



TravelGo: A Cloud-Powered Real-Time TravelBooking Platform Using AWS

Hardware Required:

Processor: Intel i5 or equivalent (minimum). RAM: 4 GB (8 GB recommended for Full Stack MERN). Storage: 128 GB SSD or 128 GB HDD. Internet Connectivity: High-speed internet (minimum 10 Mbps per system). Additional: Audio-visual setup for interactive sessions (microphone, speakers, etc.).

Software Required:

Processor: Intel i5 or equivalent (minimum). RAM: 4 GB (8 GB recommended for Full Stack MERN). Storage: 128 GB SSD or 128 GB HDD. Internet Connectivity: High-speed internet (minimum 10 Mbps per system). Additional: Audio-visual setup for interactive sessions (microphone, speakers, etc.).

System Required:

Projector and Audio System for presentations in all labs/classrooms Classrooms/Labs are equipped with systems or provisions for students to join sessions with their own laptops.

Description:

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

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

Scenarios:

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

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

Scenario 2: Real-Time Booking Confirmation with AWS SNS

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

Scenario 3: Dynamic Dashboard with Personal Travel History

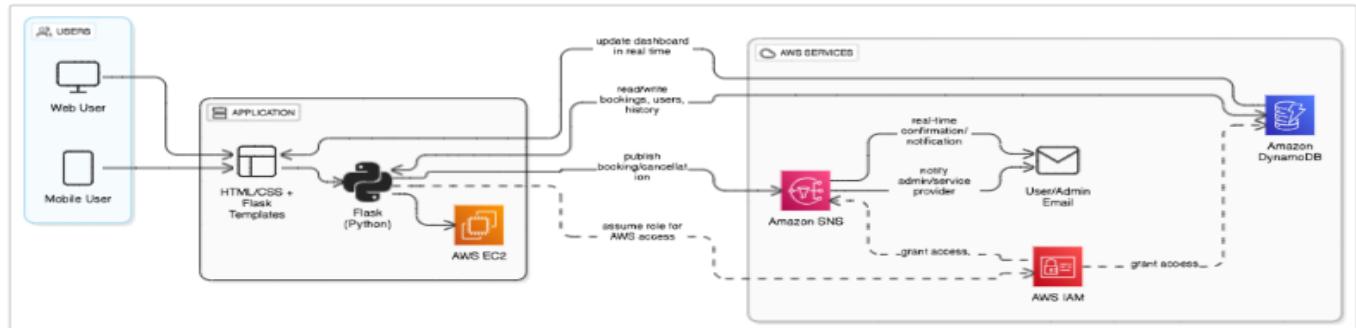
TravelGo features a dynamic user dashboard that displays all past and upcoming bookings for the logged-in user. For example, a user who has booked a flight and a hotel can view these bookings categorized by type, along with dates, price, and cancellation



options. Flask fetches this data from AWS DynamoDB, which persistently stores all user bookings. The dashboard UI, powered by responsive HTML/CSS and Flask templates, ensures users can review or manage bookings anytime, from any device, with real-time updates and quick cancellation workflows supported.

Architecture

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



Entity Relationship (ER) Diagram

An ER (Entity-Relationship) diagram visually represents the logical structure of a database by defining entities, their attributes, and the relationships between them. It helps organize data efficiently by illustrating how different components of the system interact and relate. This structured approach supports effective database normalization, data integrity, and simplified query design.





Pre-requisites

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

Project WorkFlow

Milestone 1. Backend Development and Application Setup

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

Milestone 2. AWS Account Setup and Login

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

Milestone 3. DynamoDB Database Creation and Setup

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

Milestone 4. SNS Notification Setup

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

Milestone 5. IAM Role Setup

- Create IAM Role
- Attach Policies

Milestone 6. EC2 Instance Setup

- Launch an EC2 instance to host the Flask application.



- Configure security groups for HTTP, and SSH access.

Milestone 7. Deployment on EC2

- Upload Flask Files
- Run the Flask App

Milestone 8. Testing and Deployment

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

Milestone 1: Web Application Development and Setup

Backend Development and Application Setup focuses on establishing the core structure of the application. This includes configuring the backend framework, setting up routing, and integrating database connectivity. It lays the groundwork for handling user interactions, data management, and secure access.

Important Instructions:

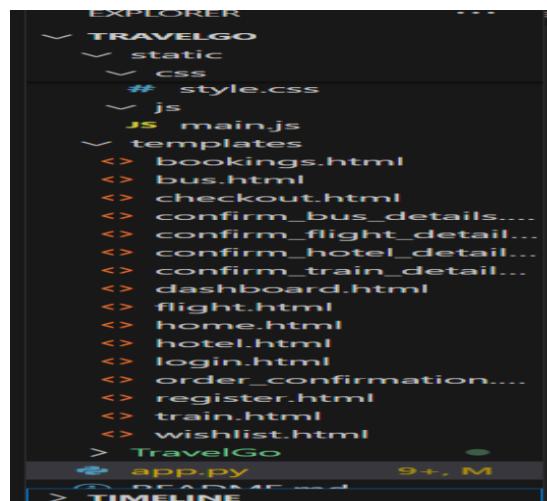
- Start by creating the necessary HTML pages and Flask routes (app.py) to build the core functionality of your application.
- During the initial development phase, store and retrieve data using Python dictionaries or lists locally. This will allow you to design, test, and validate your application logic without external database dependencies.
- Ensure your app runs smoothly with local data structures before integrating any cloud services.

Post Troven Access Activation:

- Once Troven Labs access is provided (valid for 3 hours), you must immediately proceed with Milestone 1 of your Guided Project instructions.
- At this point, modify your app.py and replace local dictionary/list operations with AWS services (such as DynamoDB, RDS, or others as per project requirements).
- Using the temporary credentials provided by Troven Labs, securely connect your application to AWS resources.
- Since the AWS configuration is lightweight and already instructed in the milestones, you should be able to complete the cloud integration efficiently within the allotted time.

LOCAL DEPLOYMENT

File Explorer:





Description:

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

- Activity 3.1: Develop the Backend Using Flask

Import libraries:

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

Description:

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

Flask App Initialization:

```
app = Flask(__name__)
app.secret_key = '584e6e31e50ccb19e9c09ecee1e9e6712bba7334fbea2489507d880cc76385fa'
```

Description:

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

Database Configuration:

```
REGION = 'ap-south-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')
bookings_table = dynamodb.Table('bookings')
```

Description:

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



- o Routes for Core Functionalities:

Home and registration routes:

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

# User registration route
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        # handle registration (email, username, password)
        return redirect(url_for('login'))
    return render_template('register.html')
```

Description:

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

Login routes:

```
# Login route
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        response = users_table.get_item(Key={'email': email})
        user = response.get('Item')

        if user and check_password_hash(user['hashed_password'], password):
            session['email'] = email
            session['username'] = user['username']
            users_table.update_one({"email":email}, {"$inc":{"login_count":1}})
            return redirect(url_for('dashboard'))
        else:
            error = 'Invalid credentials'
            return render_template('login.html', error=error)
    return render_template('login.html')
```

Description:

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



Dashboard and wishlist routes:

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

    raw_bookings = list(db.bookings.find({'user_email': session['email']}))
    bookings = []

    for b in raw_bookings:
        booking_type = b.get('type', 'flight')

        if booking_type == 'hotel':
            bookings.append({
                '_id': str(b.get('_id')),
                'booking_type': booking_type,
                'hotel_name': b.get('hotel_name', 'Unknown Hotel'),
                'location': b.get('location', ''),
                'checkin_date': b.get('checkin_date', 'N/A'),
                'checkout_date': b.get('checkout_date', 'N/A'),
                'room_type': b.get('room_type', ''),
                'num_rooms': b.get('num_rooms', 1),
                'num_nights': b.get('num_nights', 1),
                'total_price': b.get('total_price', 0),
                'booking_date': b.get('booking_date', '')
            })
        else:
            flight_class = b.get('class') or b.get('flight_class') or ''
            bookings.append({
                '_id': str(b.get('_id')),
                'booking_type': booking_type,
                'name': b.get('name') or b.get('airline') or 'Unknown',
                'source': b.get('source', ''),
                'destination': b.get('destination', ''),
                'travel_date': b.get('travel_date') or b.get('date') or 'N/A',
                'time': b.get('time', 'N/A'),
                'num_persons': b.get('num_persons') or b.get('passengers') or 1,
                'total_price': b.get('total_price', 0),
                'class': flight_class,
                'booking_date': b.get('booking_date', ''),
                'selected_seats': b.get('selected_seats', [])
            })

    return render_template('dashboard.html', bookings=bookings)
```

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

@app.route('/wishlist_data')
def wishlist_data():
    if 'email' not in session:
        return jsonify({'wishlist': []})
    items = list(wishlist_col.find({'email': session['email']}, {'_id': 0}))
    return jsonify({'wishlist': items})

@app.route('/remove_from_wishlist', methods=['POST'])
def remove_from_wishlist():
    if 'email' not in session:
        return jsonify({'success': False, 'message': 'Not logged in'})

    item_id = request.json.get('item_id')
    if not item_id:
        return jsonify({'success': False, 'message': 'Item ID not provided'})

    wishlist_col.delete_one({'email': session['email'], 'item_id': item_id})
    return jsonify({'success': True, 'message': 'Item removed from wishlist'})
```



Description:

The Dashboard shows all current bookings with details and cancellation options. The Wishlist lets users save preferred travel choices for future planning.

logout routes:

```
# ----- LOGOUT -----
@app.route('/logout')
def logout():
    session.clear() # Clear all user session data
    flash("Logged out successfully!", "success")
    return redirect(url_for('login'))
```

Description:

The logout route clears the user's session, securely logging them out of TravelGo. It then redirects them to the login page to end the current session.

Add_to_wishlist routes:

```
# Handle AJAX JSON posting - separate POST view
@app.route('/add_to_wishlist_json', methods=['POST'])
def add_to_wishlist_json():
    if 'email' not in session:
        return jsonify({'success': False, 'message': 'Not logged in'}), 401

    if not request.is_json:
        return jsonify({'success': False, 'message': 'Content-Type must be application/json'}), 415

    data = request.get_json()
    if not data.get('item_name') or not data.get('item_details'):
        return jsonify({'success': False, 'message': 'Missing fields'}), 400

    wishlist_col.insert_one([
        'email': session['email'],
        'item_id': data['item_name'], # or a unique ID
        'item_name': data['item_name'],
        'item_details': data['item_details'],
        'item_image': data.get('item_image', ''),
        'added_date': datetime.utcnow().strftime('%Y-%m-%d %H:%M:%S')
    ])
    return jsonify({'success': True, 'message': 'Item added to wishlist'})
```

Description:

The add to wishlist route saves selected travel options (bus, train, flight, or hotel) to the user's wishlist in the database. It helps users bookmark preferred choices for future booking.

Testimonials routes:

```
@app.route('/testimonials')
def testimonials():
    print("Testimonials route was triggered")
    return render_template('testimonials.html')
```



Description:

The testimonials route displays feedback and reviews shared by users about their TravelGo experience. It helps build trust and showcases user satisfaction with the platform.

Bookings routes:

```
# Separate Bookings page route
@app.route('/bookings', methods=['GET', 'POST'])
def bookings():
    if 'email' not in session:
        return redirect(url_for('login'))

    # Handle booking cancellation on POST
    if request.method == 'POST':
        booking_id = request.form.get('booking_id')
        if booking_id:
            from bson.objectid import ObjectId
            db.bookings.delete_one({'_id': ObjectId(booking_id), 'user_email': session['email']})
            # You can flash a message or redirect here
            return redirect(url_for('bookings'))

    # Fetch bookings for this user
    raw_bookings = list(db.bookings.find({'user_email': session['email']}))

    # Format bookings for template
    bookings = []
    for b in raw_bookings:
        booking_type = b.get('booking_type') or b.get('type') or 'flight' # adjust to your data

        if booking_type == 'hotel':
            bookings.append({
                '_id': str(b.get('_id')),
                'booking_type': 'hotel',
                'hotel_name': b.get('hotel_name', 'Unknown Hotel'),
                'location': b.get('location', ''),
                'checkin_date': b.get('checkin_date', 'N/A'),
                'checkout_date': b.get('checkout_date', 'N/A'),
                'room_type': b.get('room_type', ''),
                'num_rooms': b.get('num_rooms', 1),
                'num_nights': b.get('num_nights', 1),
                'total_price': b.get('total_price', 0),
            })
        else:
            bookings.append({
                '_id': str(b.get('_id')),
                'booking_type': booking_type,
                'name': b.get('name') or b.get('airline') or 'Unknown',
                'source': b.get('source', ''),
                'destination': b.get('destination', ''),
                'travel_date': b.get('travel_date') or b.get('date') or 'N/A',
                'time': b.get('time', 'N/A'),
                'num_persons': b.get('num_persons') or b.get('passengers') or 1,
                'total_price': b.get('total_price', 0),
                'class': b.get('class') or b.get('flight_class') or '',
                'selected_seats': b.get('selected_seats', None),
            })
    print("Bookings for user:", session['email'], raw_bookings)

    return render_template('bookings.html', bookings=bookings)
```

Description:

The bookings route displays all confirmed travel bookings (bus, train, flight, hotel) made by the user. It allows users to view, manage, and cancel their existing reservations.

Initiating Flask app:

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Description:

Start the Flask application on host 0.0.0.0 and port 80 in debug mode to enable live reloads and detailed error logs during development.



Milestone 2 : AWS Account Setup

Important Notice: Use Troven Labs for AWS Access

Students are strictly advised not to create their own AWS accounts, as doing so may incur charges. Instead, we have set up a dedicated section called “Labs” on the Troven platform, which provides temporary and cost-free access to AWS services.

Once your website is locally deployed and fully functional, you must proceed with integrating AWS services only through the Troven Labs environment. This ensures secure, controlled access to AWS resources without any risk of personal billing.

All steps involving AWS (such as deploying to EC2, connecting to DynamoDB, or using SNS) must be carried out within the Troven Labs platform, as we've configured temporary credentials for each student.

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.

The screenshot shows the AWS sign-up page. At the top, there's a navigation bar with links for About AWS, Contact Us, Support, English, My Account, Sign In, and Complete Sign Up. Below the navigation, a large heading says "Complete your AWS registration". A sub-headline states: "Millions of customers are using AWS cloud solutions to build applications with increased flexibility, scalability, security, and reliability". There are three callout boxes: "AWS Free Tier" (describing the free tier), "Customer success stories" (linking to company case studies), and "Contact us" (with a message input field and a yellow speech bubble icon). A "Complete sign-up" button is located at the bottom left.

The screenshot shows the "Sign up for AWS" form. It has fields for "Root user email address" (with a note about account recovery and privacy), "AWS account name" (with a note about changing it later), and "Verify email address". There's also an "OR" option and a "Sign in to an existing AWS account" link. The background features a blue wireframe cube and a hand holding a stack of cubes.



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

The screenshot shows the AWS Sign In interface on the left, where a user is selecting 'Root user' and entering their email address. On the right, there is a purple banner for the 'AI Use Case Explorer' which reads: 'Discover AI use cases, customer success stories, and expert-curated implementation plans'. Below the banner is a link 'Explore now >'.

Milestone 3 : 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.

Navigate to the DynamoDB

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

The screenshot shows the AWS Services search results for 'dynamoDB'. The search bar at the top has 'dynamoDB' typed in. The results list 'DynamoDB' as a Managed NoSQL Database, followed by 'Amazon DocumentDB' (Fully-managed MongoDB-compatible database service), 'CloudFront' (Global Content Delivery Network), and 'Athena' (Serverless interactive analytics service). Below the main results, there are sections for 'Features' and 'Settings'.

The screenshot shows the Amazon DynamoDB Dashboard. On the left, there's a navigation sidebar with links like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, Settings, and DAX (Clusters, Subnet groups, Parameter groups, Events). The main area has two sections: 'Alarms (0) Info' and 'DAX clusters (0) Info'. Both sections have search bars ('Find alarms' and 'Find clusters'), status indicators ('Status' and 'No custom alarms / No clusters'), and buttons ('Manage in CloudWatch', 'View details', 'Create cluster'). To the right, there's a 'Create resources' section with a 'Create table' button, a note about the Amazon DynamoDB Accelerator (DAX), and a 'What's new' section with a note about AWS Cost Management.

The screenshot shows the Amazon DynamoDB Tables page. The left sidebar includes links for Dashboard, Tables (selected), Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, and Integrations. The main table view shows 'Tables (0) Info' with a search bar ('Find tables') and various filter dropdowns ('Any tag key', 'Any tag value', 'Actions', 'Delete', 'Create table'). A message states 'You have no tables in this account in this AWS Region.' There's also a 'Create table' button at the bottom.

Create an DynamoDB table for storing data

- Create a travel-Users table for storing registered user details with partition key “Email” with type String and click on create tables.

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.

travel-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 class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	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.

[Cancel](#)

[Create table](#)

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

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.

Bookings

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.

booking_id

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 these settings now, choose 'Customize settings'.

Customize settings

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

Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	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.

[Cancel](#) [Create Table](#)

The trains table was created successfully.

Tables (3) Info

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion prot.
bookings	Active	user_email (\$)	booking_date (\$)	0	0	Off
trains	Active	train_number (\$)	date (\$)	0	0	Off
travelgo_users	Active	email (\$)	-	0	0	Off

Milestone 4 : SNS Notification Setup

Amazon SNS is a fully managed messaging service that enables real-time notifications through channels like SMS, email, or app endpoints. You create topics, configure subscriptions, and integrate SNS into your app to send notifications based on specific events.



SNS Topics for email notifications

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

The screenshot shows the AWS search interface with the query 'sns' entered. The search results page displays two items: 'Simple Notification Service' and 'Route 53 Resolver'. The SNS entry is highlighted with a blue border and includes a star icon indicating it's a popular service. A 'Show more ▶' link is visible at the top right of the results area.

The screenshot shows the Amazon SNS dashboard. On the left, there is a sidebar with links: Dashboard, Topics, Subscriptions, Mobile (Push notifications, Text messaging (SMS)), and a New Feature notification about FIFO topic support. The main content area features a large heading 'Amazon Simple Notification Service' with the subtext 'Pub/sub messaging for microservices and serverless applications.' Below this, a paragraph describes the service's purpose. To the right, there are two sections: 'Create topic' (with a 'Topic name' input field containing 'MyTopic' and a 'Next step' button) and 'Pricing'. At the bottom, there is a 'Topics (0)' table with columns for Name, Type, and ARN, and a 'Create topic' button.

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

The screenshot shows the 'Topics' section of the SNS dashboard. The left sidebar has a 'Topics' link selected. The main area shows a table titled 'Topics (0)' with columns for Name, Type, and ARN. A search bar is at the top of the table. Below the table, a message says 'No topics' and 'To get started, create a topic.' with a 'Create topic' button.



Choose Standard type for general notification use cases and Click on 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

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.

The screenshot shows the AWS SNS Topics page. On the left, there's a navigation sidebar with 'Amazon SNS' selected under 'Topics'. The main area displays a success message: 'Topic Travelgo created successfully. You can create subscriptions and send messages to them from this topic.' Below this, the 'Travelgo' topic details are shown in a card:

Details	
Name	Travelgo
ARN	arn:aws:sns:us-east-1:600627341644:Travelgo
Type	Standard
Display name	-
Topic owner	600627341644

At the bottom right of the card are three buttons: 'Edit', 'Delete', and 'Publish message'.



Subscribe users and admin

- Subscribe users (or admin staff) to this topic via Email. When a book request is made, notifications will be sent to the subscribed emails.

Create subscription

Details

Topic ARN
arn:aws:sns:ap-south-1:557690616836:BookingConfirmation X

Protocol
The type of endpoint to subscribe
Email ▼

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

ⓘ After your subscription is created, you must confirm it. [Info](#)

▶ **Subscription filter policy - optional** [Info](#)
This policy filters the messages that a subscriber receives.

▶ **Redrive policy (dead-letter queue) - optional** [Info](#)
Send undeliverable messages to a dead-letter queue.

[Cancel](#) Create subscription

- After subscription request for the mail confirmation

BookingConfirmation Edit Delete Publish message

Details <p>Name BookingConfirmation</p> <p>ARN arn:aws:sns:ap-south-1:557690616836:BookingConfirmation</p> <p>Type Standard</p>	<p>Display name -</p> <p>Topic owner 557690616836</p>
---	---

[Subscriptions](#) [Access policy](#) [Data protection policy](#) [Delivery policy \(HTTP/S\)](#) [Delivery status logging](#) [Encryption](#) [Tags](#) [Integrations](#)

Subscriptions (1)				Edit Delete Request confirmation Confirm subscription Create subscription
Q. Search				◀ 1 ▶ ⌂
ID	Endpoint	Status	Protocol	▼
Pending confirmation	instantlibrary2@gmail.com	Pending confirmation	EMAIL	▼

Navigate to the subscribed Email account and Click on the confirm subscription in the AWS Notification- Subscription Confirmation mail.

AWS Notification - Subscription Confirmation External Spam ×

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

Tue, Jul 1, 11:21AM (3 days ago) ★ ↵ ⋮

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

Report not spam i

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](#).



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.

BookingConfirmation

Details

Name BookingConfirmation	Display name -
ARN arn:aws:sns:ap-south-1:557690616836:BookingConfirmation	Topic owner 557690616836
Type Standard	

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

Subscriptions (1)

ID	Endpoint	Status	Protocol
da5ad931-7e8f-4ac8-b771-ec2e0cfba23	instantlibrary2@gmail.com	Confirmed	EMAIL

Milestone 5 : IAM Role Setup

IAM (Identity and Access Management) role setup involves creating roles that define specific permissions for AWS services. To set it up, you create a role with the required policies, assign it to users or services, and ensure the role has appropriate access to resources like EC2, S3, or RDS. This allows controlled access and ensures security best practices in managing AWS resources.

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 service page. A search bar at the top right contains the text 'iam'. On the left, a sidebar lists navigation options: Services, Features, Resources (which is highlighted in blue), Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main content area is titled 'Search results for "iam"' and displays a list of services:

- IAM** ☆ Manage access to AWS resources
- IAM Identity Center** ☆ Manage workforce user access to multiple AWS accounts and cloud applications
- Resource Access Manager** ☆ Share AWS resources with other accounts or AWS Organizations
- AWS App Mesh** ☆ Easily monitor and control microservices

The screenshot shows the AWS IAM Roles page. At the top, there's a search bar and a 'Create role' button. Below that, a table lists six roles. The columns are 'Role name', 'Trusted entities', and 'Last activity'. Each row has a 'Delete' button.

This screenshot shows the 'Select trusted entity' step in the 'Create role' wizard. It includes sections for 'Trusted entity type' (AWS service, AWS account, Web identity), 'Use case' (allow EC2 to call AWS services), and 'Service or use case' (set to 'EC2'). A list of EC2 permissions is shown, with 'EC2' selected. Buttons for 'Cancel' and 'Next Step' are at the bottom.

This screenshot shows the 'Add permissions' step in the 'Create role' wizard. It displays a list of 'Permissions policies' (1/955) with a search bar and filter. Two policies are selected: 'AmazonDynamoDBFullAccess' and 'AmazonSNSFullAccess'. A 'Set permissions boundary - optional' link is at the bottom. Buttons for 'Cancel', 'Previous', and 'Next Step' are visible.

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.

This screenshot shows the 'Add permissions' step in the 'Create role' wizard again. The search bar now shows 'sns'. The 'AmazonSNSFullAccess' policy is selected, along with several other SNS-related policies: 'AmazonSNSReadOnlyAccess', 'AmazonSQSFullAccess', 'AmazonSQSSimpleQueueRole', 'AmazonSimpleEmailServiceEmailDeliveryRole', and 'AmazonSimpleDeviceDefenderPublishFindingsToSNSMigrationAction'. A 'Set permissions boundary - optional' link is at the bottom. Buttons for 'Cancel', 'Previous', and 'Next Step' are at the bottom right.

Name, review, and create

Step 1: Select trusted entities

Trust policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Step 2: Add permissions

Permissions policy summary

Type	Attached as
AWS managed	Permissions policy
AWS managed	Permissions policy

Step 3: Add tags

Add tags - optional. Tags are key-value pairs 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 Create role

IAM > Roles > sns_Dynamodb_role

sns_Dynamodb_role Info

Allows EC2 instances to call AWS services on your behalf.

Summary

Creation date October 15, 2024, 23:06 (UTC+05:30)	ARN arn:aws:iam::557690616836:role/sns_Dynamodb_role	Instance profile ARN arn:aws:iam::557690616836:instance-profile/sns_Dynamodb_role
Last activity 6 days ago	Maximum session duration 1 hour	

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
AmazonDynamoDBFullAccess	AWS managed	2
AmazonSNSFullAccess	AWS managed	2

Milestone 6 : EC2 Instance Setup

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

To set up a public EC2 instance, choose an appropriate Amazon Machine Image (AMI) and instance type. Ensure the security group allows inbound traffic on necessary ports (e.g., HTTP/HTTPS for web applications). After launching the instance, associate it with an elastic IP for consistent public access, and configure your application or services to be publicly accessible.

travelgo Public

main 1 Branch 0 Tags

Go to file t +

anjali5944 Add requirements.txt 4b12927 · 4 days ago

static	Initial commit
templates	Initial commit
README.md	Initial commit
app.py	Updated app.py with DynamoDB integration
requirements.txt	Add requirements.txt

Load your Project Files to Github

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

- Click on Launch instance to launch EC2 instance

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



- Create and download the key pair for Server access.

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

Instance type

t2.micro Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0124 USD per Hour
On-Demand Windows base pricing: 0.017 USD per Hour
On-Demand RHEL base pricing: 0.0268 USD per Hour
On-Demand SUSE base pricing: 0.0124 USD per Hour

Additional costs apply for AMIs with pre-installed software

▼ 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

Select [Create new key pair](#)

Create key pair X

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

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

Key pair type

RSA
RSA encrypted private and public key pair

ED25519
ED25519 encrypted private and public key pair

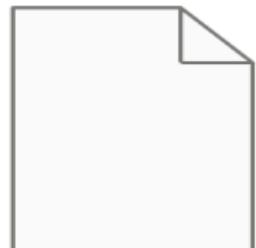
Private key file format

.pem
For use with OpenSSH

.ppk
For use with PuTTY

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

[Cancel](#) Create key pair



TravelGoneNew.pem



The screenshot shows the 'Launch instance' wizard in the AWS EC2 console. It includes sections for:

- Description:** Amazon Linux 2023 is a modern, general purpose Linux-based OS.
- Architecture:** 64-bit (x86)
- Boot mode:** uefi-preferred
- AMI ID:** ami-078264b8ba71bc45e
- Username:** ec2-user
- Verified provider:** Green button

Instance type: t2.micro (Free tier eligible) - 1 vCPU, 1 GB Memory. Current generation: true. On-Demand Windows base pricing: 0.0124 USD per Hour. On-Demand Linux base pricing: 0.0117 USD per Hour. On-Demand GPU base pricing: 0.0124 USD per Hour. On-Demand SUSE base pricing: 0.0124 USD per Hour.

Key pair (login): InstantLibrary (required) - dropdown menu. Create new key pair button.

Summary: Number of instances: 1. Software Image (AMI): Amazon Linux 2023 AMI 2023.5.2... (read more). Virtual server type (instance type): t2.micro. Firewall (security group): New security group. Storage (volumes): 1 volume(s) - 8 GiB. A callout box details the Free tier: 1,750 hours of t2.micro usage (or t3.micro in the Region) in which t2.micro is unavailable) instance usage on free-tier AMIs per month, 750 hours of paid t2.micro usage per month, 30 GB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the Internet.

Buttons: Cancel, Preview code, Launch instance (orange button).

Launch An EC2 instance to host The flask

Launch EC2 Instance

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

The screenshot shows the AWS Services navigation bar with 'Services' selected. The search bar shows 'ec2'. The main content area displays:

- Services:** EC2 (Virtual Servers in the Cloud), EC2 Image Builder (A managed service to automate build, customize and deploy OS images), Recycle Bin (Protect resources from accidental deletion).
- Features:** Resources (New), Documentation, Knowledge articles, Marketplace, Blog posts, Events, Tutorials.

- Click on Launch instance to launch EC2 instance

The screenshot shows the EC2 Instances page. The left sidebar includes:

- EC2 Dashboard
- EC2 Global View
- Events
- Instances (selected)
- Instances Types
- Launch Templates
- Spot Requests
- Savings Plans

The main content area shows:

- Instances Info:** Last updated less than a minute ago. Buttons: Connect, Instance state, Actions, Launch instances.
- Search:** Find Instance by attribute or tag (case-sensitive).
- Table Headers:** Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS.
- Message:** No instances. You do not have any instances in this region. Launch instances button.

The screenshot shows the 'Launch an instance' wizard. The steps are:

- EC2 > Instances > Launch an instance
- Launch an instance (Info)
- Name and tags (Info)

The 'Name and tags' step includes:

- Name:** InstantLibraryApp
- Add additional tags:** button.

Summary: Number of instances: 1. Software Image (AMI): Amazon Linux 2023 AMI 2023.5.2... (read more). Virtual server type (instance type): t2.micro. Firewall (security group): New security group.



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

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.

Instance type Info | Get advice

Instance type

t2.micro Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0124 USD per Hour
On-Demand Windows base pricing: 0.017 USD per Hour
On-Demand RHEL base pricing: 0.0268 USD per Hour
On-Demand SUSE base pricing: 0.0124 USD per Hour

All generations Compare instance types

Additional costs apply for AMIs with pre-installed software

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

Select Create new key pair

Create key pair

Key pair name: InstantLibrary

Key pairs allow you to connect to your instance securely. The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type:

RSA RSA encrypted private and public key pair

ED25519 ED25519 encrypted private and public key pair

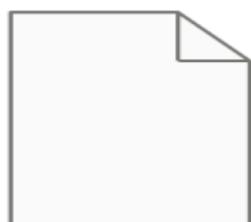
Private key file format:

.pem For use with OpenSSH

.ppk For use with PuTTY

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

Cancel Create key pair



TravelGoneNew.pem



Amazon

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-078264b8ba71bc45e	Username	ec2-user	Verified provider
--------------	--------------	-----------	----------------	--------	-----------------------	----------	----------	-------------------

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

Instance type
t2.micro 1 vCPU 1 GiB Memory Current generation: true Free tier eligible
On-Demand Linux base pricing: 0.0124 USD per Hour
On-Demand Windows base pricing: 0.017 USD per Hour
On-Demand RHEL base pricing: 0.0268 USD per Hour
On-Demand SUSE base pricing: 0.0124 USD per Hour

Additional costs apply for AMIs with pre-installed software

▼ 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 InstantLibrary [Create new key pair](#)

▼ Summary

Number of instances [Info](#)
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.5.2...read more
ami-078264b8ba71bc45e

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

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

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million IOPS, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

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

Configure security groups for HTTP, and SSH access

▼ Network settings [Info](#)

VPC - required [Info](#)
vpc-05cd7b6f19dd7211 (default) 172.31.0.0/16

Subnet [Info](#)
No preference [Create new subnet](#)

Auto-assign public IP [Info](#)
Enable Additional charges apply when outside of free tier allowance

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

Create security group [Select existing security group](#)

Security group name - required Launch-wizard-6 This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, space, and _-/. Description - required [Info](#) Launch-wizard-6 created 2025-04-14T07:20:40Z

Inbound Security Group Rules

▼ Security group rule 1 (TCP; 22; 0.0.0.0/0) Remove

Type [Info](#): ssh Protocol [Info](#): TCP Port range [Info](#): 22 Description - optional [Info](#): e.g. SSH for admin desktop

Source type [Info](#): Anywhere 0.0.0.0/0 X

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

Type [Info](#): HTTP Protocol [Info](#): TCP Port range [Info](#): 80 Description - optional [Info](#): e.g. SSH for admin desktop

Source type [Info](#): Anywhere 0.0.0.0/0 X

▼ Security group rule 5 (TCP; 5000; 0.0.0.0/0) Remove

Type [Info](#): Custom TCP Protocol [Info](#): TCP Port range [Info](#): 5000 Description - optional [Info](#): e.g. SSH for admin desktop

Source type [Info](#): Anywhere 0.0.0.0/0 X

⚠️ 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.

Add security group rule



▼ Summary

Number of instances | Info

1

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

ami-0e35ddab05955cf57

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

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

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

[Cancel](#)

[Launch instance](#)

[Preview code](#)

The screenshot shows the AWS EC2 Instances page. A green success message at the top states: "Successfully attached Studentuser to Instance i-00b88327d374fd912". Below this, the "Instances (1/1)" table lists one instance: "TravelGoProject" (Instance ID: i-00b88327d374fd912, Status: Running, Type: t2.micro). The "Actions" dropdown menu for this instance includes options like "Connect", "Launch instances", and "Modify IAM role". The left sidebar shows navigation links for EC2, Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, and Images.

- 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 EC2 Instances page with a table of instances. One instance, "travel-booking" (Instance ID: i-0c95169c57a272969, Status: Running, Type: t2.micro), is selected. The "Actions" dropdown menu for this instance includes options like "Connect", "View details", "Manage instance state", "Instance settings", "Networking", "Security" (selected), "Image and templates", and "Monitor and troubleshoot". The left sidebar shows navigation links for EC2, Instances, and Modify IAM role.

[EC2 > Instances > i-0c95169c57a272969 > Modify IAM role](#)

Modify IAM role

Attach an IAM role to your instance.

Instance ID

i-0c95169c57a272969 (travel-booking)

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.

Choose IAM role

Q |

No IAM Role

Choose this option to detach an IAM role

ec2_dynamodb.sns

arn:aws:iam::55700616836:instance-profile/ec2_dynamodb_sns

Create new IAM role

Cancel

Update IAM role



- Now connect the EC2 with the files

Connect to instance [Info](#)
Connect to your instance i-0c95169c57a272969 (travel-booking) using any of these options

EC2 Instance Connect Session Manager SSH client EC2 serial console

Instance ID [i-0c95169c57a272969 \(travel-booking\)](#)

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 [3.110.28.122](#)

IPv6 address

Username [ubuntu](#)

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

Q ubuntu [X](#)

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

[Cancel](#) [Connect](#)

```
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1024-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Tue Apr 15 15:33:31 UTC 2025

System load: 0.0          Processes:           105
Usage of /: 36.2% of 6.71GB   Users logged in: 0
Memory usage: 20%          IPv4 address for enx0: 172.31.12.110
Swap usage: 0%             IPv6 address for enx0: fe80::56d4:1ff:fe12:110

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

  https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

63 updates can be applied immediately.
33 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Tue Apr 15 15:33:32 2025 from 13.233.177.5
ubuntu@ip-172-31-12-110:~$
```

i-0c95169c57a272969 (travel-booking)
PublicIPs: 13.203.222.149 PrivateIPs: 172.31.12.110

Milestone 7: Deployment using EC2

Deployment on an EC2 instance involves launching a server, configuring security groups for public access, and uploading your application files. After setting up necessary dependencies and environment variables, start your application and ensure it's running on the correct port. Finally, bind your domain or use the public IP to make the application accessible online.

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
```

Clone Your Flask Project from GitHub

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

Run: [git clone https://github.com/anjali5944/travelgo.git](https://github.com/anjali5944/travelgo.git)

This will download your project to the EC2 instance.

- To navigate to the project directory, run the following command:
`cd <Travelgo>`
- Once inside the project directory, configure and run the Flask application by executing the following command with elevated privileges:
 - Run the Flask Application
 - `export SNS_TOPIC_ARN=arn:aws:sns:ap-south-1:557690616836:BookingConfirmation`
 - `git pull origin main` (Only if you made changes in git and want to reflect them in EC2 terminal.)

Install requirements

```
pip install flask boto3
```

- Launch the Flask app:

```
sudo -E venv/bin/python3 app.py
```

Verify the Flask app is running:

```
http://34.226.154.175:5000/
```

- Run the Flask app on the EC2 instance

```
Last login: Mon Apr 14 16:27:44 2025 from 13.233.177.5
ubuntu@ip-172-31-12-110:~/Travel-Booking-website_Ec2_Dynamodb_SNS
ubuntu@ip-172-31-12-110:~/Travel-Booking-website_Ec2_Dynamodb_SNS$ cd travel_booking_system
ubuntu@ip-172-31-12-110:~/Travel-Booking-website_Ec2_Dynamodb_SNS/travel_booking_system$ export SNS_TOPIC_ARN=arn:aws:sns:ap-south-1:557690616836:BookingConfirmation
ubuntu@ip-172-31-12-110:~/Travel-Booking-website_Ec2_Dynamodb_SNS/travel_booking_system$ git pull origin main
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (5/5), 1.68 KiB | 429.00 KiB/s, done.
From https://github.com/AlekhyaaPenubakula/Travel-Booking-website_Ec2_Dynamodb_SNS
 * branch      main      -> FETCH_HEAD
 * branch      2289f47..397e3a1 main      -> origin/main
Updating 2289f47..397e3a1
Fast-forward
 travel_booking_system/templates/hotels.html | 107 ++++++-----+
 1 file changed, 63 insertions(+), 44 deletions(-)
ubuntu@ip-172-31-12-110:~/Travel-Booking-website_Ec2_Dynamodb_SNS/travel_booking_system$ sudo -E venv/bin/python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://172.31.12.110:80
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 661-186-119
```

- Access the website through:

```
PublicIPs: http://34.226.154.175:5000/
```

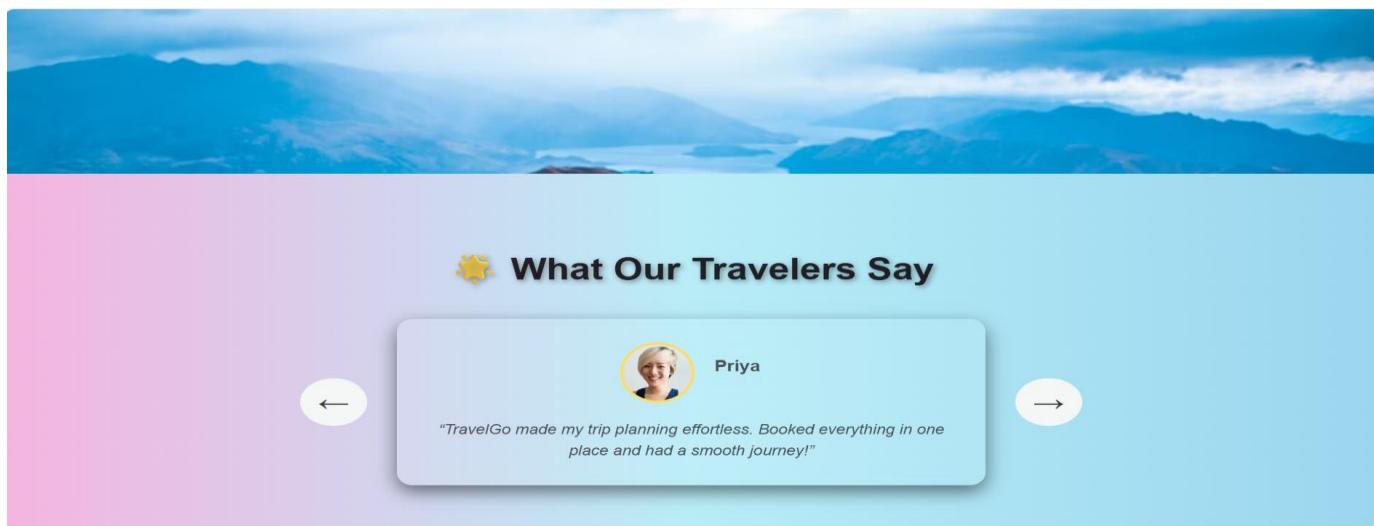
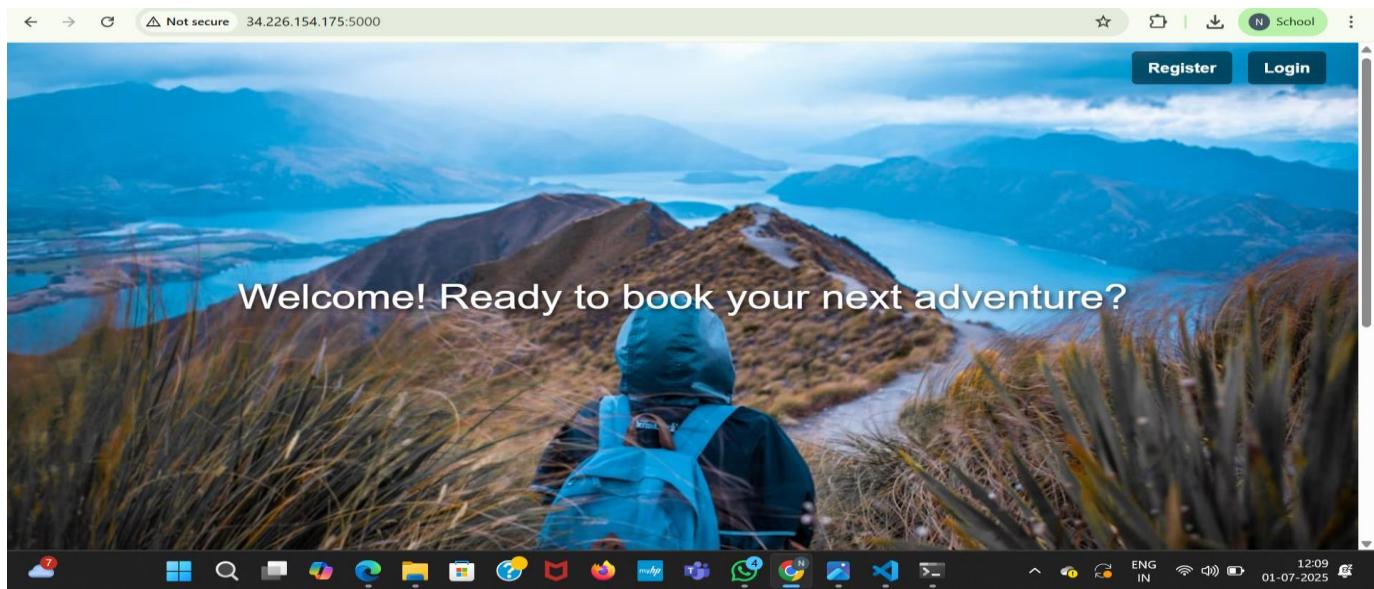
Milestone 8 : Testing and Deployment

Testing and deployment involve verifying that your application works as expected before making it publicly accessible. Start by testing locally or on a staging environment to catch bugs and ensure functionality. Once tested, deploy the application to an EC2 instance, configure necessary services, and perform a final round of live testing to confirm everything runs smoothly in the production environment.

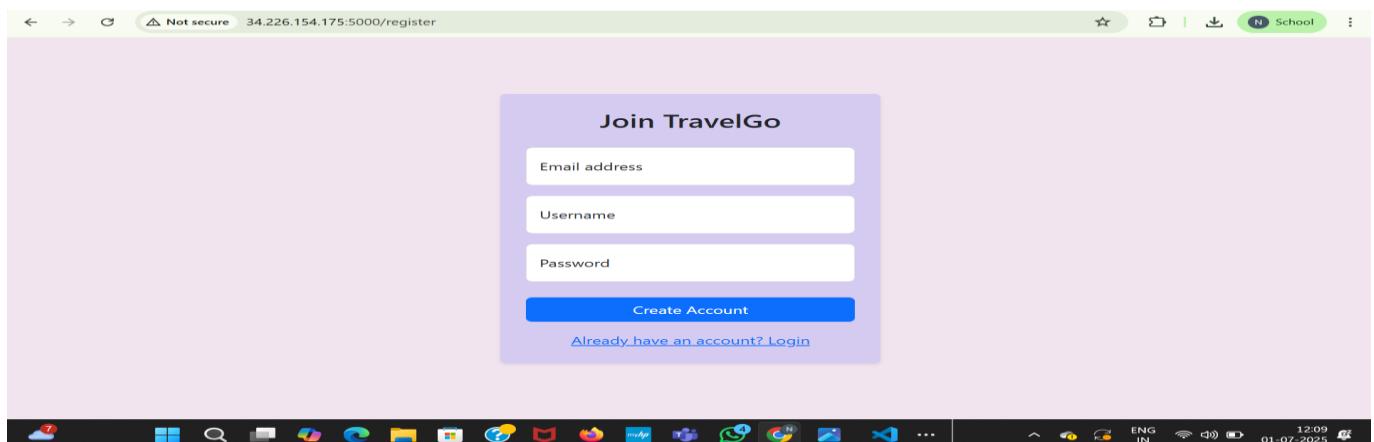


Functional Testing to verify the Project

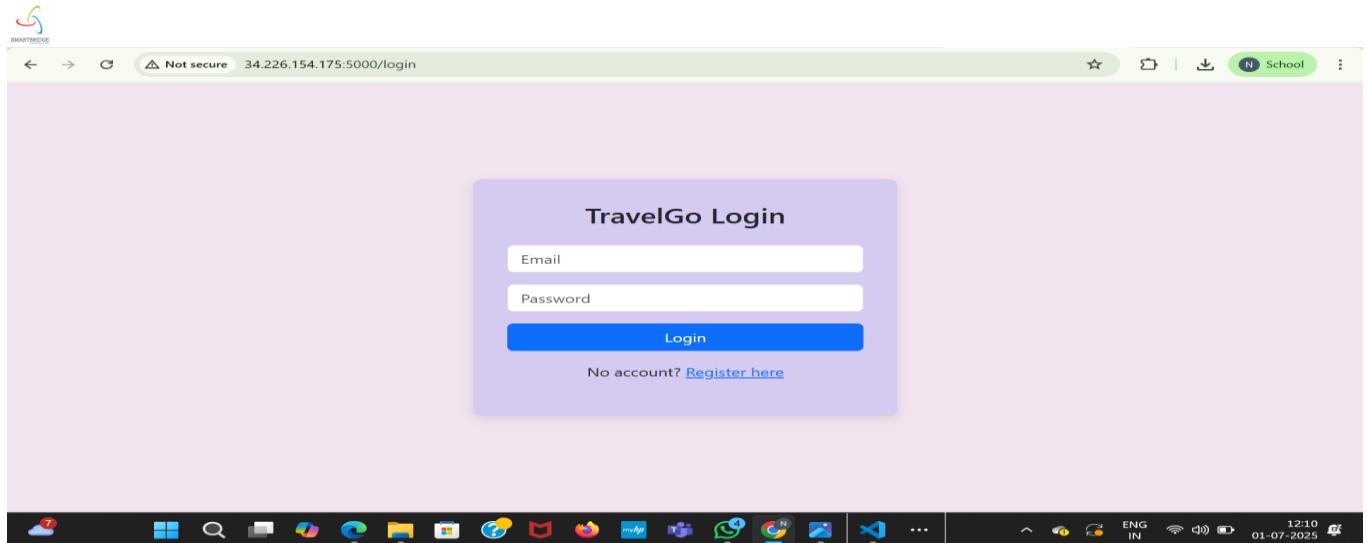
Home Page:



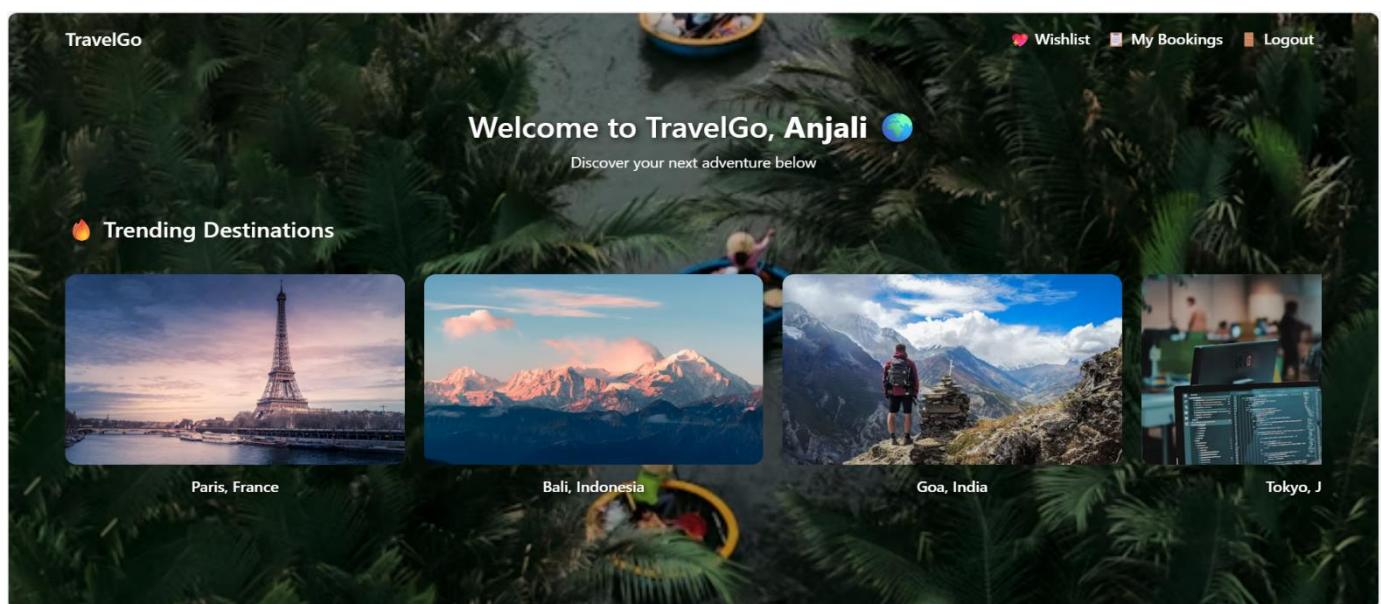
Register Page:



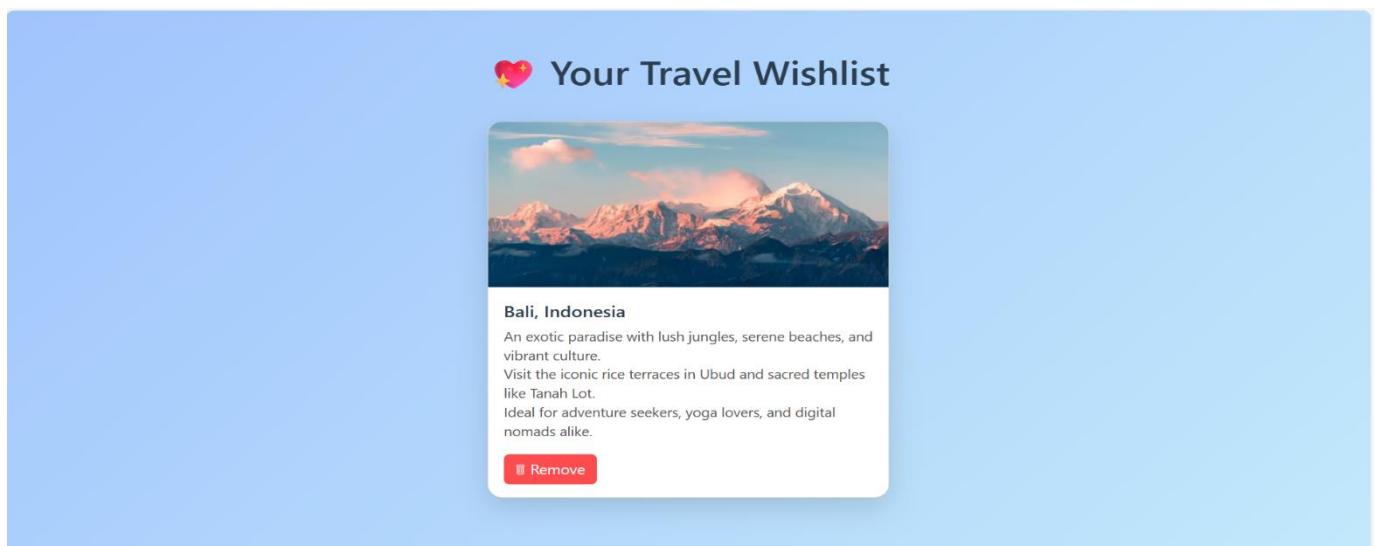
Login Page:



Dashboard page:



Wishlist page:



My Bookings:



Hello, Anjali! Ready to explore? 🌎



Bus



Train



Flight



Hotel

No bookings found.

Buses page:

TravelGo

Home Dashboard

Search & Book Buses

From: To: dd-mm-yyyy: No. of Persons: **Search**

AC Non-AC Sleeper Semi-Sleeper Seater

Sort by Price: None

TravelGo

Home Dashboard

Search & Book Buses

From: Bengaluru To: Chennai Date: 05-07-2025 No. of Persons: 2 **Search**

AC Non-AC Sleeper Semi-Sleeper Seater

Sort by Price: High to Low

SRS Travels
Non-AC Sleeper • 10:00 AM • ₹700/person **Book**



Confirm Your Bus Booking

Bus: SRS Travels
Route: Bengaluru → Chennai
Date: 2025-07-05 | **Time:** 10:00 AM
Class/Type: Non-AC Sleeper
Passengers: 2
Price per Person: ₹700.0

Total Price: ₹1400.0

Selected Seats: None

A1	A2	A3	A4
B1	B2	B3	B4
C1	C2	C3	C4
D1	D2	D3	D4

Confirm Booking

127.0.0.1 says

You can only select up to 2 seats.

OK

Bus: SRS Travels
Route: Bengaluru → Chennai
Date: 2025-07-05 | **Time:** 10:00 AM
Class/Type: Non-AC Sleeper
Passengers: 2
Price per Person: ₹700.0

Total Price: ₹1400.0

Selected Seats: None

A1	A2	A3	A4
B1	B2	B3	B4
C1	C2	C3	C4
D1	D2	D3	D4

Confirm Booking

Hello, Anjali! Ready to explore? 🌎



Bus



Train



Flight



Hotel

SRS Travels (Non-ac sleeper)

Route: Bengaluru → Chennai
Date: 2025-07-05 | **Time:** 10:00 AM
Class/Type:
Passengers: 2
Total Price: ₹1400.0
Seats: B1, B2

Cancel Booking



Trains page:

TravelGo

Dashboard

Search & Book Trains

From To dd - mm - yyyy No. of Passengers

Search

Express Superfast AC Sleeper

Sort by: None

TravelGo

Dashboard

Search & Book Trains

Bengaluru Hyderabad 05 - 07 - 2025 3

Search

Express Superfast AC Sleeper

Sort by: Price: High to Low

Sanghamitra Express (12296)

Express Sleeper • 16:00 - 04:00 • ₹650/person

Book

Confirm Your Train Booking

Train Name: Sanghamitra Express

Train No: 12296

Route: Bengaluru → Hyderabad

Departure: 16:00 on 2025-07-05

Arrival Time: 04:00

Class/Type: Express Sleeper

Price per Person: ₹650.0

Passenger: 3

Total Price: ₹1950.0

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

Selected Seats: None

Confirm Booking



Confirm Your Train Booking

Train Name: Sanghamitra Express

Train No: 12296

Route: Bengaluru → Hyderabad

Departure: 16:00 on 2025-07-05

Arrival Time: 04:00

Class/Type: Express Sleeper

Price per Person: ₹650.0

Passengers: 3

Total Price: ₹1950.0

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

Selected Seats: E1, D1, D2

[Confirm Booking](#)

TravelGo

[Wishlist](#)

[Bookings](#)

[Logout](#)

Hello, Anjali! Ready to explore? 🌎



Bus



Train



Flight



Hotel

Sanghamitra Express (Train)

Route: Bengaluru → Hyderabad

Date: 2025-07-05 | Time: 16:00

Class/Type:

Passengers: 3

Total Price: ₹1950.0

Seats: E1, D1, D2

[Cancel Booking](#)

Flights Page:

TravelGo

[Dashboard](#)

Search & Book Flights

Origin

Destination

dd - mm - yyyy

No. of Passengers

[Search](#)

Indigo

Vistara

Air India

Direct

1 Stop

Sort by: [None](#)

**Search & Book Flights**

Delhi (DEL)

Mumbai (BOM)

05-07-2025

5

Search

 Indigo Vistara Air India Direct 1 Stop

Sort by: Price: Low to High

Air India (AI 789)

Economy • 17:00 - 19:30 • Direct • ₹4500/person

Book

Confirm Your Flight Booking

Flight Name: Air India

Flight No: AI 789

Route: Delhi → Mumbai

Departure Time: 17:00 on 2025-07-05

Arrival Time: 19:30

Class: Economy

Price per Passenger: ₹4500.0

Passengers: 5

Select Your Seats:

1A	1B	1C	1D	1E	1F
2A	2B	2C	2D	2E	2F
3A	3B	3C	3D	3E	3F
4A	4B	4C	4D	4E	4F
5A	5B	5C	5D	5E	5F
6A	6B	6C	6D	6E	6F
7A	7B	7C	7D	7E	7F
8A	8B	8C	8D	8E	8F
9A	9B	9C	9D	9E	9F
10A	10B	10C	10D	10E	10F

Selected Seats: 5A, 5B, 6A, 6B, 7A

Confirm Booking

Hello, Anjali! Ready to explore? 🌎



Bus



Train



Flight



Hotel

Air India (Flight)**Route:** Delhi → Mumbai**Date:** 2025-07-05 | **Time:** N/A**Class/Type:****Passengers:** 5**Total Price:** ₹22500.0**Seats:** 5A, 5B, 6A, 6B, 7A

Cancel Booking



Hotels page:

TravelGo [Dashboard](#)

Find & Book Hotel Rooms

City/Area: dd-mm-yyyy dd-mm-yyyy No. of Rooms No. of Guests

Search

5-Star 4-Star 3-Star WiFi Pool Parking

Sort by: None

TravelGo [Dashboard](#)

Find & Book Hotel Rooms

mumbai 05 - 07 - 2025 07 - 07 - 2025 1 2

Search

5-Star 4-Star 3-Star WiFi Pool Parking

Sort by: Price: Low to High

ITC Grand Central
Mumbai • 5-Star • ₹12000/night
Amenities: WiFi, Pool, Parking, Spa

Confirm Your Hotel Booking

Hotel: ITC Grand Central
Location: Mumbai
Check-in: 2025-07-05
Check-out: 2025-07-07
Guests: 2
Rooms: 1
Room Type: Deluxe Room
Price per Night: ₹12000

Total Price: ₹12000

Confirm Booking



TravelGo

[Wishlist](#) [Bookings](#) [Logout](#)

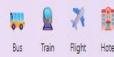
Hello, Anjali! Ready to explore? 🌎

**ITC Grand Central (Hotel)****Location:** Mumbai**Check-in:** 2025-07-05 | **Check-out:** 2025-07-07**Room Type:** Deluxe Room**Rooms:** 1 | **Nights:** 1**Total Price:** ₹12000[Cancel Booking](#)**Bookings page:**

TravelGo

[Wishlist](#) [Bookings](#) [Logout](#)

Hello, Anjali! Ready to explore? 🌎

**SRS Travels (Non-ac sleeper)**

Route: Bengaluru – Chennai

Date: 2025-07-05 | Time: 10:00 AM

Class/Type:

Passenger: 2

Total Price: ₹14000

Seats: 01, 02

[Cancel Booking](#)**Sanghamitra Express (Train)**

Route: Bengaluru – Hyderabad

Date: 2025-07-05 | Time: 16:00

Class/Type:

Passenger: 3

Total Price: ₹16500

Seats: 01, 02, 11

[Cancel Booking](#)**Air India (Flight)**

Route: Delhi – Mumbai

Date: 2025-07-05 | Time: N/A

Class/Type:

Passenger: 5

Total Price: ₹225000

Seats: SA, SB, GC, GB, TA

[Cancel Booking](#)**ITC Grand Central (Hotel)****Location:** Mumbai**Check-in:** 2025-07-05 | **Check-out:** 2025-07-07**Room Type:** Deluxe Room**Rooms:** 1 | **Nights:** 1**Total Price:** ₹12000[Cancel Booking](#)