Project Description:

Capturemoments is a digital platform designed to streamline the booking of Photographers for University and personal events. It eliminates the traditional manual scheduling system by offering an automated, user-friendly interface where People and faculty can browse, book and receive real-time updates from Phographers.

The System is developed using Flask for backend operations, Amazon Ec2 for hosting, DynamoDB for storing booking details, and **SNS (Simple Notification Service)** for email notifications. The platform ensures efficient booking, timely communication, and scalable operations, especially during peak seasons like graduation, festivals, and academic events.

Scenario 1: Photographer Booking Form Submission

The system provides a simple booking form interface. A user selects the photographer type, user ID, event date, and price. Once the form is submitted, the backend Flask application stores the data into DynamoDB.

Scenario 2: Successful Booking Confirmation

Upon submission, users are redirected to a confirmation screen that displays:

- Booking success message
- Photographer type
- Date
- Price

Scenario 3: Viewing Booking History

Users can navigate to the **Booking History Page** to view their previous bookings. The history includes:

- Photographer Type
- User ID
- Date
- Price

Scenario 4: Homepage for Navigation

The **Homepage** welcomes users and provides direct links to all important pages:

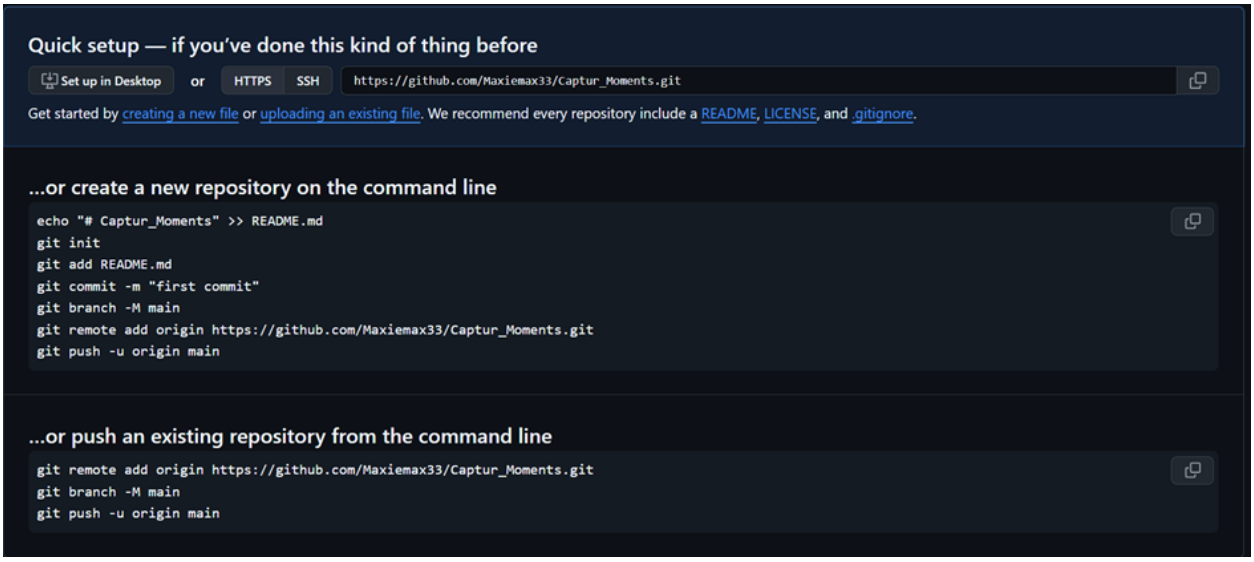
- Home
- Book Now
- Booking History

Pre-Requisites

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

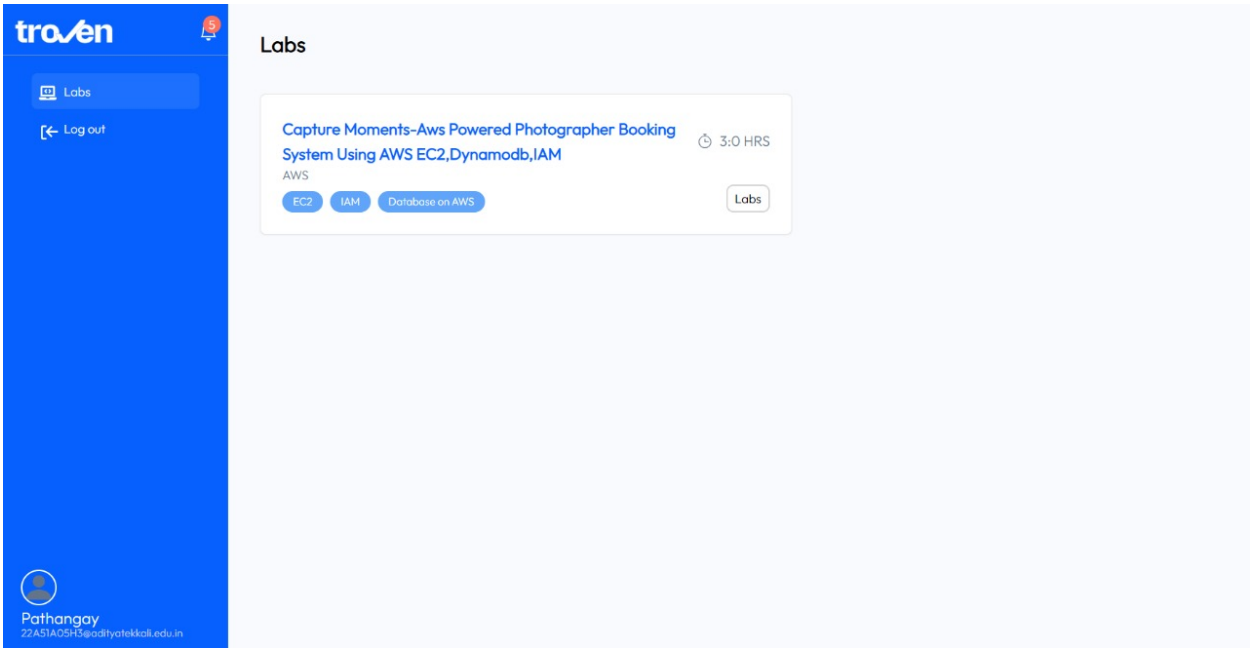
Project Work Flow

1.UPLOAD ALL FILES ON GITHUB

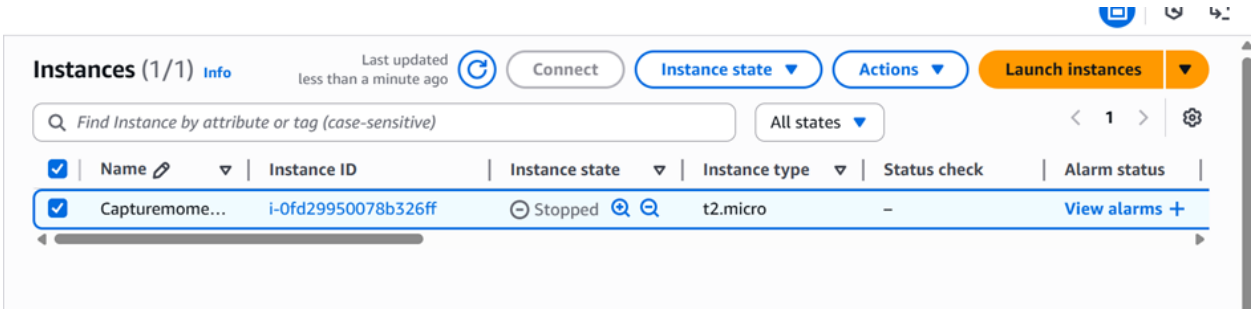


Git Bash path

2.NOW BEGIN WITH YOUR TROVEN LOGIN AND ACCESS

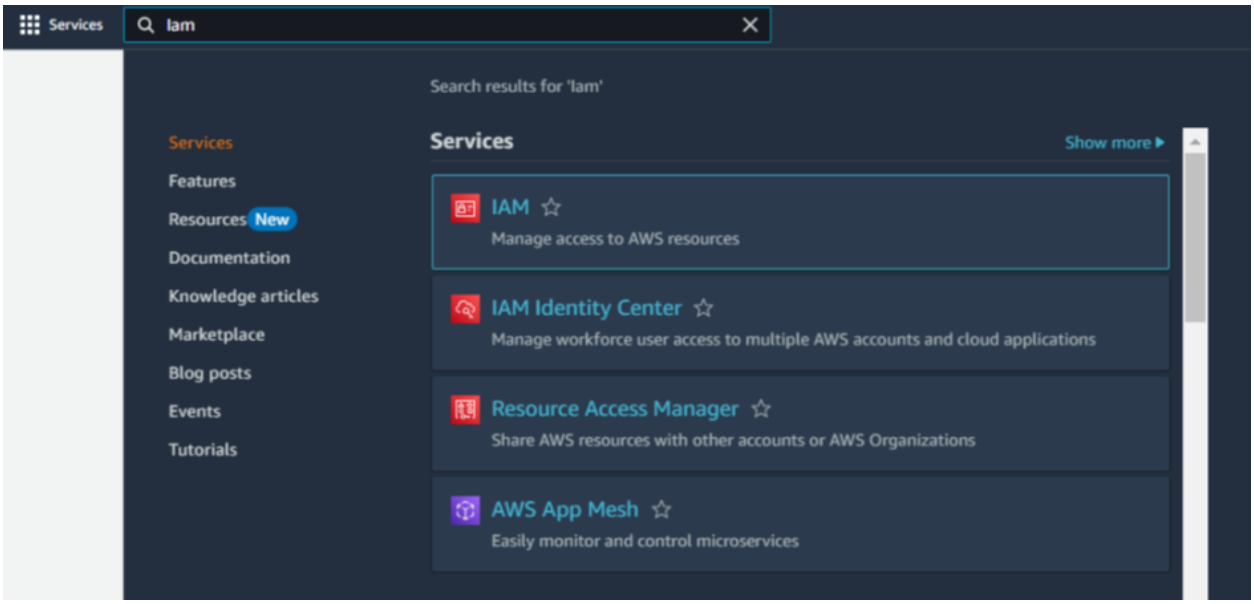


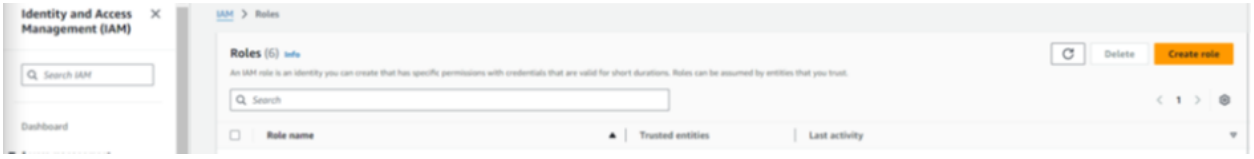
3.INITIALISED THE EC2 INSTANCE

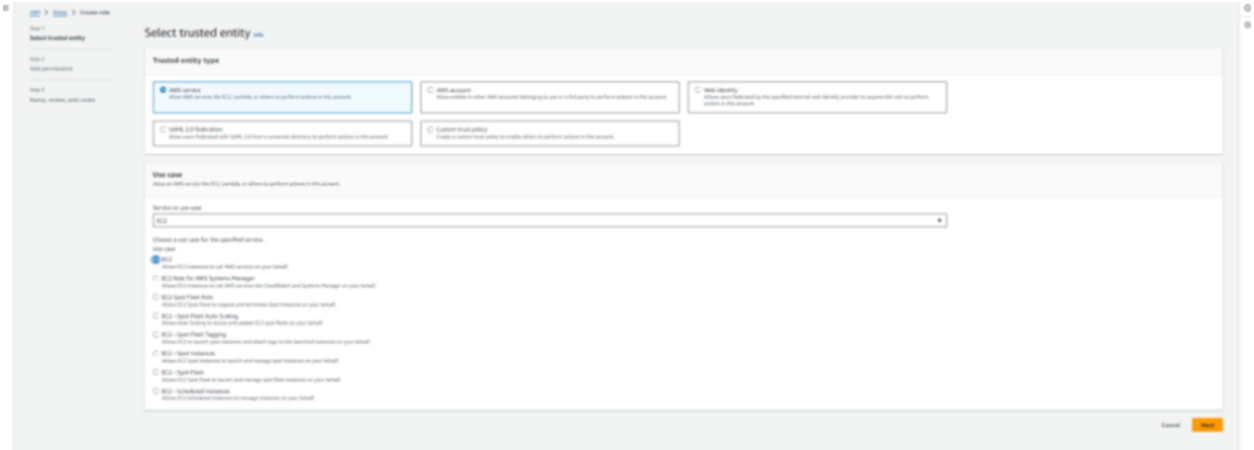


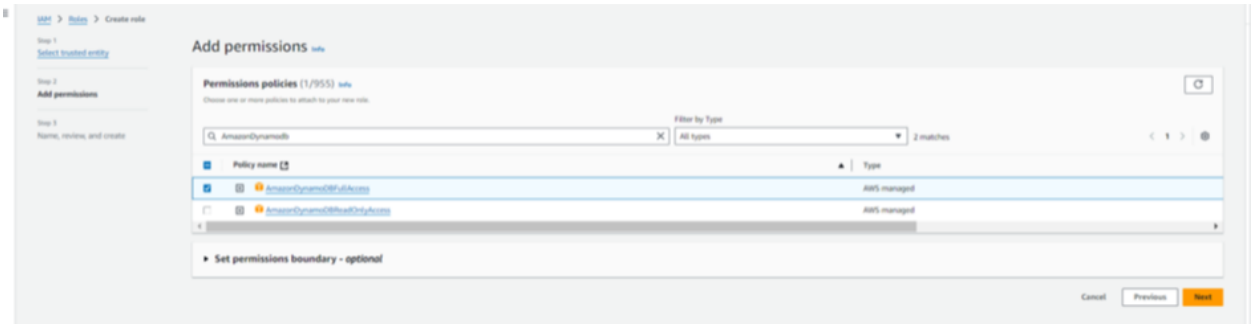
4.CREATED THE IAM ROLE

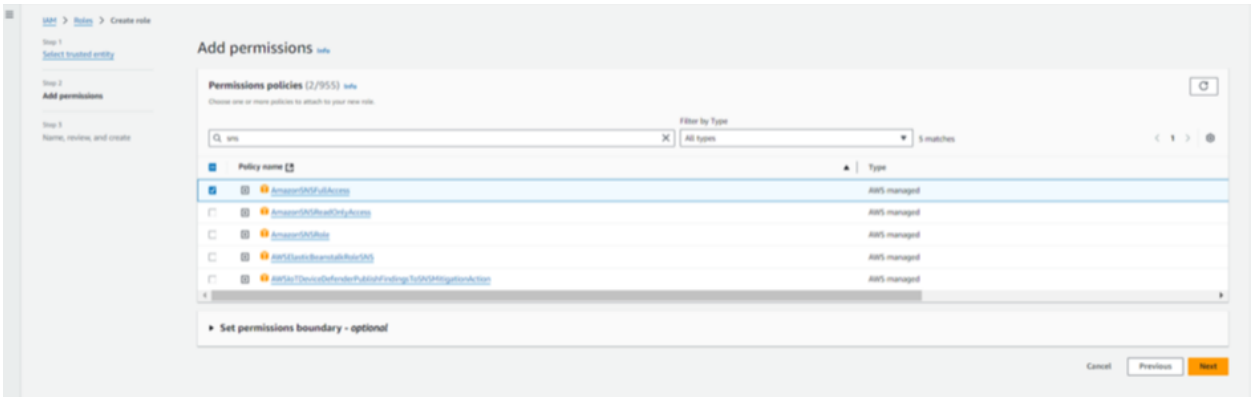
In Creation of IAM Role











- **Trusted entity type**: AWS service
- **Use case**: EC2

Attached permission:

- By ModifyIAM Role AmazonDynamoDBFullAccess

5.CREATED DYNAMODB TABLES

Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes
Resource-based policy	Not active	Yes

Tags

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

No tags are associated with the resource.

Add new tag

You can add 50 more tags.

Cancel Create table

DynamoDB > Tables > Create table

Create table

Table details Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

Users

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

email | String

1 to 255 characters and case sensitive.

Sort key - *optional*

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

Enter the sort key name | String

1 to 255 characters and case sensitive.

DynamoDB

Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations New

The Users table was created successfully.

DynamoDB > Tables

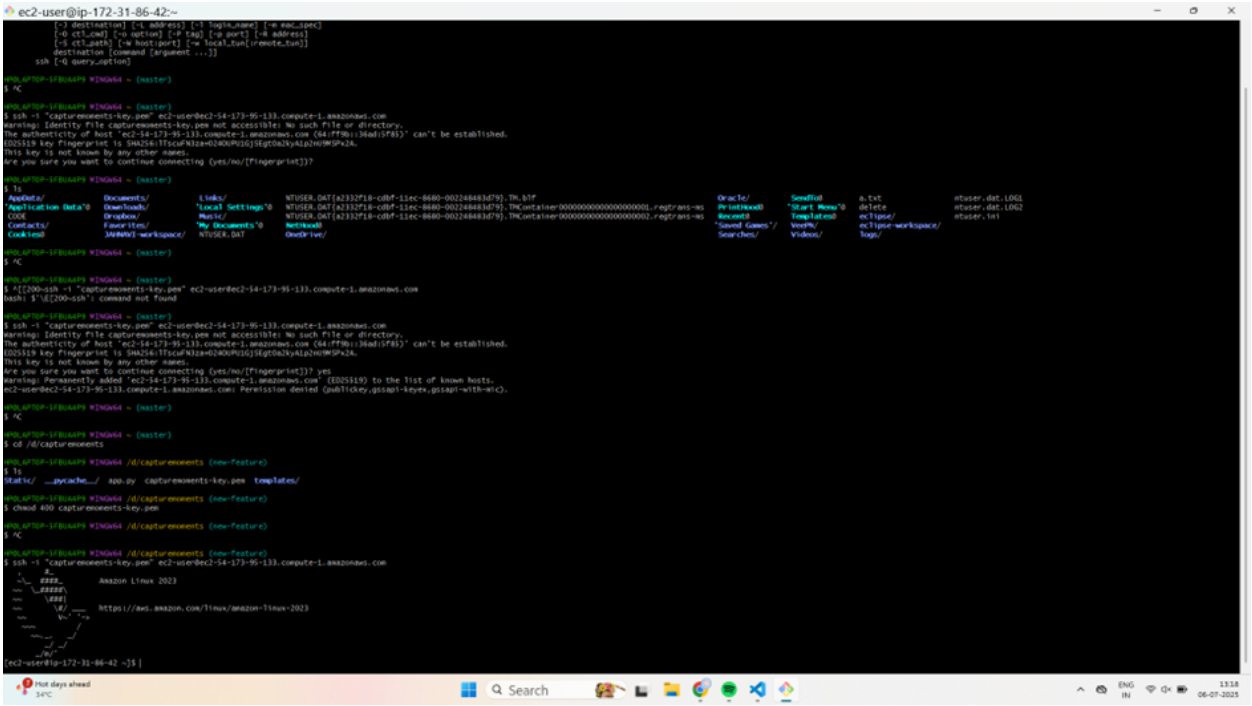
Tables (1) Info

Actions | Delete | Create table

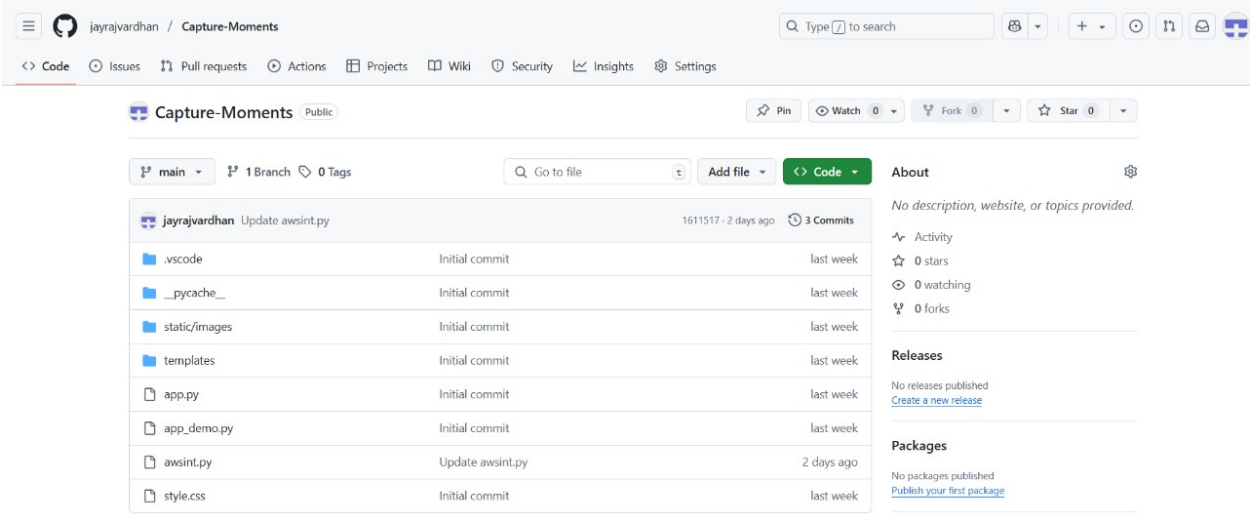
Find tables | Any tag key | Any tag value

Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size
Users	Active	email (S)	-	0	Off	Provisioned (5)	Provisioned (5)	0 bytes

6.STARTED INSTANCE IN GITBASH AND INSTALL NECESSARY DEPENDENCIES

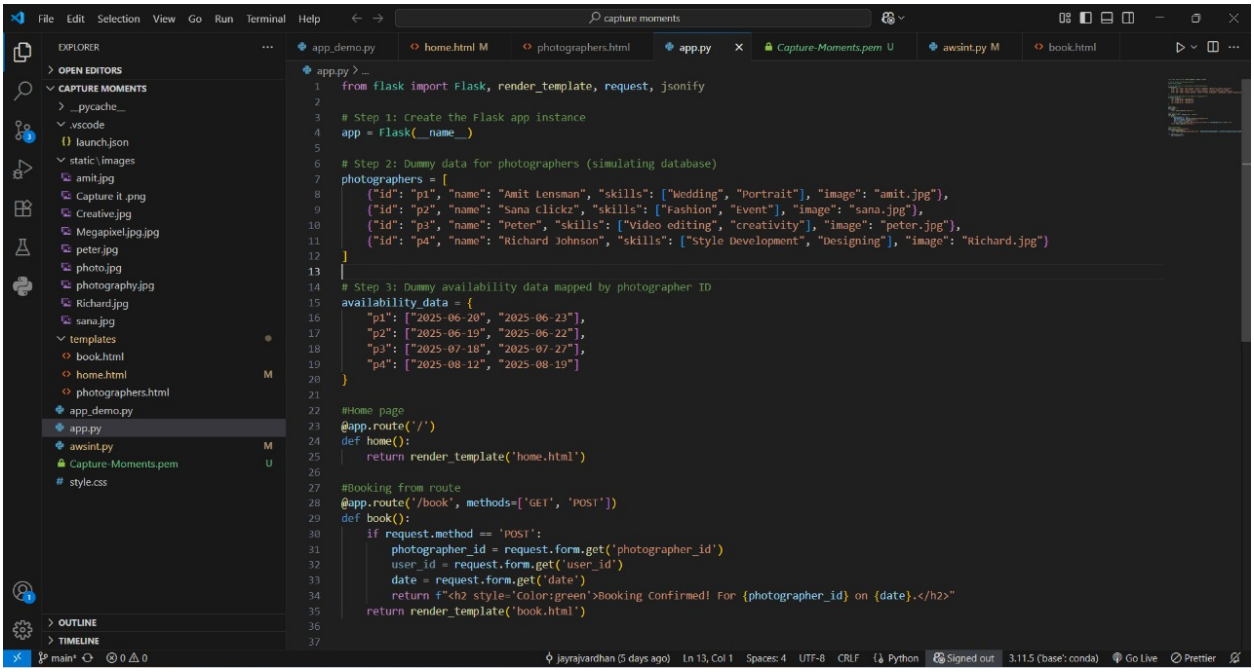


UPLOADED GITHUB REPOSITORY FILES



CAPTURE MOMENTS BOOKING SYSTEM PROJECT

MILESTONE 1: The Project Files in VISUAL STUDIO



MILESTONE 2: WORKING OF book.html code

app_demo.py | home.html M | photographers.html | app.py | Capture-Moments.pem U | awsint.py M | book.html ×

templates > book.html

```html
<head>
  <style>
    @media (max-width: 480px) {
      .container {
        padding: 30px 20px;
      }

      h2 {
        font-size: 24px;
      }
    }
  </style>
</head>
<body>

  <div class="container">
    <h2>Book a Photographer</h2>
    <form method="POST">
      <input type="text" name="photographer_id" placeholder="Photographer ID" required>
      <input type="text" name="user_id" placeholder="Your User ID" required>
      <input type="date" name="date" required>
      <button type="submit">Book Now</button>
    </form>
  </div>

</body>
</html>
```

<meta charset="UTF-8">

- **Ensures your webpage uses UTF-8 encoding to support most characters and symbols**

<title>Book Page</title>

- **Title of the browser tab.**

<style>...</style>

This block defines CSS styling for the page.

Background:Dynamically loads a photo using Flask's url_for() from the static folder.

url_for('static', filename='photography.jpg')

Working of the booking system form is

- **Book a Photographer**
- **photographer ID**
- **Your User ID**
- **Date**

app_demo.py | home.html M | photographers.html | app.py | Capture-Moments.pem U | awsint.py M | book.html ×

templates > book.html

```html
<head>
  <style>
    @media (max-width: 480px) {
      h2 {
        font-size: 24px;
      }
    }
  </style>
</head>
<body>

  <div class="container">
    <h2>Book a Photographer</h2>
    <form method="POST">
      <input type="text" name="photographer_id" placeholder="Photographer ID" required>
      <input type="text" name="user_id" placeholder="Your User ID" required>
      <input type="date" name="date" required>
      <button type="submit">Book Now</button>
    </form>
  </div>

</body>
</html>
```

MILESTONE 3: INTRODUCTION OF THE WEBSITE AT home.html

```
templates > home.html
<html lang="en">
  <head>
    <style>
      .button:hover {
        background-color: #f1c40f;
        color: #2c3e50;
        transform: scale(1.05);
        box-shadow: 0 8px 16px rgba(0, 0, 0, 0.3);
      }

      /* Responsive */
      @media (max-width: 600px) {
        h1 {
          font-size: 2.5rem;
        }
        .button {
          width: 100%;
          max-width: 240px;
        }
      }
    </style>
  </head>
  <body>
    <h1>Capture Moments</h1>
    <p>You can explore platform for booking professional photographers!</p>

    <div class="buttons">
      <a href="/book" class="button">Book a Photographer</a>
      <a href="/show-photographers" class="button">View Photographers</a>
    </div>
  </body>
</html>
```

- **It visualise the welcome home page of capture moments project**
- **It visualise the welcome home page of capture moments project**

MILESTONE4: STORING THE DATA OF BOOKING SYSTEM AND SHOWS THE COPY of booking.html

templates > photographers.html

```html
<head>
  <style>
    .back-link {
      font-weight: bold;
    }

    .back-link:hover {
      text-decoration: underline;
    }
  </style>
</head>
<body>
  <h2>Our Professional Photographers</h2>
  <div class="card-container">
    {% for p in photographers %}
      <div class="photographer-card">
        <img src="{{ url_for('static', filename='images/' + p.image) }}" alt="{{ p.name }}">
        <div class="photographer-name">{{ p.name }}</div>
        <div class="skills"><strong>Skills:</strong> {{ p.skills | join(', ') }}</div>
      <div class="availability"><strong>Available on:</strong> {{ availability_data[p.id] | join(', ') }}</div>

        <a href="/book" class="book-btn">Book Now</a>
      </div>
    {% endfor %}
  </div>
  <a href="/" class="back-link">← Back to Home</a>
</body>
</html>
```

<u>MILESTONE 5:</u> WORKING OF app.py

- **APP.py Runs with the flask app .**
- **APP.py Manages the html codes to run properly .**
- **APP.py is**

```python
from flask import Flask, render_template, request, jsonify

# Step 1: Create the Flask app instance
app = Flask(__name__)

# Step 2: Dummy data for photographers (simulating database)
photographers = [
    {"id": "p1", "name": "Amit Lensman", "skills": ["Wedding", "Portrait"], "image": "amit.jpg"},
    {"id": "p2", "name": "Sana Clickz", "skills": ["Fashion", "Event"], "image": "sana.jpg"},
    {"id": "p3", "name": "Peter", "skills": ["Video editing", "creativity"], "image": "peter.jpg"},
    {"id": "p4", "name": "Richard Johnson", "skills": ["Style Development", "Designing"], "image": "Richard.jpg"}
]

# Step 3: Dummy availability data mapped by photographer ID
availability_data = {
    "p1": ["2025-06-20", "2025-06-23"],
    "p2": ["2025-06-19", "2025-06-22"],
    "p3": ["2025-07-18", "2025-07-27"],
    "p4": ["2025-08-12", "2025-08-19"]
}
```

the major role in hosting the website .

MILESTONE 6 :

Working Of AWS.int :

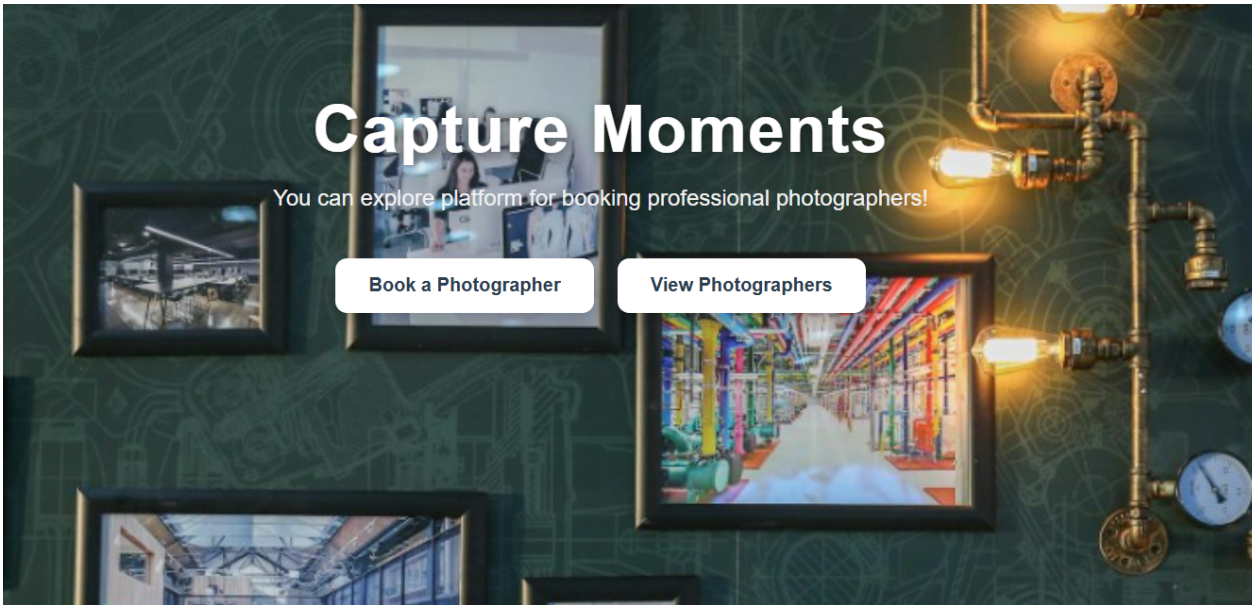
1. Deploys the actual website
2. It helps to install boto3 application

awsint.py > ...

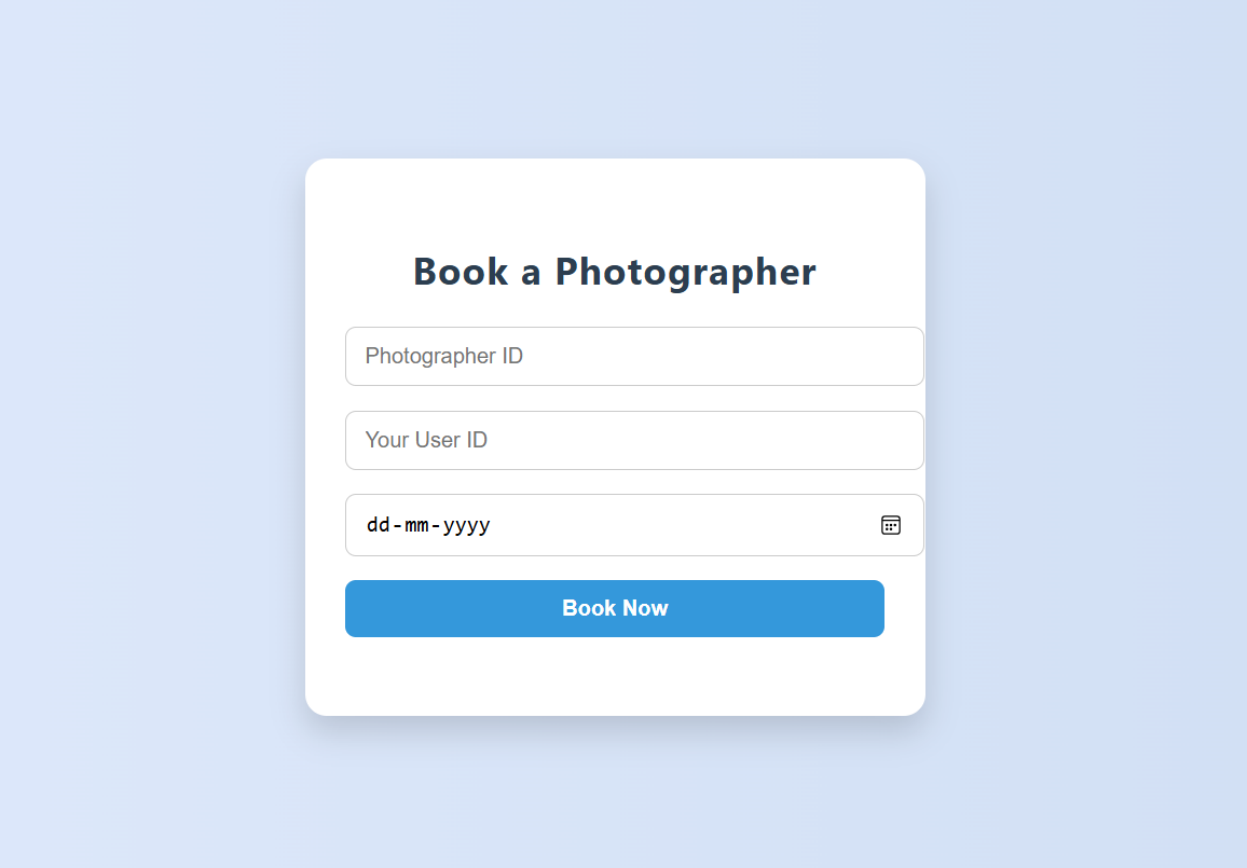
```python
from flask import Flask, render_template, request, jsonify
import boto3
import uuid
from datetime import datetime

# Step 1: Create the Flask app instance
app = Flask(__name__)

# Step 2: Connect to DynamoDB
dynamodb = boto3.resource('dynamodb', region_name='ap-south-1')  # Replace with your region

# Tables
photographers_table = dynamodb.Table('photographers')
bookings_table = dynamodb.Table('booking')

# Home Page
@app.route('/')
def home():
    return render_template('home.html')
```

FINAL OUTPUT OF THE CAPTURE MOMENTS WEBSITE

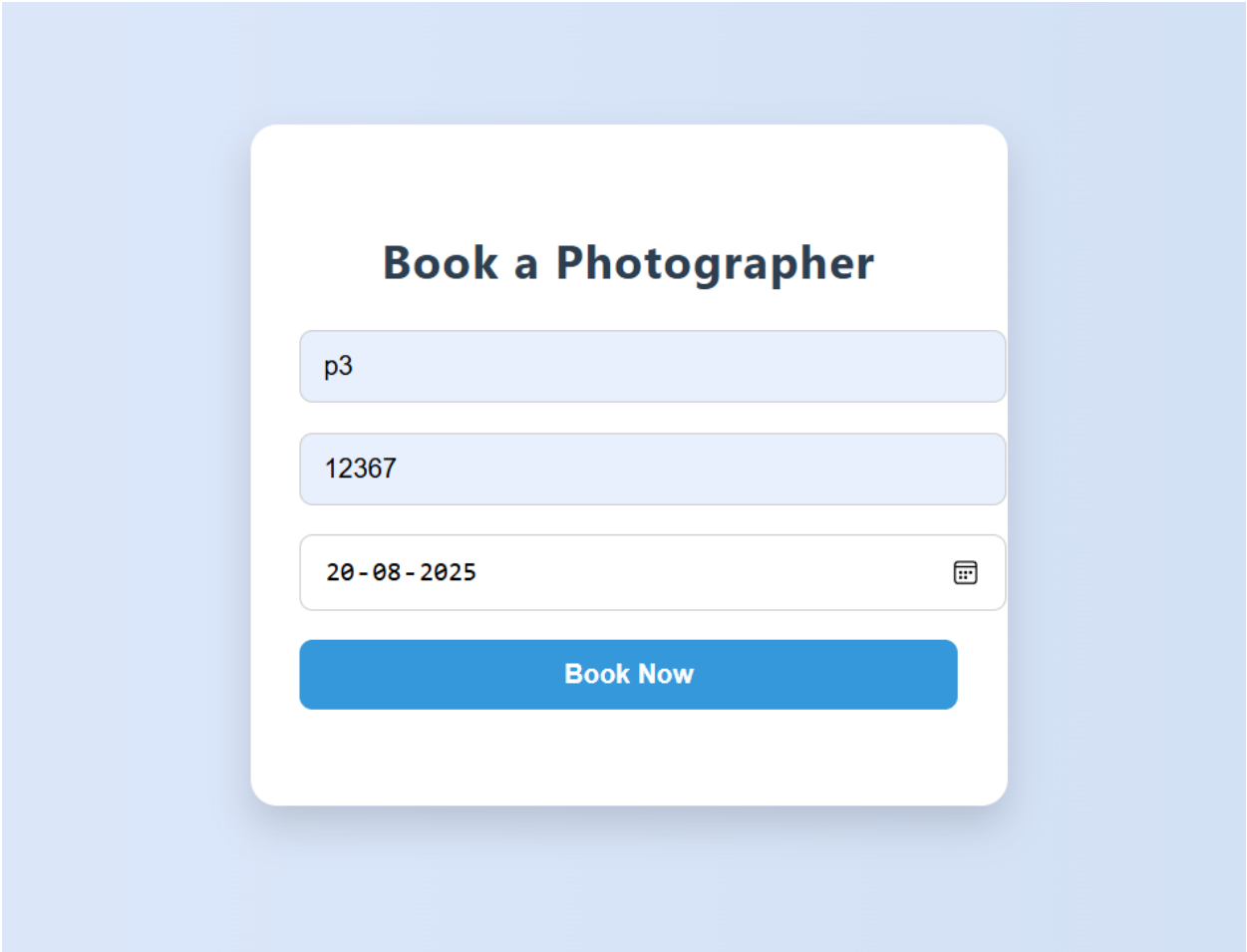
HOME PAGE

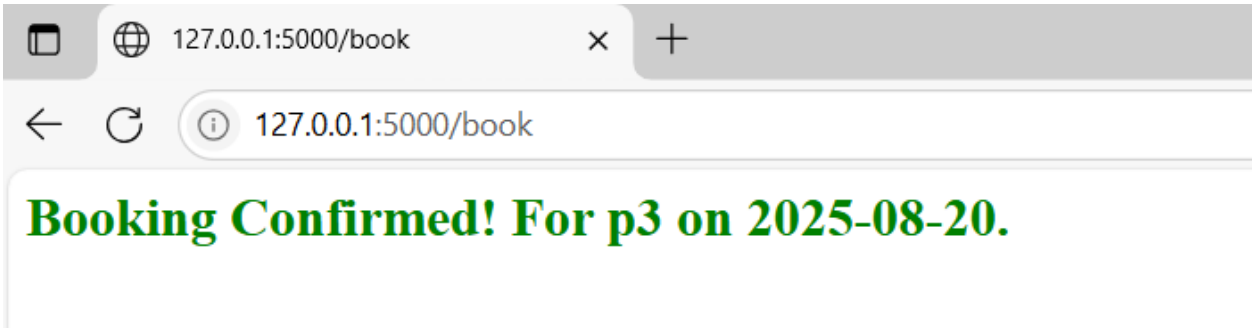


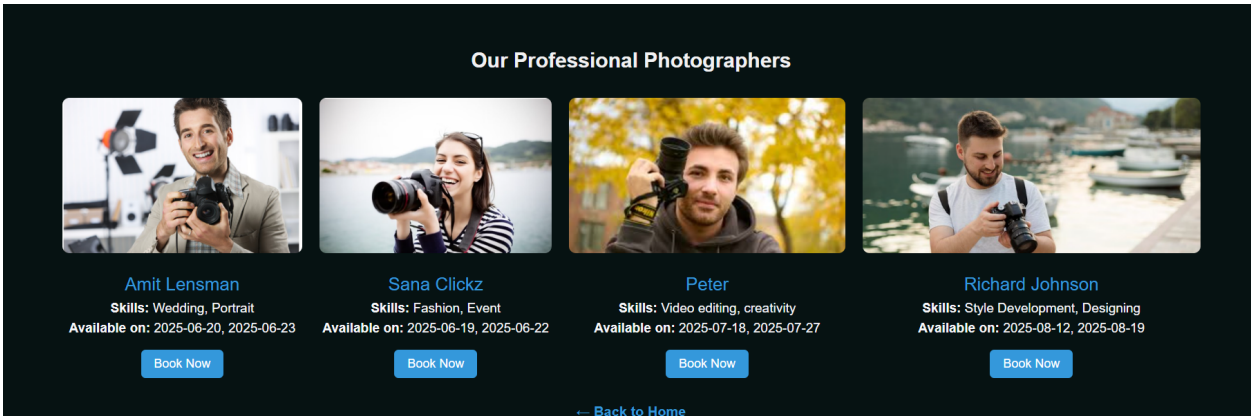
BOOKING PHOTOGRAPHERS PAGE



Process of booking phographers




127.0.0.1:5000/book
127.0.0.1:5000/book
Booking Confirmed! For p3 on 2025-08-20.


Our Professional Photographers
Amit Lensman
Skills: Wedding, Portrait
Available on: 2025-06-20, 2025-06-23
Book Now
Sana Clickz
Skills: Fashion, Event
Available on: 2025-06-19, 2025-06-22
Book Now
Peter
Skills: Video editing, creativity
Available on: 2025-07-18, 2025-07-27
Book Now
Richard Johnson
Skills: Style Development, Designing
Available on: 2025-08-12, 2025-08-19
Book Now
← Back to Home

CONCLUSION OF THE CAPTURE MOMENTS BOOKING SYSTEM PROJECT

- **"Capture Moments" project stands as a modern solution that bridges photographers and clients through a seamless, cloud-powered booking platform.**
- **Using the power of AWS EC2 and DynamoDB, supported by a Flask web framework, this system ensures flexibility, scalability, and reliability.**
- **From launching an EC2 instance, configuring security, to developing a user-friendly interface — every step reflected a deeper understanding of cloud infrastructure and full-stack development.**
- **This journey wasn't just about deploying code; it was about building a meaningful, accessible application that connects people to capture life's most precious memories**
- **As this project concludes, it opens doors for future enhancements, new features, and wider adoption — making every moment count, one booking at a time.**