

## **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 Photographers.

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


Quick setup — if you've done this kind of thing before

 Set up in Desktop or **HTTPS** **SSH** `https://github.com/Maxiemax33/Captur_Moments.git` 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).


...or create a new repository on the command line

```
echo "# Captur_Moments" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Maxiemax33/Captur_Moments.git
git push -u origin main
```



...or push an existing repository from the command line

```
git remote add origin https://github.com/Maxiemax33/Captur_Moments.git
git branch -M main
git push -u origin main
```



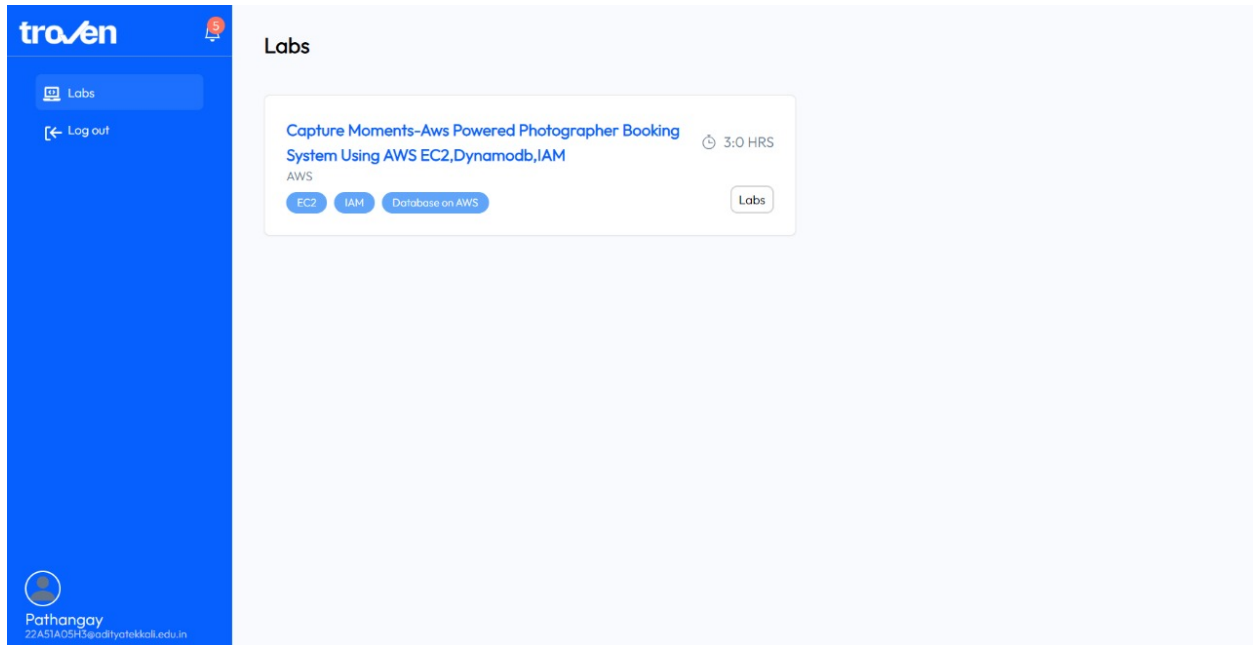
## Git Bash path

MINGW64:/d/capturemoments

```
HP@LAPTOP-5FBUA4P9 MINGW64 ~ (master)
$ cd d:\capturemoments

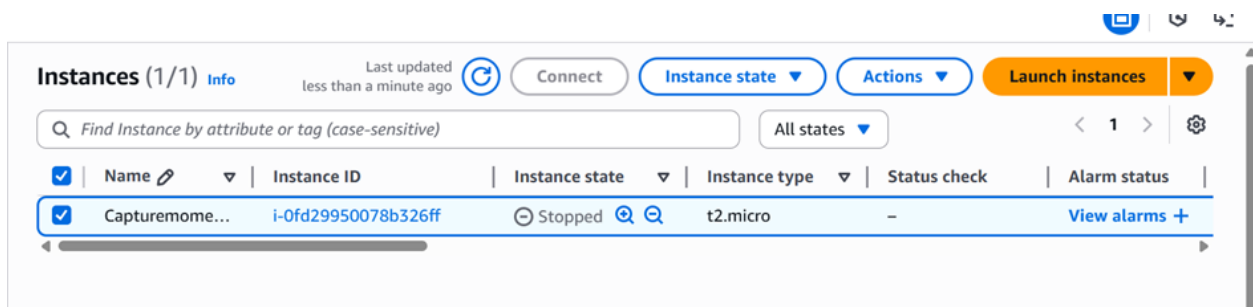
HP@LAPTOP-5FBUA4P9 MINGW64 /d/capturemoments (new-feature)
$
```

## 2.NOW BEGIN WITH YOUR TROVEN LOGIN AND ACCESS



The screenshot shows the Troven Labs dashboard. On the left is a blue sidebar with the Troven logo, a 'Labs' button, a 'Log out' button, and a user profile for 'Pathangay' with email '22ASIA05H3@adityatekkali.edu.in'. The main area is titled 'Labs' and features a card for 'Capture Moments-Aws Powered Photographer Booking System Using AWS EC2,Dynamodb,IAM'. The card includes an 'AWS' label, buttons for 'EC2', 'IAM', and 'Database on AWS', and a 'Labs' button. A timer indicates '3:0 HRS'.

## 3.INITIALISED THE EC2 INSTANCE

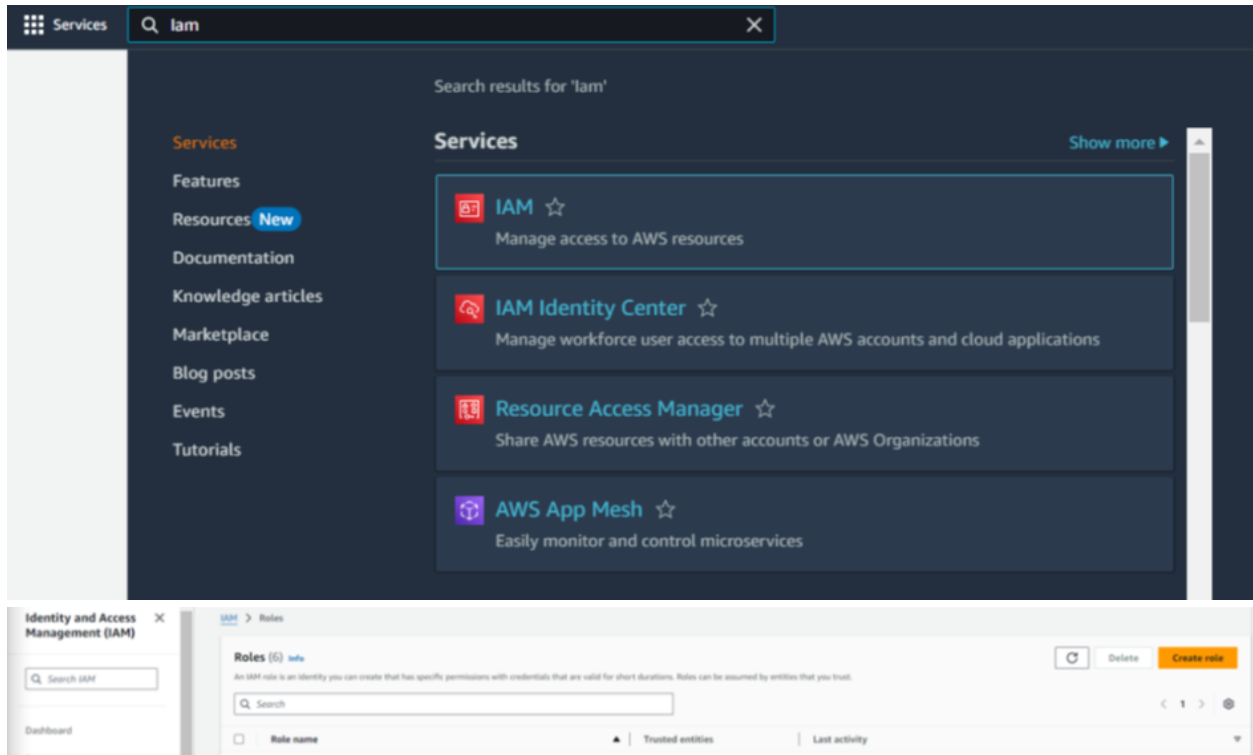


The screenshot shows the AWS Management Console 'Instances' page. It displays one instance named 'Capturemome...' with ID 'i-0fd29950078b326ff', state 'Stopped', and type 't2.micro'. The instance is associated with a 'Status check' of '-' and an 'Alarm status' of 'View alarms +'. The page includes a search bar, filters, and a 'Launch instances' button.

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
<input checked="" type="checkbox"/>	Capturemome...	i-0fd29950078b326ff	Stopped	t2.micro	-	View alarms +

## 4.CREATED THE IAM ROLE

### In Creation of IAM Role



The image shows two screenshots from the AWS IAM console. The top screenshot is a search results page for 'iam'. The search bar at the top contains 'iam'. The left sidebar shows navigation links: Services, Features, Resources (marked as 'New'), Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main content area, titled 'Search results for 'iam'', lists several services: IAM (Manage access to AWS resources), IAM Identity Center (Manage workforce user access to multiple AWS accounts and cloud applications), Resource Access Manager (Share AWS resources with other accounts or AWS Organizations), and AWS App Mesh (Easily monitor and control microservices). The bottom screenshot shows the 'IAM > Roles' page. It has a search bar and a 'Create role' button. Below the search bar, there is a table with columns for 'Role name', 'Trusted entities', and 'Last activity'. The table is currently empty.

**Select trusted entity**

**Trusted entity type**

- ☒ **AWS service**  
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.
- ☐ **AWS account**  
Allow actions in your AWS account belonging to you or a third party to perform actions in this account.
- ☐ **Open identity**  
Allow users associated by the specified external web identity provider to access this role to perform actions in this account.
- ☐ **OAuth 2.0 Federation**  
Allow your OAuth 2.0 client (OAuth 2.0 client) to access this role to perform actions in this account.
- ☐ **Custom trust policy**  
Custom trust policy to enable others to perform actions in this account.

**Use case**  
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or user role

EC2

Choose a use case for the specified service.

Use case

- ☒ **EC2**  
Allow EC2 instances to call AWS services on your behalf.
- ☐ **EC2 Role for AWS Systems Manager**  
Allow EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.
- ☐ **EC2 Spot Fleet Role**  
Allow EC2 Spot Fleet to request and manage Spot instances on your behalf.
- ☐ **EC2 - Spot Fleet Auto Scaling**  
Allow EC2 Spot Fleet to request and manage EC2 Spot Fleet on your behalf.
- ☐ **EC2 - Spot Fleet Tagging**  
Allow EC2 Spot Fleet to request and manage EC2 Spot Fleet on your behalf.
- ☐ **EC2 - Spot Instance**  
Allow EC2 Spot Instance to request and manage EC2 Spot Instance on your behalf.
- ☐ **EC2 - Spot Instance**  
Allow EC2 Spot Instance to request and manage EC2 Spot Instance on your behalf.
- ☐ **EC2 - Scheduled instances**  
Allow EC2 Spot Instance to request and manage EC2 Spot Instance on your behalf.
- ☐ **EC2 - Scheduled instances**  
Allow EC2 Spot Instance to request and manage EC2 Spot Instance on your behalf.

Cancel **Next**

**Add permissions**

**Permissions policies (1/955)**

Choose one or more policies to attach to your new role.

Filter by Type: All types 2 matches

Policy name	Type
<input checked="" type="checkbox"/> <b>AmazonDynamoDBFullAccess</b>	AWS managed
<input type="checkbox"/> <b>AmazonDynamoDBReadOnlyAccess</b>	AWS managed

Set permissions boundary - optional

Cancel Previous **Next**

**Add permissions**

**Permissions policies (2/955)**

Choose one or more policies to attach to your new role.

Filter by Type: All types 5 matches

Policy name	Type
<input checked="" type="checkbox"/> <b>AmazonS3FullAccess</b>	AWS managed
<input type="checkbox"/> <b>AmazonS3OutpostsFullAccess</b>	AWS managed
<input type="checkbox"/> <b>AmazonS3OutpostsReadOnlyAccess</b>	AWS managed
<input type="checkbox"/> <b>AmazonS3OutpostsReadOnlyAccess</b>	AWS managed
<input type="checkbox"/> <b>AmazonS3OutpostsReadOnlyAccess</b>	AWS managed
<input type="checkbox"/> <b>AmazonS3OutpostsReadOnlyAccess</b>	AWS managed
<input type="checkbox"/> <b>AmazonS3OutpostsReadOnlyAccess</b>	AWS managed

Set permissions boundary - optional

Cancel Previous **Next**

- **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.

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.

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.

String ▼

1 to 255 characters and case sensitive.

DynamoDB



The Users table was created successfully.



Dashboard

**Tables**

Explore items

PartiQL editor

Backups

Exports to S3

Imports from S3

Integrations [New](#)

...

[DynamoDB](#) > [Tables](#)

Tables (1) [Info](#)



Actions ▼

Delete

Create table

Any tag key ▼

Any tag value ▼


< 1 > ⚙

<input type="checkbox"/>	Name ▲	Status ▼	Partition key ▼	Sort key ▼	Indexes ▼	Deletion protection ▼	Read capacity mode ▼	Write capacity mode ▼	Total size ▼
<input type="checkbox"/>	<a href="#">Users</a>	Active	email (S)	-	0	Off	Provisioned (S)	Provisioned (S)	0 bytes












[illegible]

## UPLOADED GITHUB REPOSITORY FILES

 jayrajvardhan / Capture-Moments

Type ↵ to search

 Code  Issues  Pull requests  Actions  Projects  Wiki  Security  Insights  Settings

Capture-Moments Public

Pin Watch 0 Fork 0 Star 0

main 1 Branch 0 Tags

Go to file

Add file

Code

About

No description, website, or topics provided.

Activity

0 stars

0 watching

0 forks

Releases

No releases published

[Create a new release](#)

Packages

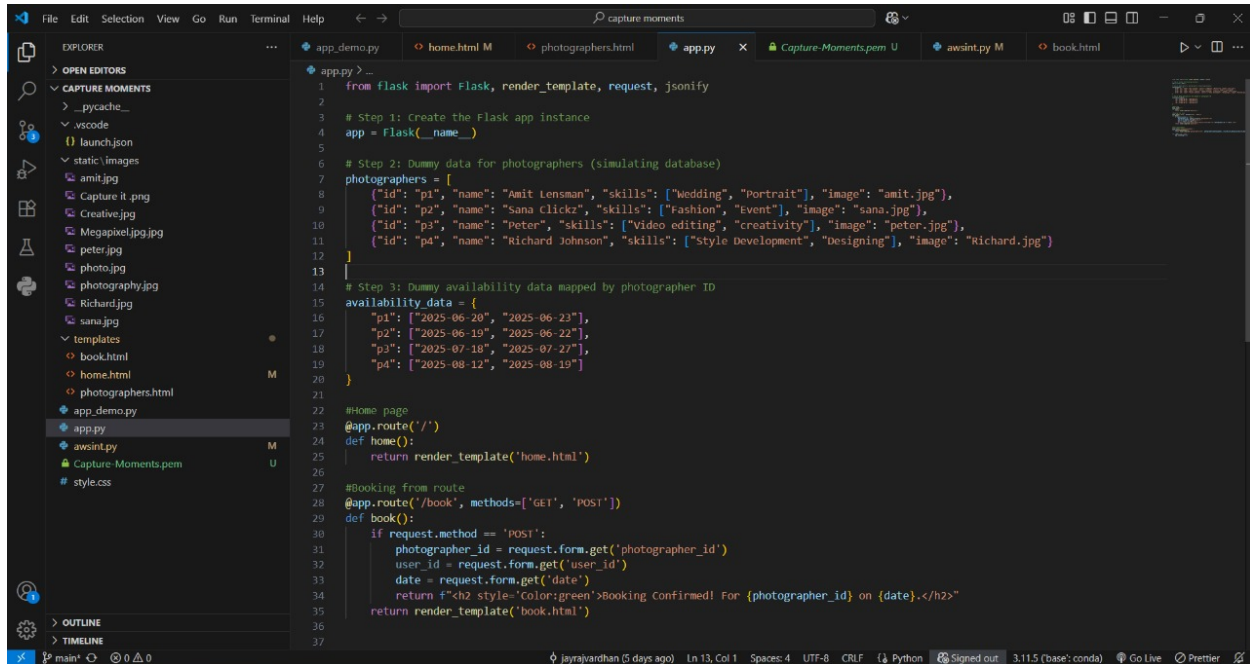
No packages published

[Publish your first package](#)

jayrajvardhan Update awsint.py 1611517 · 2 days ago 3 Commits		
.vscode	Initial commit	last week
__pycache__	Initial commit	last week
static/images	Initial commit	last week
templates	Initial commit	last week
app.py	Initial commit	last week
app_demo.py	Initial commit	last week
awsint.py	Update awsint.py	2 days ago
style.css	Initial commit	last week

# CAPTURE MOMENTS BOOKING SYSTEM PROJECT

## MILESTONE 1: The Project Files in VISUAL STUDIO



```
1 from flask import Flask, render_template, request, jsonify
2
3 # Step 1: Create the Flask app instance
4 app = Flask(__name__)
5
6 # Step 2: Dummy data for photographers (simulating database)
7 photographers = [
8     {"id": "p1", "name": "Amit Ionsman", "skills": ["Wedding", "Portrait"], "image": "amit.jpg"},
9     {"id": "p2", "name": "Sana Clicks", "skills": ["Fashion", "Event"], "image": "sana.jpg"},
10    {"id": "p3", "name": "Peter", "skills": ["Video editing", "creativity"], "image": "peter.jpg"},
11    {"id": "p4", "name": "Richard Johnson", "skills": ["Style Development", "Designing"], "image": "Richard.jpg"}
12 ]
13
14 # Step 3: Dummy availability data mapped by photographer ID
15 availability_data = {
16     "p1": ["2025-06-20", "2025-06-23"],
17     "p2": ["2025-06-19", "2025-06-22"],
18     "p3": ["2025-07-18", "2025-07-27"],
19     "p4": ["2025-08-12", "2025-08-19"]
20 }
21
22 #Home page
23 @app.route('/')
24 def home():
25     return render_template("home.html")
26
27 #Booking from route
28 @app.route('/book', methods=['GET', 'POST'])
29 def book():
30     if request.method == 'POST':
31         photographer_id = request.form.get('photographer_id')
32         user_id = request.form.get('user_id')
33         date = request.form.get('date')
34         return f"<h2 style='color:green'>Booking Confirmed! For {photographer_id} on {date}</h2>"
35     return render_template("book.html")
36
37
```

## 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 X
templates > book.html
3  <head>
7  <style>
66  @media (max-width: 480px) {
67  .container {
69      padding: 30px 20px;
70  }
71
72  h2 {
73      font-size: 24px;
74  }
75  }
76  </style>
77  </head>
78  <body>
79
80  <div class="container">
81      <h2>Book a Photographer</h2>
82      <form method="POST">
83          <input type="text" name="photographer_id" placeholder="Photographer ID" required>
84          <input type="text" name="user_id" placeholder="Your User ID" required>
85          <input type="date" name="date" required>
86          <button type="submit">Book Now</button>
87      </form>
88  </div>
89
90  </body>
91  </html>
92
```

**<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 X
templates > book.html
3  <head>
7  <style>
66  @media (max-width: 480px) {
72      h2 {
73          font-size: 24px;
74      }
75  }
76  </style>
77  </head>
78  <body>
79
80  <div class="container">
81      <h2>Book a Photographer</h2>
82      <form method="POST">
83          <input type="text" name="photographer_id" placeholder="Photographer ID" required>
84          <input type="text" name="user_id" placeholder="Your User ID" required>
85          <input type="date" name="date" required>
86          <button type="submit">Book Now</button>
87      </form>
88  </div>
89
90  </body>
91  </html>
92
```

## MILESTONE 3: INTRODUCTION OF THE WEBSITE AT home.html

```
templates > home.html
2  <html lang="en">
3  <head>
6  <style>
63  .button:hover {
64    background-color: #f1c40f;
65    color: #2c3e50;
66    transform: scale(1.05);
67    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.3);
68  }
69
70  /* Responsive */
71  @media (max-width: 600px) {
72    h1 {
73      font-size: 2.5rem;
74    }
75    .button {
76      width: 100%;
77      max-width: 240px;
78    }
79  }
80  </style>
81  </head>
82  <body>
83    <h1>Capture Moments</h1>
84    <p>You can explore platform for booking professional photographers!</p>
85
86    <div class="buttons">
87      <a href="/book" class="button">Book a Photographer</a>
88      <a href="/show-photographers" class="button">View Photographers</a>
89    </div>
90  </body>
91  </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
3  <head>
5  <style>
70  .back-link {
75      font-weight: bold;
76  }
77
78  .back-link:hover {
79      text-decoration: underline;
80  }
81  </style>
82 </head>
83 <body>
84  <h2>Our Professional Photographers</h2>
85  <div class="card-container">
86      {% for p in photographers %}
87          <div class="photographer-card">
88              
89              <div class="photographer-name">{{ p.name }}</div>
90              <div class="skills"><strong>Skills:</strong> {{ p.skills | join(', ') }}</div>
91              <div class="availability"><strong>Available on:</strong> {{ availability_data[p.id] | join(', ') }}</div>
92              <a href="/book" class="book-btn">Book Now</a>
93          </div>
94      {% endfor %}
95  </div>
96  <a href="/" class="back-link">< Back to Home</a>
97 </body>
98 </html>
```

## MILESTONE 5: WORKING OF app.py

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

```
1 from flask import Flask, render_template, request, jsonify
2
3 # Step 1: Create the Flask app instance
4 app = Flask(__name__)
5
6 # Step 2: Dummy data for photographers (simulating database)
7 photographers = [
8     {"id": "p1", "name": "Amit Lensman", "skills": ["Wedding", "Portrait"], "image": "amit.jpg"},
9     {"id": "p2", "name": "Sana Clickz", "skills": ["Fashion", "Event"], "image": "sana.jpg"},
10    {"id": "p3", "name": "Peter", "skills": ["Video editing", "creativity"], "image": "peter.jpg"},
11    {"id": "p4", "name": "Richard Johnson", "skills": ["Style Development", "Designing"], "image": "Richard.jpg"}
12 ]
13
14 # Step 3: Dummy availability data mapped by photographer ID
15 availability_data = {
16     "p1": ["2025-06-20", "2025-06-23"],
17     "p2": ["2025-06-19", "2025-06-22"],
18     "p3": ["2025-07-18", "2025-07-27"],
19     "p4": ["2025-08-12", "2025-08-19"]
20 }
```

the major role in hosting the website .



## MILESTONE 6 :

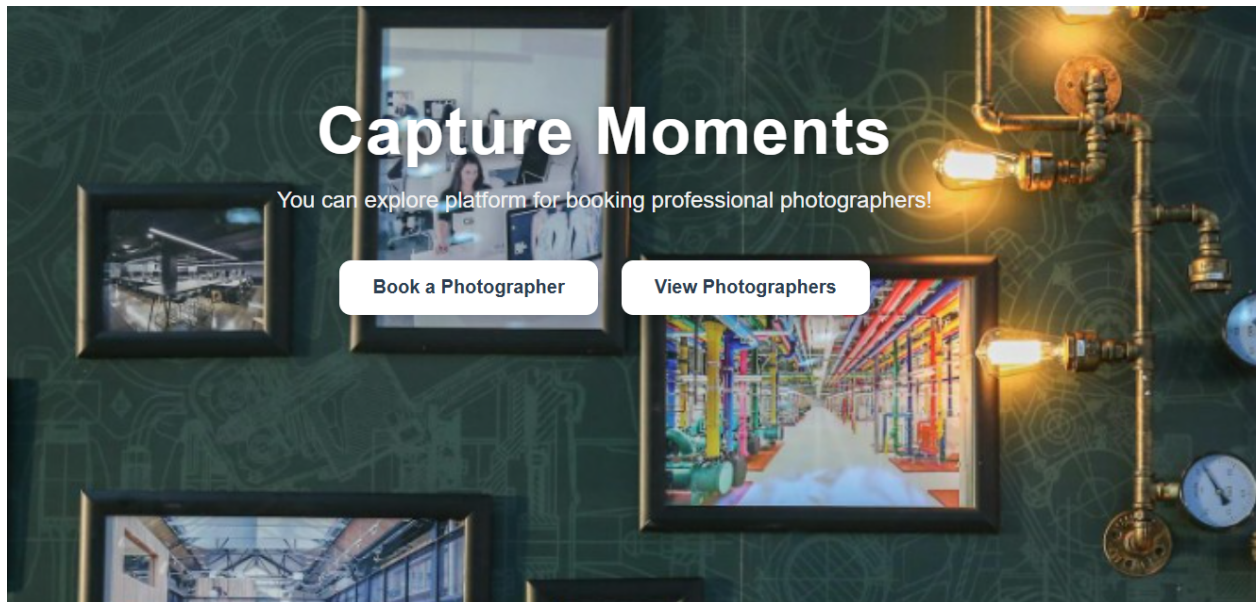
Working Of AWS.int :

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

```
awsint.py > ...
1 | from flask import Flask, render_template, request, jsonify
2 | import boto3
3 | import uuid
4 | from datetime import datetime
5 |
6 | # Step 1: Create the Flask app instance
7 | app = Flask(__name__)
8 |
9 | # Step 2: Connect to DynamoDB
10 | dynamodb = boto3.resource('dynamodb', region_name='ap-south-1') # Replace with your region
11 |
12 | # Tables
13 | photographers_table = dynamodb.Table('photographers')
14 | bookings_table = dynamodb.Table('booking')
15 |
16 | # Home Page
17 | @app.route('/')
18 | def home():
19 |     return render_template('home.html')
20 |
```

## FINAL OUTPUT OF THE CAPTURE MOMENTS WEBSITE

### HOME PAGE



## **BOOKING PHOTOGRAPHERS PAGE**

### **Book a Photographer**



## Process of booking phographers

### Book a Photographer





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.**