

Name : GINJUPALLI Kiran

Email : 228x1a05e1@khitguntur.ac.in

College : [Kallam Haranadhareddy Institute of Technology](#)

Branch : CSE

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

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

Project Description:

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

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

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

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

Scenario 2: Real-Time Booking Confirmation with AWS SNS

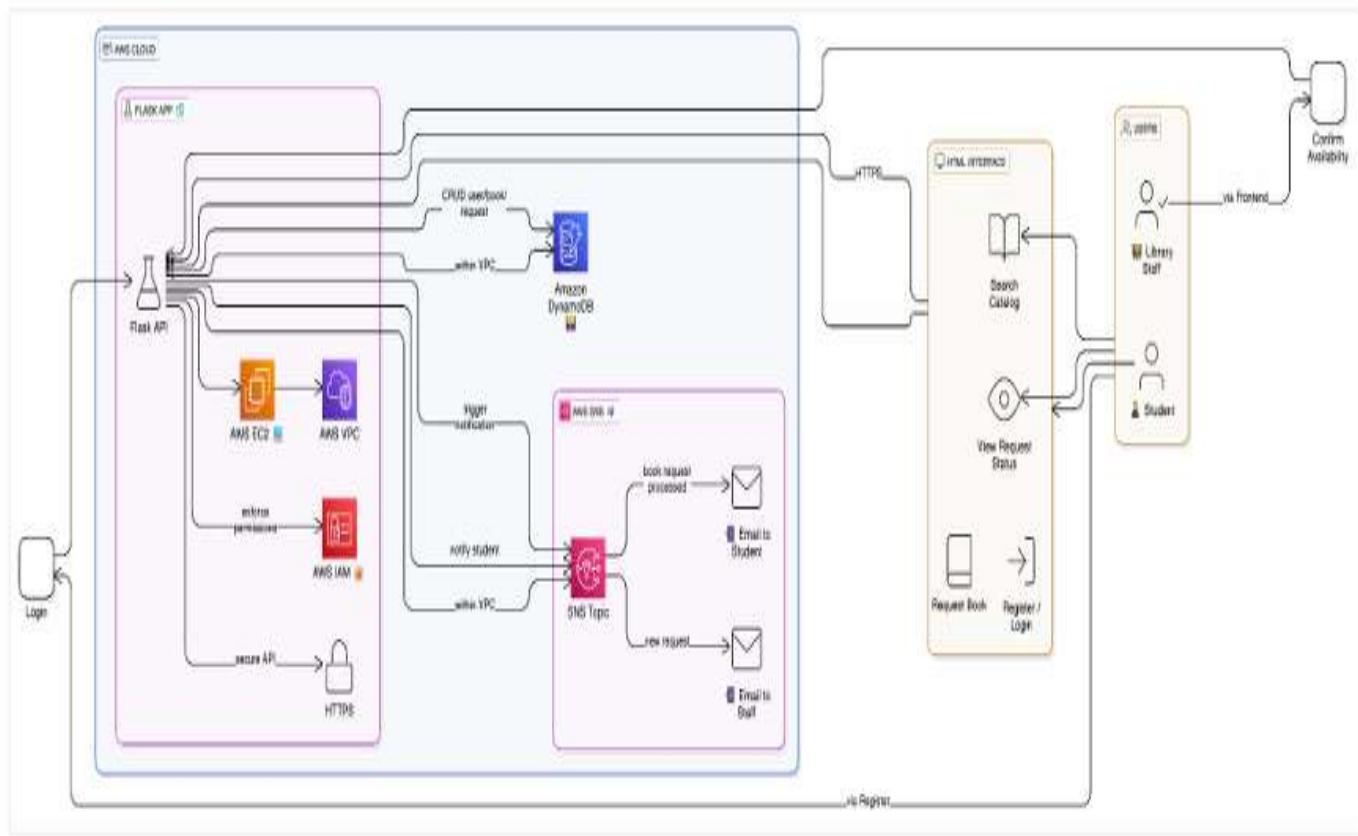
Once a booking is made—whether it's a train ticket or a hotel stay—TravelGo uses AWS SNS to instantly notify the user. For example, after a student books a hotel in Chennai, SNS sends a real-time email notification confirming the booking with all the relevant details. This notification is triggered from the

Flask backend after the booking is successfully recorded in DynamoDB. Additionally, SNS can alert admin or service providers, ensuring transparency and real-time updates on every transaction.

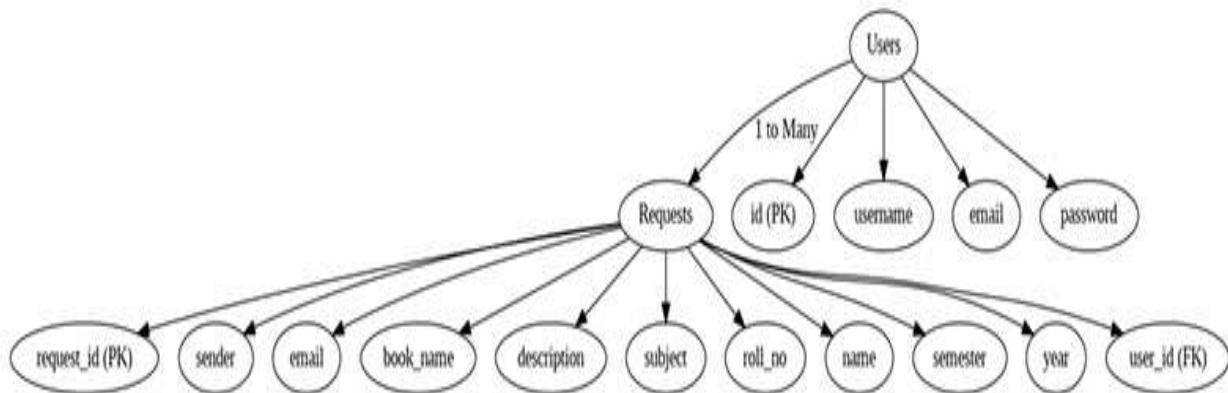
Scenario 3: Dynamic Dashboard with Personal Travel History

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

AWS ARCHITECTURE



Entity Relationship (ER) Diagram:



Project WorkFlow:

1. AWS troven access Setup and Login

1.1: Set up an troven access

1.2: Open the AWS Management Console

2. DynamoDB Database Creation and Setup

2.1: Create a DynamoDB Table.

2.2: Configure Attributes for User Data and Booking Requests.

3. SNS Notification Setup Activity

3.1: Create SNS topics.

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

4. Backend Development and Application Setup

4.1: Develop the Backend Using Flask.

4.2: Integrate AWS Services Using boto3.

5. IAM Role Setup

5.1: Create IAM Role

5.2: Attach Policies

6. EC2 Instance Setup Activity

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

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, bookings, and notifications.

1. AWS troven access Setup and Login

- Activity 1.1: Set up an troven access

- Activity 1.2: Open the AWS Management Console

AWS Final Deployment

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

Total Task: 5 Recommended Duration: 03:00:00 HRS Current Lab Duration: 02:12:31 HRS Validation Status: 0/5

The region should be set to N Virginia.

Initiate Labs

Open Console End Lab

Once you complete a task you can click on Validate and check if the task is done correctly. Click Validate All to check the status all at once.

Lab Guide Overviews

Task Number	Description	Action
1	Launch an EC2 Instance (TravelGo)	AWS Validate
2	Create IAM Role (TravelGo)	AWS Validate
3	Setup DynamoDB (TravelGo)	AWS Validate
4	Configure SNS (TravelGo)	AWS Validate

30°C Cloudy GINJUPALLI 226x105sel@khilgurur.ac.in

Search web &... easy 5 180 mins Difficulty Task Duration

ENG IN 01-07-2025 17:04

DynamoDB Database Creation and Setup

Activity 2.1: Create a DynamoDB Table

To create a DynamoDB table, go to the AWS Management Console, navigate to **DynamoDB**, and click

The screenshot shows the 'Create table' wizard in the AWS Management Console. In the 'Table details' section, the table name is set to 'bookings1'. The 'Partition key' is defined as 'user_email' of type 'String'. A secondary 'Sort key - optional' is defined as 'Booking_id' of type 'String'. In the 'Table settings' section, the 'Default settings' option is selected, indicating the fastest way to create the table. The 'Editable after creation' link is visible. The bottom of the screen shows the standard AWS navigation bar with CloudShell, Feedback, and other links.

"Create table." Specify a **table name** and define a **primary key** (e.g., Email as Partition Key). Choose additional settings like **read/write capacity mode** (on-demand or provisioned) and click **"Create."** Your table will be ready in seconds for data operations.

Activity 2.2: Configure Attributes for User Data and Booking Requests

3. SNS

Notification Setup

For **SNS Notification Setup**, go to the AWS Console and open the **Simple Notification Service (SNS)** dashboard. Click **"Create topic"**, choose the **Standard** type, and name your topic. After creating it, copy the **Topic ARN**.

The screenshot shows the AWS SNS 'Topics' dashboard. A new topic named 'TravelGo' has been created. The 'Details' section shows the Name as 'TravelGo', ARN as 'arn:aws:sns:us-east-1:418272775181:TravelGo', and Type as 'Standard'. The 'Subscriptions' tab is active, showing one subscription with ID 'b7d6f5bc-7710-49de-97bc-ddecff5fd023', Endpoint '228x1a05e1@khitguntur.ac.in', and Status 'Confirmed'. Other tabs include Access policy, Data protection policy, Delivery policy (HTTP/S), Delivery status logging, Encryption, Tags, and Integrations. The bottom of the screen shows the standard AWS navigation bar with CloudShell, Feedback, and other links.

Then, **subscribe email addresses** (students/admins) to the topic and confirm the subscription via email.

Activity 3.1: Create SNS topics.

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

The screenshot shows the 'Create subscription' page in the AWS SNS console. The top navigation bar includes the AWS logo, search bar, and account information: United States (N. Virginia) and rsoaccount-new/6801da4369d20120be221457 @ rsosandboxnew68. The main content area is titled 'Create subscription' and has a 'Details' section. In the 'Topic ARN' field, the value 'arn:aws:sns:us-east-1:418272775181:TravelGo' is entered. The 'Protocol' dropdown is set to 'Email'. The 'Endpoint' field contains the email address '228x1a05e1@khitguntur.ac.in'. A note below the endpoint says, 'After your subscription is created, you must confirm it.' A 'Subscription filter policy - optional' section is present, followed by a 'Redrive policy (dead-letter queue) - optional' section. At the bottom right are 'Cancel' and 'Create subscription' buttons.

Backend Development and Application Setup

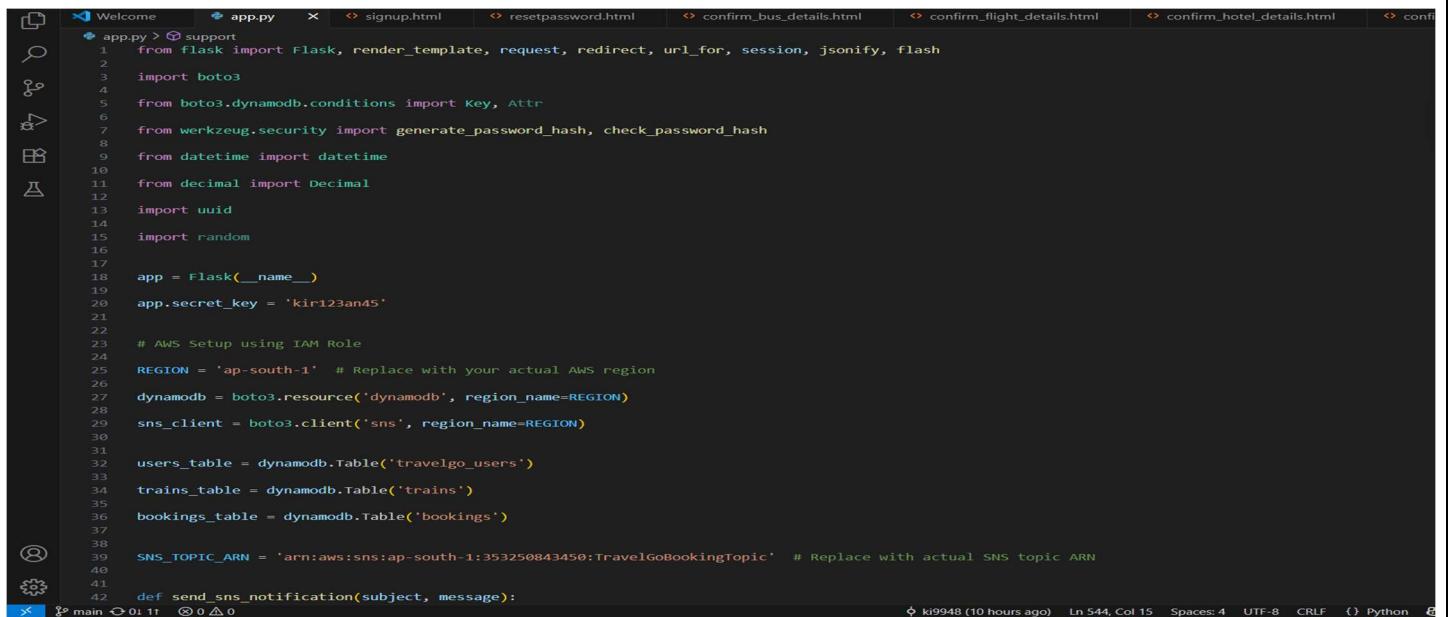
The screenshot shows a confirmation message from the Simple Notification Service. It features the AWS logo and the text 'Simple Notification Service'. The main message is 'Subscription confirmed! You have successfully subscribed. Your subscription's id is: arn:aws:sns:us-east-1:241533142623:TravelGo1:11ec3460-05a0-45b3-815e-44f651059b28. If it was not your intention to subscribe, [click here to unsubscribe](#).'. The entire message is enclosed in a light green rounded rectangle.

4.

In
the

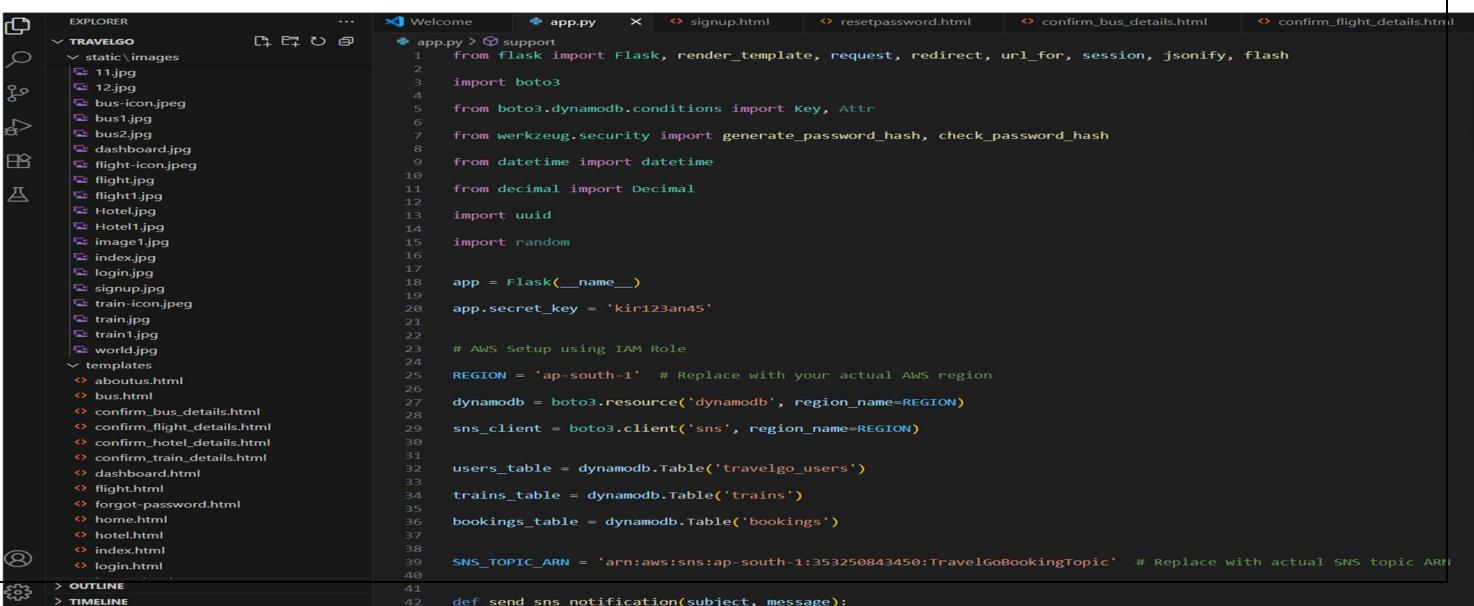
Backend Development and Application Setup, use Flask to build the web application and integrate AWS services via the boto3 library. Set up routes for user registration, login, book requests, and page navigation. Connect to DynamoDB to store user and request data, and configure AWS SNS to send notifications. Organize files into app.py, templates/, and static/ folders for a clean project structure

Activity 4.1: Develop the Backend Using Flask.



```
1  #!/usr/bin/env python
2
3  from flask import Flask, render_template, request, redirect, url_for, session, jsonify, flash
4
5  import boto3
6
7  from boto3.dynamodb.conditions import Key, Attr
8
9  from werkzeug.security import generate_password_hash, check_password_hash
10 from datetime import datetime
11 from decimal import Decimal
12 import uuid
13 import random
14
15 app = Flask(__name__)
16 app.secret_key = 'kir123an45'
17
18 # AWS Setup using IAM Role
19
20 REGION = 'ap-south-1' # Replace with your actual AWS region
21
22 dynamodb = boto3.resource('dynamodb', region_name=REGION)
23
24 sns_client = boto3.client('sns', region_name=REGION)
25
26
27 users_table = dynamodb.Table('travelgo_users')
28
29 trains_table = dynamodb.Table('trains')
30
31 bookings_table = dynamodb.Table('bookings')
32
33
34 SNS_TOPIC_ARN = 'arn:aws:sns:ap-south-1:353250843450:TravelGoBookingTopic' # Replace with actual SNS topic ARN
35
36
37
38
39 def send sns_notification(subject, message):
40
41
42
```

Activity 4.2: Integrate AWS Services Using boto3



```
1  #!/usr/bin/env python
2
3  from flask import Flask, render_template, request, redirect, url_for, session, jsonify, flash
4
5  import boto3
6
7  from boto3.dynamodb.conditions import Key, Attr
8
9  from werkzeug.security import generate_password_hash, check_password_hash
10 from datetime import datetime
11 from decimal import Decimal
12 import uuid
13 import random
14
15 app = Flask(__name__)
16 app.secret_key = 'kir123an45'
17
18 # AWS Setup using IAM Role
19
20 REGION = 'ap-south-1' # Replace with your actual AWS region
21
22 dynamodb = boto3.resource('dynamodb', region_name=REGION)
23
24 sns_client = boto3.client('sns', region_name=REGION)
25
26
27 users_table = dynamodb.Table('travelgo_users')
28
29 trains_table = dynamodb.Table('trains')
30
31 bookings_table = dynamodb.Table('bookings')
32
33
34 SNS_TOPIC_ARN = 'arn:aws:sns:ap-south-1:353250843450:TravelGoBookingTopic' # Replace with actual SNS topic ARN
35
36
37
38
39 def send sns_notification(subject, message):
40
41
42
```

```
* app.py > ...
1  from flask import Flask, render_template, request, redirect, url_for, session, jsonify, flash
2  import boto3
3  from boto3.dynamodb.conditions import Key, Attr
4  from werkzeug.security import generate_password_hash, check_password_hash
5  from datetime import datetime
6  from decimal import Decimal
7  import uuid
8  import random
9
10 app = Flask(__name__)
11 app.secret_key = 'kir123an45' # IMPORTANT: Change this to a strong, random key in production!
12
13 # AWS Setup using IAM Role
14 REGION = 'us-east-1' # Replace with your actual AWS region
15 dynamodb = boto3.resource('dynamodb', region_name=REGION)
16 sns_client = boto3.client('sns', region_name=REGION)
17
18 users_table = dynamodb.Table('travelgo_users')
19 trains_table = dynamodb.Table('trains') # Note: This table is declared but not used in the provided routes.
20 bookings_table = dynamodb.Table('bookings')
21
22 SNS_TOPIC_ARN = 'arn:aws:sns:us-east-1:418272775181:TravelGo:b7d6f5bc-7710-49de-97bc-ddecff5fd023' # Replace with actual SNS topic
23
```

Cod

Code for SNS

```
3
4  # Function to send SNS notifications
5  # This function is duplicated in the original code, removing the duplicate.
6  def send_sns_notification(subject, message):
7      try:
8          sns_client.publish(
9              TopicArn=SNS_TOPIC_ARN,
10             Subject=subject,
11             Message=message
12         )
13     except Exception as e:
14         print(f"SNS Error: Could not send notification - {e}")
15         # Optionally, flash an error message to the user or log it more robustly.
16
```

CODE for Register

```
"__routes__
@app.route('/')
def index():
    return render_template('index.html')

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

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

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

@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'))
```

Code for Dashboard

```
36
37 @app.route('/dashboard')
38 def dashboard():
39     if 'email' not in session:
40         return redirect(url_for('login'))
41     user_email = session['email']
42
43     # Query bookings for the logged-in user using the primary key 'user_email'
44     # No GSI is needed here as 'user_email' is likely the partition key for the bookings_table.
45     response = bookings_table.query(
46         KeyConditionExpression=Key('user_email').eq(user_email),
47         ScanIndexForward=False # Get most recent bookings first
48     )
49     bookings = response.get('Items', [])
50
51     # Convert Decimal types from DynamoDB to float for display if necessary
52     for booking in bookings:
53         if 'total_price' in booking:
54             try:
55                 booking['total_price'] = float(booking['total_price'])
56             except (TypeError, ValueError):
57                 booking['total_price'] = 0.0 # Default value if conversion fails
58
59     return render_template('dashboard.html', username=user_email, bookings=bookings)
```

Code for Cancel booking

```

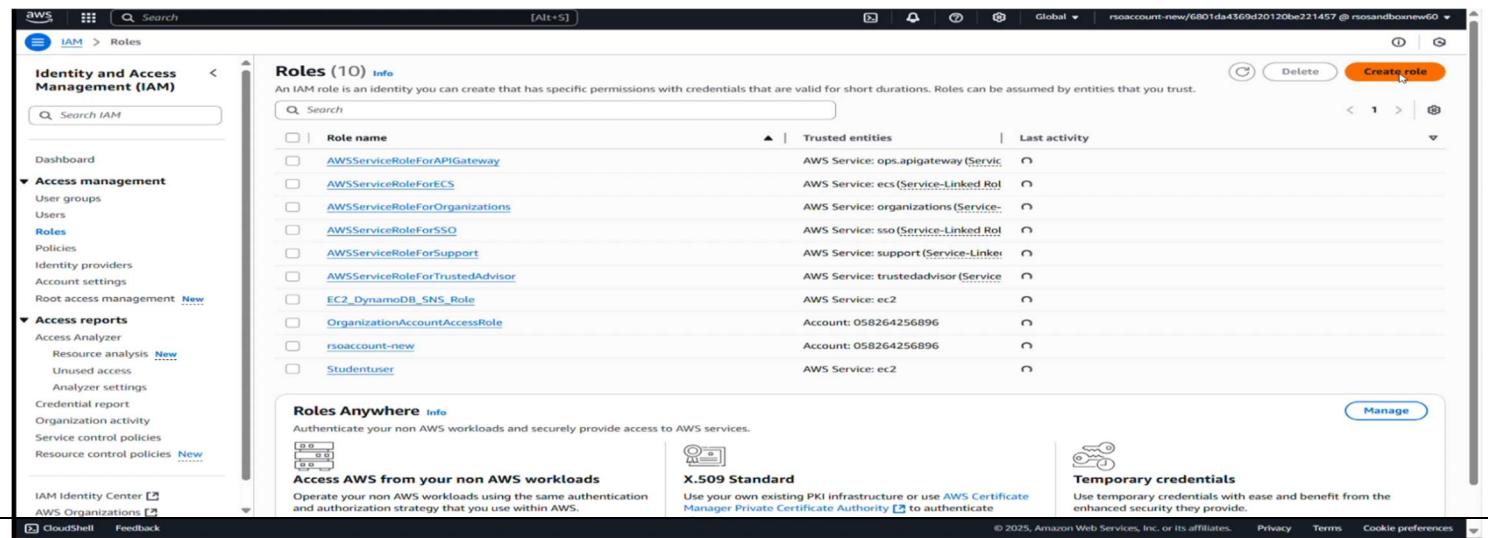
1  @app.route('/cancel_booking', methods=['POST'])
2  def cancel_booking():
3      if 'email' not in session:
4          return redirect(url_for('login'))
5
6      booking_id = request.form.get('booking_id')
7      user_email = session['email']
8      booking_date = request.form.get('booking_date') # This is crucial as it's the sort key
9
10     if not booking_id or not booking_date:
11         flash("Error: Booking ID or Booking Date is missing for cancellation.", 'error')
12         return redirect(url_for('dashboard'))
13
14     try:
15         # Delete item using the primary key (user_email and booking_date)
16         # This does not use GSI, so it remains unchanged.
17         bookings_table.delete_item(
18             Key={'user_email': user_email, 'booking_date': booking_date}
19         )
20         flash(f"Booking {booking_id} cancelled successfully!", 'success')
21     except Exception as e:
22         flash(f"Failed to cancel booking {booking_id}: {str(e)}", 'error')
23
24     return redirect(url_for('dashboard'))
25
26
27 if __name__ == '__main__':
28     # IMPORTANT: In a production environment, disable debug mode and specify a production-ready host.
29     app.run(debug=True, host='0.0.0.0')
30
31

```

Setup

For IAM Role Setup, go to the AWS Console and create a new IAM Role for your EC2 instance. Attach AmazonDynamoDBFullAccess and AmazonSNSFullAccess policies to allow interaction with DynamoDB and SNS. Assign this role to your EC2 instance during or after launch. This ensures secure and authorized access to AWS resources from your Flask backend.

Activity 5.1: Create IAM Role



Activity 5.2: Attach Policies

The screenshot shows the 'Add permissions' step of the IAM role creation wizard. The left sidebar indicates 'Step 2: Add permissions' is selected. The main area displays a search bar for 'Permissions policies' containing 'sns'. A table lists five AWS managed policies: 'AmazonSNSFullAccess' (selected), 'AmazonSNSReadOnlyAccess', 'AmazonSNSRole', 'AWSelasticBeanstalkRoleSNS', and 'AWSIoTDeviceDefenderPublishFindingsToSNSSignatureVerification'. The table includes columns for 'Policy name', 'Type', and 'Description'. Buttons for 'Cancel', 'Previous', and 'Next' are at the bottom right.

6. EC2 Instance Setup Activity

In the **EC2 Instance Setup**, launch a new EC2 instance using **Amazon Linux 2 or Ubuntu**, and choose the **t2.micro** type for free-tier eligibility. Configure a **key pair** for secure SSH access and set **security groups** to allow **HTTP** (port 80) and **SSH** (port 22). Attach the appropriate **IAM role** to the instance. This setup prepares the server environment to host your Flask application.

The screenshot shows the 'Step 3: Add tags' section of the IAM role creation wizard. It includes a code editor with a JSON policy document, a 'Permissions policy summary' table, and a 'Add tags - optional' section. The table lists three attached policies: 'AmazonDynamoDBFullAccess', 'AmazonEC2FullAccess', and 'AmazonSNSFullAccess'. The 'Add tags' section shows no existing tags and a button to 'Add new tag'. Navigation buttons for 'Cancel', 'Previous', and 'Create role' are at the bottom right.

The screenshot shows the 'Launch an instance' wizard in the AWS Management Console. The current step is 'Name and tags'. A blue box highlights the 'Name' field, which contains 'TravelGo1'. To the right, a summary panel shows 'Number of instances' set to 1. Below the summary, a note about the free tier is visible, and a large orange 'Launch instance' button is at the bottom right.

create a key pair:

The screenshot shows the 'Launch an instance' wizard with the 'Create key pair' dialog open. The 'Key pair name' field is filled with 'travel119'. The 'Key pair type' section has 'RSA' selected. The 'Private key file format' section has '.pem' selected. A note at the bottom of the dialog says: '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.' An orange 'Create key pair' button is at the bottom right of the dialog. The background shows the 'Summary' step of the wizard.

Edit of Security group:

The screenshot shows the AWS EC2 console with the URL [https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#SecurityGroups:sg-0a6c8b20300891292](#). The page title is "Edit inbound rules". The main content area displays three security group rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-088a2cf6fd085c825	SSH	TCP	22	Custom	0.0.0.0/0
sgr-0c7741e22e1e12161	HTTPS	TCP	443	Custom	0.0.0.0/0
-	Custom TCP	TCP	5000	Anywhere	0.0.0.0/0

Below the table is a blue "Add rule" button. A yellow warning message at the bottom left states: "⚠ 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." To the right of the message are "Cancel", "Preview changes", and a highlighted "Save rules" button.

At the bottom of the page, there are links for CloudShell, Feedback, and legal notices: © 2025, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

Screenshot of the AWS EC2 Instances page showing three running t2.micro instances. The context menu for the first instance (travelgo99) is open, displaying options like 'Connect', 'Instance state', 'Actions', and 'Launch instances'.

Instances (1/3) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Avg CPU Utilization
travelgo99	i-0ecb2302d26567b67	Running	t2.micro	Initializing	View alarms +	us-east-1
TravelGo	i-0e5af7d5c05f2398b	Running	t2.micro	2/2 checks	Get Windows password	us-east-1
TravelGo1	i-04eaa784c914b3b3	Running	t2.micro	2/2 checks	Modify IAM role	us-east-1

i-0ecb2302d26567b67 (travelgo99)

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

Security details

IAM Role: -

Owner ID: 241533142623

Launch time: Tue Jul 01 2025 12:24:03 GMT+0530 (India Standard Time)

Security groups: sg-0a6c8b20300891292 (launch-wizard-3)

Inbound rules:

Name	Security group rule ID	Port range	Protocol	Source	Security groups	Description
-	sgr-088a2cf6fd085c825	22	TCP	0.0.0.0/0	launch-wizard-3	-
-	sgr-0c7741e22e1e12161	443	TCP	0.0.0.0/0	launch-wizard-3	-

Outbound rules:

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

EC2 Connect:

Screenshot of the AWS EC2 Connect page for instance i-0ecb2302d26567b67 (travelgo99).

Connect Info

Connect to an instance using the browser-based client.

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

Instance ID: i-0ecb2302d26567b67 (travelgo99)

- Open an SSH client.
- Locate your private key file. The key used to launch this instance is travel119.pem.
- Run this command, if necessary, to ensure your key is not publicly viewable:
 chmod 400 "travel119.pem"
- Connect to your instance using its Public DNS:
 ec2-3-86-105-106.compute-1.amazonaws.com

Example:
 ssh -i "travel119.pem" ec2-user@ec2-3-86-105-106.compute-1.amazonaws.com

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

Cancel

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

Milestone 7: Deployment on EC2

Activity 7.1: Install Software on the EC2 Instance

Install Python3, Flask, and Git: On Amazon Linux 2:

```
sudo yum update -y
```

```
sudo yum install python3
```

```
git sudo pip3 install flask boto3
```

Verify Installations:

```
flask --version git --version
```

Activity 7.2:Clone Your Flask Project from GitHub

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

Run: ‘git clone <https://github.com/your-github-username/your-repository-name.git>’

Note: change your-github-username and your-repository-name with your credentials here: ‘git clone
<https://github.com/AlekhyaPenubakula/InstantLibrary.git>’

- This will download your project to the EC2 instance.

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

```
cd InstantLibrary
```

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

Run the Flask Application

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

```

install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\91781> ssh -i "C:\Users\91781\Downloads\kiran-travelgo.pem" ec2-user@ec2-54-205-203-16.compute-1.amazonaws.com
The authenticity of host 'ec2-54-205-203-16.compute-1.amazonaws.com (54.205.203.16)' can't be established.
ED25519 key fingerprint is SHA256:Bz+Y3Le7rmgyn5pJ+UjGxYE/AuELQuFJ3MTkofrDMeQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-205-203-16.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

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

[ec2-user@ip-172-31-88-122 ~]$ sudo yum install git -y
Amazon Linux 2023 Kernel Livepatch repository                               175 kB/s | 17 kB    00:00
Dependencies resolved.
=====
Package          Architecture      Version       Repository      Size
=====
Installing:
git              x86_64          2.47.1-1.amzn2023.0.3      amazonlinux   52 k
Installing dependencies:
git-core          x86_64          2.47.1-1.amzn2023.0.3      amazonlinux   4.5 M
git-core-doc      noarch          2.47.1-1.amzn2023.0.3      amazonlinux   2.8 M
perl-Error        noarch          1:0.17029-5.amzn2023.0.2      amazonlinux   41 k
perl-File-Find    noarch          1.37-477.amzn2023.0.7      amazonlinux   25 k
perl-Git          noarch          2.47.1-1.amzn2023.0.3      amazonlinux   40 k
perl-TermReadKey x86_64          2.38-9.amzn2023.0.2       amazonlinux   36 k
perl-lib          x86_64          0.65-477.amzn2023.0.7      amazonlinux   15 k

Transaction Summary
=====
Install 8 Packages

Total download size: 7.5 M
Installed size: 37 M
Downloading Packages:
(1/8): git-2.47.1-1.amzn2023.0.3.x86_64.rpm           1.4 MB/s | 52 kB    00:00
(2/8): perl-Error-0.17029-5.amzn2023.0.2.noarch.rpm     1.6 MB/s | 41 kB    00:00

```

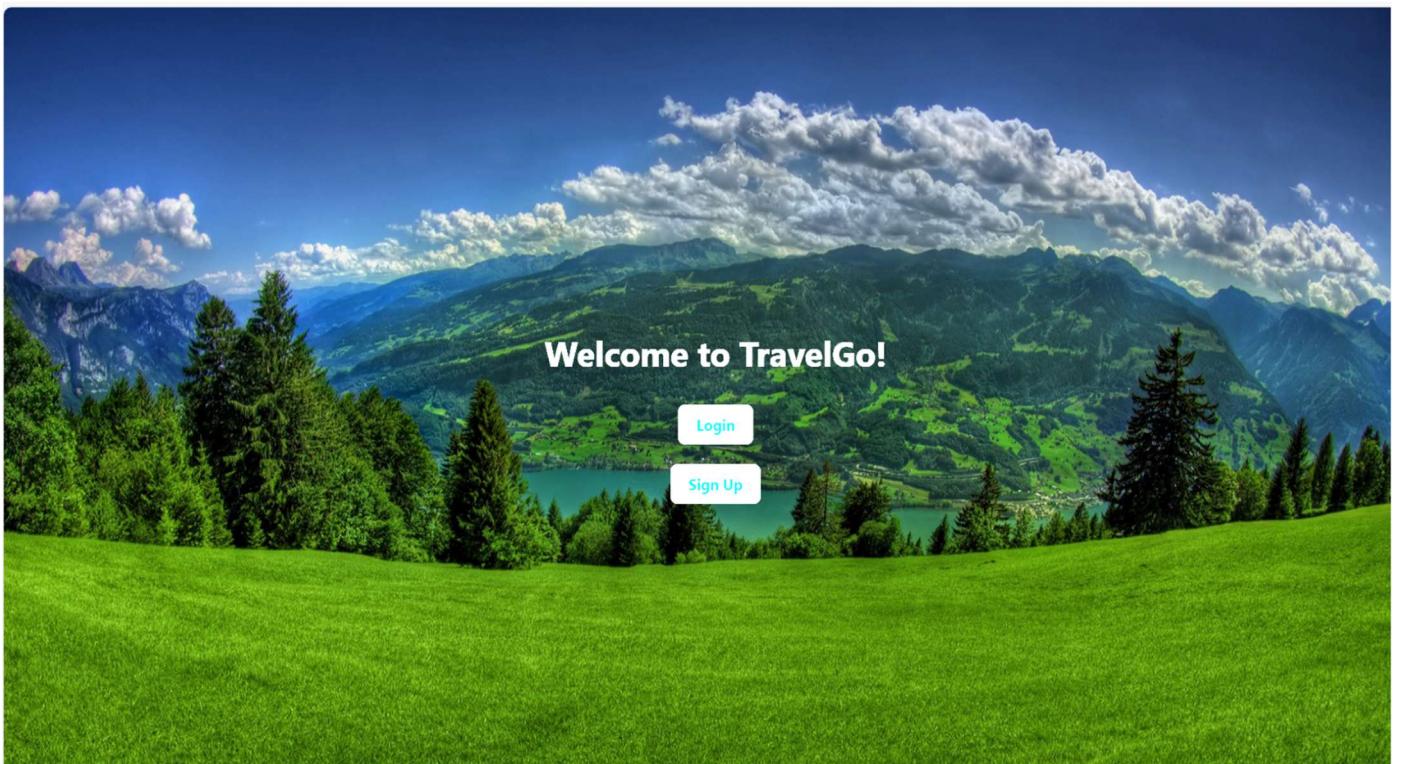
GitHub link:

The screenshot shows a GitHub repository page for the user 'ki9948'. The repository name is 'testupload'. The page includes a file tree on the left showing files like 'app.py', 'README.md', and various directory structures ('static', 'templates', 'venv', 'venv_new'). On the right, there's a 'Code' dropdown menu open, showing options for cloning the repository via HTTPS, SSH, or GitHub CLI. The HTTPS URL is highlighted. Below the clone options are links to 'Open with GitHub Desktop' and 'Download ZIP'. To the right of the code menu, there's an 'About' section with a note: 'No description, website, or topics provided.' It also lists repository statistics: 0 stars, 0 forks, 0 watching, and 0 commits. There are sections for 'Releases' (no releases published) and 'Packages'.

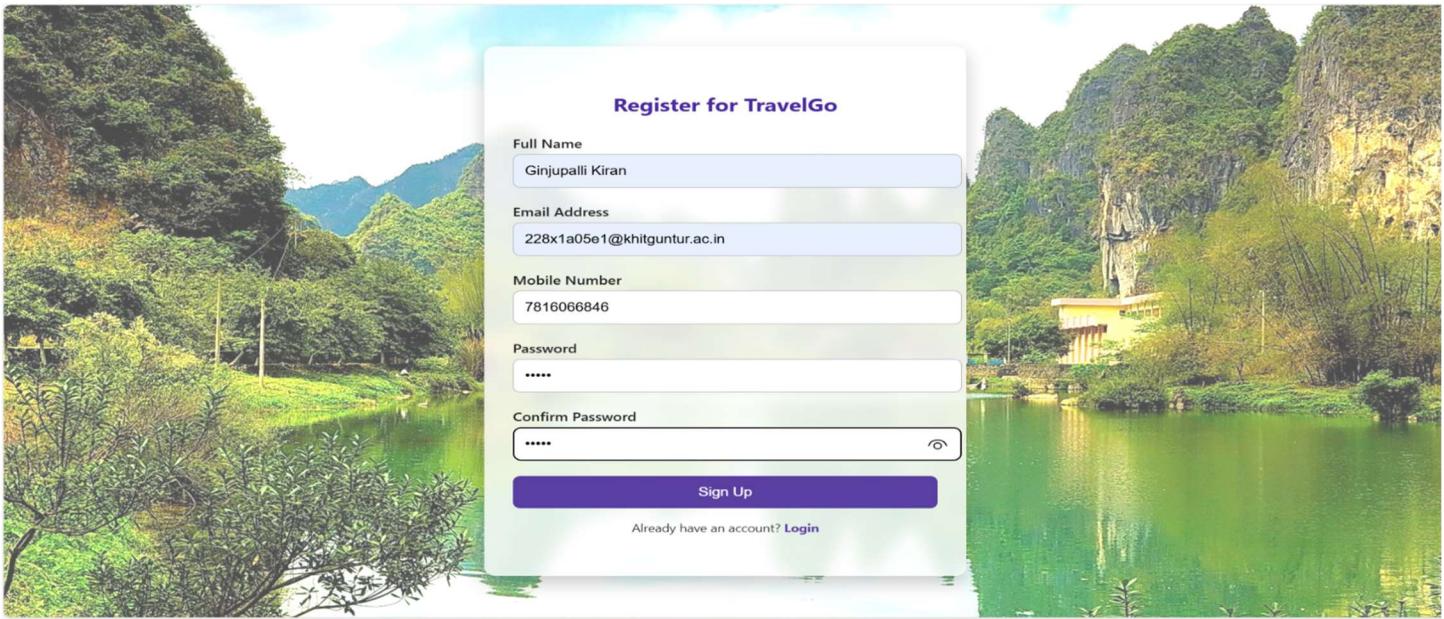
```

[ec2-user@ip-172-31-88-122 ~]$ pip install itsdangerous Werkzeug Flask Jinja2 MarkupSafe Click
Collecting itsdangerous>=2.2.0
  Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Collecting blinker>=1.9.0
  Downloading blinker-1.9.0-py3-none-any.whl (8.5 kB)
Collecting werkzeug>=3.1.0
  Downloading werkzeug-3.1.3-py3-none-any.whl (224 kB)
|██████████| 224 kB 56.0 MB/s
Collecting zipp>=3.20
  Downloading zipp-3.23.0-py3-none-any.whl (10 kB)
Installing collected packages: zipp, markupsafe, werkzeug, jinja2, itsdangerous, importlib-metadata, click, blinker, flask
Successfully installed blinker-1.9.0 click-8.1.8 flask-3.1.1 importlib-metadata-8.7.0 itsdangerous-2.2.0 jinja2-3.1.6 markupsafe-3.0.2 werkzeug-3.1.3 zipp-3
.23.0
[ec2-user@ip-172-31-88-122 ~]$ pip install boto3
Defaulting to user installation because normal site-packages is not writeable
Collecting boto3
  Downloading boto3-1.39.0-py3-none-any.whl (139 kB)
|██████████| 139 kB 13.8 MB/s
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3.9/site-packages (from boto3) (0.10.0)
Collecting botocore<1.40.0,>=1.39.0
  Downloading botocore-1.39.0-py3-none-any.whl (13.8 MB)
|██████████| 13.8 MB 46.6 MB/s
Collecting s3transfer<0.14.0,>=0.13.0
  Downloading s3transfer-0.13.0-py3-none-any.whl (85 kB)
|██████████| 85 kB 6.3 MB/s
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.0->boto3) (1.25.10)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.0->boto3) (2.8.1)
Requirement already satisfied: six>=1.5 in /usr/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.40.0,>=1.39.0->boto3) (1.15.0)
Installing collected packages: botocore, s3transfer, boto3
Successfully installed boto3-1.39.0 botocore-1.39.0 s3transfer-0.13.0
[ec2-user@ip-172-31-88-122 ~]$ cd TravelGok
[ec2-user@ip-172-31-88-122 TravelGok]$ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.31.88.122:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 696-443-691
[ec2-user@ip-172-31-88-122 TravelGok]$
```

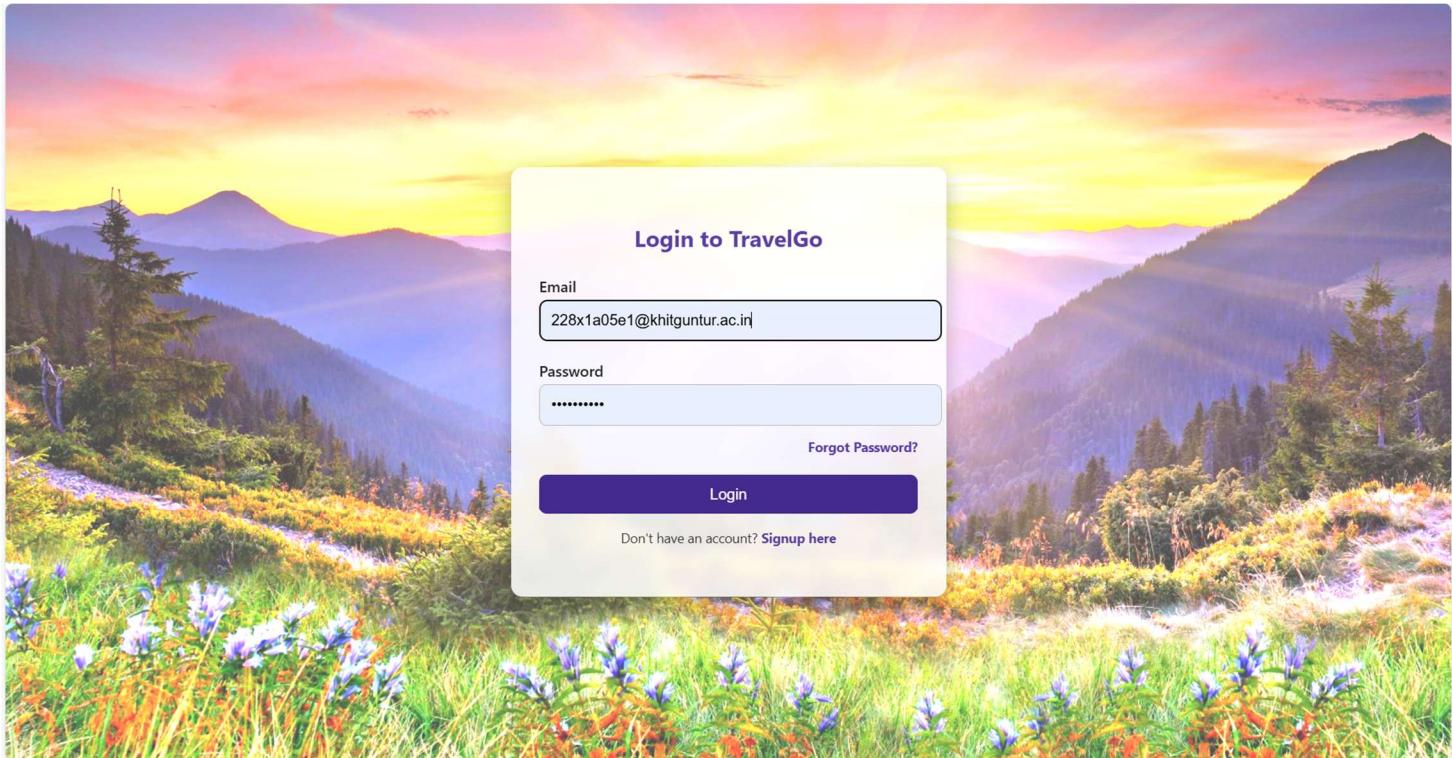
Index page:



Register page:



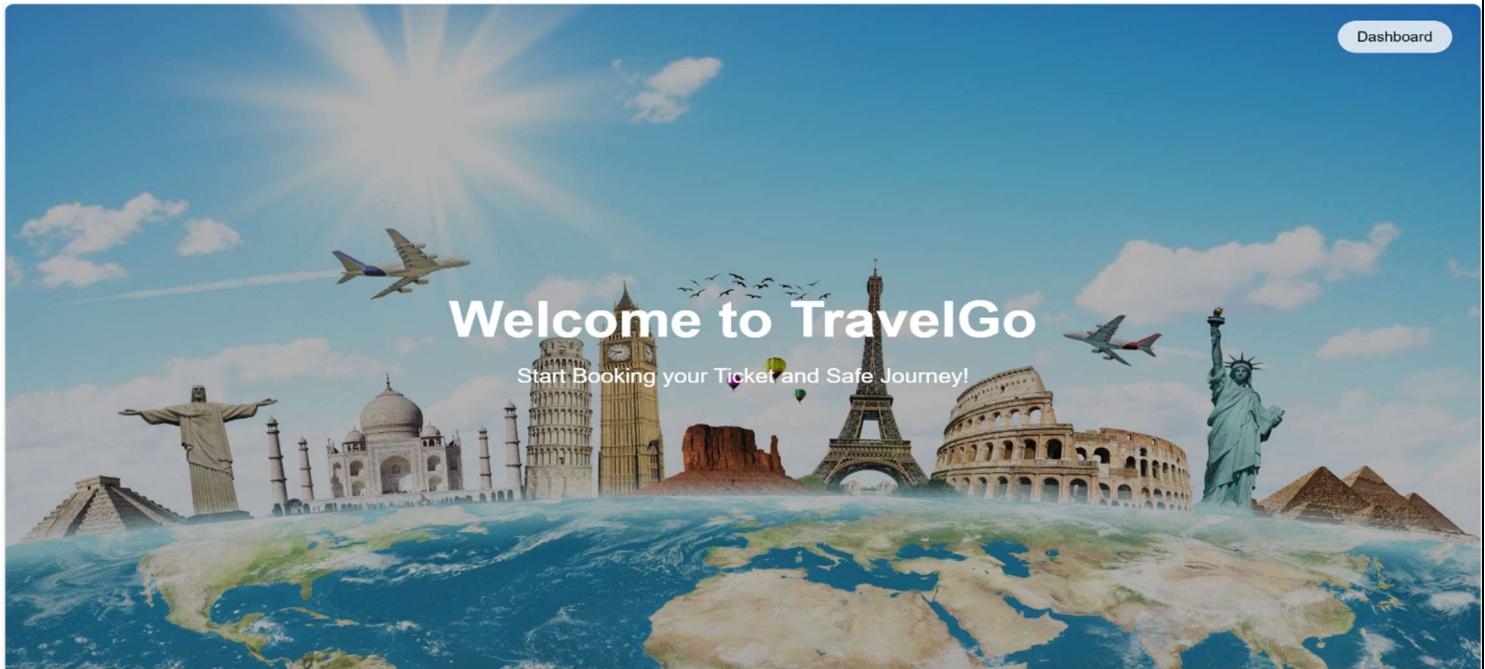
Login page:



Home Code:

```
  Welcome   app.py M  home.html X
templates > home.html > html > head > style
  2  <html lang="en">
  3  <head>
  4    <style>
49
50      .dashboard-btn:hover {
51        background-color: #f0c040;
52        color: #000;
53      }
54
55      .content {
56        height: 100%;
57        display: flex;
58        justify-content: center;
59        align-items: center;
60        text-align: center;
61        color: white;
62        padding: 20px;
63      }
64
65      .content h1 {
66        font-size: 60px;
67        margin-bottom: 20px;
68      }
69
70      .content p {
71        font-size: 24px;
72      }
73    </style>
74  </head>
75  <body>
76    <div class="overlay">
77      <a href="/dashboard" class="dashboard-btn">Dashboard</a>
78      <div class="content">
79        <div>
80          <h1>Welcome to TravelGo</h1>
81          <p>Start Booking your Ticket and Safe Journey!</p>
82        </div>
83      </div>
84    </div>
85  </body>
86 </html>
```

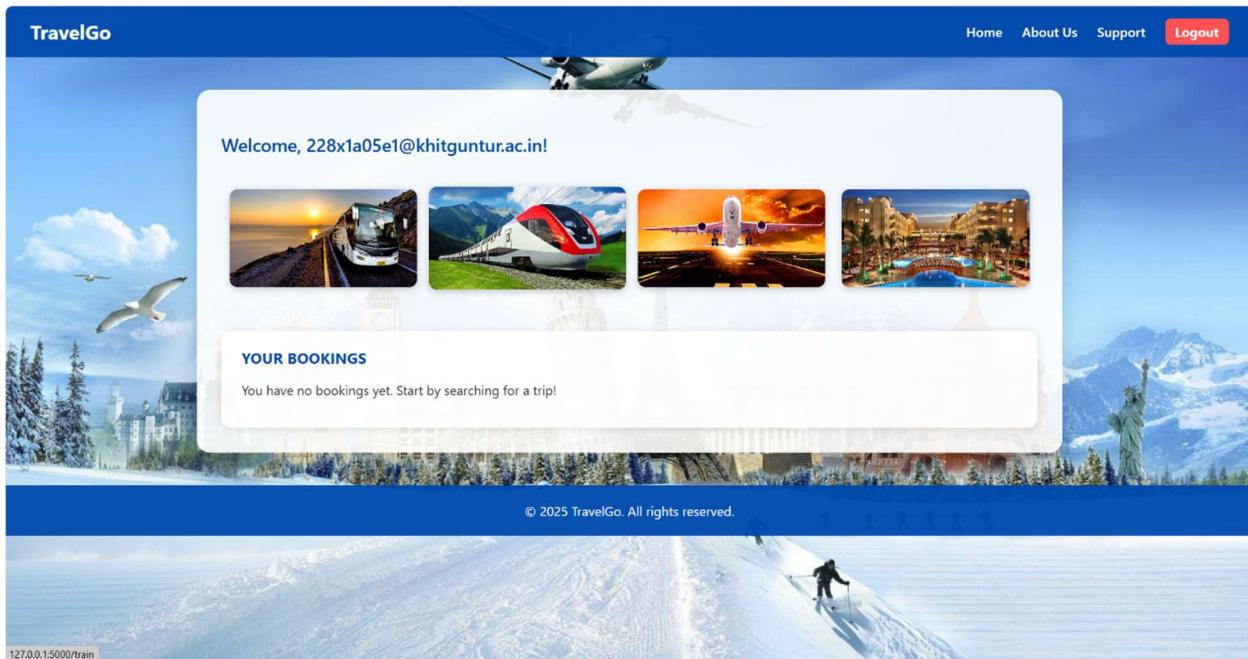
Home page:



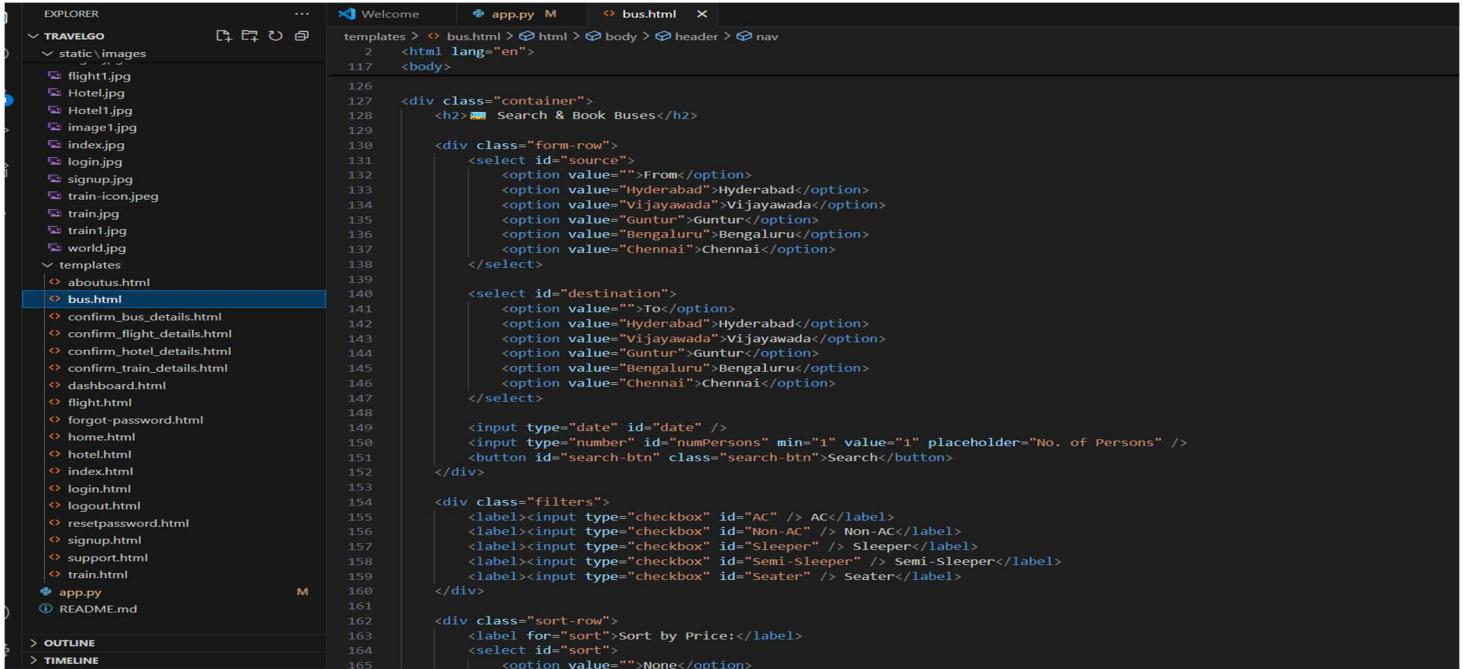
Dashboard code:

```
 1 Welcome   app.py M dashboard.html X
 2 templates > dashboard.html > html > body > div.dashboard-container > div.bookings > div.booking-item > div.booking-actions > form > button.cancel-l
 3
 4
 5
 6
 7
 8
 9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 28
 29
 30
 31
 32
 33
 34
 35
 36
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49
 50
 51
 52
 53
 54
 55
 56
 57
 58
 59
 60
 61
 62
 63
 64
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
```

Dashboard page:



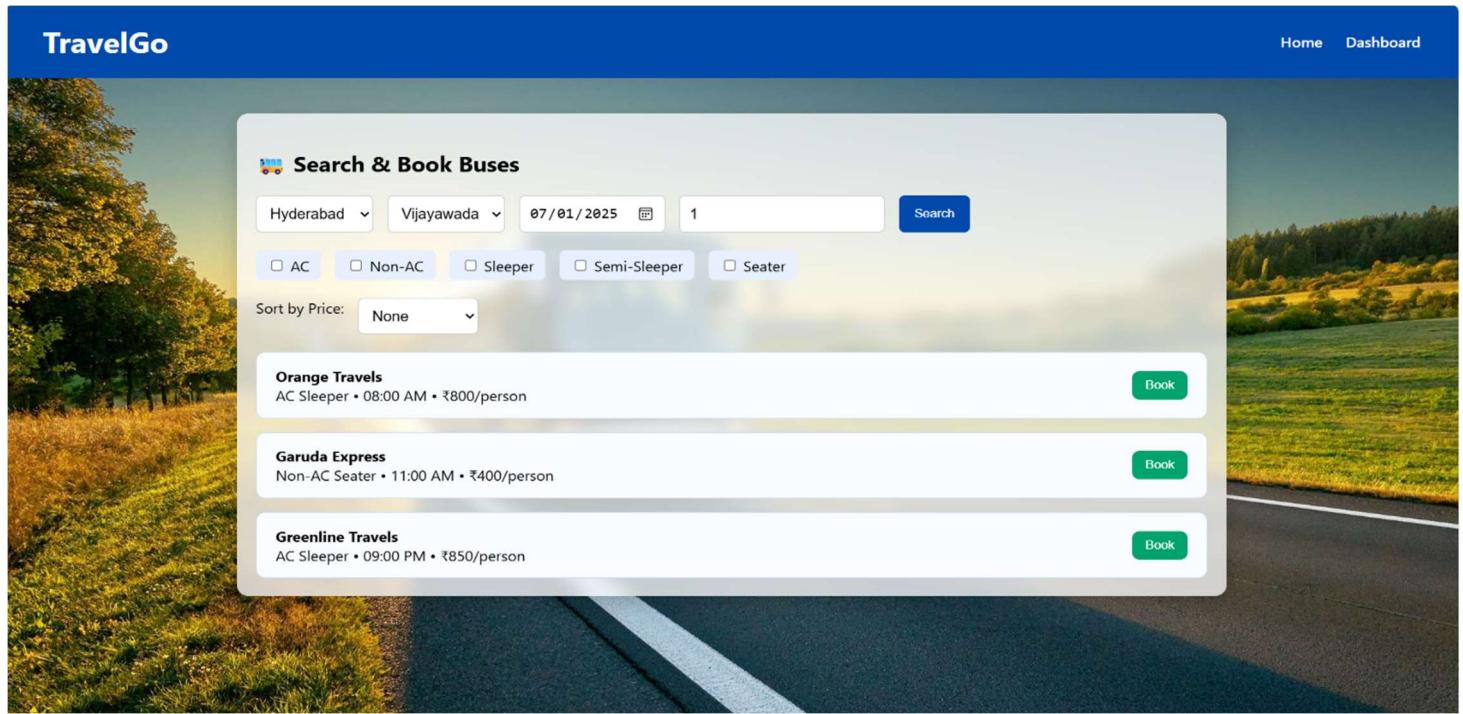
Bus Booking code:



```
EXPLORER
  TRAVELGO
    static\images
      flight1.jpg
      Hotel.jpg
      Hotel1.jpg
      image1.jpg
      index.jpg
      login.jpg
      signup.jpg
      train-icon.jpg
      train.jpg
      train1.jpg
      world.jpg
    templates
      aboutus.html
      bus.html
        confirm_bus_details.html
        confirm_flight_details.html
        confirm_hotel_details.html
        confirm_train_details.html
        dashboard.html
        flight.html
        forgot-password.html
        home.html
        hotel.html
        index.html
        login.html
        logout.html
        resetpassword.html
        signup.html
        support.html
        train.html
      app.py
      README.md

  Welcome app.py M bus.html X
templates > bus.html > html > body > header > nav
  2   <html lang="en">
  3     <body>
  117       <div class="container">
  126         <h2> Search & Book Buses</h2>
  128
  129         <div class="form-row">
  130           <select id="source">
  131             <option value="">From</option>
  132             <option value="Hyderabad">Hyderabad</option>
  133             <option value="Vijayawada">Vijayawada</option>
  134             <option value="Guntur">Guntur</option>
  135             <option value="Bengaluru">Bengaluru</option>
  136             <option value="Chennai">Chennai</option>
  137           </select>
  138
  139           <select id="destination">
  140             <option value="">To</option>
  141             <option value="Hyderabad">Hyderabad</option>
  142             <option value="Vijayawada">Vijayawada</option>
  143             <option value="Guntur">Guntur</option>
  144             <option value="Bengaluru">Bengaluru</option>
  145             <option value="Chennai">Chennai</option>
  146           </select>
  147
  148           <input type="date" id="date" />
  149           <input type="number" id="numPersons" min="1" value="1" placeholder="No. of Persons" />
  150           <button id="search-btn" class="search-btn">Search</button>
  151         </div>
  152
  153         <div class="filters">
  154           <label><input type="checkbox" id="AC" /> AC</label>
  155           <label><input type="checkbox" id="Non-AC" /> Non-AC</label>
  156           <label><input type="checkbox" id="Sleeper" /> Sleeper</label>
  157           <label><input type="checkbox" id="Semi-Sleeper" /> Semi-Sleeper</label>
  158           <label><input type="checkbox" id="Seater" /> Seater</label>
  159         </div>
  160
  161         <div class="sort-row">
  162           <label for="sort">Sort by Price:</label>
  163           <select id="sort">
  164             <option value="">None</option>
  165           </select>
  166         </div>
```

Bus booking page:



Bus Confirm Booking code:

```
File Edit Selection View Go Run Terminal Help ... Welcome app.py M confirm_bus_details.html templates > confirm_bus_details.html > html > body > script > addEventListener('click') callback
186 <html lang="en">
187   <body>
188     <div class="container">
189       <div class="booking-summary">
200         ...
211       </div>
212       <div id="seat-map" class="seat-map"></div>
213       <input type="hidden" id="selectedSeatsInput" />
214       ...
215       <p><strong>Bus Name:</strong> {{ booking.name }}</p>
216       <p><strong>Route:</strong> {{ booking.source }} to {{ booking.destination }}</p>
217       <p><strong>Date:</strong> {{ booking.travel_date }}</p>
218       <p><strong>Time:</strong> {{ booking.time }}</p>
219       <p><strong>Type:</strong> {{ booking.type }}</p>
220       <p><strong>Passengers:</strong> {{ booking.num_persons }}</p>
221       <p><strong>Selected Seats:</strong> <span id="selectedSeatsDisplay">None</span></p>
222       ...
223       <p class="total-price">Total Price: ₹{{ ":.2f".format(booking.total_price) }}</p>
224     </div>
225     <div class="buttons-container">
226       <button id="confirmBookingBtn">Confirm Booking</button>
227       <a href="/bus" class="btn btn-secondary">Go Back to Search</a>
228     </div>
229   </div>
230   ...
231   </div>
232   ...
233   <footer>
234     &copy; 2025 TravelGo. All rights reserved.
235   </footer>
236   ...
237   ...
238   ...
239   <script>
240     const seatLabels = [
241       "S1", "S2", "", "S3", "S4",
242       "S5", "S6", "", "S7", "S8",
243       "S9", "S10", "", "S11", "S12",
244       "S13", "S14", "", "S15", "S16",
245       "S17", "S18", "", "S19", "S20",
246       "S21", "S22", "", "S23", "S24",
247       "S25", "S26", "", "S27", "S28"
248     ];

```

Bus confirm booking page:

Booking Details:

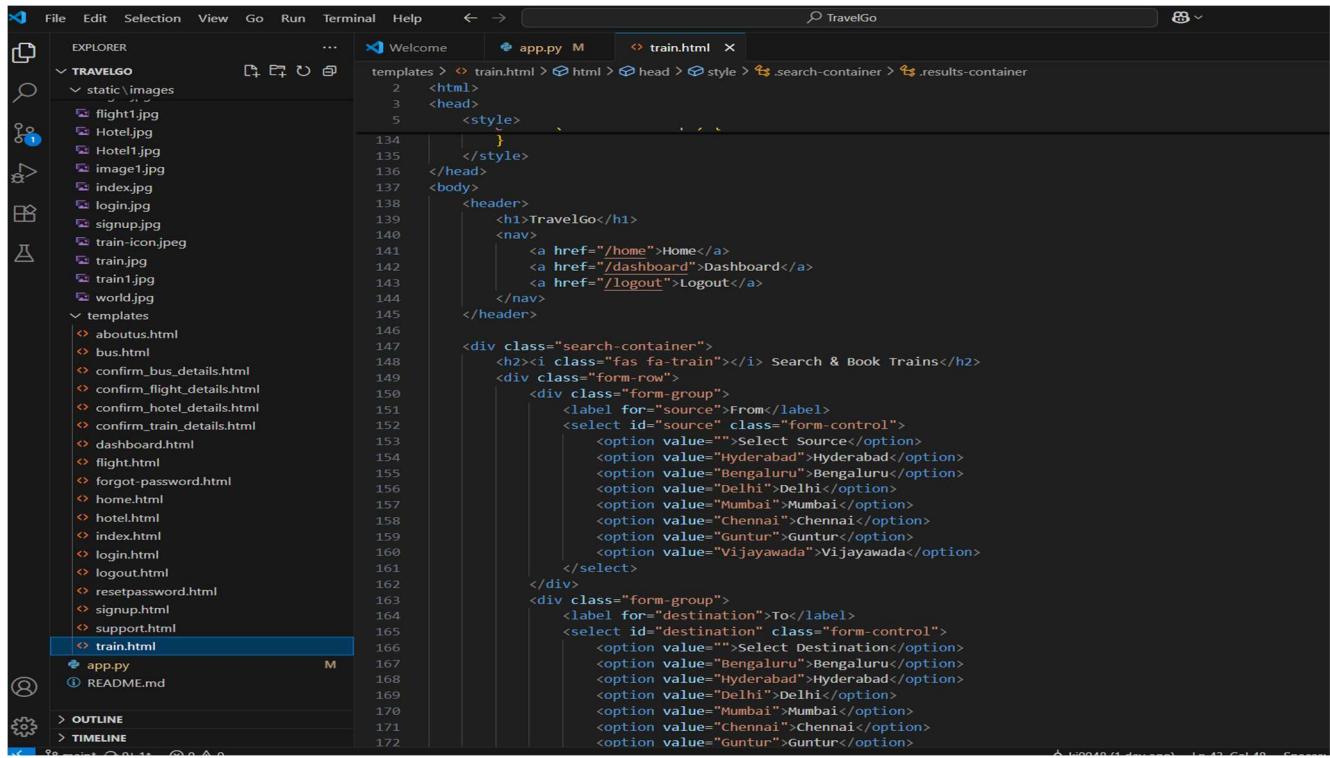
Select Your Seats:

Legend: Available (Grey), Selected (Green), Booked (Black), Women Only (Pink)

S1	S2	S3 ♀	S4
S5	S6	S7 ♀	S8
S9	S10	S11	S12
S13	S14	S15 ♀	S16
S17	S18	S19	S20
S21	S22 ♀	S23	S24
S25	S26	S27	S28 ♀

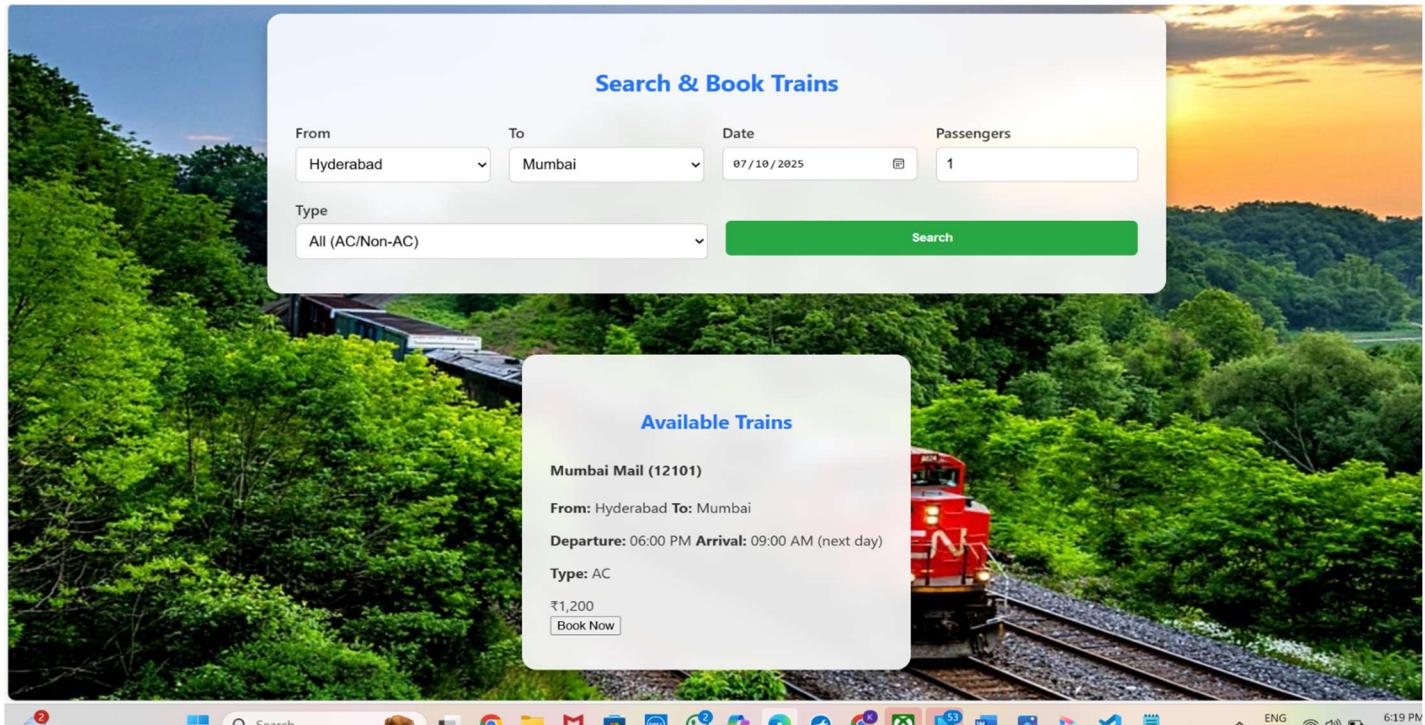
Bus Name: Orange Travels
Route: Hyderabad to Vijayawada
Date: 2025-07-01
Time: 08:00 AM
Type: AC Sleeper

Train code :

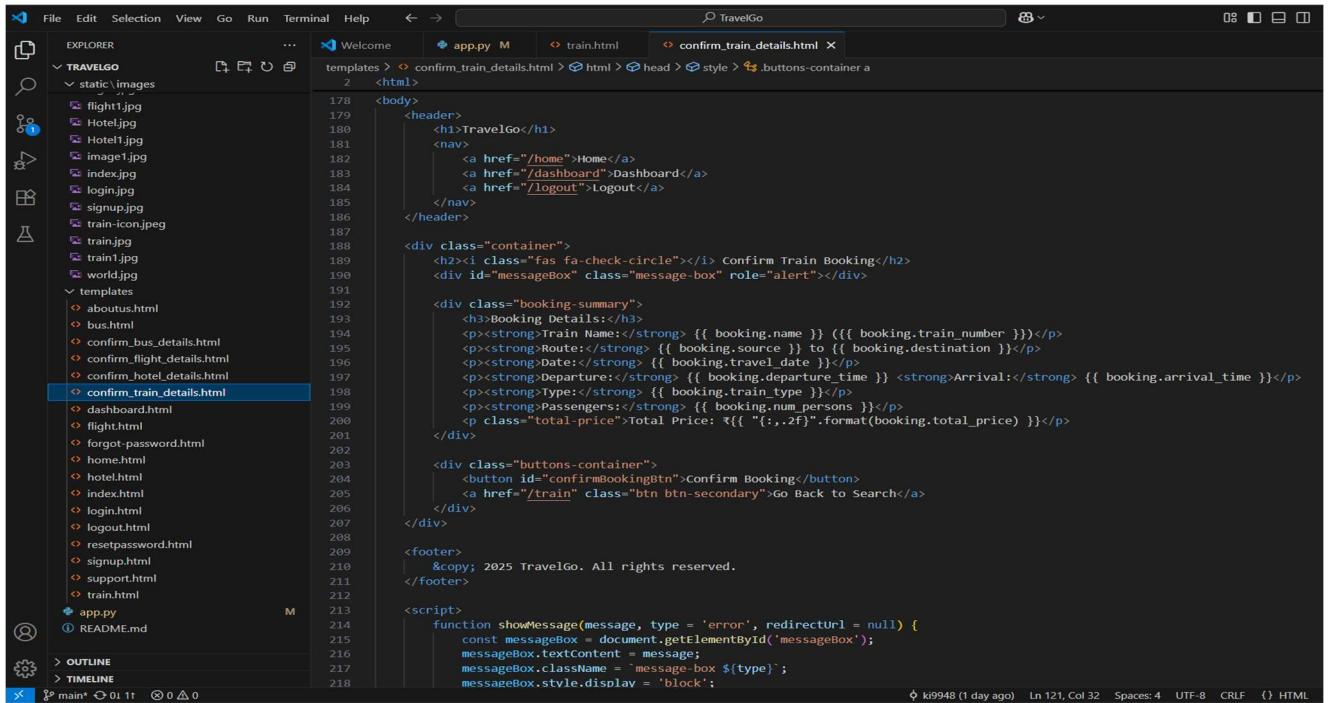


```
File Edit Selection View Go Run Terminal Help ← → Welcome app.py M train.html
EXPLORER templates > train.html > html > head > style > .search-container > .results-container
static/images
  flight1.jpg
  Hotel.jpg
  Hotel1.jpg
  image1.jpg
  index.jpg
  login.jpg
  signup.jpg
  train-icon.jpeg
  train.jpg
  train1.jpg
  world.jpg
templates
  aboutus.html
  bus.html
  confirm_bus_details.html
  confirm_flight_details.html
  confirm_hotel_details.html
  confirm_train_details.html
  dashboard.html
  flight.html
  forgot-password.html
  home.html
  hotel.html
  index.html
  login.html
  logout.html
  resetpassword.html
  signup.html
  support.html
  train.html
  app.py
  README.md
> OUTLINE
> TIMELINE
134   <html>
135     <head>
136       <style>
137         ...
138       </style>
139     </head>
140     <body>
141       <header>
142         <h1>TravelGo</h1>
143         <nav>
144           <a href="/home">Home</a>
145           <a href="/dashboard">Dashboard</a>
146           <a href="/logout">Logout</a>
147         </nav>
148       </header>
149       <div class="search-container">
150         <h2><i class="fas fa-train"></i> Search & Book Trains</h2>
151         <div class="form-row">
152           <div class="form-group">
153             <label for="source">From</label>
154             <select id="source" class="form-control">
155               <option value="">Select Source</option>
156               <option value="Hyderabad">Hyderabad</option>
157               <option value="Bengaluru">Bengaluru</option>
158               <option value="Delhi">Delhi</option>
159               <option value="Mumbai">Mumbai</option>
160               <option value="Chennai">Chennai</option>
161               <option value="Guntur">Guntur</option>
162               <option value="Vijayawada">Vijayawada</option>
163             </select>
164           </div>
165           <div class="form-group">
166             <label for="destination">To</label>
167             <select id="destination" class="form-control">
168               <option value="">Select Destination</option>
169               <option value="Bengaluru">Bengaluru</option>
170               <option value="Hyderabad">Hyderabad</option>
171               <option value="Delhi">Delhi</option>
172               <option value="Mumbai">Mumbai</option>
173               <option value="Chennai">Chennai</option>
174               <option value="Guntur">Guntur</option>
175             </select>
176           </div>
177         </div>
178       </div>
179     </body>
180   </html>
```

Train booking page:



Train confirm booking code:



The screenshot shows a code editor with the 'confirm_train_details.html' file open. The file contains HTML, CSS, and JavaScript code for a booking confirmation page. The code includes sections for header navigation, booking summary, and a footer copyright notice. A script block handles message display logic.

```
<body>
  <header>
    <h1>TravelGo</h1>
    <nav>
      <a href="/home">Home</a>
      <a href="/dashboard">Dashboard</a>
      <a href="/logout">Logout</a>
    </nav>
  </header>

  <div class="container">
    <h2><i class="fas fa-check-circle"></i> Confirm Train Booking</h2>
    <div id="messageBox" class="message-box" role="alert"></div>

    <div class="booking-summary">
      <h3>Booking Details:</h3>
      <p><strong>Train Name:</strong> {{ booking.name }} ({{ booking.train_number }})</p>
      <p><strong>Route:</strong> {{ booking.source }} to {{ booking.destination }}</p>
      <p><strong>Date:</strong> {{ booking.travel_date }}</p>
      <p><strong>Departure:</strong> {{ booking.departure_time }} <strong>Arrival:</strong> {{ booking.arrival_time }}</p>
      <p><strong>Type:</strong> {{ booking.train_type }}</p>
      <p><strong>Passengers:</strong> {{ booking.num_persons }}</p>
      <p class="total-price">Total Price: ₹{{ (".,.2f").format(booking.total_price) }}</p>
    </div>

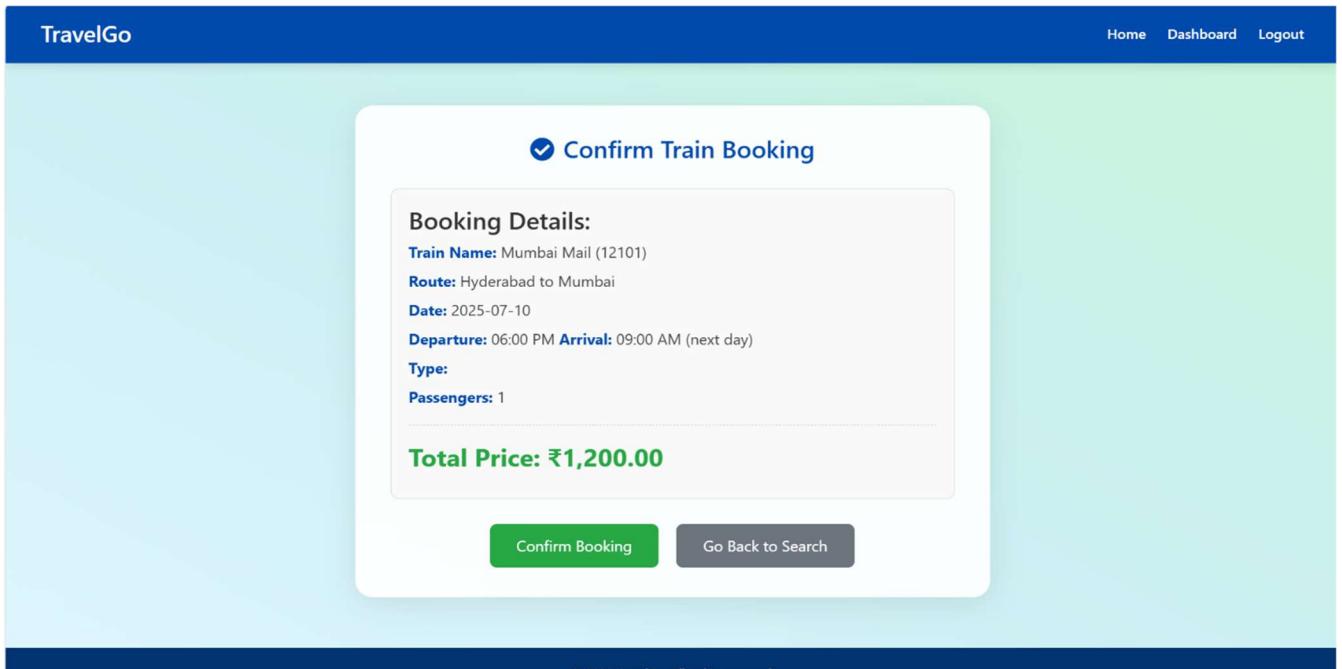
    <div class="buttons-container">
      <button id="confirmBookingBtn">Confirm Booking</button>
      <a href="/train" class="btn btn-secondary">Go Back to Search</a>
    </div>
  </div>

  <footer>
    &copy; 2025 TravelGo. All rights reserved.
  </footer>

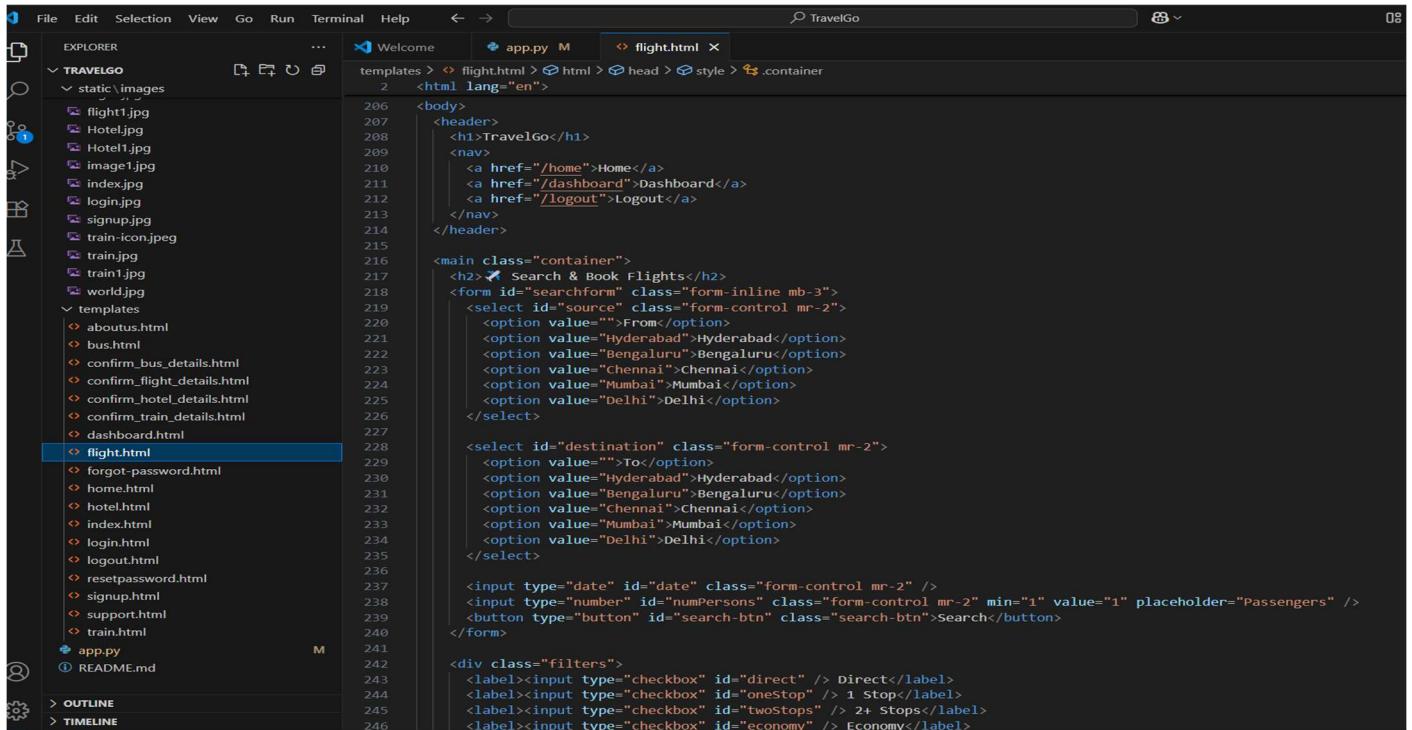
  <script>
    function showMessage(message, type = 'error', redirectUrl = null) {
      const messageBox = document.getElementById('messageBox');
      messageBox.textContent = message;
      messageBox.className = `message-box ${type}`;
      messageBox.style.display = 'block';
    }
  </script>

```

Train confirm booking page:



Flight code:



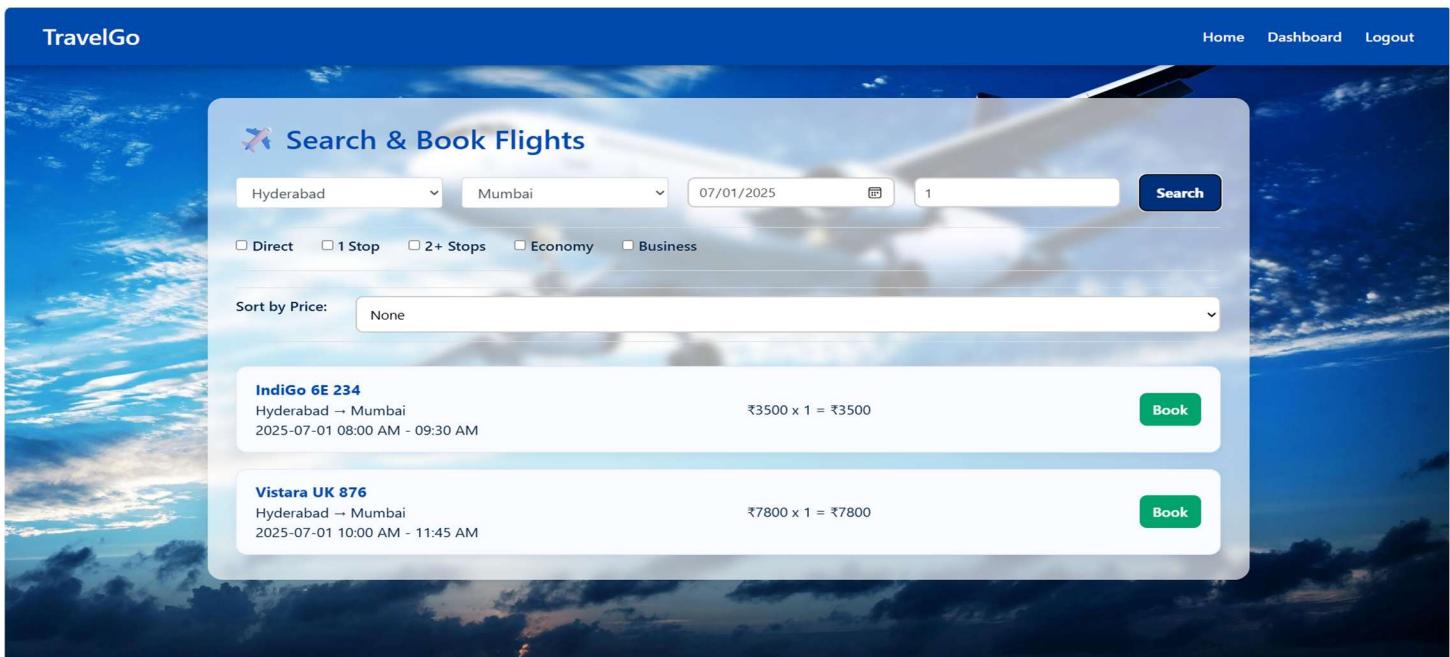
The screenshot shows a code editor interface with a dark theme. On the left is a file explorer sidebar containing project files like 'TRAVELOGO', 'static\images', 'templates', and 'flight.html'. The main pane displays the 'flight.html' template code. The code includes an HTML header with navigation links for 'Home', 'Dashboard', and 'Logout'. It features a search form with dropdowns for 'From' (Hyderabad, Bengaluru, Chennai, Mumbai, Delhi) and 'To' (Hyderabad, Bengaluru, Chennai, Mumbai, Delhi). Below the form is a date input field, a passenger count input, and a 'Search' button. A 'filters' section contains checkboxes for 'Direct', '1 Stop', '2+ Stops', 'Economy', and 'Business'. The code uses Bootstrap classes for styling.

```
<body>
  <header>
    <h1>TravelGo</h1>
    <nav>
      <a href="/home">Home</a>
      <a href="/dashboard">Dashboard</a>
      <a href="/logout">Logout</a>
    </nav>
  </header>

  <main class="container">
    <h2>Search & Book Flights</h2>
    <form id="searchform" class="form-inline mb-3">
      <select id="source" class="form-control mr-2">
        <option value="">From</option>
        <option value="Hyderabad">Hyderabad</option>
        <option value="Bengaluru">Bengaluru</option>
        <option value="Chennai">Chennai</option>
        <option value="Mumbai">Mumbai</option>
        <option value="Delhi">Delhi</option>
      </select>
      <select id="destination" class="form-control mr-2">
        <option value="">To</option>
        <option value="Hyderabad">Hyderabad</option>
        <option value="Bengaluru">Bengaluru</option>
        <option value="Chennai">Chennai</option>
        <option value="Mumbai">Mumbai</option>
        <option value="Delhi">Delhi</option>
      </select>
      <input type="date" id="date" class="form-control mr-2" />
      <input type="number" id="numPersons" class="form-control mr-2" min="1" value="1" placeholder="Passengers" />
      <button type="button" id="search-btn" class="search-btn">Search</button>
    </form>
    <div class="filters">
      <label><input type="checkbox" id="direct" /> Direct</label>
      <label><input type="checkbox" id="oneStop" /> 1 Stop</label>
      <label><input type="checkbox" id="twoStops" /> 2+ Stops</label>
      <label><input type="checkbox" id="economy" /> Economy</label>
    </div>
  </main>

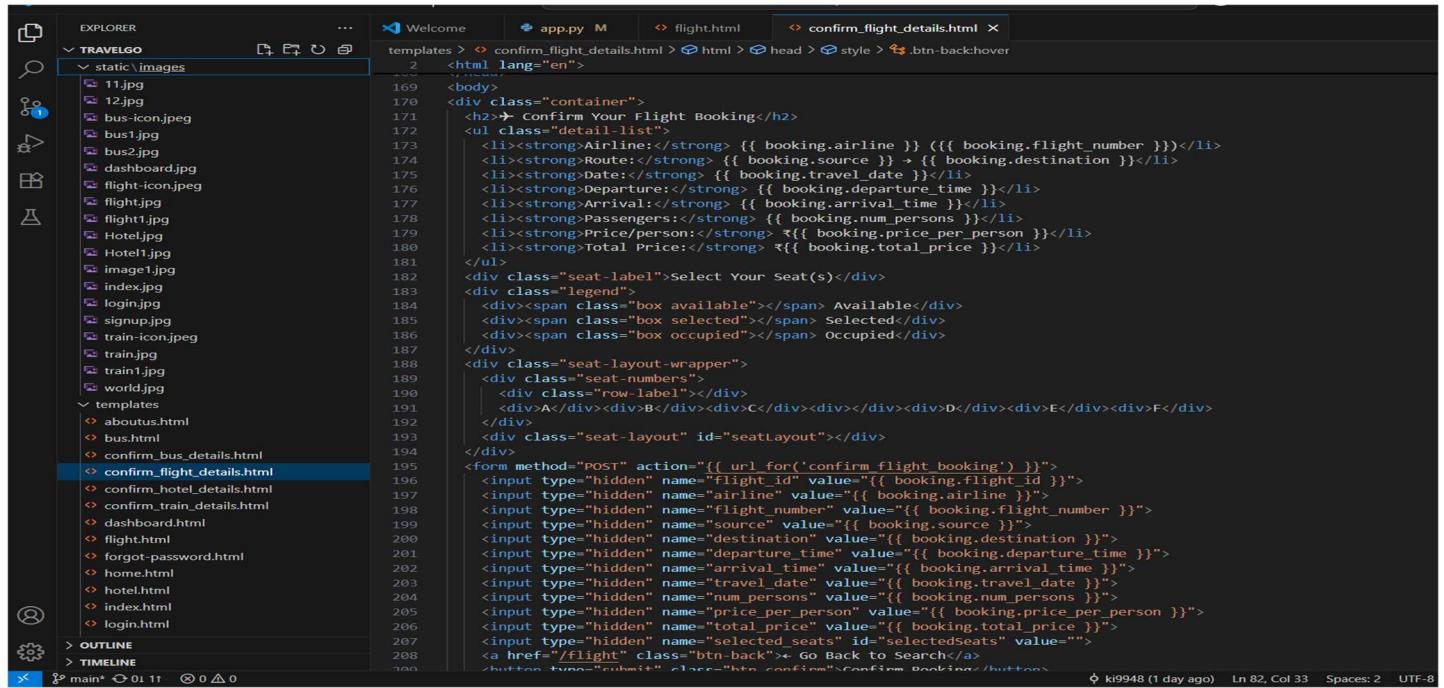
```

Flight page:



The screenshot shows the 'TravelGo' application's flight search interface. At the top, there's a header bar with 'TravelGo' and navigation links for 'Home', 'Dashboard', and 'Logout'. The main content area has a blue header 'Search & Book Flights' with a background image of an airplane in flight. Below the header is a search form with dropdowns for 'From' (Hyderabad) and 'To' (Mumbai), a date input (07/01/2025), and a search button. Underneath the form are checkboxes for travel options: 'Direct', '1 Stop', '2+ Stops', 'Economy', and 'Business'. A 'Sort by Price:' dropdown is set to 'None'. Two flight options are listed: 'IndiGo 6E 234' from Hyderabad to Mumbai on 2025-07-01 at 08:00 AM for ₹3500, and 'Vistara UK 876' from Hyderabad to Mumbai on 2025-07-01 at 10:00 AM for ₹7800. Each option has a green 'Book' button.

Flight confirm booking code:

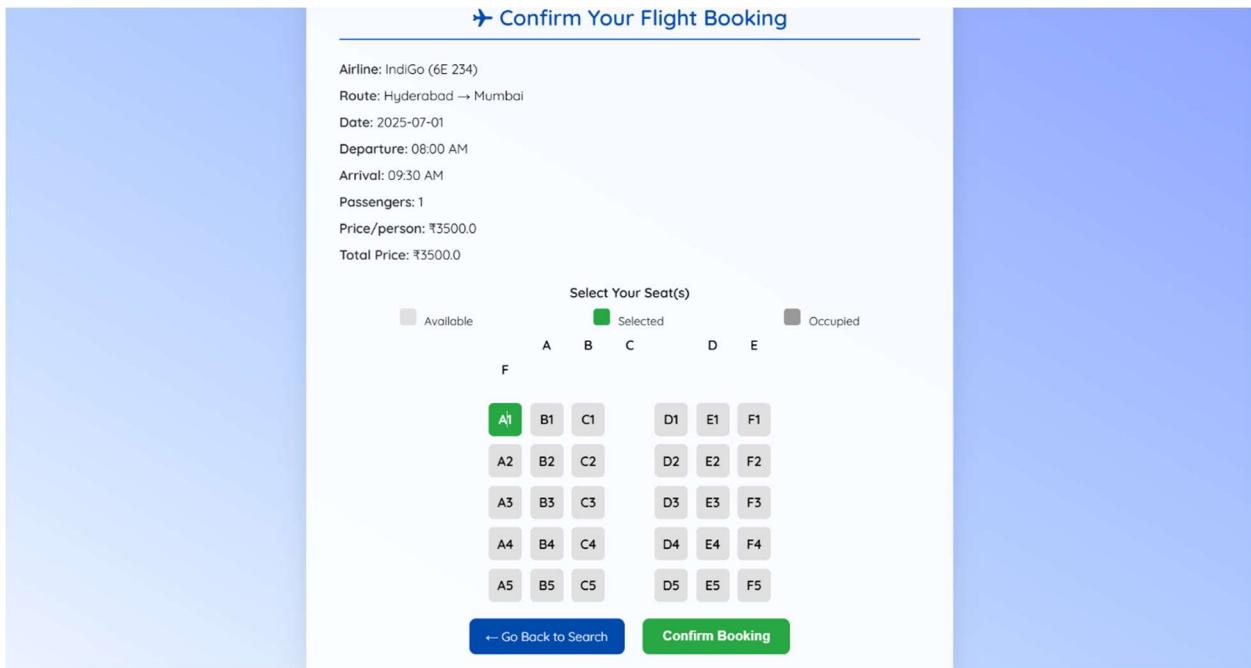


The screenshot shows a code editor interface with the following details:

- EXPLORER** panel on the left showing project structure:
 - TRAVELGO
 - static\images (containing 11.jpg, 12.jpg, bus-icon.jpeg, bus1.jpg, bus2.jpg, dashboard.jpg, flight-icon.jpeg, flight1.jpg, flight11.jpg, Hotel.jpg, Hotel1.jpg, image1.jpg, index.jpg, login.jpg, signup.jpg, train-icon.jpeg, train.jpg, train1.jpg, world.jpg)
 - templates (containing aboutus.html, bus.html, confirm_bus_details.html, confirm_flight_details.html, confirm_hotel_details.html, confirm_train_details.html, dashboard.html, flight.html, forgot-password.html, home.html, hotel.html, index.html, login.html)
- CODE** tab selected, showing the content of `confirm_flight_details.html`. The code is a template for a flight booking confirmation page.
- STATUS** bar at the bottom showing file statistics: k9948 (1 day ago), Ln 82, Col 33, Spaces: 2, UTF-8.

```
templates > confirm_flight_details.html > html > head > style > .btn-back:hover
2 <html lang="en">
169 <body>
170 <div class="container">
171 <h2> Confirm Your Flight Booking</h2>
172 <ul class="detail-list">
173 <li><strong>Airline:</strong> {{ booking.airline }} ({{ booking.flight_number }})</li>
174 <li><strong>Route:</strong> {{ booking.source }} -> {{ booking.destination }}</li>
175 <li><strong>Date:</strong> {{ booking.travel_date }}</li>
176 <li><strong>Departure:</strong> {{ booking.departure_time }}</li>
177 <li><strong>Arrival:</strong> {{ booking.arrival_time }}</li>
178 <li><strong>Passengers:</strong> {{ booking.num_persons }}</li>
179 <li><strong>Price/person:</strong> ₹{{ booking.price_per_person }}</li>
180 <li><strong>Total Price:</strong> ₹{{ booking.total_price }}</li>
181 </ul>
182 <div class="seat-label">Select Your Seat(s)</div>
183 <div class="legend">
184 <div><span class="box available"></span> Available</div>
185 <div><span class="box selected"></span> Selected</div>
186 <div><span class="box occupied"></span> Occupied</div>
187 </div>
188 <div class="seat-layout-wrapper">
189 <div class="seat-numbers">
190 <div class="row-label">A</div>
191 <div>A</div><div>B</div><div>C</div><div>D</div><div>E</div><div>F</div>
192 </div>
193 <div class="seat-layout" id="seatLayout"></div>
194 </div>
195 <form method="POST" action="{{ url_for('confirm_flight_booking') }}">
196 <input type="hidden" name="flight_id" value="{{ booking.flight_id }}"/>
197 <input type="hidden" name="airline" value="{{ booking.airline }}"/>
198 <input type="hidden" name="flight_number" value="{{ booking.flight_number }}"/>
199 <input type="hidden" name="source" value="{{ booking.source }}"/>
200 <input type="hidden" name="destination" value="{{ booking.destination }}"/>
201 <input type="hidden" name="departure_time" value="{{ booking.departure_time }}"/>
202 <input type="hidden" name="arrival_time" value="{{ booking.arrival_time }}"/>
203 <input type="hidden" name="travel_date" value="{{ booking.travel_date }}"/>
204 <input type="hidden" name="num_persons" value="{{ booking.num_persons }}"/>
205 <input type="hidden" name="price_per_person" value="{{ booking.price_per_person }}"/>
206 <input type="hidden" name="total_price" value="{{ booking.total_price }}"/>
207 <input type="hidden" name="selected_seats" id="selectedSeats" value="">
208 <a href="/flight" class="btn-back">< Go Back to Search</a>
209 <button type="submit" class="btn-confirm">Confirm Booking</button>
210 </form>
```

Flight confirm booking page:



The screenshot shows a web page titled "Confirm Your Flight Booking" with the following content:

Airline: IndiGo (6E 234)
Route: Hyderabad → Mumbai
Date: 2025-07-01
Departure: 08:00 AM
Arrival: 09:30 AM
Passengers: 1
Price/person: ₹3500.0
Total Price: ₹3500.0

Select Your Seat(s)

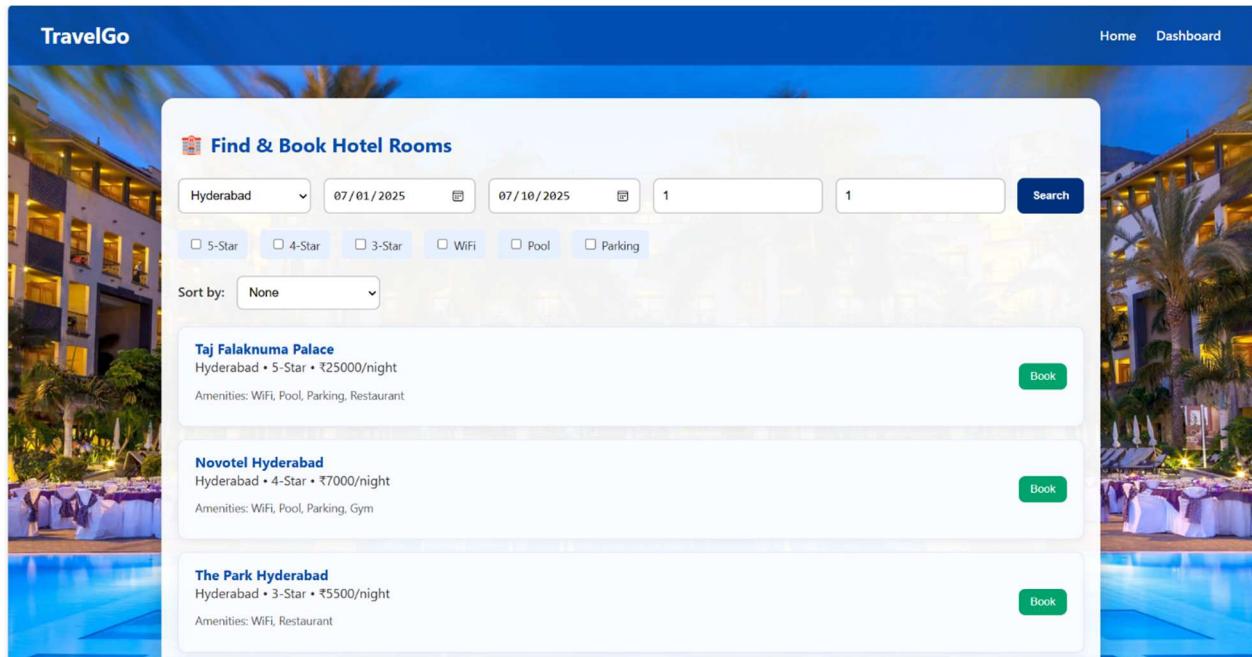
	A	B	C	D	E	F
Available	A1	B1	C1	D1	E1	F1
Selected	A2	B2	C2	D2	E2	F2
Occupied	A3	B3	C3	D3	E3	F3
	A4	B4	C4	D4	E4	F4
	A5	B5	C5	D5	E5	F5

[← Go Back to Search](#) [Confirm Booking](#)

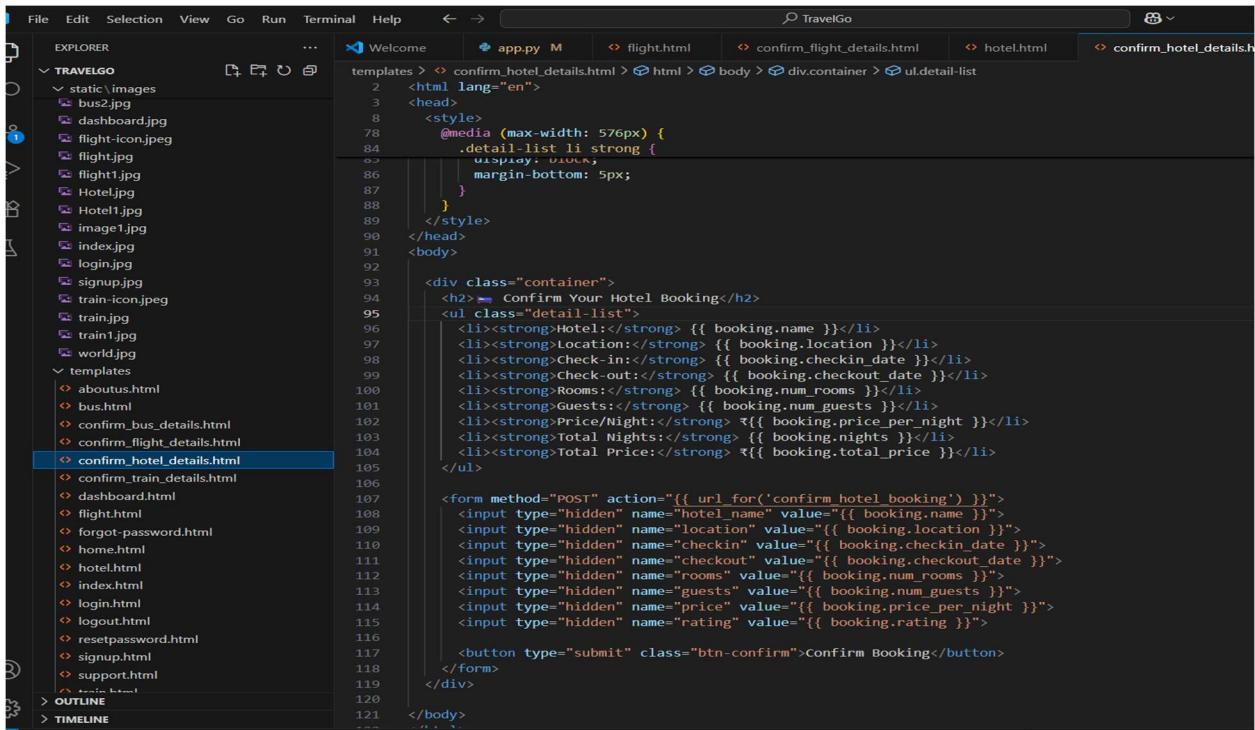
Hotel code:

```
<div class="container">
    <h2> Find & Book Hotel Rooms</h2>
    <div class="form-row">
        <select id="location">
            <option value="">Select City</option>
            <option value="Hyderabad">Hyderabad</option>
            <option value="Mumbai">Mumbai</option>
            <option value="Delhi">Delhi</option>
            <option value="Bangalore">Bangalore</option>
        </select>
        <input type="date" id="checkinDate" />
        <input type="date" id="checkoutDate" />
        <input type="number" id="numRooms" min="1" value="1" placeholder="No. of Rooms" />
        <input type="number" id="numGuests" min="1" value="1" placeholder="No. of Guests" />
        <button id="search-btn" class="search-btn">Search</button>
    </div>
    <div class="filters">
        <label><input type="checkbox" id="fiveStar" /> 5-Star</label>
        <label><input type="checkbox" id="fourStar" /> 4-Star</label>
        <label><input type="checkbox" id="threeStar" /> 3-Star</label>
        <label><input type="checkbox" id="WiFi" /> WiFi</label>
        <label><input type="checkbox" id="Pool" /> Pool</label>
        <label><input type="checkbox" id="Parking" /> Parking</label>
    </div>
    <div class="sort-row">
        <label for="sort">Sort by:</label>
        <select id="sort">
            <option value="">None</option>
            <option value="price-low">Price: Low to High</option>
            <option value="price-high">Price: High to Low</option>
            <option value="rating-high">Rating: High to Low</option>
        </select>
    </div>
    <div id="hotel-list" class="hotel-list"></div>
</div>
```

Hotel page:

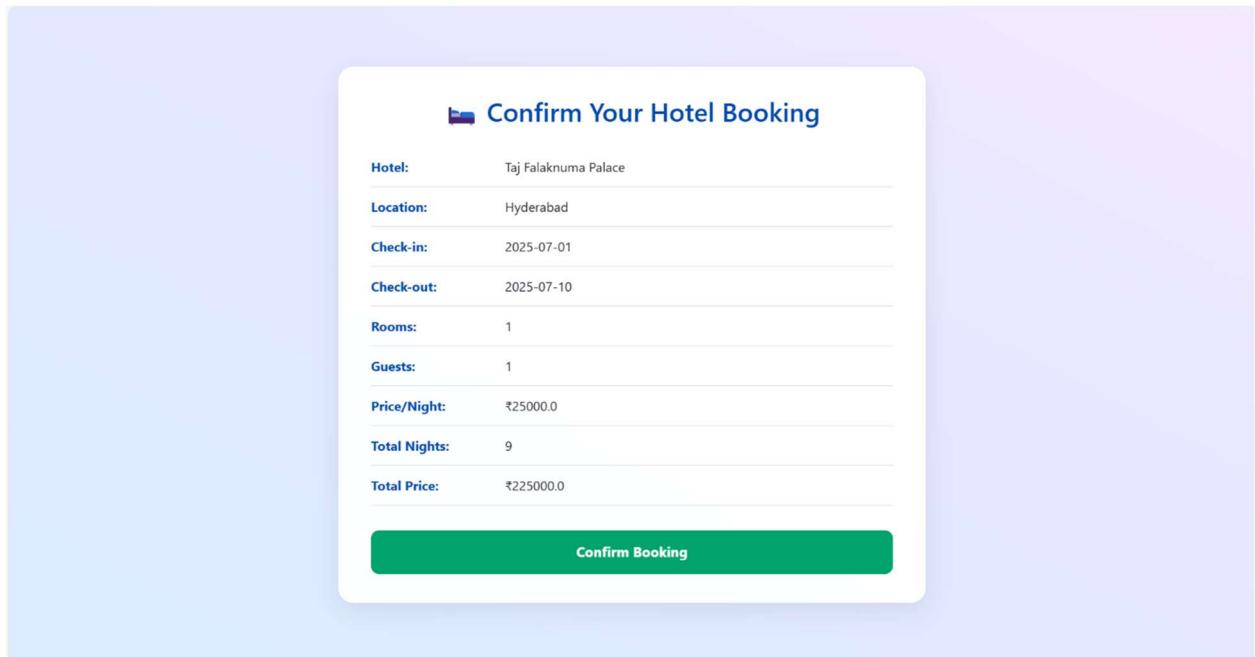


Hotel confirm booking code:



```
File Edit Selection View Go Run Terminal Help ← → TravelGo
EXPLORER ... templates > confirm_hotel_details.html app.py M flight.html confirm_flight_details.html hotel.html confirm_hotel_details.html
TRAVELGO static\images bus2.jpg dashboard.jpg flight-icon.jpeg flight.jpg flight1.jpg Hotel.jpg Hotel1.jpg image1.jpg index.jpg login.jpg signup.jpg train-icon.jpg train.jpg train1.jpg world.jpg
templates > aboutus.html bus.html confirm_bus_details.html confirm_flight_details.html confirm_hotel_details.html
confirm_train_details.html dashboard.html flight.html forgot-password.html home.html hotel.html index.html login.html logout.html resetpassword.html signup.html support.html
OUTLINE TIMELINE
<html lang="en">
  <head>
    <style>
      @media (max-width: 576px) {
        .detail-list li strong {
          display: block;
          margin-bottom: 5px;
        }
      }
    </style>
  </head>
  <body>
    <div class="container">
      <h2>Confirm Your Hotel Booking</h2>
      <ul class="detail-list">
        <li><strong>Hotel:</strong> {{ booking.name }}</li>
        <li><strong>Location:</strong> {{ booking.location }}</li>
        <li><strong>Check-in:</strong> {{ booking.checkin_date }}</li>
        <li><strong>Check-out:</strong> {{ booking.checkout_date }}</li>
        <li><strong>Rooms:</strong> {{ booking.num_rooms }}</li>
        <li><strong>Guests:</strong> {{ booking.num_guests }}</li>
        <li><strong>Price/Night:</strong> ₹{{ booking.price_per_night }}</li>
        <li><strong>Total Nights:</strong> {{ booking.nights }}</li>
        <li><strong>Total Price:</strong> ₹{{ booking.total_price }}</li>
      </ul>
      <form method="POST" action="{{ url_for('confirm_hotel_booking') }}">
        <input type="hidden" name="hotel_name" value="{{ booking.name }}"/>
        <input type="hidden" name="location" value="{{ booking.location }}"/>
        <input type="hidden" name="checkin" value="{{ booking.checkin_date }}"/>
        <input type="hidden" name="checkout" value="{{ booking.checkout_date }}"/>
        <input type="hidden" name="rooms" value="{{ booking.num_rooms }}"/>
        <input type="hidden" name="guests" value="{{ booking.num_guests }}"/>
        <input type="hidden" name="price" value="{{ booking.price_per_night }}"/>
        <input type="hidden" name="rating" value="{{ booking.rating }}"/>
        <button type="submit" class="btn-confirm">Confirm Booking</button>
      </form>
    </div>
  </body>
</html>
```

Hotel confirm booking page:

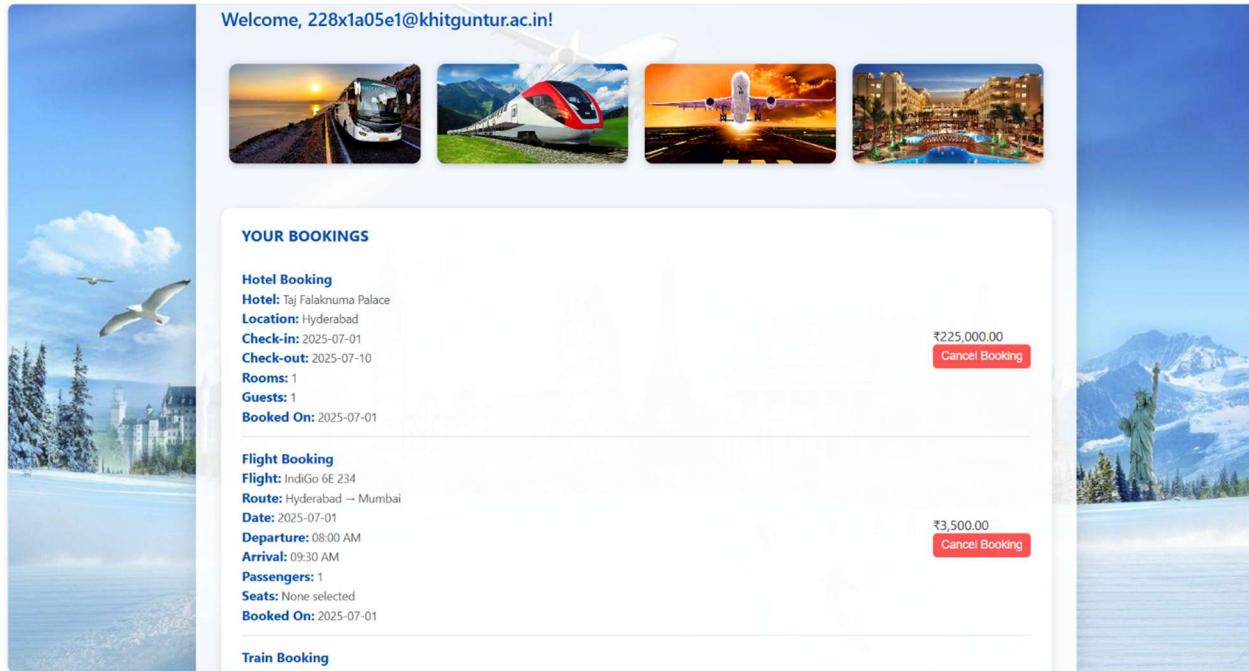


The screenshot shows a web application interface titled "Confirm Your Hotel Booking". The page lists the following booking details:

- Hotel:** Taj Falaknuma Palace
- Location:** Hyderabad
- Check-in:** 2025-07-01
- Check-out:** 2025-07-10
- Rooms:** 1
- Guests:** 1
- Price/Night:** ₹25000.0
- Total Nights:** 9
- Total Price:** ₹225000.0

At the bottom of the form is a large green button labeled "Confirm Booking".

Bookings page:



Welcome, 228x1a05e1@khitguntur.ac.in!

YOUR BOOKINGS

Hotel Booking

Hotel: Taj Falaknuma Palace
Location: Hyderabad
Check-in: 2025-07-01
Check-out: 2025-07-10
Rooms: 1
Guests: 1
Booked On: 2025-07-01

₹225,000.00
Cancel Booking

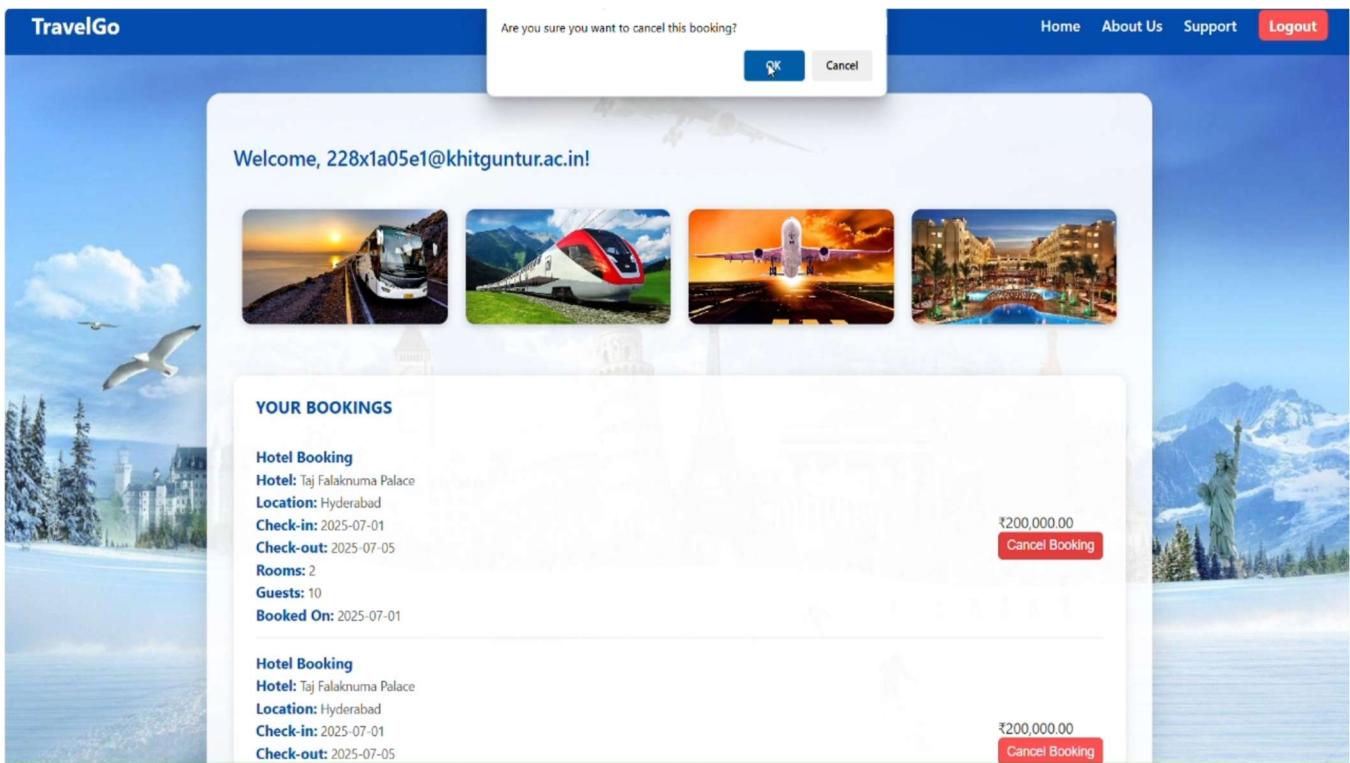
Flight Booking

Flight: IndiGo 6E 234
Route: Hyderabad → Mumbai
Date: 2025-07-01
Departure: 08:00 AM
Arrival: 09:30 AM
Passengers: 1
Seats: None selected
Booked On: 2025-07-01

₹3,500.00
Cancel Booking

Train Booking

Cancel booking page:



Are you sure you want to cancel this booking?

OK Cancel

Welcome, 228x1a05e1@khitguntur.ac.in!

YOUR BOOKINGS

Hotel Booking

Hotel: Taj Falaknuma Palace
Location: Hyderabad
Check-in: 2025-07-01
Check-out: 2025-07-05
Rooms: 2
Guests: 10
Booked On: 2025-07-01

₹200,000.00
Cancel Booking

Hotel Booking

Hotel: Taj Falaknuma Palace
Location: Hyderabad
Check-in: 2025-07-01
Check-out: 2025-07-05

₹200,000.00
Cancel Booking

Conclusion:

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

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

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