

TravelGo – Your Cloud-Powered Travel Companion

Project Description:

TravelGo is a comprehensive, full-stack travel booking platform built to revolutionize the way users plan and manage their travel. Designed with a modern, intuitive user interface and powered by robust cloud infrastructure, TravelGo enables seamless reservations for buses, trains, flights, and hotels—all from a single, unified dashboard.

Developed using Flask as the backend framework, TravelGo is hosted on Amazon EC2, ensuring scalability and high availability. It leverages Amazon DynamoDB for high-performance, NoSQL data storage, making it ideal for handling user profiles, travel listings, and booking records.

Scenario 1: Multi-Mode, End-to-End Booking Experience

Imagine a traveler planning a business trip from **Hyderabad to Bangalore**. With TravelGo:

They log into the platform and enter travel details. The system presents a curated list of **bus**, **train**, and **flight** options with dynamic pricing and availability. After selecting a train and a hotel near their destination, the user confirms both bookings in a single session.

Scenario 2: Automated, Real-Time Booking Confirmations

A student booking a hotel in Chennai for exam preparation completes the reservation within seconds. Immediately:

TravelGo stores the booking information securely in DynamoDB. AWS SNS sends a customized confirmation email with check-in details and cancellation policy.

Simultaneously, hotel staff receive a booking alert, ensuring rooms are reserved accordingly—minimizing booking errors and improving service reliability. AWS SNS sends a customized confirmation email with check-in details and cancellation policy.

Simultaneously, hotel staff receive a booking alert, ensuring rooms are reserved accordingly—minimizing booking errors and improving service reliability.

Scenario 3: Centralized Dashboard with Travel Summary

A regular traveler logs in to check upcoming plans. In their dashboard:

They see all their bookings—like a flight to Delhi and a hotel in Connaught Place—clearly displayed. Each entry includes booking ID, price, travel date, status (confirmed/cancelled), and a cancel or reschedule button. Each entry include booking ID, price, travel date, status (confirmed/cancelled), and a cancel or reschedule button. booking ID, price, travel date, status (confirmed/cancelled), and a cancel or reschedule button. Flask retrieves the data from DynamoDB and renders it using Jinja templates, ensuring fast, dynamic loading and real-time data updates.

Project WorkFlow:

1. AWS Account Setup and Login

Activity 1.1: Set up an AWS account if not already done.

Activity 1.2: Log in to the AWS Management Console

2. DynamoDB Database Creation and Setup

Activity 2.1: Create a DynamoDB Table.

Activity 2.2: Configure Attributes for User Data and Book Requests.

3. SNS Notification Setup Activity

Activity 3.1: Create SNS topics for book request notifications.

Activity 3.2: Subscribe users and library staff to SNS email notifications.

4. Backend Development and Application Setup

Activity 4.1: Develop the Backend Using Flask.

Activity 4.2: Integrate AWS Services Using boto3.

5. IAM Role Setup

Activity 5.1: Create IAM Role Activity

Activity 5.2: Attach Policies

6. EC2 Instance Setup Activity

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

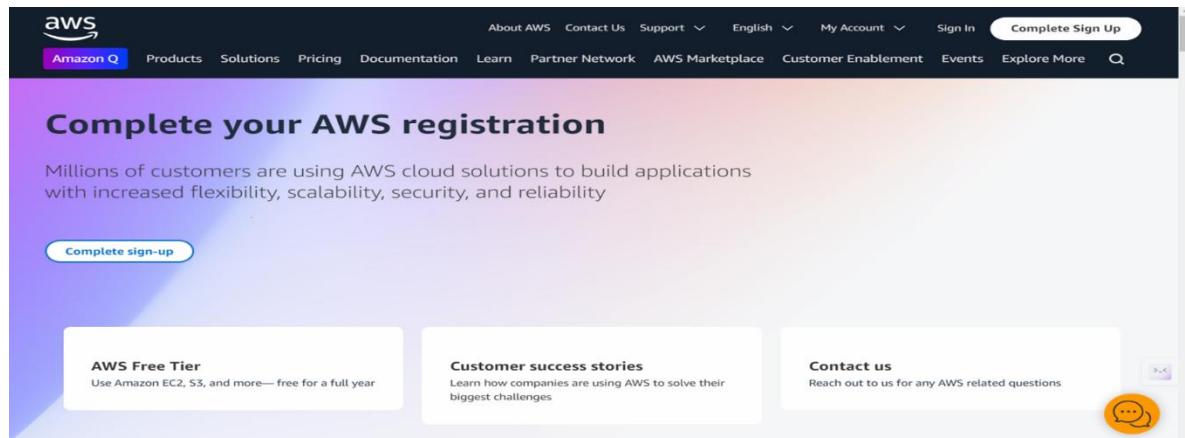
8. Testing and Deployment

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

Milestone 1: AWS Account Setup and Login

- **Activity 1.1: Set up an AWS account if not already done.**

- Sign up for an AWS account and configure billing settings.



- **Activity 1.2: 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 'IAM user sign in' page, featuring fields for 'Account ID or alias', 'Remember this account' (checkbox), 'IAM username', 'Password', 'Show Password' (checkbox), 'Having trouble?' (link), a large orange 'Sign in' button, and a smaller 'Sign in using root user email' link at the bottom. On the right is the 'Amazon Lightsail' landing page, which has a dark background with a bright orange swoosh graphic. It features the 'Amazon Lightsail' logo, the text 'Lightsail is the easiest way to get started on AWS', a 'Learn more »' button, and a cartoon character giving a thumbs up.

Milestone 2: DynamoDB Database Creation and Setup

● Activity 2.1: Navigate to the DynamoDB

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

The screenshot shows the AWS Console homepage with several service cards. The 'Recently visited' card includes links to IAM, EC2, DynamoDB, Simple Notification Service, Billing and Cost Management, AWS Marketplace, Route 53, and S3. The 'Applications' card shows 0 applications in the Asia Pacific (Mumbai) region. The 'Welcome to AWS' card has a 'Getting started with AWS' link. The 'AWS Health' card shows 0 open issues. The 'Cost and usage' card displays current month costs and total cost.

The main content area is focused on the **Amazon DynamoDB** service. It features a large heading: "A fast and flexible NoSQL database service for any scale". Below this, a description states: "DynamoDB is a fully managed, key-value, and document database that delivers single-digit-millisecond performance at any scale." A prominent orange "Create table" button is located in a "Get started" callout box. To the right, a "Pricing" section explains that DynamoDB charges for reading, writing, and storing data, with options for on-demand and provisioned capacity modes. A "Learn more about pricing" link is provided. At the bottom, there's a "Documentation" link and copyright information: "© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

On the left sidebar, under the **DynamoDB** heading, are links for Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. Under the **DAX** heading, are links for Clusters, Subnet groups, Parameter groups, and Events. At the bottom of the sidebar are CloudShell, Feedback, and a search bar.

The bottom-most window is titled "Tables" and shows a message: "You have no tables in this account in this AWS Region." It includes a "Create table" button and a header with columns: Name, Status, Partition key, Sort key, Indexes, Deletion protection, Read capacity mode, Write capacity mode, and Total size. There are also "Actions", "Delete", and "Create table" buttons at the top of this window.

- **Activity 2.2: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.

The screenshot shows the 'Create table' wizard in the AWS DynamoDB console. It consists of two main sections: 'Table details' and 'Table settings'.

Table details

- Table name:** bookings
- Partition key:** user_email (String)
- Sort key - optional:** booking_date (String)

Table settings

- Default settings:** On-demand capacity mode, 0 maximum read/write units, no local/global secondary indexes, AWS owned encryption, off deletion protection, and no resource-based policy.
- Customize settings:** Tags section (No tags associated).

At the bottom right of the wizard, there are 'Cancel' and 'Create table' buttons.

- Follow the same steps to create a requests table with Email as the primary key for book requests data.

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.

travelgo_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 modify

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

aws Search [Alt+S] Asia Pacific (Mumbai) midhunkotagiri

DynamoDB > Tables > Create table

Capacity mode	On-demand	Yes
Maximum read capacity units	-	Yes
Maximum write capacity units	-	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	AWS owned key	Yes
Deletion protection	Off	Yes
Resource-based policy	Not active	Yes

Tags

Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

Add new tag

You can add 50 more tags.

Create table

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 AWS Services Catalog interface. On the left, there's a sidebar with navigation links for DynamoDB, DAX, and other AWS services. The main area displays two services: "Simple Notification Service" and "Route 53 Resolver". Each service card includes a brief description and a "Show more" link.

The screenshot shows the Amazon SNS service page. It features a "New Feature" announcement about in-place message archiving and replay for FIFO topics. Below this, there's a section titled "Application Integration" with a large heading "Amazon Simple Notification Service" and a sub-section "Pub/sub messaging for microservices and serverless applications." To the right, there's a "Create topic" form with a "Topic name" field containing "MyTopic" and a "Next step" button. At the bottom, there's a "Pricing" section.

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

The screenshot shows the Amazon SNS Topics page. It lists three topics: "Topic 1", "Topic 2", and "Topic 3". There are buttons for "Edit", "Delete", "Publish message", and "Create topic". A search bar at the top allows filtering by Name or ARN.

- o Choose **Standard type** for general notification use cases and Click on Create Topic.

Amazon SNS > Topics > Create topic

New Feature
Amazon SNS now supports High Throughput FIFO topics. [Learn more](#)

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:
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.

Maximum 100 characters.

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

- **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.

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

AWS Notification - Subscription Confirmation Spam x Print Flag

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

Tue 1 Jul, 11:15 (2 days ago) ☆ 😊 ↶ ⋮

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

Report as not spam ⓘ

You have chosen to subscribe to the topic:
`arn:aws:sns:us-east-1:600627364806: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](#)

↶ Reply ↷ Forward 😊

 Simple Notification Service

Subscription confirmed!

You have successfully subscribed.

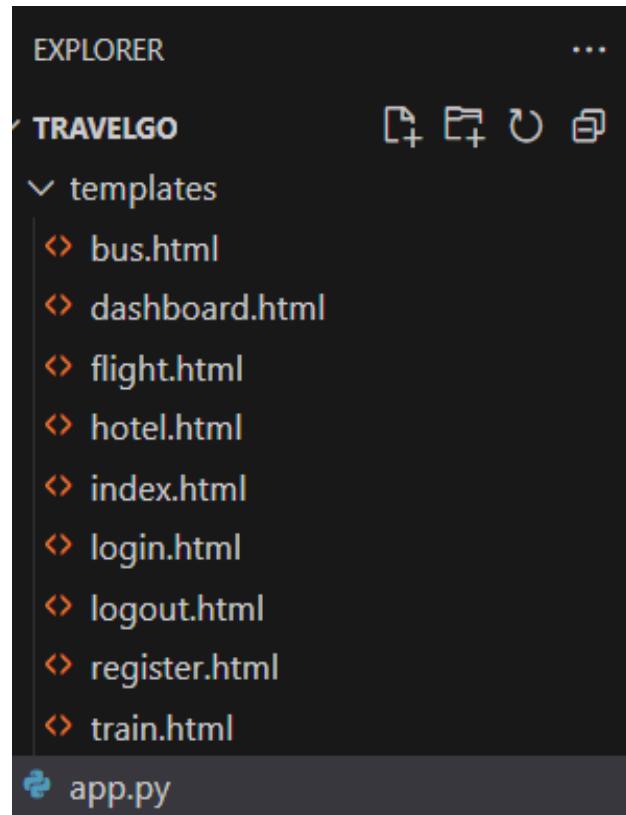
Your subscription's id is:
`arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4`

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

Milestone 4:Backend Development and Application Setup

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

- File Explorer Structure



Description: set up the INSTANT LIBRARY 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., computer_science.html, data_science.html), and utility pages (e.g., request-form.html, statistics.html).

Description of the code :

- **Flask App Initialization**

```
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
...
```

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:**

```
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 ap-south-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:ap-south-1:353250843450:TravelGoBookingTopic' # Replace with actual ARN

# Function to send SNS notifications
# This function is duplicated in the original code, removing the duplicate.
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}")
        # Optionally, flash an error message to the user or log it more robustly.
```

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:

```
# Routes
@app.route('/')
def home():
    return render_template('index.html', logged_in='email' in session)
```

Description: Define the home route (/) of the Flask application. The home() function renders the index.html template. It also checks if the user is logged in by verifying if 'email' exists in the session. If so, it passes logged_in=True to the template, allowing the frontend to dynamically adjust the UI (e.g., show or hide login/register buttons).

● Register Route:

```
@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')
```

Description: Handles the /register route for user signup using GET and POST methods. Checks if the user already exists in the travelgo_users table and prevents duplicates. If new, hashes the password, stores the user in DynamoDB, and redirects to the login page.

```
@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'], password):
            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')
```

Description: Handles the /login route for user authentication via GET and POST methods. On POST, it retrieves the user from travelgo_users by email and verifies the password using check_password_hash(). If valid, it stores user details in the session and updates the login count in DynamoDB. On failure, it flashes an error message and re-renders the login page.

- **Dashboard Route:**

```
@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)
```

Description: Defines the /dashboard route to display the logged-in user's booking history. Checks session for a valid login, then queries the bookings table using the user's email to fetch bookings in reverse chronological order. Converts any Decimal values from DynamoDB to float for proper display in the HTML template. Renders dashboard.html, passing the user's email and their bookings for display.

- **Booking Route:**

```
# Booking Routes
@app.route('/bus')
def bus_booking():
    if 'email' not in session:
        flash('Please login to book tickets.', 'error')
        return redirect(url_for('login'))
    return render_template('bus.html')

@app.route('/train')
def train_booking():
    if 'email' not in session:
        flash('Please login to book tickets.', 'error')
        return redirect(url_for('login'))
    return render_template('train.html')

@app.route('/flight')
def flight_booking():
    if 'email' not in session:
        flash('Please login to book tickets.', 'error')
        return redirect(url_for('login'))
    return render_template('flight.html')

@app.route('/hotel')
def hotel_booking():
    if 'email' not in session:
        flash('Please login to book tickets.', 'error')
        return redirect(url_for('login'))
    return render_template('hotel.html')
```

Description: Defines individual booking routes for bus, train, flight, and hotel pages. Each route first checks if the user is logged in by verifying the session; if not, it redirects to the login page with an error message. If the user is authenticated, the corresponding booking template (bus.html, train.html, etc.) is rendered.

● Deployment Code:

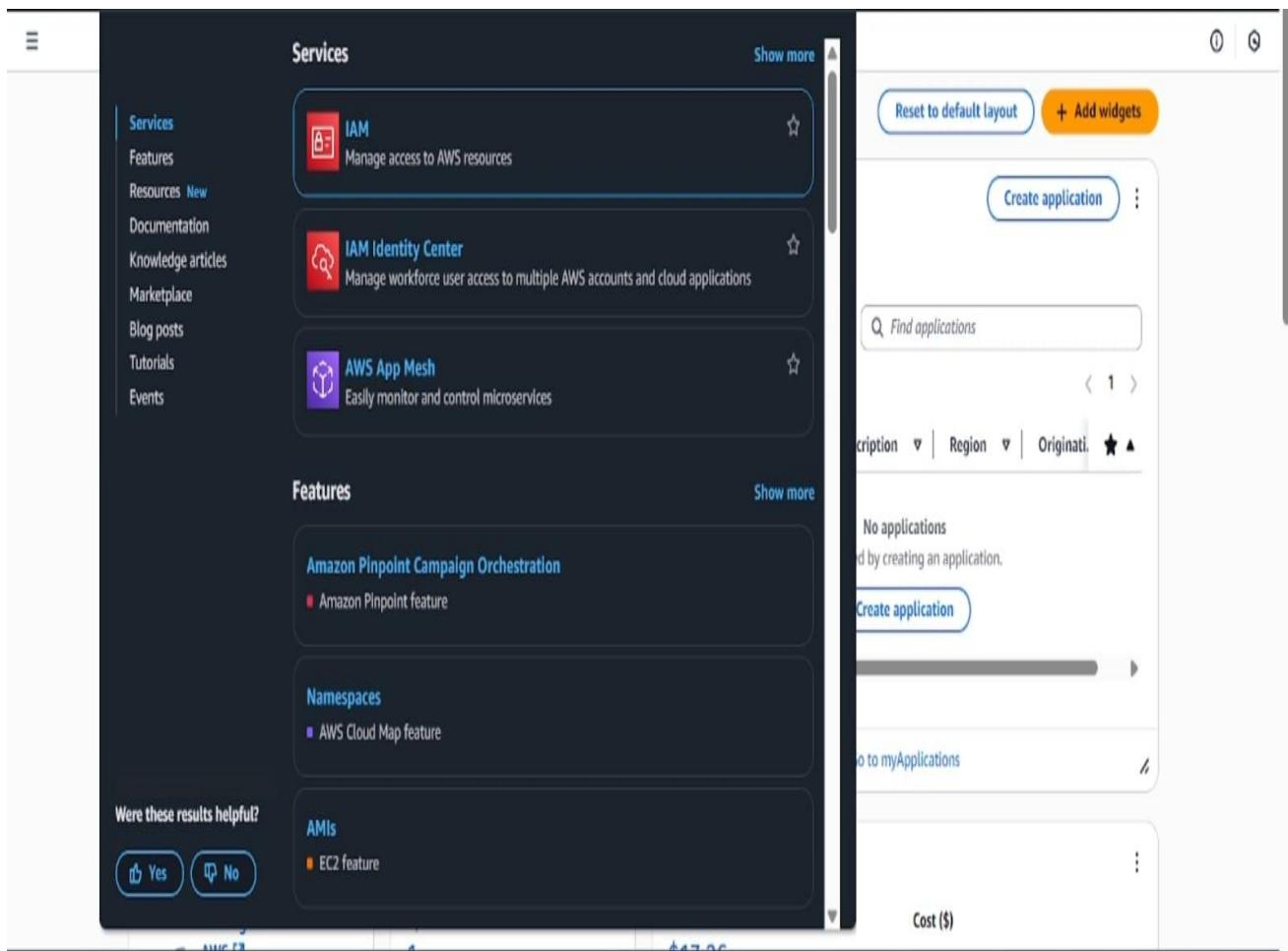
```
if __name__ == '__main__':
    # IMPORTANT: In a production environment, disable debug mode and specify a production-ready host.
    app.run(debug=True, host='0.0.0.0', port=5000)
```

Description: Starts the Flask application when the script is run directly. Runs the app in debug mode for development, allowing live code reloading and error display. Binds the server to all IP addresses (0.0.0.0) on port 5000, making it accessible on the local network or EC2 instance.

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 IAM Dashboard. On the left, a sidebar lists navigation options: Dashboard, Access management (User groups, Users, Roles, Policies), Identity providers, Account settings, Root access management (New), and Access reports (Access Analyzer, Resource analysis New, Unused access, Analyzer settings). A blue banner at the top right says "New access analyzers available" with a "Create new analyzer" button. The main area is titled "IAM Dashboard" with a "Security recommendations" section containing two items: "Root user has MFA" (Having multi-factor authentication (MFA) for the root user improves security for this account) and "Root user has no active access keys" (Using access keys attached to an IAM user instead of the root user improves security). To the right is a "AWS Account" summary with fields for Account ID (353250843450), Account Alias (Create), and Sign-in URL (https://353250843450.signin.aws.amazon.com/console).

The screenshot shows the "Create role" step of a wizard. The top bar shows the path: IAM > Roles > Create role. A search bar says "Filter service or use case". Below it is a "Commonly used services" dropdown menu with "EC2" selected. A tooltip for "EC2" says "with SAML 2.0 from perform actions in". Other services listed include Lambda, Amazon Aurora DSQL, Amazon EMR Serverless, Amazon OpenSearch Service, Amazon Q Business, Amazon Grafana, Amplify, and API Gateway. At the bottom of the dropdown is a placeholder "Choose a service or use case". Buttons for "Cancel" and "Next" are at the bottom right.

● 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.

● **AmazonEC2FullAccess:** Provides full permissions to manage all EC2 resources, including instances, volumes, and networking.

The screenshot shows the 'Add permissions' step of creating a new IAM role. The left sidebar indicates Step 1: Select trusted entity, Step 2: Add permissions (which is selected), and Step 3: Name, review, and create. The main area is titled 'Add permissions' and shows a search bar for 'Permissions policies (2/1057)'. A filter bar at the top right says 'Filter by Type' with 'Q: Dynamo' and 'All types' selected. Below is a table of policies:

Policy name	Type	Description
AmazonDynamoDBFullAccess	AWS managed	Provides full access
AmazonDynamoDBFullAccess_v2	AWS managed	Provides full access
AmazonDynamoDBFullAccesswithDataPipeline	AWS managed	This policy is on a
AmazonDynamoDBReadOnlyAccess	AWS managed	Provides read only
AWSLambdaDynamoDBExecutionRole	AWS managed	Provides list and

The screenshot shows the 'Add permissions' step of creating a new IAM role. The left sidebar indicates Step 1: Select trusted entity, Step 2: Add permissions (which is selected), and Step 3: Name, review, and create. The main area is titled 'Add permissions' and shows a search bar for 'Permissions policies (3/1057)'. A filter bar at the top right says 'Filter by Type' with 'Q: SNS' and 'All types' selected. Below is a table of policies:

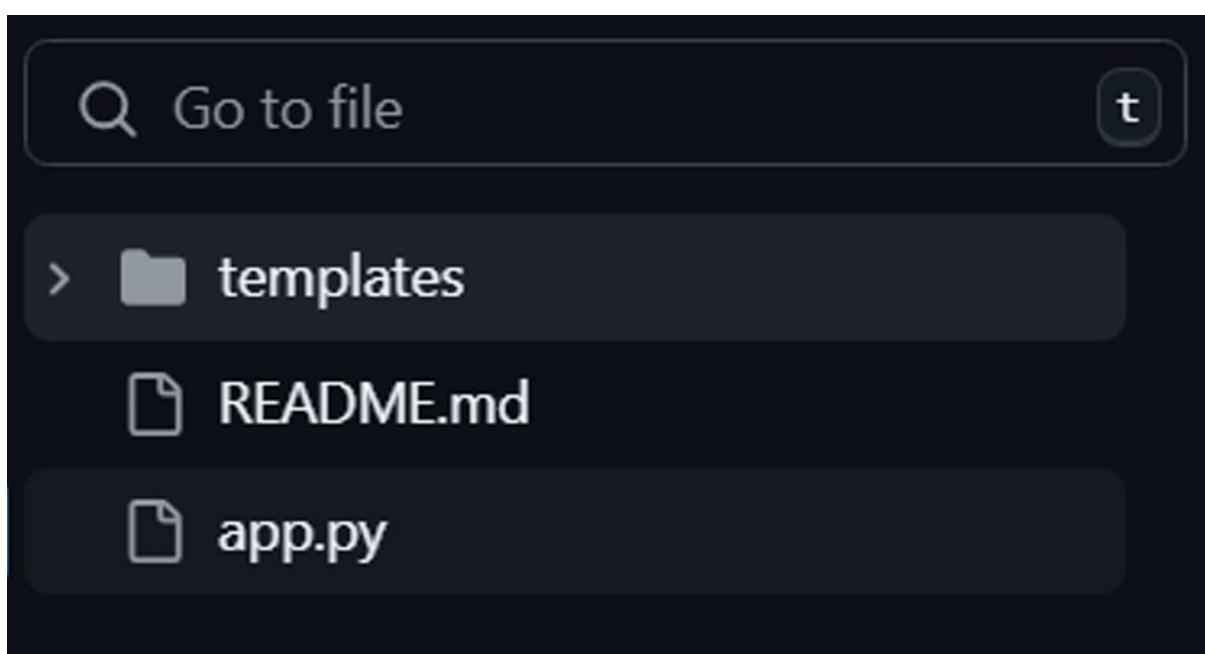
Policy name	Type	Description
AmazonSNSFullAccess	AWS managed	Provides full acce
AmazonSNSReadOnlyAccess	AWS managed	Provides read onl
AmazonSNSRole	AWS managed	Default policy fo
AWSElasticBeanstalkRoleSNS	AWS managed	(Elastic Beanstalk
AWSIoTDeviceDefenderPublishFindingsToSNSMitigatio...	AWS managed	Provides message

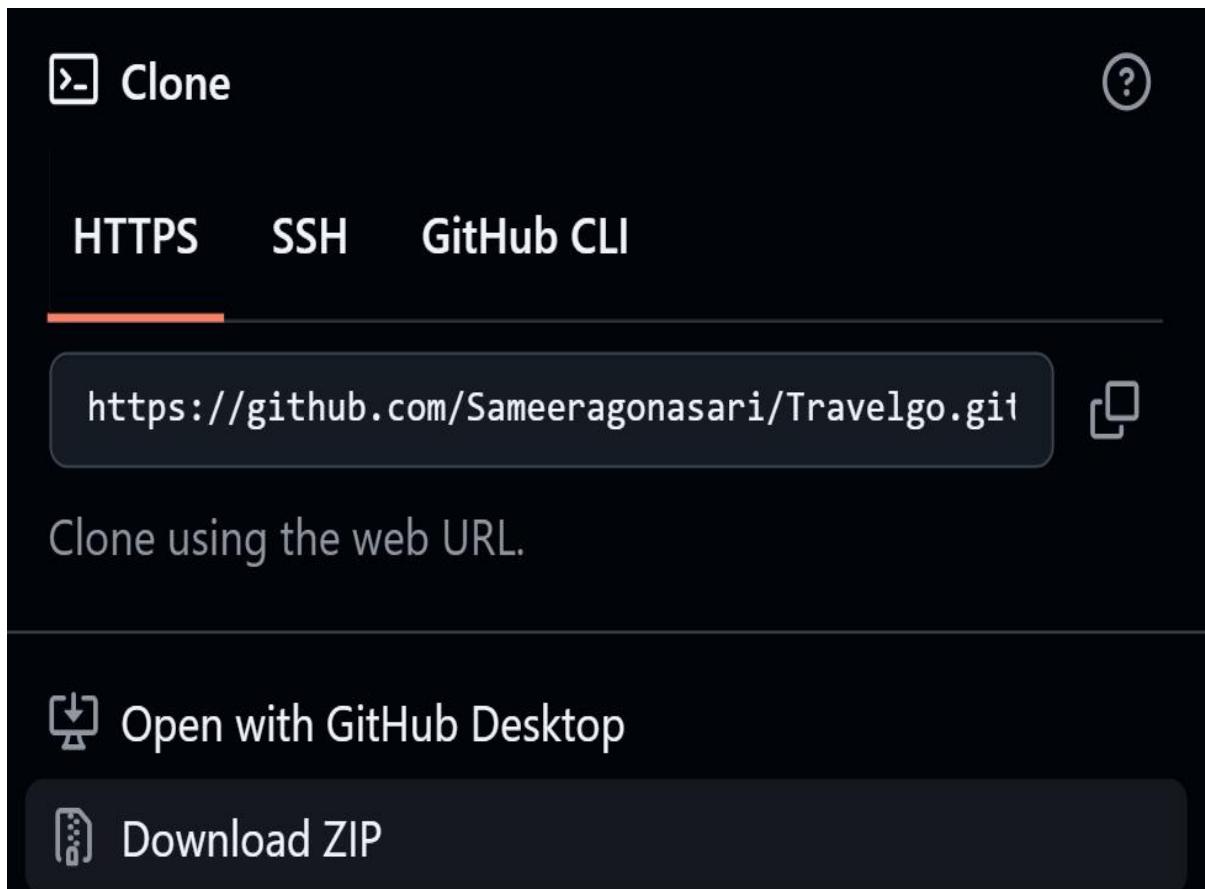
The screenshot shows the AWS IAM Roles page under the 'Create role' section. A search bar at the top is empty. Below it is a table with the following columns: 'Policy name' (with a dropdown arrow icon), 'Type' (with a downward arrow icon), and 'Description'. The table lists several AWS managed policies:

Policy name	Type	Description
AmazonEC2ContainerRegistryFullAccess	AWS managed	Provides adminis
AmazonEC2ContainerRegistryPowerUser	AWS managed	Provides full acc
AmazonEC2ContainerRegistryPullOnly	AWS managed	Provides access to
AmazonEC2ContainerRegistryReadOnly	AWS managed	Provides read-on
AmazonEC2ContainerServiceAutoscaleRole	AWS managed	Policy to enable t
AmazonEC2ContainerServiceEventsRole	AWS managed	Policy to enable t
AmazonEC2ContainerServiceforEC2Role	AWS managed	Default policy for
AmazonEC2ContainerServiceRole	AWS managed	Default policy for
AmazonEC2FullAccess	AWS managed	Provides full acce
AmazonEC2ReadOnlyAccess	AWS managed	Provides read onl

Milestone 6: EC2 Instance Setup

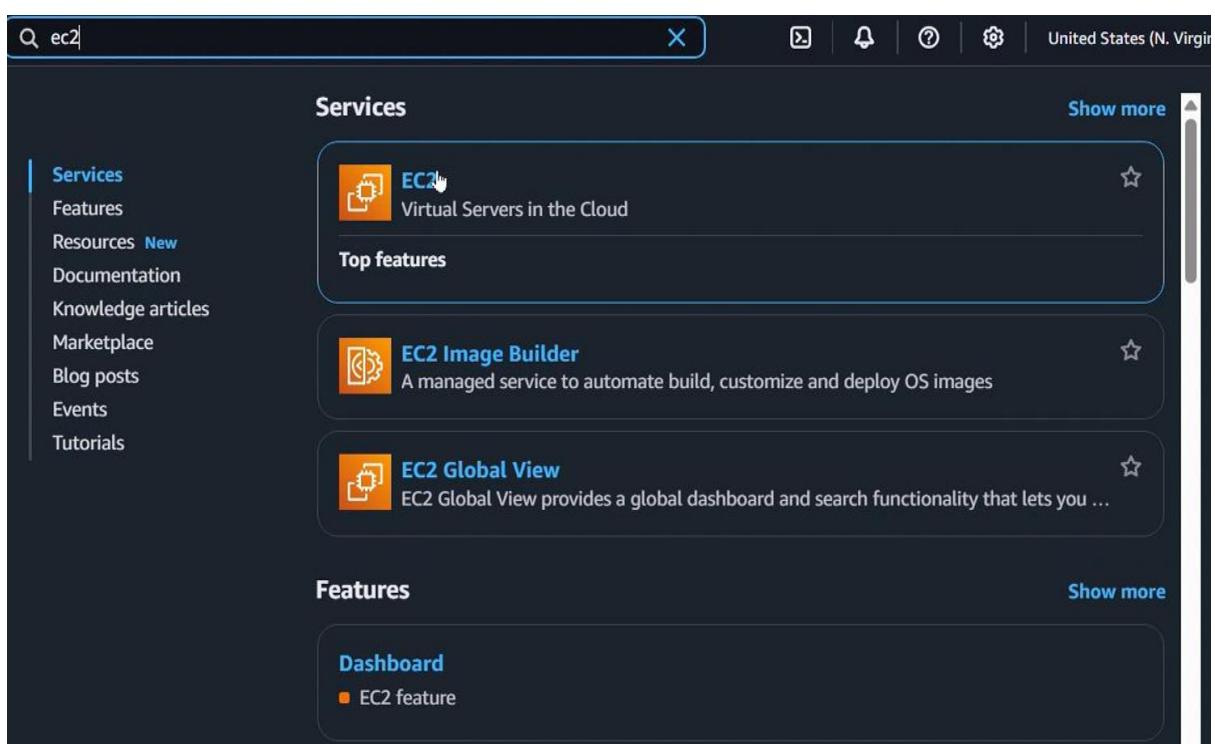
- **Note:** Load your Flask app and Html files into GitHub repository.





- **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 image shows the AWS EC2 service page. The search bar at the top has 'ec2' typed into it. The main area is titled 'Services' and features a 'Top features' section with three items: 'EC2' (Virtual Servers in the Cloud), 'EC2 Image Builder' (A managed service to automate build, customize and deploy OS images), and 'EC2 Global View' (EC2 Global View provides a global dashboard and search functionality that lets you ...). Below this is another section titled 'Features' with a single item: 'Dashboard' (EC2 feature). On the left side, there's a sidebar with links: Services, Features, Resources (New), Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials.

- Click on **Launch instance** to launch EC2 instance.

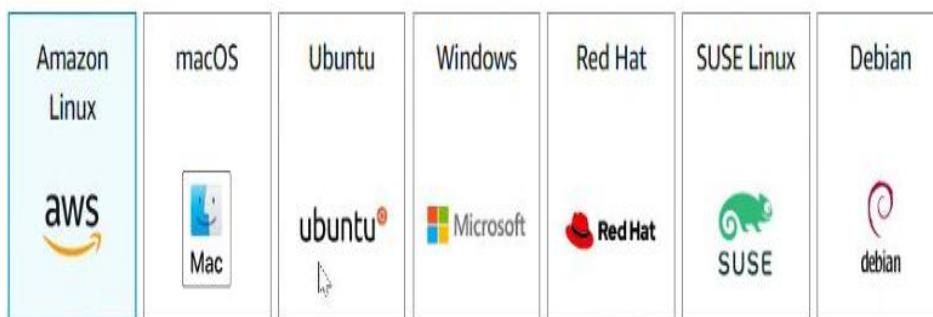
The screenshot shows the AWS EC2 homepage. On the left, there's a sidebar with options like Dashboard, Instances, and Launch instance. The main area features a large heading "Amazon Elastic Compute Cloud (EC2)" and sub-headings "Create, manage, and monitor virtual servers in the cloud." Below this is a paragraph about the service's breadth and depth, followed by a "Launch a virtual server" call-to-action button, which is highlighted with a yellow box. There's also a "View dashboard" button and a "Get started" section with a "Take our walkthroughs to help you" link.

The screenshot shows the "Launch an instance" wizard. Step 1 is titled "Name and tags". It has a "Name" field containing "Travelgoapplication1" and a "Software Image (AMI)" dropdown set to "Amazon Linux 2023 AMI 2023.7.2...". Step 2 is titled "Application and OS Images (Amazon Machine Image)". A summary section on the right shows 1 instance selected, the software image chosen, and the instance type set to "t2.micro".

- Choose Amazon Linux 2 or Ubuntu as the AMI and t2.micro as the instance type (free-tier eligible).

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

Quick Start



Browse 

Including
AWS, Mark
the Con

Amazon Machine Image (AMI)

Amazon Linux 2023 kernel-6.1 AMI

Free tier 

ami-05ffe3c48a9991133 (64-bit (x86), uefi-preferred) / ami-022bbd2ccaf21691f (64-bit (Arm), uefi)

Virtualization: hvm ENA enabled: true Root device type: ebs

- Create and download the key pair for Server access.

Create key pair

X

Key pair name

Key pairs allow you to connect to your instance securely.

travel-go-application-1

I

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

Key pair type



RSA

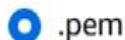
RSA encrypted private and public key pair



ED25519

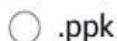
ED25519 encrypted private and public key pair

Private key file format



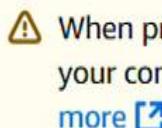
.pem

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](#)

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

Edit inbound rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-04f74a082f34acd4e	Custom TCP	TCP	5000	Custom	0.0.0.0/0
sgr-0e91ec1a5eaa31f68	HTTPS	TCP	443	Custom	0.0.0.0/0
sgr-0a2fdcbe6ce079d31	SSH	TCP	22	Custom	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.

Add rule Cancel Preview changes Save rules

- 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.

Instances (1/1) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
TravelGoproject	i-0b6de2ae9071ed0a7	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-18-206

i-0b6de2ae9071ed0a7 (TravelGoproject)

Security details

IAM Role	Owner ID	Launch time
-	686255958327	Tue Jul 01 2025 11:17:47 GMT+0530 (India Standard Time)

Security groups

sg-074f829b17da4071a (launch-wizard-1)
--

sg-074f829b17da4071a - launch-wizard-1

Details

Security group name	Security group ID	Description	VPC ID
launch-wizard-1	sg-074f829b17da4071a	launch-wizard-1 created 2025-07-01T05:45:44.696Z	vpc-0ed9be657d2390aa2
Owner	Inbound rules count	Outbound rules count	
686255958327	3 Permission entries	1 Permission entry	

Inbound rules (3)

Name	Security group rule ID	IP version	Type	Protocol	Port range
-	sgr-04f74a082f34acd4e	IPv4	Custom TCP	TCP	5000
-	sgr-0e91ec1a5eaa31f68	IPv4	HTTPS	TCP	443
-	sgr-0a2fdcb6ce079d31	IPv4	SSH	TCP	22

Modify IAM role Info

Attach an IAM role to your instance.

Instance ID

i-04e6efc7081801414 (travelgoapplication)

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.

studentuser



Create new IAM role

[Cancel](#)

[Update IAM role](#)

● Now connect the EC2 with the files

The screenshot shows the AWS Management Console with the AWS logo and search bar at the top. The navigation bar includes links for EC2, Instances, and the specific instance ID i-0b6de2ae9071ed0a7. Below the navigation is a breadcrumb trail: EC2 > Instances > i-0b6de2ae9071ed0a7 > Connect to instance. The main content area displays the "Connect" section, which includes tabs for EC2 Instance Connect, Session Manager, SSH client, and EC2 serial console. A warning message states: "⚠️ Unable to verify public subnet" and explains that the user lacks permission to perform the ec2:DescribeRouteTables action. It also notes that the instance must be in a public subnet to use EC2 Instance Connect. At the bottom, there are fields for Instance ID (i-0b6de2ae9071ed0a7) and connection methods: Public IPv4 address (selected) and IPv6 address.

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::686255958327: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-0661d905cf55c8217](#) 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-0b6de2ae9071ed0a7 (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

18.206.251.133

IPv6 address

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 install git -y

sudo yum install python3 -y

sudo yum install python3-pip -y

Pip install flask

Pip install boto3

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

```

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

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

S C:\Users\sameera.gonasari> ssh -i "C:\Users\sameera.gonasari\Downloads\travel-go-appl.pem" ec2-user@ec2-52-202-39-229.compute-1.amazonaws.com
The authenticity of host 'ec2-52-202-39-229.compute-1.amazonaws.com (64:ff9b::34ca:27e5)' can't be established.
ED25519 key fingerprint is SHA256:7B4eQMGlhXQnx9gJloaiMpM70AAb2ih9pRll09tJNzA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-52-202-39-229.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

      _#
     /_#####
    /_#####
   \###|
   \#/  ___ https://aws.amazon.com/linux/amazon-linux-2023
   \~'`'-->
      /
     /_
    /_/
   /m/' 

ec2-user@ip-172-31-81-54 ~]$ sudo yum git -y
No such command: git. Please use /usr/bin/yum --help
It could be a YUM plugin command, try: "yum install 'dnf-command(git)'"
ec2-user@ip-172-31-81-54 ~]$ sudo yum install git -y
Amazon Linux 2023 Kernel Livepatch repository                                171 kB/s | 17 kB   00:00
Dependencies resolved.
=====
Package          Architecture Version       Repository      Size
=====
=====

```

Verify the Flask app is running:

<http://your-ec2-public-ip> Run the Flask app on the EC2 instance

```

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://192.168.227.139:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 347-234-071

```

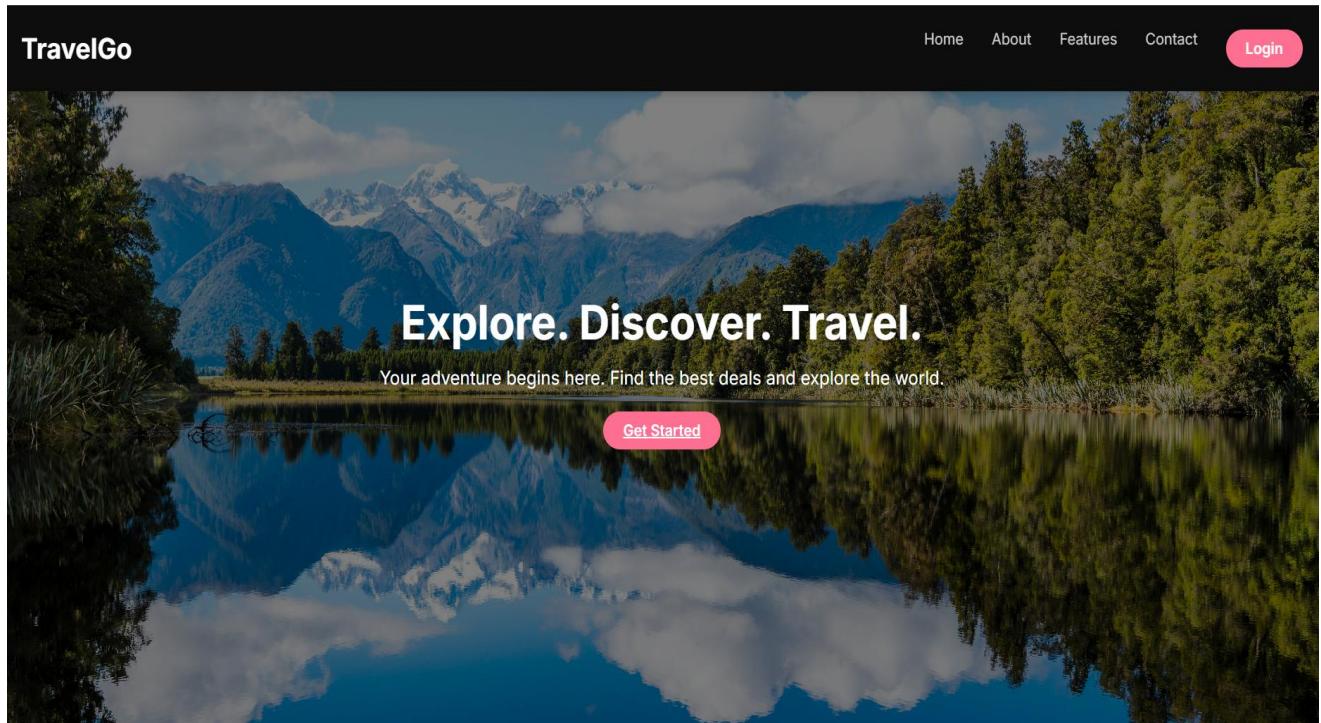
Access the website through:

Public IPs: <http://52.202.39.229:5000/>

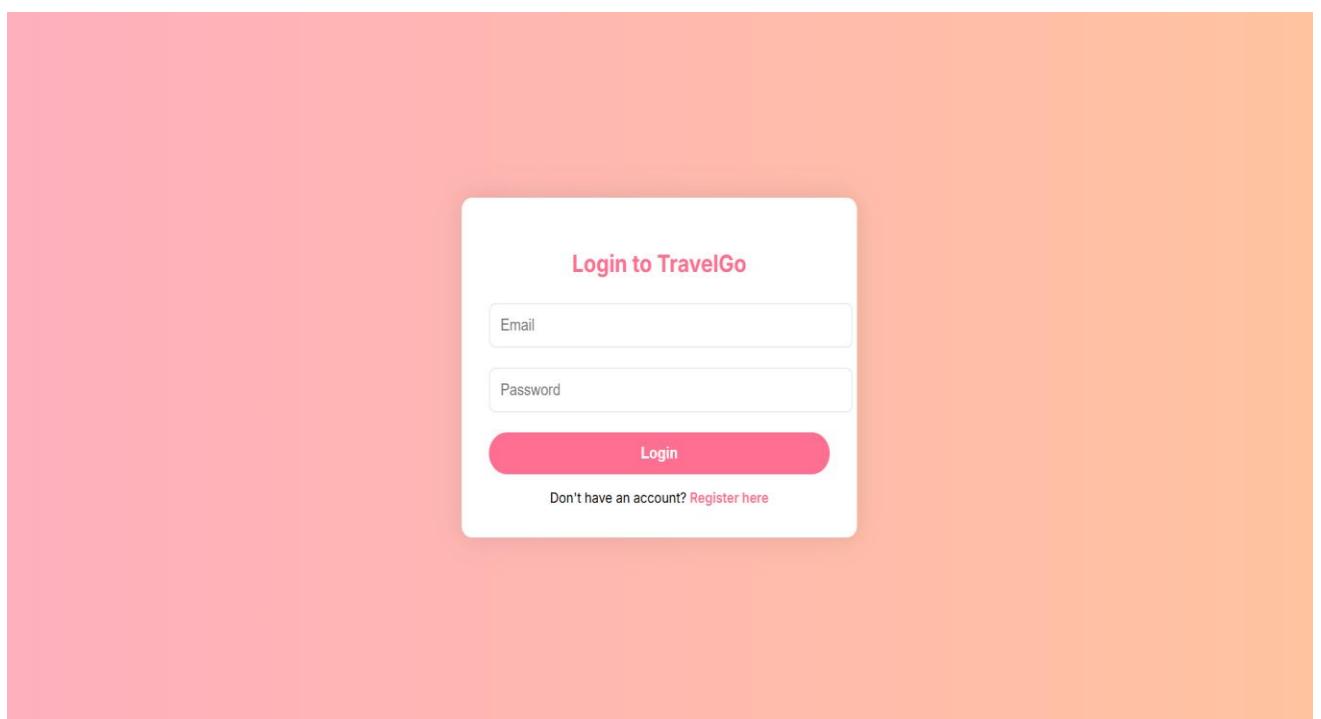
Milestone 8: Testing and Deployment

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

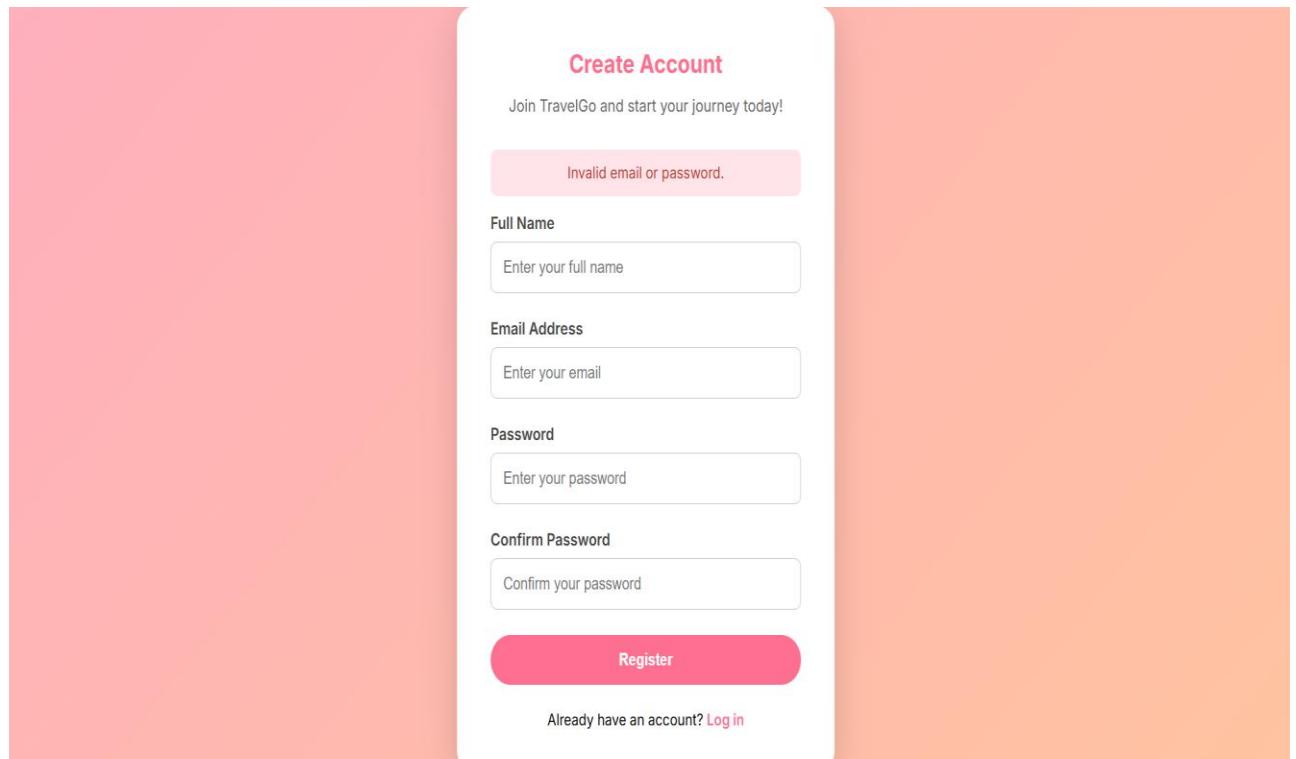
Index page:



Login page:



Register page:



The image shows the 'Create Account' registration form for TravelGo. The background is divided into three horizontal sections: light red on the left, white in the center containing the form, and light orange on the right.

Create Account

Join TravelGo and start your journey today!

Invalid email or password.

Full Name
Enter your full name

Email Address
Enter your email

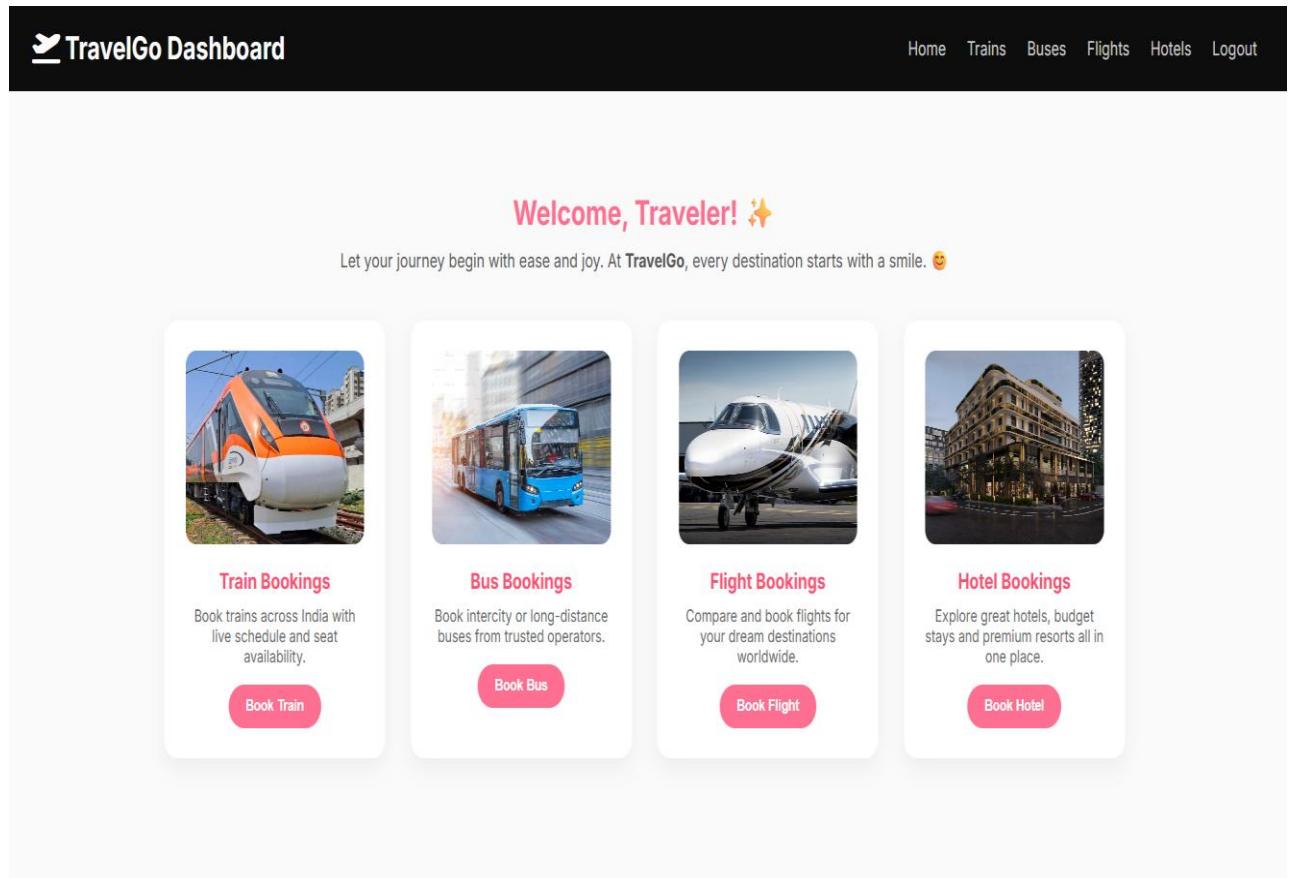
Password
Enter your password

Confirm Password
Confirm your password

Register

Already have an account? [Log in](#)

Dashboard page:



The dashboard features a dark header bar with the 'TravelGo Dashboard' logo and navigation links for Home, Trains, Buses, Flights, Hotels, and Logout.

Welcome, Traveler! 🌟

Let your journey begin with ease and joy. At TravelGo, every destination starts with a smile. 😊

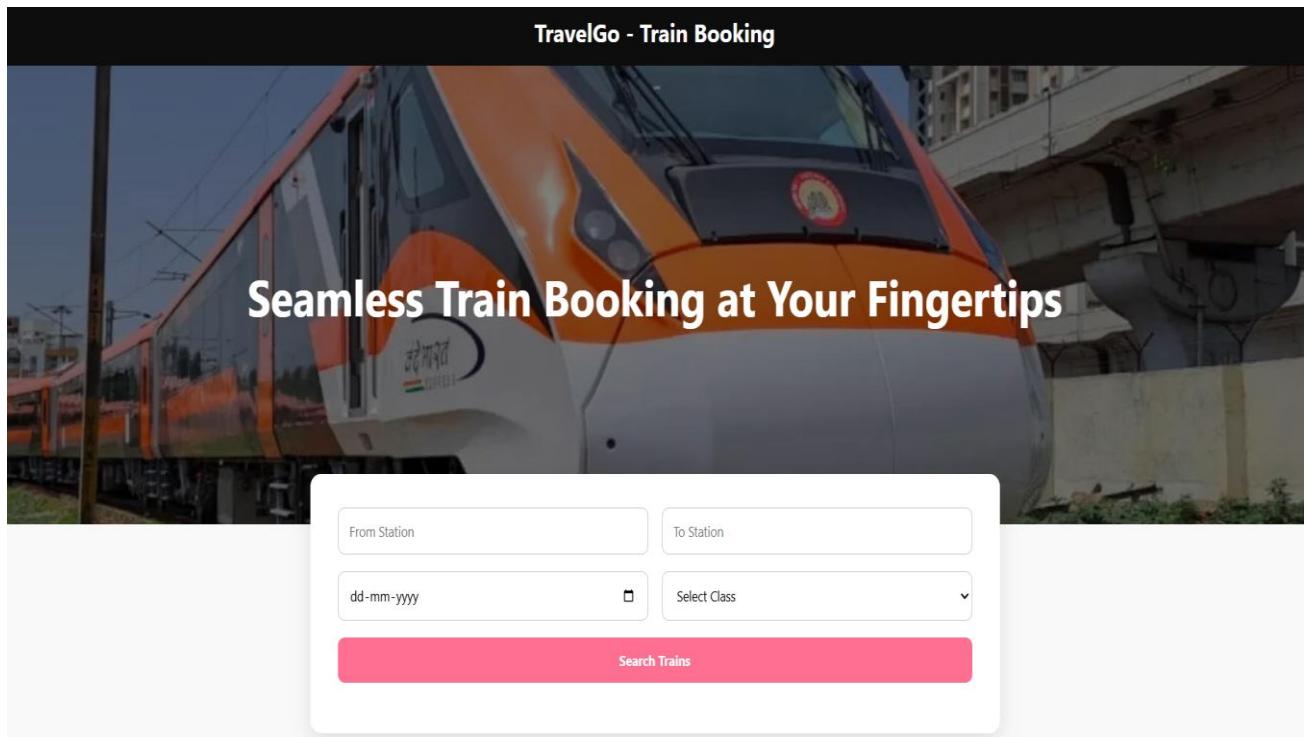
Train Bookings
Book trains across India with live schedule and seat availability.
[Book Train](#)

Bus Bookings
Book intercity or long-distance buses from trusted operators.
[Book Bus](#)

Flight Bookings
Compare and book flights for your dream destinations worldwide.
[Book Flight](#)

Hotel Bookings
Explore great hotels, budget stays and premium resorts all in one place.
[Book Hotel](#)

Train booking page:



The homepage features a large background image of a modern orange and silver high-speed train. At the top center, the text "TravelGo - Train Booking" is displayed. Below the image, the slogan "Seamless Train Booking at Your Fingertips" is prominently shown in white. A search form is centered below the slogan, containing fields for "From Station" and "To Station", a date input field "dd-mm-yyyy", a dropdown menu for "Select Class", and a large pink "Search Trains" button.

Available Trains

- Rajdhani Express
- Shatabdi Express
- Duronto Express
- Vande Bharat Express
- Garib Rath
- Tejas Express

Available Seats:

- A1-1
- A1-2
- A1-3

Confirm Booking

Bus booking page:

TravelGo - Bus Booking

Book Your Bus Journey Effortlessly

Search Buses

Available Buses

VRL Travels - AC Sleeper

Orange Travels - Non-AC

KSRTC Express

SRS Travels - Seater

KPN Travels - Deluxe

APSRTC Garuda

Available Seats:

S1 S2 S3

Confirm Booking

Flight booking page:

TravelGo - Flight Booking

Book Your Perfect Flight with TravelGo

Search Flights

Available Flights

IndiGo Airlines

Departure: 09:30 AM - 12:45 PM
From: Delhi (DEL) → To: Mumbai (BOM)

₹3499

[Book Now](#)

Air India

Departure: 01:00 PM - 03:30 PM
From: Delhi (DEL) → To: Bengaluru (BLR)

₹4250

[Book Now](#)

SpiceJet

Departure: 06:00 PM - 08:15 PM
From: Mumbai (BOM) → To: Chennai (MAA)

₹3899

[Book Now](#)

Available Seats:

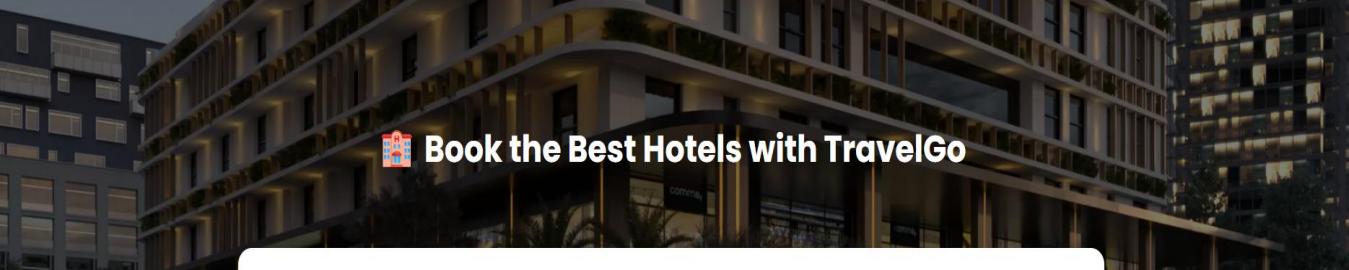
A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
A11	A12	A13	A14	A15	A16	A17	A18	A19	
A20	A21	A22	A23	A24	A25				

[Confirm Booking](#)

Hotel booking page:

TravelGo

Home Hotels Support Login



Book the Best Hotels with TravelGo

Search Hotels

Location dd-mm-yyyy dd-mm-yyyy 1 Adult

0 Children Standard Search



City Lights Hotel

★★★

Modern stay with rooftop lounge and city views.

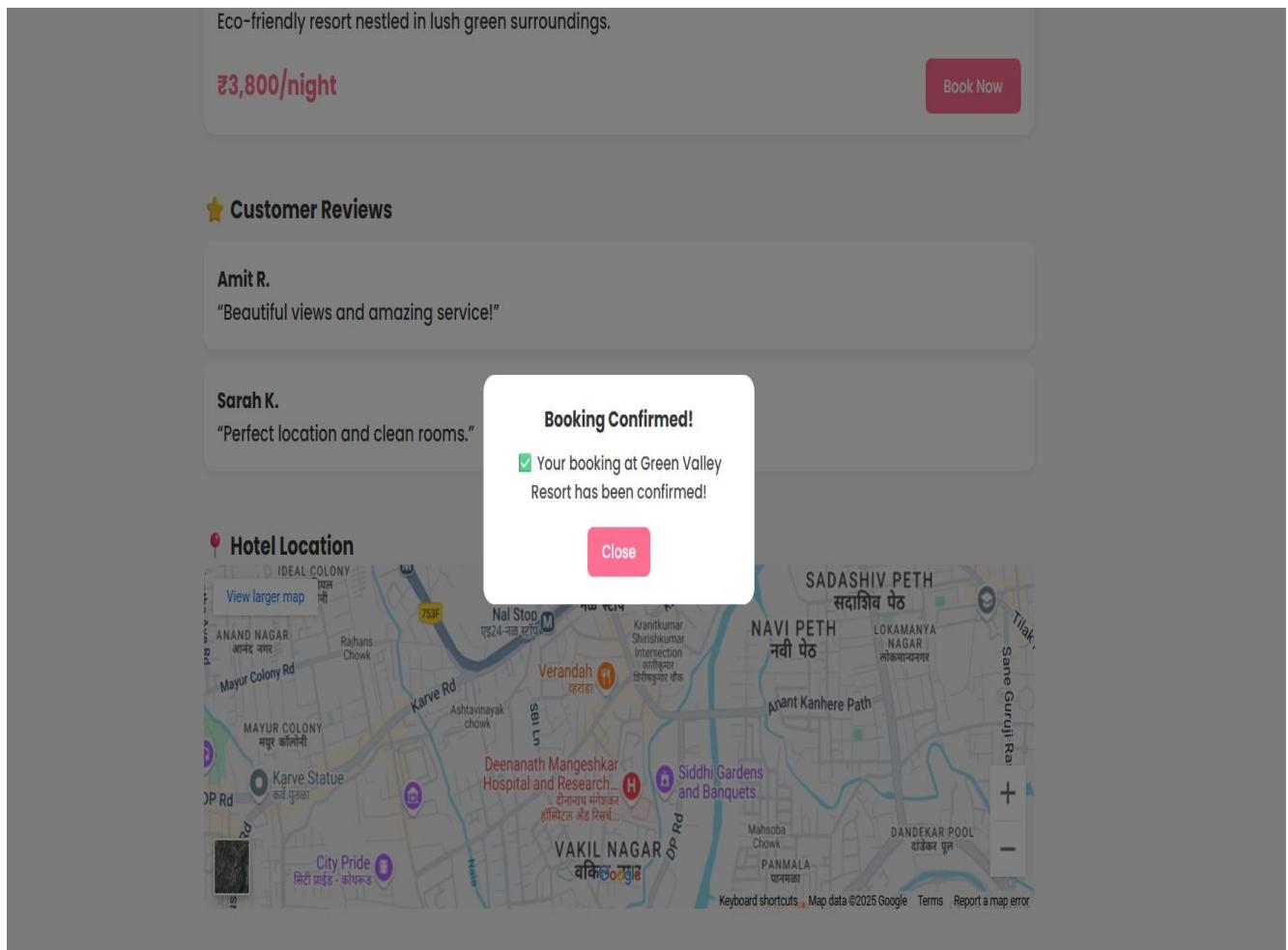
€3,000/night

Book Now



Green Valley Resort

★★★★★



Conclusion:

TravelGo successfully integrates modern cloud technologies to deliver a seamless, user friendly travel booking experience. By combining Flask, AWS EC2, DynamoDB, and SNS, the platform ensures real-time responsiveness, secure data handling, and instant user notifications. Its unified interface for booking buses, trains, flights, and hotels—along with dynamic dashboards and filtering options—makes it a scalable and reliable solution for modern travel needs. TravelGo not only enhances user convenience but also demonstrates the practical application of full-stack development and cloud services in solving real-world problems.