

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

Project Description:

TravelGo is a comprehensive, 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 for backend development, 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 various 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.

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

TravelGo offers a unified platform for booking buses, trains, flights, and hotels, allowing users to seamlessly plan entire trips. For instance, a user can log in, select their preferred transport, and then book accommodation, all within one interface. Flask efficiently manages real-time retrieval of diverse travel listings and processes user input. Hosted on AWS EC2, the platform remains highly responsive even during peak traffic, ensuring multiple users can browse and book without any delays, providing a truly integrated travel planning experience.

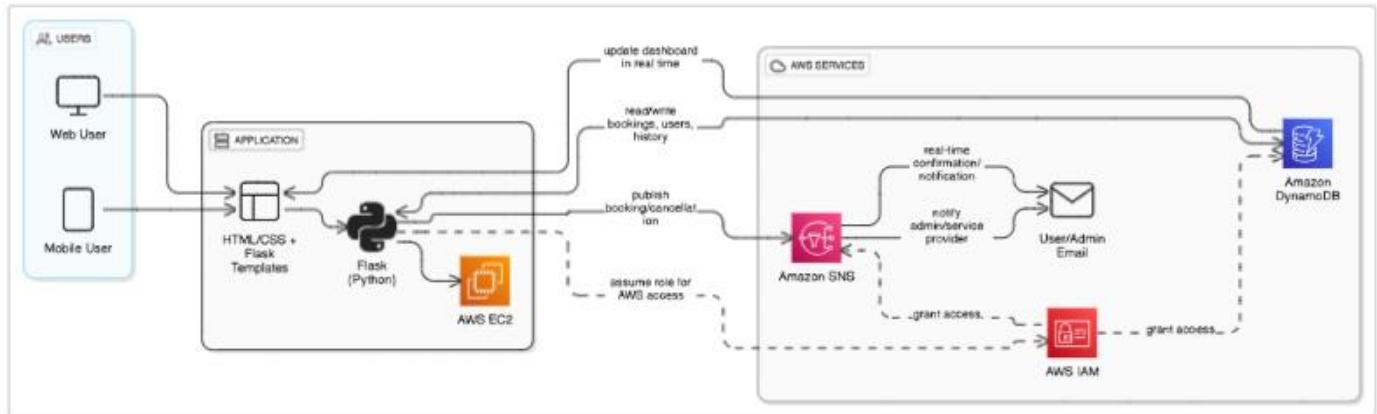
Scenario 2: Real-Time Booking Confirmation and Updates with AWS SNS

Upon successful booking, TravelGo instantly notifies the user using AWS SNS, sending real-time email confirmations with all relevant details. For example, after a student books a hotel, they immediately receive an email confirming their reservation. This notification is triggered from the Flask backend once the booking is securely recorded in DynamoDB. AWS SNS can also alert admin staff or service providers, ensuring transparency and real-time updates across all stakeholders for every transaction, from initial booking to any subsequent cancellations.

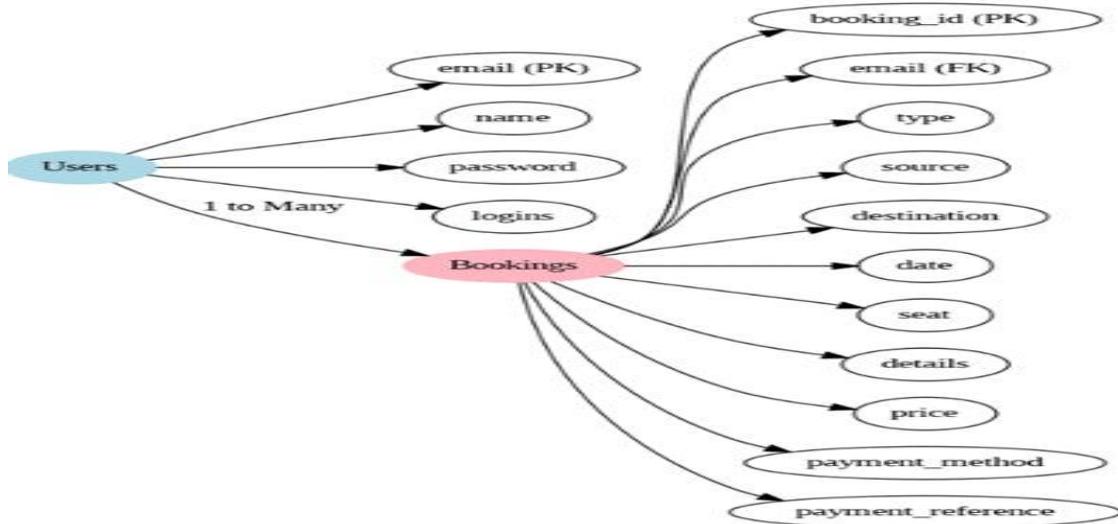
Scenario 3: Dynamic User Dashboard with Comprehensive Travel History & Management

TravelGo features a dynamic and personalized user dashboard that provides a comprehensive overview of all past and upcoming bookings. A logged-in user can easily view their flight and hotel reservations, categorized by type, along with essential details like dates and prices. Flask efficiently fetches this personalized data from AWS DynamoDB, which persistently stores all user bookings. The responsive UI, powered by HTML/CSS and Flask templates, ensures users can review, manage, and quickly cancel their bookings anytime, from any device, with real-time updates.

AWS ARCHITECTURE



Entity Relationship (ER)Diagram:



Pre-requisites:

1. **AWS Account Setup:** [AWS Account Setup](#)
2. **Understanding IAM:** [IAM Overview](#)
3. **Amazon EC2 Basics:** [EC2 Tutorial](#)
4. **DynamoDB Basics:** [DynamoDB Introduction](#)
5. **SNS Overview:** [SNS Documentation](#)
6. **Git Version Control:** [Git Documentation](#)

Project WorkFlow:

1. Backend Development and Application Setup

Activity 1.1: Develop the Backend Using Flask.

Activity 1.2: Integrate AWS Services Using boto3

2. AWS Account Setup and Login

Activity 2.1: AWS Account Setup and Login

Activity 2.2: Log in to the AWS Management Console.

3. DynamoDB Database Creation and Setup

Activity 3.1: Create a DynamoDB Table.

Activity 3.2: Configure Attributes for User Data and Book Requests.

4. SNS Notification Setup

Activity 4.1: Create SNS topics for book request notifications.

Activity 4.2: Subscribe users and library staff to SNS email notifications.

5. IAM Role Setup

Activity 5.1: Create IAM Role

Activity 5.2: Attach Policies

6. EC2 Instance Setup

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

Activity 6.2: Configure security groups for HTTP, and SSH access.

7. Deployment on EC2

Activity 7.1: Upload Flask Files

Activity 7.2: Run the Flask App

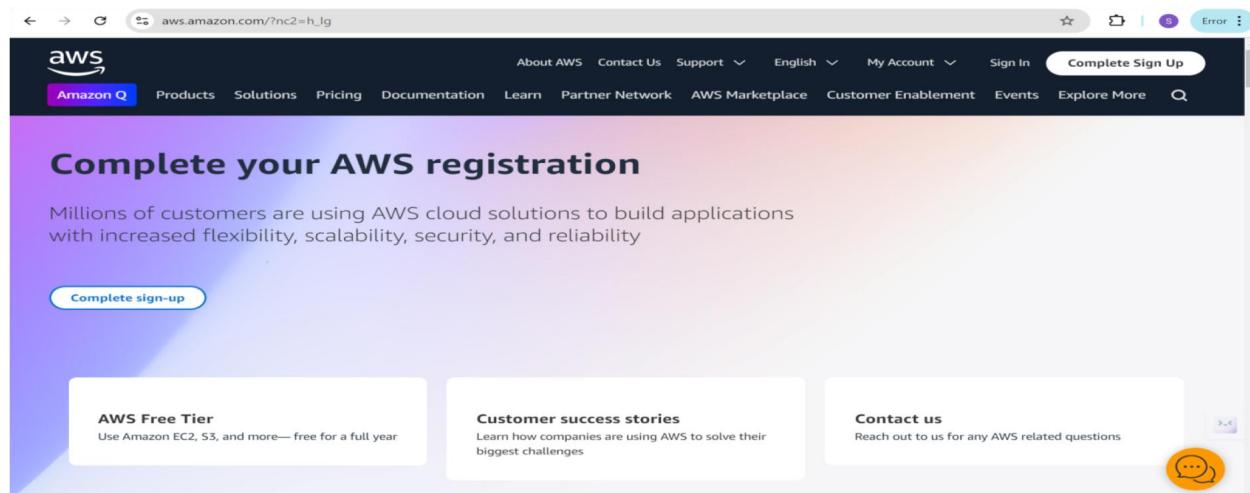
8. Testing and Deployment

Activity 8.1: Conduct functional testing to verify user registration, login, book requests, and notifications.

Milestone 1: AWS Account Setup and Login

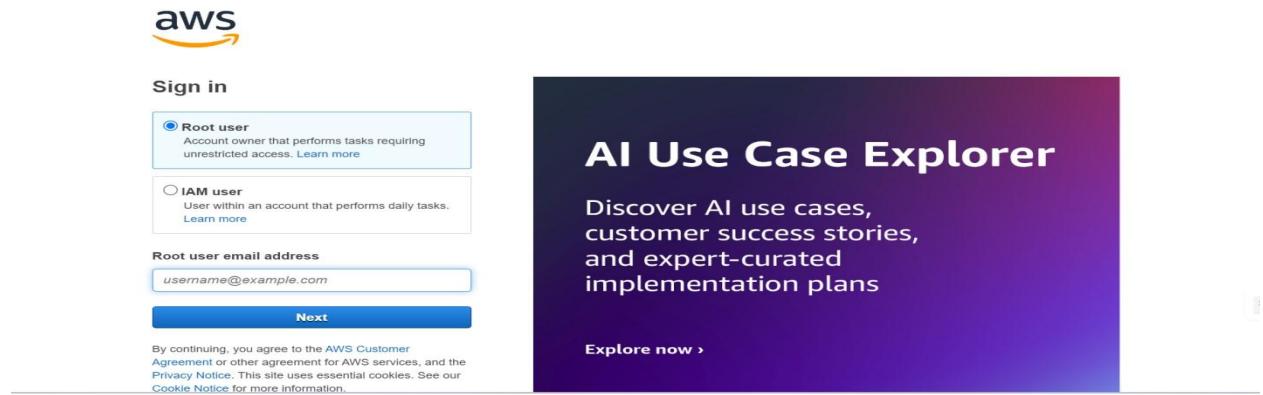
This is for your understanding only, please refrain from creating an AWS account. A temporary account will be provided via Troven.

- Go to the AWS website (<https://aws.amazon.com/>).
- 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.



- **Activity 1: Log in to the AWS Management Console**

- After setting up your account, log in to the [AWS Management Console](#).



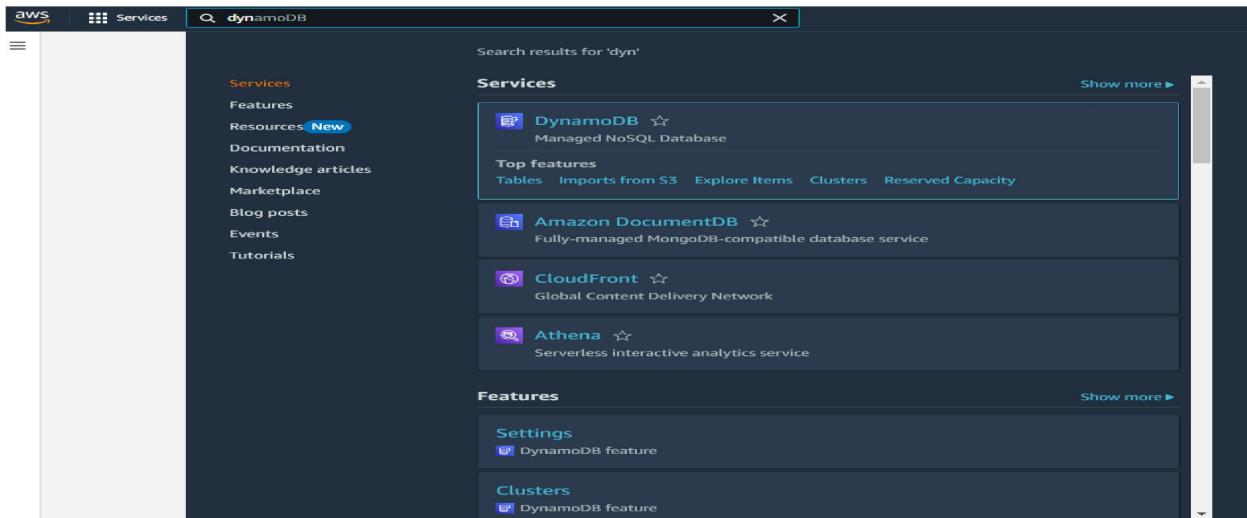
The image shows two side-by-side screenshots. On the left is the AWS Sign In page, featuring the AWS logo at the top, followed by a 'Sign in' form. It includes two radio button options: 'Root user' (selected) and 'IAM user'. Below these are fields for 'Root user email address' (containing 'username@example.com') and a 'Next' button. A small note at the bottom states: '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.' On the right is a dark purple banner titled 'AI Use Case Explorer'. The text reads: 'Discover AI use cases, customer success stories, and expert-curated implementation plans'. At the bottom of the banner is a blue 'Explore now >' button.

Milestone 2: DynamoDB Database Creation and Setup

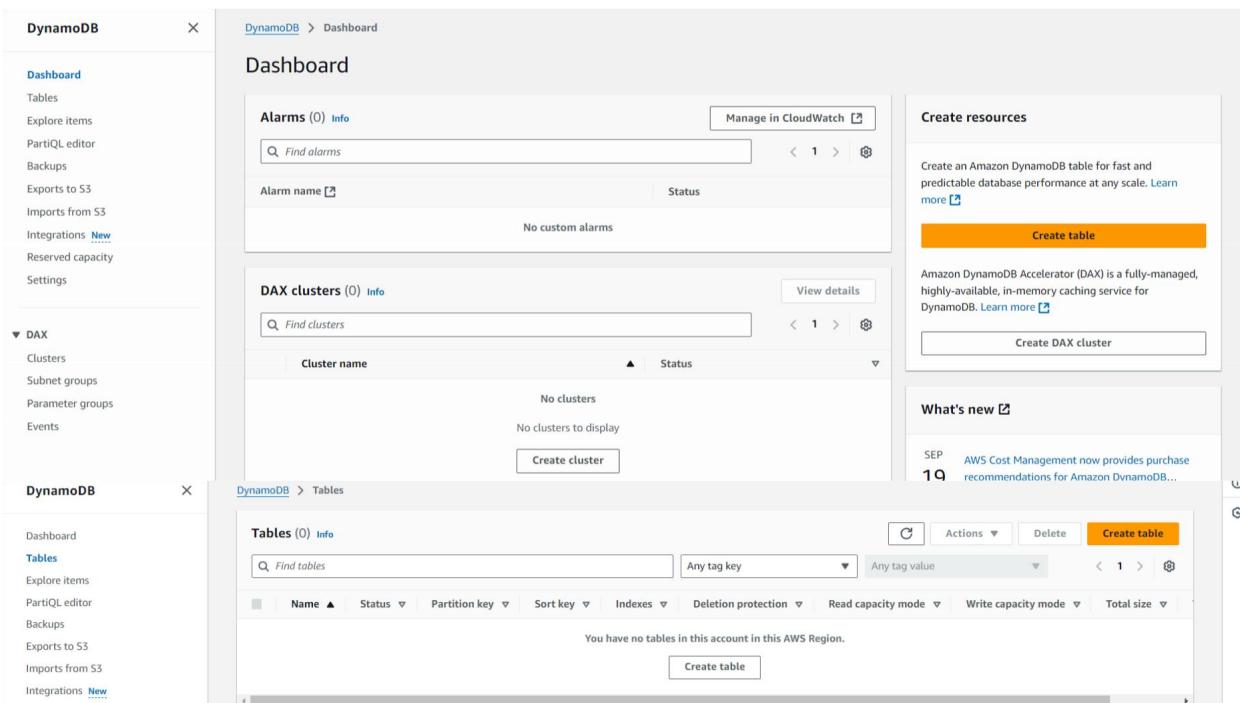
Database Creation and Setup involves initializing a cloud-based NoSQL database to store and manage application data efficiently. This step includes defining tables, setting primary keys, and configuring read/write capacities. It ensures scalable, high-performance data storage for seamless backend operations.

- **Activity 2.1: Navigate to the DynamoDB**

- In the AWS Console, navigate to DynamoDB and click on create tables.

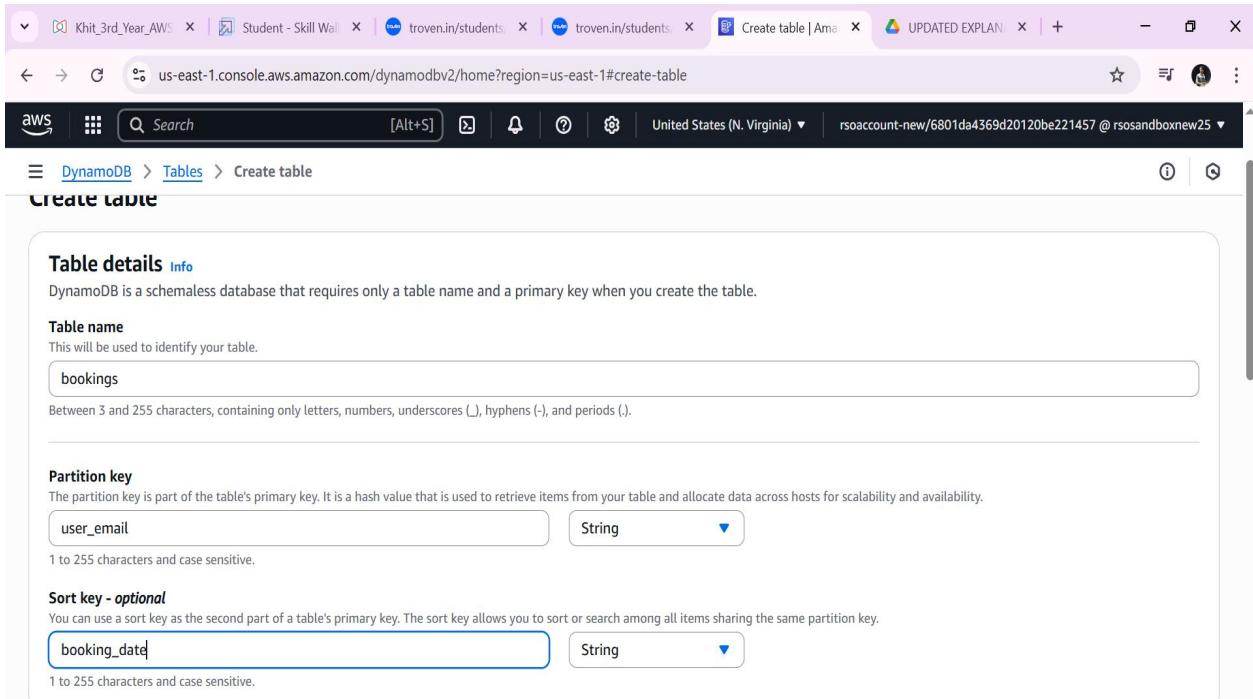


The image shows a screenshot of the AWS Services search interface. The search bar at the top contains the text 'dynamoDB'. The results are displayed under the 'Services' section. The first result is 'DynamoDB' (Managed NoSQL Database), which is highlighted with a blue border. Below it are other services: 'Amazon DocumentDB' (Fully-managed MongoDB-compatible database service), 'CloudFront' (Global Content Delivery Network), and 'Athena' (Serverless interactive analytics service). Further down the list are 'Features' (Settings, Clusters) and 'Clusters' (DynamoDB feature). On the left sidebar, there are links for 'Services', 'Features', 'Resources' (New), 'Documentation', 'Knowledge articles', 'Marketplace', 'Blog posts', 'Events', and 'Tutorials'.



The screenshot shows the AWS DynamoDB Dashboard. On the left, there are two tabs: "DynamoDB" and "DynamoDB". The "DynamoDB" tab is active, displaying the "Tables" section. It includes links for "Explore items", "PartiQL editor", "Backups", "Exports to S3", "Imports from S3", "Integrations", "Reserved capacity", and "Settings". Below these, under "DAX", there are links for "Clusters", "Subnet groups", "Parameter groups", and "Events". The main content area is titled "Dashboard" and contains sections for "Alarms" (0) and "DAX clusters" (0). The "Alarms" section has a search bar and a table with columns for "Alarm name" and "Status". The "DAX clusters" section also has a search bar and a table with columns for "Cluster name" and "Status". A "Create resources" sidebar on the right offers options to "Create table" or "Create DAX cluster". A "What's new" section at the bottom right mentions "AWS Cost Management now provides purchase recommendations for Amazon DynamoDB...".

● Activity 2.2: Create a DynamoDB table for storing registration details and book requests.



The screenshot shows the "Create table" wizard in the AWS DynamoDB console. The first step, "Table details", is completed. The table name is "bookings". The partition key is "user_email" of type "String". The sort key is "booking_date" of type "String". Both fields have a note indicating they can be between 1 and 255 characters and are case sensitive. At the bottom, there is a note about using a sort key as the second part of a primary key. The second step, "Sort key - optional", is partially visible below.

Khit_3rd_Year_AWS | Student - Skill Wall | troven.in/students | troven.in/students | List tables | Amazon | UPDATED EXPLAN | +

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

AWS Search [Alt+S] United States (N. Virginia) rsoaccount-new/6801da4369d20120be221457 @ rsosandboxnew25

DynamoDB > Tables

DynamoDB

- Dashboard
- Tables**
- Explore items
- PartiQL editor
- Backups
- Exports to S3
- Imports from S3
- Integrations **New**
- Reserved capacity
- Settings

DAX

- Clusters
- Subnet groups

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 2:25 PM 6/30/2025

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 X

The bookings table was created successfully. X

Tables (1) Info

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection
bookings	Active	user_email (\$)	booking_date (\$)	0	0	Off

Khit_3rd_Year_AWS | Student - Skill Wall | troven.in/students | troven.in/students | Create table | Amazon | UPDATED EXPLAN | +

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

AWS Search [Alt+S] United States (N. Virginia) rsoaccount-new/6801da4369d20120be221457 @ rsosandboxnew25

DynamoDB > Tables > 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.

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.

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.

String

Enter the sort key name
1 to 255 characters and case sensitive.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 2:25 PM 6/30/2025

Khit_3rd_Year_AWS | Student - Skill Wall | troven.in/students | troven.in/students | List tables | Amazon | UPDATED EXPLAN | +

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

AWS Search [Alt+S] United States (N. Virginia) rsoaccount-new/6801da4369d20120be221457 @ rsandboxnew25

DynamoDB > Tables

DynamoDB

- Dashboard
- Tables**
- Explore items
- PartiQL editor
- Backups
- Exports to S3
- Imports from S3
- Integrations **New**
- Reserved capacity
- Settings

DAX

- Clusters
- Subnet groups

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.

The travelgo_user table was created successfully.

Tables (2)

<input type="checkbox"/>	Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection
<input type="checkbox"/>	bookings	Active	user_email (\$)	booking_date (\$)	0	0	<input checked="" type="checkbox"/> Off
<input type="checkbox"/>	travelgo_user	Active	email (\$)	-	0	0	<input checked="" type="checkbox"/> Off

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

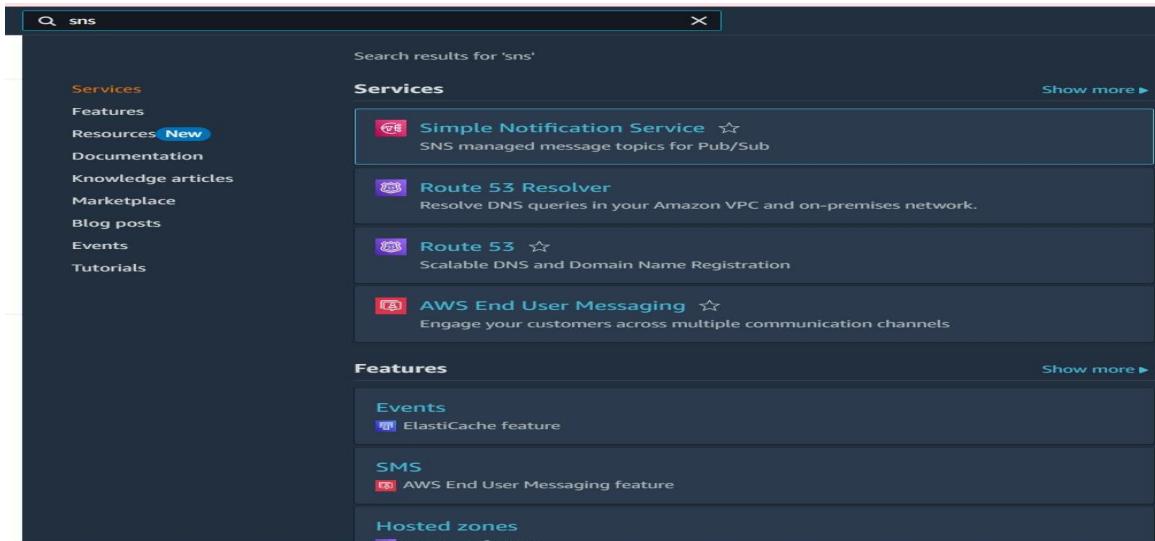
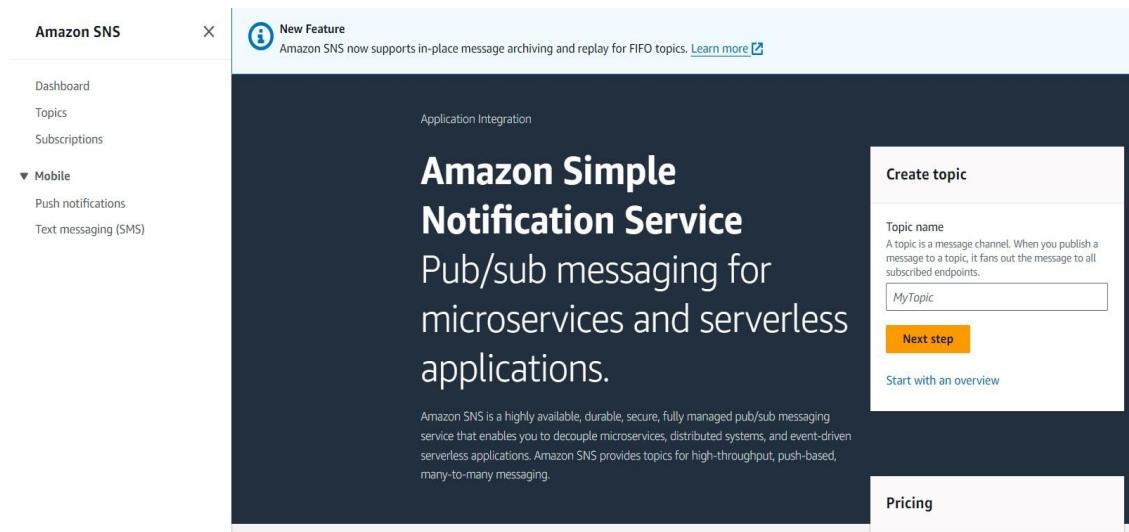
Type here to search

I created three dynamo Tables for my project they are bookings, trains and travelgo user .

Milestone 3: SNS Notification Setup

- **Activity 3.1: Create SNS topics for sending email notifications to users and library staff.**

- In the AWS Console, search for SNS and navigate to the SNS Dashboard.

The screenshot shows the Amazon SNS dashboard. On the left, there's a sidebar with links like Dashboard, Topics, Subscriptions, Mobile (Push notifications, Text messaging (SMS)), and Application Integration. The main content area features a 'New Feature' message about in-place message archiving and replay for FIFO topics. Below it, the text 'Amazon Simple Notification Service' is prominently displayed, followed by a description of its use for pub/sub messaging in microservices and serverless applications. At the bottom, there's a detailed description of what Amazon SNS is and how it works. On the right side, there's a 'Create topic' button and a 'Pricing' section.

- Click on **Create Topic** and choose a name for the topic
- Choose Standard type for general notification use cases and Click on Create Topic.

us-east-1.console.aws.amazon.com/sns/v3/home?region=us-east-1#/create-topic

AWS Search [Alt+S] United States (N. Virginia) rsosaccount-new/6801da4369d20120be221457 @rsoandboxnew25

Amazon SNS > Topics > Create topic

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.
My Topic

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

Type here to search

► **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.

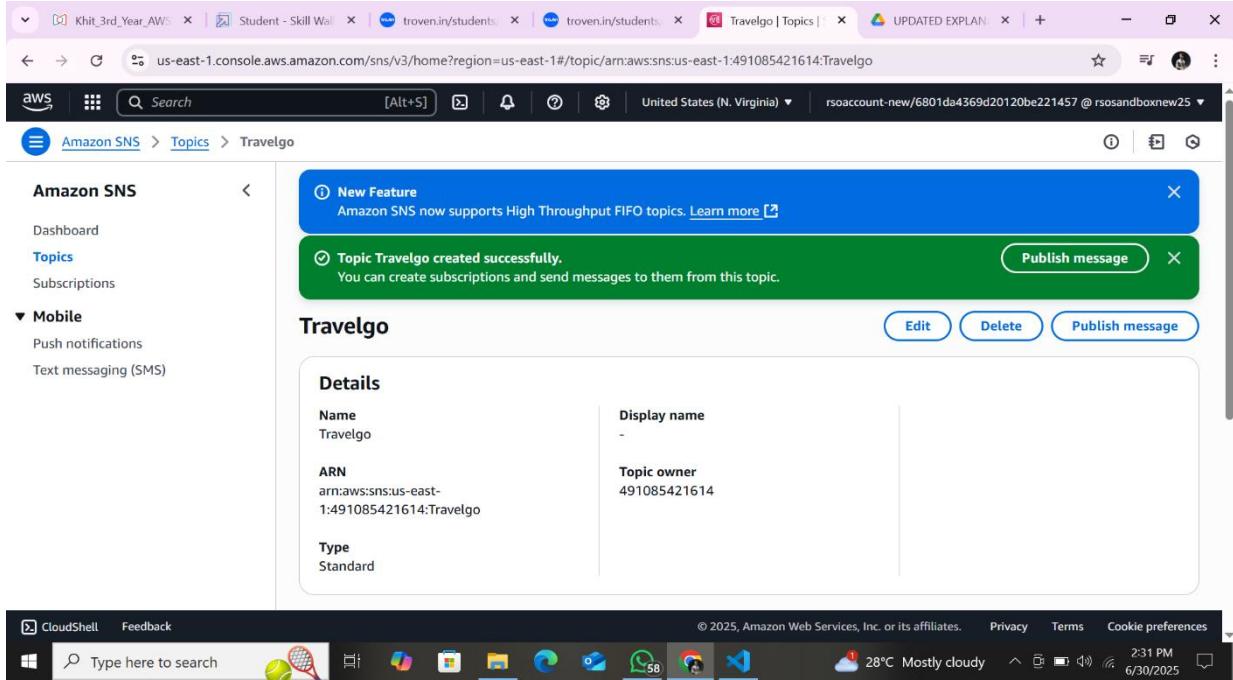
► **Delivery status logging - optional** [Info](#)
These settings configure the logging of message delivery status to CloudWatch Logs.

► **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**.



The screenshot shows the AWS SNS Topics page. On the left, there's a sidebar with 'Amazon SNS' navigation: Dashboard, Topics (which is selected and highlighted in blue), Subscriptions, and Mobile (Push notifications, Text messaging (SMS)). The main content area shows a green success message: 'Topic Travelgo created successfully. You can create subscriptions and send messages to them from this topic.' Below this, the 'Travelgo' topic is listed with its details: Name (Travelgo), Display name (-), ARN (arn:aws:sns:us-east-1:491085421614:Travelgo), Topic owner (491085421614), and Type (Standard). There are 'Edit', 'Delete', and 'Publish message' buttons. At the bottom of the page, there's a Windows taskbar with various icons and a system status bar showing '28°C Mostly cloudy', '2:31 PM', and '6/30/2025'.

- **Activity 3.2: Subscribe users and staff to relevant SNS topics to receive real-time notifications when a book request is made.**

- Subscribe users (or admin staff) to this topic via Email. When a book request is made, notifications will be sent to the subscribed emails.
- After subscription request for the mail confirmation
- Navigate to the subscribed Email account and Click on the confirm subscription in the AWS Notification- Subscription Confirmation mail.

AWS Notification - Subscription Confirmation Inbox x

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

9

You have chosen to subscribe to the topic:

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

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](#)

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

You have chosen to subscribe to the topic:

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

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:ap-south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4

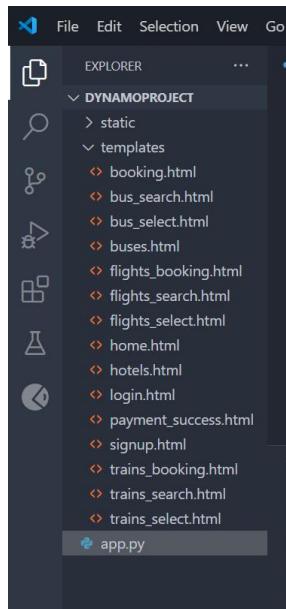
If it was not your intention to subscribe, [click here to unsubscribe](#).

- Successfully done with the SNS mail subscription and setup, now store the ARN link.

Milestone 4:Backend Development and Application Setup

- **Activity 4.1: Develop the backend using Flask**

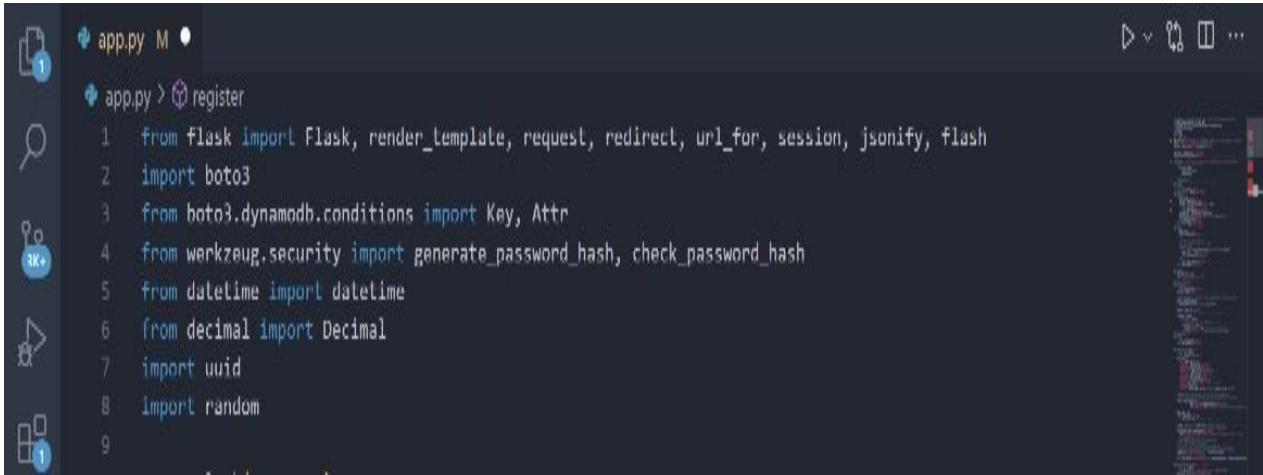
- File Explorer Structure



Description: set up the TRAVEL GO project with an app.py file, a static/ folder for assets, and a templates/ directory containing all required HTML pages like home, login, register, subject-specific pages (e.g. buse.html, hotel.html, train.html, flight.html).

Description of the code :

- **Flask App Initialization**



```

app.py M
app.py > register
1 from flask import Flask, render_template, request, redirect, url_for, session, jsonify, flash
2 import boto3
3 from boto3.dynamodb.conditions import Key, Attr
4 from werkzeug.security import generate_password_hash, check_password_hash
5 from datetime import datetime
6 from decimal import Decimal
7 import uuid
8 import random
9

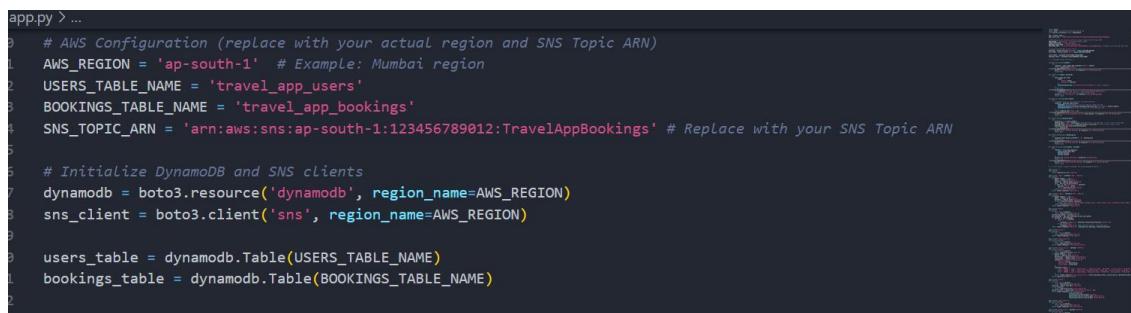
```

Description: import essential libraries including Flask utilities for routing, Boto3 for DynamoDB operations, SMTP and email modules for sending mails, and Bcrypt for password hashing and verification

```
app = Flask(__name__)
```

Description: initialize the Flask application instance using Flask(__name__) to start building the web app.

- **Dynamodb Setup:**



```

app.py > ...
# AWS Configuration (replace with your actual region and SNS Topic ARN)
AWS_REGION = 'ap-south-1' # Example: Mumbai region
USERS_TABLE_NAME = 'travel_app_users'
BOOKINGS_TABLE_NAME = 'travel_app_bookings'
SNS_TOPIC_ARN = 'arn:aws:sns:ap-south-1:123456789012:TravelAppBookings' # Replace with your SNS Topic ARN

# Initialize DynamoDB and SNS clients
dynamodb = boto3.resource('dynamodb', region_name=AWS_REGION)
sns_client = boto3.client('sns', region_name=AWS_REGION)

users_table = dynamodb.Table(USERS_TABLE_NAME)
bookings_table = dynamodb.Table(BOOKINGS_TABLE_NAME)

```

```

app = Flask(__name__)
app.secret_key = 'your_secret_key_here' # IMPORTANT: Change this to a strong, random key in production!
REGION = 'us-east-1' # Replace with your actual AWS region
dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
sns_client = boto3.client('sns', region_name='us-east-1')

users_table = dynamodb.Table('travelgo_users')
trains_table = dynamodb.Table('trains') # Note: This table is declared but not used in the provided routes,
bookings_table = dynamodb.Table('bookings')

```

Description: Initialize the DynamoDB resource for the us-east-1 region and set up access to the Users and Requests tables for storing user details and book requests.

- **SNS Connection**

```

SNS_TOPIC_ARN = 'arn:aws:sns:us-east-1:976193227152:Travelgo:50a3ac4c-1230-4659-91e2-99aac2bab817'
def send sns_notification(subject, message):
    try:
        sns_client.publish(
            TopicArn=SNS_TOPIC_ARN,
            Subject=subject,
            Message=message
        )
    except Exception as e:
        print(f"SNS Error: Could not send notification - {e}")

```

Description: Configure SNS to send notifications when a book request is submitted. Paste your stored ARN link in the sns_topic_arn space, along with the region name where the SNS topic is created. Also, specify the chosen email service in SMTP_SERVER (e.g., Gmail, Yahoo, etc.) and enter the subscribed email in the SENDER_EMAIL section. Create an 'App password' for the email ID and store it in the SENDER_PASSWORD section.

- **Routes for Web Pages**

- **Home Route:**

```

@app.route('/home')
def home():
    if 'user' not in session:
        return redirect(url_for('login'))
    current_user_phone = session['user']
    user_bookings = get_user_bookings(current_user_phone)
    for booking in user_bookings:
        if 'booked_at' in booking:
            try:
                booking['booked_at'] = datetime.fromisoformat(booking['booked_at'])
            except ValueError:
                booking['booked_at'] = None # Handle potential malformed dates
    return render_template('home.html', bookings=user_bookings, datetime=datetime)

```

Description: define the home route / to automatically redirect users to the register page when they access the base URL.

- **Register Route:**

```

@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        phone = request.form['phone']
        password = request.form['password']
        if not create_user(phone, password):
            return "User with this phone number already exists. Please login or use a different phone number."
        return redirect(url_for('login'))
    return render_template('signup.html')

```

Description: define /register route to validate registration form fields, hash the user password using Bcrypt, store the new user in DynamoDB with a login count, and send an SNS notification on successful registration

- **login Route (GET/POST):**

```

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        phone = request.form['phone']
        password = request.form['password']
        user = get_user_by_phone(phone)
        if user and user['password'] == password:
            session['user'] = phone
            return redirect(url_for('home'))
        return "Invalid credentials"
    return render_template('login.html')

```

Description: define /login route to validate user credentials against DynamoDB, check the password using Bcrypt, update the login count on successful authentication, and redirect users to the home page

- Train , Bus and Hotel routes:

```

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

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

@app.route('/trains_select', methods=['POST'])
def trains_select():
    if 'user' not in session:
        return redirect(url_for('login'))
    if request.method == 'POST':
        source = request.form['source']
        destination = request.form['destination']
        travel_date = request.form['travel_date']
        session['train_search_criteria'] = {
            'source': source,
            'destination': destination,
            'travel_date': travel_date
        }
    available_trains = [
        {'id': 'TRN001', 'name': 'Express Rail', 'departure_time': '06:00 AM', 'arrival_time': '10:00 AM', 'price': 800, 'class_options': ['Sleeper', 'AC3']},
        {'id': 'TRN002', 'name': 'City Link', 'departure_time': '09:30 AM', 'arrival_time': '01:30 PM', 'price': 750, 'class_options': ['Sleeper', 'AC2']},
        {'id': 'TRN003', 'name': 'Night Rider', 'departure_time': '08:00 PM', 'arrival_time': '06:00 AM', 'price': 1200, 'class_options': ['AC1', 'AC2', 'AC3']}
    ]
    return render_template('trains_select.html', trains=available_trains, source=source, destination=destination, travel_date=travel_date)
    return redirect(url_for('home'))

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

```

Description: define /home-page to render the main homepage, /ebook-buttons to handle subject selection and redirection, and /<subject>.html dynamic route to render subject-specific pages like Mathematics or English.

- Request Routes:

```

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

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

@app.route('/trains_select', methods=['POST'])
def trains_select():
    if 'user' not in session:
        return redirect(url_for('login'))
    if request.method == 'POST':
        source = request.form['source']
        destination = request.form['destination']
        travel_date = request.form['travel_date']
        session['train_search_criteria'] = {
            'source': source,
            'destination': destination,
            'travel_date': travel_date
        }
    available_trains = [
        {'id': 'TRN001', 'name': 'Express Rail', 'departure_time': '06:00 AM', 'arrival_time': '10:00 AM', 'price': 800, 'class_options': ['Sleeper', 'AC3']},
        {'id': 'TRN002', 'name': 'City Link', 'departure_time': '09:30 AM', 'arrival_time': '01:30 PM', 'price': 750, 'class_options': ['Sleeper', 'AC2']},
        {'id': 'TRN003', 'name': 'Night Rider', 'departure_time': '08:00 PM', 'arrival_time': '06:00 AM', 'price': 1200, 'class_options': ['AC1', 'AC2', 'AC3']}
    ]
    return render_template('trains_select.html', trains=available_trains, source=source, destination=destination, travel_date=travel_date)
    return redirect(url_for('home'))

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

```

Description: define /request-form route to capture book request details from users, store the request in DynamoDB, send a thank-you email to the user, notify the admin, and confirm submission with a success message.

Deployment Code:

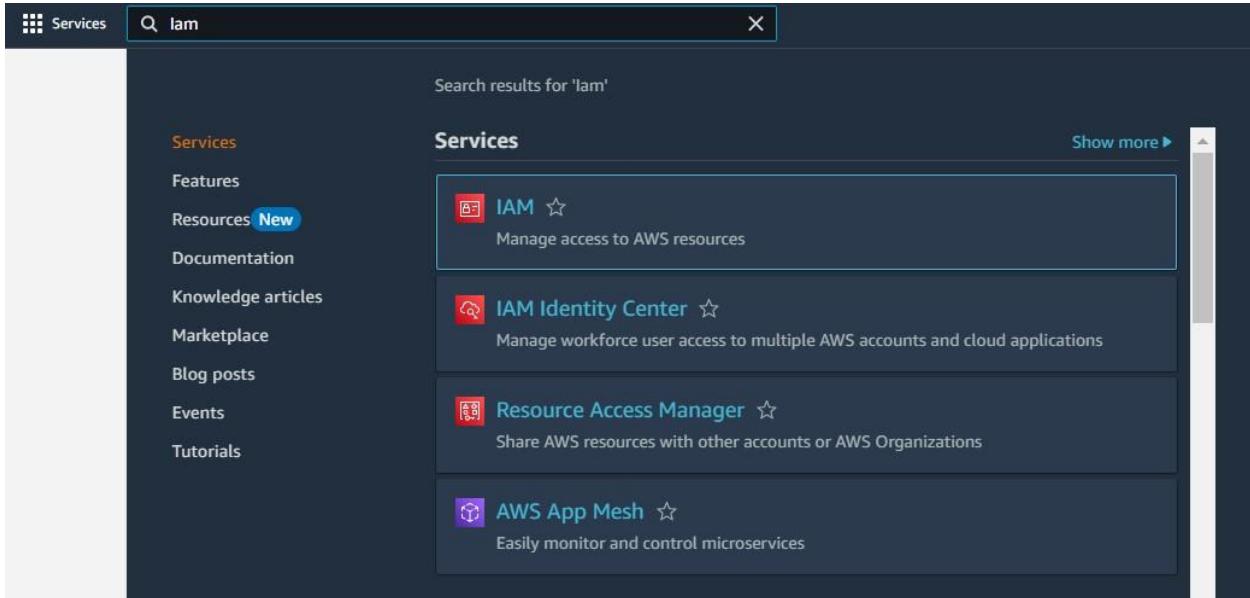
```
if __name__ == "__main__":
|   app.run(host='0.0.0.0', port=80, debug=True)
```

Description: start the Flask server to listen on all network interfaces (0.0.0.0) at port 80 with debug mode enabled for development and testing.

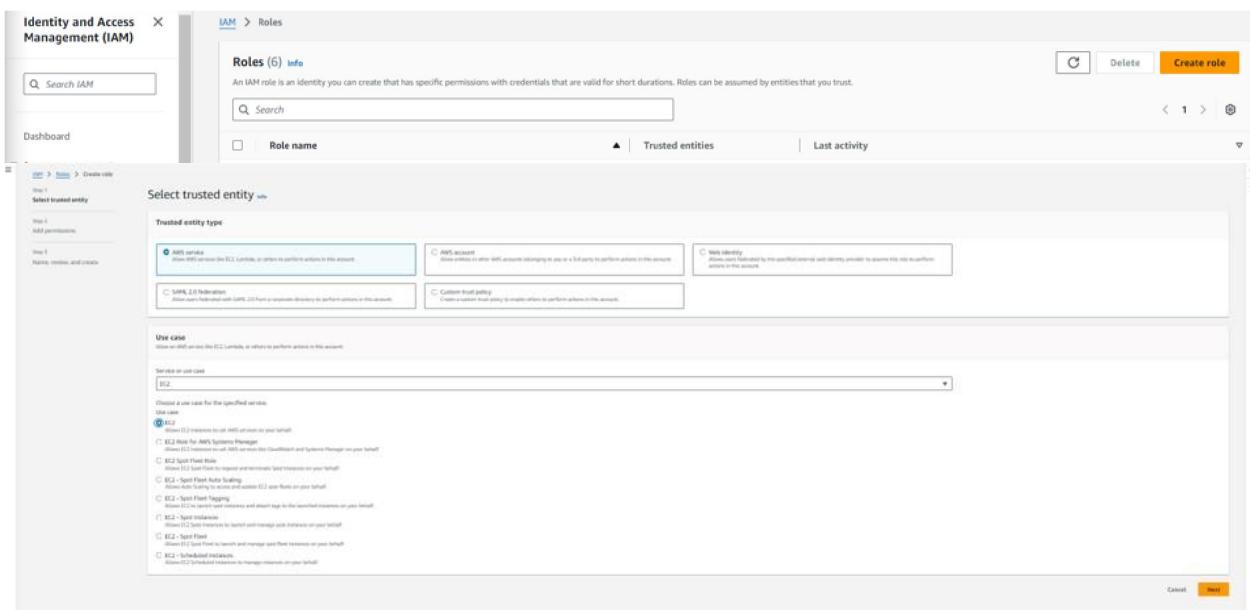
Milestone 5: IAM Role Setup

• Activity 5.1: Create IAM Role.

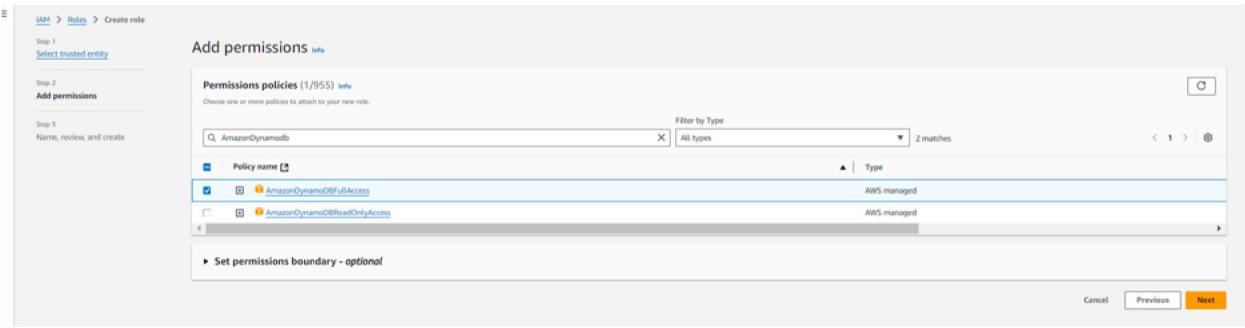
- In the AWS Console, go to IAM and create a new IAM Role for EC2 to interact with DynamoDB and SNS.



The screenshot shows the AWS Services search results for 'iam'. The search bar at the top contains 'iam'. Below it, there is a sidebar with various links: Services, Features, Resources (New), Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main content area displays four services: IAM, IAM Identity Center, Resource Access Manager, and AWS App Mesh. Each service has a small icon, a name, a star rating, and a brief description.



The screenshot shows the 'Create role' wizard in the IAM service. The first step, 'Select trusted entity', is displayed. It includes sections for 'Trusted entity type' (AWS service, AWS account, IAM role or federated user, Custom trust policy) and 'Use case'. Under 'Use case', the 'EC2 - Amazon EC2 can invoke the AWS Lambda function on your behalf' option is selected. Other options include 'Amazon RDS for MySQL', 'Amazon RDS for MariaDB', 'Amazon RDS for Oracle', 'Amazon RDS for PostgreSQL', 'Amazon RDS for Amazon Aurora', 'Amazon RDS for Amazon Neptune', 'Amazon RDS Auto Scaling', 'Amazon Kinesis Data Stream', 'Amazon Kinesis Data Firehose', 'Amazon Kinesis Video Stream', 'Amazon Kinesis Video Processor', and 'Amazon Kinesis Video Analytics'. At the bottom right of the wizard, there are 'Cancel' and 'Next Step' buttons.



IAM > Roles > Create role

Step 1 Select trusted entity

Step 2 Add permissions

Step 3 Name, review, and create

Add permissions

Permissions policies (1/955) [Info](#)

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

Filter by Type: All types | 2 matches

Policy name: AmazonDynamoDB

- AmazonDynamoDBFullAccess** AWS managed
- AmazonDynamoDBReadOnlyAccess** AWS managed

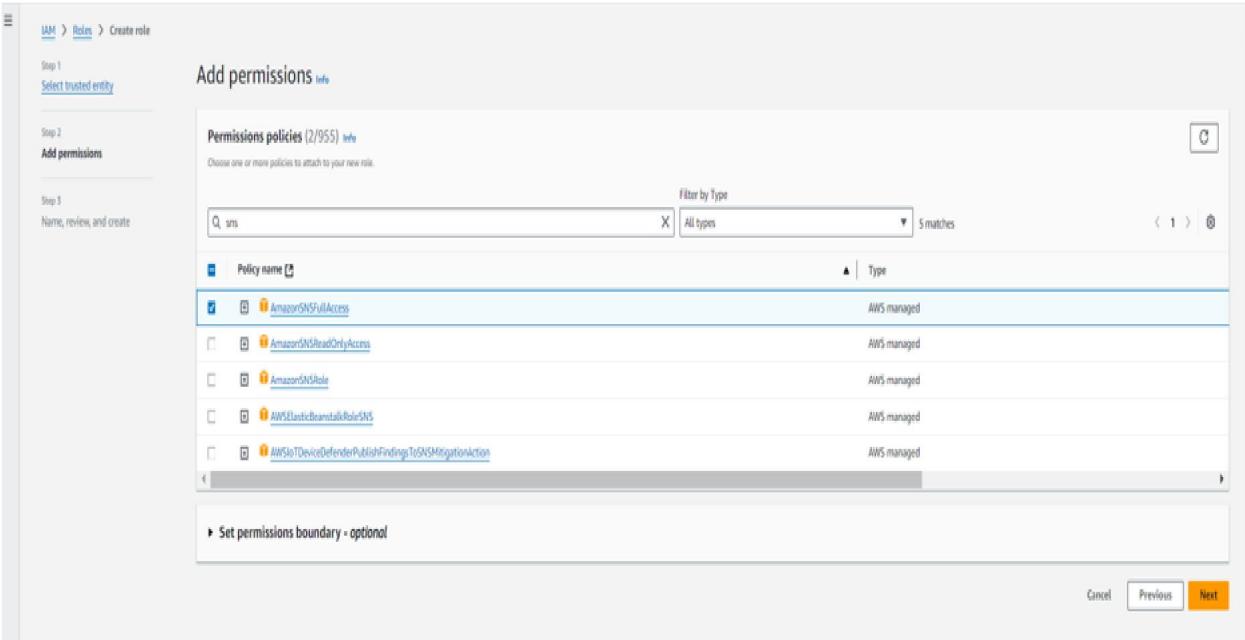
Set permissions boundary - optional

Cancel Previous Next

● Activity 5.2: Attach Policies.

Attach the following policies to the role:

- **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

Permissions policies (2/955) [Info](#)

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

Filter by Type: All types | 5 matches

Policy name: sns

- AmazonSNSFullAccess** AWS managed
- AmazonSNSReadOnlyAccess** AWS managed
- AmazonSNSRole** AWS managed
- AWSLambdaBasicExecutionRole** AWS managed
- AWSIoTDevice DefenderPublishFindingsToSNSMigrationACTION** AWS managed

Set permissions boundary - optional

Cancel Previous Next

Name, review, and create

Role details

Role name
 Enter a meaningful name to identify this role.

Maximum 50 characters. Use underscores and -_.(.-)_. characters.

Description
 Add a description for this role.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: ~!@#\$%^&*()_-+=`{|}~`

Step 1: Select trusted entities

Trust policy

```
1: { "Version": "2012-10-17", 2: "Statement": [ 3: { "Effect": "Allow", 4: "Principal": "AWS", 5: "Action": "sts:AssumeRole" } ] }
```

Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached as
AmazonDynamoDBFullAccess	AWS managed	Policies policy
AmazonSNSFullAccess	AWS managed	Policies policy

Step 3: Add tags

Add tags - optional Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

[Add new tag](#) You can add up to 50 more tags.

[Cancel](#) [Previous](#) [Next step](#)

IAM > Roles > sns_Dynamodb_role

sns_Dynamodb_role [Info](#)

Allows EC2 instances to call AWS services on your behalf.

Summary

Creation date	ARN	Instance profile ARN
October 13, 2024, 23:06 (UTC+05:30)	arn:aws:iam::557690616836:role/sns_Dynamodb_role	arn:aws:iam::557690616836:instance-profile/sns_Dynamodb_role
Last activity	Maximum session duration	
6 days ago	1 hour	

[Edit](#)

[Permissions](#) [Trust relationships](#) [Tags](#) [Last Accessed](#) [Revoke sessions](#)

Permissions policies (2) [Info](#)

You can attach up to 10 managed policies.

Policy name		Type	Attached entities
<input type="checkbox"/>	AmazonDynamoDBFullAccess	AWS managed	4
<input type="checkbox"/>	AmazonSNSFullAccess	AWS managed	2

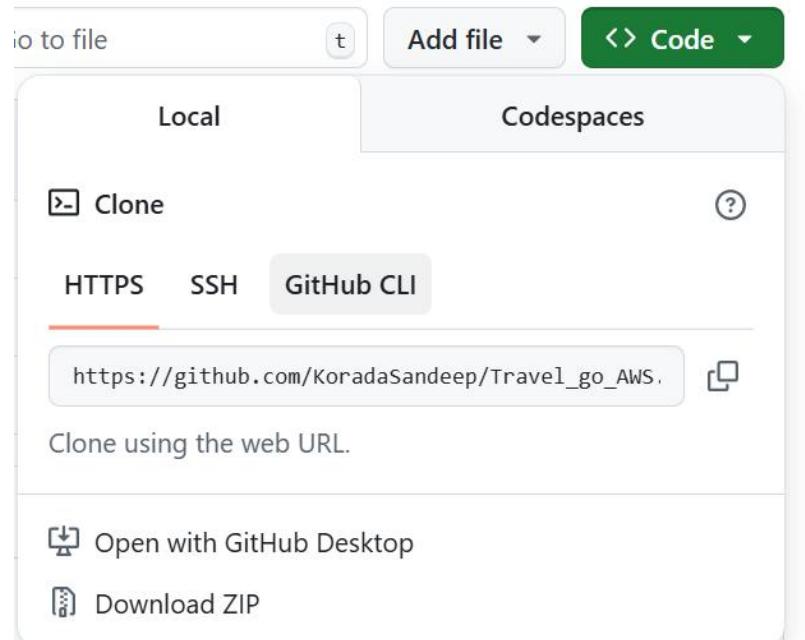
Filter by Type: All types

[Search](#) [Simulate](#) [Remove](#) [Add permissions](#)

Milestone 6: EC2 Instance Setup

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

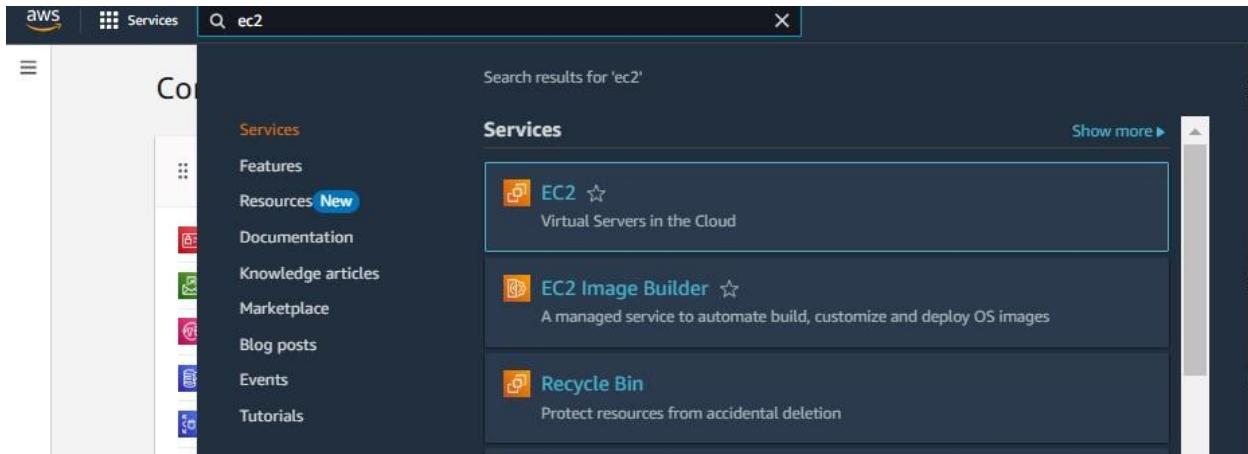
 static	Initial commit
 templates	Update statistics.html
 app.py	Update app.py



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

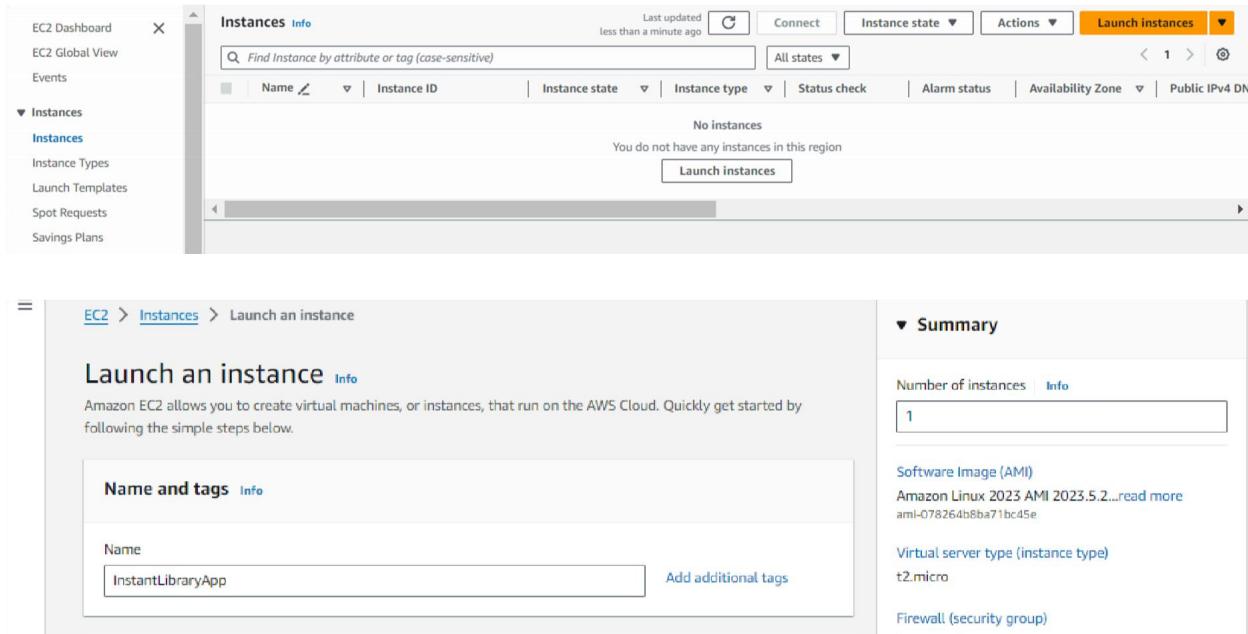
- **Launch EC2 Instance**

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



The screenshot shows the AWS CloudSearch interface. The search bar at the top contains the query 'ec2'. Below the search bar, there is a sidebar with links to 'Services', 'Features', 'Resources New', 'Documentation', 'Knowledge articles', 'Marketplace', 'Blog posts', 'Events', and 'Tutorials'. The main content area displays search results for 'ec2' under the heading 'Services'. The first result is 'EC2' with the subtext 'Virtual Servers in the Cloud'. The second result is 'EC2 Image Builder' with the subtext 'A managed service to automate build, customize and deploy OS images'. The third result is 'Recycle Bin' with the subtext 'Protect resources from accidental deletion'. A 'Show more ▶' link is located in the top right corner of the results list.

- Click on Launch instance to launch EC2 instance



The screenshot shows the AWS EC2 Instances page and the 'Launch an instance' wizard.

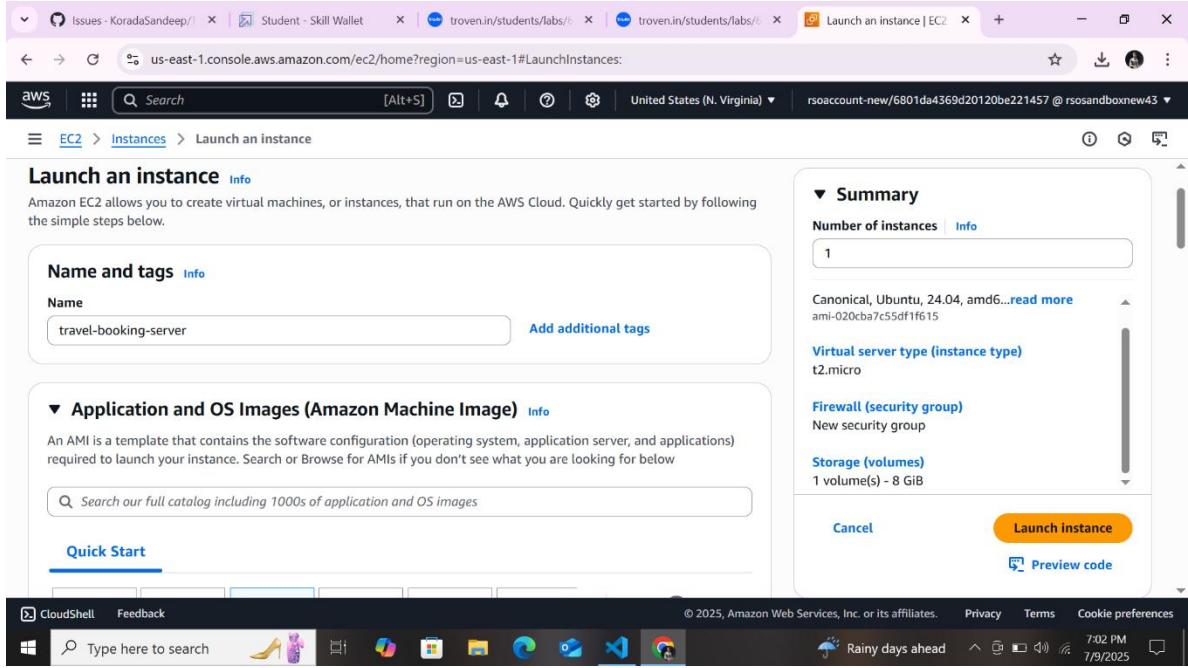
EC2 Instances Page:

- The left sidebar shows navigation options: EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instances Types, Launch Templates, Spot Requests, and Savings Plans.
- The main content area is titled 'Instances Info' and shows a table header with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS.
- A message at the bottom states: "No instances" and "You do not have any instances in this region".
- A prominent 'Launch instances' button is located at the bottom right of the table area.

'Launch an instance' Wizard:

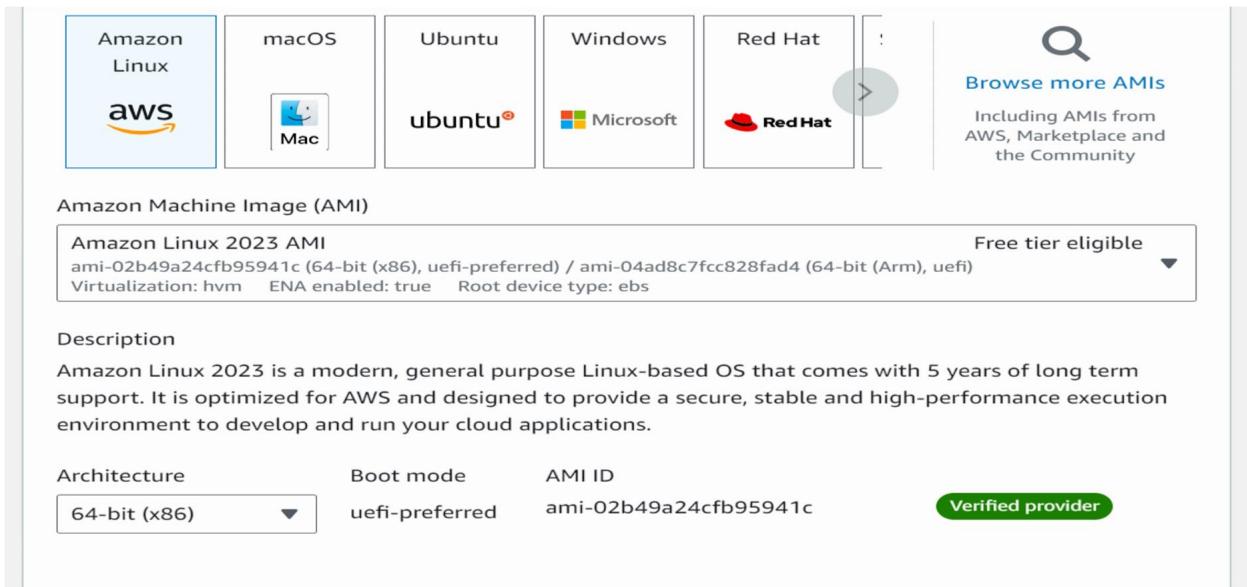
- The title bar shows the path: EC2 > Instances > Launch an instance.
- The main section is titled 'Launch an instance' with a sub-link 'Info'.
- The text below says: "Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below."
- A 'Name and tags' section contains a 'Name' input field with the value 'InstantLibraryApp' and a 'Add additional tags' link.
- To the right, a 'Summary' section includes:
 - Number of instances:
 - Software Image (AMI): Amazon Linux 2023.5.2...read more
 - Virtual server type (instance type): t2.micro
 - Firewall (security group): (not specified)

- Choose Amazon Linux 2



The screenshot shows the 'Launch an instance' wizard in the AWS Management Console. The instance type is set to 't2.micro' and the AMI is 'Canonical, Ubuntu, 24.04, amd64'. The storage volume is 8 GB. The 'Launch instance' button is highlighted.

the AMI and t2.micro as the instance type (free-tier eligible).



The screenshot shows the AWS Marketplace search results for 'Ubuntu'. It lists several options: Amazon Linux, macOS, Ubuntu, Windows, and Red Hat. A search bar at the top right says 'Search' and 'Browse more AMIs'. Below the search bar, it says 'Including AMIs from AWS, Marketplace and the Community'.

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI	Free tier eligible
ami-02b49a24cfb95941c (64-bit (x86), uefi-preferred) / ami-04ad8c7fcc828fad4 (64-bit (Arm), uefi) Virtualization: hvm ENA enabled: true Root device type: ebs	▼

Description

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Architecture 64-bit (x86)	Boot mode uefi-preferred	AMI ID ami-02b49a24cfb95941c	Verified provider
------------------------------	-----------------------------	---------------------------------	-------------------

- Create and download the key pair for Server access.

Issues - KoradaSandeep/ | Student - Skill Wallet | troven.in/students/labs/6 | troven.in/students/labs/6 | Launch an instance | EC2 | +

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

aws | Search [Alt+S] | United States (N. Virginia) | rsaccount-new/6801da4369d20120be221457 @ rsosandboxnew43

EC2 > Instances > Launch an instance

Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro

Family: t2 1 vCPU 1 GiB Memory Current generation: true

All generations

[Compare instance types](#)

Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

travel_go

[Create new key pair](#)

Summary

Number of instances [Info](#)

1

Canonical, Ubuntu, 24.04, amd64... [read more](#)
ami-020cba7c55df1f615

Virtual server type (instance type)
t2.micro

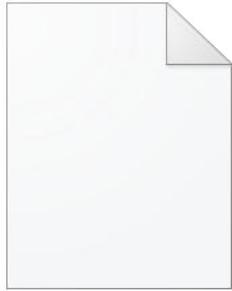
Firewall (security group)
New security group

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

[Cancel](#) [Launch instance](#) [Preview code](#)

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

Type here to search Rainy days ahead 7:02 PM 7/9/2025



travel_go.pem

- **Activity 6.2:Configure security groups for HTTP, and SSH access.**

Issues · KoradaSandeep/ · Student - Skill Wallet · troven.in/students/labs/ · troven.in/students/labs/ · Launch an instance | EC2 · + · - · □ · X

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

aws Search [Alt+S] United States (N. Virginia) rsoaccount-new/6801da4369d20120be221457 @ rsosandboxnew43

EC2 > Instances > Launch an instance

Type | Info

Protocol | Info

Port range | Info

ssh	TCP	22
-----	-----	----

Source type | Info

Source | Info

Description - optional | Info

Anywhere	Add CIDR, prefix list or security	e.g. SSH for admin desktop
0.0.0.0/0		

▼ Security group rule 2 (TCP; 80, 0.0.0.0/0)

Remove

▼ Summary

Number of instances | Info

1

Canonical, Ubuntu, 24.04, amd64... [read more](#)

ami-020cba7c55df1f615

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

Cancel

Launch instance

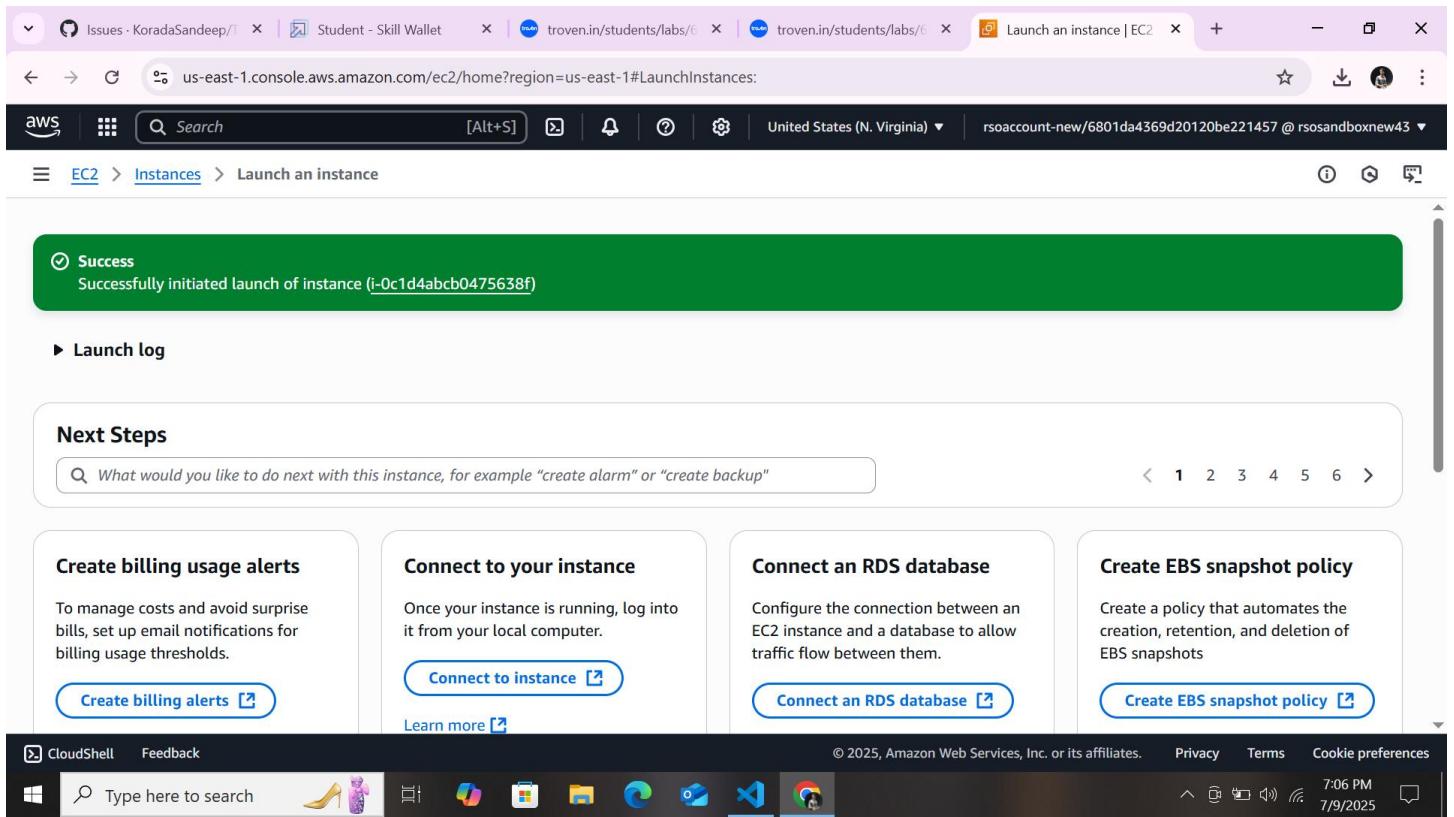
Preview code

⚠ 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.

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

Type here to search

Rainy days ahead 7:02 PM 7/9/2025



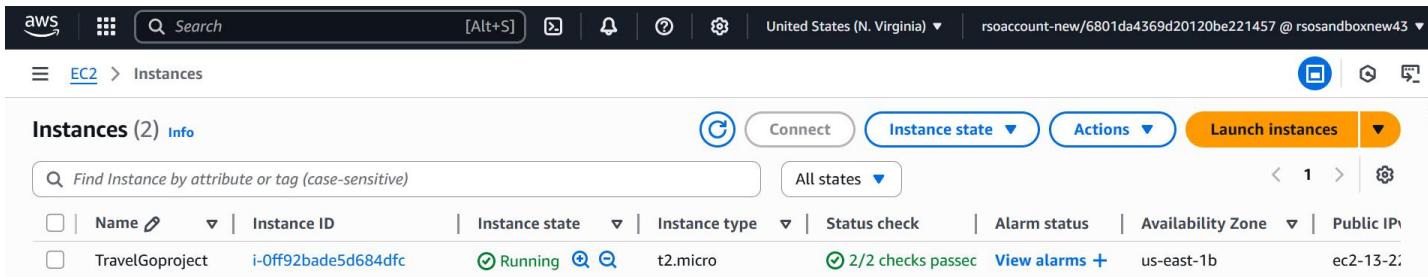
The screenshot shows the AWS Management Console EC2 service. A green success message box at the top states: "Success: Successfully initiated launch of instance (i-0c1d4abcb0475638f)". Below this, there is a "Launch log" section and a "Next Steps" summary.

Next Steps:

- Create billing usage alerts**: To manage costs and avoid surprise bills, set up email notifications for billing usage thresholds. Includes a "Create billing alerts" button.
- Connect to your instance**: Once your instance is running, log into it from your local computer. Includes a "Connect to instance" button and a "Learn more" link.
- Connect an RDS database**: Configure the connection between an EC2 instance and a database to allow traffic flow between them. Includes a "Connect an RDS database" button.
- Create EBS snapshot policy**: Create a policy that automates the creation, retention, and deletion of EBS snapshots. Includes a "Create EBS snapshot policy" button.

At the bottom, the Windows taskbar is visible, showing the Start button, search bar, pinned icons (File Explorer, Edge, File History, Task View, Mail, Photos, OneDrive), and system status indicators (Wi-Fi, battery, date/time).

- To connect to EC2 using **EC2 Instance Connect**, start by ensuring that an **IAM role** is attached to your EC2 instance. You can do this by selecting your instance, clicking on **Actions**, then navigating to **Security** and selecting **Modify IAM Role** to attach the appropriate role. After the IAM role is connected, navigate to the **EC2** section in the **AWS Management Console**. Select the **EC2 instance** you wish to connect to. At the top of the **EC2 Dashboard**, click the **Connect** button. From the connection methods presented, choose **EC2 Instance Connect**. Finally, click **Connect** again, and a new browser-based terminal will open, allowing you to access your EC2 instance directly from your browser.



The screenshot shows the AWS Management Console EC2 service. The "Instances" table lists one instance:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
TravelGoproject	i-0ff92bade5d684dfc	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-13-2-

i-Off92bade5d684dfc (TravelGoproject)
⚙️ | >

[Details](#) | [Status and alarms](#) | [Monitoring](#) | [Security](#) | [Networking](#) | [Storage](#) | [Tags](#)
▼ Instance summary [Info](#)
Instance ID
 i-Off92bade5d684dfc

Public IPv4 address
 13.220.213.173 | [open address](#) 
Private IPv4 addresses
 172.31.94.144

IPv6 address
 -

Instance state
 Running

Public DNS

 ec2-13-220-213-173.compute-
 1.amazonaws.com
 [open address](#) 
Hostname type
 IP name: ip-172-31-94-144.ec2.internal

Private IP DNS name (IPv4 only)
 ip-172-31-94-144.ec2.internal

KoradaSan | Student - | troven.in/s | troven.in/s | Modify IAM Role | EC2 Instances | Google Ge... | botocore.e... | +

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyIAMRole:instanceId=i-Off92bade5d684dfc

aws | Search [Alt+S] | United States (N. Virginia) | rsoaccount-new/6801da4369d20120be221457 @ rsosandboxnew43

EC2 > Instances > i-Off92bade5d684dfc > Modify IAM role

Modify IAM role [Info](#)

Attach an IAM role to your instance.

Instance ID
 i-Off92bade5d684dfc (TravelGoproject)

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_SNS_Role1   [Create new IAM role](#) 

[Cancel](#) [Update IAM role](#)

- Now connect the EC2 with the files

Connect to instance Info

Connect to your instance i-001861022fbcac290 (InstantLibraryApp) using any of these options

[EC2 Instance Connect](#)

[Session Manager](#)

[SSH client](#)

[EC2 serial console](#)



Port 22 (SSH) is open to all IPv4 addresses

Port 22 (SSH) is currently open to all IPv4 addresses, indicated by **0.0.0.0/0** in the inbound rule in [your security group](#). For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: 13.233.177.0/29. [Learn more](#).

Instance ID

i-001861022fbcac290 (InstantLibraryApp)

Connection Type

Connect using EC2 Instance Connect

Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.

Public IPv4 address

13.200.229.59

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.

ec2-user



i **Note:** In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

[Cancel](#)

[Connect](#)

```
A newer release of "Amazon Linux" is available.
Version 2023.6.20241010:
Run "/usr/bin/dnf check-release-update" for full release and version update info
      #          Amazon Linux 2023
      #\_###\_
      ## \###\
      ##  \#/   https://aws.amazon.com/linux/amazon-linux-2023
      ##  V~'-->
      ~~~ .-' / \
      ~~~ \_/_/ \
      ~~~ \_m_ \
Last login: Tue Oct 16 04:17:59 2024 from 13.233.177.3
[ec2-user@ip-172-31-3-5 ~]$
```

i-001861022fbcac290 (InstantLibraryApp)

PublicIPs: 13.201.74.42 PrivateIPs: 172.31.3.5

part
ternz

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/your-github-username/your-repository-name.git>' Note:

change your-github-username and your-repository-name with your credentials here: 'git clone https://github.com/AlekhyaPenubakula/InstantLibrary.git' • This will download your project to the 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
```

```
A newer release of "Amazon Linux" is available.
Version 2023.6.20241010:
Run "/usr/bin/dnf check-release-update" for full release and version update info
      #
      #\###          Amazon Linux 2023
~~ \###\
~~ \|##|
~~ \|/| https://aws.amazon.com/linux/amazon-linux-2023
~~ V~' '-->
~~ .-/
~~ / /
/m/.*

Last login: Tue Oct 15 04:17:59 2024 from 13.233.177.3
[ec2-user@ip-172-31-3-5 ~]$ git clone https://github.com/Alekhyapenubakula/InstantLibrary.git
fatal: destination path 'InstantLibrary' already exists and is not an empty directory.
[ec2-user@ip-172-31-3-5 ~]$ cd InstantLibrary
[ec2-user@ip-172-31-3-5 InstantLibrary]$ cd InstantLibrary
[ec2-user@ip-172-31-3-5 InstantLibrary]$ flask run --host=0.0.0.0 --port=80
 * Debug mode: off
Permission denied
[ec2-user@ip-172-31-3-5 InstantLibrary]$ ^C
[ec2-user@ip-172-31-3-5 InstantLibrary]$ ^C
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
 * Debug mode: off
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://127.0.0.1:80
 * Running on http://172.31.3.5:80
Press CTRL+C to quit
^C[ec2-user@ip-172-31-3-5 InstantLibrary]$
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
 * Debug mode: off
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://127.0.0.1:80
 * Running on http://172.31.3.5:80
Press CTRL+C to quit
183.82.125.56 - - [22/Oct/2024 07:42:00] "GET / HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /register HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /static/images/library3.jpg HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /favicon.ico HTTP/1.1" 404 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /static/images/library3.jpg HTTP/1.1" 304 -
183.82.125.56 - - [22/Oct/2024 07:42:21] "POST /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:24] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:27] "POST /login HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:28] "GET /home-page HTTP/1.1" 200 -

i-001861022fbcac290 (InstantLibraryApp)
Public IPs: 13.201.74.42 Private IPs: 172.31.3.5
```

Verify the Flask app is running: <http://your-ec2-public-ip>

- Run the Flask app on the EC2 instance

```
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
 * Debug mode: off
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://127.0.0.1:80
 * Running on http://172.31.3.5:80
Press CTRL+C to quit
183.82.125.56 - - [22/Oct/2024 07:42:00] "GET / HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /register HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /static/images/library3.jpg HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /favicon.ico HTTP/1.1" 404 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /static/images/library3.jpg HTTP/1.1" 304 -
183.82.125.56 - - [22/Oct/2024 07:42:21] "POST /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:24] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:27] "POST /login HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:28] "GET /home-page HTTP/1.1" 200 -
```

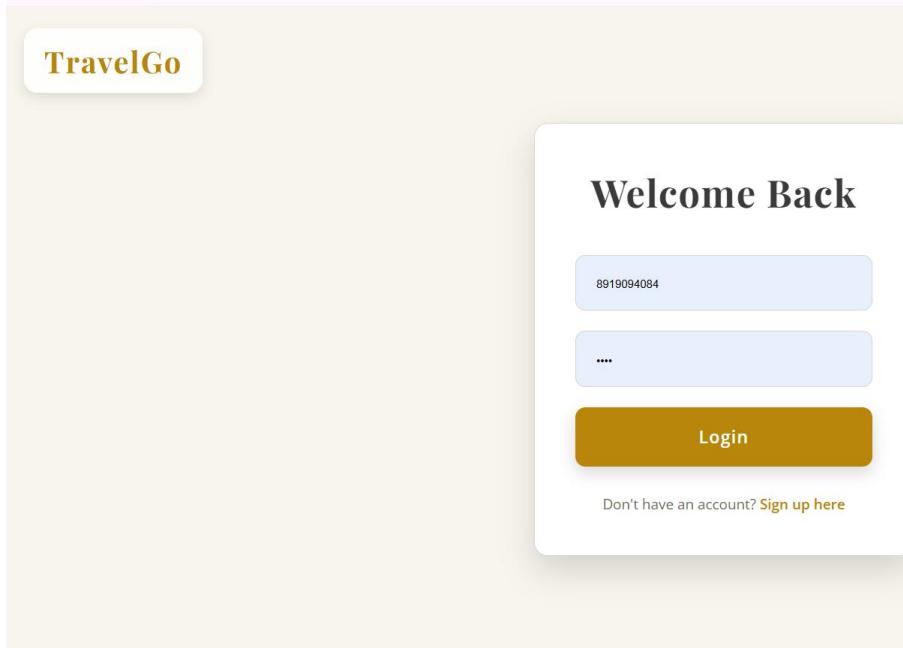
Access the website through:

PublicIPs: <http://54.221.143.23:5000>

Milestone 8: Testing and Deployment

- **Activity 8.1: Conduct functional testing to verify user registration, login, book requests, and notifications.**

Login Page:



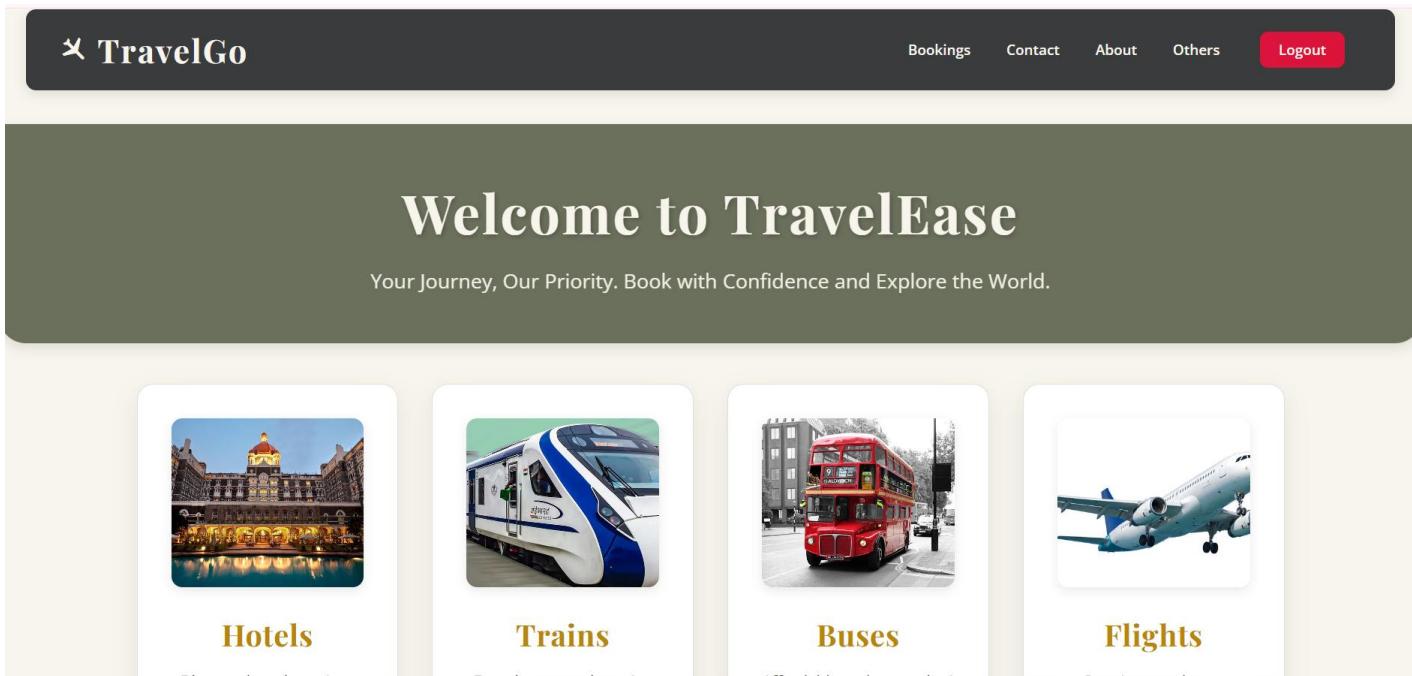
Register Page:

TravelGo

Create Account

 Phone Number 8919094084 Confirm PasswordSign UpAlready have an account? [Login here](#)

Home page:



The screenshot shows the TravelGo homepage. At the top, there is a dark header bar with the "TravelGo" logo on the left and navigation links for "Bookings", "Contact", "About", "Others", and "Logout" on the right. Below the header, a large banner features the text "Welcome to TravelEase" in a large serif font, followed by the tagline "Your Journey, Our Priority. Book with Confidence and Explore the World." in a smaller sans-serif font. The main content area contains four cards, each representing a travel service: "Hotels" (showing a building at night), "Trains" (showing a modern high-speed train), "Buses" (showing a red double-decker bus), and "Flights" (showing an airplane in flight). Each card has its respective name centered below it.

About Bookings page:

Your Latest Bookings

None Booking: (Booked: 2025-07-09 13:21) Cancel

None Booking: (Booked: 2025-07-09 08:08) Cancel

Buses Booking: To tirupati on 2025-07-26 (Booked: 2025-07-09 08:08) Cancel

None Booking: (Booked: 2025-07-09 08:04) Cancel

None Booking: (Booked: 2025-07-08 17:02) Cancel

Contact Page:

Connect With Us

Our dedicated team is here to assist you with any travel inquiries or support you may need.
Reach out to us anytime!

Email: support@travelease.com
Phone: [+1 \(800\) 555-TRAVEL](tel:+1800555TRAVEL)
Address: 456 Wanderlust Way, Global City, Earth

About page:

About TravelEase

At TravelEase, we believe every journey should be effortless and memorable. We are committed to providing a seamless booking experience for all your travel needs, from luxurious hotels to efficient transportation.

Our mission is to empower your adventures, ensuring comfort, convenience, and unparalleled service every step of the way. Travel with Ease, Explore with Joy.

Hotels Booking :

TravelGo

Book a Hotel

Select a Hotel

City

Check-in: mm/dd/yyyy Check-out: mm/dd/yyyy

Adults: 1 Children: 0

Select Room Type

Special Requests (e.g., non-smoking, extra bed)

Estimated Price: \$0.00

Book Now

[Back to Home](#)

Trains Booking:

TravelGo

Search for Trains

Search Trains

[Back to Home](#)

TravelGo

Available Trains

Route: srikakulam to tirupati
Date: 2025-07-12

Express Rail (TRNoo1)

Departure: 06:00 AM
Arrival: 10:00 AM
Price: \$800
Classes: Sleeper, AC3

Select Train

City Link (TRNoo2)

Departure: 09:30 AM
Arrival: 01:30 PM
Price: \$750
Classes: Sleeper, AC2

Select Train

Night Rider (TRNoo3)

Departure: 08:00 PM
Arrival: 06:00 AM
Price: \$1200

Select Train

TravelGo

Book Train: TRNoo1

srikakulam

tirupati

07/12/2025

Adults: 1 Children: 0

Select Class

Assigned Seat(s): A5-B34

Book Now

[Back to Train Selection](#)

Buses Booking:

TravelGo

Search for Buses

From (City)

To (City)

mm/dd/yyyy

Search Buses

[Back to Home](#)

Comfort Bus

Departure: 01:00 PM

Price: \$550

Seats Available: 20

Select Bus

TravelGo

Select Seats for Bus: bus_001

1

Select Your Seats

A1	A2	A3	A4
B1	B2	B3	B4
C1	C2	C3	C4
D1	D2	D3	D4
E1	E2	E3	E4
F1	F2	F3	F4
G1	G2	G3	G4
H1	H2	H3	H4
I1	I2	I3	I4
I1	I2	I3	I4

Flights Booking:

TravelGo

Search for Flights

srikakulam

tirupati

Departure:

07/18/2025

Return (optional):

mm/dd/yyyy

Search Flights

Back to Home

TravelGo

Available Flights

Route: srikakulam to tirupati
 Departure Date: 2025-07-18

AirConnect (FLT101)

Departure: 07:00 AM
 Arrival: 09:00 AM
 Price: \$3500
 Classes: Economy, Business

[Select Flight](#)
SkyHigh (FLT102)

Departure: 11:00 AM
 Arrival: 01:00 PM
 Price: \$4200
 Classes: Economy, Premium Economy

[Select Flight](#)
Global Wings (FLT103)

Departure: 03:00 PM
 Arrival: 05:00 PM
 Price: \$5000

[Select Flight](#)
TravelGo

Complete Your Booking

Flight: FLT101
 Route: srikakulam to tirupati
 Departure: 2025-07-18

Adults: Children:

Premium Economy

Assigned Seat(s): F7D

[Confirm Booking](#)
[Back to Flight Selection](#)

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.