event:
        cos_result = round(math.cos(math.radians(float(event['Cos']))), 4)
        trig_results['Cos']=store_and_prepare_response('Cos',event['Cos'],
cos_result)
    # Tan calculation

if'Tan'ineventtan_result=round(math.tan(math.radians(float(event['Tan
']))),4)
trig_results['Tan']=store_and_prepare_response('Tan',event['Tan'],
tan_result)

```

```

# aSin calculation
if 'aSin' in event:
    asin_value = float(event['aSin'])
    if -1 <= asin_value <= 1:
        asin_result = round(math.degrees(math.asin(asin_value)), 4)
    trig_results['aSin']=store_and_prepare_response('aSin',event['aSin'],
asin_result)
    else:
        trig_results['aSin'] = "Value out of range for asin calculation"
# aCos calculation
if 'aCos' in event:
    acos_value = float(event['aCos'])
    if -1 <= acos_value <= 1:
        acos_result = round(math.degrees(math.acos(acos_value)), 4)
    trig_results['aCos']=store_and_prepare_response('aCos',event['aCos'],
acos_result)
    else:
        trig_results['aCos'] = "Value out of range for acos calculation"
# aTan calculation
if 'aTan' in event:
    atan_result=round(math.degrees(math.atan(float(event['aTan']))), 4)
    trig_results['aTan']=store_and_prepare_response('aTan',
event['aTan'], atan_result)
except ValueError as e:
    return {
        'statusCode': 400,
        'body': json.dumps({

```

```

        'errorMessage': str(e)
    })
}
# Return a properly formatted JSON object
return {
    'statusCode': 200,
    'body': json.dumps(trig_results)
}

```

| | | — logarithm.py

Source Code without Database Connection

```

import json
import math
# Helper function to store results in DynamoDB and prepare response
def store_and_prepare_response(calculation_type, value, result):
    table.put_item(
        Item={
            'ID': f'{calculation_type}_{value}',
            'Result': str(result),
            'LatestCalculationTime': now
        }
    )
    return {
        'float': result,
        'int': int(result)
    }

```

Define the handler function that the Lambda service will use as an entry point

```
def lambda_handler(event, context):
```

```
    try:
```

```
        value = float(event['value'])
```

```
        base = float(event['base'])
```

```
        if value <= 0 or base <= 0:
```

```
            raise ValueError("Both value and base must be greater than zero.")
```

```
        if base == 1:
```

```
            raise ValueError("Base cannot be 1.")
```

```
        # Perform the calculation
```

```
        result = math.log(value, base)
```

Store the result in DynamoDB and prepare the response

```
response_body=store_and_prepare_response("logarithm",f"{value}_base_{base}", result)
```

```
    response = {
```

```
        "statusCode": 200,
```

```
        "body": json.dumps(response_body)
```

```
    }
```

```
except Exception as e:
```

```
    response = {
```

```
        "statusCode": 400,  
        "body": json.dumps({"error": str(e)})  
    }
```

return response

Source Code with Database Connection

```
import json
```

```
import math
```

```
import boto3
```

```
from time import gmtime, strftime
```

```
# Create a DynamoDB object using the AWS SDK
```

```
dynamodb = boto3.resource('dynamodb')
```

```
# Use the DynamoDB object to select our table
```

```
table = dynamodb.Table('awsprojectDB')
```

```
# Store the current time in a human-readable format in a variable
```

```
now = strftime('%a, %d %b %Y %H:%M:%S +0000', gmtime())
```

```
# Helper function to store results in DynamoDB and prepare response
```

```
def store_and_prepare_response(calculation_type, value, result):
```

```
    table.put_item(  
        Item={
```

```
            'ID': f'{calculation_type}_{value}',
```

```
            'Result': str(result),
```

```
            'LatestCalculationTime': now
```

```
        }
```

```
    )
```

```
return {  
    'float': result,  
    'int': int(result)  
}
```

Define the handler function that the Lambda service will use as an entry point

```
def lambda_handler(event, context):
```

```
    try:
```

```
        value = float(event['value'])
```

```
        base = float(event['base'])
```

```
        if value <= 0 or base <= 0:
```

```
            raise ValueError("Both value and base must be greater than  
zero.")
```

```
        if base == 1:
```

```
            raise ValueError("Base cannot be 1.")
```

```
        # Perform the calculation
```

```
        result = math.log(value, base)
```

```
        # Store the result in DynamoDB and prepare the response
```

```
        response_body=store_and_prepare_response("logarithm",  
f'{value}_base_{base}', result)
```

```
        response = {
```

```
    "statusCode": 200,  
    "body": json.dumps(response_body)  
}
```

except Exception as e:

```
    response = {  
        "statusCode": 400,  
        "body": json.dumps({"error": str(e)})  
    }
```

return response

| — Frontend/

| | — index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Math Project Login</title>

<style>

@import

url('https://fonts.googleapis.com/css?family=Montserrat:400,800');

*** {**

box-sizing: border-box;

}

```
body {  
  background: #f6f5f7;  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  flex-direction: column;  
  font-family: 'Montserrat', sans-serif;  
  height: 100vh;  
  margin: -20px 0 50px;  
}
```

```
h1 {  
  font-weight: bold;  
  margin: 0;  
}
```

```
p {  
  font-size: 14px;  
  font-weight: 100;  
  line-height: 20px;  
  letter-spacing: 0.5px;  
  margin: 20px 0 30px;  
}
```

```
button {  
  border-radius: 20px;
```



```
border: 1px solid #FF4B2B;  
background-color: #FF4B2B;  
color: #FFFFFF;  
font-size: 12px;  
font-weight: bold;  
padding: 12px 45px;  
letter-spacing: 1px;  
text-transform: uppercase;  
transition: transform 80ms ease-in;  
}
```

```
button:active {  
    transform: scale(0.95);  
}
```

```
button:focus {  
    outline: none;  
}
```

```
button.ghost {  
    background-color: transparent;  
    border-color: #FFFFFF;  
}
```

```
form {  
    background-color: #FFFFFF;  
    display: flex;
```

```
align-items: center;
justify-content: center;
flex-direction: column;
padding: 0 50px;
height: 100%;
text-align: center;
}
```

```
input {
  background-color: #eee;
  border: none;
  padding: 12px 15px;
  margin: 8px 0;
  width: 100%;
}
```

```
.container {
  background-color: #fff;
  border-radius: 10px;
  box-shadow: 0 14px 28px rgba(0, 0, 0, 0.25), 0 10px 10px rgba(0, 0, 0, 0.22);
  position: relative;
  overflow: hidden;
  width: 768px;
  max-width: 100%;
  min-height: 480px;
}
```

```
.form-container {  
    position: absolute;  
    top: 0;  
    height: 100%;  
    transition: all 0.6s ease-in-out;  
}
```

```
.sign-in-container {  
    left: 0;  
    width: 50%;  
    z-index: 2;  
}
```

```
.container.right-panel-active .sign-in-container {  
    transform: translateX(100%);  
}
```

```
.sign-up-container {  
    left: 0;  
    width: 50%;  
    opacity: 0;  
    z-index: 1;  
}
```

```
.container.right-panel-active .sign-up-container {  
    transform: translateX(100%);
```

```
opacity: 1;  
z-index: 5;  
animation: show 0.6s;  
}
```

```
@keyframes show {  
  0%, 49.99% {  
    opacity: 0;  
    z-index: 1;  
  }
```

```
    50%, 100% {  
      opacity: 1;  
      z-index: 5;  
    }  
}
```

```
.overlay-container {  
  position: absolute;  
  top: 0;  
  left: 50%;  
  width: 50%;  
  height: 100%;  
  overflow: hidden;  
  transition: transform 0.6s ease-in-out;  
  z-index: 100;  
}
```

```
.container.right-panel-active .overlay-container {  
  transform: translateX(-100%);  
}
```

```
.overlay {  
  background: #FF416C;  
  background: -webkit-linear-gradient(to right, #FF4B2B, #FF416C);  
  background: linear-gradient(to right, #FF4B2B, #FF416C);  
  background-repeat: no-repeat;  
  background-size: cover;  
  background-position: 0 0;  
  color: #FFFFFF;  
  position: relative;  
  left: -100%;  
  height: 100%;  
  width: 200%;  
  transform: translateX(0);  
  transition: transform 0.6s ease-in-out;  
}
```

```
.container.right-panel-active .overlay {  
  transform: translateX(50%);  
}
```

```
.overlay-panel {  
  position: absolute;  
  display: flex;
```

```
align-items: center;  
justify-content: center;  
flex-direction: column;  
padding: 0 40px;  
text-align: center;  
top: 0;  
height: 100%;  
width: 50%;  
transform: translateX(0);  
transition: transform 0.6s ease-in-out;  
}
```

```
.overlay-left {  
  transform: translateX(-20%);  
}
```

```
.container.right-panel-active .overlay-left {  
  transform: translateX(0);  
}
```

```
.overlay-right {  
  right: 0;  
  transform: translateX(0);  
}
```

```
.container.right-panel-active .overlay-right {  
  transform: translateX(20%);
```

```
}
```

```
.math-container {  
  display: none;  
  flex-direction: column;  
  align-items: center;  
  width: 100%;  
}
```

```
.button-container {  
  display: flex;  
  justify-content: center;  
  margin: 20px;  
}
```

```
.button-container button {  
  background-color: #86C232;  
  border-color: #86C232;  
  color: #FFFFFFF;  
  font-family: system-ui;  
  font-size: 16px;  
  font-weight: bold;  
  margin: 10px;  
  padding: 15px;  
  cursor: pointer;  
  border-radius: 5px;  
}
```

```
.form-wrapper {  
  display: flex;  
  justify-content: space-around;  
  width: 100%;  
}
```

```
.form-container-inner {  
  display: none;  
  flex-direction: column;  
  align-items: center;  
  margin: 20px;  
  padding: 20px;  
  border: 1px solid #86C232;  
  border-radius: 10px;  
  width: 45%;  
}
```

```
.form-container-inner.active {  
  display: flex;  
}
```

```
form {  
  display: flex;  
  flex-direction: column;  
  align-items: center;  
}
```



```
label, input {  
    margin: 5px;  
    font-size: 16px;  
}
```

```
input {  
    padding: 8px;  
    border-radius: 5px;  
    border: 1px solid #86C232;  
    width: 200px;  
}
```

```
button[type="submit"] {  
    width: 150px;  
}
```

```
.result {  
    margin: 10px;  
    color: #86C232;  
    font-size: 16px;  
    white-space: pre-wrap;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div class="container" id="container">
```

```
<div class="form-container sign-up-container">
```

```

    <form id="register-form" onsubmit="registerUser(event)">
        <h1>Create Account</h1>
        <input type="text" id="register-name" placeholder="Name"
required />
        <input type="email" id="register-email" placeholder="Email"
required />
        <input          type="password"          id="register-password"
placeholder="Password" required />
        <button type="submit">Sign Up</button>
    </form>
</div>
<div class="form-container sign-in-container">
    <form id="login-form" onsubmit="loginUser(event)">
        <h1>Sign In</h1>
        <input type="email" id="login-email" placeholder="Email"
required />
        <input          type="password"          id="login-password"
placeholder="Password" required />
        <button type="submit">Sign In</button>
    </form>
</div>
<div class="overlay-container">
    <div class="overlay">
        <div class="overlay-panel overlay-left">
            <h1>Welcome Back!</h1>
            <p>To keep connected with us please login with your personal
info</p>

```

```

        <button class="ghost" id="signIn">Sign In</button>
    </div>
    <div class="overlay-panel overlay-right">
        <h1>Hello, Friend!</h1>
        <p>Enter your personal details and start your journey with
us</p>
        <button class="ghost" id="signUp">Sign Up</button>
    </div>
</div>
</div>
</div>
</div>

<div class="math-container" id="math-container">
    <h1>MATHEMATICAL CALCULATION</h1>
    <div class="button-container">
        <button          onclick="showForm('powerForm')">Power
Functions</button>
        <button          onclick="showForm('trigForm')">Trigonometric
Functions</button>
        <button          onclick="showForm('logForm')">Logarithm
Functions</button>
    </div>

    <div class="form-wrapper">
        <div id="powerForm" class="form-container-inner">
            <h2>Power Functions</h2>

```

```

        <form                                onsubmit="event.preventDefault();
callPowerAPI(document.getElementById('base').value,
document.getElementById('exponent').value);">
        <label>Base number:</label>
        <input type="text" id="base" placeholder="Enter base">
        <label>...to the power of:</label>
        <input  type="text"  id="exponent"  placeholder="Enter
exponent">
        <button type="submit">CALCULATE</button>
    </form>
    <pre id="powerResult" class="result"></pre>
</div>

```

```

<div id="trigForm" class="form-container-inner">
    <h2>Trigonometric Functions</h2>
    <form onsubmit="event.preventDefault(); callTrigAPI({
        Sin: document.getElementById('sinInput').value,
        Cos: document.getElementById('cosInput').value,
        Tan: document.getElementById('tanInput').value,
        aSin: document.getElementById('asinInput').value,
        aCos: document.getElementById('acosInput').value,
        aTan: document.getElementById('atanInput').value
    });">
        <label>Sin Of:</label>
        <input  type="text"  id="sinInput"  placeholder="Enter
value">
        <label>Cos Of:</label>

```

```

        <input type="text" id="cosInput" placeholder="Enter
value">

        <label>Tan Of:</label>

        <input type="text" id="tanInput" placeholder="Enter
value">

        <label>aSin Of:</label>

        <input type="text" id="asinInput" placeholder="Enter
value">

        <label>aCos Of:</label>

        <input type="text" id="acosInput" placeholder="Enter
value">

        <label>aTan Of:</label>

        <input type="text" id="atanInput" placeholder="Enter
value">

        <button type="submit">CALCULATE</button>

    </form>

    <pre id="trigResult" class="result"></pre>
</div>

<div id="logForm" class="form-container-inner">
    <h2>Logarithm Functions</h2>

    <form
                                onsubmit="event.preventDefault();
callLogAPI(document.getElementById('logBase').value,
document.getElementById('logNumber').value);">

        <label>Base of logarithm:</label>

        <input type="text" id="logBase" placeholder="Enter
base">

```

```
        <label>Number:</label>
        <input type="text" id="logNumber" placeholder="Enter
number">
        <button type="submit">CALCULATE</button>
    </form>
    <pre id="logResult" class="result"></pre>
</div>
</div>
</div>

<script>
    const signUpButton = document.getElementById('signUp');
    const signInButton = document.getElementById('signIn');
    const container = document.getElementById('container');
    const mathContainer = document.getElementById('math-
container');

    let users = [];

    signUpButton.addEventListener('click', () => {
        container.classList.add('right-panel-active');
    });

    signInButton.addEventListener('click', () => {
        container.classList.remove('right-panel-active');
    });
```

```

function registerUser(event) {
    event.preventDefault();
    const name = document.getElementById('register-name').value;
    const email = document.getElementById('register-email').value;
    const password = document.getElementById('register-
password').value;

    users.push({ name, email, password });
    alert('Registration successful!');
    container.classList.remove('right-panel-active');
}

function loginUser(event) {
    event.preventDefault();
    const email = document.getElementById('login-email').value;
    const password = document.getElementById('login-
password').value;

    const user = users.find(u => u.email === email && u.password
=== password);
    if (user) {
        alert('Login successful!');
        container.style.display = 'none';
        mathContainer.style.display = 'flex';
    } else {
        alert('Invalid email or password!');
    }
}

```

```
}
```

```
function showForm(formId) {
```

```
document.getElementById('powerForm').classList.remove('active');  
document.getElementById('trigForm').classList.remove('active');  
document.getElementById('logForm').classList.remove('active');  
if (formId) {  
    document.getElementById(formId).classList.add('active');  
}  
}
```

```
function callPowerAPI(base, exponent) {
```

```
    var myHeaders = new Headers();  
    myHeaders.append("Content-Type", "application/json");  
    var raw = JSON.stringify({ "base": base, "exponent": exponent  
});  
    var requestOptions = {  
        method: 'POST',  
        headers: myHeaders,  
        body: raw,  
        redirect: 'follow'  
};  
    fetch("https://5ov3hm0id2.execute-api.us-east-  
1.amazonaws.com/dev/power", requestOptions)  
        .then(response => response.json())  
        .then(result => {
```



```

        document.getElementById('powerResult').innerText =
result.body;
    })
    .catch(error => console.log('error', error));
}

function callTrigAPI(payload) {
    var myHeaders = new Headers();
    myHeaders.append("Content-Type", "application/json");
    var raw = JSON.stringify(payload);
    var requestOptions = {
        method: 'POST',
        headers: myHeaders,
        body: raw,
        redirect: 'follow'
    };
    fetch('https://5ov3hm0id2.execute-api.us-east-
1.amazonaws.com/dev/trig', requestOptions)
        .then(response => response.json())
        .then(result => {
            document.getElementById('trigResult').innerText =
JSON.stringify(result, null, 4);
        })
        .catch(error => console.log('error', error));
}

function callLogAPI(base, number) {

```

```
var myHeaders = new Headers();  
myHeaders.append("Content-Type", "application/json");  
var raw = JSON.stringify({ "base": base, "number": number });  
var requestOptions = {  
    method: 'POST',  
    headers: myHeaders,  
    body: raw,  
    redirect: 'follow'  
};  
fetch("https://5ov3hm0id2.execute-api.us-east-  
1.amazonaws.com/dev/log", requestOptions)  
    .then(response => response.json())  
    .then(result => {  
        document.getElementById('logResult').innerText=result.body;  
    })  
    .catch(error => console.log('error', error));  
}  
</script>  
</body>  
</html>
```

CHAPTER 7

TESTING

SYSTEM TESTING

Testing performs a very critical role for quality assurance and ensuring the reliability of the software. The success of testing for errors in program depends critically on the test cases.

The basic levels of testing are:-

1. Unit testing
2. Integrating testing
3. System testing
4. acceptance testing

SYSTEM TESTING:

Software testing is a critical element of the software quality assurance and represents the ultimate review of specification, design and coding. Testing presents an interesting anomaly for the software. The testing phase involves testing of a system using various test data. Preparation of the test data plays a vital role in the system testing.

The development of software system involves a series of production activities where opportunities for injection of human errors are enormous. Errors may begin to occur at the very inception of the process where the objectives may be enormously or imperfectly specified as well in later design and development stages. Because of human inability to perform and communicate with perfection, software development is followed by assurance activity.

Quality assurance is the review of software products and related documentation for completeness, correctness, reliability and maintainability.

TEST CASE DESIGN:

The test case design for software should be done in such a way that the errors can be found out with a minimum amount of time and effort. A test case is a set of data that the system will process as normal as input. Mainly there are two methods for the test case design-black box test and white box testing.

This method focuses on the functional requirements of the software. It attempts to find out the errors of the following categories-incorrect and missing functions, interface errors, errors in data structure or external database access, performance errors and initialization and termination errors.

WHITE BOX TEST:

This method is also called path testing, is a test design method that uses the control structure of the procedural design test cases. It is predicted on a close examination of procedural design. Logical paths through which the software are tested by providing test cases that exercise specific sets of condition and or loops.

Mainly there are two levels of testing, namely unit testing and integration testing.

TESTING STRATEGY:

An initial testing strategy is developed upon completion of the analysis stage and is refined and updated during the design and build stages of the project.

The testing strategies followed in the project are as follows:

UNIT TESTING:

Unit testing focuses on verification effort on the smallest unit of software design module. Using the unit test plans. Prepared in the design phase of the system as a guide, important control paths are tested. Boundary conditions were checked. All independent paths were exercises to ensure that all statements in the module are executed at least once and all error handling situation. This testing was carried out during the programming itself. At the end of this testing phase, each unit was found to be working satisfactorily, as regard to the output from the module,

INTEGRATION TESTING:

Data can be lost across an interface one module can have effect on another's sub functions, when combined may not produce the desired major function; global data structure can be present problems. Integration testing is a symmetric technique for constructing tests to uncover errors associated with the interface. All modules are combined in this testing step. Then the entire program was tested as a whole.

BLACK BOX TESTING:

VALIDATION PROGRAM

At the culmination of the integration testing, the software was completely assembled as a package, interfacing errors have been uncovered and corrected and a final series of software validation begun. Here we test the system in a manner that can be reasonable expected by customer the system was tested again system requirement

OUTPUT TESTING:

After performing validation test the next phase in output test of the system, since no system could be useful if it does not produce the desired output in a desired format by considering the format of the report/output, output/report is generated or displayed and is tested.

Here output format is considered in two ways; one is the screen and other is a printed form.

TESTING PROCESS

First thing in the testing process is to make plans. In the test plan, test cases are selected to ensure that there ii's an error in the program then it is executed by one of the test cases. The success of testing depends critically on these test cases selected.

TEST CASE SPECIFICATION:

It is the major activity in the testing process. Test cases are specified for testing each unit, the specification gives all the test cases inputs to be used and the outputs expected for these test cases.

PERFORMANCE AND LIMITATION

PERFORMANCE:-

- It can be upgraded in more future
- All modules have been tested and it is working properly
- Basic security features has been given > It supports various reports
- It does not have any constraints

LIMITATIONS:-

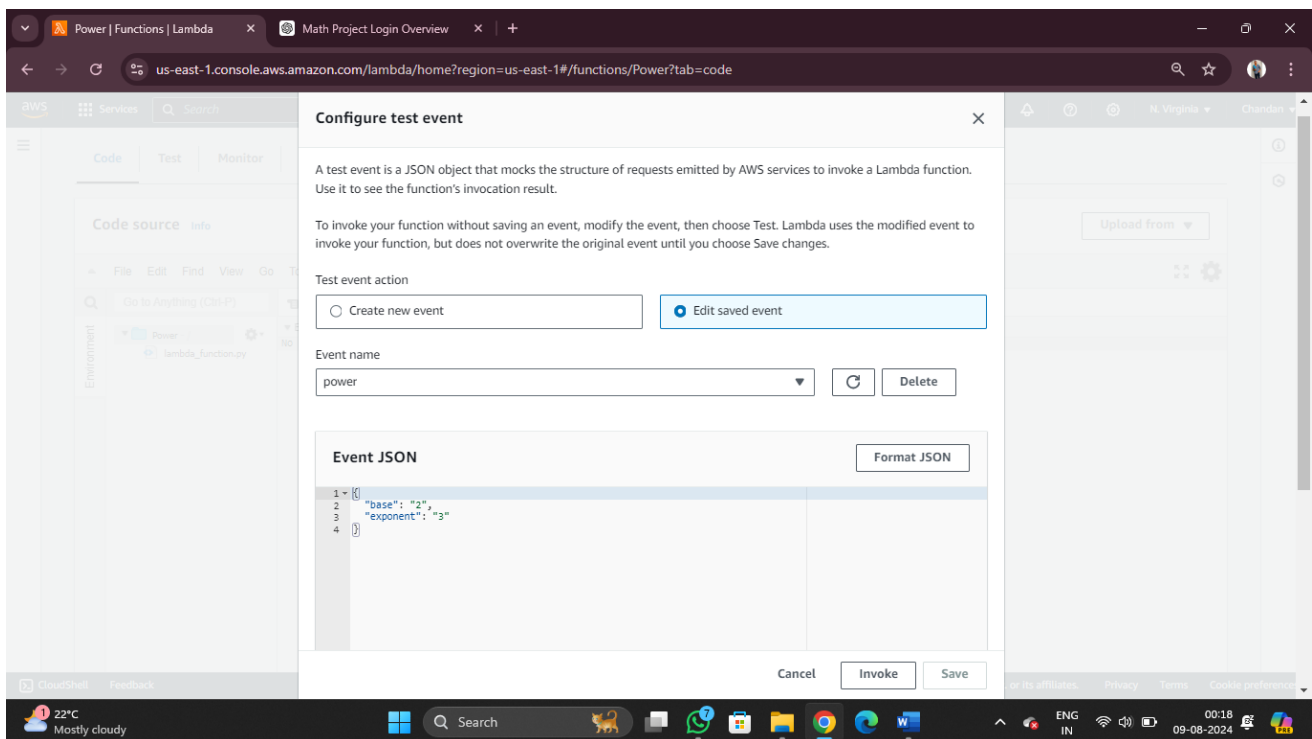
- Less security is given for the project.

TESTINGS DONE IN LAMBDA

➤ **POWER**

Configuring Test Event

```
{  
  
  "base": "2",  
  
  "exponent": "3"  
}
```



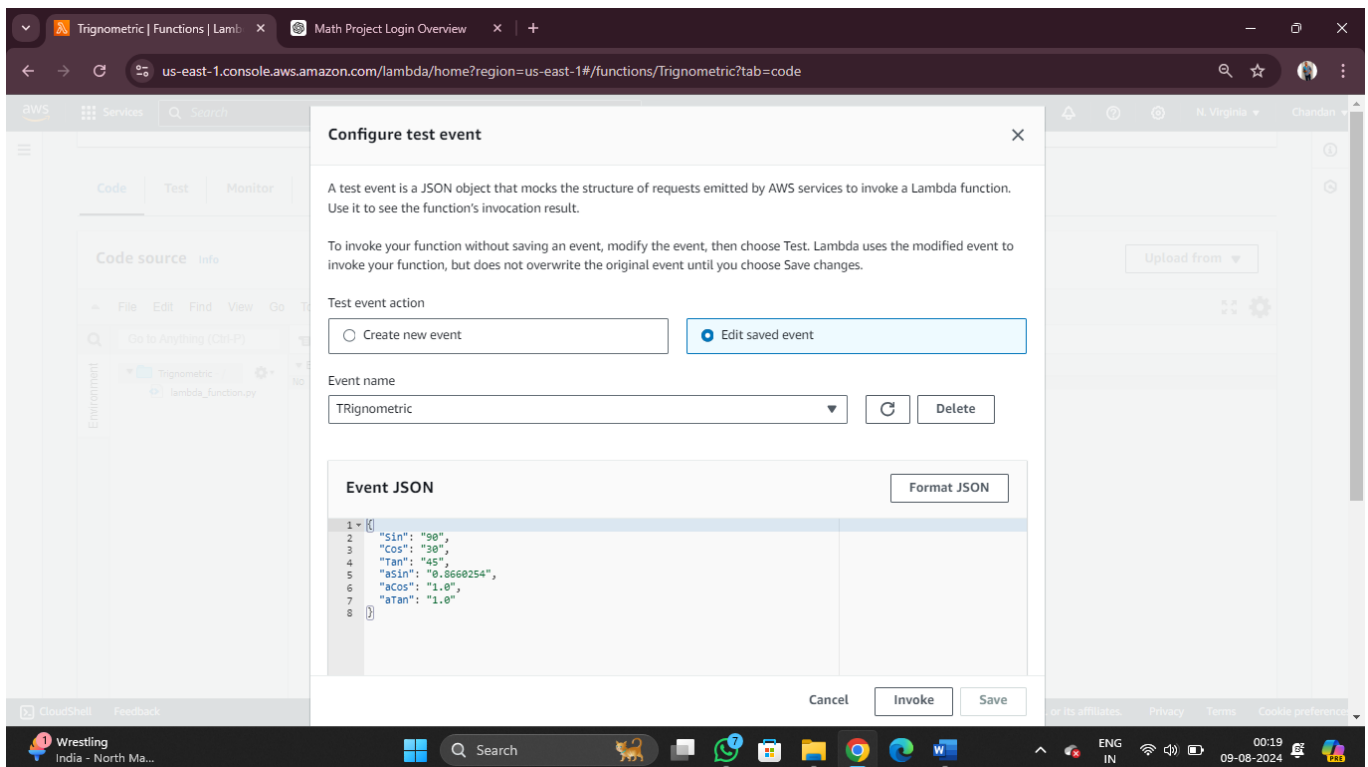
Output

```
{  
  "statusCode": 200,  
  "body": "{\"result\": 8.0}"  
}
```

➤ TRIGNOMETRIC

Configuring Test Event

```
{  
  "Sin": "90",  
  "Cos": "30",  
  "Tan": "45",  
  "aSin": "0.8660254",  
  "aCos": "1.0",  
  "aTan": "1.0"  
}
```



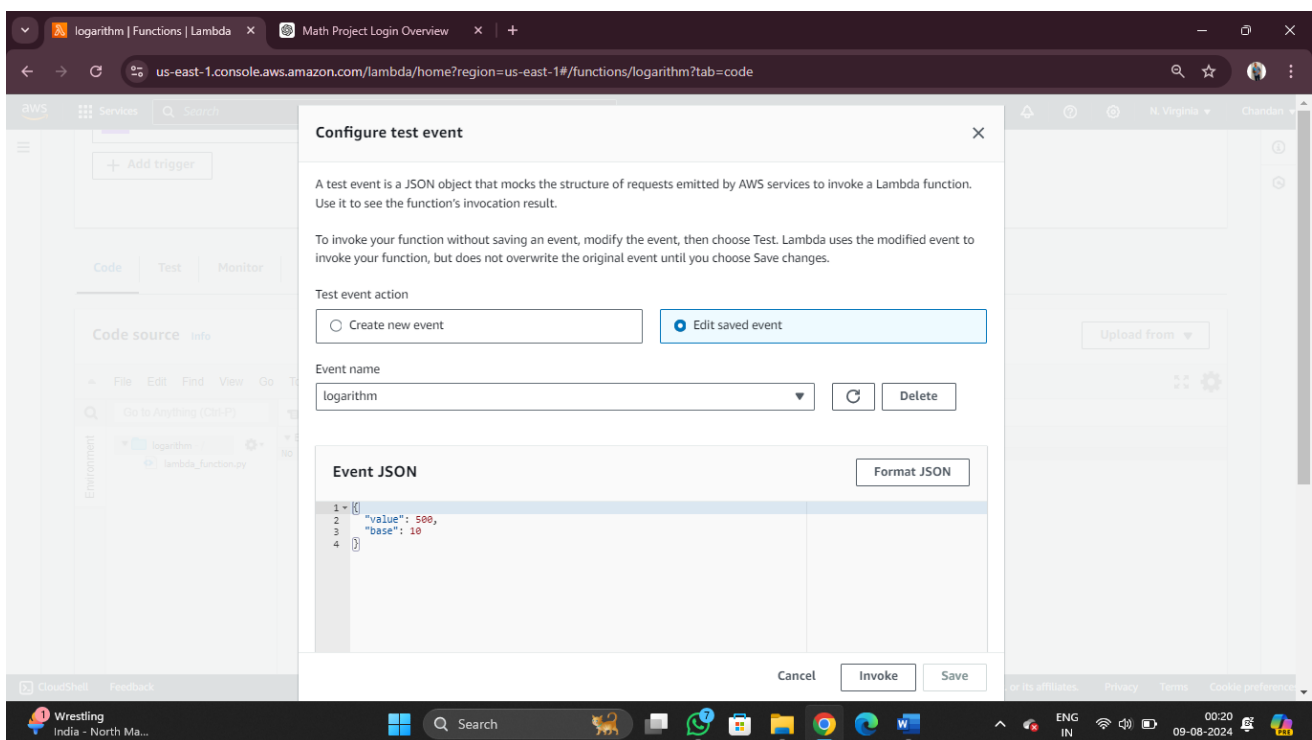
Output

```
{  
  "statusCode": 200,  
  "body": "{\"Sin\": {\"float\": 1.0, \"int\": 1}, \"Cos\": {\"float\": 0.866, \"int\": 0}, \"Tan\": {\"float\": 1.0, \"int\": 1}, \"aSin\": {\"float\": 60.0, \"int\": 60}, \"aCos\": {\"float\": 0.0, \"int\": 0}, \"aTan\": {\"float\": 45.0, \"int\": 45}}\"  
}
```

➤ LOGARITHM

Configuring Test Event

```
{  
  "value": 100,  
  "base": 10  
}
```



Output:

```
{  
  "statusCode": 200,  
  "body": "{\\"float\\": 2.0, \\"int\\": 2}"  
}
```

CHAPTER 8

FORMS

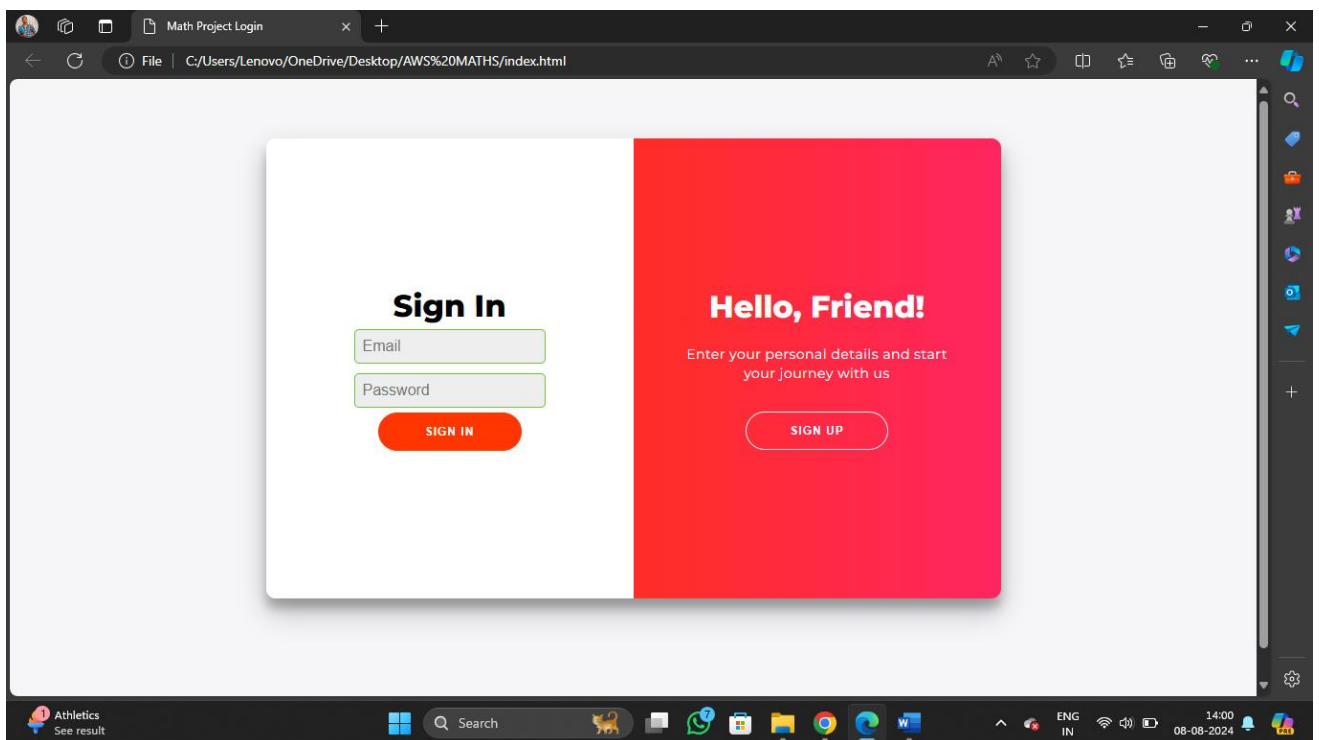
1)REGESTRATION FORM

The screenshot shows a web browser window with the title 'Math Project Login'. The address bar displays the file path 'C:/Users/Lenovo/OneDrive/Desktop/AWS%20MATHS/index.html'. The main content area features a registration form with a red background on the left and a white background on the right. The red background contains the text 'Welcome Back!' and 'To keep connected with us please login with your personal info' with a 'SIGN IN' button. The white background contains the title 'Create Account' and three input fields for 'Name', 'Email', and 'Password', followed by a 'SIGN UP' button. The Windows taskbar at the bottom shows the time as 14:00 on 08-08-2024.

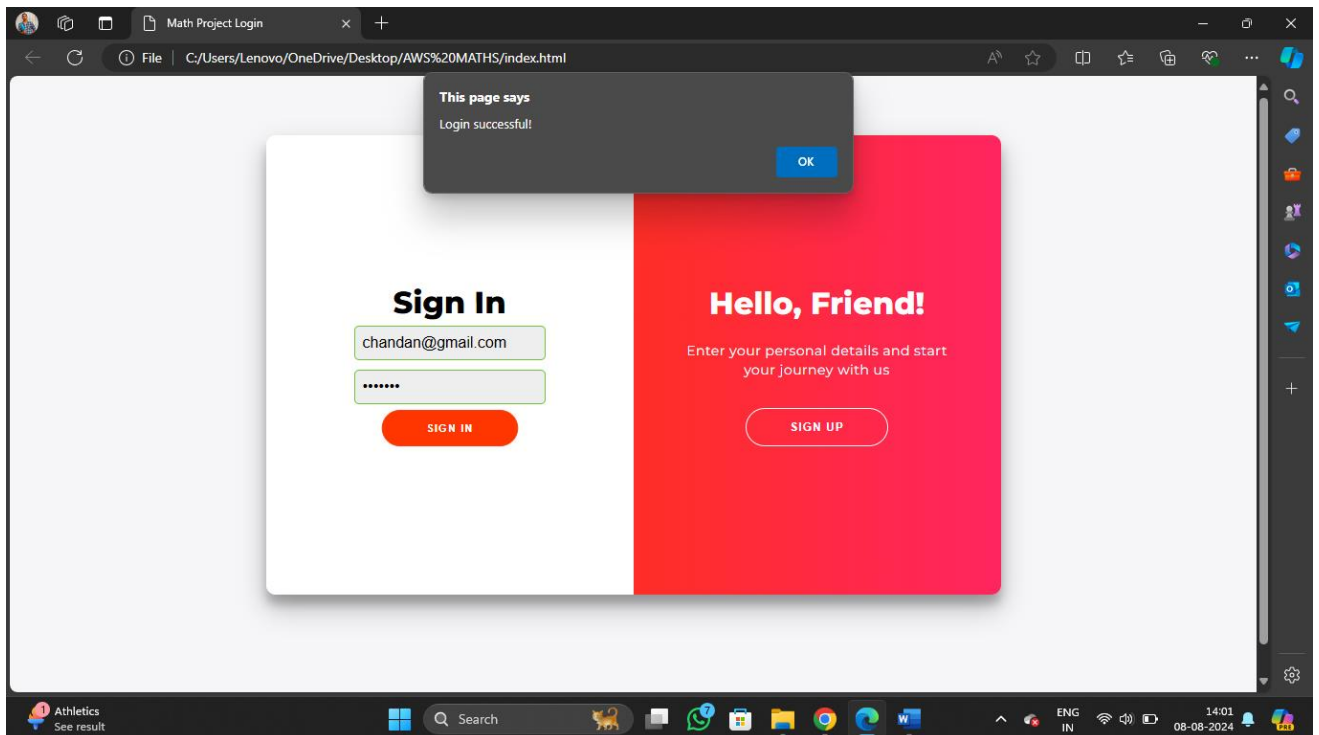
2)REGESTRATION SUCCESSFUL FORM

The screenshot shows the same web browser window as before, but with a dark grey notification box overlaid on the form. The notification box contains the text 'This page says' and 'Registration successful!' with an 'OK' button. The registration form is still visible in the background, with the 'Name' field filled with 'chandan', the 'Email' field filled with 'chandan@gmail.com', and the 'Password' field filled with '*****'. The 'SIGN UP' button is highlighted in red. The Windows taskbar at the bottom shows the time as 20:59 on 08-08-2024.

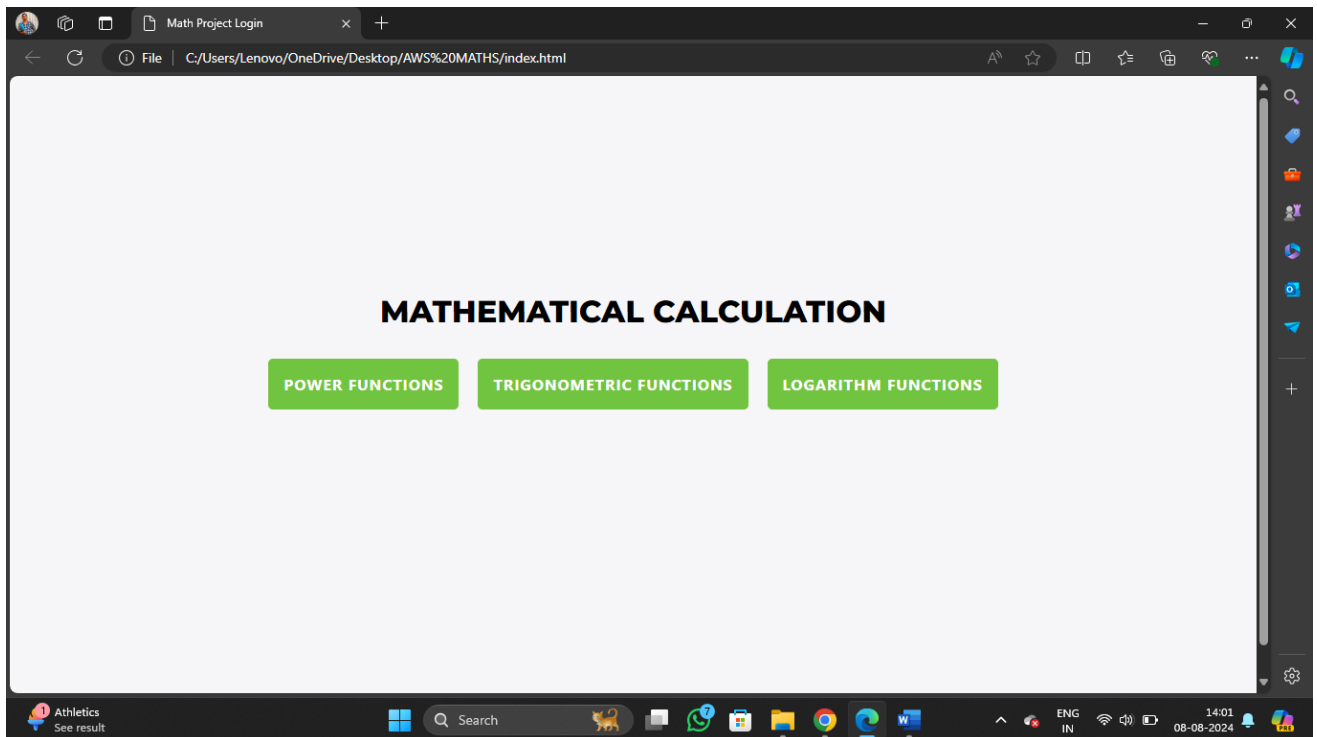
3) LOGIN FORM



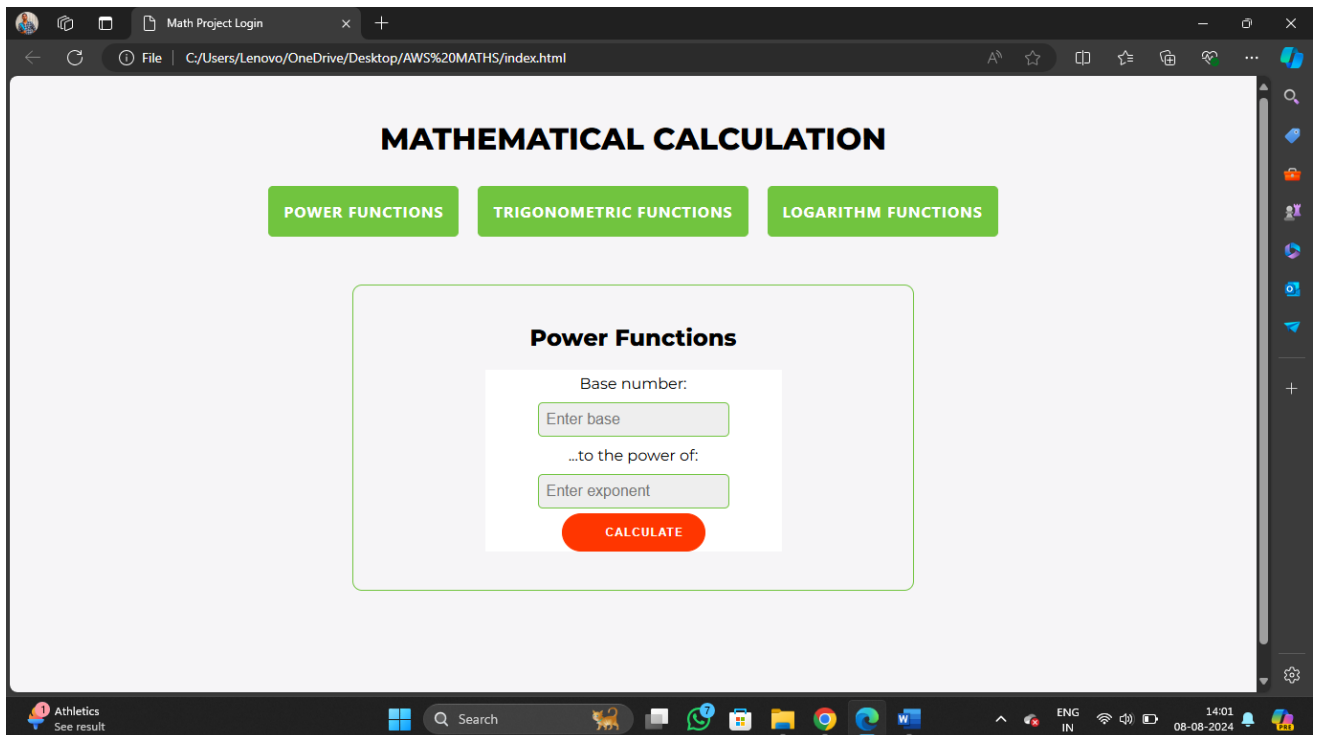
4) LOGIN SUCCESSFUL FORM



5)MATHEMATICAL CALCULATION FORM



6)POWER FUNCTION FORM WITHOUT OUTPUT



7) POWER FUNCTION FORM WITH OUTPUT

The screenshot shows a web browser window with the title "Math Project Login". The address bar shows the file path "C:/Users/Lenovo/OneDrive/Desktop/AWS%20MATHS/index.html". The main heading is "MATHEMATICAL CALCULATION". Below it are three green buttons: "POWER FUNCTIONS", "TRIGONOMETRIC FUNCTIONS", and "LOGARITHM FUNCTIONS". The "POWER FUNCTIONS" button is selected. The "Power Functions" section contains a form with two input fields: "Base number:" with the value "4" and "...to the power of:" with the value "4". Below the inputs is a red "CALCULATE" button. The output is displayed as `{"result": 256.0}`. The Windows taskbar at the bottom shows the date "08-08-2024" and time "21:07".

MATHEMATICAL CALCULATION

POWER FUNCTIONS **TRIGONOMETRIC FUNCTIONS** **LOGARITHM FUNCTIONS**

Power Functions

Base number:

...to the power of:

CALCULATE

`{"result": 256.0}`

8) TRIGNOMETRIC FUNCTION FORM WITHOUT OUTPUT

The screenshot shows the same web browser window as above, but the "TRIGNOMETRIC FUNCTIONS" button is selected. The "Trigonometric Functions" section contains a form with eight input fields, each with a label and a placeholder "Enter value": "Sin Of:", "Cos Of:", "Tan Of:", "aSin Of:", "aCos Of:", "aTan Of:", and two unlabeled fields. Below the inputs is a red "CALCULATE" button. The Windows taskbar at the bottom shows the date "08-08-2024" and time "14:01".

MATHEMATICAL CALCULATION

POWER FUNCTIONS **TRIGNOMETRIC FUNCTIONS** **LOGARITHM FUNCTIONS**

Trigonometric Functions

Sin Of:

Cos Of:

Tan Of:

aSin Of:

aCos Of:

aTan Of:

CALCULATE

9) TRIGNOMETRIC FUNCTION FORM WITH OUTPUT

MATHEMATICAL CALCULATION

POWER FUNCTIONS TRIGNOMETRIC FUNCTIONS LOGARITHM FUNCTIONS

Trigonometric Functions

Sin Of:

Cos Of:

Tan Of:

aSin Of:

aCos Of:

aTan Of:

CALCULATE

```
{
  "statusCode": 200,
  "body": {
    "\sin": {
      "float": 1.0,
      "int": 1
    },
    "\cos": {
      "float": 0.7071,
      "int": 0
    },
    "\tan": {
      "float": -0.0,
      "int": 0
    },
    "aSin": {
      "float": 89.986,
      "int": 89
    },
    "aCos": {
      "float": 55.3968,
      "int": 55
    },
    "aTan": {
      "float": 89.0452,
      "int": 89
    }
  }
}
```

10) LOGARITHM FUNCTION FORM WITHOUT OUTPUT

MATHEMATICAL CALCULATION

POWER FUNCTIONS TRIGNOMETRIC FUNCTIONS LOGARITHM FUNCTIONS

Logarithm Functions

Base of logarithm:

Number:

CALCULATE

11) LOGARITHM FUNCTION FORM WITH OUTPUT

The screenshot shows a web browser window with the title "Math Project Login". The address bar shows the file path "C:/Users/Lenovo/OneDrive/Desktop/AWS%20MATHS/index.html". The main content area is titled "MATHEMATICAL CALCULATION" and features three green buttons: "POWER FUNCTIONS", "TRIGONOMETRIC FUNCTIONS", and "LOGARITHM FUNCTIONS". The "LOGARITHM FUNCTIONS" button is selected, and a modal window titled "Logarithm Functions" is displayed. This modal contains two input fields: "Base of logarithm:" with the value "10" and "Number:" with the value "500". Below these fields is a red "CALCULATE" button. At the bottom of the modal, the output is shown as a JSON object: `{"answer": "'2'"}`. The Windows taskbar at the bottom shows the date and time as "08-08-2024" and "21:15", along with various system icons and a search bar.

MATHEMATICAL CALCULATION

POWER FUNCTIONS TRIGONOMETRIC FUNCTIONS LOGARITHM FUNCTIONS

Logarithm Functions

Base of logarithm:
10

Number:
500

CALCULATE

`{"answer": "'2'"}`

CHAPTER 9

CONCLUSION

This project is a **Mathematical Calculation Application** with a **user registration and login system** integrated with front-end HTML, CSS, and JavaScript, and back-end APIs hosted on AWS. The primary purpose of the project is to allow users to perform various mathematical calculations, including power functions, trigonometric functions, and logarithm functions, through an interactive and user-friendly web interface.

Key Features

1. **User Authentication:**
 - **Registration:** Users can create an account by providing their name, email, and password. The registration information is stored in a client-side array for simplicity.
 - **Login:** Registered users can log in using their email and password. Successful login transitions the user to the mathematical calculation section of the application.
2. **Mathematical Calculation Interface:**
 - **Power Functions:** Allows users to calculate power by providing a base and an exponent.
 - **Trigonometric Functions:** Users can calculate the sine, cosine, tangent, arcsine, arccosine, and arctangent of given values.
 - **Logarithm Functions:** Users can compute logarithms by providing the base and the number.

3. **API Integration:**

- Each mathematical function (power, trigonometric, logarithm) is linked to a specific API endpoint. The application sends requests with the necessary parameters, and the server responds with the results, which are then displayed on the page.

Technical Summary

- **Frontend:**

- The application is built with standard HTML, CSS, and JavaScript. The UI is modern and responsive, using flexbox for layout and a clean, minimalistic design inspired by popular web applications.
- The dynamic effects, such as switching between the login and registration forms, are handled using CSS transitions.

- **Backend:**

- The backend API is deployed on AWS, likely using AWS Lambda, which handles the mathematical computations. The frontend interacts with these endpoints via fetch API calls in JavaScript.
- Three endpoints are utilized:
 - **Power Function Endpoint:** Accepts base and exponent values, returning the calculated power.
 - **Trigonometric Function Endpoint:** Processes various trigonometric and inverse trigonometric calculations.
 - **Logarithm Function Endpoint:** Computes the logarithm of a given number with a specified base.

Potential Enhancements

1. Security Improvements:

- Implement proper user authentication and password management, possibly using OAuth or JWT tokens for secure sessions.
- Store user data on a secure backend database instead of in a client-side array.

2. Enhanced User Experience:

- Provide more descriptive error messages during registration and login.
- Add more mathematical functions or tools to expand the utility of the application.

3. Scalability:

- Implement a backend database (like DynamoDB) to persist user information and calculation history, enabling a richer user experience.
- Optimize the frontend to handle more complex calculations or a larger user base.

Conclusion

This project successfully demonstrates how to build a user-friendly web application with basic user authentication and mathematical calculation capabilities. It is a solid foundation for further development, whether for educational purposes, as a hobby project, or as a prototype for more complex systems.

CHAPTER 10

FUTURE ENHANCEMENT

Future Enhancements

1. Database Integration:

- **User Data Persistence:** Move from client-side storage to a server-side database (e.g., DynamoDB). This will allow users to maintain their profiles, save preferences, and access calculation history across sessions and devices.
- **Calculation History:** Implement a feature to save and view past calculations, enabling users to track their work over time. This history could be filterable by date, type of calculation, or specific parameters.

2. Enhanced Security Measures:

- **User Authentication:** Implement OAuth 2.0 or JWT-based authentication for secure login and session management. Consider integrating with popular identity providers like Google or Facebook for a seamless login experience.
- **Password Encryption:** Use strong hashing algorithms like bcrypt for password storage, ensuring user credentials are securely stored in the database.
- **Two-Factor Authentication (2FA):** Add an extra layer of security by requiring users to verify their identity using a second factor, such as a mobile app or SMS code

3. **Expanded Mathematical Capabilities:**

- **Additional Functions:** Introduce new mathematical functions such as matrix operations, statistical calculations, calculus (differentiation and integration), and number theory.
- **Graphical Visualization:** Incorporate graphing capabilities to visually represent functions and their results. For example, users could graph power functions or trigonometric curves directly within the application.
- **Custom Function Editor:** Allow users to create and save custom functions by combining basic operations and functions, expanding the application's versatility.

4. **User Interface and Experience Improvements:**

- **Responsive Design:** Ensure the application is fully responsive across all devices, including smartphones and tablets, to reach a broader audience.
- **Theme Customization:** Allow users to choose or customize themes, including color schemes and font styles, to personalize their experience.
- **Localization and Internationalization:** Add support for multiple languages and regional settings to make the application accessible to a global audience.

5. **Performance Optimization:**

- **Asynchronous Processing:** Enhance the performance of API calls by optimizing the backend processing, potentially using

asynchronous operations or multi-threading to handle complex calculations more efficiently.

- **Caching Frequently Used Calculations:** Implement caching mechanisms for frequently requested calculations to reduce server load and improve response times.

6. **Mobile Application Development:**

- **Mobile App:** Develop a mobile version of the application for iOS and Android, offering a native experience with additional features like offline mode, push notifications, and integration with device sensors (e.g., accelerometer for trigonometric calculations).

7. **Integration with Educational Platforms:**

- **E-Learning Integration:** Partner with educational platforms to offer the application as a learning tool. Create guided exercises, tutorials, and assessments to help students understand mathematical concepts.
- **Collaboration Features:** Implement features that allow users to share calculations, work collaboratively on problems in real-time, or participate in community challenges and leaderboards.

8. **AI-Powered Assistance:**

- **AI Tutor:** Incorporate AI to provide real-time hints, tips, or explanations as users perform calculations. The AI could also suggest alternative methods or highlight common mistakes.
- **Natural Language Processing (NLP):** Enable users to input queries or problems in natural language (e.g., “What’s the sine of

45 degrees?”), with the application parsing and solving the problem accordingly.

These enhancements will significantly improve the application’s functionality, security, user experience, and educational value, transforming it from a basic tool into a comprehensive, versatile platform for mathematical exploration and learning.

CHAPTER 11

BIBLIOGRAPHY

Bibliography

1. HTML, CSS, and JavaScript:

- **MDN Web Docs:** Comprehensive documentation on web technologies including HTML, CSS, and JavaScript, which served as a guide for building the frontend of the application.
 - Mozilla Developer Network (MDN). "HTML: HyperText Markup Language." MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/HTML>.
 - Mozilla Developer Network (MDN). "CSS: Cascading Style Sheets." MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/CSS>.
 - Mozilla Developer Network (MDN). "JavaScript." MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.

2. Web Design and User Interface:

- **Google Fonts:** Used for importing the Montserrat font, contributing to the visual design and typography of the application.
- Google. "Montserrat." Google Fonts. <https://fonts.google.com/specimen/Montserrat>.
- **CSS Tricks:** Provided practical tips and tricks for implementing responsive design and transitions.
- Chris Coyier. "A Complete Guide to Flexbox." CSS Tricks. <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>.

3. **AWS Lambda and API Development:**

- **AWS Documentation:** Guidelines and best practices for deploying serverless functions using AWS Lambda, which were essential for setting up the backend APIs.
- Amazon Web Services (AWS). "AWS Lambda Developer Guide." Amazon. <https://docs.aws.amazon.com/lambda/latest/dg/welcome.html>.
- **Serverless Framework:** Helped in organizing and deploying the serverless APIs for the mathematical functions.
- Serverless Framework. "Serverless Framework Documentation." <https://www.serverless.com/framework/docs/>.

4. **Mathematical Functions and Algorithms:**

- **Wolfram Alpha:** Served as a reference for validating the mathematical functions and formulas implemented in the application.
- Wolfram Alpha LLC. "Mathematical Functions." Wolfram Alpha. <https://www.wolframalpha.com/>.
- **Math.js:** A comprehensive library for JavaScript, offering a range of mathematical functions, which provided inspiration for the mathematical computations.
- Jos de Jong. "Math.js: An Extensive Math Library for JavaScript and Node.js." <https://mathjs.org/>.

5. **User Authentication and Security:**

- **OWASP Foundation:** Provided guidelines on securing web applications, particularly in the areas of user authentication and data protection.
- OWASP Foundation. "OWASP Top Ten Web Application Security Risks." <https://owasp.org/www-project-top-ten/>.

- **Bcrypt:** Information on using bcrypt for password hashing to enhance security.
 - Coda Hale. "bcrypt: A Blowfish-based password hashing function." <https://github.com/pyca/bcrypt>.
6. **Frontend API Integration:**
- **Fetch API Documentation:** Offered guidance on how to interact with server-side APIs using JavaScript's Fetch API.
 - Mozilla Developer Network (MDN). "Using Fetch." MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch.
7. **Educational and Mathematical Resources:**
- **Khan Academy:** Provided insights into how mathematical concepts can be taught and demonstrated online, influencing the educational aspects of the application.
 - Khan Academy. "Mathematics Courses." <https://www.khanacademy.org/math>.
 - **Mathematical Association of America (MAA):** A source for mathematical references and education-focused resources.
 - Mathematical Association of America. "MAA Mathematical Sciences Digital Library." <https://www.maa.org/>.

This bibliography reflects the sources used to inform the development of the Mathematical Calculation Application, covering various aspects from web development and design to backend API creation and mathematical functions.