TravelGo: A Cloud-Powered Real-Time Travel Booking Platform Using AWS

Project Description:

TravelGo is a full-stack, cloud-based travel booking platform designed to simplify the process of reserving buses, trains, flights, and hotels through a unified interface. Built using Flask as the backend framework, the application is deployed on Amazon EC2 and leverages DynamoDB for efficient storage of user data and bookings. TravelGo allows users to register, log in, search for transportation and accommodation options, and book their travel with ease. Once a booking is confirmed or cancelled, users receive real-time email notifications powered by AWS Simple Notification Service (SNS), keeping them informed throughout their journey.

The platform’s user-friendly interface supports dynamic seat selection for buses, hotel filtering based on preferences such as luxury or budget, and provides booking summaries along with centralized cancellation management. By combining cloud scalability, responsive design, and secure session handling, TravelGo delivers a seamless and real-time travel planning experience for users.

Scenarios:

Scenario 1: Hassle-Free Multi-Mode Travel Booking Experience

TravelGo offers users a unified platform to search and book buses, trains, flights, and hotels all in one place. For instance, a user planning a trip from Hyderabad to Bangalore can log in, select their preferred mode of transport, choose from available options, and proceed to booking. Flask manages the backend operations such as retrieving travel listings and processing user input in real-time. Hosted on AWS EC2, the platform remains responsive even during high-traffic hours like weekends or holiday seasons, allowing multiple users to browse and book without delay.

Scenario 2: Real-Time Booking Confirmation with AWS SNS

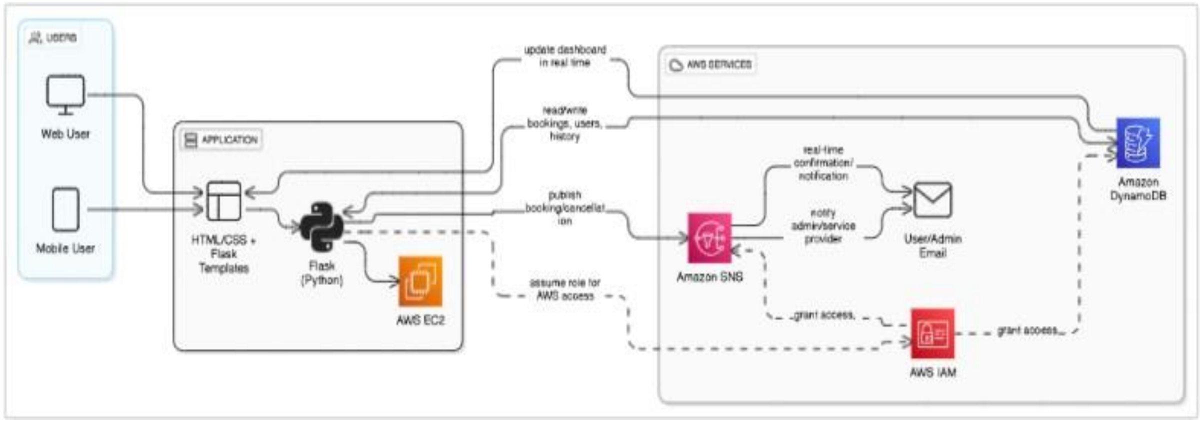
Once a booking is made—whether it’s a train ticket or a hotel stay—TravelGo uses AWS SNS to instantly notify the user. For example, after a student books a hotel in Chennai, SNS sends a real- time email notification confirming the booking with all the relevant details. This notification is triggered from the Flask backend after the booking is successfully recorded in DynamoDB. Additionally, SNS can alert admin or service providers, ensuring transparency and real-time updates on every transaction.

Scenario 3: Dynamic Dashboard with Personal Travel History

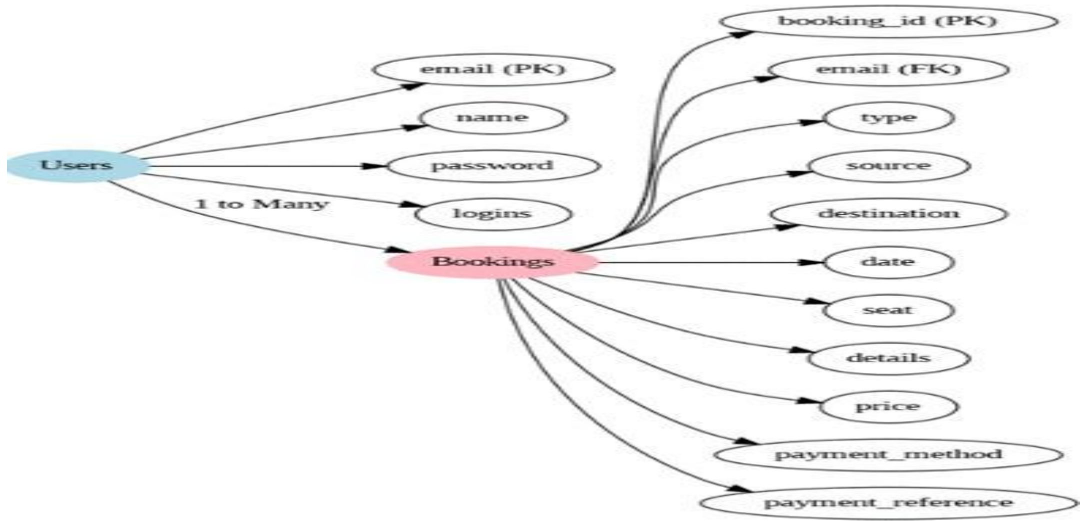
TravelGo features a dynamic user dashboard that displays all past and upcoming bookings for the logged-in user. For example, a user who has booked a flight and a hotel can view these bookings categorized by type, along with dates, price, and cancellation options. Flask fetches this data from AWS DynamoDB, which persistently stores all user bookings. The dashboard UI, powered by responsive HTML/CSS and Flask templates, ensures users can review or manage bookings anytime, from any device, with real-time updates and quick cancellation workflows supported.

AWS ARCHITECTURE:

This AWS-based architecture powers a scalable and secure web application using Amazon EC2 for hosting the backend, with a lightweight framework like Flask handling core logic. Application data is stored in Amazon DynamoDB, ensuring fast, reliable access, while user access is managed through AWS IAM for secure authentication and control. Real-time alerts and system notifications are enabled via Amazon SNS, enhancing communication and user engagement


USERS
Web User
Mobile User
APPLICATION
HTML/CSS + Flask Templates
Flask (Python)
AWS EC2
update dashboard in real time
read/write bookings, users, history
publish booking/cancellation
assume role for AWS access
AWS SERVICES
real-time confirmation/ notification
notify admin/service provider
User/Admin Email
Amazon SNS
grant access
AWS IAM
grant access
Amazon DynamoDB

Entity Relationship (ER)Diagram:



Pre-requisites:

- AWS Account Setup : https://docs.aws.amazon.com/accounts/latest/reference/getting-started.html
- AWS IAM (Identity and Access Management) : https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html
- AWS EC2 (Elastic Compute Cloud) : https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html
- AWS DynamoDB : https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.ht ml
- Amazon SNS : https://docs.aws.amazon.com/sns/latest/dg/welcome.html
- Git Documentation : https://git-scm.com/doc
- VS Code Installation : (download the VS Code using the below link or you can get that in Microsoft store) https://code.visualstudio.com/download

Project WorkFlow:

Milestone 1. Backend Development and Application Setup

- Develop the Backend Using Flask.
- Integrate AWS Services Using boto3.

Milestone 2. AWS Account Setup and Login

- Set up an AWS account if not already done.
- Log in to the AWS Management Console

Milestone 3. DynamoDB Database Creation and Setup

- Create a DynamoDB Table.
- Configure Attributes for User Data and Book Requests**.**

Milestone 4. SNS Notification Setup

- Create SNS topics for book request notifications.
- Subscribe users and library staff to SNS email notifications**.**

Milestone 5. IAM Role Setup

- Create IAM Role
- Attach Policies

Milestone 6. EC2 Instance Setup

- Launch an EC2 instance to host the Flask application.
- Configure security groups for HTTP, and SSH access.

Milestone 7. Deployment on EC2

- Upload Flask Files
- Run the Flask App

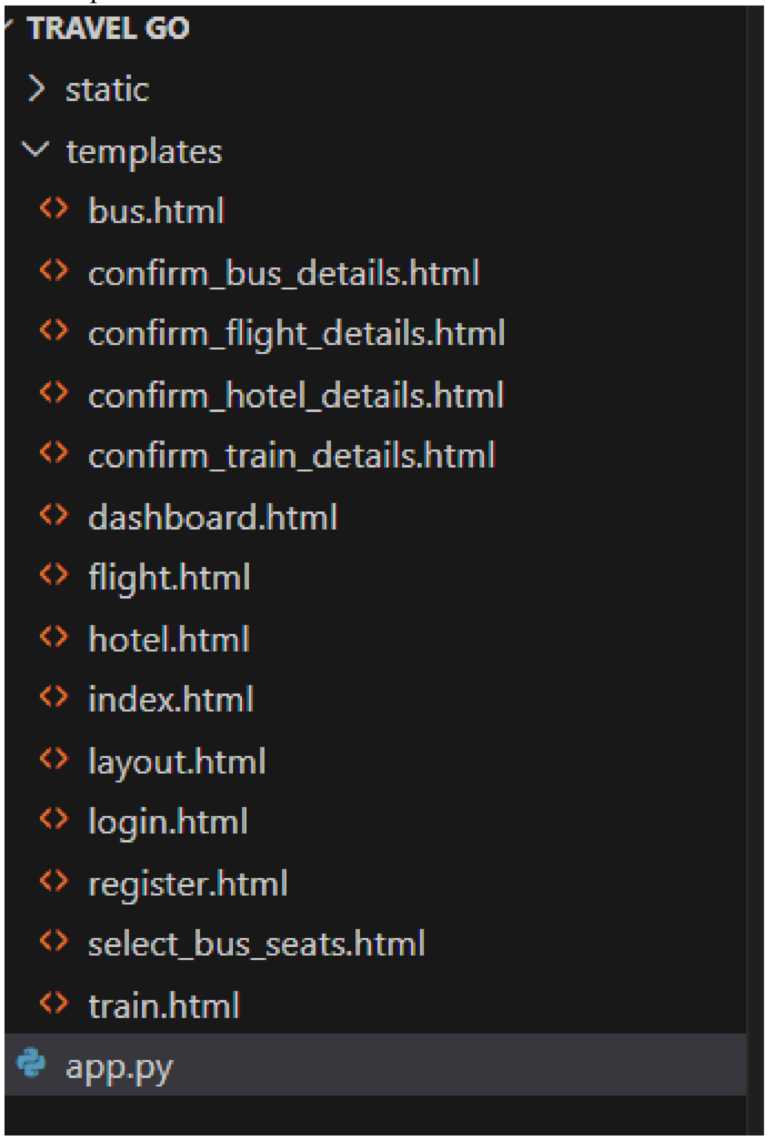
Milestone 8. Testing and Deployment

- Conduct functional testing to verify user registration, login, book requests, and notifications**.**

Milestone 1: Backend Development and Application SetUp

- **Activity 1.1:** Develop the Backend Using Flask

1. File Explorer Structure

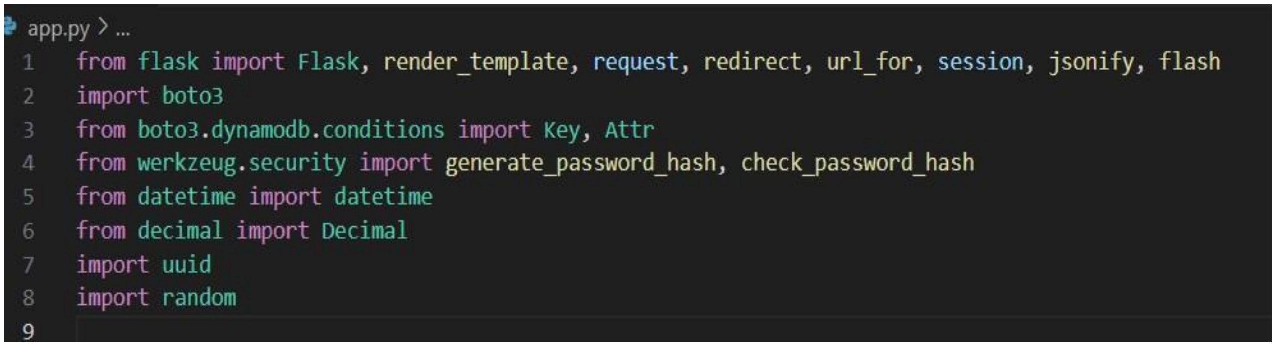


```
TRAVEL GO
  > static
  v templates
      bus.html
      confirm_bus_details.html
      confirm_flight_details.html
      confirm_hotel_details.html
      confirm_train_details.html
      dashboard.html
      flight.html
      hotel.html
      index.html
      layout.html
      login.html
      register.html
      select_bus_seats.html
      train.html
    app.py
```

Description: Organize the project with HTML templates for each feature (e.g., login, wishlist, quiz, checkout) under the templates folder and manage backend logic in app.py .

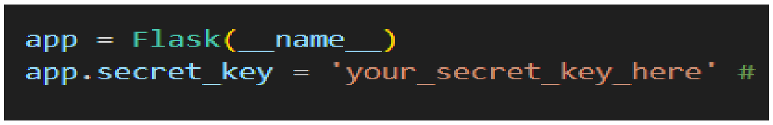
Description of the code :

? Flask App Initialization



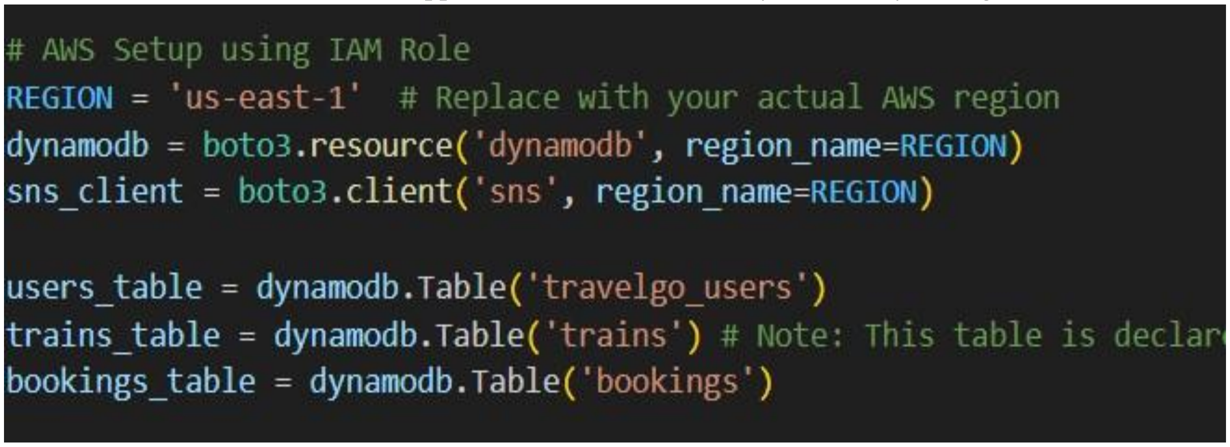
```
app.py > ...
from flask import Flask, render_template, request, redirect, url_for, session, jsonify, flash
import boto3
from boto3.dynamodb.conditions import Key, Attr
from werkzeug.security import generate_password_hash, check_password_hash
from datetime import datetime
from decimal import Decimal
import uuid
import random

```

- Import essential Flask modules for web handling, Boto3 for AWS integration, Werkzeug for password hashing, and datetime for timestamp management.



```
app = Flask(__name__)
app.secret_key = 'your_secret_key_here' #
```

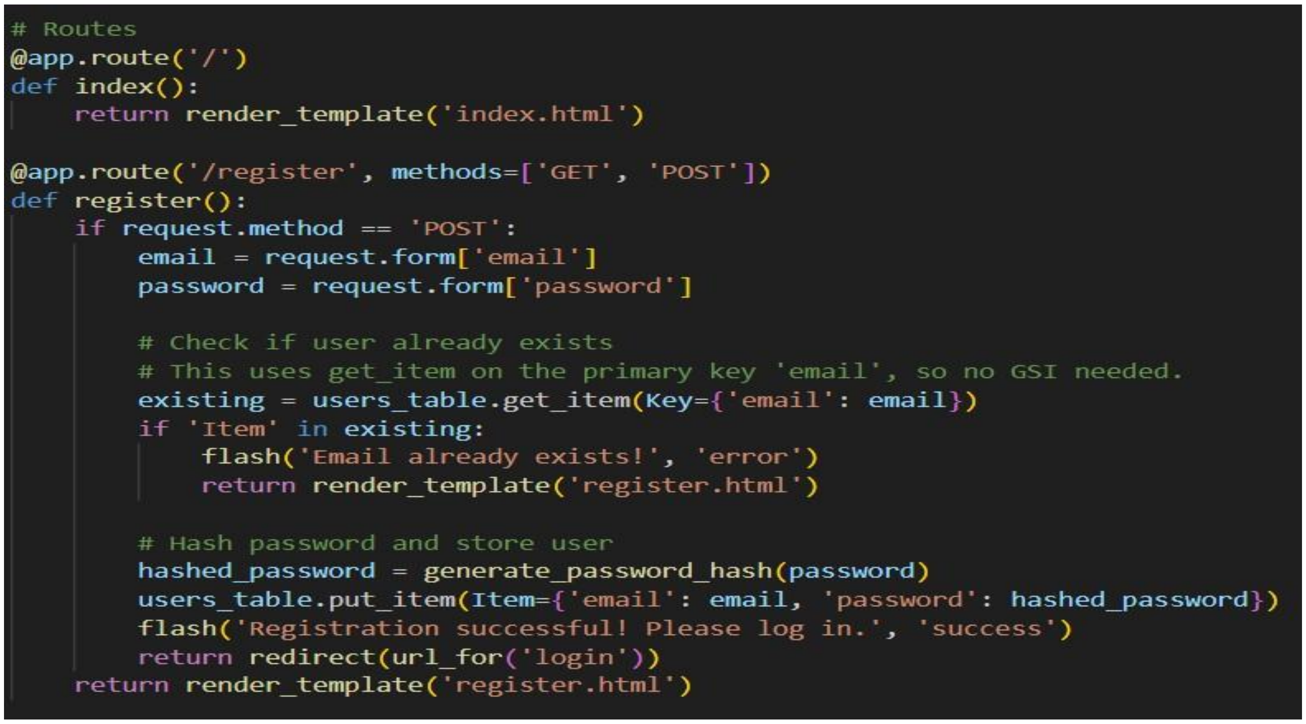
- Initialize the Flask application and set a secret key to securely manage user sessions and form data.



```
# AWS Setup using IAM Role
REGION = 'us-east-1'  # Replace with your actual AWS region
dynamodb = boto3.resource('dynamodb', region_name=REGION)
sns_client = boto3.client('sns', region_name=REGION)

users_table = dynamodb.Table('travelgo_users')
trains_table = dynamodb.Table('trains') # Note: This table is declar
bookings_table = dynamodb.Table('bookings')
```

- Connect to DynamoDB using Boto3 and define references to the UserTable and WishlistTable for user and wishlist data operations.

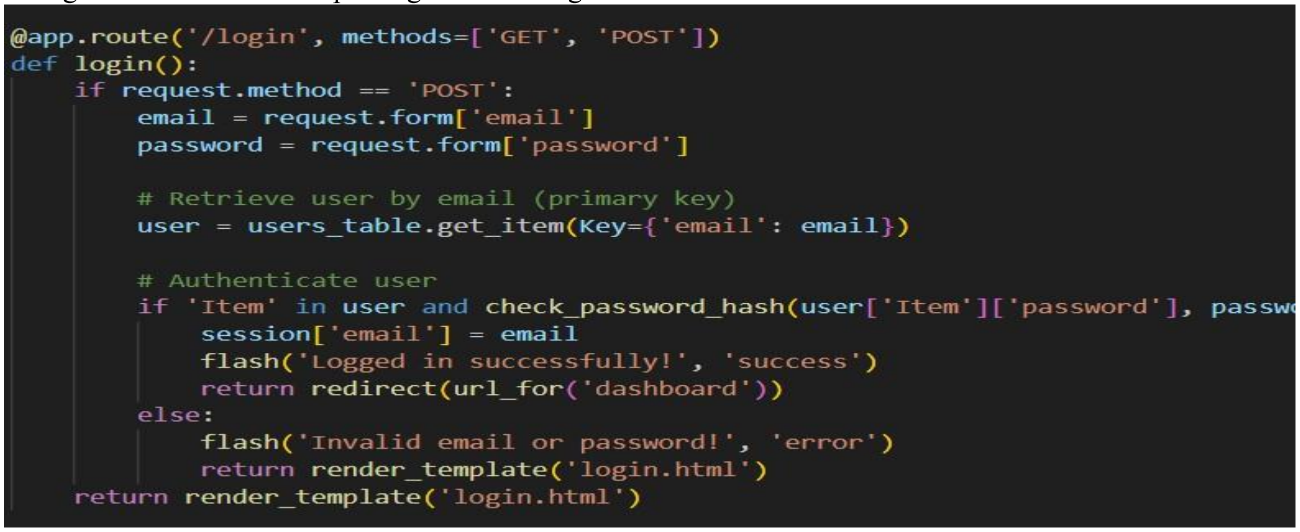
Routes for Core Functionalities:



```python
# Routes
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']

        # Check if user already exists
        # This uses get_item on the primary key 'email', so no GSI needed.
        existing = users_table.get_item(Key={'email': email})
        if 'Item' in existing:
            flash('Email already exists!', 'error')
            return render_template('register.html')

        # Hash password and store user
        hashed_password = generate_password_hash(password)
        users_table.put_item(Item={'email': email, 'password': hashed_password})
        flash('Registration successful! Please log in.', 'success')
        return redirect(url_for('login'))
    return render_template('register.html')
```

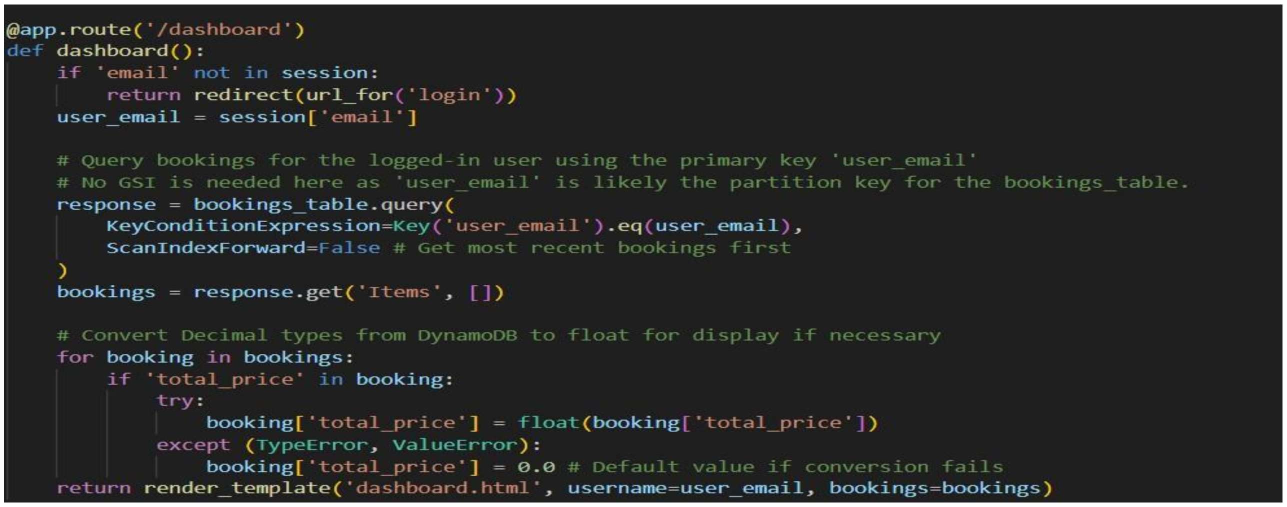
Create the home and registration routes, where the registration route securely hashes user passwords and stores user data in DynamoDB upon form submission.

Login route: Implement the user login route to validate credentials using DynamoDB and securely manage session data while updating the user's login count.

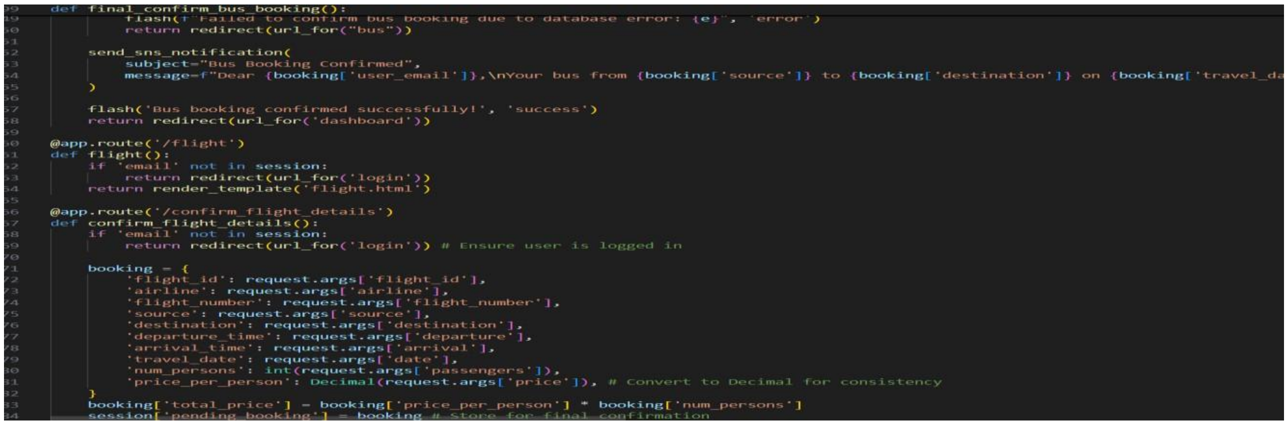


```python
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']

        # Retrieve user by email (primary key)
        user = users_table.get_item(Key={'email': email})

        # Authenticate user
        if 'Item' in user and check_password_hash(user['Item']['password'], passw
            session['email'] = email
            flash('Logged in successfully!', 'success')
            return redirect(url_for('dashboard'))
        else:
            flash('Invalid email or password!', 'error')
            return render_template('login.html')
    return render_template('login.html')
```

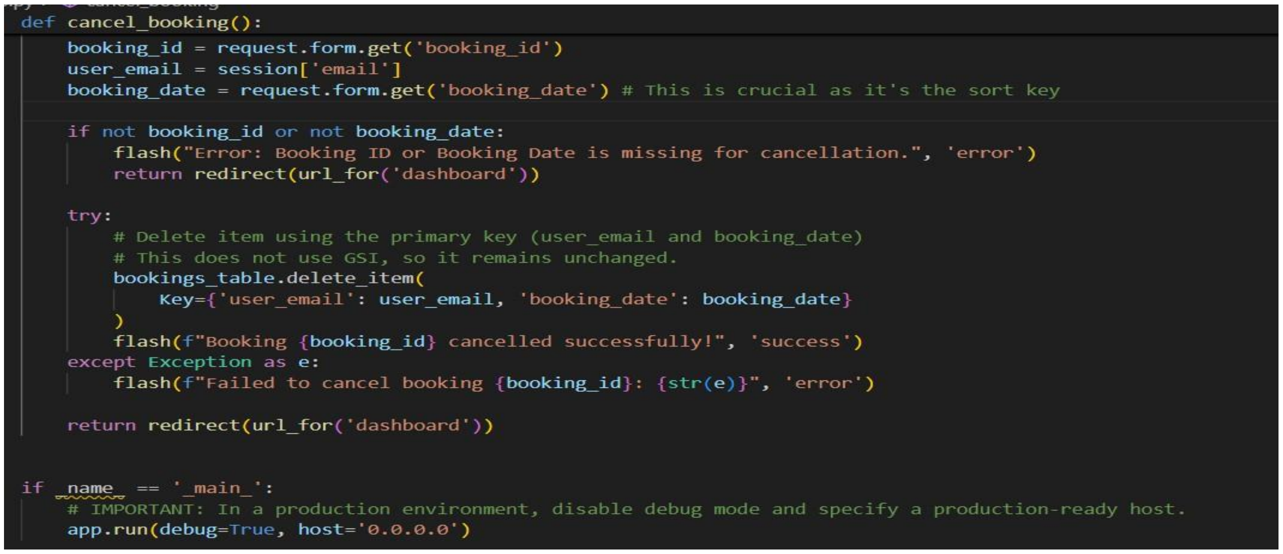
- **Dashboard Route**: Secure the user dashboard and implement a route to add items to the wishlist, storing them in DynamoDB with item details and a timestamp.



```python
@app.route('/dashboard')
def dashboard():
    if 'email' not in session:
        return redirect(url_for('login'))
    user_email = session['email']

    # Query bookings for the logged-in user using the primary key 'user_email'
    # No GSI is needed here as 'user_email' is likely the partition key for the bookings_table.
    response = bookings_table.query(
        KeyConditionExpression=Key('user_email').eq(user_email),
        ScanIndexForward=False # Get most recent bookings first
    )
    bookings = response.get('Items', [])

    # Convert Decimal types from DynamoDB to float for display if necessary
    for booking in bookings:
        if 'total_price' in booking:
            try:
                booking['total_price'] = float(booking['total_price'])
            except (TypeError, ValueError):
                booking['total_price'] = 0.0 # Default value if conversion fails
    return render_template('dashboard.html', username=user_email, bookings=bookings)
```

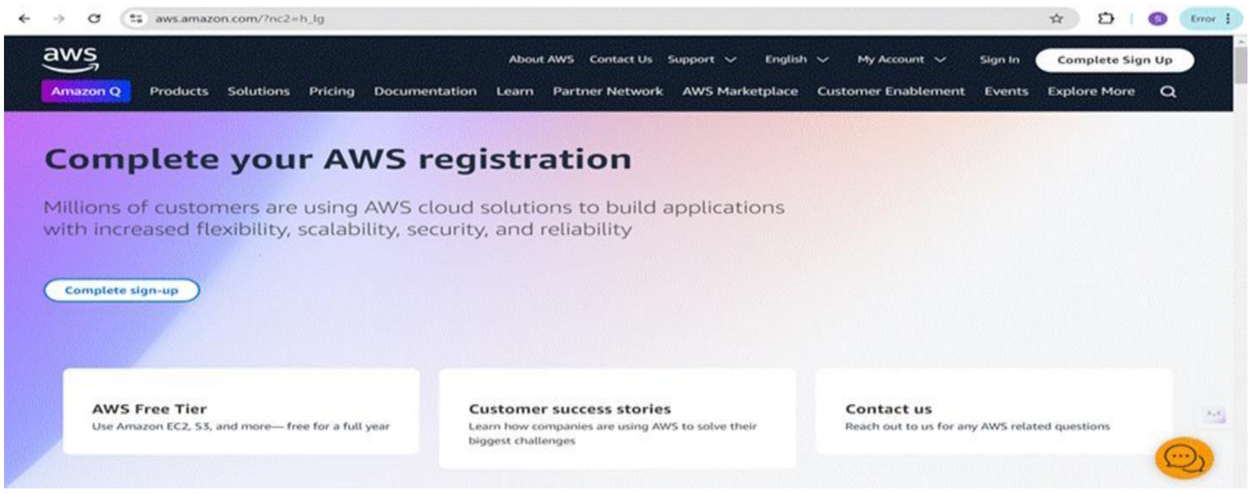


```python
def final_confirm_bus_booking():
        flash(f"Failed to confirm bus booking due to database error: {e}", 'error')
        return redirect(url_for("bus"))

    send_sns_notification(
        subject="Bus Booking Confirmed",
        message=f"Dear {booking['user_email']},\nYour bus from {booking['source']} to {booking['destination']} on {booking['travel_da
    )

    flash('Bus booking confirmed successfully!', 'success')
    return redirect(url_for('dashboard'))

@app.route('/flight')
def flight():
    if 'email' not in session:
        return redirect(url_for('login'))
    return render_template('flight.html')

@app.route('/confirm_flight_details')
def confirm_flight_details():
    if 'email' not in session:
        return redirect(url_for('login')) # Ensure user is logged in

    booking = {
        'flight_id': request.args['flight_id'],
        'airline': request.args['airline'],
        'flight_number': request.args['flight_number'],
        'source': request.args['source'],
        'destination': request.args['destination'],
        'departure_time': request.args['departure'],
        'arrival_time': request.args['arrival'],
        'travel_date': request.args['date'],
        'num_persons': int(request.args['passengers']),
        'price_per_person': Decimal(request.args['price']), # Convert to Decimal for consistency
    }
    booking['total_price'] = booking['price_per_person'] * booking['num_persons']
    session['pending_booking'] = booking # Store for final confirmation
```



```python
def cancel_booking():
    booking_id = request.form.get('booking_id')
    user_email = session['email']
    booking_date = request.form.get('booking_date') # This is crucial as it's the sort key

    if not booking_id or not booking_date:
        flash("Error: Booking ID or Booking Date is missing for cancellation.", 'error')
        return redirect(url_for('dashboard'))

    try:
        # Delete item using the primary key (user_email and booking_date)
        # This does not use GSI, so it remains unchanged.
        bookings_table.delete_item(
            Key={'user_email': user_email, 'booking_date': booking_date}
        )
        flash(f"Booking {booking_id} cancelled successfully!", 'success')
    except Exception as e:
        flash(f"Failed to cancel booking {booking_id}: {str(e)}", 'error')

    return redirect(url_for('dashboard'))


if __name__ == '__main__':
    # IMPORTANT: In a production environment, disable debug mode and specify a production-ready host.
    app.run(debug=True, host='0.0.0.0')
```

- **Milestone 2: AWS Account Setup and Login.**
    - **Activity 2.1:** Set up an AWS account if not already done.
        - Sign up for an AWS account and configure billing settings.



Complete your AWS registration

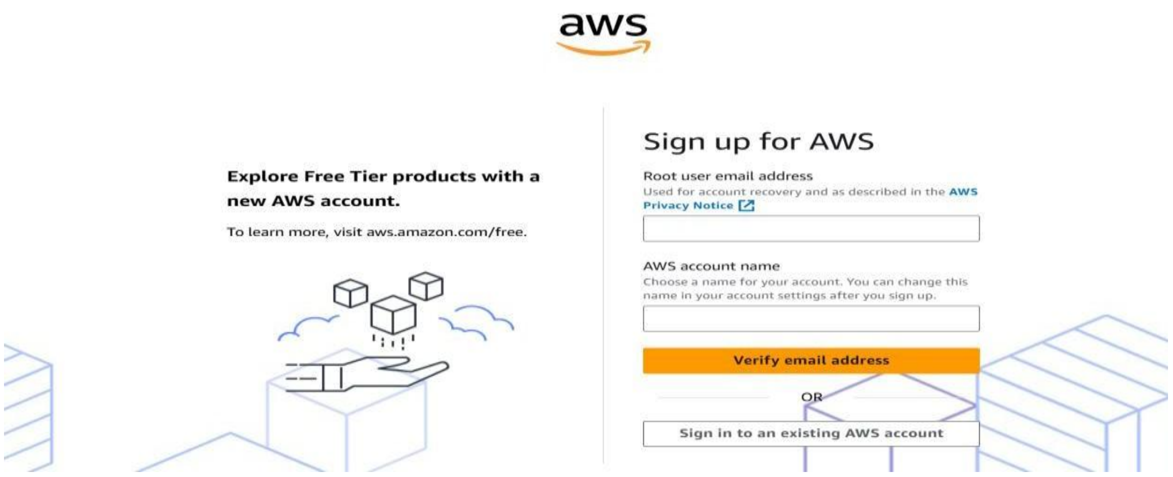
Millions of customers are using AWS cloud solutions to build applications with increased flexibility, scalability, security, and reliability

AWS Free Tier: Use Amazon EC2, S3, and more— free for a full year

Customer success stories: Learn how companies are using AWS to solve their biggest challenges

Contact us: Reach out to us for any AWS related questions

- Click on the "Create an AWS Account" button.
- Follow the prompts to enter your email address and choose a password.
- Provide the required account information, including your name, address, and phone number.
- Enter your payment information. (Note: While AWS offers a free tier, a credit card or debit card is required for verification.)
- Complete the identity verification process.
- Choose a support plan (the basic plan is free and sufficient for starting).
- Once verified, you can sign in to your new AWS accounts**.**


Sign up for AWS

Explore Free Tier products with a new AWS account.

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

Root user email address

Used for account recovery and as described in the AWS Privacy Notice

AWS account name

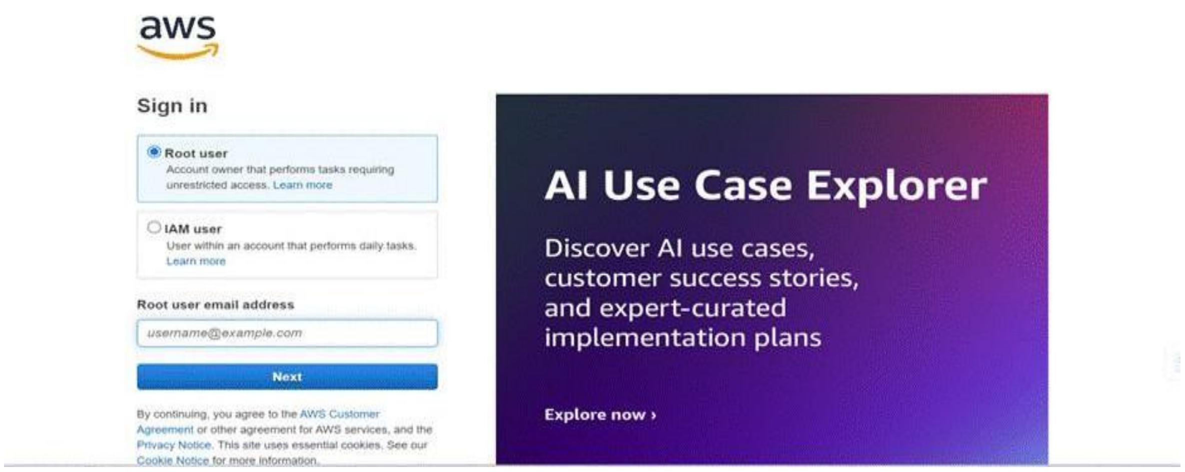
Choose a name for your account. You can change this name in your account settings after you sign up.

Verify email address

OR

Sign in to an existing AWS account

- **Activity2.2 :** Log in to the AWS Management Console
  - After setting up your account, log in to the AWS Management Console**.**


Sign in

Root user
Account owner that performs tasks requiring unrestricted access. Learn more

IAM user
User within an account that performs daily tasks. Learn more

Root user email address

username@example.com

Next

By continuing, you agree to the AWS Customer Agreement or other agreement for AWS services, and the Privacy Notice. This site uses essential cookies. See our Cookie Notice for more information.

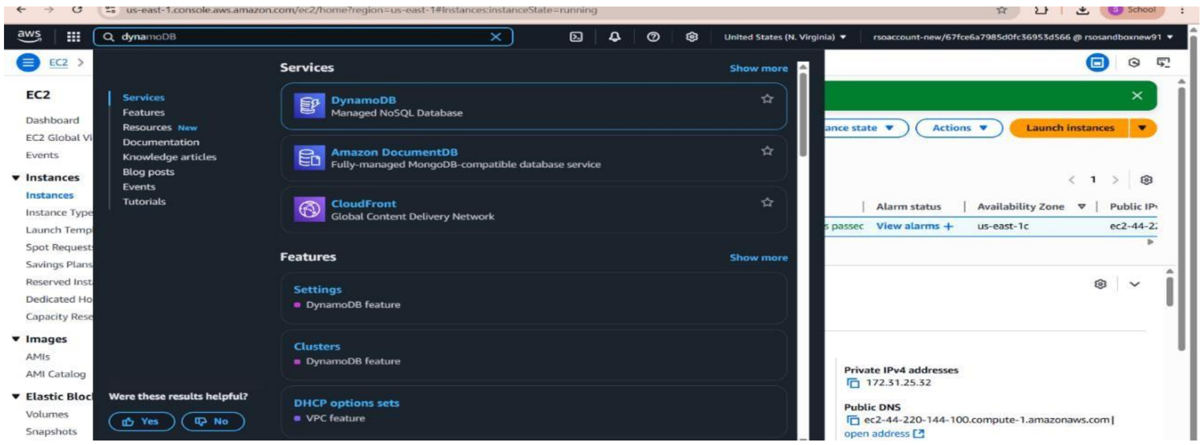
AI Use Case Explorer

Discover AI use cases, customer success stories, and expert-curated implementation plans

Explore now ›

Milestone 3: DynamoDB Database Creation and Setup

- **Activity 3.1:**Navigate to the DynamoDB
  - In the AWS Console, navigate to DynamoDB and click on create tables**.**


us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:instanceState=running

dynamoDB

United States (N. Virginia) ▾ rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91 ▾

Services — Show more

DynamoDB
Managed NoSQL Database

Amazon DocumentDB
Fully-managed MongoDB-compatible database service

CloudFront
Global Content Delivery Network

Features — Show more

Settings
DynamoDB feature

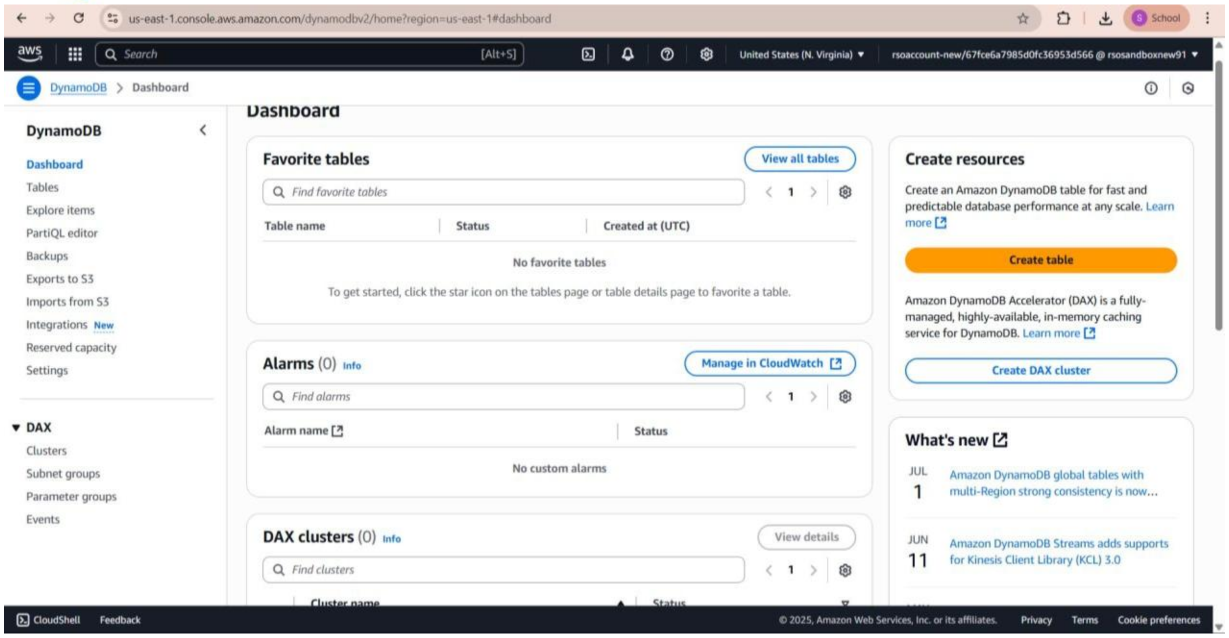
Clusters
DynamoDB feature

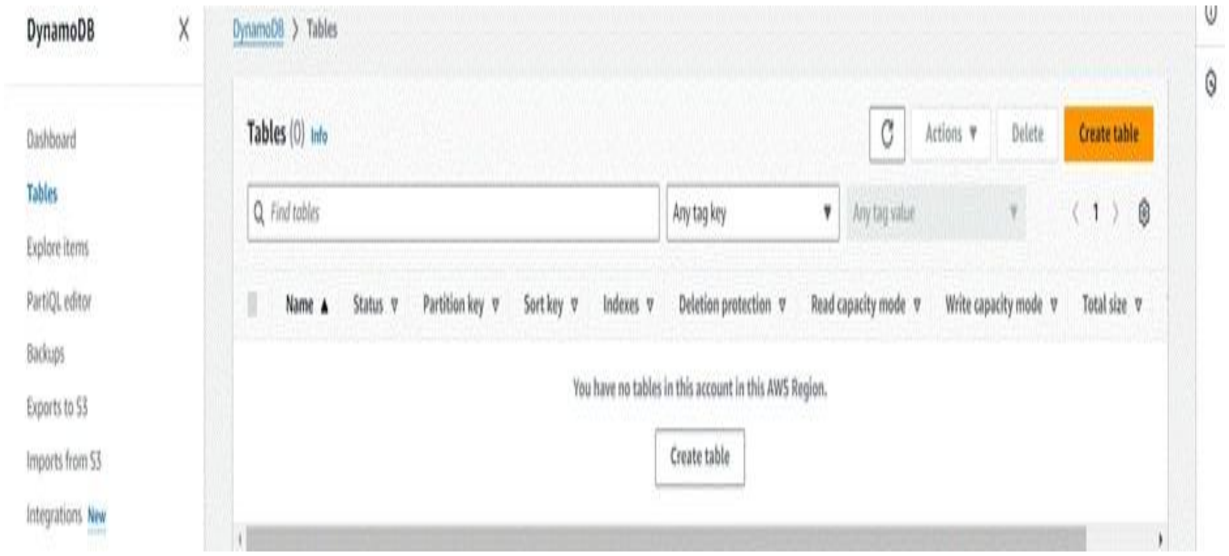
DHCP options sets
VPC feature

Were these results helpful? Yes No

Private IPv4 addresses
172.31.25.32

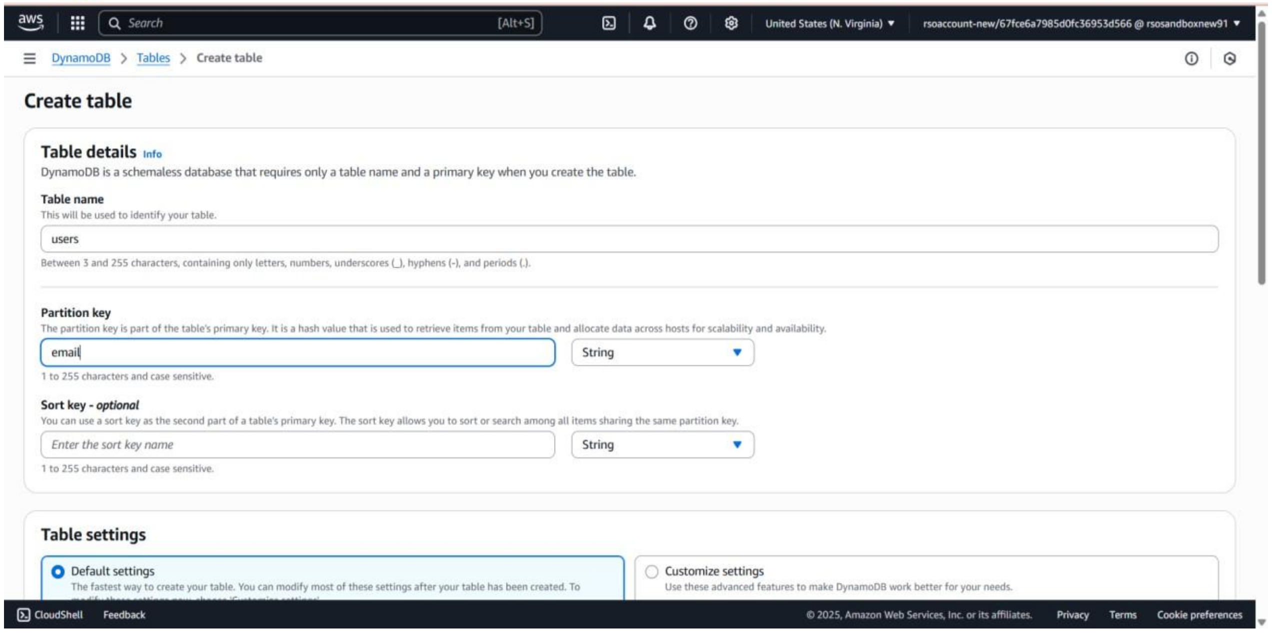
Public DNS
ec2-44-220-144-100.compute-1.amazonaws.com
open address


us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#dashboard
DynamoDB > Dashboard
DynamoDB
Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations New
Reserved capacity
Settings
DAX
Clusters
Subnet groups
Parameter groups
Events
Dashboard
Favorite tables
View all tables
Find favorite tables
Table name
Status
Created at (UTC)
No favorite tables
To get started, click the star icon on the tables page or table details page to favorite a table.
Alarms (0) Info
Manage in CloudWatch
Find alarms
Alarm name
Status
No custom alarms
DAX clusters (0) Info
View details
Find clusters
Create resources
Create an Amazon DynamoDB table for fast and predictable database performance at any scale. Learn more
Create table
Amazon DynamoDB Accelerator (DAX) is a fully-managed, highly-available, in-memory caching service for DynamoDB. Learn more
Create DAX cluster
What's new
JUL 1 Amazon DynamoDB global tables with multi-Region strong consistency is now...
JUN 11 Amazon DynamoDB Streams adds supports for Kinesis Client Library (KCL) 3.0
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



Activity 3.2: Create a DynamoDB table for storing registration details and book requests.

- Create Users table with partition key “Email ” with type String and click on create tables.



aws | Search [Alt+S] | United States (N. Virginia) ▼ | rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91 ▼

DynamoDB > Tables > Create table

Create table

Table details Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.
users
Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.
email | String
1 to 255 characters and case sensitive.

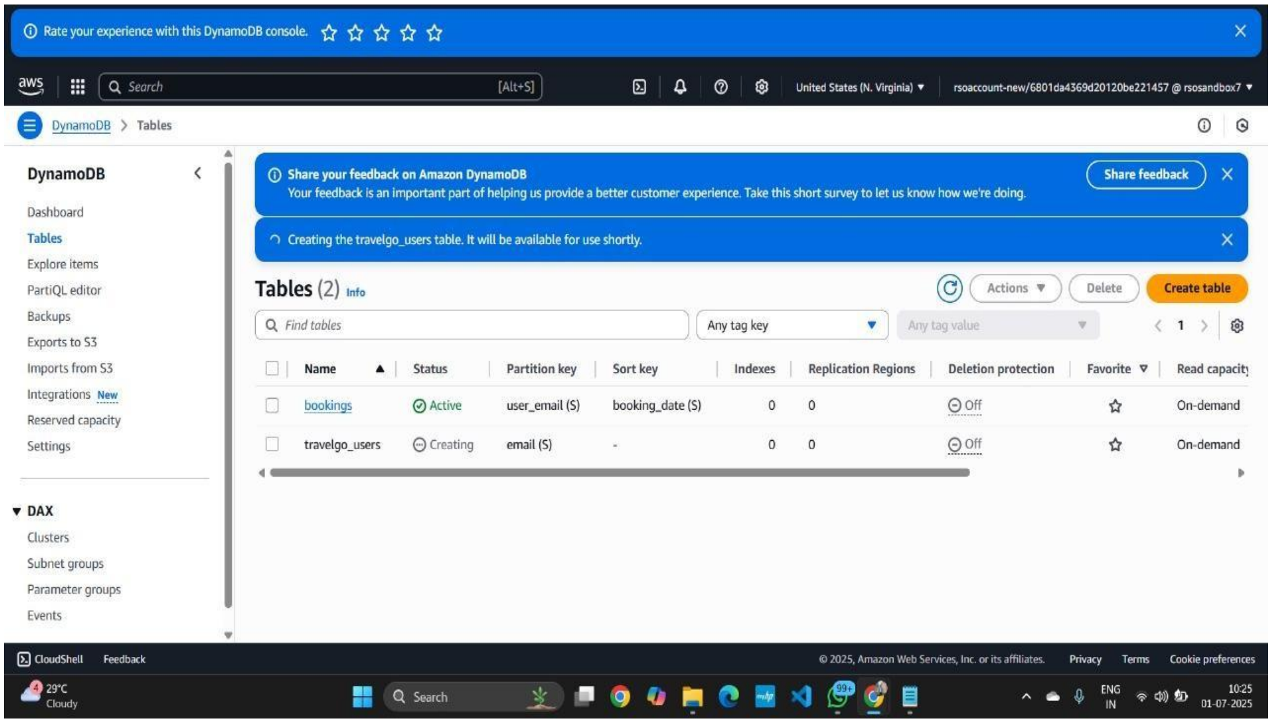
Sort key - *optional*
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.
Enter the sort key name | String
1 to 255 characters and case sensitive.

Table settings

● Default settings
The fastest way to create your table. You can modify most of these settings after your table has been created. To

○ Customize settings
Use these advanced features to make DynamoDB work better for your needs.

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



Rate your experience with this DynamoDB console. ☆ ☆ ☆ ☆ ☆

aws | Search [Alt+S] | United States (N. Virginia) ▼ | rsoaccount-new/6801da4369d20120be221457 @ rsosandbox7 ▼

DynamoDB > Tables

Share your feedback on Amazon DynamoDB
Your feedback is an important part of helping us provide a better customer experience. Take this short survey to let us know how we're doing. Share feedback

Creating the travelgo_users table. It will be available for use shortly.

Tables (2) Info | Actions | Delete | Create table

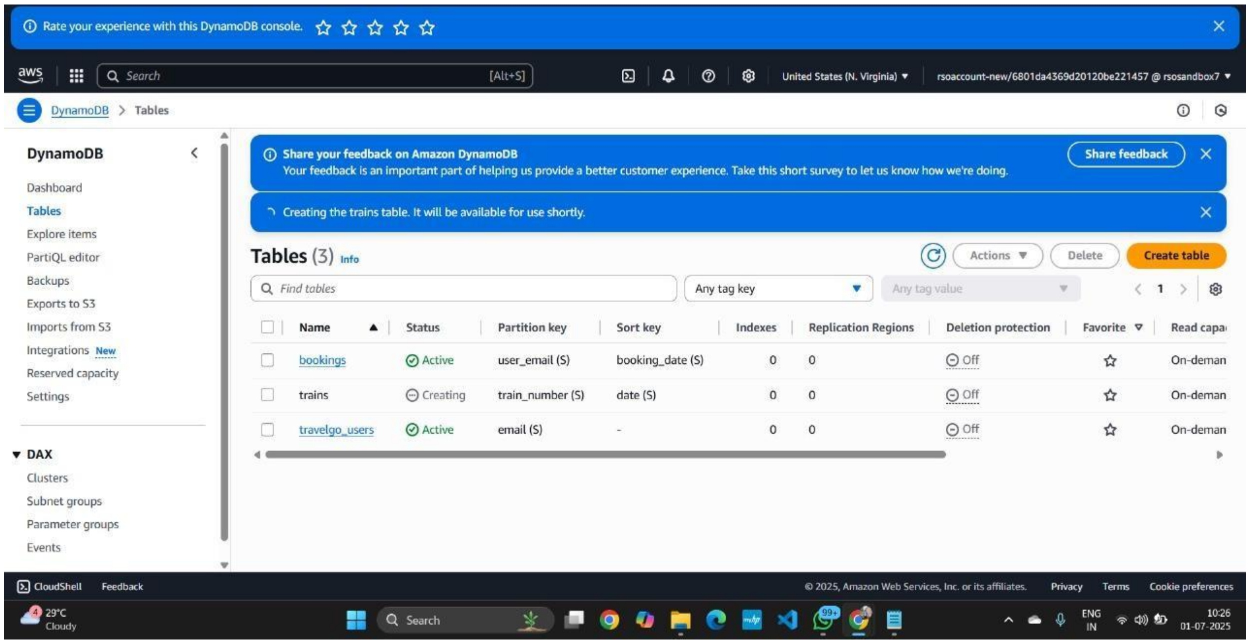
Find tables | Any tag key | Any tag value | 1

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity
bookings	Active	user_email (S)	booking_date (S)	0	0	Off	☆	On-demand
travelgo_users	Creating	email (S)	-	0	0	Off	☆	On-demand

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

29°C Cloudy | Search | ENG IN | 10:25 01-07-2025

- Follow the same steps to create a Bookings for storing booking records with email as the partition key and booking_id as the sort key.



Rate your experience with this DynamoDB console.

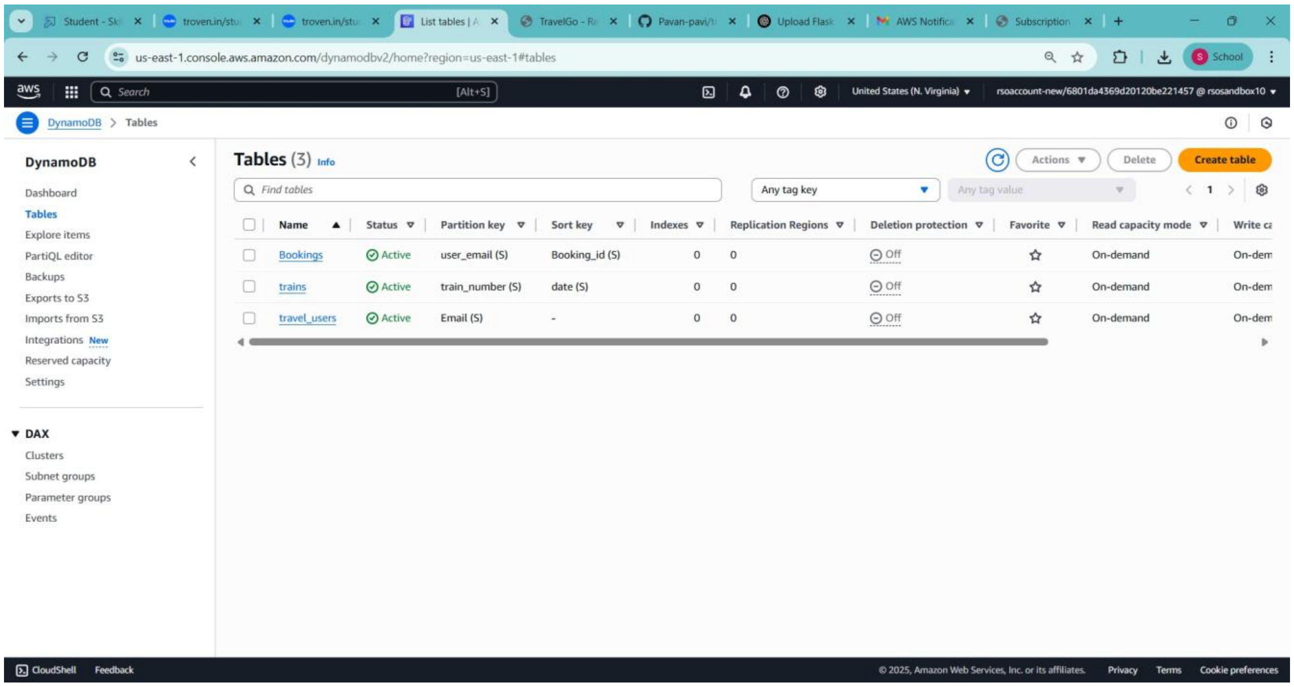
DynamoDB > Tables

Share your feedback on Amazon DynamoDB
Your feedback is an important part of helping us provide a better customer experience. Take this short survey to let us know how we're doing.

Creating the trains table. It will be available for use shortly.

Tables (3)

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capa
bookings	Active	user_email (S)	booking_date (S)	0	0	Off	☆	On-deman
trains	Creating	train_number (S)	date (S)	0	0	Off	☆	On-deman
travelgo_users	Active	email (S)	-	0	0	Off	☆	On-deman



us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#tables

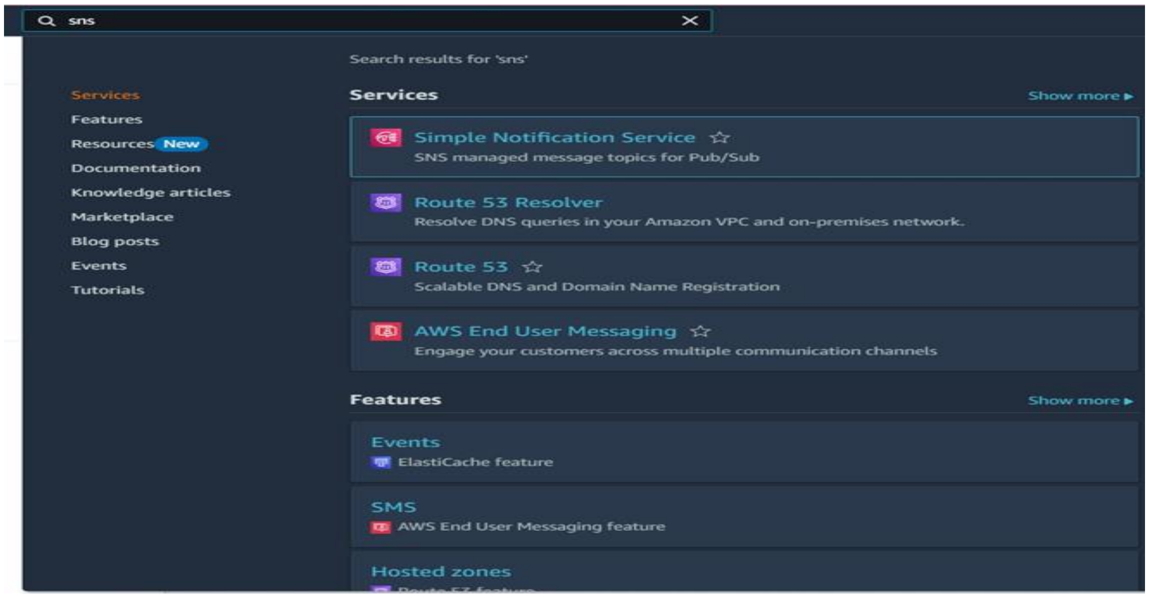
DynamoDB > Tables

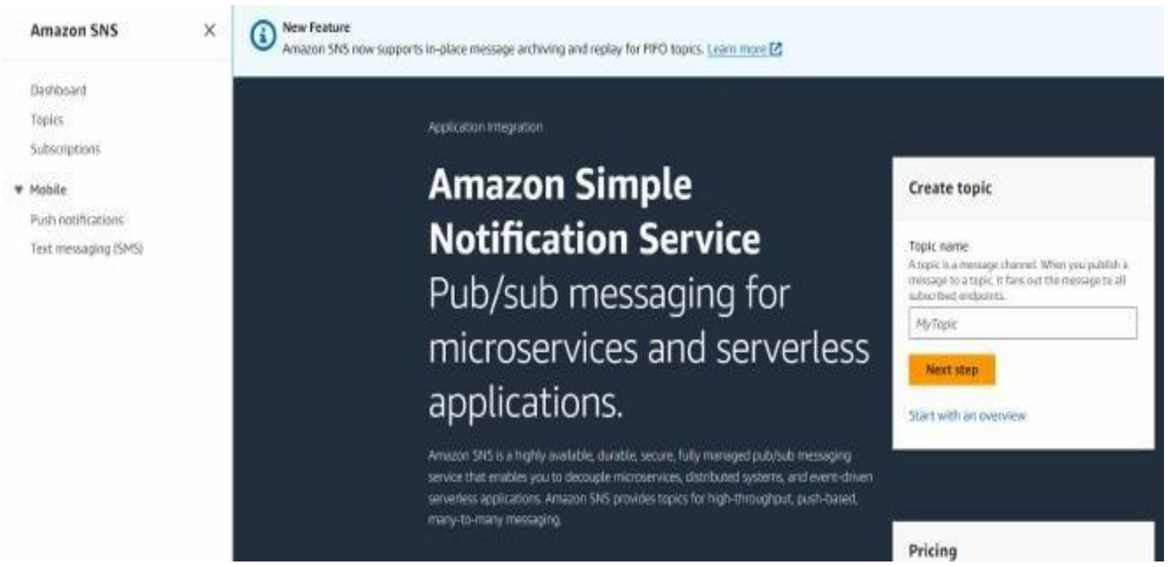
Tables (3)

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity mode	Write ca
Bookings	Active	user_email (S)	Booking_id (S)	0	0	Off	☆	On-demand	On-dem
trains	Active	train_number (S)	date (S)	0	0	Off	☆	On-demand	On-dem
travel_users	Active	Email (S)	-	0	0	Off	☆	On-demand	On-dem

Milestone 4. SNS Notification Setup

- **Activiy4.1:** Create SNS topics for book request notifications**.**


sns
Search results for 'sns'
Services
Features
Resources New
Documentation
Knowledge articles
Marketplace
Blog posts
Events
Tutorials
Services
Show more ▸
Simple Notification Service
SNS managed message topics for Pub/Sub
Route 53 Resolver
Resolve DNS queries in your Amazon VPC and on-premises network.
Route 53
Scalable DNS and Domain Name Registration
AWS End User Messaging
Engage your customers across multiple communication channels
Features
Show more ▸
Events
ElastiCache feature
SMS
AWS End User Messaging feature
Hosted zones



Amazon SNS now supports in-place message archiving and replay for FIFO topics. Learn more

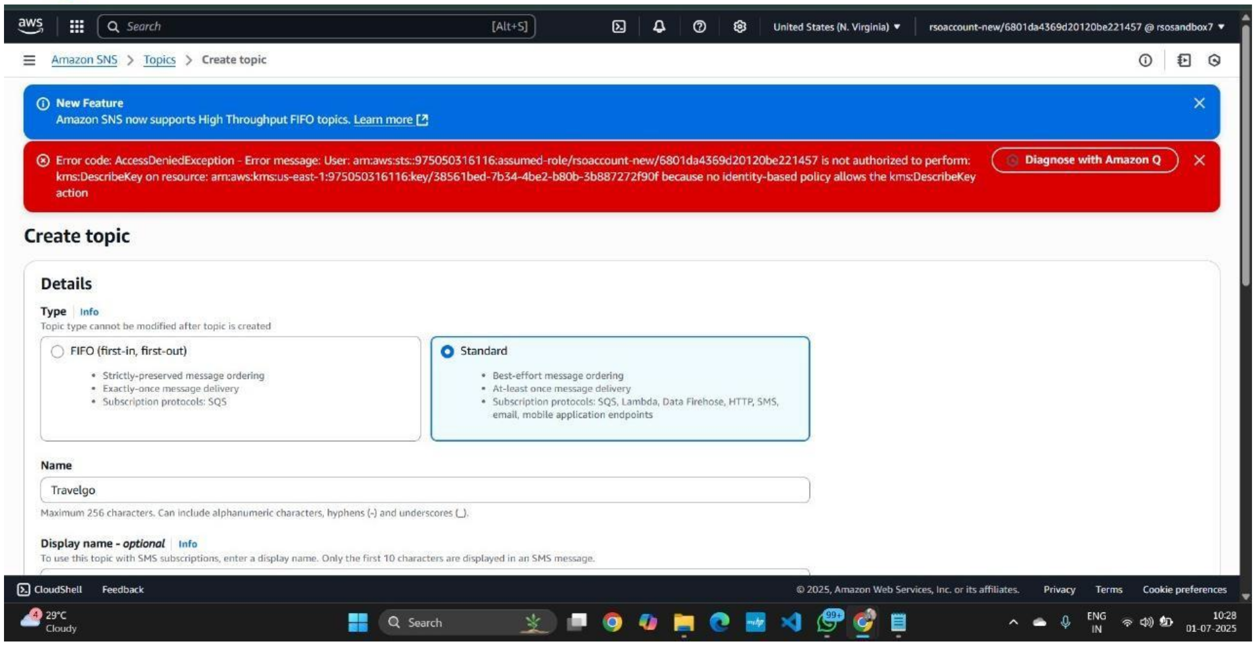
Amazon Simple Notification Service

Pub/sub messaging for microservices and serverless applications.

Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and event-driven serverless applications. Amazon SNS provides topics for high-throughput, push-based, many-to-many messaging.

A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.

- Click on Create Topic and choose a name for the topic



Amazon SNS > Topics > Create topic

New Feature
Amazon SNS now supports High Throughput FIFO topics. Learn more

Error code: AccessDeniedException - Error message: User: arn:aws:sts::975050316116:assumed-role/rsoaccount-new/6801da4369d20120be221457 is not authorized to perform: kms:DescribeKey on resource: arn:aws:kms:us-east-1:975050316116:key/38561bed-7b34-4be2-b80b-3b887272f90f because no identity-based policy allows the kms:DescribeKey action

Create topic

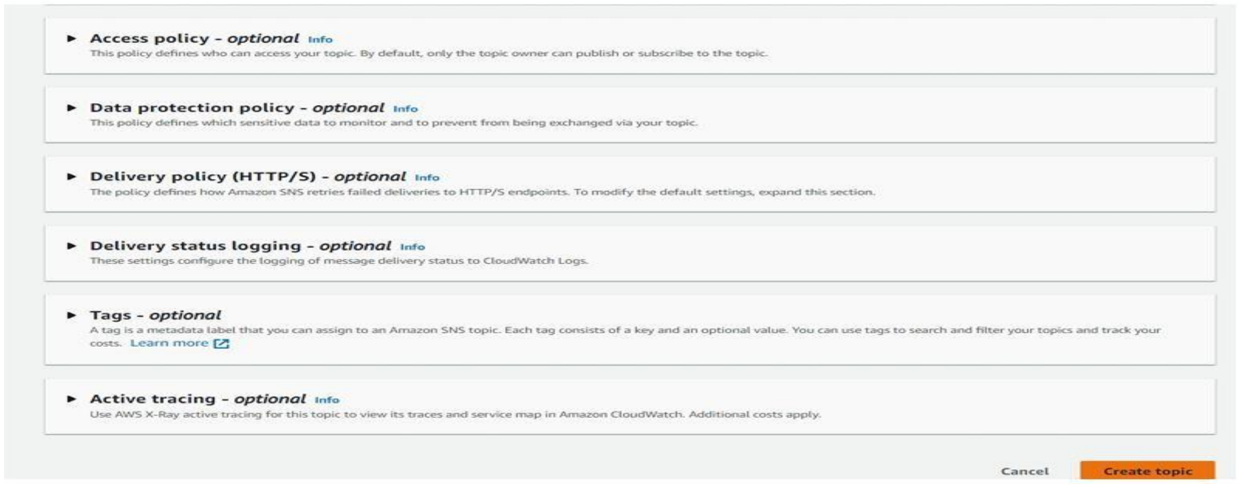
Details

Type Info
Topic type cannot be modified after topic is created

- FIFO (first-in, first-out)
  - Strictly-preserved message ordering
  - Exactly-once message delivery
  - Subscription protocols: SQS
- [✓ Standard]
  - Best-effort message ordering
  - At-least once message delivery
  - Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SMS, email, mobile application endpoints

Name
Travelgo
Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - *optional* Info
To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.



► **Access policy - *optional*** Info
This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic.

► **Data protection policy - *optional*** Info
This policy defines which sensitive data to monitor and to prevent from being exchanged via your topic.

► **Delivery policy (HTTP/S) - *optional*** Info
The policy defines how Amazon SNS retries failed deliveries to HTTP/S endpoints. To modify the default settings, expand this section.

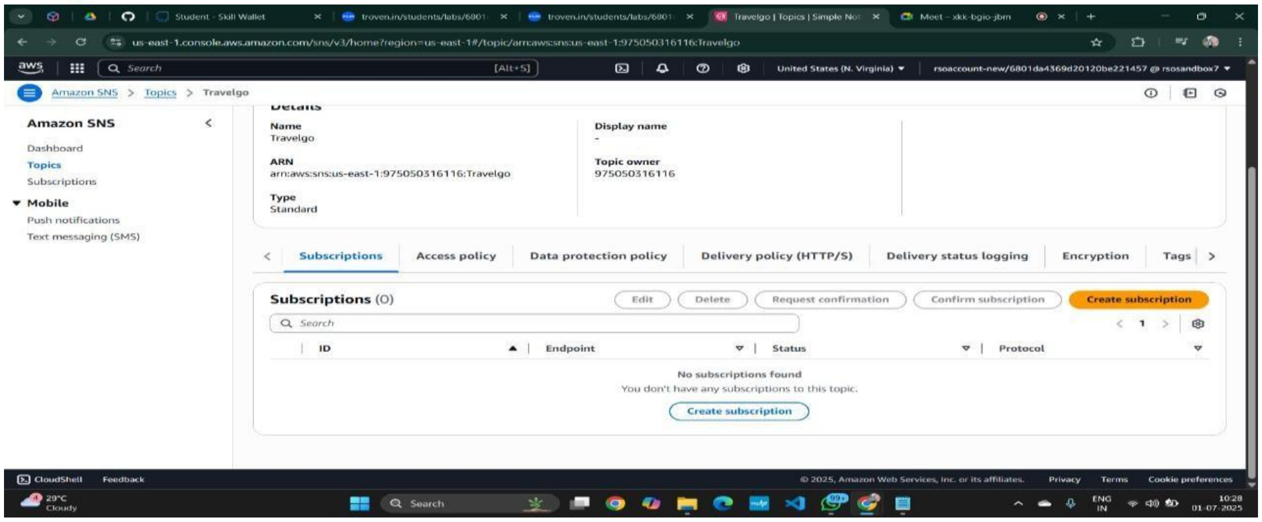
► **Delivery status logging - *optional*** Info
These settings configure the logging of message delivery status to CloudWatch Logs.

► **Tags - *optional***
A tag is a metadata label that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track your costs. Learn more

► **Active tracing - *optional*** Info
Use AWS X-Ray active tracing for this topic to view its traces and service map in Amazon CloudWatch. Additional costs apply.

Cancel | Create topic

- Configure the SNS topic and note down the Topic ARN



us-east-1.console.aws.amazon.com/sns/v3/home?region=us-east-1#/topic/arn:aws:sns:us-east-1:975050316116:Travelgo

Amazon SNS > Topics > Travelgo

Details

Name	Display name
Travelgo	-
ARN	Topic owner
arn:aws:sns:us-east-1:975050316116:Travelgo	975050316116
Type	
Standard	

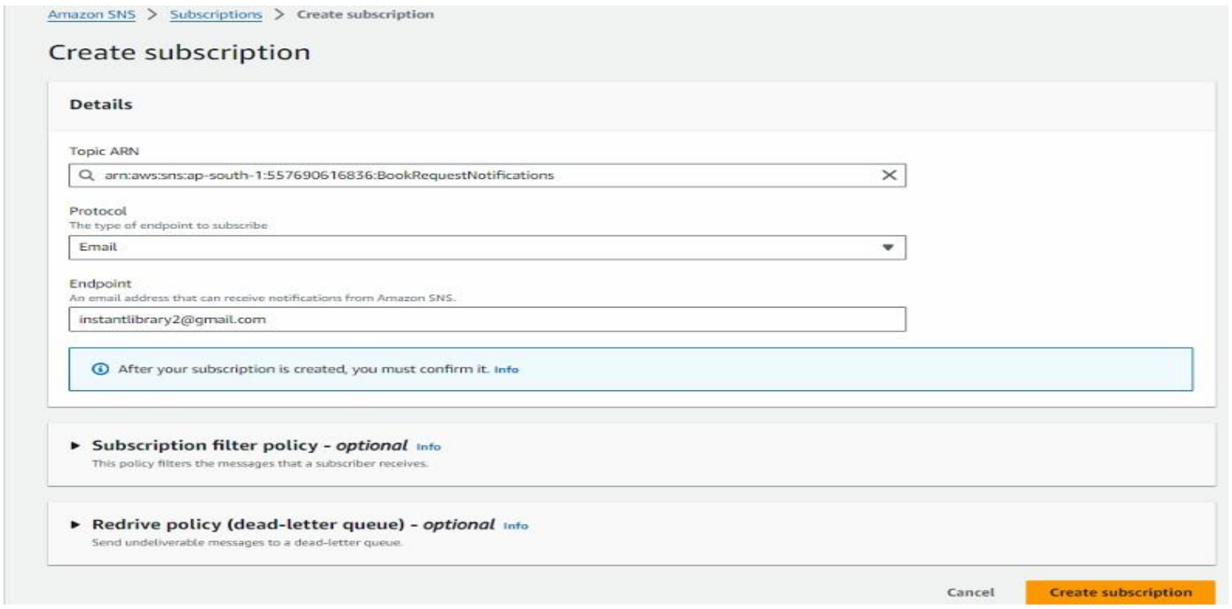
Subscriptions | Access policy | Data protection policy | Delivery policy (HTTP/S) | Delivery status logging | Encryption | Tags

Subscriptions (0)

ID	Endpoint	Status	Protocol

No subscriptions found
You don't have any subscriptions to this topic.

- Subscribe users and library staff to SNS email notifications.



Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN: arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications

Protocol
The type of endpoint to subscribe
Email

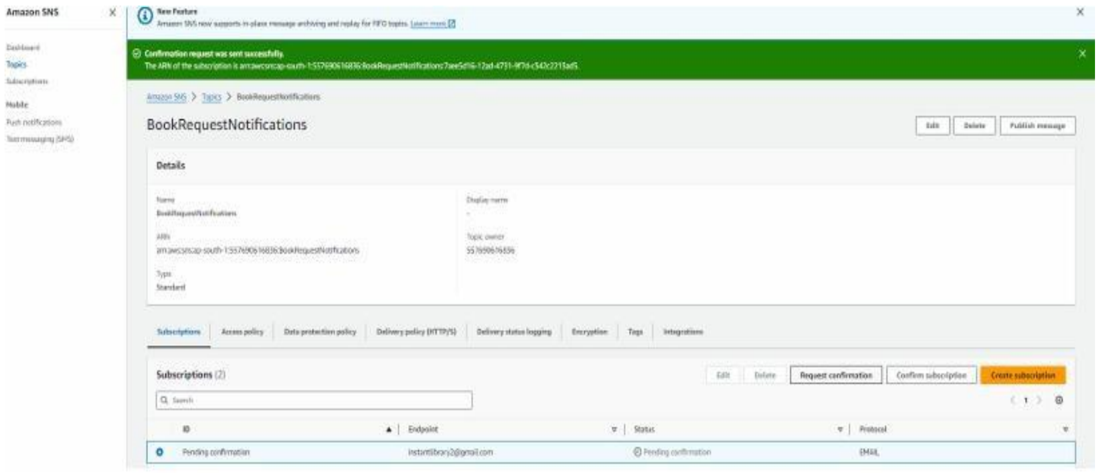
Endpoint
An email address that can receive notifications from Amazon SNS.
instantlibrary2@gmail.com

After your subscription is created, you must confirm it. Info

▸ Subscription filter policy - *optional* Info
This policy filters the messages that a subscriber receives.

▸ Redrive policy (dead-letter queue) - *optional* Info
Send undeliverable messages to a dead-letter queue.

Cancel | Create subscription



New Feature

Amazon SNS now supports in-place message archiving and replay for FIFO topics. Learn more

Confirmation request was sent successfully.

The ARN of the subscription is arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:7aee5d16-12ad-4731-9f7d-c542c2715ad5.

BookRequestNotifications

Details

Name: BookRequestNotifications

Display name: -

ARN: arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications

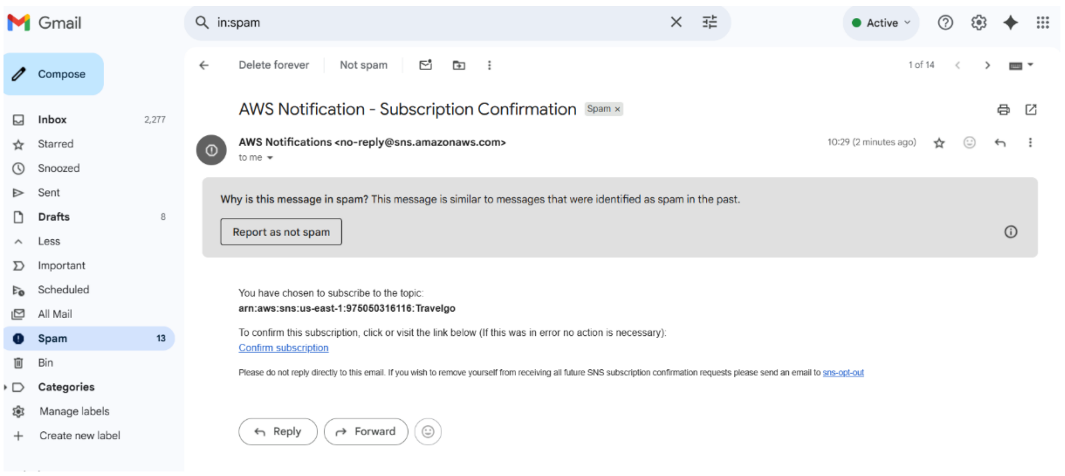
Topic owner: 557690616836

Type: Standard

Subscriptions (2)

ID	Endpoint	Status	Protocol
Pending confirmation	instantlibrary2@gmail.com	Pending confirmation	EMAIL

- After Comfirmation of Subscription going to Mail for comfirm Mail .



AWS Notification - Subscription Confirmation

AWS Notifications <no-reply@sns.amazonaws.com>

to me

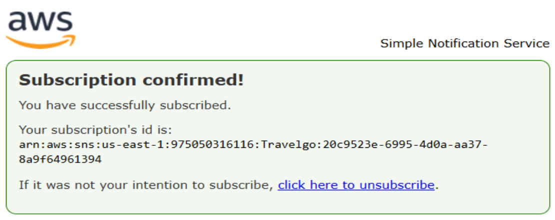
10:29 (2 minutes ago)

Why is this message in spam? This message is similar to messages that were identified as spam in the past.

You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:975050316116:Travelgo

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
Confirm subscription

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to sns-opt-out



Simple Notification Service

Subscription confirmed!

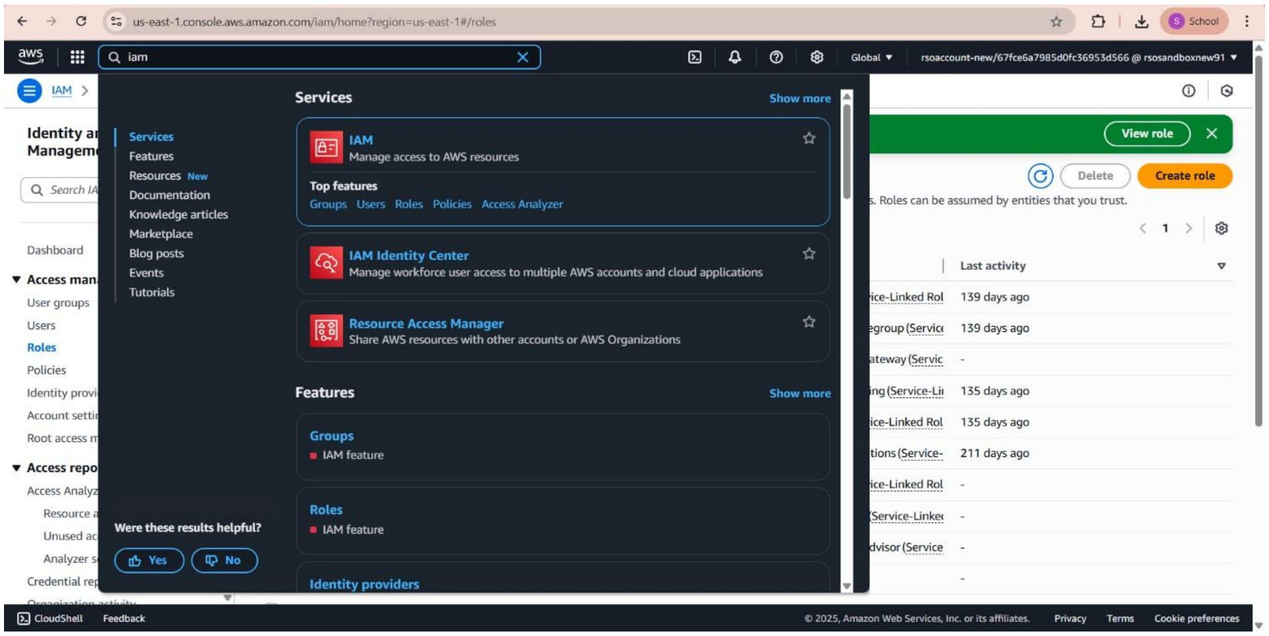
You have successfully subscribed.

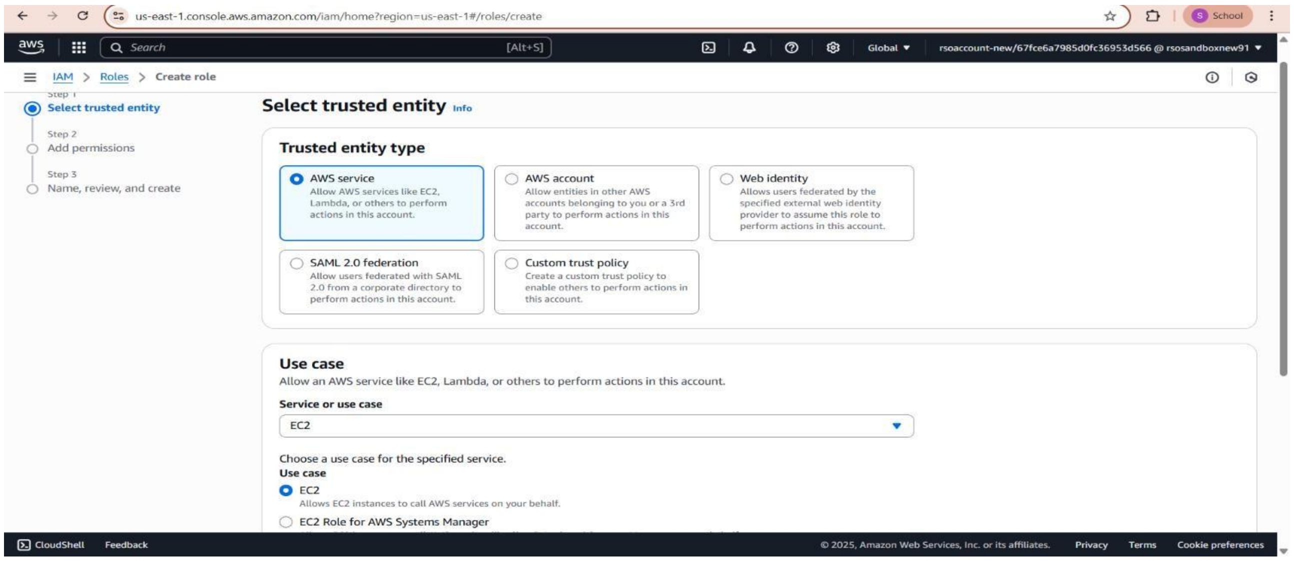
Your subscription's id is:
arn:aws:sns:us-east-1:975050316116:Travelgo:20c9523e-6995-4d0a-aa37-8a9f64961394

If it was not your intention to subscribe, click here to unsubscribe.

Milestone 5 : IAM Role Setup.

- **Activity 5.1 :** In the AWS Console, go to IAM and create a new IAM Role for EC2 to interact with DynamoDB .


us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/roles
iam
Services
Features
Resources New
Documentation
Knowledge articles
Marketplace
Blog posts
Events
Tutorials
IAM
Manage access to AWS resources
Top features
Groups Users Roles Policies Access Analyzer
IAM Identity Center
Manage workforce user access to multiple AWS accounts and cloud applications
Resource Access Manager
Share AWS resources with other accounts or AWS Organizations
Features
Groups
IAM feature
Roles
IAM feature
Identity providers
Were these results helpful?
Yes
No
View role
Delete
Create role
Last activity
Dashboard
User groups
Users
Roles
Policies
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/roles/create
IAM > Roles > Create role
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
CloudShell
Feedback

Select trusted entity

Trusted entity type

- [✓ AWS service] Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account: Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity: Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation: Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy: Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

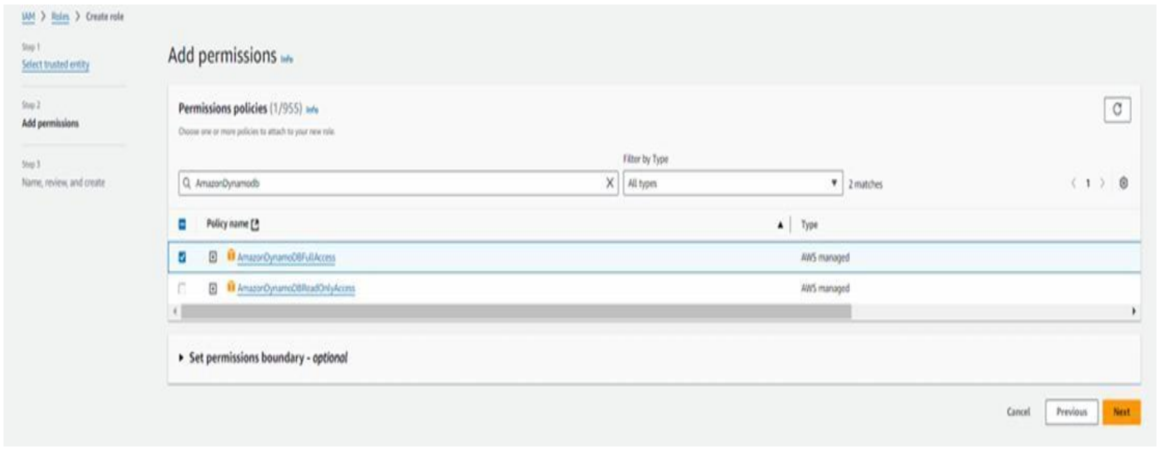
Service or use case: EC2

Choose a use case for the specified service.

Use case

- [✓ EC2] Allows EC2 instances to call AWS services on your behalf.
- EC2 Role for AWS Systems Manager

- **Activity 5.2 : Attach Policies**
- AmazonDynamoDBFullAccess: Allows EC2 to perform read/write operations on DynamoDB.
- AmazonSNSFullAccess: Grants EC2 the ability to send notifications via SNS.



IAM > Roles > Create role

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Add permissions Info

Permissions policies (1/955) Info

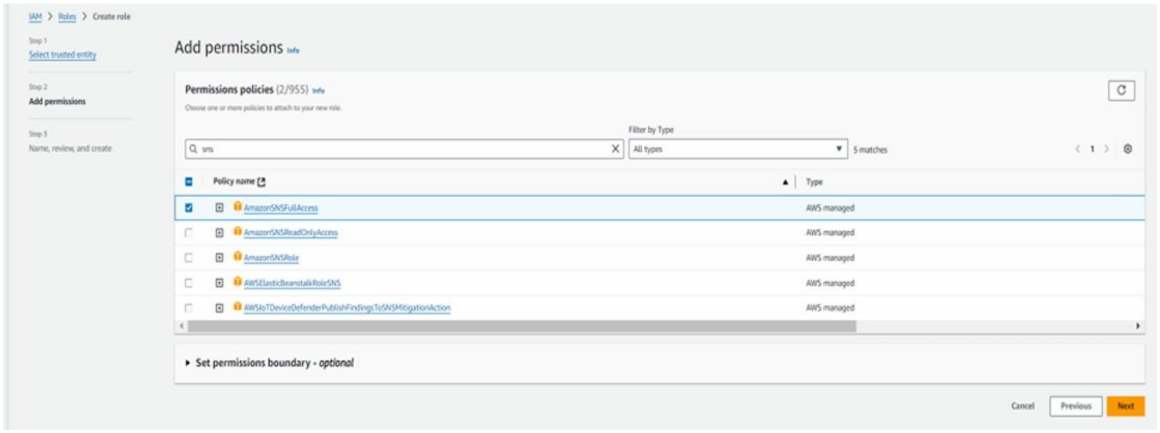
Choose one or more policies to attach to your new role.

Search: AmazonDynamodb — Filter by Type: All types — 2 matches

☐	Policy name	Type
✓	AmazonDynamoDBFullAccess	AWS managed
☐	AmazonDynamoDBReadOnlyAccess	AWS managed

▸ Set permissions boundary - *optional*

Cancel | Previous | Next



IAM > Roles > Create role

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Add permissions Info

Permissions policies (2/955) Info

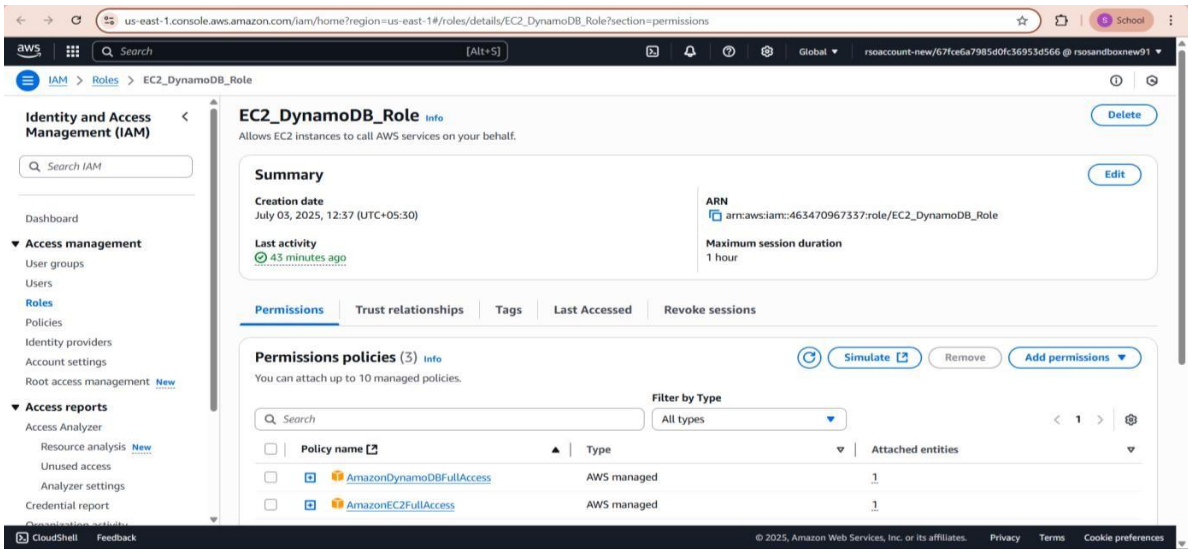
Choose one or more policies to attach to your new role.

Search: sns — Filter by Type: All types — 5 matches

☐	Policy name	Type
✓	AmazonSNSFullAccess	AWS managed
☐	AmazonSNSReadOnlyAccess	AWS managed
☐	AmazonSNSRole	AWS managed
☐	AWSElasticBeanstalkRoleSNS	AWS managed
☐	AWSIoTDeviceDefenderPublishFindingsToSNSMitigationAction	AWS managed

▸ Set permissions boundary - *optional*

Cancel | Previous | Next



us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/roles/details/EC2_DynamoDB_Role?section=permissions

Global ▾ rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91 ▾

IAM > Roles > EC2_DynamoDB_Role

Identity and Access Management (IAM)

Dashboard
▾ Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management New
▾ Access reports
Access Analyzer
Resource analysis New
Unused access
Analyzer settings
Credential report

EC2_DynamoDB_Role Info

Allows EC2 instances to call AWS services on your behalf.

Summary

Creation date
July 03, 2025, 12:37 (UTC+05:30)

Last activity
43 minutes ago

ARN
arn:aws:iam::463470967337:role/EC2_DynamoDB_Role

Maximum session duration
1 hour

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Permissions policies (3) Info

You can attach up to 10 managed policies.

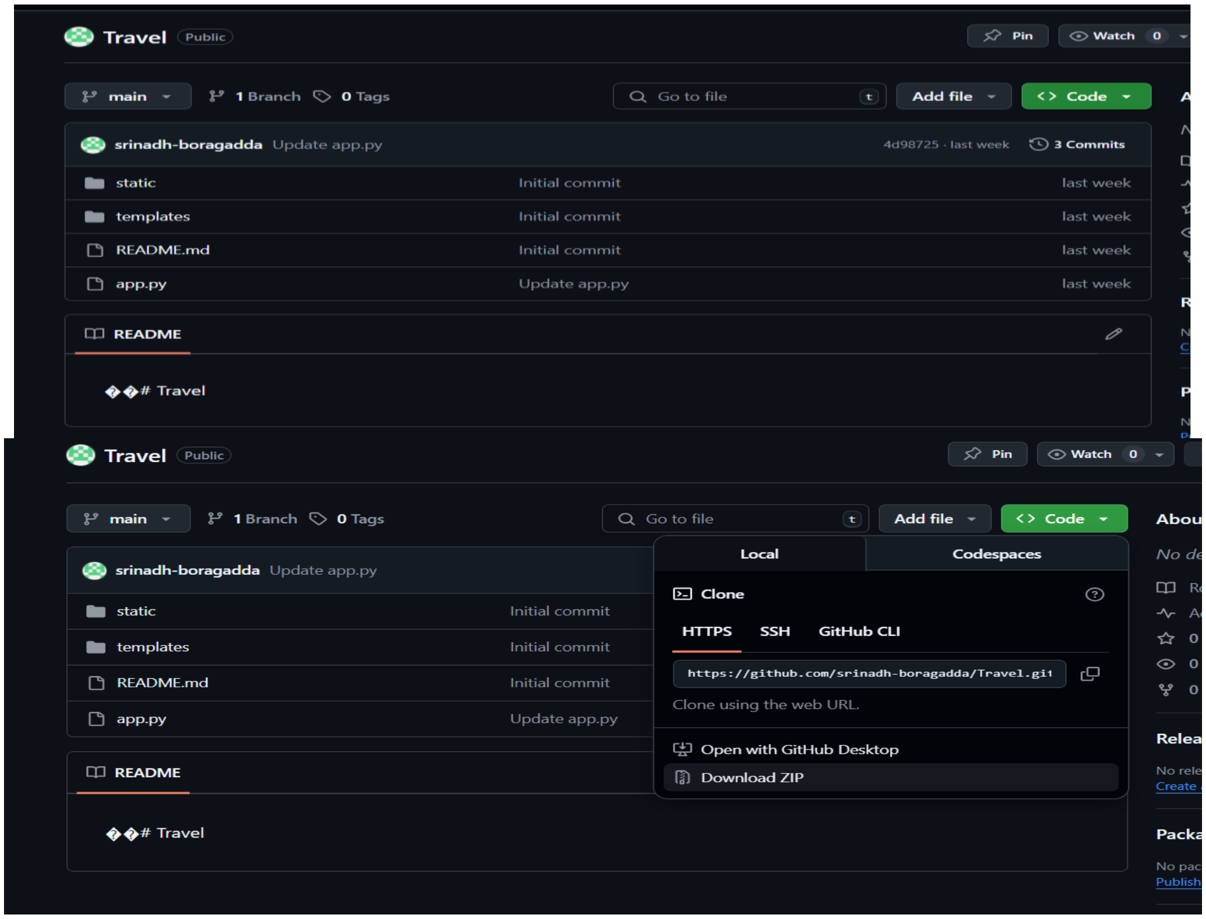
Filter by Type: All types

☐	Policy name	Type	Attached entities
☐	AmazonDynamoDBFullAccess	AWS managed	1
☐	AmazonEC2FullAccess	AWS managed	1

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

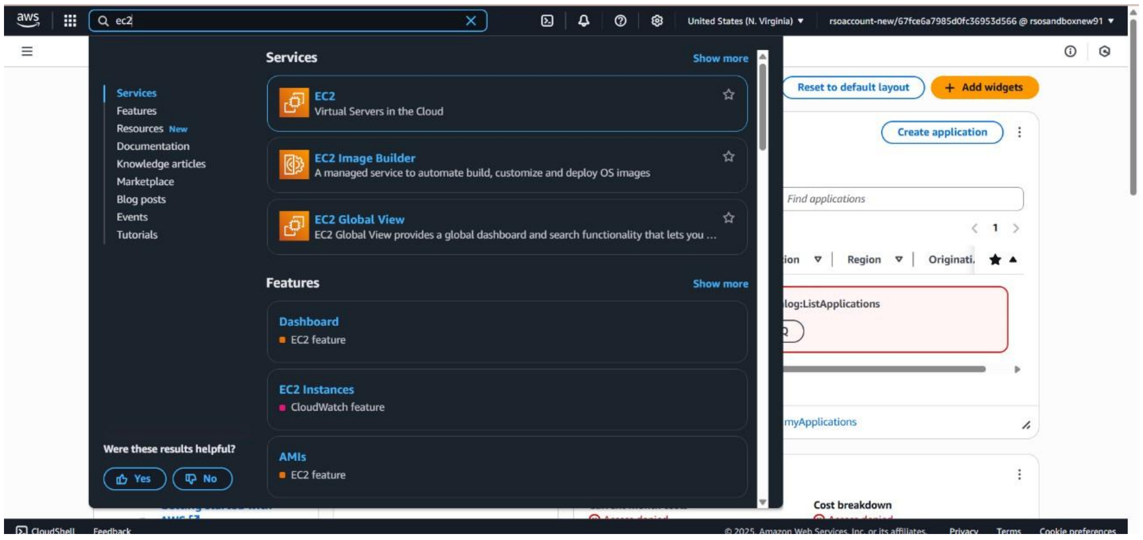
Milestone 6. EC2 Instance Setup

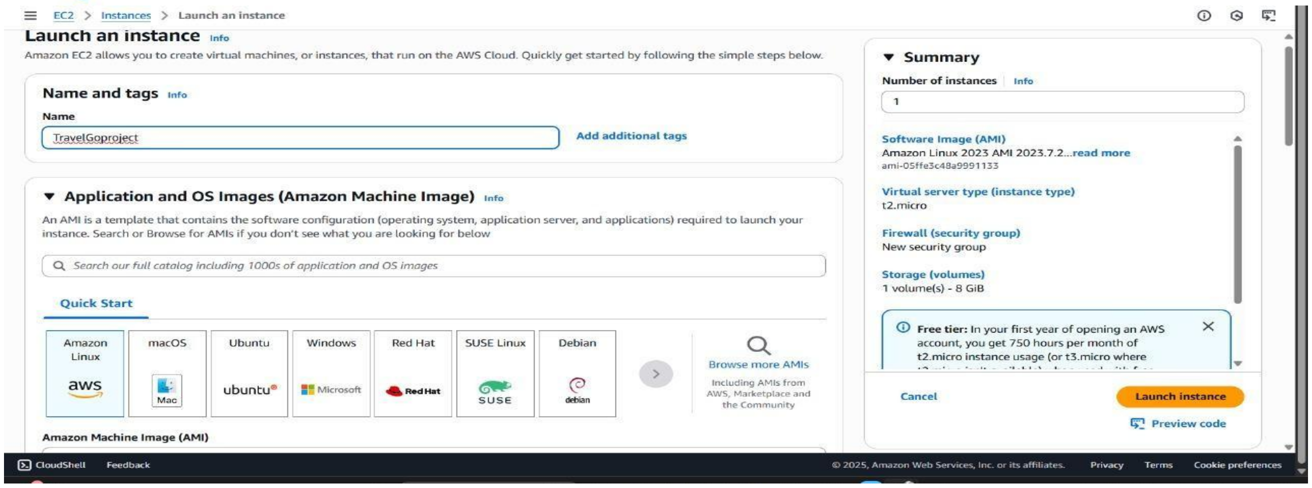
Note: Load your Flask app and Html files into GitHub repository .


Travel Public
Pin
Watch 0
main
1 Branch
0 Tags
Go to file
Add file
Code
srinadh-boragadda Update app.py
4d98725 · last week
3 Commits
static Initial commit last week
templates Initial commit last week
README.md Initial commit last week
app.py Update app.py last week
README
��# Travel
Local
Codespaces
Clone
HTTPS
SSH
GitHub CLI
https://github.com/srinadh-boragadda/Travel.git
Clone using the web URL.
Open with GitHub Desktop
Download ZIP

Activity 6.1: Launch an EC2 instance to host the Flask application.

- In the AWS Console, navigate to EC2 and launch a new instance.





EC2 > Instances > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: TravelGoproject

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Linux | Debian

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Summary

Number of instances Info: 1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.7.2...read more
ami-05ffe3c48a9991133

Virtual server type (instance type)
t2.micro

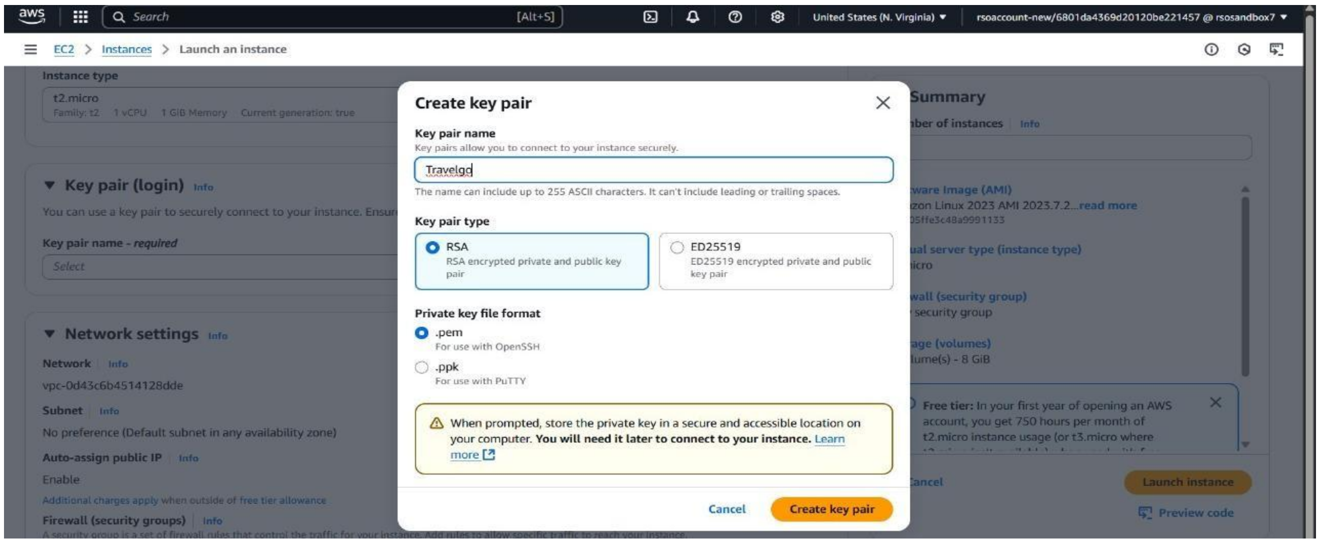
Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where

Cancel | Launch instance | Preview code

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



Search [Alt+S] United States (N. Virginia) rsoaccount-new/6801da4369d20120be221457 @ rsosandbox7

EC2 > Instances > Launch an instance

Instance type
t2.micro
Family: t2 1 vCPU 1 GiB Memory Current generation: true

Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensur

Key pair name - *required*
Select

Network settings Info

Network Info
vpc-0d43c6b4514128dde

Subnet Info
No preference (Default subnet in any availability zone)

Auto-assign public IP Info
Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) Info

Create key pair

Key pair name
Key pairs allow you to connect to your instance securely.
Travelgo

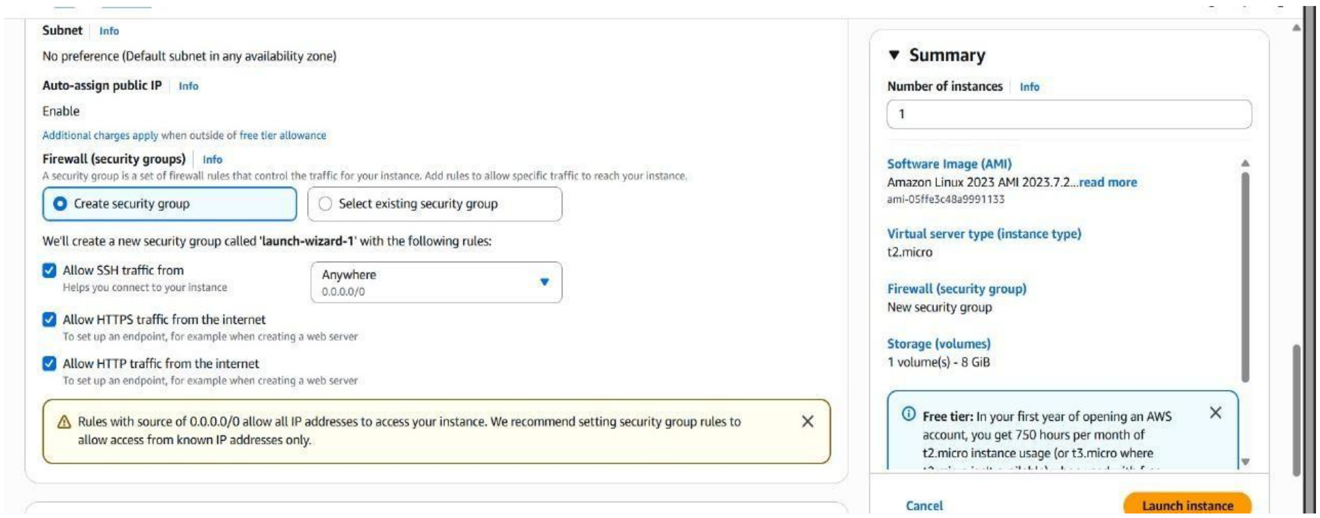
The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type
[✓ RSA] RSA encrypted private and public key pair
ED25519 ED25519 encrypted private and public key pair

Private key file format
[✓ .pem] For use with OpenSSH
.ppk For use with PuTTY

When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn more

Cancel | Create key pair



Subnet Info
No preference (Default subnet in any availability zone)

Auto-assign public IP Info
Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

[✓ Create security group] Select existing security group

We'll create a new security group called '**launch-wizard-1**' with the following rules:

✓ Allow SSH traffic from — Anywhere 0.0.0.0/0
Helps you connect to your instance

✓ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

✓ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Summary

Number of instances Info: 1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.7.2...read more
ami-05ffe3c48a9991133

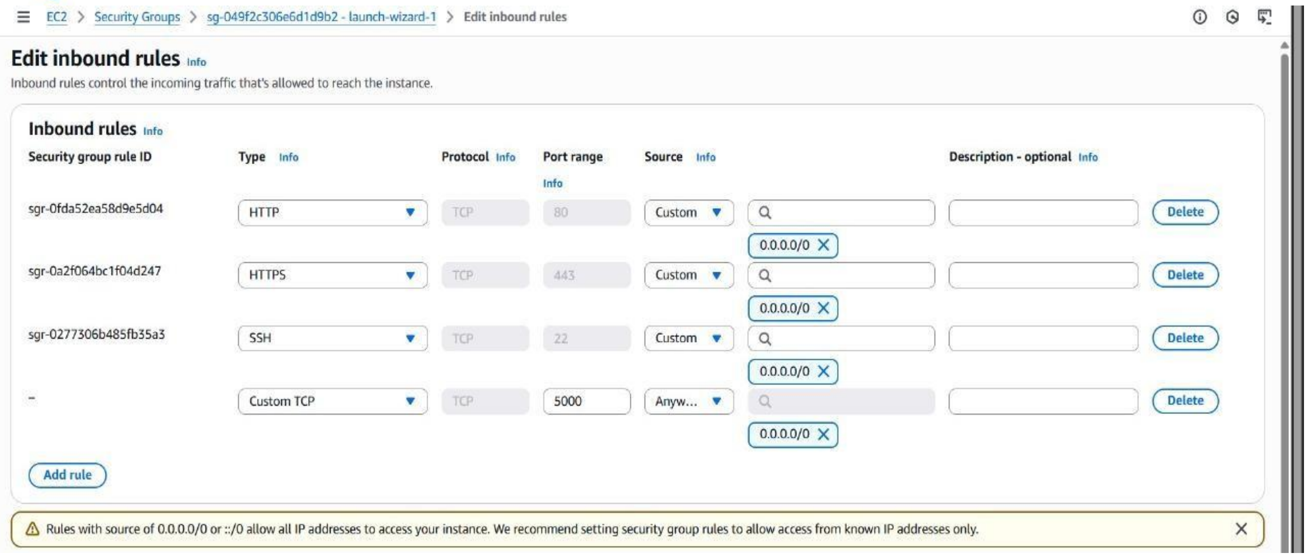
Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where

Cancel | Launch instance


EC2 > Security Groups > sg-049f2c306e6d1d9b2 - launch-wizard-1 > Edit inbound rules

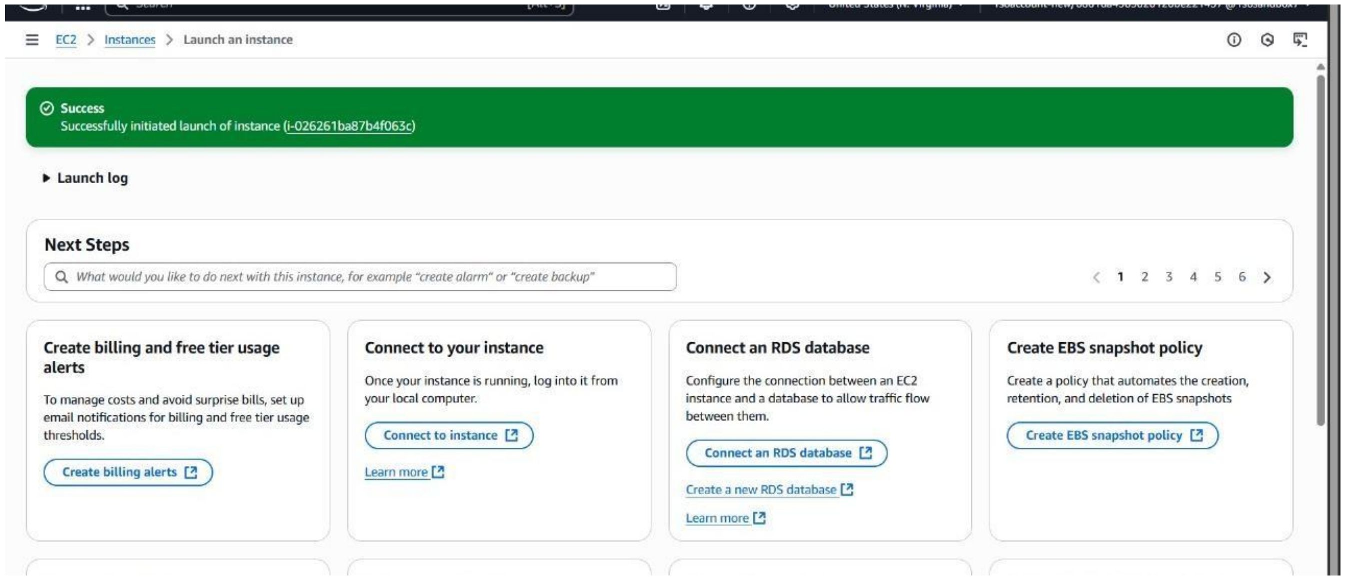
Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-0fda52ea58d9e5d04	HTTP	TCP	80	Custom	0.0.0.0/0	
sgr-0a2f064bc1f04d247	HTTPS	TCP	443	Custom	0.0.0.0/0	
sgr-0277306b485fb35a3	SSH	TCP	22	Custom	0.0.0.0/0	
–	Custom TCP	TCP	5000	Anyw...	0.0.0.0/0	

Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.


EC2 > Instances > Launch an instance

Success
Successfully initiated launch of instance (i-026261ba87b4f063c)

Launch log

Next Steps

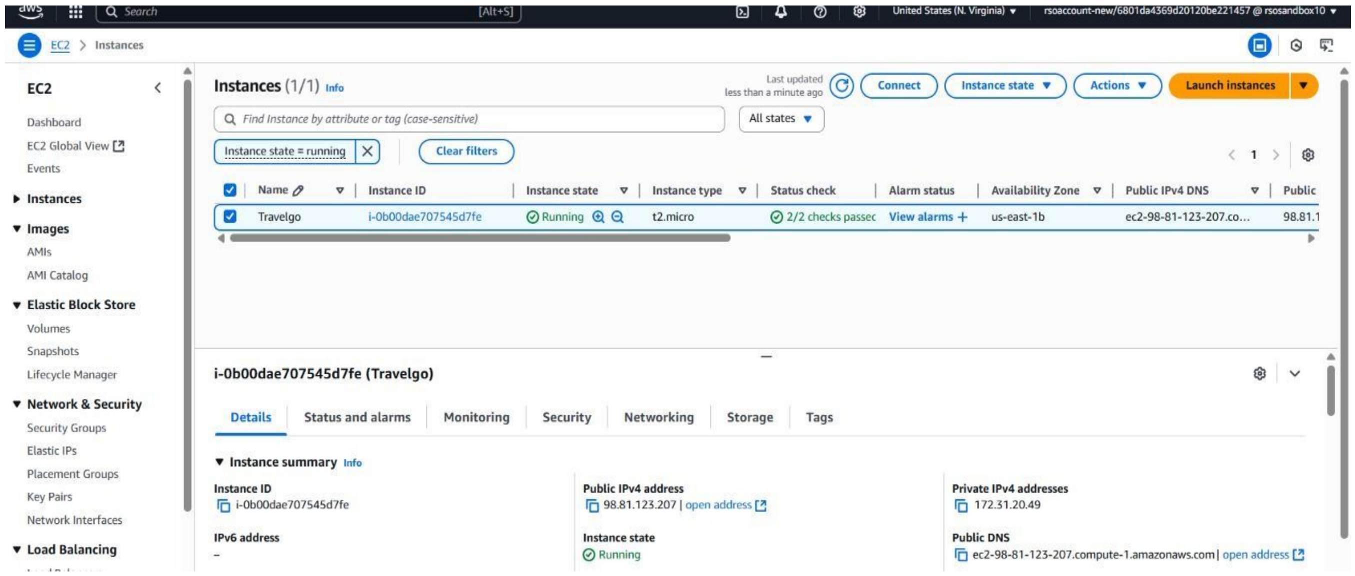
What would you like to do next with this instance, for example "create alarm" or "create backup"

Create billing and free tier usage alerts
To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.
Create billing alerts

Connect to your instance
Once your instance is running, log into it from your local computer.
Connect to instance
Learn more

Connect an RDS database
Configure the connection between an EC2 instance and a database to allow traffic flow between them.
Connect an RDS database
Create a new RDS database
Learn more

Create EBS snapshot policy
Create a policy that automates the creation, retention, and deletion of EBS snapshots
Create EBS snapshot policy


United States (N. Virginia) rsoaccount-new/6801da4369d20120be221457 @ rsosandbox10

EC2 > Instances

Instances (1/1)

Last updated less than a minute ago

Instance state = running

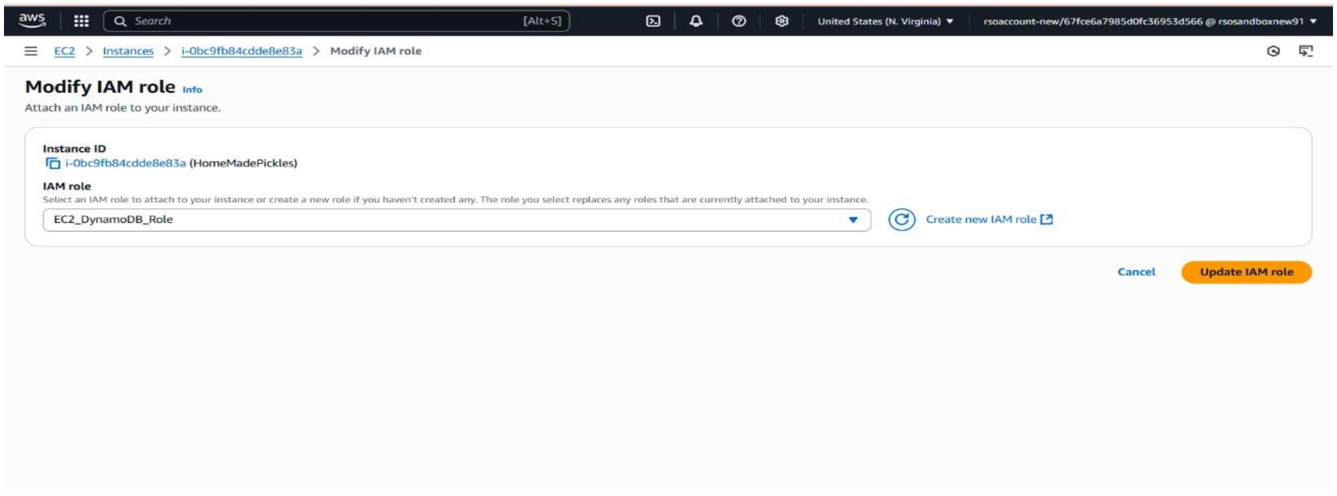
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public
Travelgo	i-0b00dae707545d7fe	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b	ec2-98-81-123-207.co...	98.81.1

i-0b00dae707545d7fe (Travelgo)

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary

Instance ID i-0b00dae707545d7fe	Public IPv4 address 98.81.123.207 | open address	Private IPv4 addresses 172.31.20.49
IPv6 address –	Instance state Running	Public DNS ec2-98-81-123-207.compute-1.amazonaws.com| open address



aws | Search [Alt+S] | United States (N. Virginia) ▼ | rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91 ▼

EC2 > Instances > i-0bc9fb84cdde8e83a > Modify IAM role

Modify IAM role Info

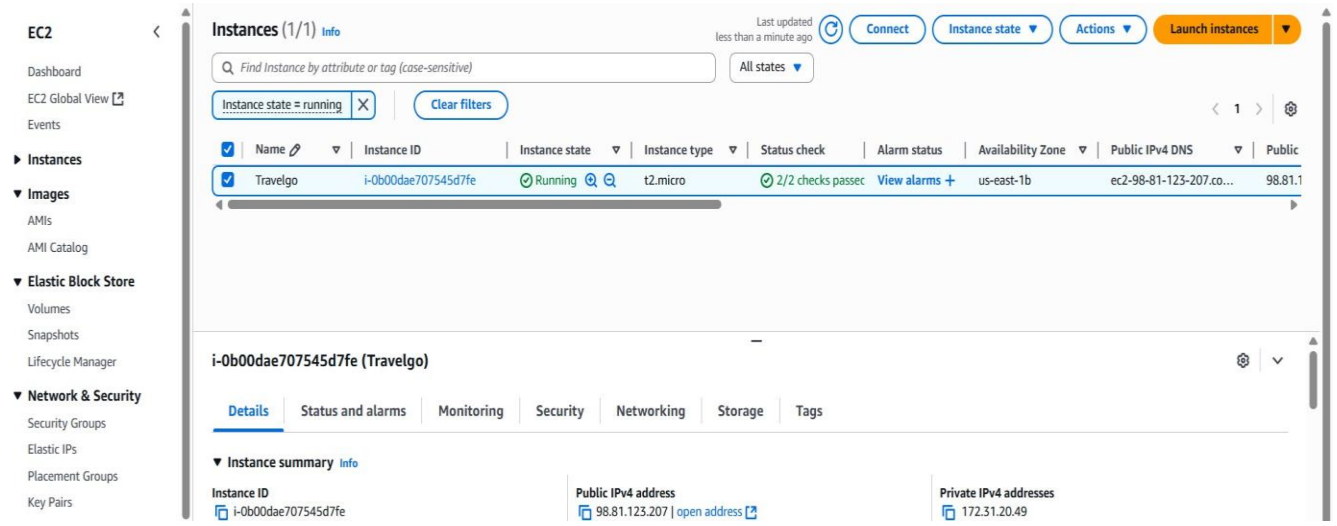
Attach an IAM role to your instance.

Instance ID
i-0bc9fb84cdde8e83a (HomeMadePickles)

IAM role
Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

EC2_DynamoDB_Role ▼ | Create new IAM role

Cancel | Update IAM role



Instances (1/1) Info

Last updated less than a minute ago | Connect | Instance state ▼ | Actions ▼ | Launch instances ▼

Find Instance by attribute or tag (case-sensitive) | All states ▼

Instance state = running | Clear filters

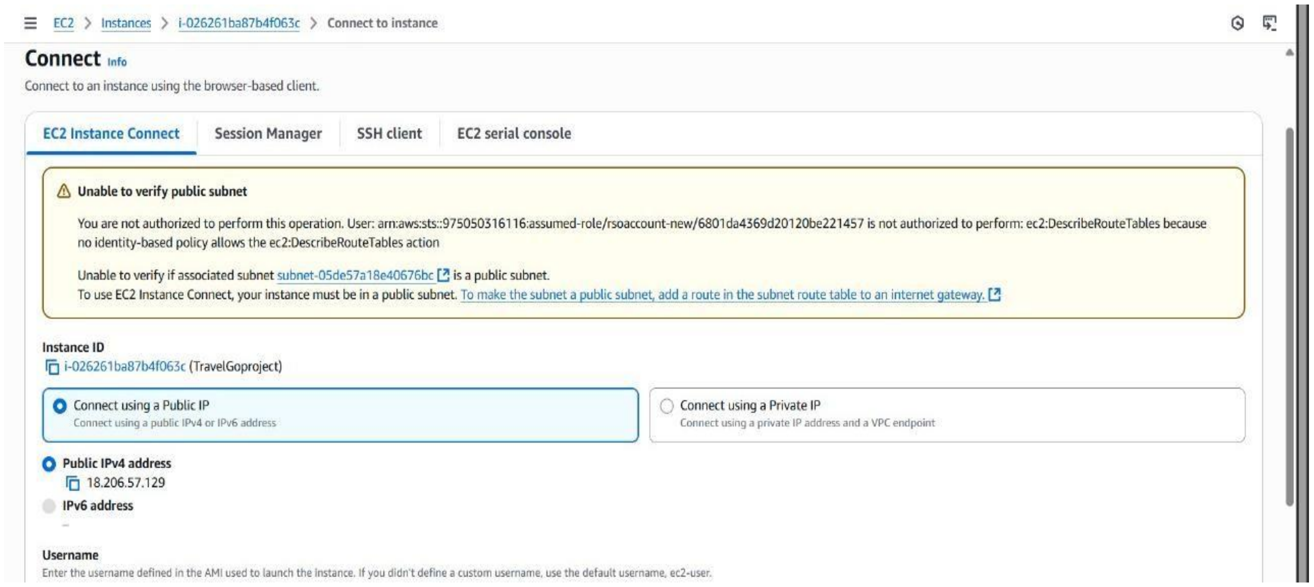
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public
☑	Travelgo	i-0b00dae707545d7fe	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b	ec2-98-81-123-207.co...	98.81.1

i-0b00dae707545d7fe (Travelgo)

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

▼ Instance summary Info

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0b00dae707545d7fe	98.81.123.207 | open address	172.31.20.49



EC2 > Instances > i-026261ba87b4f063c > Connect to instance

Connect Info

Connect to an instance using the browser-based client.

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

⚠ **Unable to verify public subnet**

You are not authorized to perform this operation. User: arn:aws:sts::975050316116:assumed-role/rsoaccount-new/6801da4369d20120be221457 is not authorized to perform: ec2:DescribeRouteTables because no identity-based policy allows the ec2:DescribeRouteTables action

Unable to verify if associated subnet subnet-05de57a18e40676bc is a public subnet.
To use EC2 Instance Connect, your instance must be in a public subnet. To make the subnet a public subnet, add a route in the subnet route table to an internet gateway.

Instance ID
i-026261ba87b4f063c (TravelGoproject)

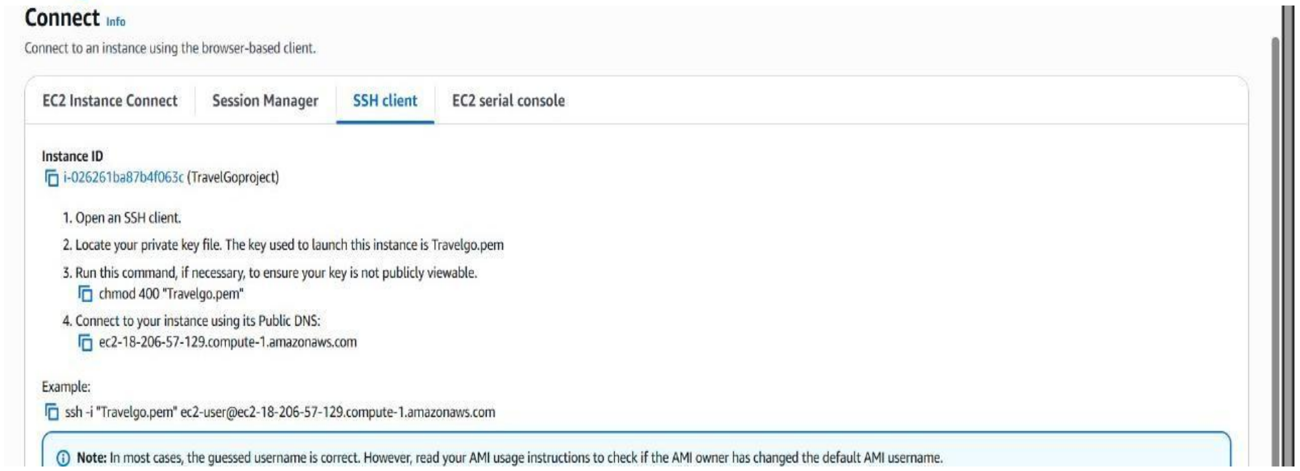
● Connect using a Public IP
Connect using a public IPv4 or IPv6 address

○ Connect using a Private IP
Connect using a private IP address and a VPC endpoint

● **Public IPv4 address**
18.206.57.129

○ **IPv6 address**
–

Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.



Connect Info

Connect to an instance using the browser-based client.

Instance ID

i-026261ba87b4f063c (TravelGoproject)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is Travelgo.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.

```
chmod 400 "Travelgo.pem"
```

4. Connect to your instance using its Public DNS:

```
ec2-18-206-57-129.compute-1.amazonaws.com
```

Example:

```
ssh -i "Travelgo.pem" ec2-user@ec2-18-206-57-129.compute-1.amazonaws.com
```

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Milestone 7: Deployment on EC2

Activity 7.1: Install Software on the EC2 Instance

- **Install Python3, Flask, and Git:**
- **On Amazon Linux 2:**
- **sudo yum update -y**
- **sudo yum install python3 git**
- **sudo pip3 install flask boto3**
- **Verify Installations:**
- **flask --version**
- **git –version**

Activity 7.2 : Clone Your Flask Project from GitHub.

Clone your project repository from GitHub into the EC2 instance using Git.

Run :' git clone https://github.com/srinadh-boragadda/Travel ' .

This will download the Project to Ec2 instance.

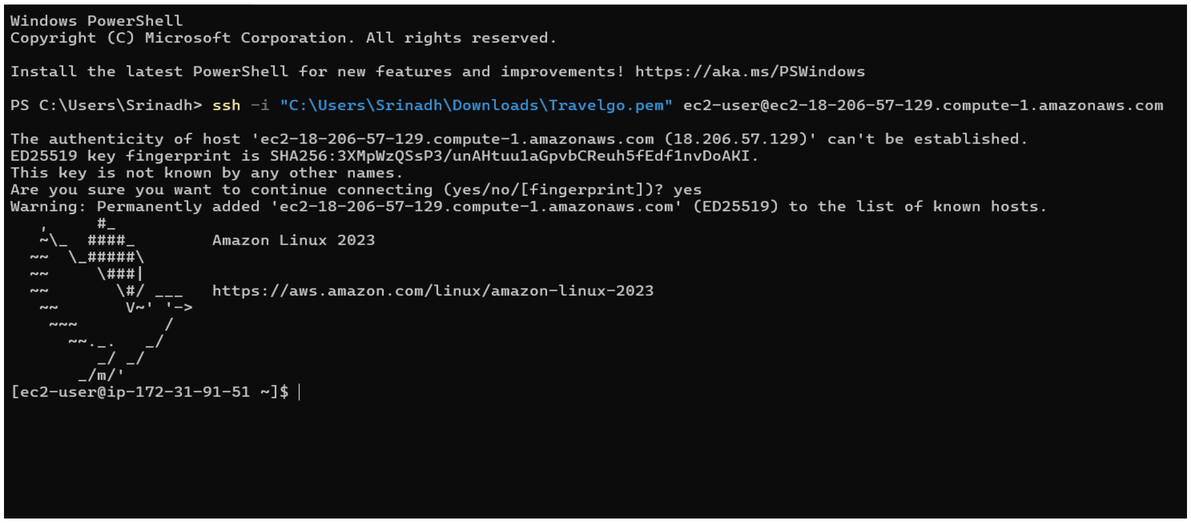
To navigate to the project directory, run the following command:

cd InstantLibrary

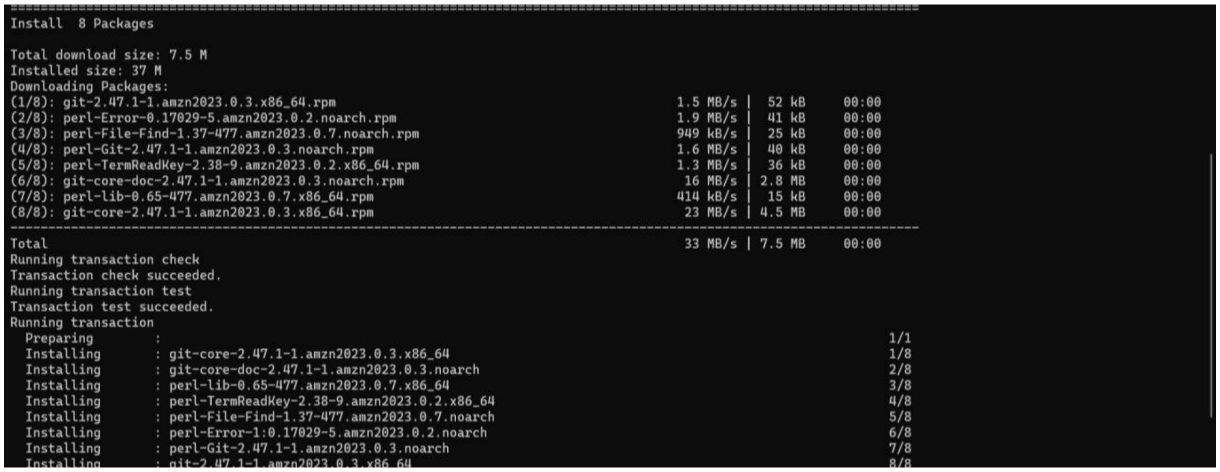
Once inside the project directory, configure and run the Flask application by executing the following command with elevated privileges:

Run the Flask Application

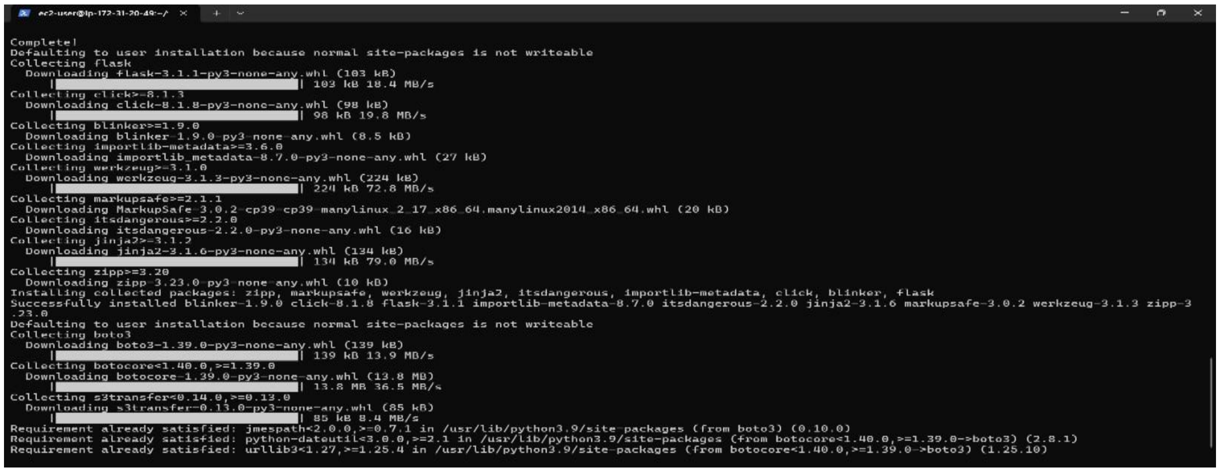
sudo flask run --host=0.0.0.0 --port=80



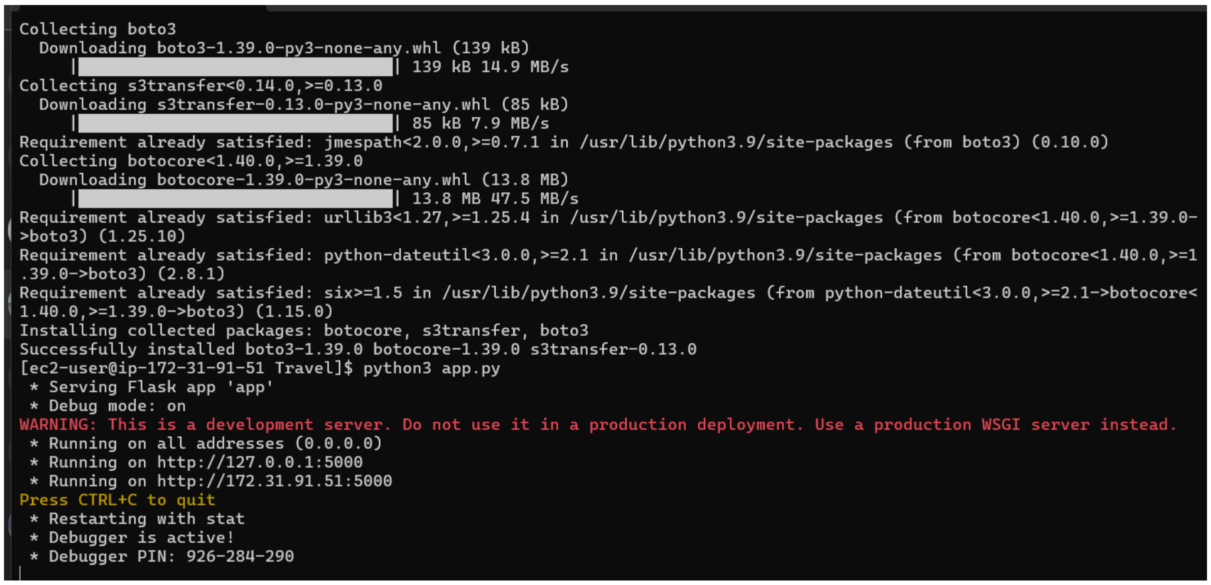
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Srinadh> ssh -i "C:\Users\Srinadh\Downloads\Travelgo.pem" ec2-user@ec2-18-206-57-129.compute-1.amazonaws.com

The authenticity of host 'ec2-18-206-57-129.compute-1.amazonaws.com (18.206.57.129)' can't be established.
ED25519 key fingerprint is SHA256:3XMpWzQSsP3/unAHtuu1aGpvbCReuh5fEdf1nvDoAKI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-18-206-57-129.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-91-51 ~]$ |
```



```
Install  8 Packages

Total download size: 7.5 M
Installed size: 37 M
Downloading Packages:
(1/8): git-2.47.1-1.amzn2023.0.3.x86_64.rpm                              1.5 MB/s |  52 kB     00:00
(2/8): perl-Error-0.17029-5.amzn2023.0.2.noarch.rpm                      1.9 MB/s |  41 kB     00:00
(3/8): perl-File-Find-1.37-477.amzn2023.0.7.noarch.rpm                   949 kB/s |  25 kB     00:00
(4/8): perl-Git-2.47.1-1.amzn2023.0.3.noarch.rpm                         1.6 MB/s |  40 kB     00:00
(5/8): perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64.rpm                   1.3 MB/s |  36 kB     00:00
(6/8): git-core-doc-2.47.1-1.amzn2023.0.3.noarch.rpm                      16 MB/s | 2.8 MB     00:00
(7/8): perl-lib-0.65-477.amzn2023.0.7.x86_64.rpm                         414 kB/s |  15 kB     00:00
(8/8): git-core-2.47.1-1.amzn2023.0.3.x86_64.rpm                          23 MB/s | 4.5 MB     00:00
-----------------------------------------------------------------------------------------------------
Total                                                                     33 MB/s | 7.5 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                        1/1
  Installing       : git-core-2.47.1-1.amzn2023.0.3.x86_64                                  1/8
  Installing       : git-core-doc-2.47.1-1.amzn2023.0.3.noarch                              2/8
  Installing       : perl-lib-0.65-477.amzn2023.0.7.x86_64                                  3/8
  Installing       : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64                            4/8
  Installing       : perl-File-Find-1.37-477.amzn2023.0.7.noarch                            5/8
  Installing       : perl-Error-1:0.17029-5.amzn2023.0.2.noarch                             6/8
  Installing       : perl-Git-2.47.1-1.amzn2023.0.3.noarch                                  7/8
  Installing       : git-2.47.1-1.amzn2023.0.3.x86_64                                       8/8
```



```
Complete!
Defaulting to user installation because normal site-packages is not writeable
Collecting flask
  Downloading flask-3.1.1-py3-none-any.whl (103 kB)
     |████████████████████████████████| 103 kB 18.4 MB/s
Collecting click>=8.1.3
  Downloading click-8.1.8-py3-none-any.whl (98 kB)
     |████████████████████████████████| 98 kB 19.8 MB/s
Collecting blinker>=1.9.0
  Downloading blinker-1.9.0-py3-none-any.whl (8.5 kB)
Collecting importlib-metadata>=3.6.0
  Downloading importlib_metadata-8.7.0-py3-none-any.whl (27 kB)
Collecting werkzeug>=3.1.0
  Downloading werkzeug-3.1.3-py3-none-any.whl (224 kB)
     |████████████████████████████████| 224 kB 72.8 MB/s
Collecting markupsafe>=2.1.1
  Downloading MarkupSafe-3.0.2-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (20 kB)
Collecting itsdangerous>=2.2.0
  Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Collecting jinja2>=3.1.2
  Downloading jinja2-3.1.6-py3-none-any.whl (134 kB)
     |████████████████████████████████| 134 kB 79.0 MB/s
Collecting zipp>=3.20
  Downloading zipp-3.23.0-py3-none-any.whl (10 kB)
Installing collected packages: zipp, markupsafe, werkzeug, jinja2, itsdangerous, importlib-metadata, click, blinker, flask
Successfully installed blinker-1.9.0 click-8.1.8 flask-3.1.1 importlib-metadata-8.7.0 itsdangerous-2.2.0 jinja2-3.1.6 markupsafe-3.0.2 werkzeug-3.1.3 zipp-3
.23.0
Defaulting to user installation because normal site-packages is not writeable
Collecting boto3
  Downloading boto3-1.39.0-py3-none-any.whl (139 kB)
     |████████████████████████████████| 139 kB 13.9 MB/s
Collecting botocore<1.40.0,>=1.39.0
  Downloading botocore-1.39.0-py3-none-any.whl (13.8 MB)
     |████████████████████████████████| 13.8 MB 36.5 MB/s
Collecting s3transfer<0.14.0,>=0.13.0
  Downloading s3transfer-0.13.0-py3-none-any.whl (85 kB)
     |████████████████████████████████| 85 kB 8.4 MB/s
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3.9/site-packages (from boto3) (0.10.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.0->boto3) (2.8.1)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.0->boto3) (1.25.10)
```



```
Collecting boto3
  Downloading boto3-1.39.0-py3-none-any.whl (139 kB)
     |████████████████████████████████| 139 kB 14.9 MB/s
Collecting s3transfer<0.14.0,>=0.13.0
  Downloading s3transfer-0.13.0-py3-none-any.whl (85 kB)
     |████████████████████████████████| 85 kB 7.9 MB/s
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3.9/site-packages (from boto3) (0.10.0)
Collecting botocore<1.40.0,>=1.39.0
  Downloading botocore-1.39.0-py3-none-any.whl (13.8 MB)
     |████████████████████████████████| 13.8 MB 47.5 MB/s
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.0-
>boto3) (1.25.10)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1
.39.0->boto3) (2.8.1)
Requirement already satisfied: six>=1.5 in /usr/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<
1.40.0,>=1.39.0->boto3) (1.15.0)
Installing collected packages: botocore, s3transfer, boto3
Successfully installed boto3-1.39.0 botocore-1.39.0 s3transfer-0.13.0
[ec2-user@ip-172-31-91-51 Travel]$ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.31.91.51:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 926-284-290
```

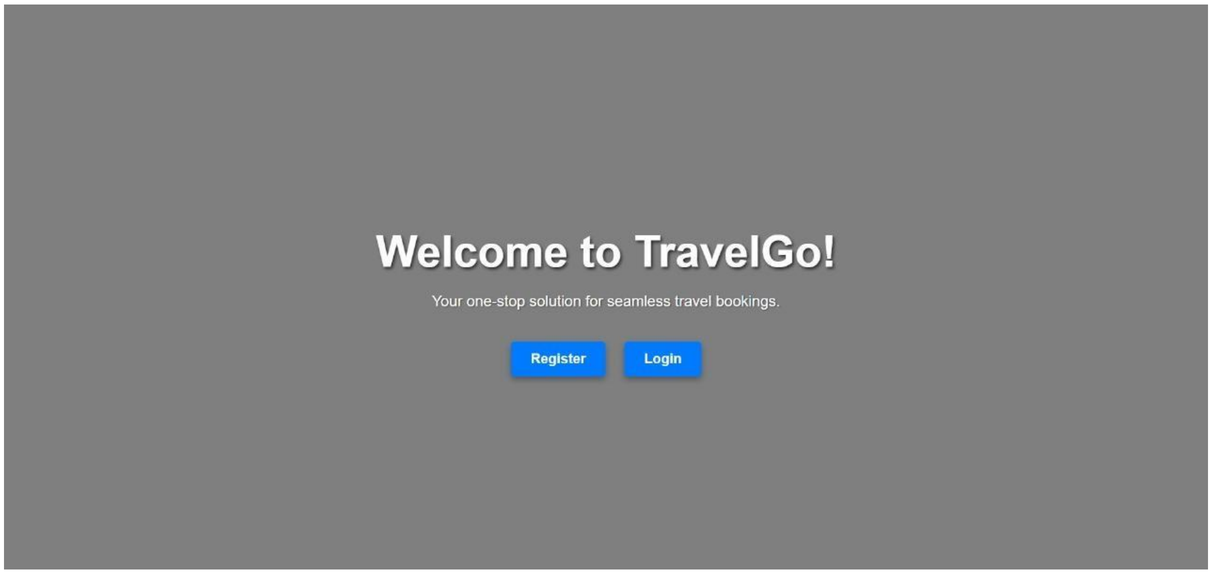
Access the website through:

Public IPs: ‘ http://18.206.57.129:5000/ ’

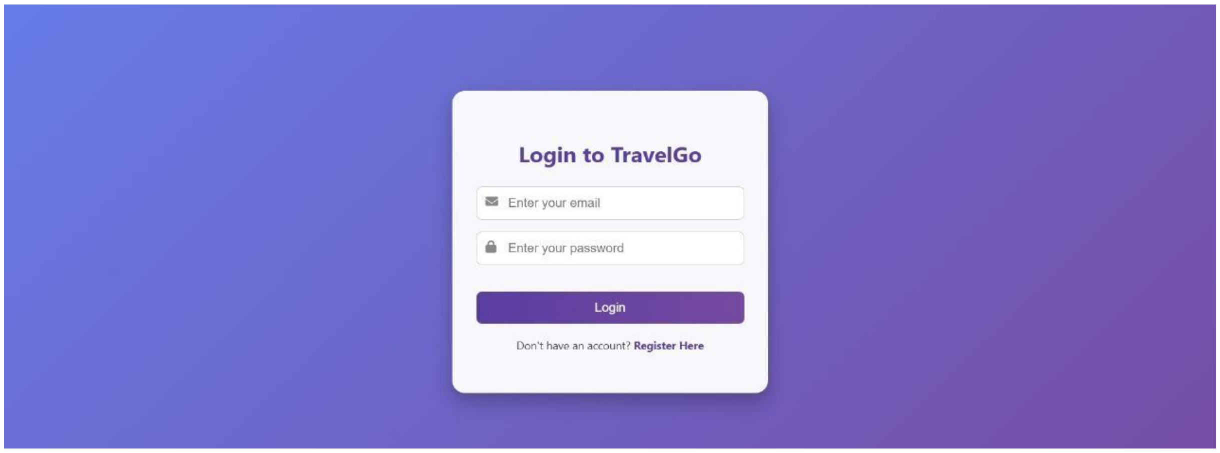
Milestone 8. Testing and Deployment

Activity 8.1 : Conduct functional testing to verify user signup, login, buy/sell stocks and notifications.

HOME PAGE:



LOGIN PAGE :


Login to TravelGo

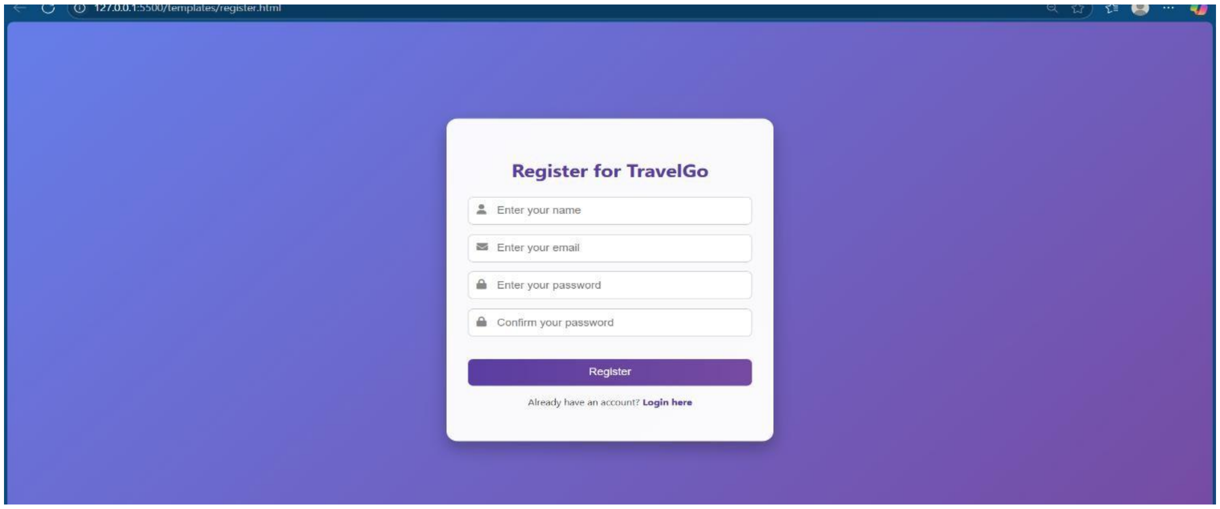
Enter your email

Enter your password

Login

Don't have an account? **Register Here**

SIGNUP PAGE :


Register for TravelGo

Enter your name

Enter your email

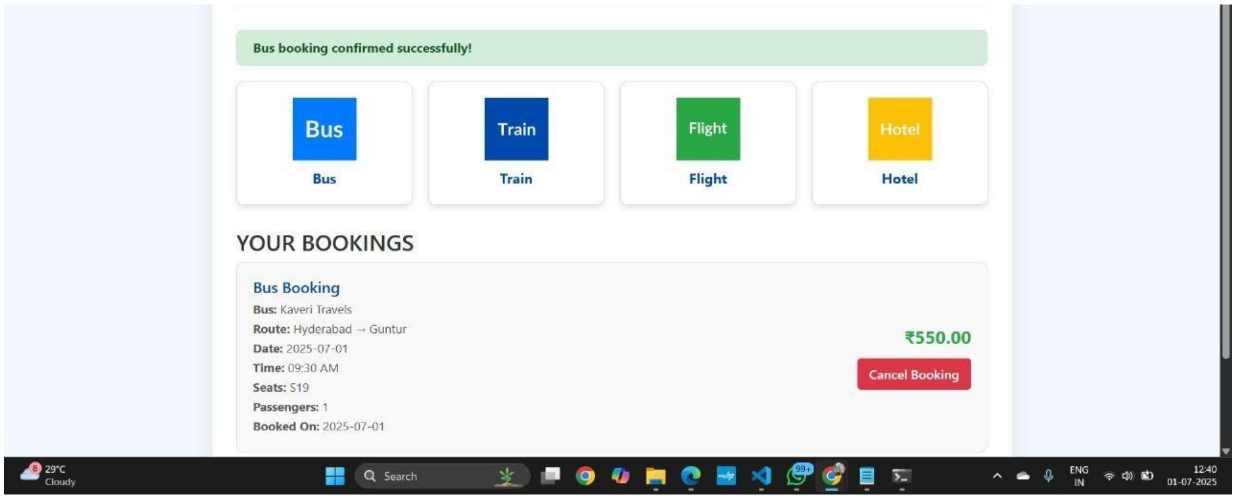
Enter your password

Confirm your password

Register

Already have an account? **Login here**

Dashboard:


Bus booking confirmed successfully!

YOUR BOOKINGS

Bus Booking

Bus: Kaveri Travels

Route: Hyderabad → Guntur

Date: 2025-07-01

Time: 09:30 AM

Seats: S19

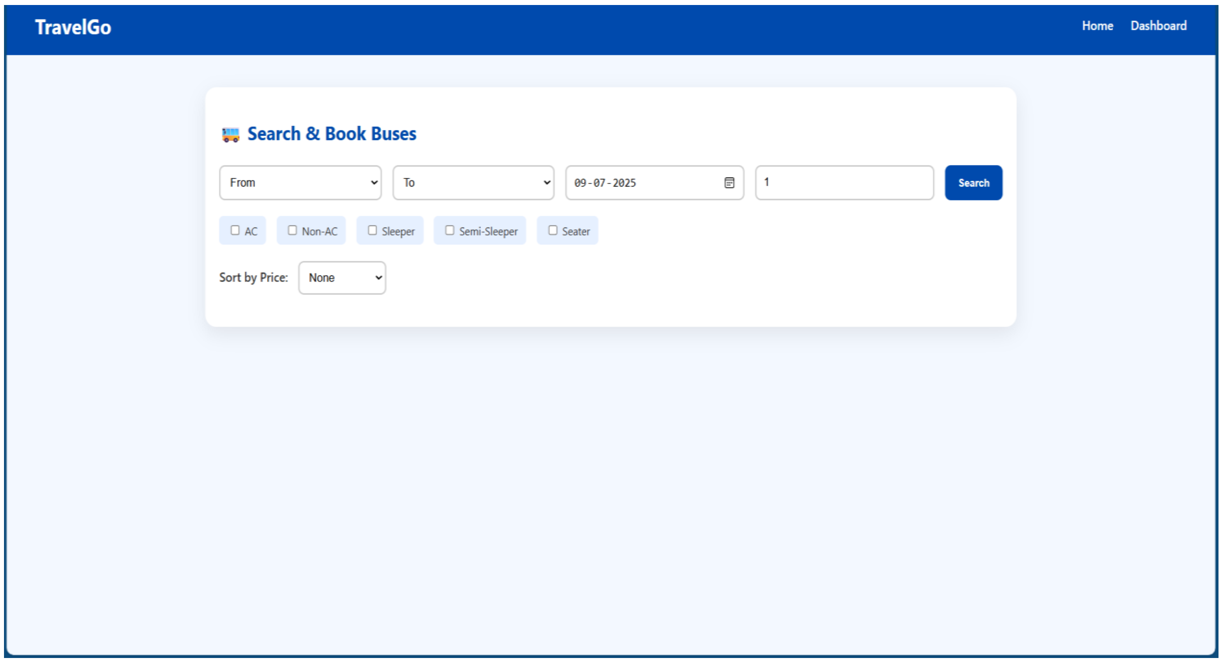
Passengers: 1

Booked On: 2025-07-01

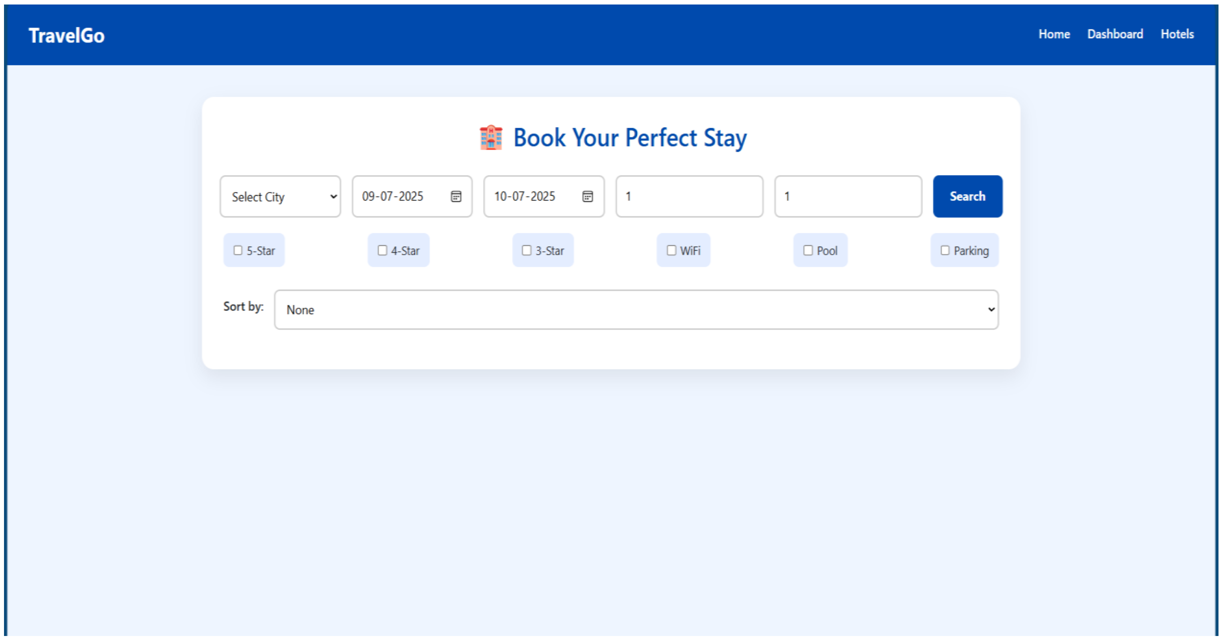
₹550.00

Cancel Booking

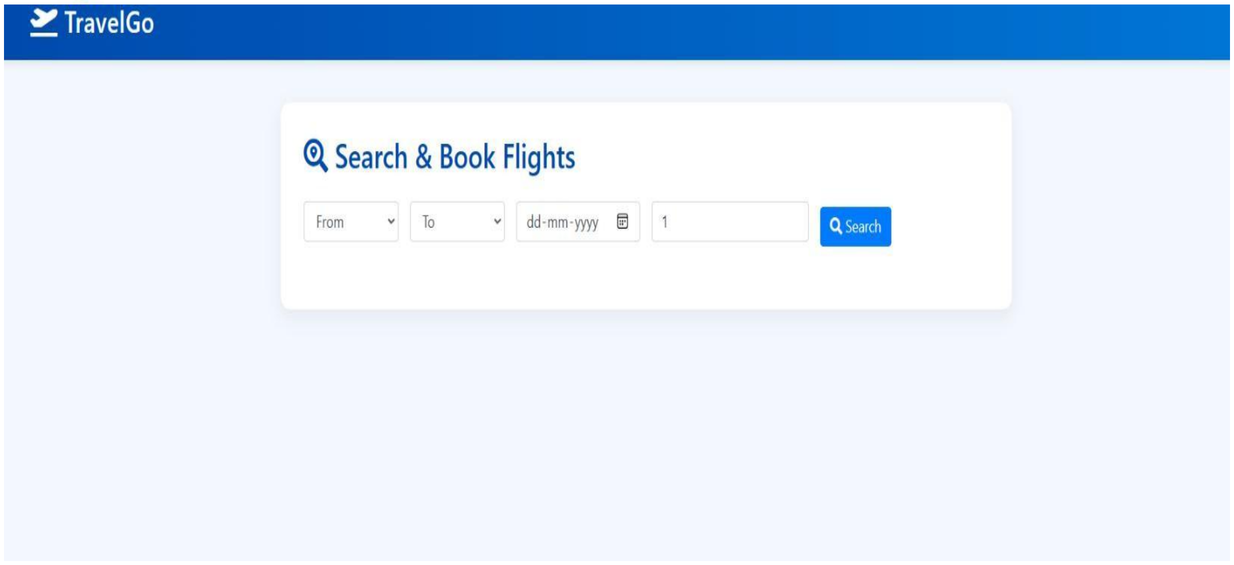
BUS PAGE :



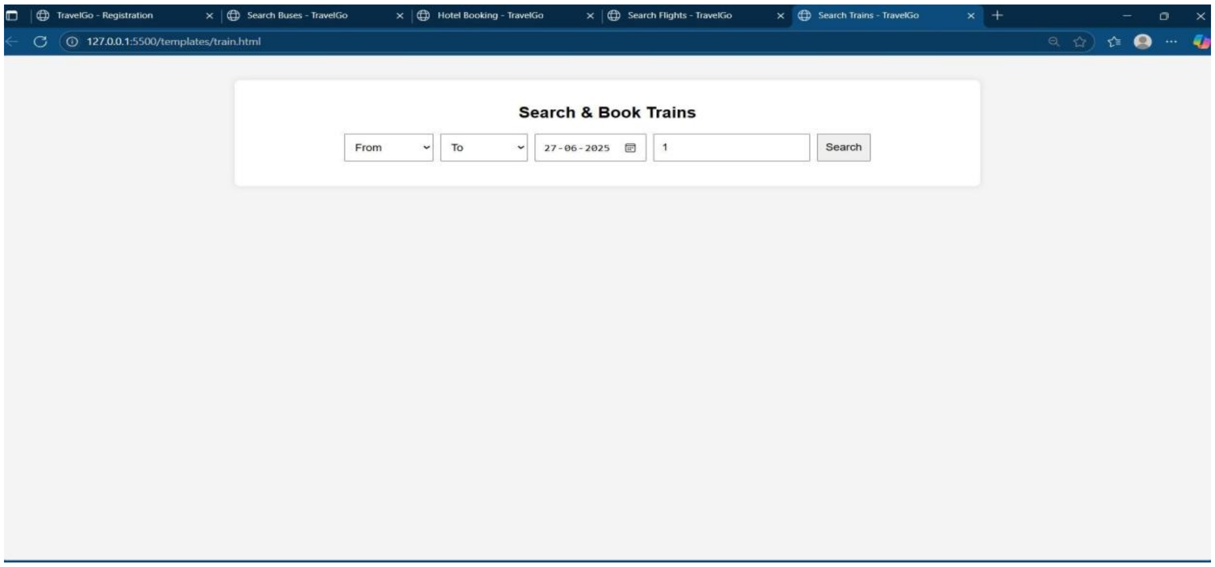
HOTEL PAGE :



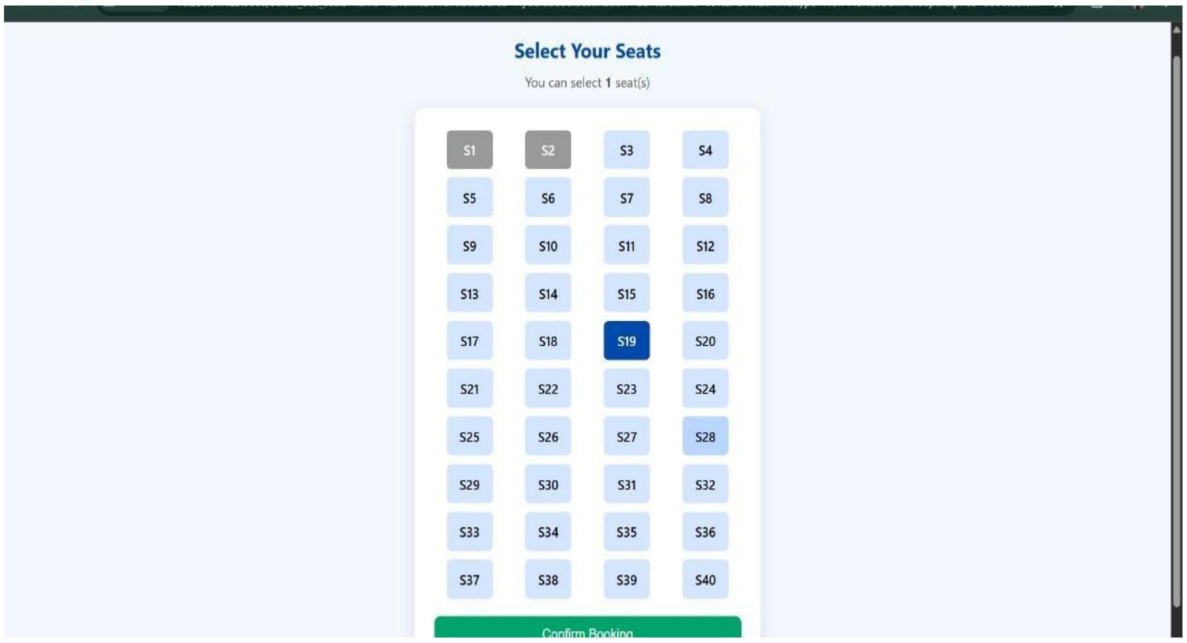
FLIGHT PAGE :



TRAIN PAGE :

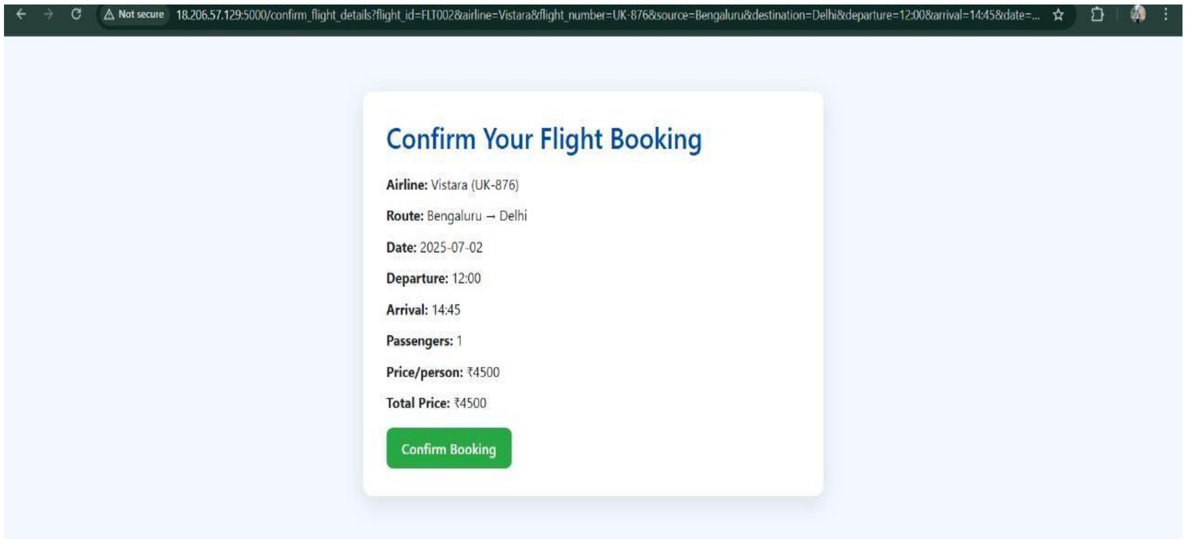


SEAT SELECT PAGE :


Select Your Seats

You can select **1** seat(s)

FLIGHT BOOKING PAGE :


```
18.206.57.129:5000/confirm_flight_details?flight_id=FLT002&airline=Vistara&flight_number=UK-876&source=Bengaluru&destination=Delhi&departure=12:00&arrival=14:45&date=...
```

Confirm Your Flight Booking

Airline: Vistara (UK-876)

Route: Bengaluru → Delhi

Date: 2025-07-02

Departure: 12:00

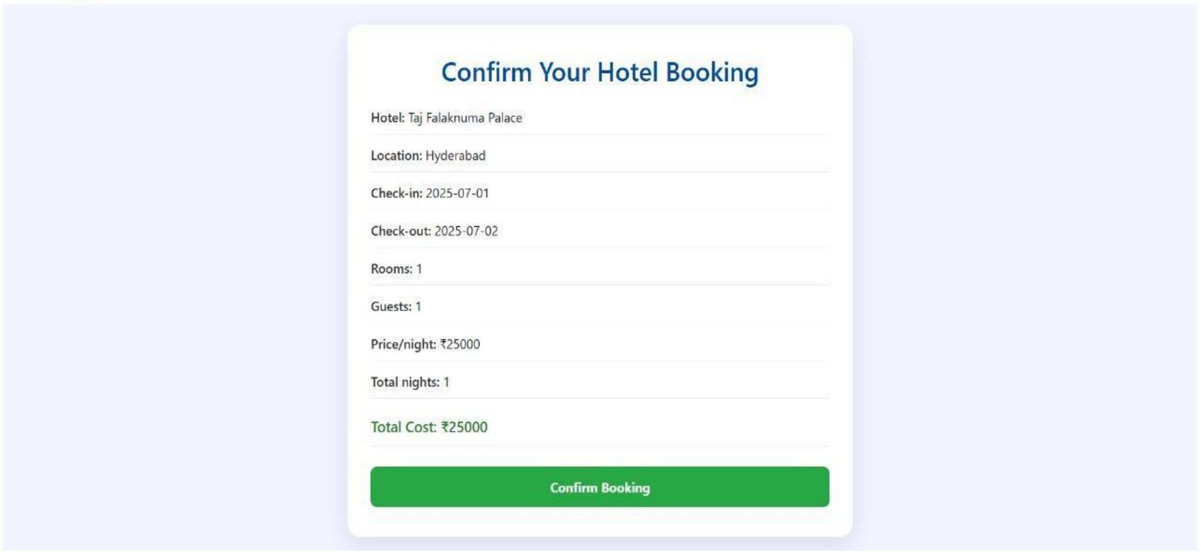
Arrival: 14:45

Passengers: 1

Price/person: ₹4500

Total Price: ₹4500

HOTEL BOOKING PAGE :



Confirm Your Hotel Booking

Hotel: Taj Falaknuma Palace

Location: Hyderabad

Check-in: 2025-07-01

Check-out: 2025-07-02

Rooms: 1

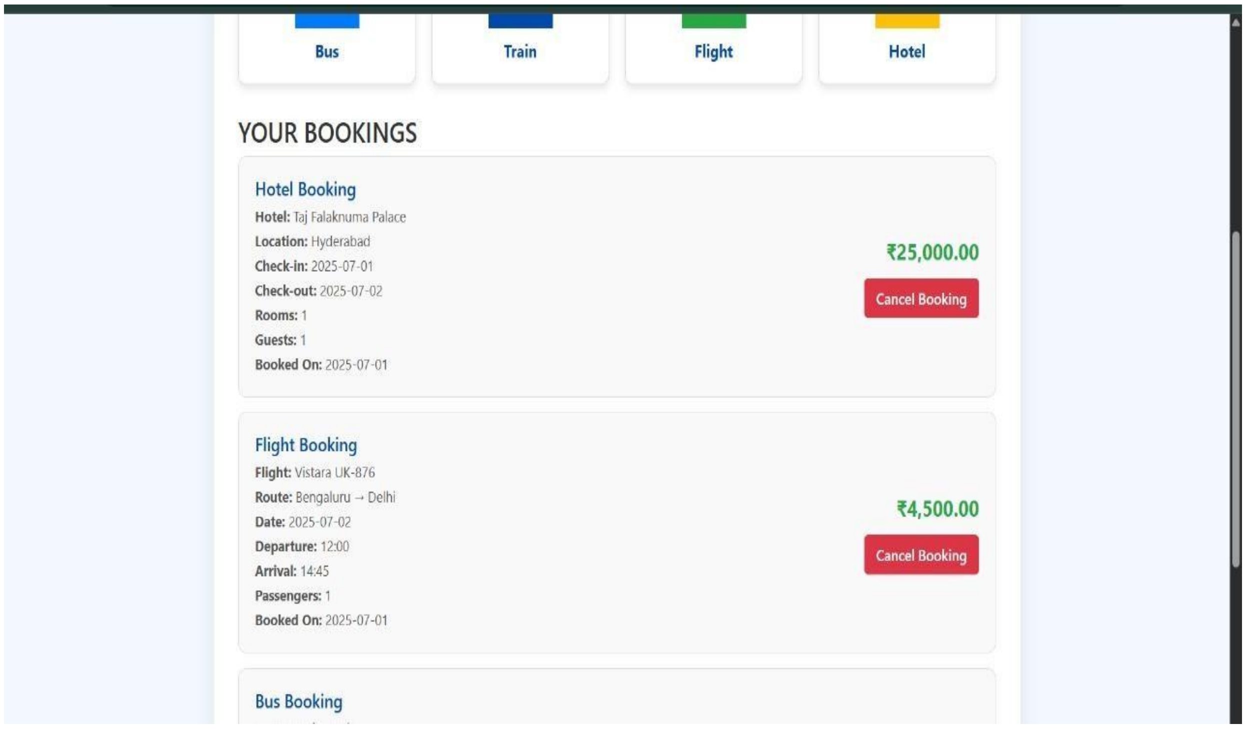
Guests: 1

Price/night: ₹25000

Total nights: 1

Total Cost: ₹25000

BOOKINGS :



YOUR BOOKINGS

Hotel Booking

Hotel: Taj Falaknuma Palace
Location: Hyderabad
Check-in: 2025-07-01
Check-out: 2025-07-02
Rooms: 1
Guests: 1
Booked On: 2025-07-01

₹25,000.00

Flight Booking

Flight: Vistara UK-876
Route: Bengaluru → Delhi
Date: 2025-07-02
Departure: 12:00
Arrival: 14:45
Passengers: 1
Booked On: 2025-07-01

₹4,500.00

Bus Booking

Conclusion :

The TravelGo Website has been successfully developed and deployed using a scalable and cloud-native architecture. Leveraging AWS services such as EC2 for hosting, DynamoDB for real-time data management, and SNS for instant booking and cancellation notifications, the platform provides a seamless travel booking experience for users. TravelGo enables registered users to search and book buses, trains, flights, and hotels in a centralized, intuitive interface, eliminating the complexities of navigating multiple travel services.

The cloud infrastructure ensures high availability and smooth performance even during peak usage, while the Flask backend ensures efficient handling of user authentication, dynamic booking flows, and data transactions. Real-time notification integration via AWS SNS allows users to receive booking confirmations and cancellations immediately via email, improving communication and user engagement.

In summary, the TravelGo Website offers a modern, reliable, and user-friendly solution for managing travel and accommodation needs. It highlights the potential of cloud-based platforms in building unified travel systems, simplifying operations, and enhancing the overall user experience.

……………………………………..THE END……………………………………