

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

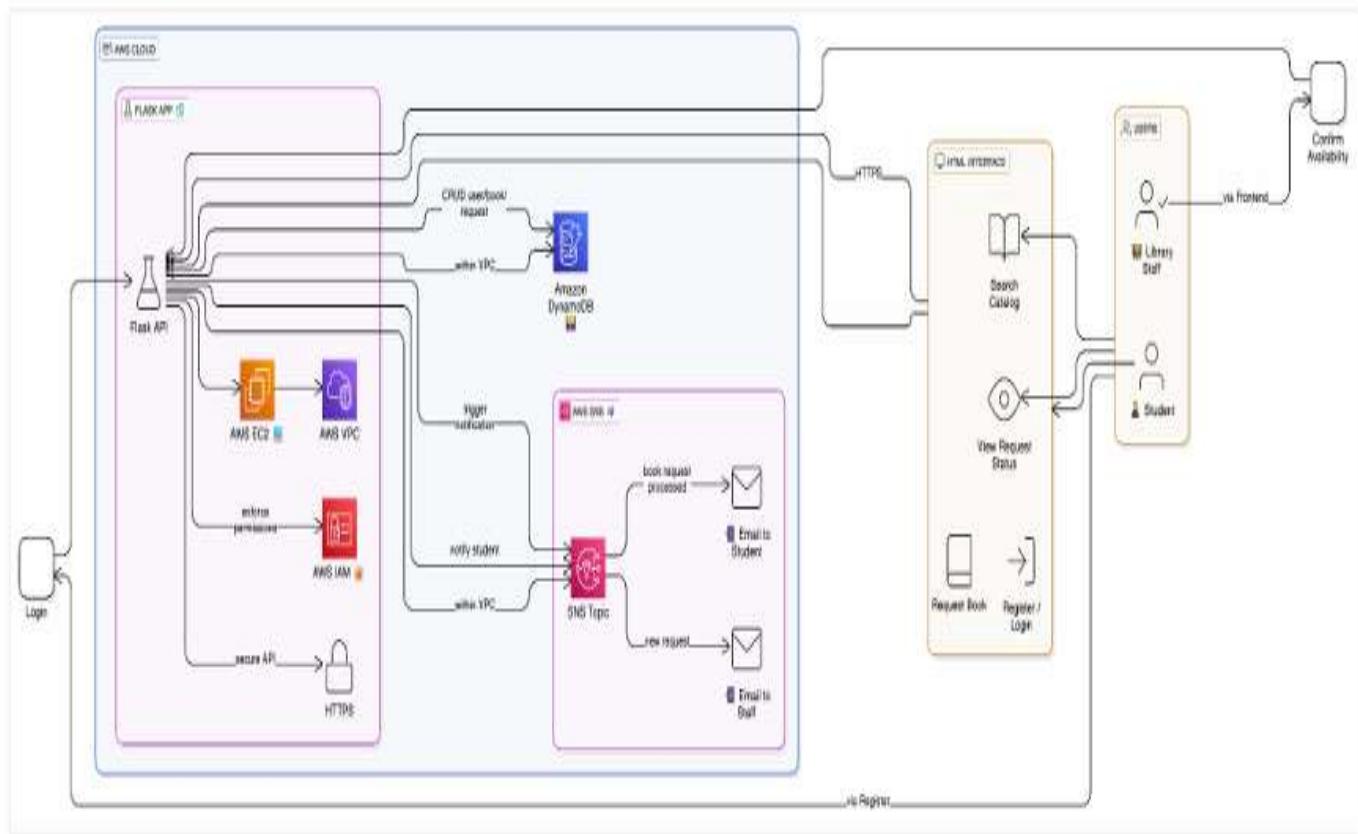
Once a booking is made—whether it's a train ticket or a hotel stay—TravelGo uses AWS SNS to instantly notify the user. For example, after a student books a hotel in Chennai, SNS sends a real-time email notification confirming the booking with all the relevant details. This notification is triggered from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

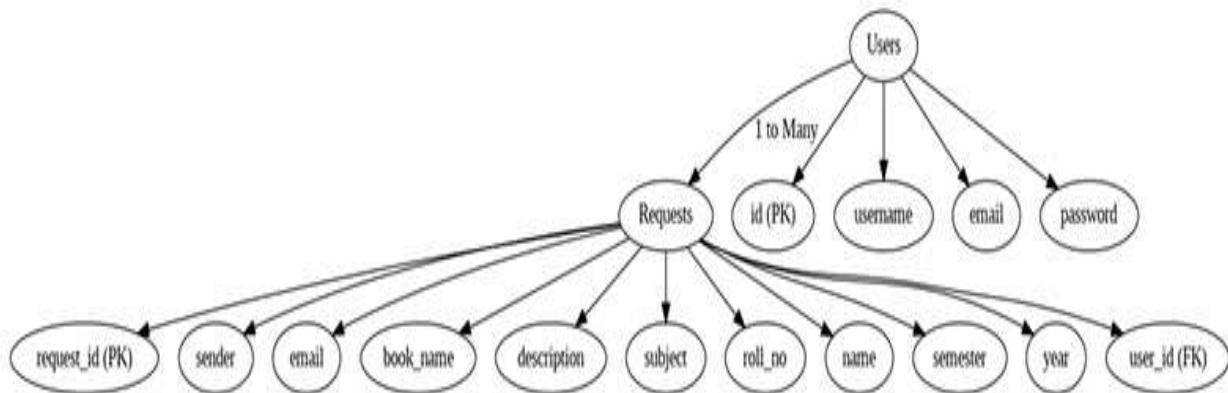
Scenario 3: Dynamic Dashboard with Personal Travel History

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

The screenshot shows the troven platform interface. On the left, there's a sidebar with a profile icon, the name 'GINJUPALLI', and an email address '228xla05e@khitguntur.oc.in'. The main area displays the 'AWS- TravelGo: A Cloud-Powered Real-Time Travel Booking Platform Using AWS' lab details. The table includes:

Platform	Lab	Duration	Difficulty	Progress
AWS	1	2 hour(s) 0 minute(s)	Expert	AWS

Below the table, the 'Overview' section describes TravelGo as a cloud-based travel booking platform. The 'Skills' section lists EC2, IAM, Database on AWS, and Monitoring and Observability. The 'Lab' section shows a summary for 'AWS Final Deployment' with 3/5 tasks validated, marked as 'Attempted'. It also shows the difficulty level as 'easy', 5 tasks, and a duration of 180 mins.

- Activity 1.2: Open the AWS Management Console

The screenshot shows a web browser window with multiple tabs open at the top. The active tab is titled "AWS Final Deployment" and displays the following content:

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

AWS - TravelGo: A Cloud-Powered Real-Time Travel Booking Platform Using AWS Allotted Duration: 02:00:00 HRS Duration Left: 01:12:31 HRS ⓘ

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

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 Search web &... 17:04 01-07-2025

2. 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 "**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.

The screenshot shows the AWS Management Console interface for DynamoDB. On the left, there's a navigation sidebar with links for Dashboard, Tables (which is selected), Explore items, PartQL editor, Backups, Exports to S3, Integrations, Reserved capacity, and Settings. Below that is a DAX section with Clusters, Subnet groups, Parameter groups, and Events. The main content area is titled "Tables (3) info". It features a search bar and filters for Name, Status, Partition key, Sort key, Indexes, Replication Regions, Deletion protection, Favorite, and Read capacity mode. A table lists three tables: "bookings" (Active, user_email (S), booking_date (S)), "trains" (Active, train_number (S), date (S)), and "TravelGo_users" (Active, email (S)). Each row has "Actions" and "Delete" buttons. At the top right, there's a "Create table" button. A feedback survey banner at the top says "Share your feedback on Amazon DynamoDB" and "Your Feedback is an important part of helping us provide a better customer experience. Take this short survey to let us know how we're doing." The bottom of the page includes CloudShell, Feedback, and standard AWS footer links.

Activity 2.2: Configure Attributes for User Data and Booking Requests

The screenshot shows the "Create table" wizard in the AWS Management Console. The first step, "Table details", asks for a "Table name" (set to "bookings1") and specifies it must be between 3 and 255 characters. The second step, "Partition key", defines "user_email" as the primary key type "String". The third step, "Sort key - optional", defines "Booking_id" as the sort key type "String". The fourth step, "Table settings", offers "Default settings" (selected) or "Customize settings". Under "Default table settings", it says these are the default settings for the new table and can be changed later. A table shows settings like "Setting" (e.g., "Provisioned Throughput") and "Value" (e.g., "100"). The bottom of the page includes CloudShell, Feedback, and standard AWS footer links.

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

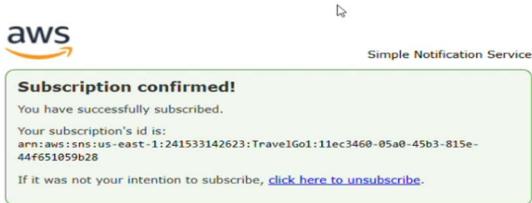
Activity 3.1: Create SNS topics.

The screenshot shows the AWS SNS Topics page. A blue banner at the top indicates a new feature: "Amazon SNS now supports High Throughput FIFO topics. Learn more". The main section displays a topic named "TravelGo". The "Details" panel shows the Name as "TravelGo", ARN as "arn:aws:sns:us-east-1:418272775181:TravelGo", and Type as "Standard". The "Subscriptions" tab is selected, showing one subscription entry:

ID	Endpoint	Status	Protocol
b7d6f5bc-7710-49de-97bc-ddecff5fd023	228x1a05e1@khitguntur.ac.in	Confirmed	EMAIL

Activity 3.2: Subscribe users to SNS email notifications

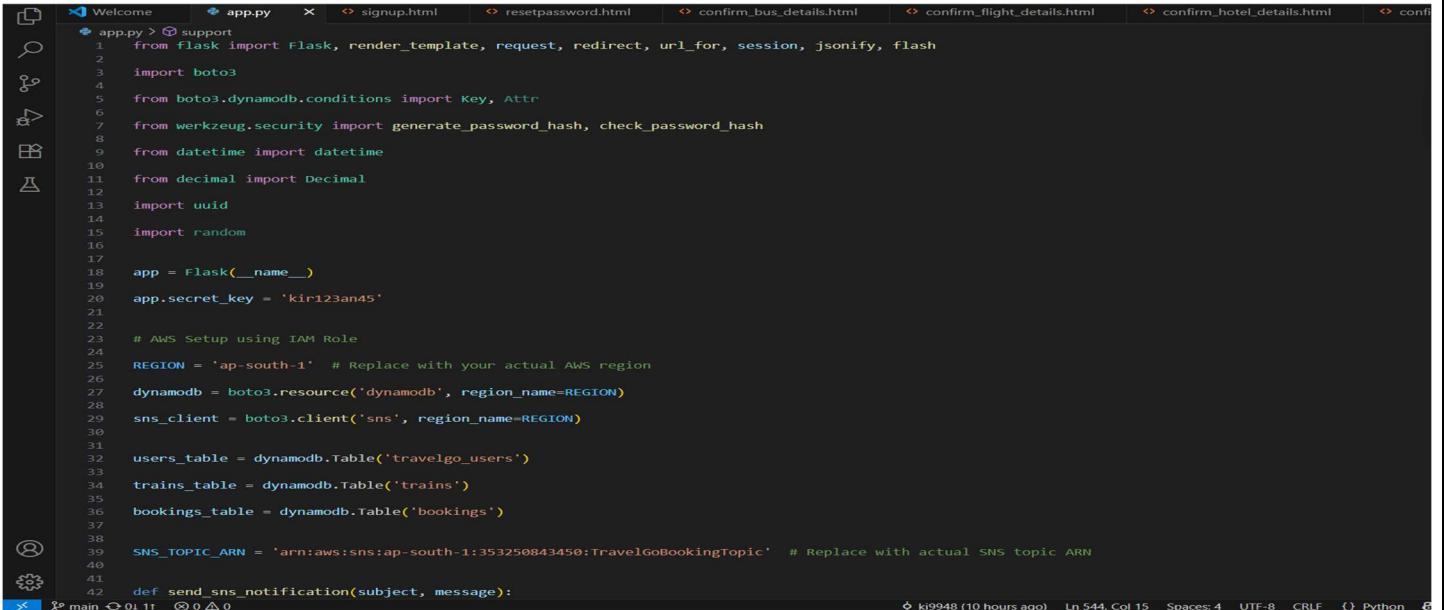
The screenshot shows the "Create subscription" page. A blue banner at the top indicates a new feature: "Amazon SNS now supports High Throughput FIFO topics. Learn more". The "Details" section includes fields for "Topic ARN" (set to "arn:aws:sns:us-east-1:418272775181:TravelGo"), "Protocol" (set to "Email"), and "Endpoint" (set to "228x1a05e1@khitguntur.ac.in"). Below these, a note says "After your subscription is created, you must confirm it." The "Subscription filter policy - optional" section notes that this policy filters messages. The "Redrive policy (dead-letter queue) - optional" section notes that undeliverable messages are sent to a dead-letter queue. At the bottom right are "Cancel" and "Create subscription" buttons.



4. Backend Development and Application Setup

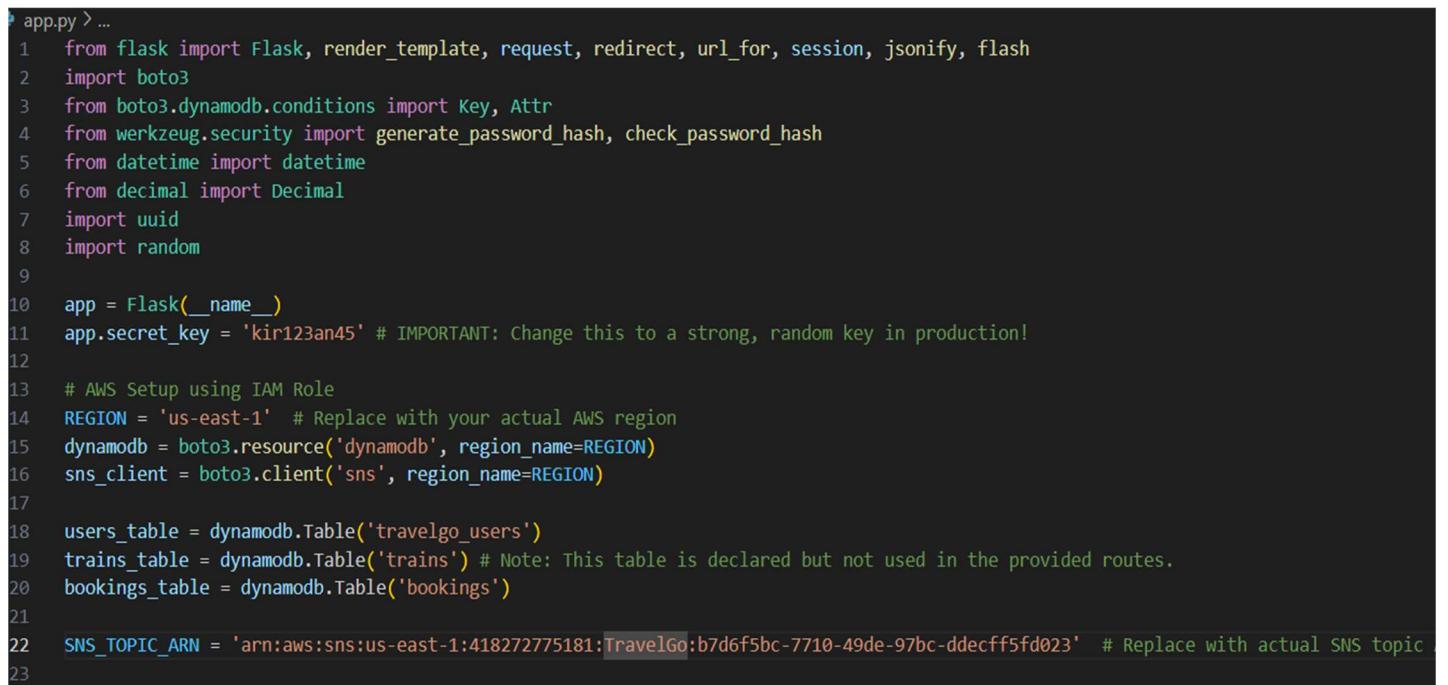
In the Backend Development and Application Setup, use Flask to build the web application and integrate AWS services via the boto3 library. Set up routes for user registration, login, book requests, and page navigation. Connect to DynamoDB to store user and request data, and configure AWS SNS to send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

Activity 4.1:Develop the Backend Using Flask.



```
>Welcome app.py signup.html resetpassword.html confirm_bus_details.html confirm_flight_details.html confirm_hotel_details.html config.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'
12
13 # AWS Setup using IAM Role
14 REGION = 'ap-south-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')
20 bookings_table = dynamodb.Table('bookings')
21
22 SNS_TOPIC_ARN = 'arn:aws:sns:ap-south-1:353250843450:TravelGoBookingTopic' # Replace with actual SNS topic ARN
23
24 def send sns_notification(subject, message):
25     pass
```

Activity 4.2: Integrate AWS Services Using boto3



```
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
```

Code for SNS

```
# This function is duplicated in the original code, removing the duplicate.
def send_sns_notification(subject, message):
    try:
        sns_client.publish(
            TopicArn=SNS_TOPIC_ARN,
            Subject=subject,
            Message=message
        )
    except Exception as e:
        print(f"SNS Error: Could not send notification - {e}")
        # Optionally, flash an error message to the user or log it more robustly.

# Routes
@app.route('/')
```

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

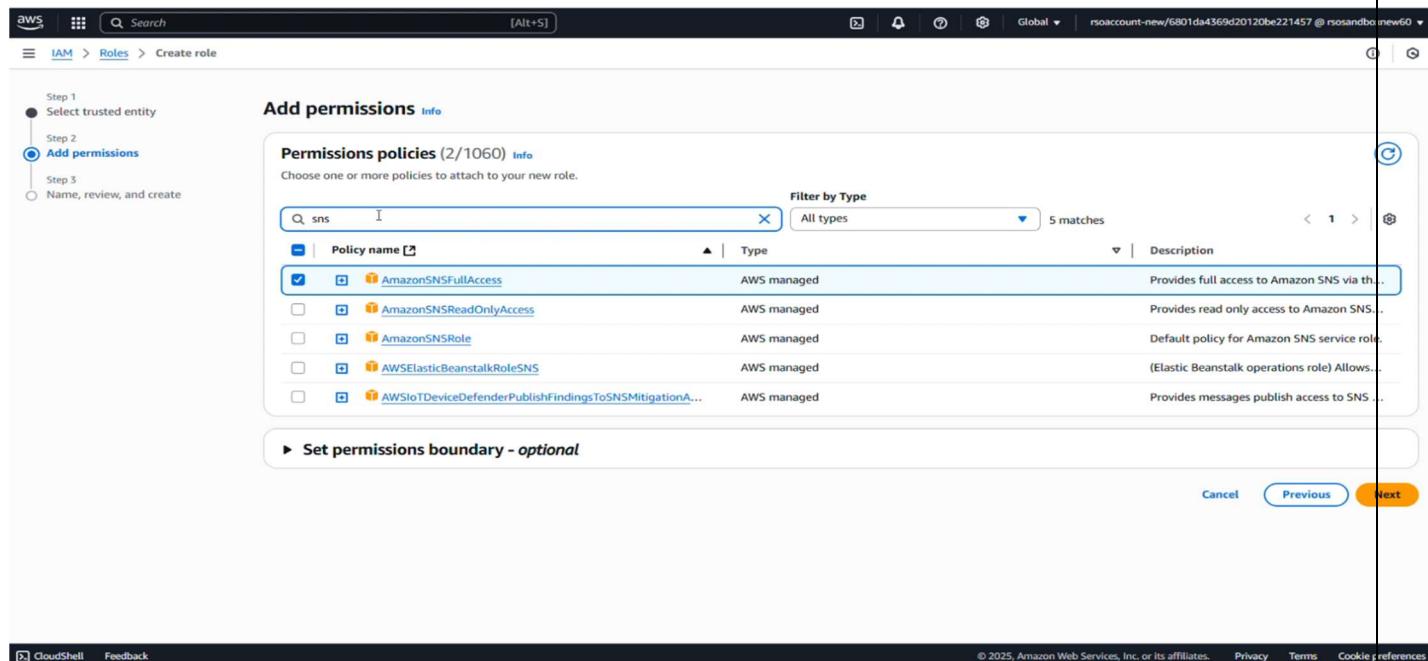
```
1
2 @app.route('/cancel_booking', methods=['POST'])
3 def cancel_booking():
4     if 'email' not in session:
5         return redirect(url_for('login'))
6
7     booking_id = request.form.get('booking_id')
8     user_email = session['email']
9     booking_date = request.form.get('booking_date') #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

5. IAM Role Setup

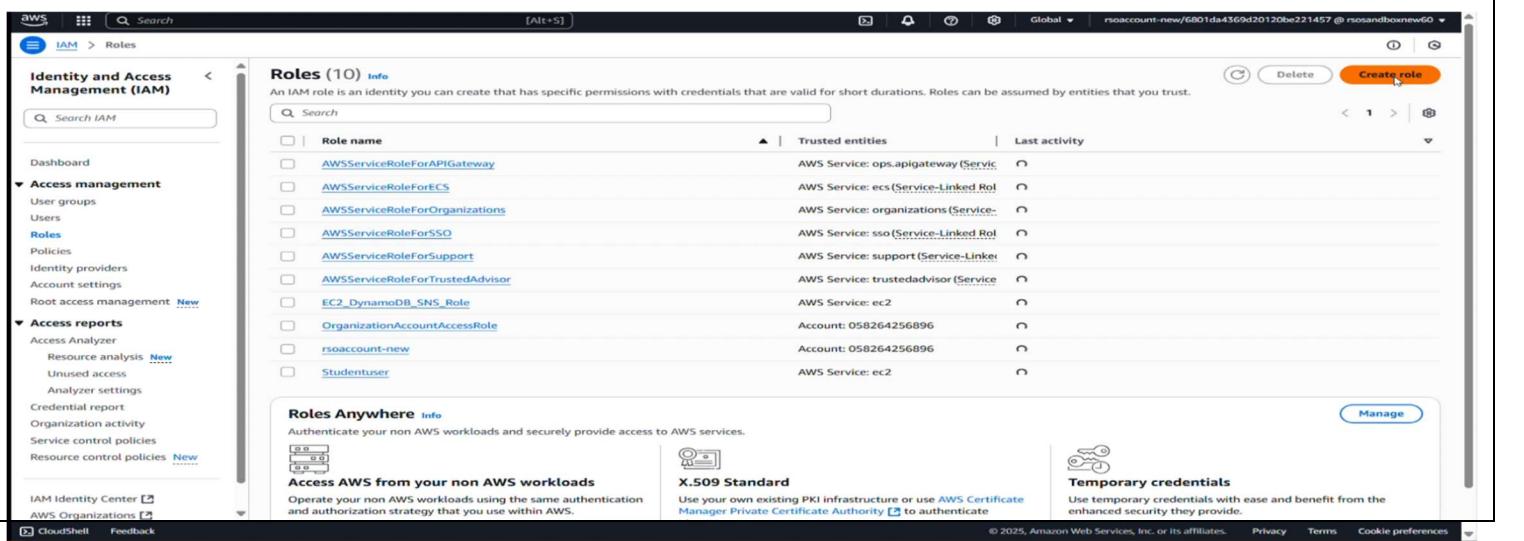
For IAM Role Setup, go to the AWS Console and create a new IAM Role for your EC2 instance. Attach AmazonDynamoDBFullAccess and AmazonSNSFullAccess policies to allow interaction with DynamoDB and SNS. Assign this role to your EC2 instance during or after launch. This ensures secure and authorized access 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 creating a new IAM role. On the left, a navigation pane lists three steps: 'Select trusted entity', 'Add permissions' (which is selected), and 'Name, review, and create'. The main area is titled 'Add permissions' and shows a search bar with 'sns' and a dropdown menu for 'Policy name'. A list of policies is displayed, with 'AmazonSNSFullAccess' checked. Other visible policies include 'AmazonSNSReadOnlyAccess', 'AmazonSNSRole', 'AWSLambdaBeanstalkRoleSNS', and 'AWSIoTDeviceDefenderPublishFindingsToSNSSignatureVerification'. A note at the bottom says '▶ Set permissions boundary - optional'. At the bottom right are 'Cancel', 'Previous', and 'Next' buttons.



The screenshot shows the 'Roles' page in the AWS IAM console. The left sidebar includes sections for Identity and Access Management (IAM), Access management, Access reports, and various AWS services like CloudShell and Feedback. The main area displays a table of roles with columns for 'Role name', 'Trusted entities', and 'Last activity'. Roles listed include 'AWSServiceRoleForAPIGateway', 'AWSServiceRoleForEC2', 'AWSServiceRoleForOrganizations', 'AWSServiceRoleForSSO', 'AWSServiceRoleForSupport', 'AWSServiceRoleForTrustedAdvisor', 'EC2_DynamoDB_SNS_Role', 'OrganizationAccountAccessRole', 'rsoaccount-new', and 'Studentuser'. A 'Create role' button is located at the top right. At the bottom, there are sections for 'Roles Anywhere', 'Access AWS from your non AWS workloads', 'X.509 Standard', and 'Temporary credentials'.

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.

[IAM](#) > [Roles](#) > Create role

AC L O I T : [
7 "sts:AssumeRole"
8],
9 "Principal": {
10 "Service": [
11 "ec2.amazonaws.com"
12]
13 }
14 }
15 }
16]

Step 2: Add permissions

[Edit](#)

Permissions policy summary

Policy name	Type	Attached as
AmazonDynamoDBFullAccess	AWS managed	Permissions policy
AmazonEC2FullAccess	AWS managed	Permissions policy
AmazonSNSFullAccess	AWS managed	Permissions policy

Step 3: Add tags

Add tags - optional Info

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

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

[Cancel](#) [Previous](#) [Create role](#)

The screenshot shows the 'Launch an instance' wizard on the AWS EC2 service. The current step is 'Name and tags'. A 'Name' field contains 'TravelGo1'. Below it is a 'Virtual server type (instance type)' section with a dropdown menu set to 'Select'. To the right, there's a summary panel with a 'Free tier' note about usage and costs.

create a key pair:

The screenshot shows the 'Create key pair' dialog box overlaid on the EC2 launch wizard. In the dialog, the 'Key pair name' field is filled with 'travel119'. Under 'Key pair type', the 'RSA' option is selected. Under 'Private key file format', the '.pem' option is selected. A warning message at the bottom of the dialog states: '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.' A 'Create key pair' button is at the bottom right of the dialog.

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 navigation path is "EC2 > Security Groups > sg-0a6c8b20300891292 - launch-wizard-3 > Edit inbound rules". The top bar includes the AWS logo, search bar, and account information: "United States (N. Virginia) rsoaccount-new/6801da4369d20120be221457 @ rsosandboxnew60".

Inbound 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

Add rule

Warning: 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.

Buttons: Cancel, Preview changes, Save rules.

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

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:

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
```

```

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
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

```

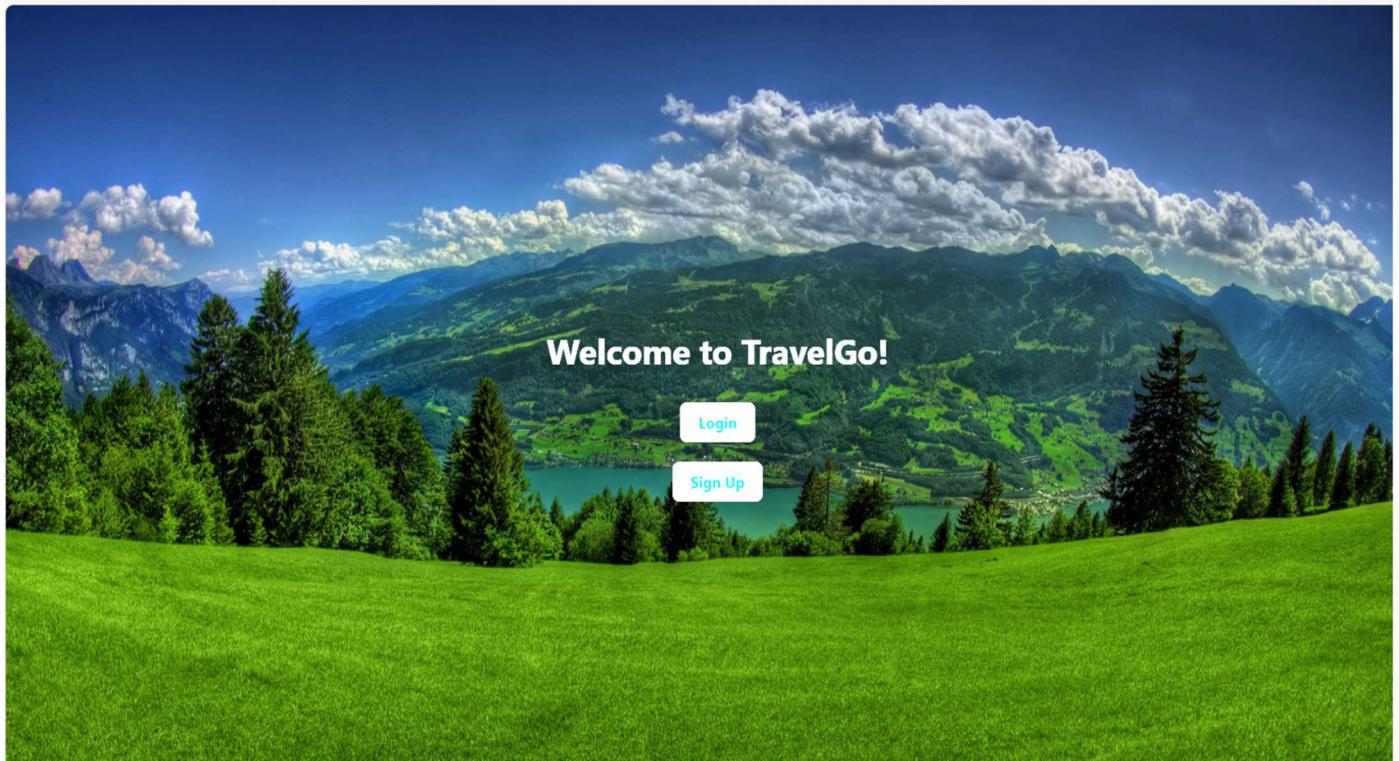
The screenshot shows a GitHub repository named 'testupload'. The repository is public and has 1 branch and 0 tags. The file list includes 'app.py' (updated), 'README.md' (first commit), 'venv_new' (added full project files), 'venv' (added full project files), 'templates' (added full project files), and 'static' (added full project files). A pull request by user 'ki9948' titled 'Update app.py' is visible. On the right, there's a 'Code' dropdown menu with 'Local' and 'Codespaces' tabs, a 'Clone' section with HTTPS, SSH, and GitHub CLI options, and a URL 'https://github.com/ki9948/testupload.git'. Below it are 'Clone using the web URL.', 'Open with GitHub Desktop', and 'Download ZIP' buttons. The repository has 0 forks, 0 stars, 0 releases published, and 0 watching.

```

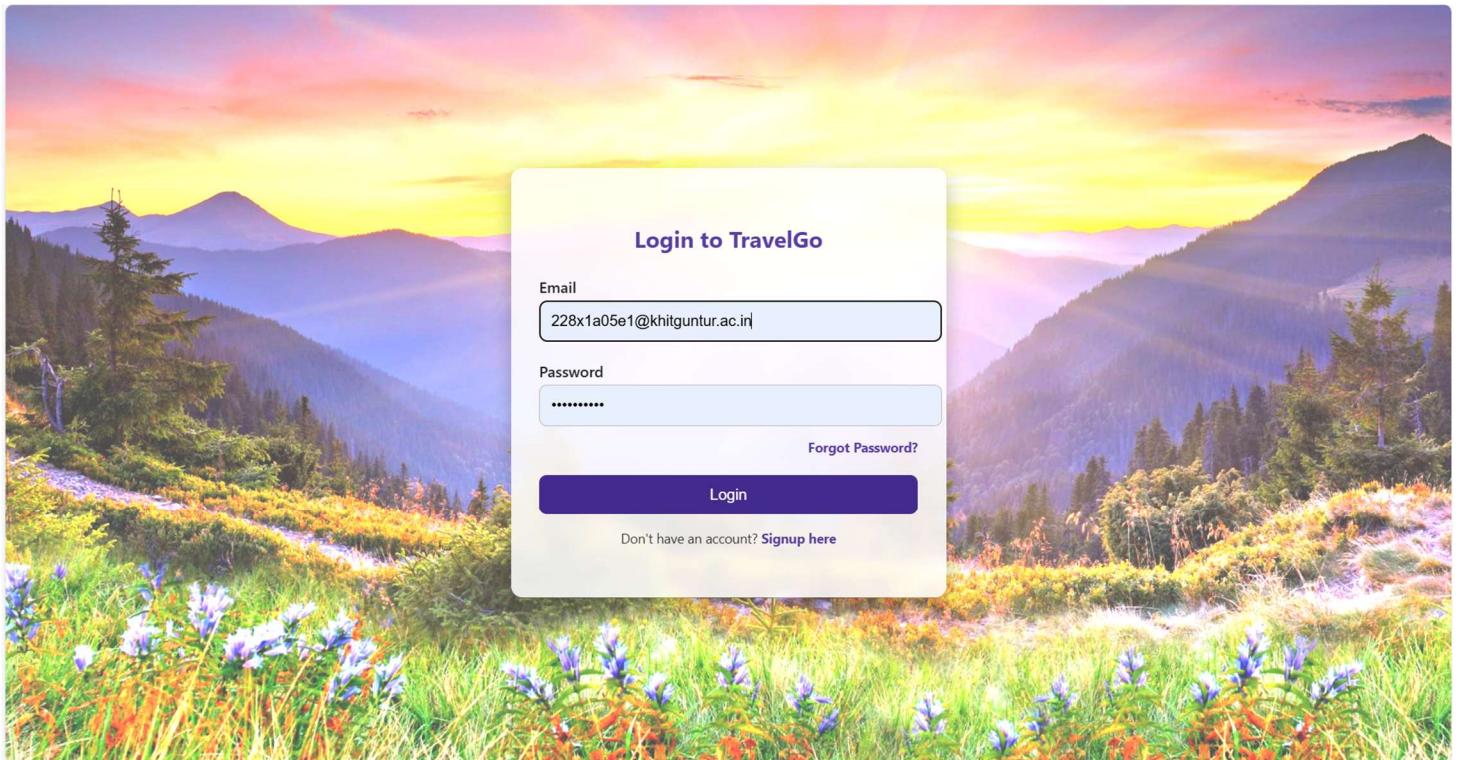
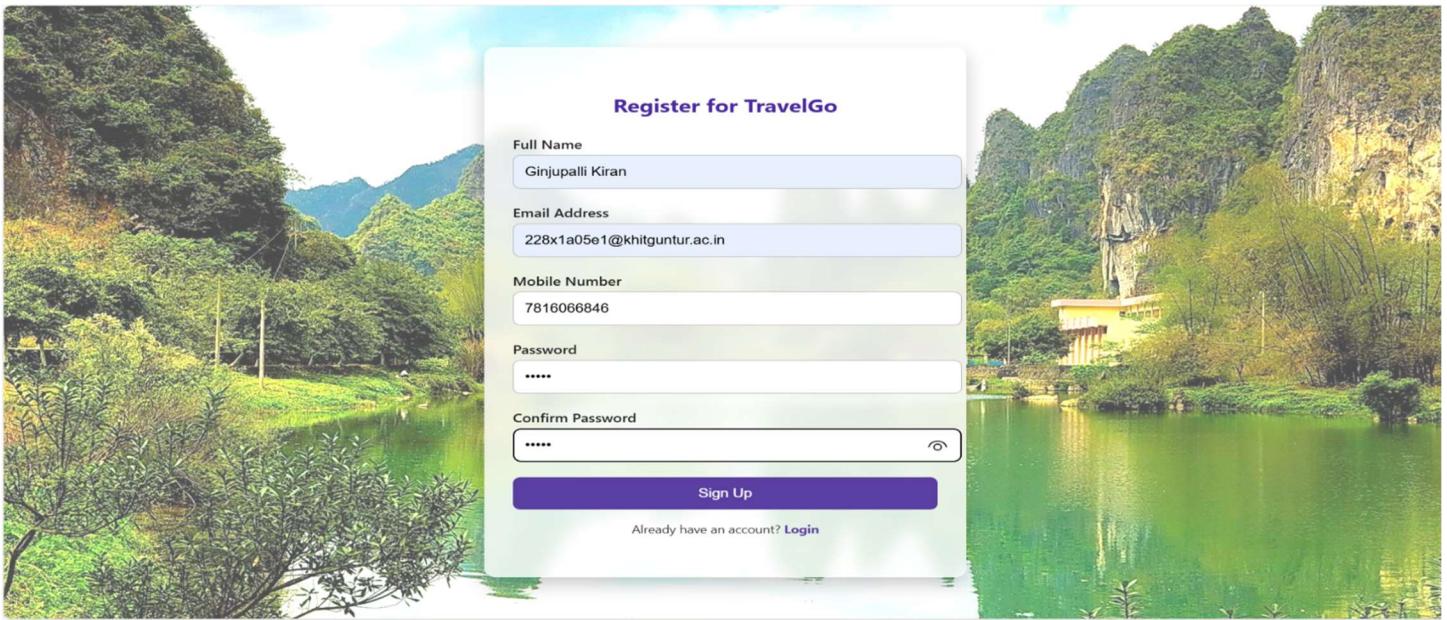
Collecting itsdangerous>=2.2.0
  Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Collecting blinker>=1.9.0
  Downloading blinker-1.9.0-py3-none-any.whl (8.5 kB)
Collecting werkzeug>=3.1.0
  Downloading werkzeug-3.1.3-py3-none-any.whl (224 kB)
|██████████| 224 kB 56.0 MB/s
Collecting zipp>=3.20
  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
152.50.200.20 - [01/Jul/2025:05:40:24] "GET / HTTP/1.1" 200

```

Index page:



Register page:



Login page:

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

Home Code:

Home page:

```
 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
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
```

Dashboard code:

Dashboard page:

Welcome, 228x1a05e1@khitguntur.ac.in!

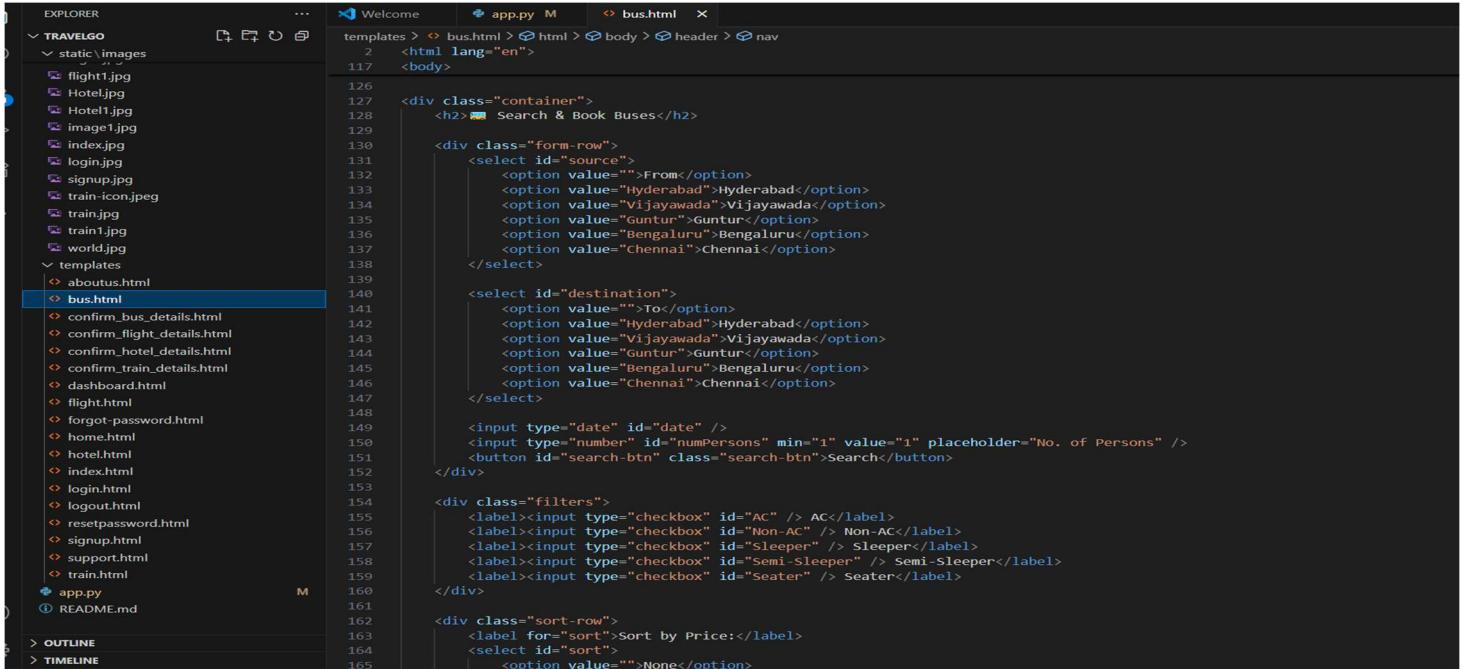


YOUR BOOKINGS

You have no bookings yet. Start by searching for a trip!

© 2025 TravelGo. All rights reserved.

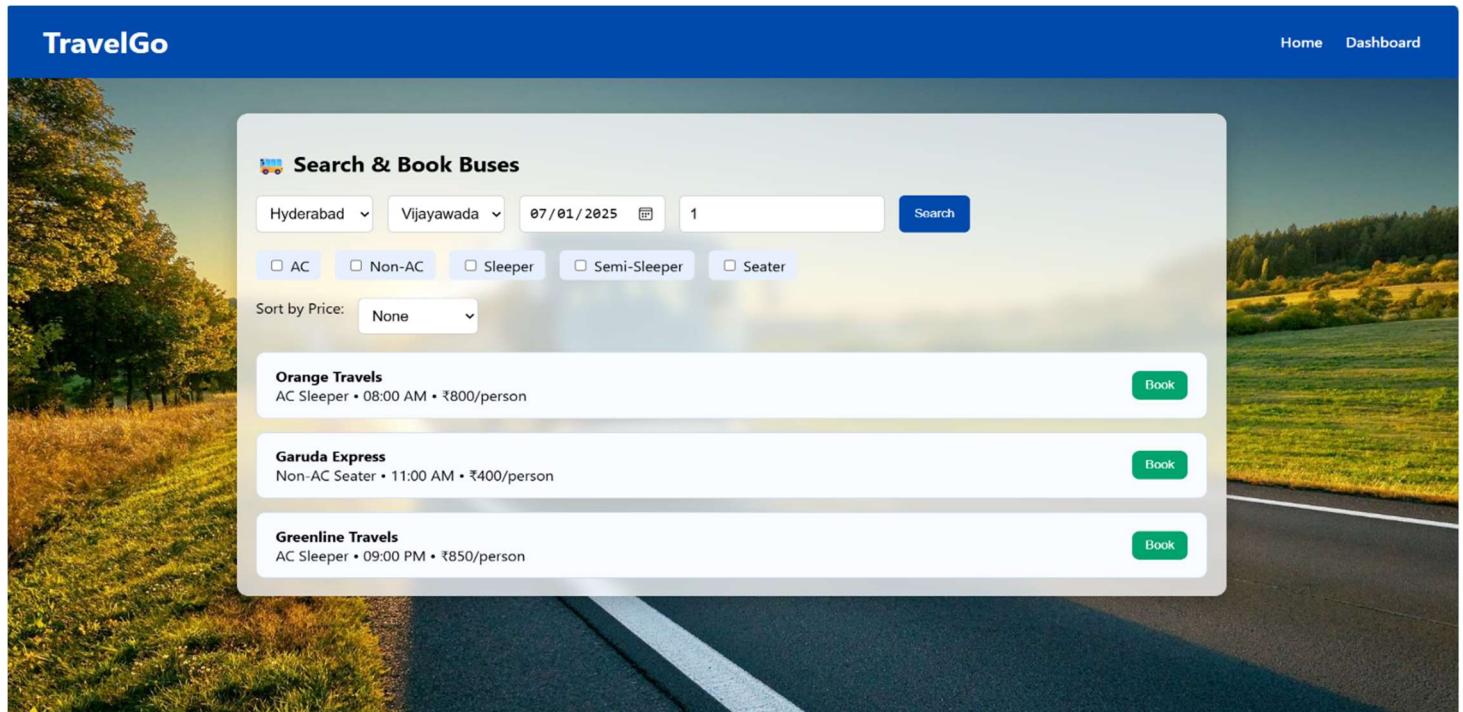
Bus Booking code:



```
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">
190         ...
191       </div>
192       <div id="seat-map" class="seat-map"></div>
193       <input type="hidden" id="selectedSeatsInput" />
194
195       <p><strong>Bus Name:</strong> {{ booking.name }}</p>
196       <p><strong>Route:</strong> {{ booking.source }} to {{ booking.destination }}</p>
197       <p><strong>Date:</strong> {{ booking.travel_date }}</p>
198       <p><strong>Time:</strong> {{ booking.time }}</p>
199       <p><strong>Type:</strong> {{ booking.type }}</p>
200       <p><strong>Passengers:</strong> {{ booking.num_persons }}</p>
201       <p><strong>Selected Seats:</strong>
202         <span id="selectedSeatsDisplay">None</span>
203       </p>
204       <p class="total-price">Total Price: ₹{{ ":.2f".format(booking.total_price) }}</p>
205
206       <div class="buttons-container">
207         <button id="confirmBookingBtn">Confirm Booking</button>
208         <a href="/bus" class="btn btn-secondary">Go Back to Search</a>
209       </div>
210     </div>
211
212     <script>
213       const seatLabels = [
214         "S1", "S2", "", "S3", "S4",
215         "S5", "S6", "", "S7", "S8",
216         "S9", "S10", "", "S11", "S12",
217         "S13", "S14", "", "S15", "S16",
218         "S17", "S18", "", "S19", "S20",
219         "S21", "S22", "", "S23", "S24",
220         "S25", "S26", "", "S27", "S28"
221       ];
222     </script>
223   </div>
224
225   <footer>
226     &copy; 2025 TravelGo. All rights reserved.
227   </footer>
228
229 </body>
230 </html>
```

Bus confirm booking page:

Booking Details:

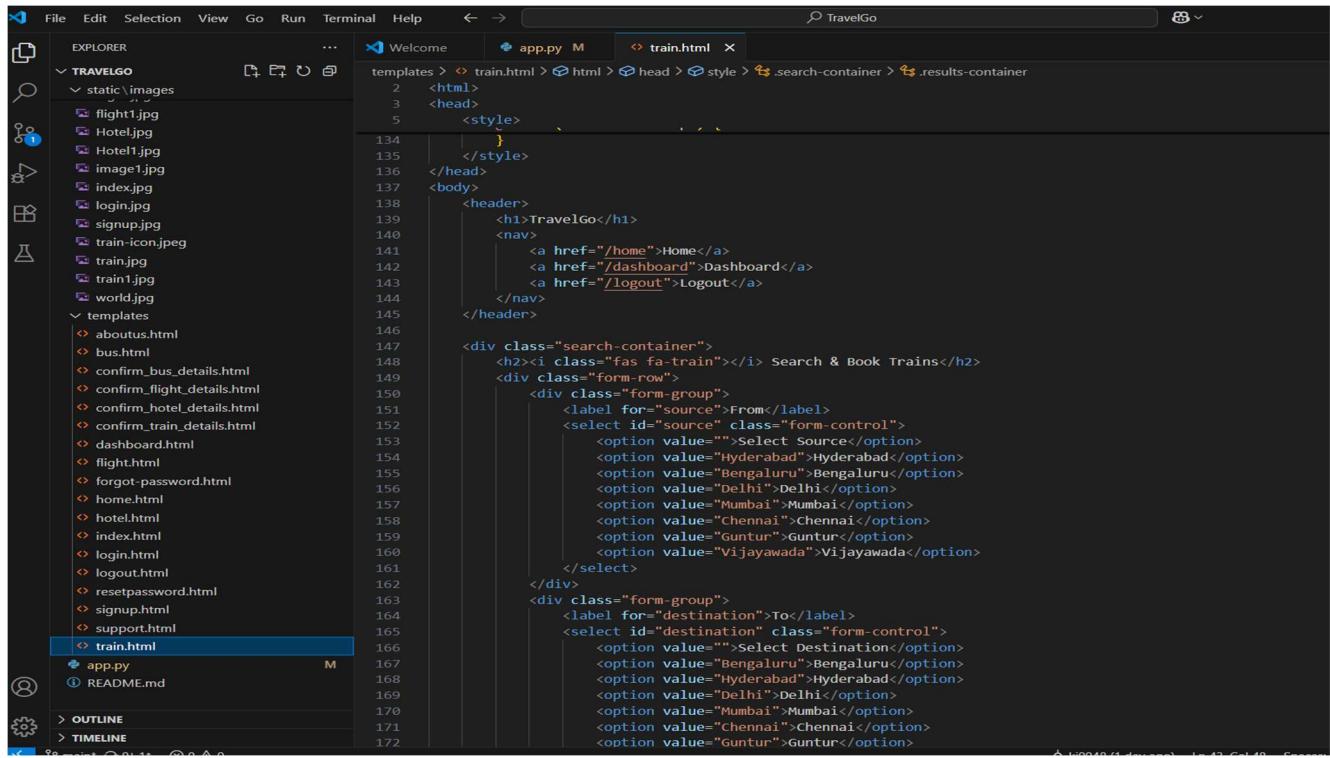
Select Your Seats:

Available Selected Booked Women Only

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

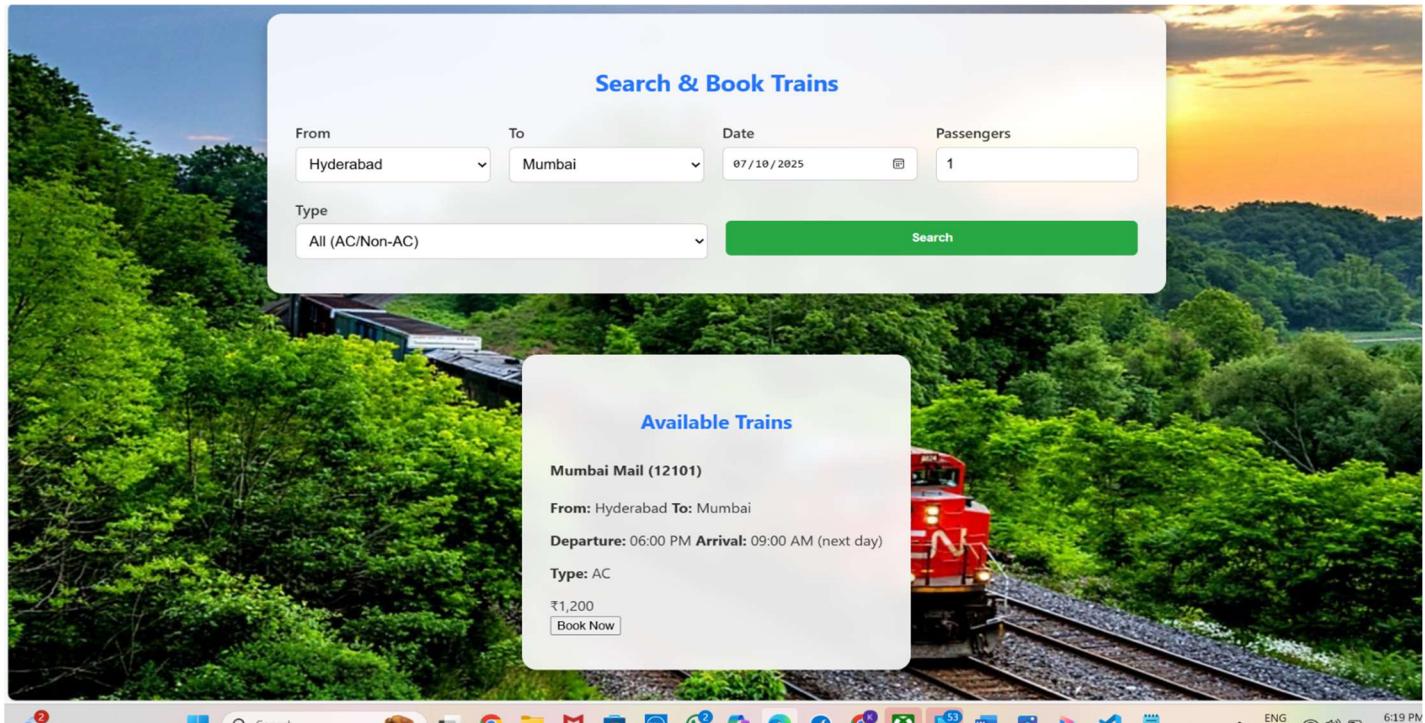
Bus Name: Orange Travels
Route: Hyderabad to Vijayawada
Date: 2025-07-01
Time: 08:00 AM
Type: AC Sleeper

Train code :

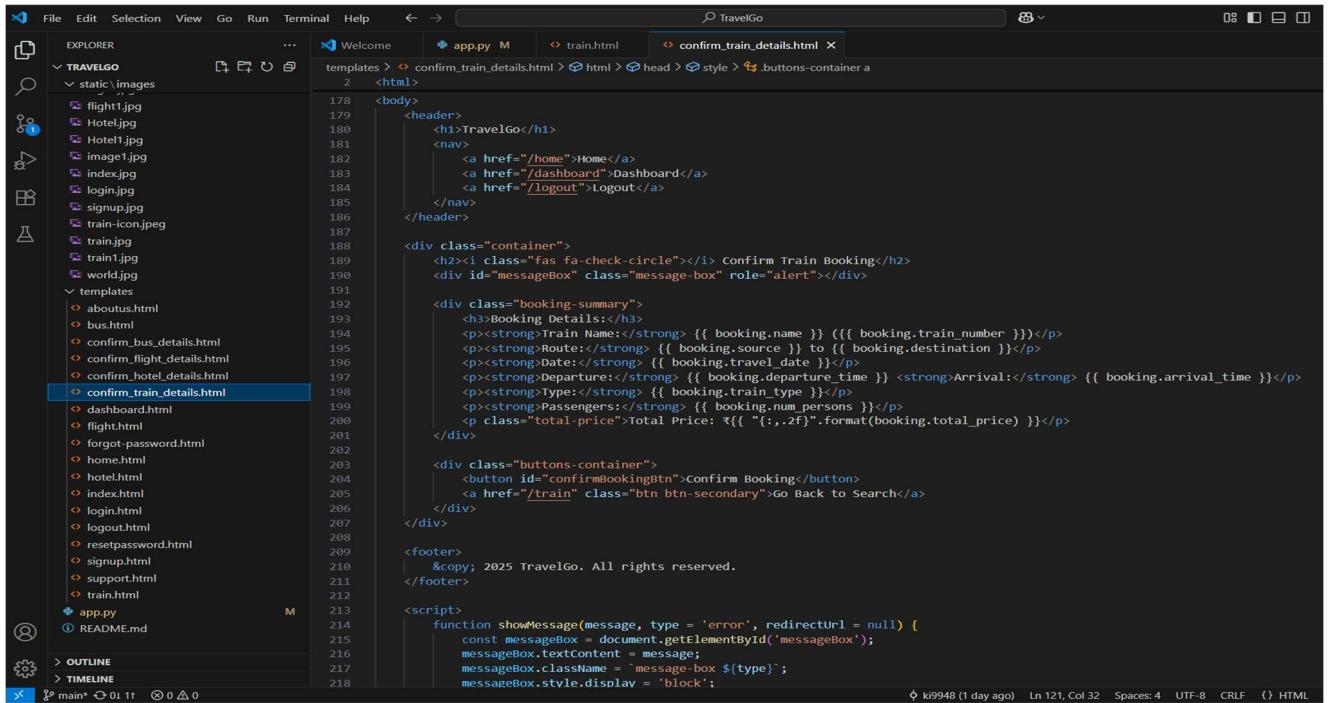


```
File Edit Selection View Go Run Terminal Help ← → Welcome app.py M train.html
EXPLORER templates > train.html > html > head > style > .search-container > .results-container
static/images
  flight1.jpg
  Hotel.jpg
  Hotel1.jpg
  image1.jpg
  index.jpg
  login.jpg
  signup.jpg
  train-icon.jpeg
  train.jpg
  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:



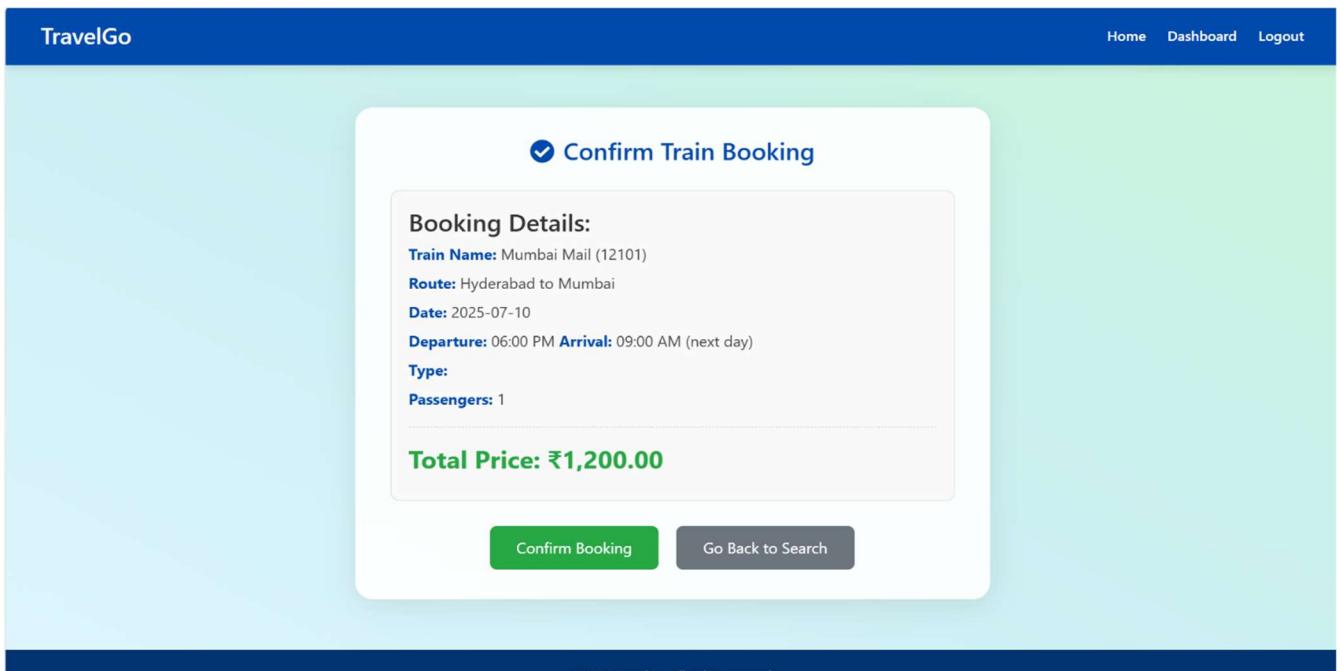
Train confirm booking code:



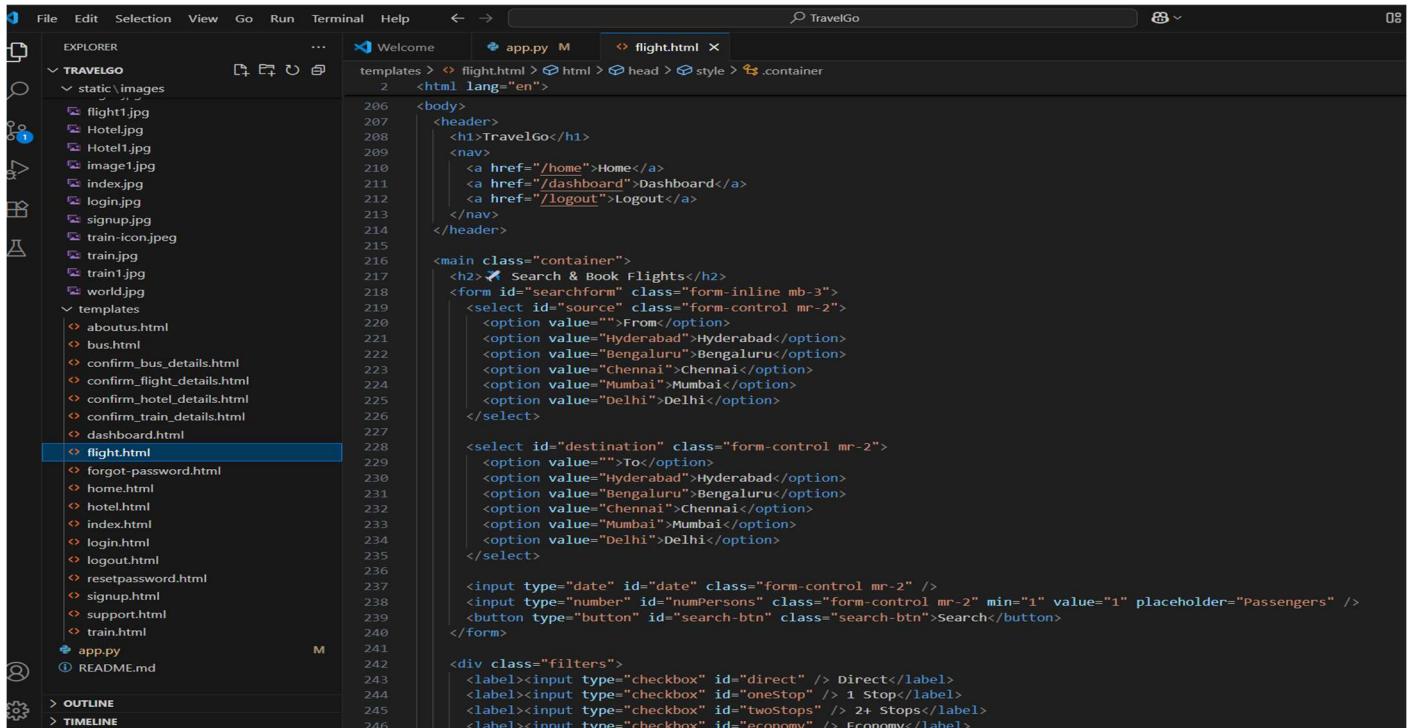
The screenshot shows a code editor interface with the following details:

- File Path:** templates > confirm_train_details.html
- Code Editor:** Visual Studio Code
- File Explorer:** Shows the project structure for 'TRAVELGO' with files like 'flight1.jpg', 'Hotel.jpg', 'image1.jpg', 'index.jpg', 'login.jpg', 'signup.jpg', 'train-icon.jpg', 'train.jpg', 'train1.jpg', 'world.jpg', 'aboutus.html', 'bus.html', 'confirm_bus_details.html', 'confirm_flight_details.html', 'confirm_hotel_details.html', 'confirm_train_details.html' (highlighted in blue), 'dashboard.html', 'forgot-password.html', 'home.html', 'hotel.html', 'index.html', 'login.html', 'logout.html', 'resetpassword.html', 'signup.html', 'support.html', 'train.html', 'app.py', and 'README.md'.
- Code Content:** The code is an HTML template for a 'Confirm Train Booking' page. It includes a header with navigation links for Home, Dashboard, and Logout. The main content area displays booking details such as Train Name (Mumbai Mail (12101)), Route (Hyderabad to Mumbai), Date (2025-07-10), Departure (06:00 PM), Arrival (09:00 AM next day), Type (Passenger), and Passengers (1). A message box at the bottom displays the total price as ₹1,200.00. The footer contains a copyright notice for 2025 TravelGo.
- Status Bar:** Shows file statistics: 19948 (1 day ago), Line count: 121, Column count: 32, Spaces: 4, UTF-8, CRLF, and HTML.

Train confirm booking page:



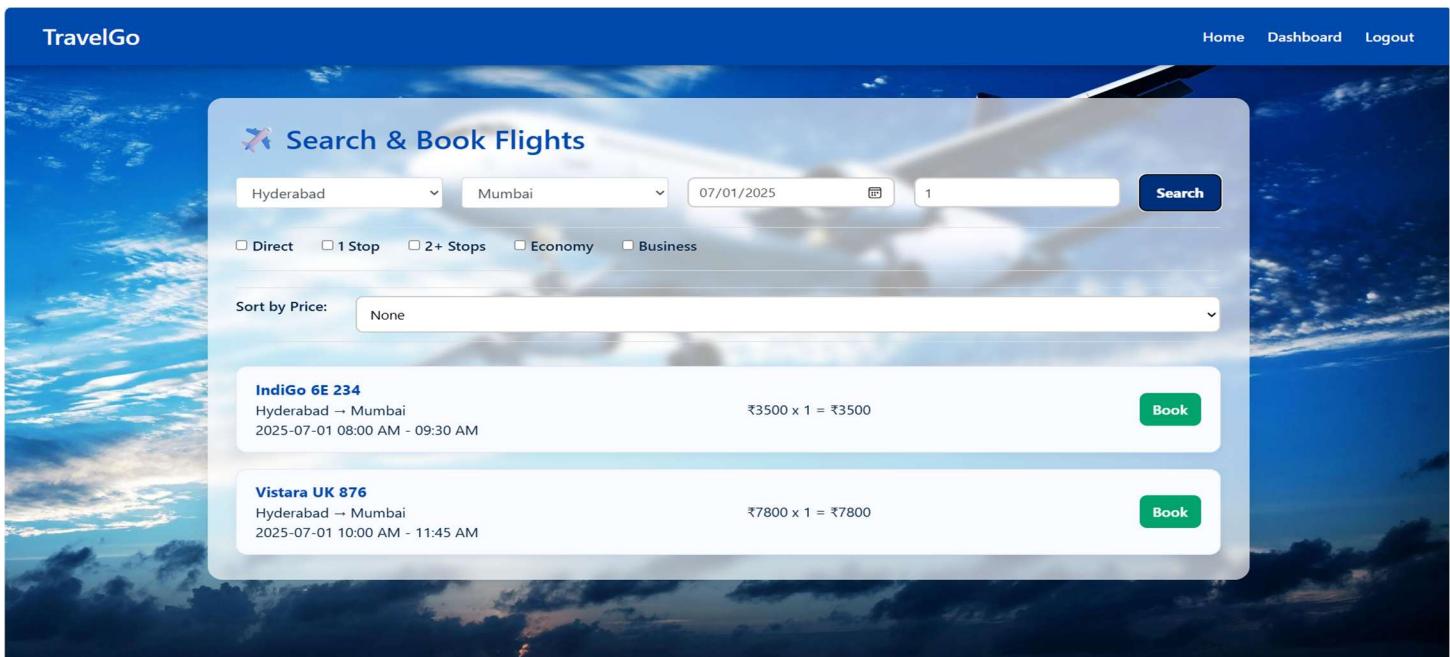
Flight code:



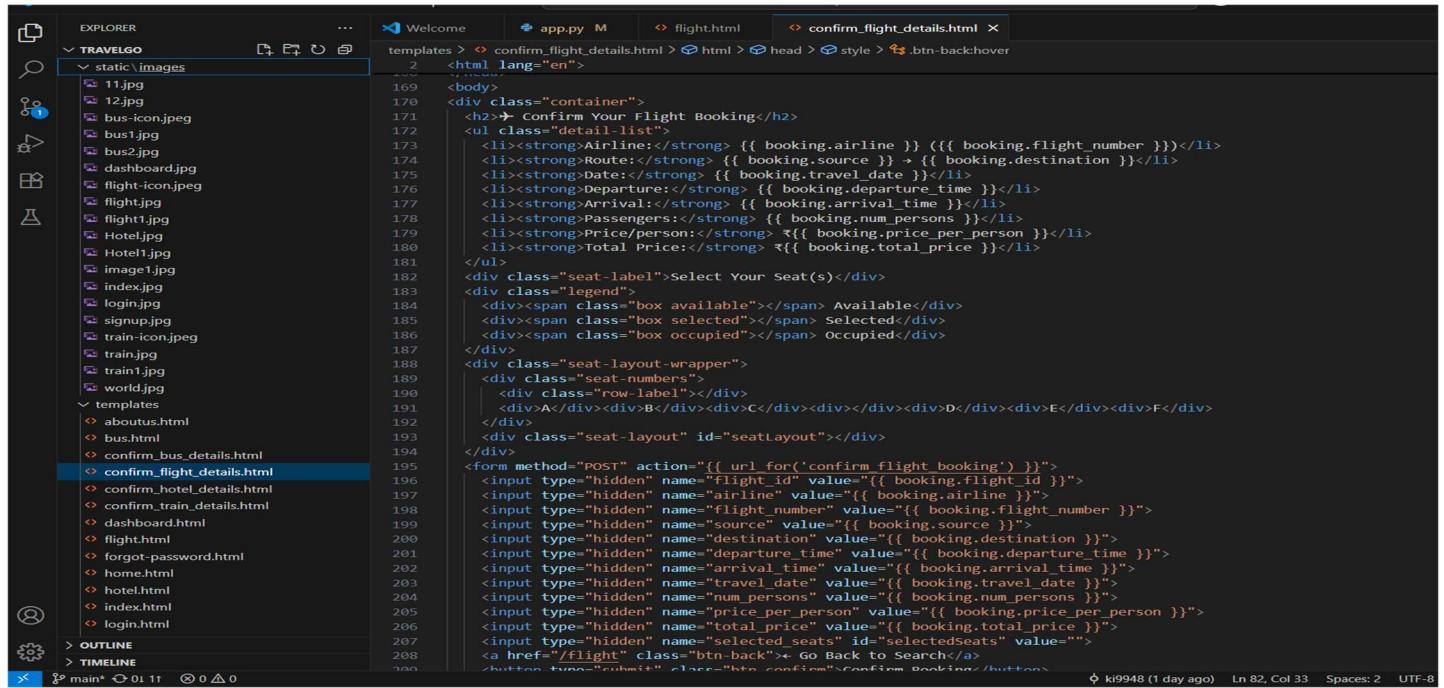
```
206 <body>
207   <header>
208     <h1>TravelGo</h1>
209     <nav>
210       <a href="/home">Home</a>
211       <a href="/dashboard">Dashboard</a>
212       <a href="/logout">Logout</a>
213     </nav>
214   </header>
215
216   <main class="container">
217     <h2>Search & Book Flights</h2>
218     <form id="searchform" class="form-inline mb-3">
219       <select id="source" class="form-control mr-2">
220         <option value="">From</option>
221         <option value="Hyderabad">Hyderabad</option>
222         <option value="Bengaluru">Bengaluru</option>
223         <option value="Chennai">Chennai</option>
224         <option value="Mumbai">Mumbai</option>
225         <option value="Delhi">Delhi</option>
226       </select>
227
228       <select id="destination" class="form-control mr-2">
229         <option value="">To:</option>
230         <option value="Hyderabad">Hyderabad</option>
231         <option value="Bengaluru">Bengaluru</option>
232         <option value="Chennai">Chennai</option>
233         <option value="Mumbai">Mumbai</option>
234         <option value="Delhi">Delhi</option>
235       </select>
236
237       <input type="date" id="date" class="form-control mr-2" />
238       <input type="number" id="numPersons" class="form-control mr-2" min="1" value="1" placeholder="Passengers" />
239       <button type="button" id="search-btn" class="search-btn">Search</button>
240     </form>
241
242     <div class="filters">
243       <label><input type="checkbox" id="direct" /> Direct</label>
244       <label><input type="checkbox" id="oneStop" /> 1 Stop</label>
245       <label><input type="checkbox" id="twoStops" /> 2+ Stops</label>
246       <label><input type="checkbox" id="economy" /> Economy</label>

```

Flight page:



Flight confirm booking code:

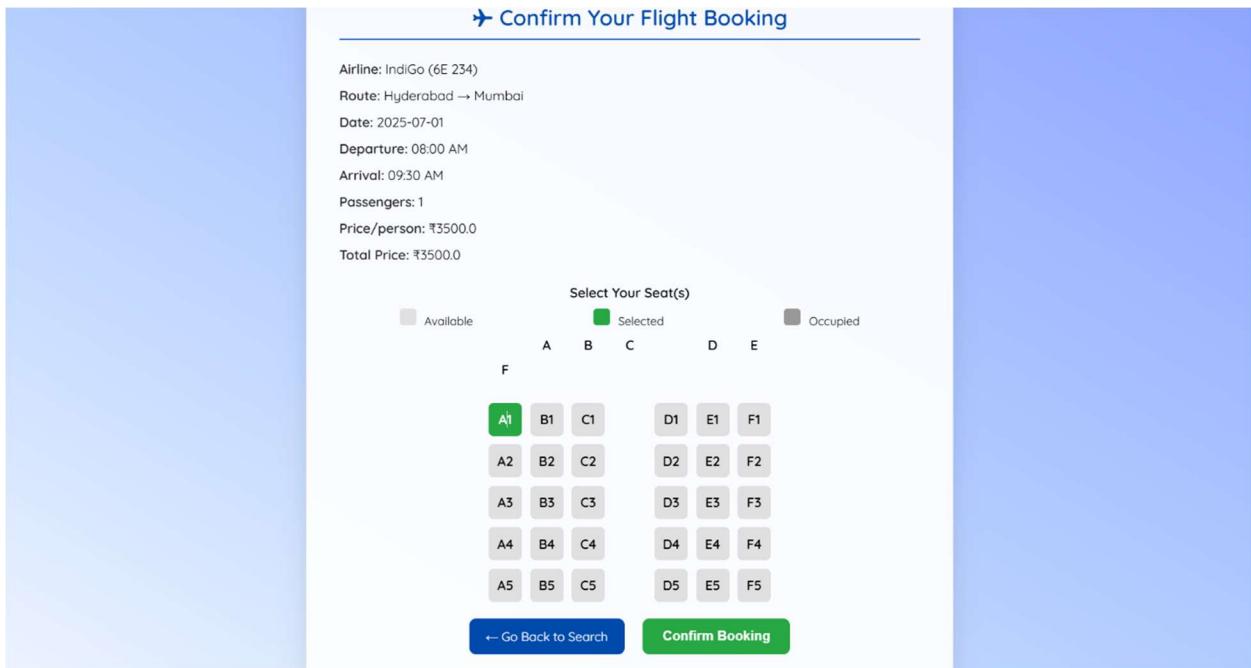


The screenshot shows a code editor interface with the following details:

- 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)

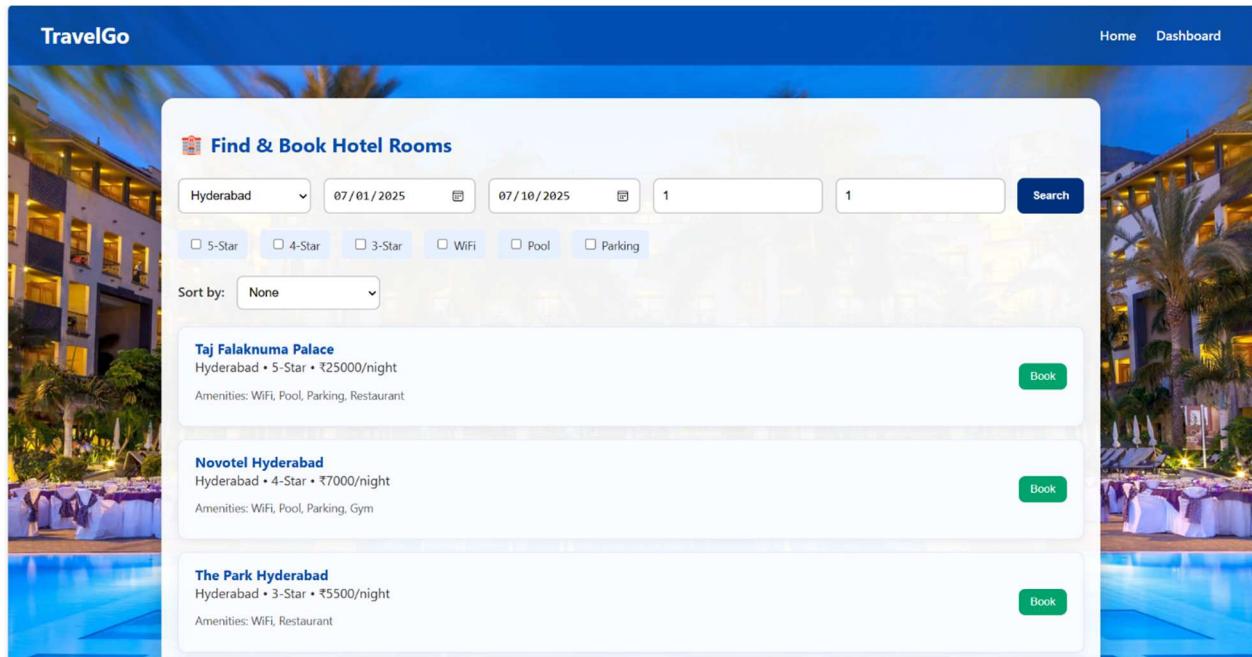
	A	B	C	D	E	F
Available	A1	B1	C1	D1	E1	F1
Selected	A2	B2	C2	D2	E2	F2
Occupied	A3	B3	C3	D3	E3	F3
	A4	B4	C4	D4	E4	F4
	A5	B5	C5	D5	E5	F5

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

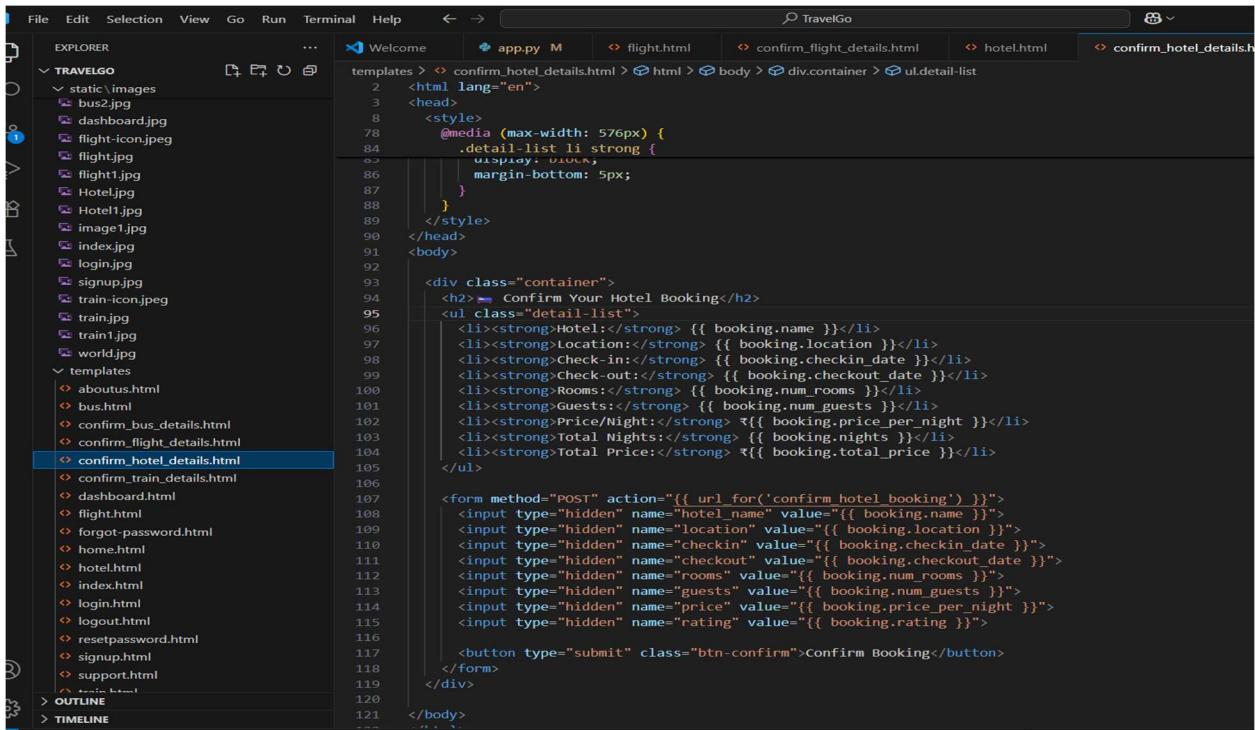
Hotel code:

```
<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:

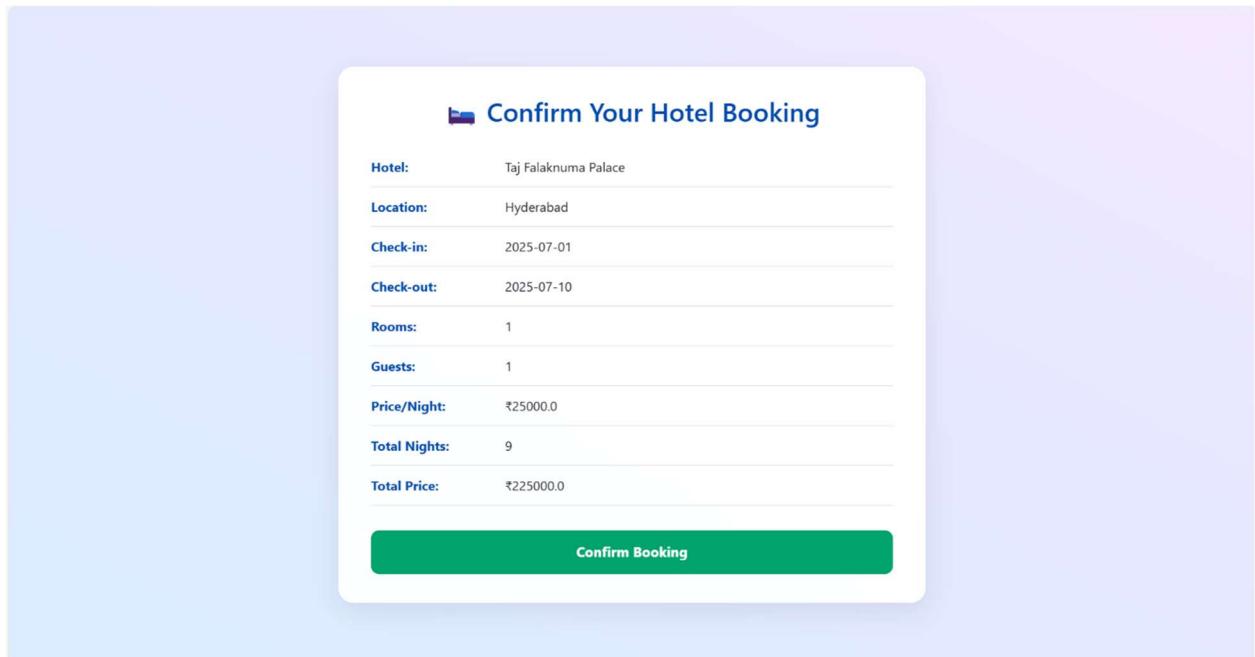


Hotel confirm booking code:



```
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:

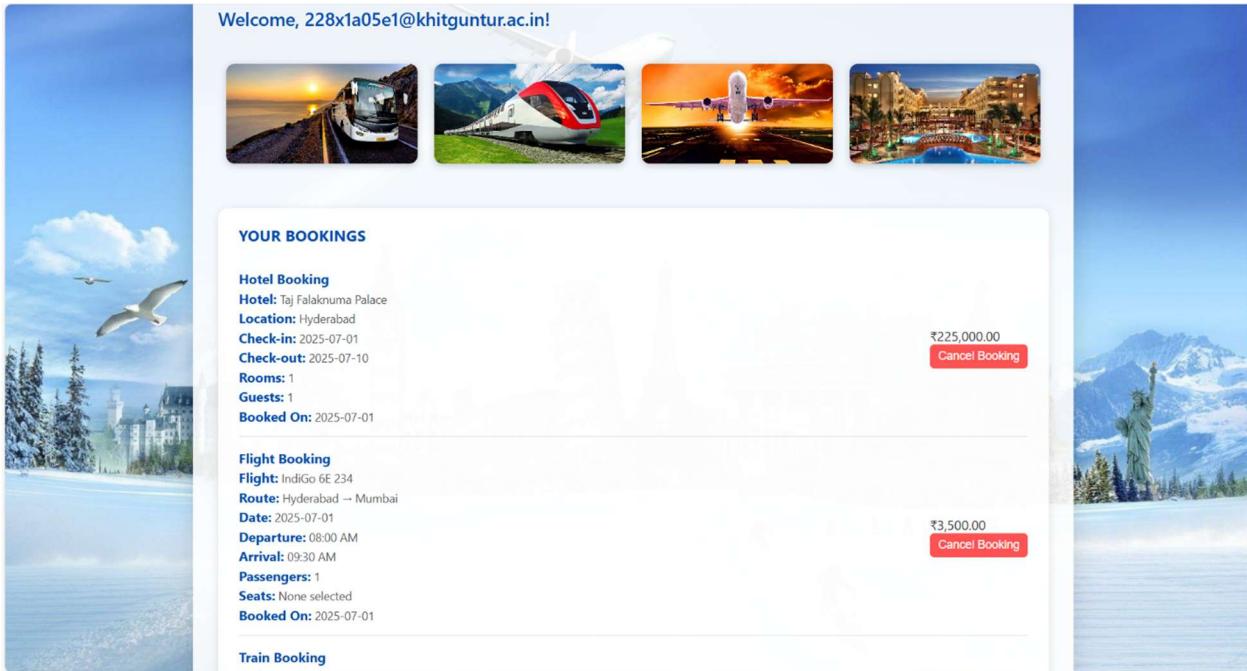


The screenshot shows a web application interface titled "Confirm Your Hotel Booking". The page lists the following booking details:

- 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

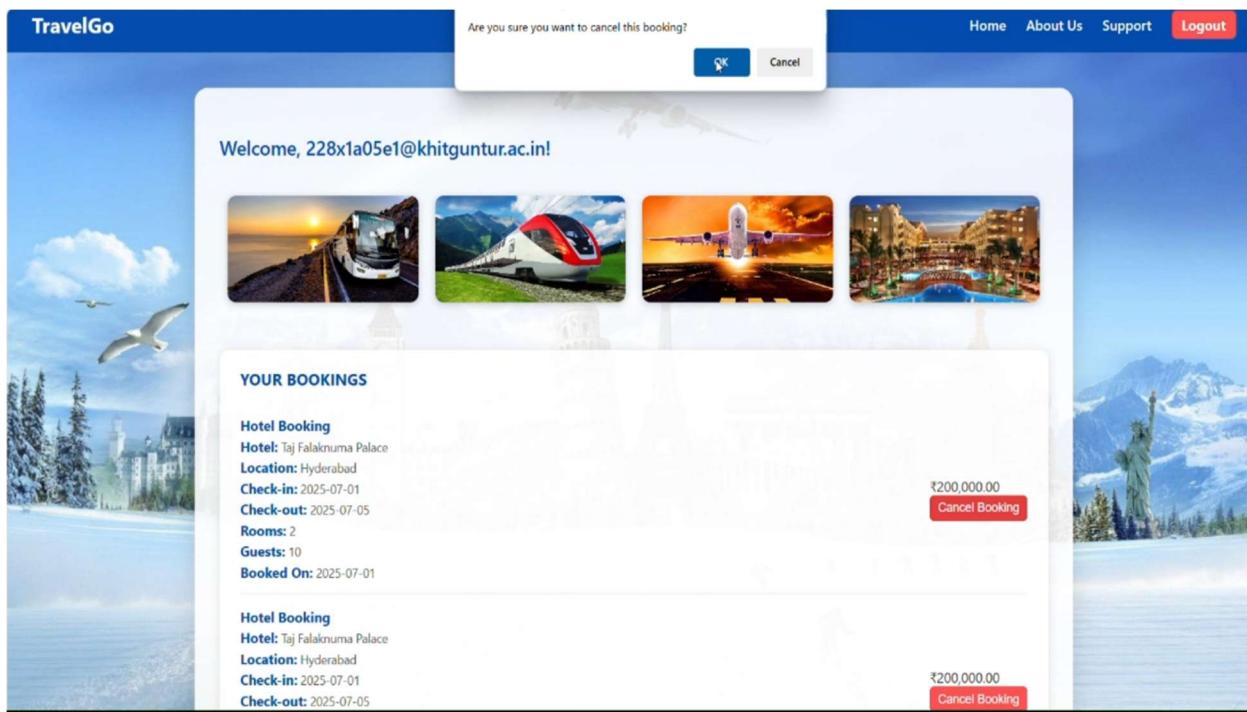
₹225,000.00
Cancel Booking

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 p



TravelGo

Are you sure you want to cancel this booking?

OK Cancel

Welcome, 228x1a05e1@khitguntur.ac.in!

YOUR BOOKINGS

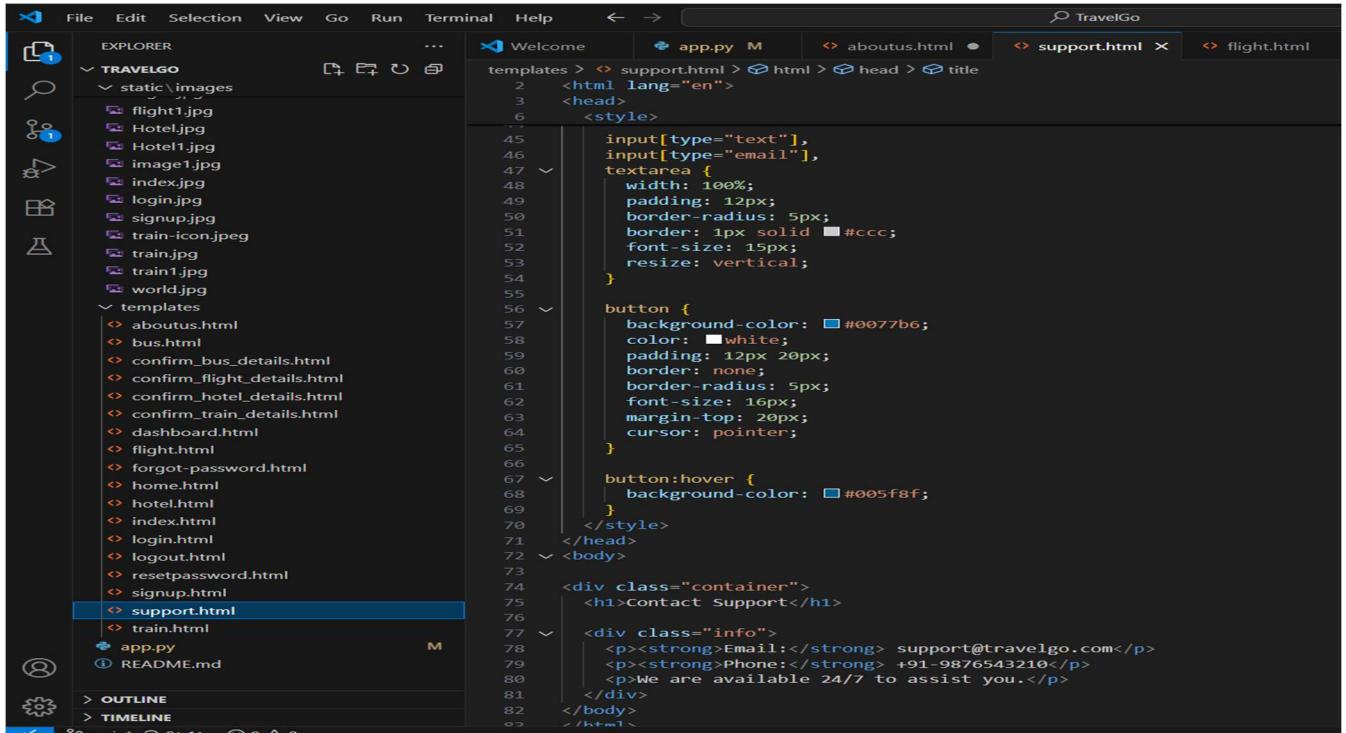
Hotel Booking
Hotel: Taj Falaknuma Palace
Location: Hyderabad
Check-in: 2025-07-01
Check-out: 2025-07-05
Rooms: 2
Guests: 10
Booked On: 2025-07-01

₹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

Support code:



The screenshot shows a code editor interface with a dark theme. On the left is a file explorer window titled 'EXPLORER' showing a directory structure for a project named 'TRAVELGO'. The 'templates' folder contains several HTML files, including 'support.html', which is currently selected and highlighted with a blue background. Other files in the templates folder include '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', and 'train.html'. The main editor area displays the content of the 'support.html' file. The code includes CSS styles for input fields and buttons, and HTML structure for a contact page. The title of the browser tab is 'TravelGo'.

```
templates > support.html > html > head > title
2   <html lang="en">
3     <head>
4       <style>
5         input[type="text"],
6           input[type="email"],
7             textarea {
8               width: 100%;
9               padding: 12px;
10              border-radius: 5px;
11              border: 1px solid #ccc;
12              font-size: 15px;
13              resize: vertical;
14            }
15
16           button {
17             background-color: #0077b6;
18             color: white;
19             padding: 12px 20px;
20             border: none;
21             border-radius: 5px;
22             font-size: 16px;
23             margin-top: 20px;
24             cursor: pointer;
25           }
26
27           button:hover {
28             background-color: #005f8f;
29           }
30         </style>
31       </head>
32     <body>
33       <div class="container">
34         <h1>Contact Support</h1>
35
36         <div class="info">
37           <p><strong>Email:</strong> support@travelgo.com</p>
38           <p><strong>Phone:</strong> +91-9876543210</p>
39           <p>We are available 24/7 to assist you.</p>
40         </div>
41
42       </body>
43     </html>
```

Support page:

Contact Support

Email: support@travelgo.com

Phone: +91-9876543210

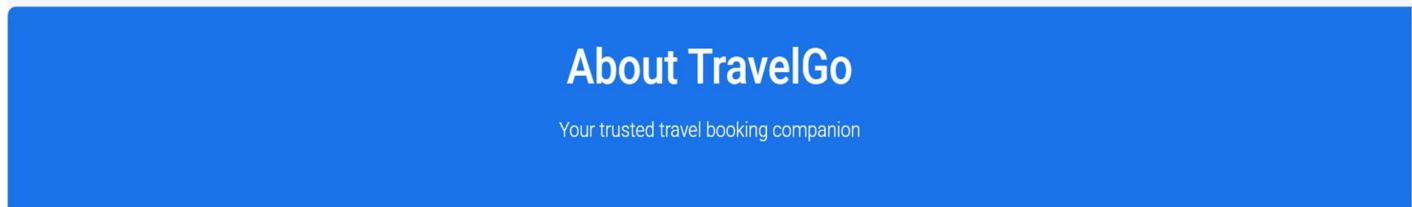
We are available 24/7 to assist you.

aboutus code:

The screenshot shows a code editor interface with the title bar "TravelGo". The left sidebar displays a file tree for a project named "TRAVELGO". The "templates" folder contains several HTML files, including "aboutus.html", which is currently selected and shown in the main editor area. The code in "aboutus.html" is a well-structured HTML document with sections for the header, main content (offer, mission, contact), and footer. It includes various icons and bullet points. The status bar at the bottom indicates "Ln 111, Col 16" and "Spaces: 2".

```
<html lang="en">
  <body>
    <div class="header">
      <h1>About TravelGo</h1>
      <p>Your trusted travel booking companion</p>
    </div>
    <div class="container">
      <h2>We Offer</h2>
      <p>We provide seamless access to:</p>
      <ul>
        <li>Bus ticket bookings across cities</li>
        <li>Train travel with real-time availability</li>
        <li>Flight reservations with the best fares</li>
        <li>Affordable hostel bookings with reviews and ratings</li>
      </ul>
      <p>
        ✓ Easy-to-use interface<br>
        ✓ 24/7 support and assistance<br>
        ✓ Trusted by thousands of happy travelers<br>
        ✓ Safe and secure booking process
      </p>
      <h2>Our Mission</h2>
      <p>We aim to empower every traveler with convenience, transparency, and the freedom to plan their journey stress-free. With TravelGo, you can book your entire trip from a single platform, saving time and money.</p>
      <h2>Contact Us</h2>
      <p>Email: <a href="mailto:support@travelgo.com">support@travelgo.com</a><br>
        | Phone: +91-9876543210
      </p>
    </div>
    <footer>
      &copy; 2025 TravelGo. All rights reserved.
    </footer>
  </body>
</html>
```

about us page:



TravelGo is an all-in-one travel platform that makes planning and booking your journeys simple, quick, and reliable. Whether you're looking to catch a **bus**, **train**, **flight**, or book a comfortable **hostel** for your stay, TravelGo has you covered.

We Offer

We provide seamless access to:

- 🚍 Bus ticket bookings across cities
 - 🚉 Train travel with real-time availability
 - ✈ Flight reservations with the best fares
 - 💳 Affordable hostel bookings with reviews and ratings
- ✓ Easy-to-use interface
 ✓ 24/7 support and assistance
 ✓ Trusted by thousands of happy travelers
 ✓ Safe and secure booking process

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.