**Capture Moments-Aws powered Photographer booking system**

### Project Description:

At Capture Moments Photography, the growing client base has created challenges in managing bookings and connecting customers with the right photographers. The manual booking process was inefficient and prone to errors, causing scheduling conflicts and customer dissatisfaction.

To solve this, Capture Moments developed a cloud-based Photographer Booking Platform. Using Flask for backend development, AWS EC2 for hosting, and DynamoDB for managing data, the system allows clients to register, log in, and book photographers online. This cloud-based solution enhances the booking experience, providing seamless service for all clients while optimizing photographer scheduling and data management.

#### Scenario 1: Efficient Booking System for Clients

In the Capture Moments Booking Platform, AWS EC2 ensures a reliable infrastructure to manage multiple clients accessing the platform simultaneously. For example, a client can log in, navigate to the booking page, and easily submit a request for their preferred photographer. Flask handles backend operations, efficiently retrieving and processing user data in real-time. The cloud-based architecture allows the platform to handle a high volume of booking requests during peak wedding seasons, ensuring smooth operation without delays.

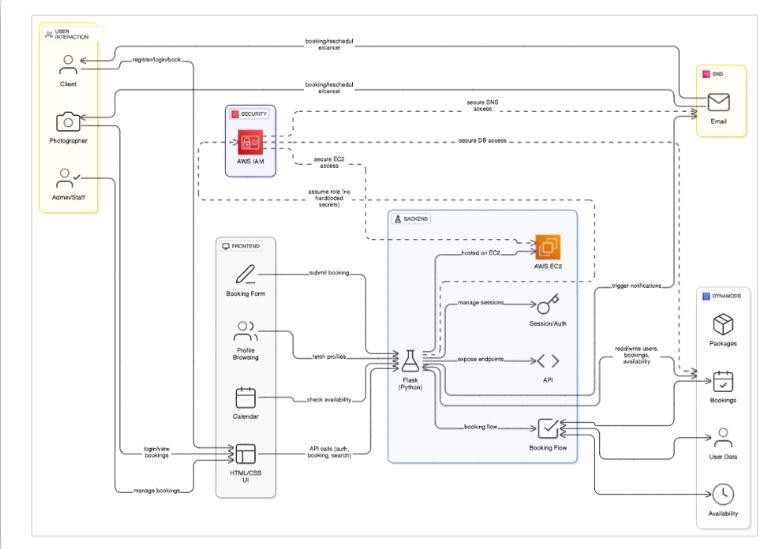
#### Scenario 2: Centralized Booking Management

When clients book photography sessions, the Capture Moments Booking System processes these requests through a streamlined workflow. For instance, a client books a wedding photographer, and Flask processes the request, storing all the booking details securely in DynamoDB. This centralized database approach ensures all booking information is maintained in a single source of truth, allowing administrative staff to track upcoming appointments, manage photographer schedules, and maintain complete records of client preferences and requirements.

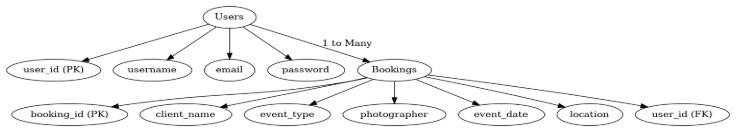
#### Scenario 3: Easy Access to Photography Services

The Capture Moments Booking Platform provides clients with easy access to available photographers and packages. For example, a client logs in and views the list of available photographers specializing in their event type. They can quickly check availability for specific dates or book immediately. Flask manages real-time data fetching from DynamoDB, while EC2 hosting ensures the platform performs seamlessly even when multiple clients access it simultaneously during popular booking periods, offering a smooth and uninterrupted user experience.

AWS ARCHITECTURE



Entity Relationship (ER)Diagram:



### Pre-requisites:

1. .**AWS Account Setup**: [AWS Account Setup](https://youtu.be/CjKhQoYeR4Q?si=ui8Bvk_M4FfVM-Dh)
2. **Understanding IAM**: [IAM Overview](https://youtu.be/gsgdAyGhV0o?si=3qg-bULgkD4LXNvR)
3. **Amazon EC2 Basics**: [EC2 Tutorial](https://youtu.be/8TlukLu11Yo?si=MUj0nEAOESRhHUIz)
4. **DynamoDB Basics**: [DynamoDB Introduction](https://docs.aws.amazon.com/dynamodb)
5. **Git Version Control**: [Git Documentation](https://git-scm.com/doc)

### Project WorkFlow:

#### AWS Account Setup and Login

**Activity 1.1:** Set up an AWS account if not already done**. Activity 1.2:** Log in to the AWS Management Console

#### DynamoDB Database Creation and Setup

**Activity 2.1**: Create a DynamoDB Table.

**Activity 2.2**: Configure Attributes for User Data and Book Requests.

#### Backend Development and Application Setup

**Activity 3.1**:Develop the Backend Using Flask.

**Activity 3.2**: Integrate AWS Services Using boto3.

#### IAM Role Setup

**Activity 4.1**: Create IAM Role

**Activity 4.2**: Attach Policies

#### EC2 Instance Setup

**Activity 5.1**: Launch an EC2 instance to host the Flask application.

**Activity 5.2**: Configure security groups for HTTP, and SSH access.

1. **Deployment on EC2 Activity 6.1**:Upload Flask Files **Activity 6.2**: Run the Flask App

#### Testing and Deployment

**Activity 7.1**: Conduct functional testing to verify user registration, login, book requests, and notifications.

## Milestone 1: AWS Account Setup and Login

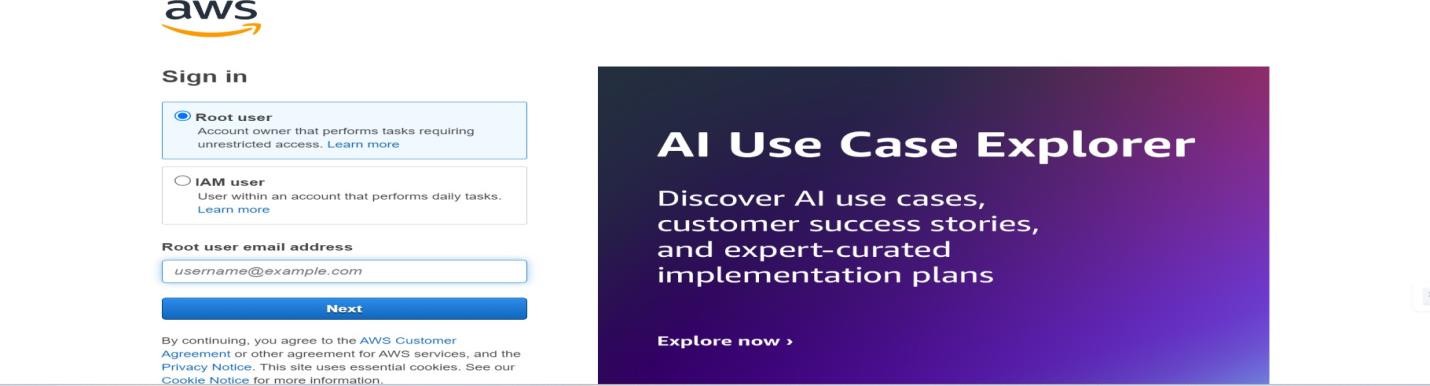
#### Activity 1.1: Set up an AWS account if not already done.

* + - Sign up for an AWS account and configure billing settings.



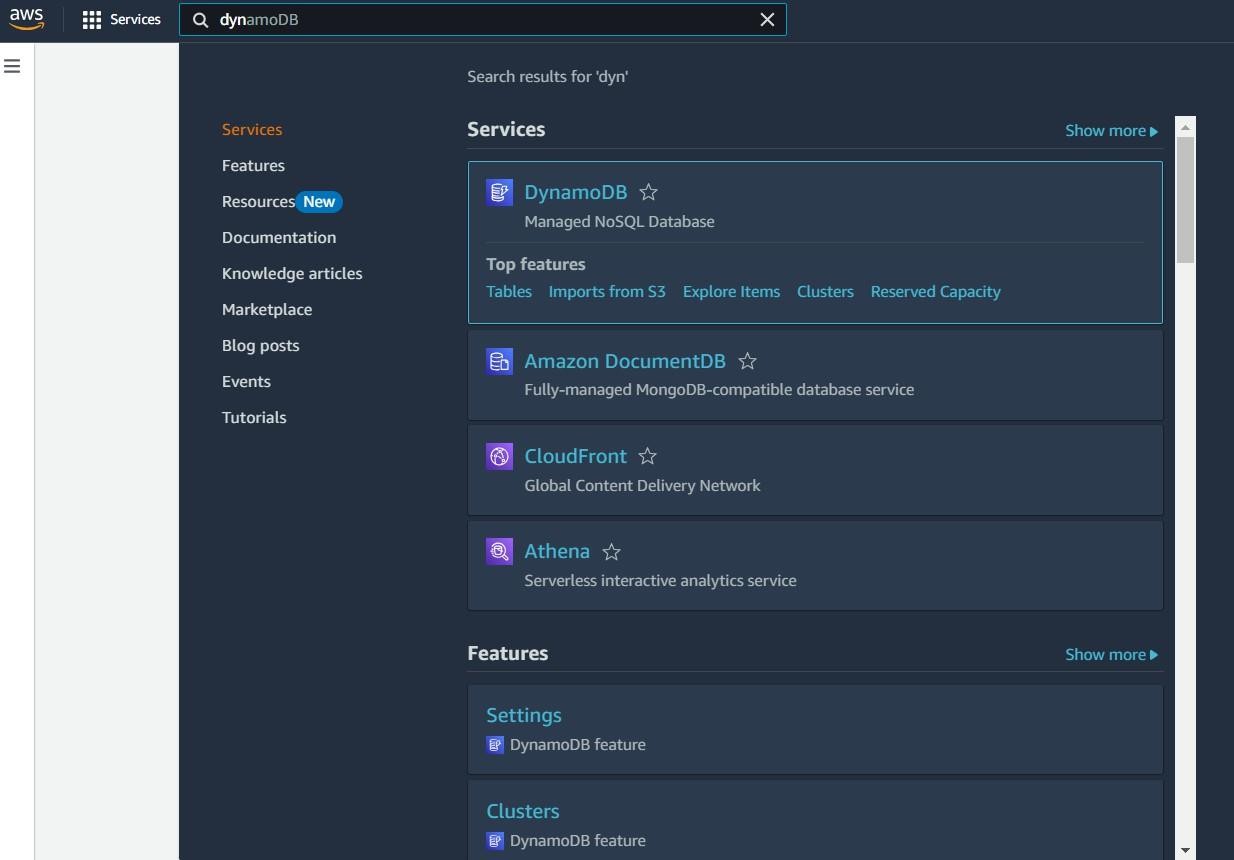
#### Activity 1.2: Log in to the AWS Management Console

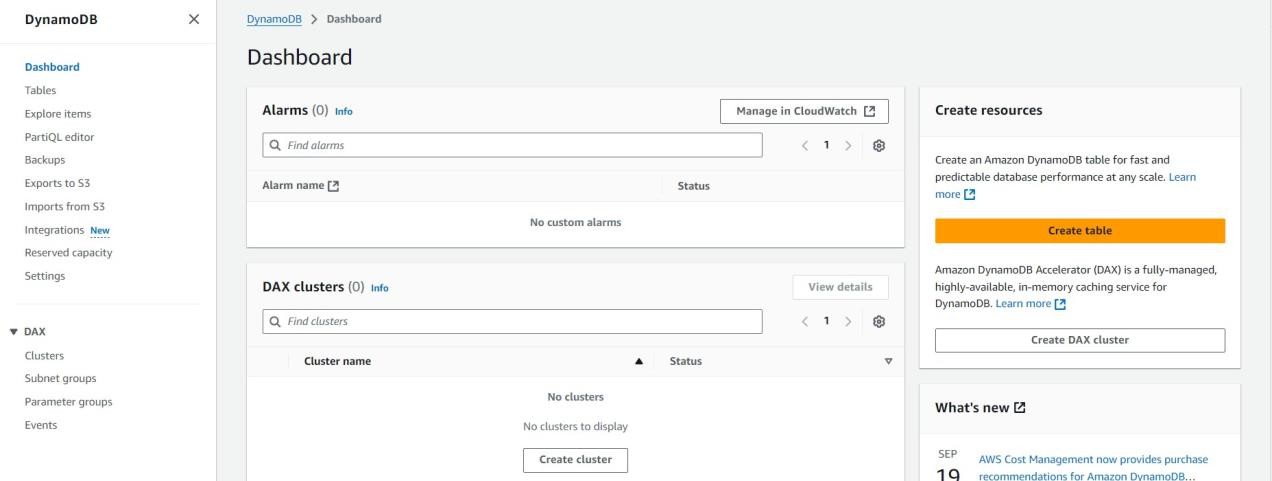
* + - After setting up your account, log in to the [AWS Management Console](https://aws.amazon.com/console/).

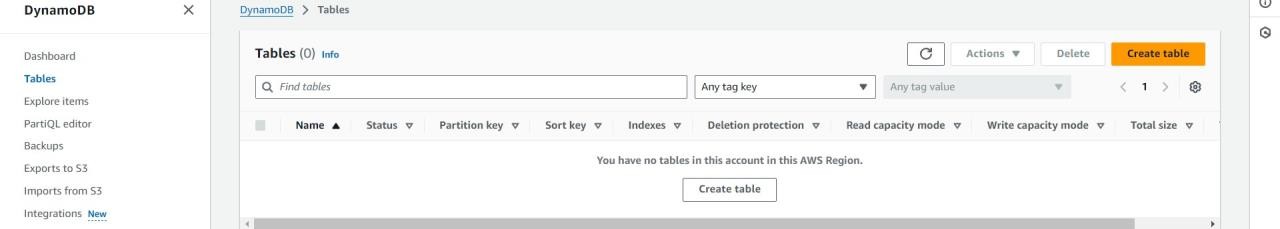


## Milestone 2: DynamoDB Database Creation and Setup

#### Activity 2.1:Navigate to the DynamoDB

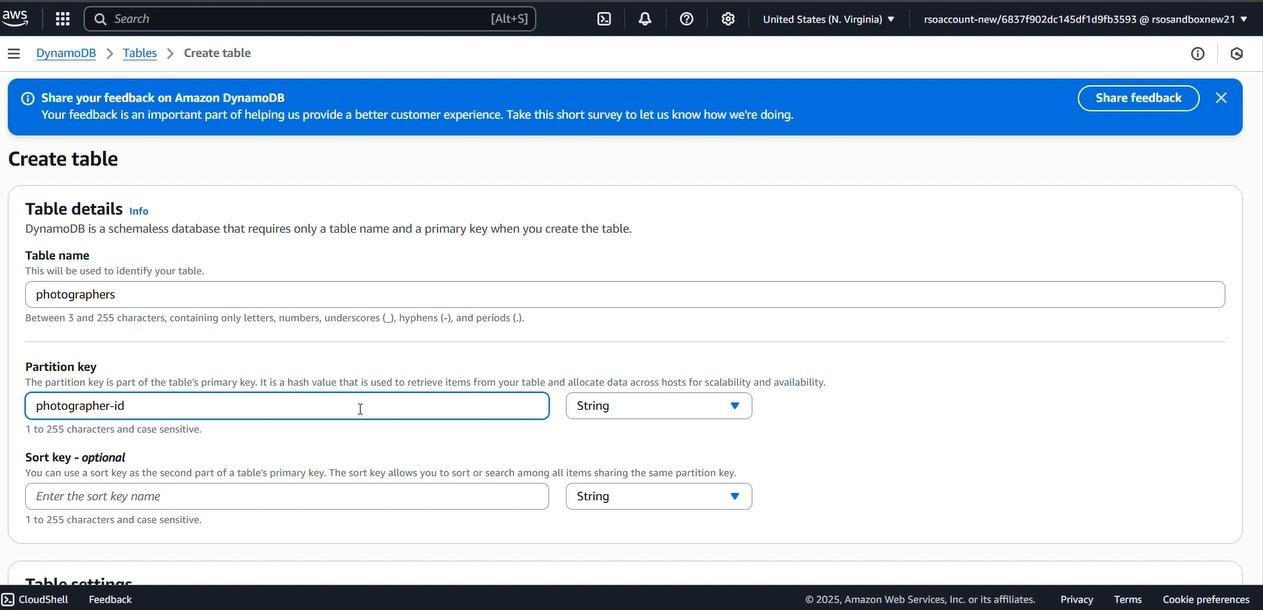
* + - In the AWS Console, navigate to DynamoDB and click on create tables.
    - 

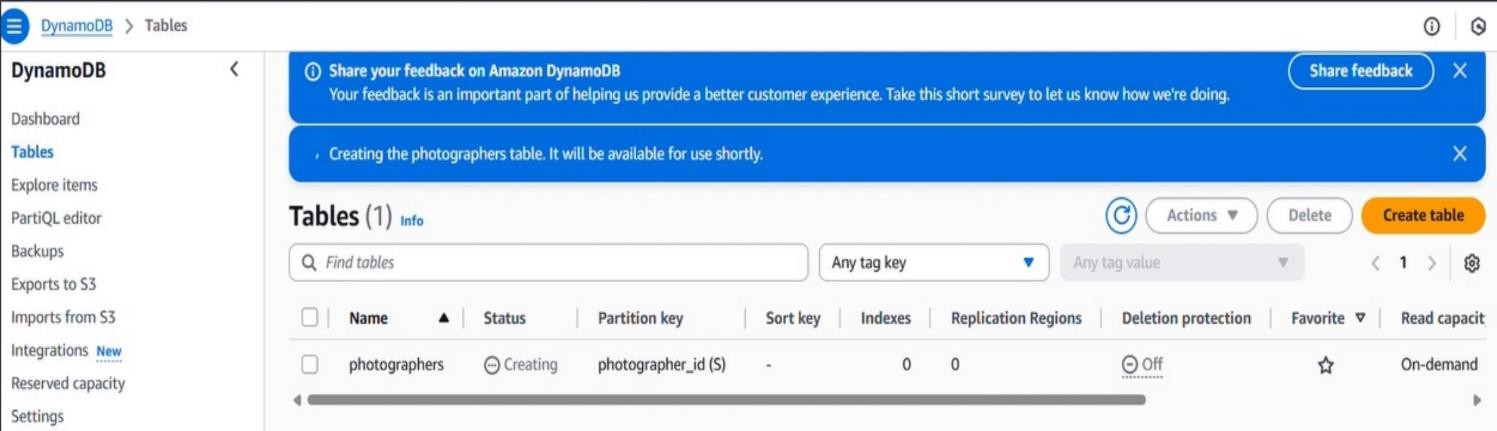
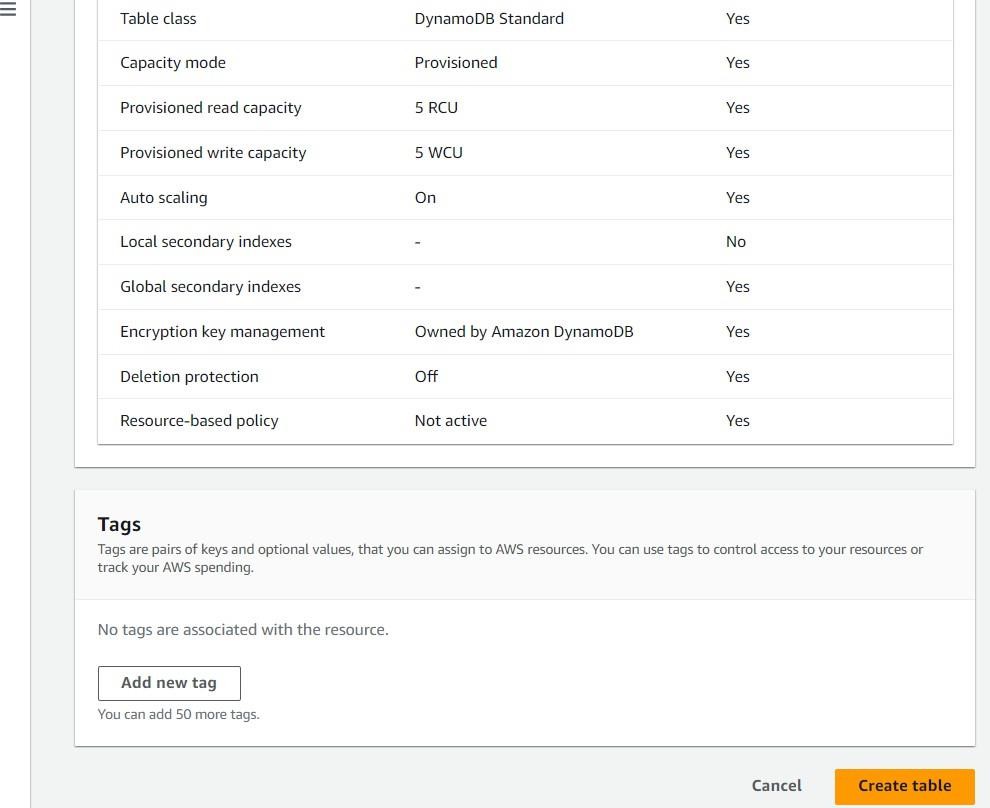




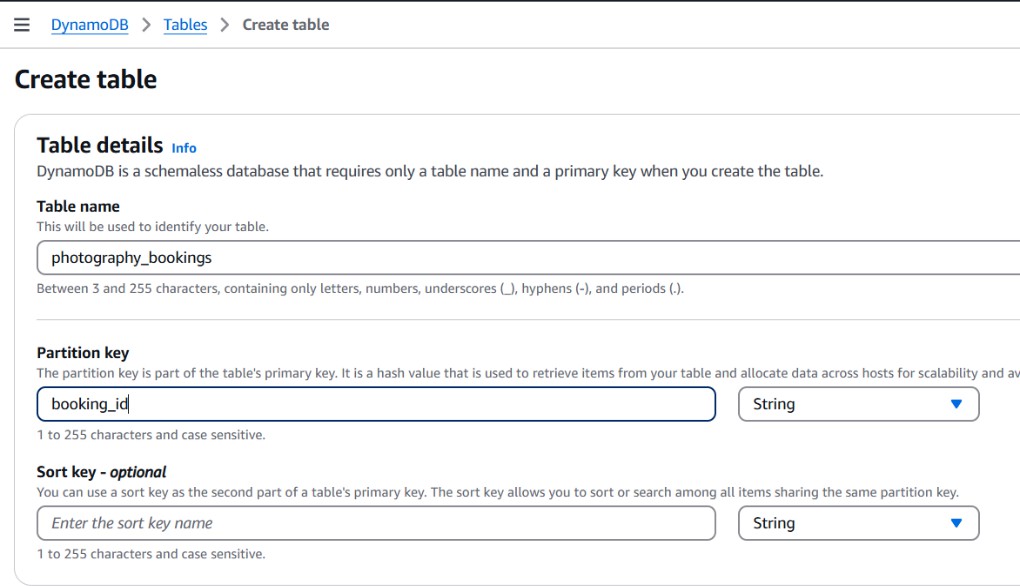
#### Activity 2.2:Create a DynamoDB table for storing registration details and book requests.

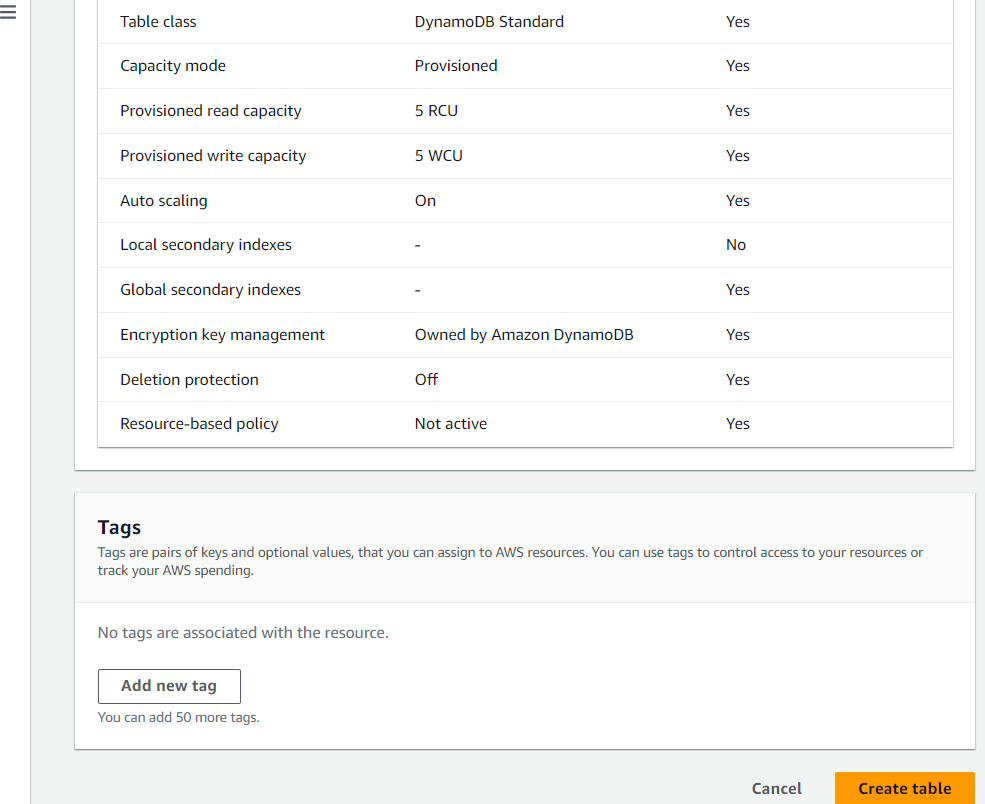
* + - Create photographers table with partition key “photographer\_id” with type String and click on create tables.

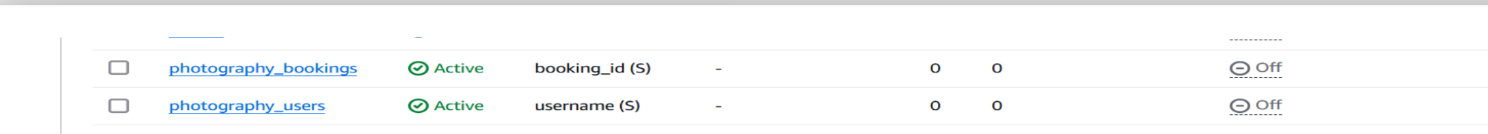




* + - Follow the same steps to create a requests table with photographer\_id as the primary key for book requests data.



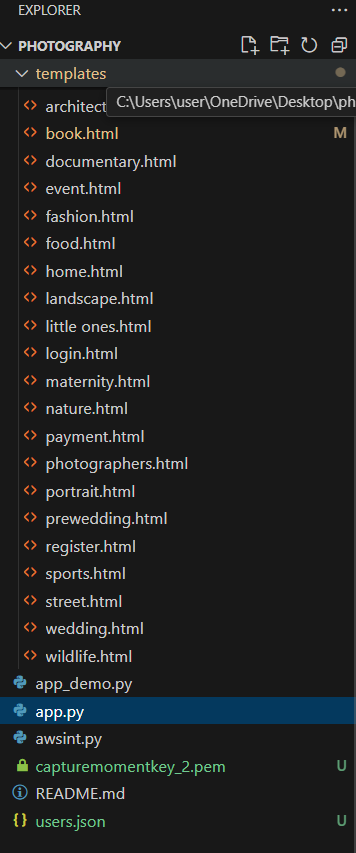




## Milestone 3:Backend Development and Application Setup

#### Activity 3.1: Develop the backend using Flask

* + - File Explorer Structure





**Description:** This is a Flask-based photography booking web application project. It includes multiple HTML templates for various photography categories like wedding, nature, fashion, and more. The project supports user registration, login, booking, and payment features. It also showcases photographer profiles and integrates with AWS services. The core logic is handled in Python scripts like app.py, and user data is managed using a JSON file.

### Description of the code :

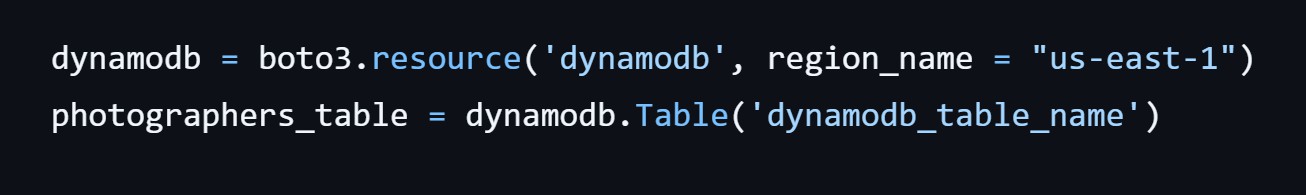
#### Flask App Initialization

**Description:** This code imports required libraries. boto3 is used to interact with AWS services like DynamoDB, and Flask and jsonify are imported from the Flask framework to create a web application and return JSON responses.



**Description:** initialize the Flask application instance using Flask(name) to start building the web app.

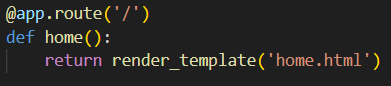
#### Dynamodb Setup:

****

**Description:** initialize the DynamoDB resource for the us-east-1 region and set up access to the photographers table for storing details.

### Routes for Web Pages

#### Home Route:



**Description:** This code defines the home route ('/') of a Flask web application. When users access the root URL, the home() function is called and it renders and returns the home.html template as the response**.**

#### Register Route:

**Description:** This code defines a /register route in a Flask app that handles both GET and POST requests. It processes user registration by collecting form data (username, email, password), checks if the user already exists, hashes the password for security, saves the user data, and then redirects to the login page with a success message. If the method is GET, it simply renders the registration form.

#### login Route (GET/POST):

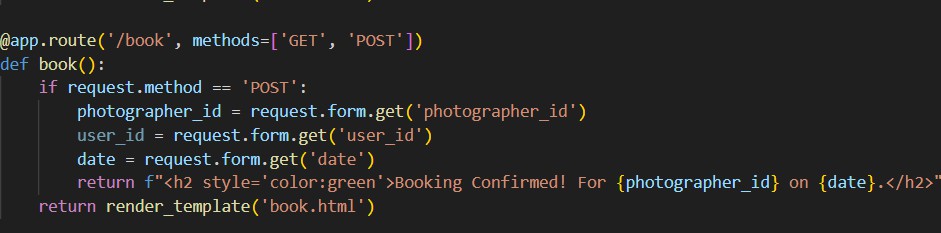
**Description:** This code defines a /login route in a Flask app that handles both GET and POST requests. When a user submits the login form (POST), it retrieves the entered credentials, verifies the user using a lookup function, and checks the password hash. If valid, it stores the username in the session and redirects to the home page with a success message. If authentication fails, it flashes an error message and reloads the login page. On a GET request, it simply displays the login.html template.

#### Home, E- book buttons and subject routes:





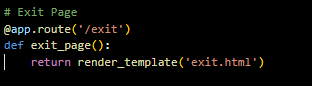
**Description:** This code is part of a Flask web application that manages routing for different types of photography pages. Each route is associated with a specific category such as wedding, fashion, maternity, wildlife, etc. When a user visits a particular URL, Flask looks at the defined routes, matches the one requested, and then executes the corresponding function. This function uses render\_template() to return an HTML page that represents the content for that photography style.

* + **Request Routes:**

**Description:** This Python code is part of a Flask web application designed for a photography website. It defines individual routes for various photography categories such as wedding, event, fashion, maternity, nature, wildlife, and more. Each route corresponds to a specific photography type and loads an HTML page using the render\_template() function. For example, when a user visits

/wedding, the wedding() function is executed and displays wedding.html, which contains content related to wedding photography**.**

#### Exit Route:

****

**Description:** define /exit route to render the exit.html page when the user chooses to leave or close the application.

**Deployment Code:**

****

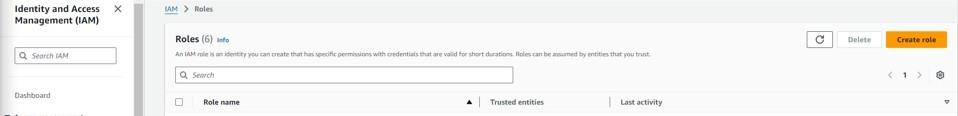
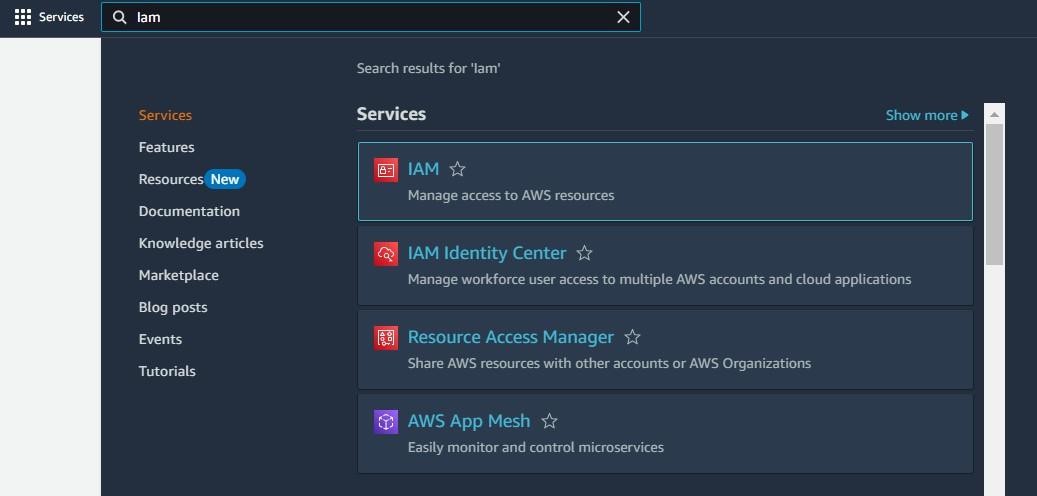
**Description:** start the Flask server to listen on all network interfaces (0.0.0.0) at port 80

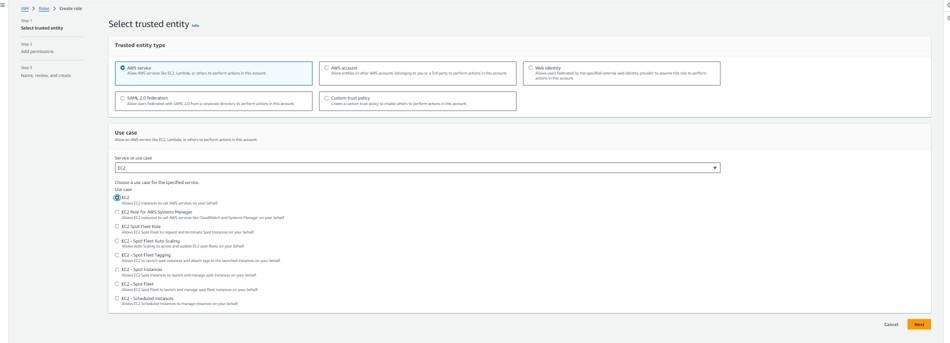
with debug mode enabled for development and testing.

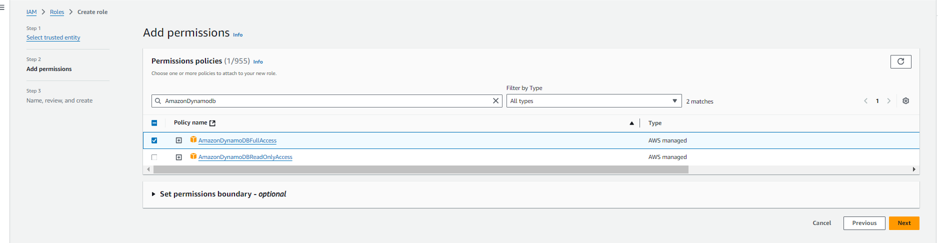
## Milestone 5: IAM Role Setup

#### Activity 5.1:Create IAM Role.

* In the AWS Console, go to IAM and create a new IAM Role for EC2 to interact with DynamoDB and SNS.



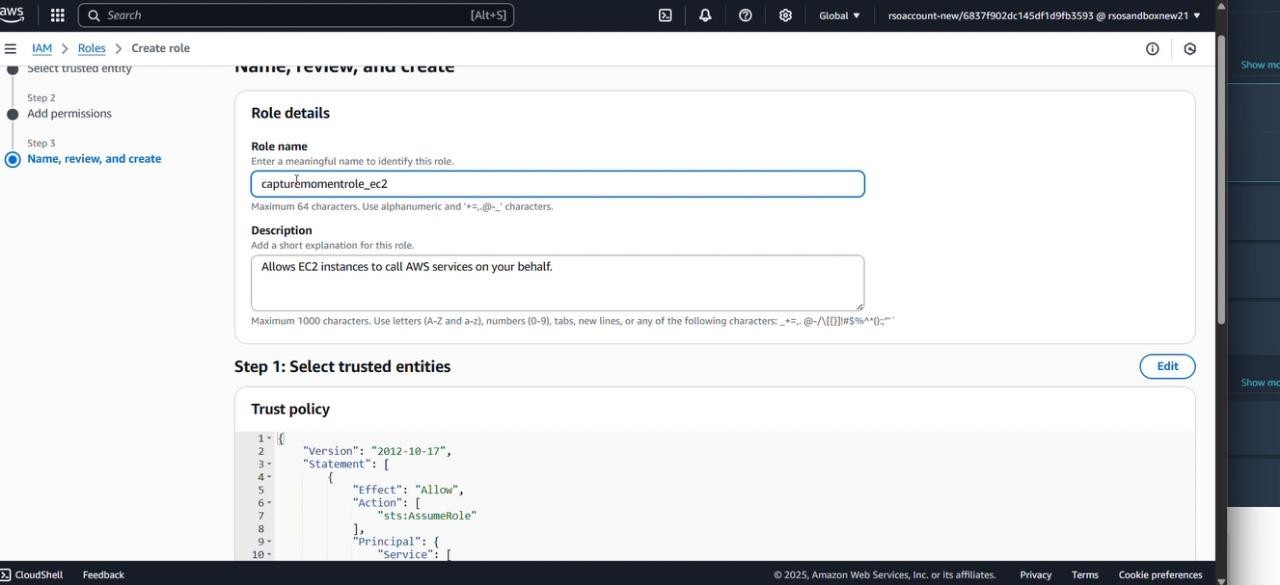




#### Activity 5.2: Attach Policies.

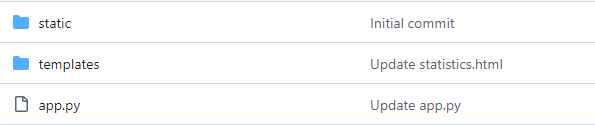
Attach the following policies to the role:

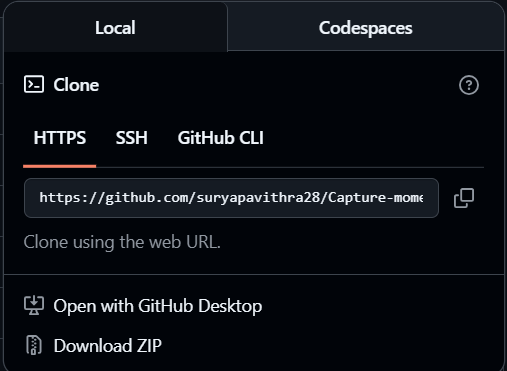
* + AmazonDynamoDBFullAccess: Allows EC2 to perform read/write operations on DynamoDB.



## Milestone 6: EC2 Instance Setup

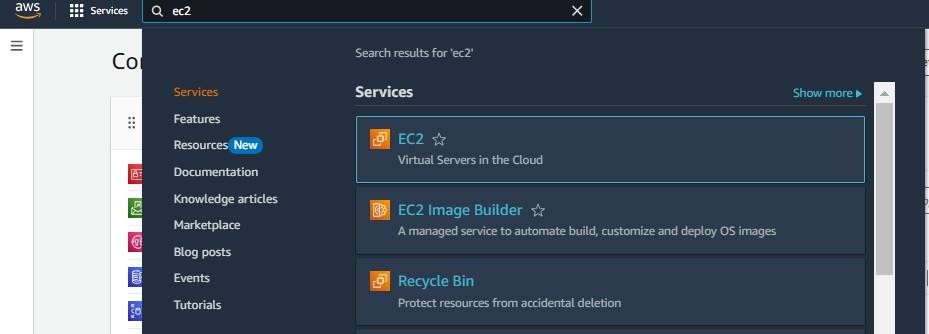
* **Note: Load your Flask app and Html files into GitHub repository.**

****

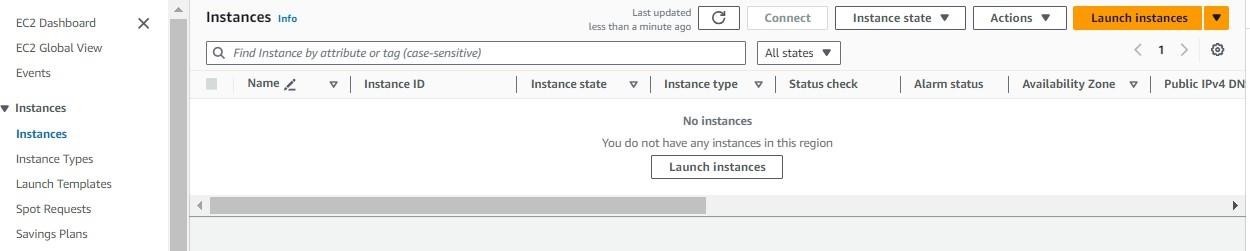
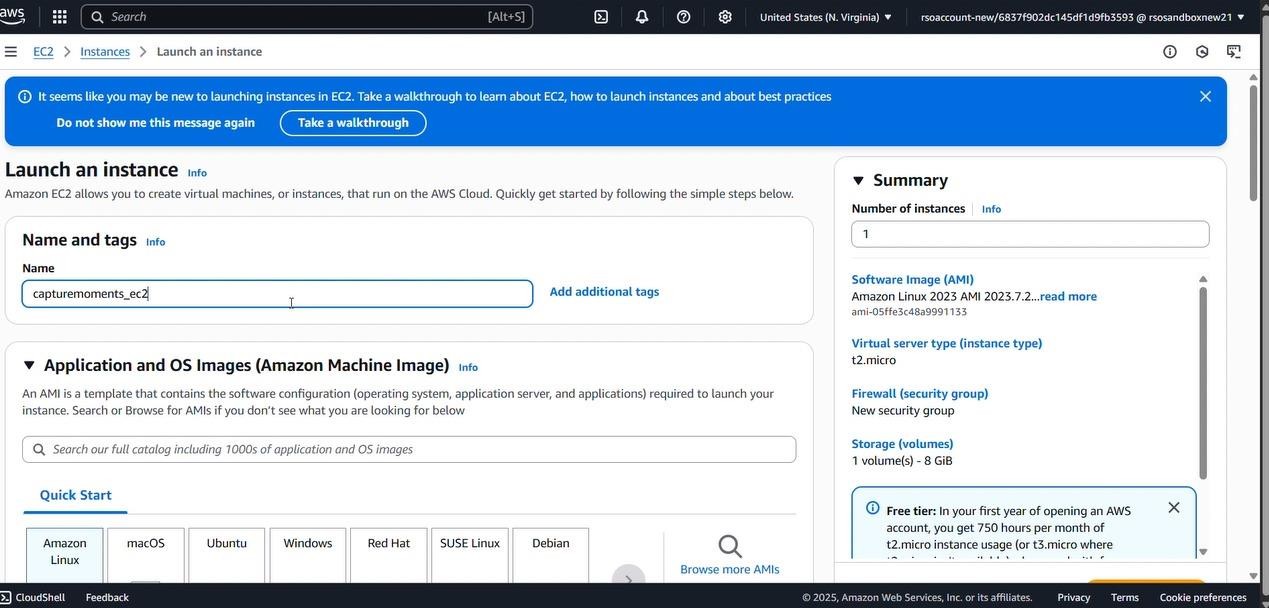


#### Activity 6.1: Launch an EC2 instance to host the Flask application.

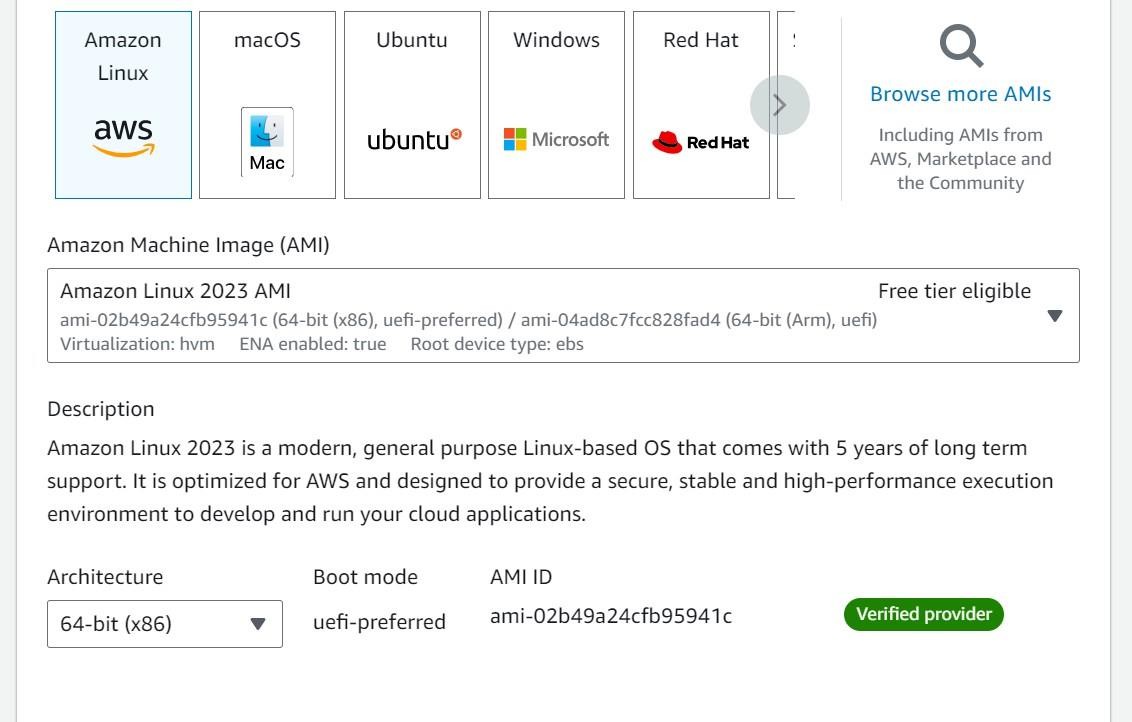
* + **Launch EC2 Instance**
    - In the AWS Console, navigate to EC2 and launch a new instance.



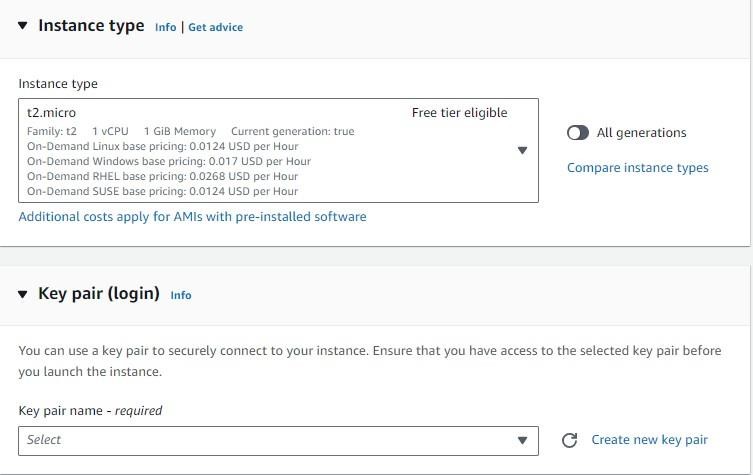
* Click on Launch instance to launch EC2 instance

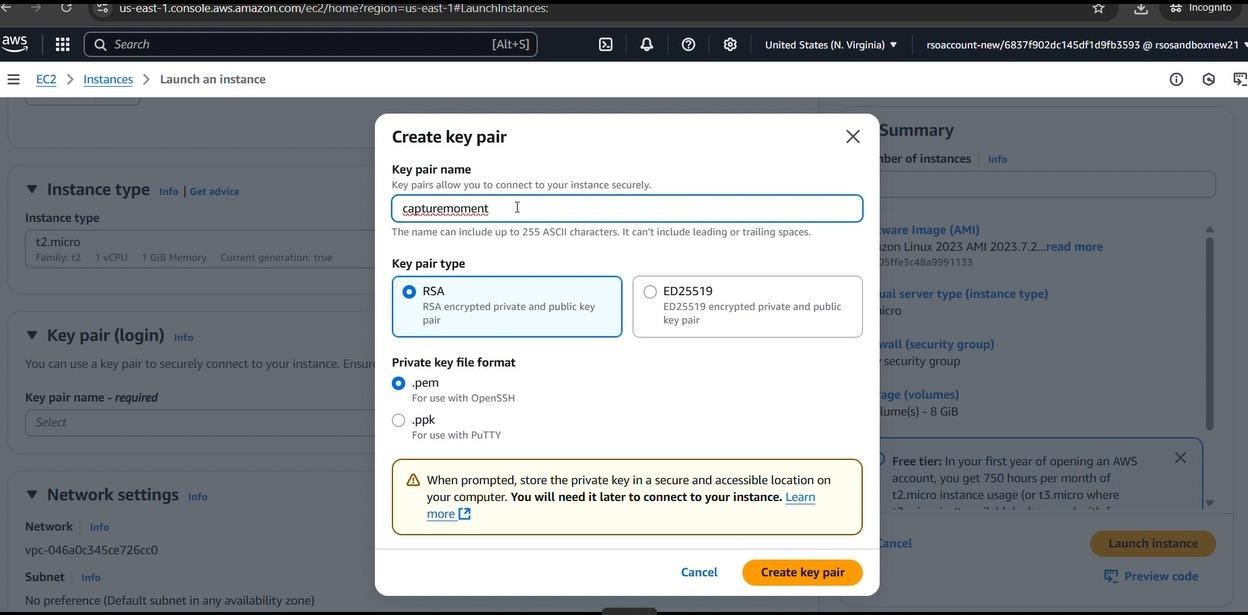


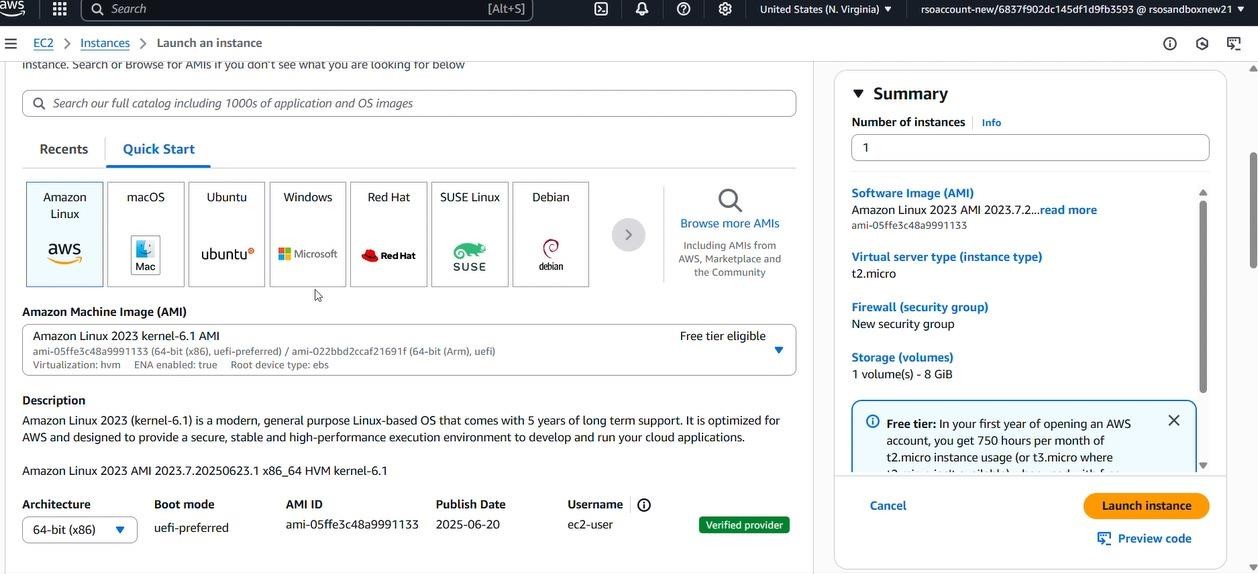
* + Choose Amazon Linux 2 or Ubuntu as the AMI and t2.micro as the instance type (free-tier eligible).



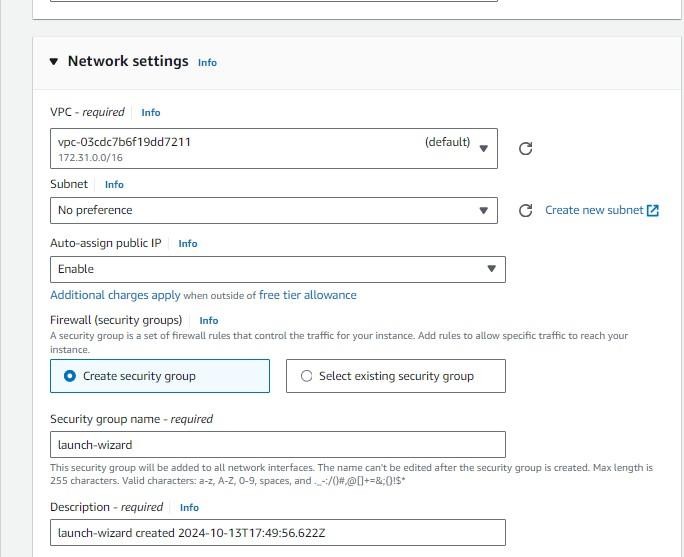
* Create and download the key pair for Server access.

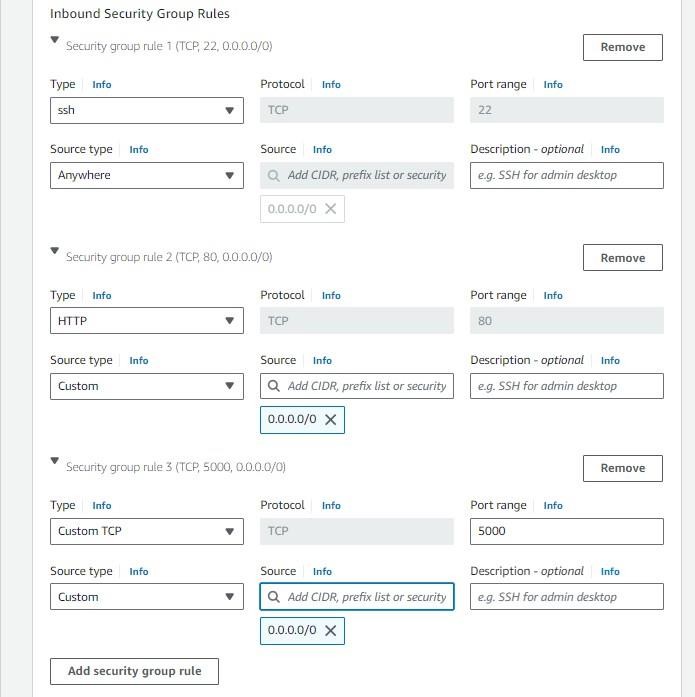


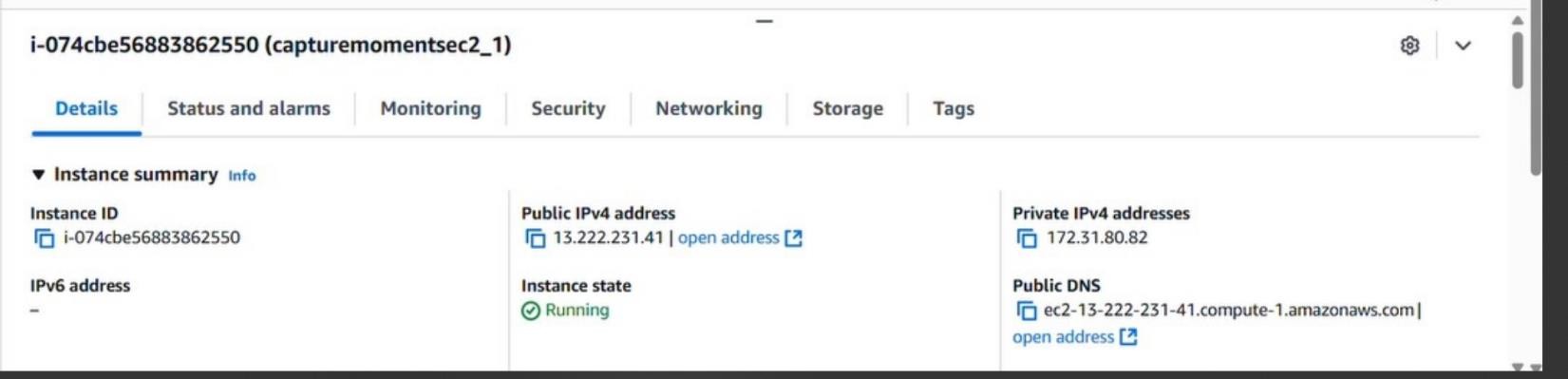


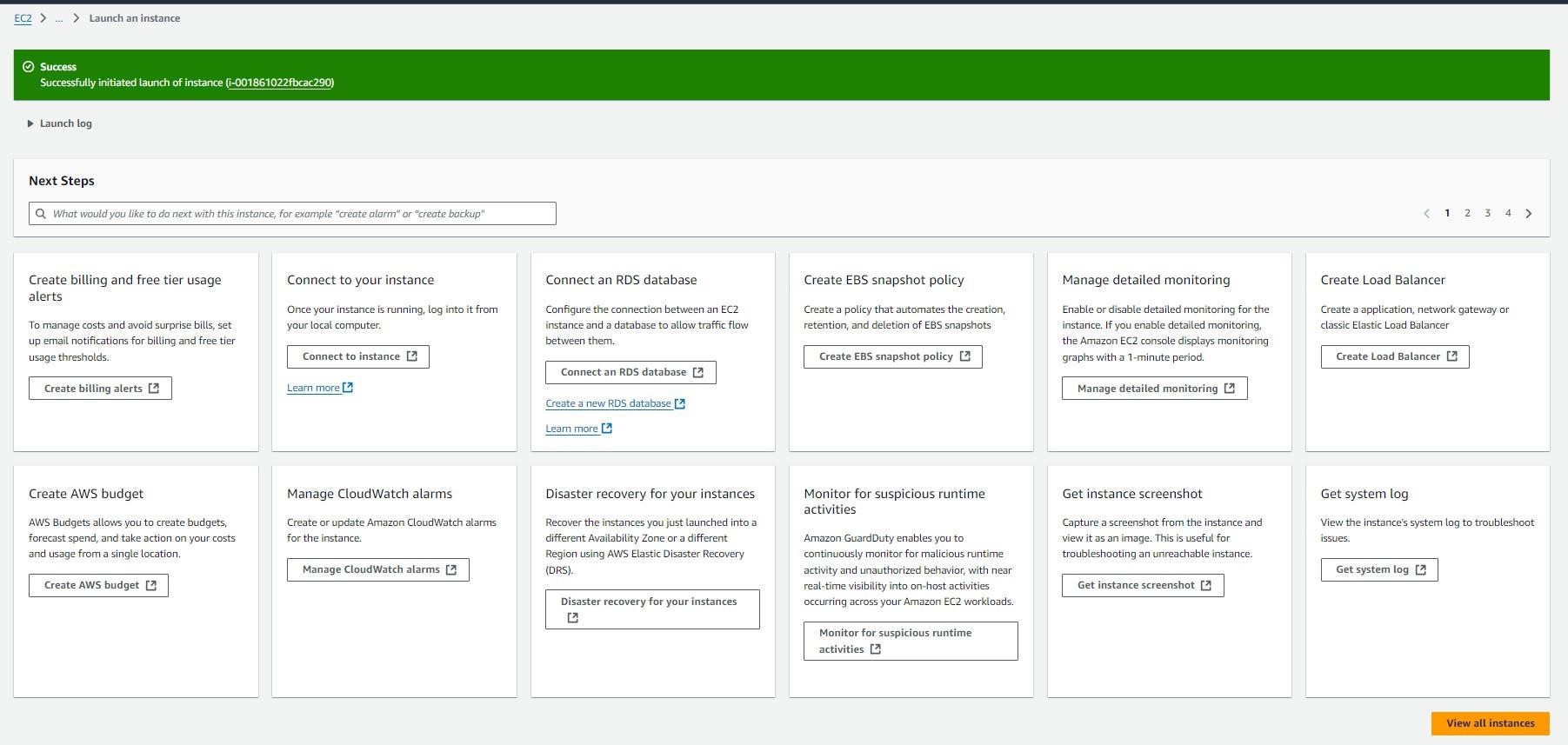


#### Activity 6.2:Configure security groups for HTTP, and SSH access.

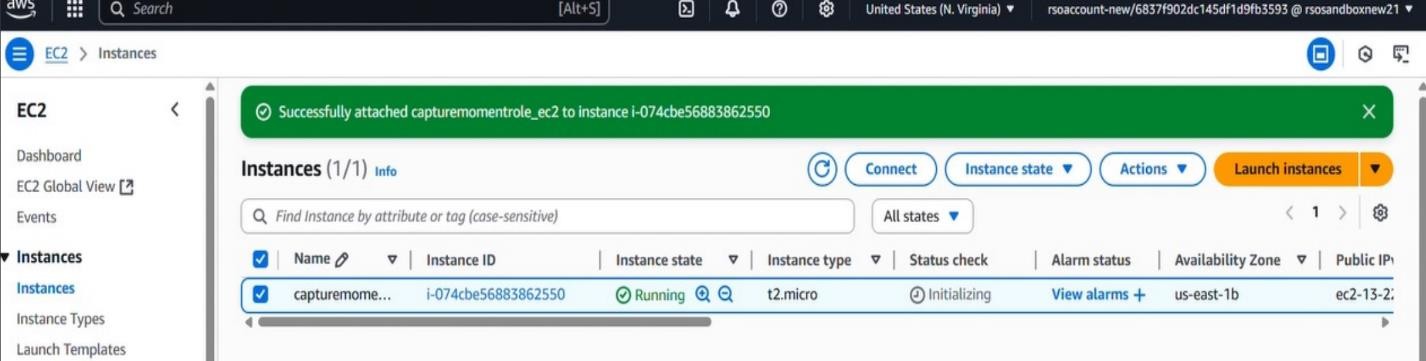
****

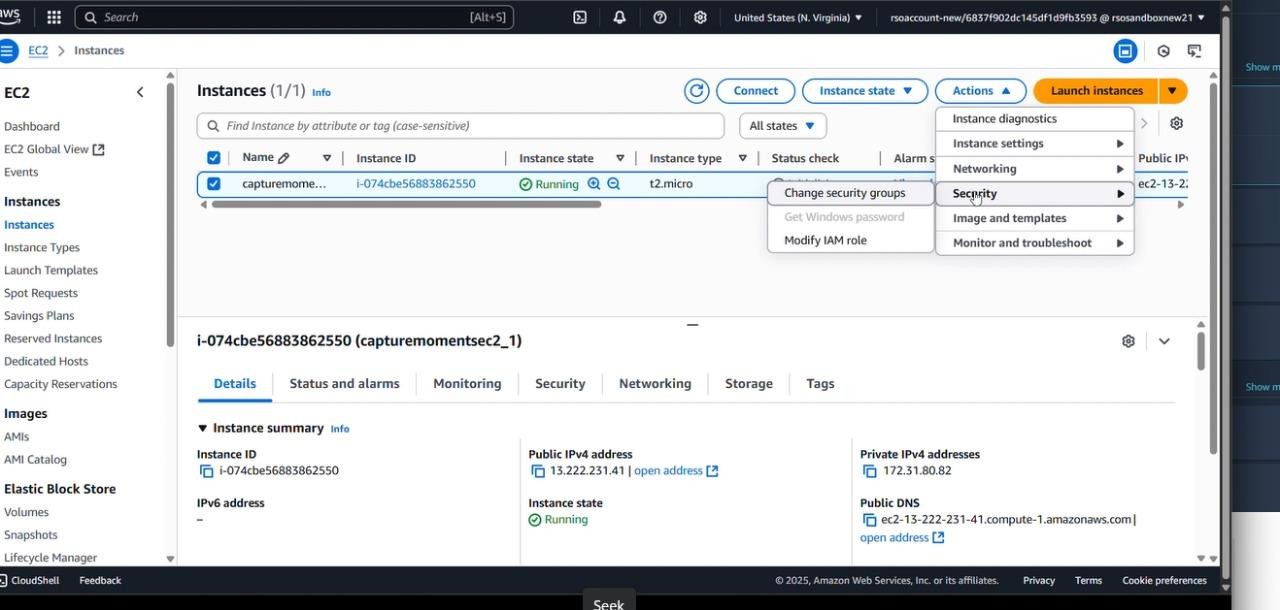
****

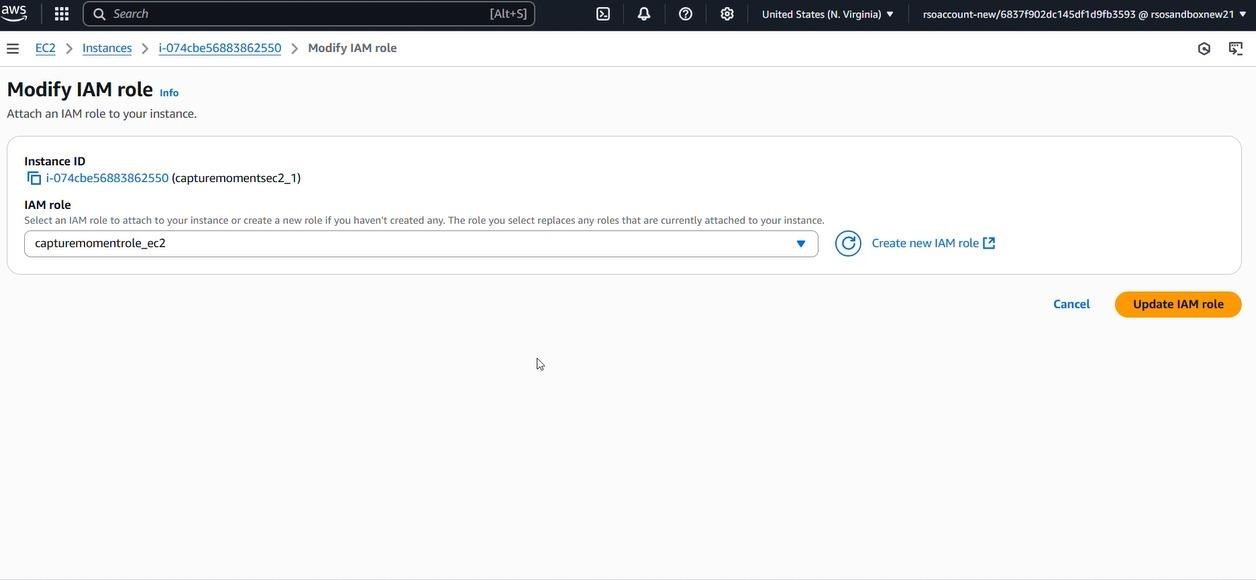
****



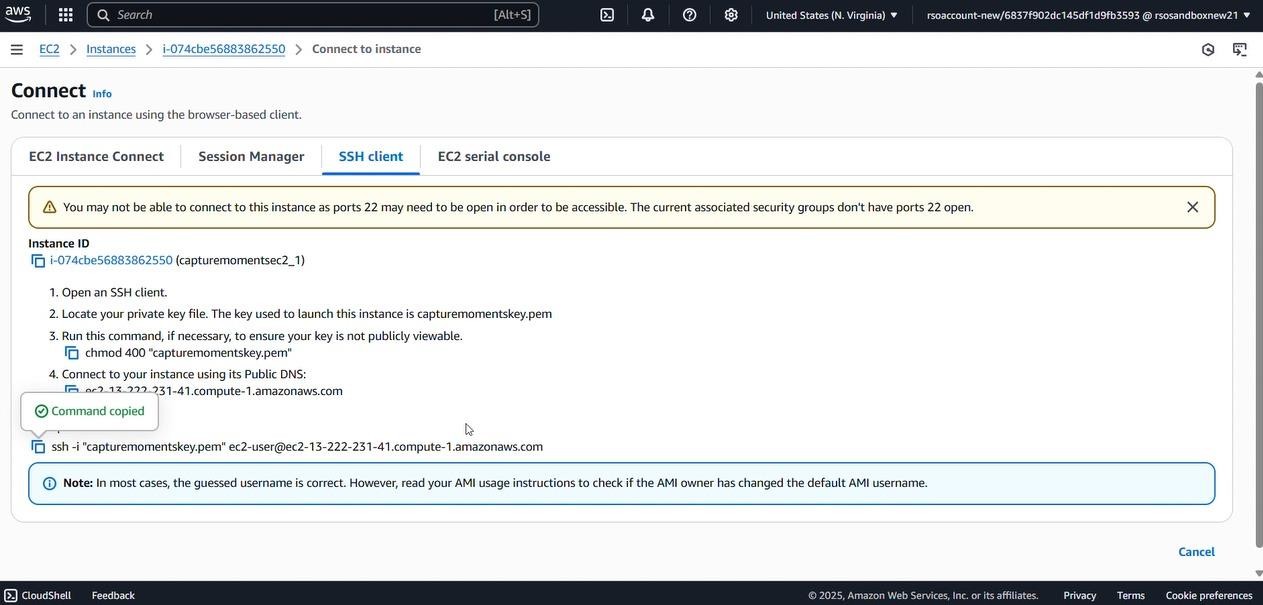
* To connect to EC2 using **EC2 Instance Connect**, start by ensuring that an **IAM role** is attached to your EC2 instance. You can do this by selecting your instance, clicking on **Actions**, then navigating to **Security** and selecting **Modify IAM Role** to attach the appropriate role. After the IAM role is connected, navigate to the **EC2** section in the **AWS Management Console**. Select the **EC2 instance** you wish to connect to. At the top of the **EC2 Dashboard**, click the **Connect** button. From the connection methods presented, choose **EC2 Instance Connect**. Finally, click **Connect** again, and a new browser-based terminal will open, allowing you to access your EC2 instance directly from your browser.







* Now connect the EC2 with the files





**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/suryapavithra28/Capture-moments.git

Note: change your-github-username and your-repository-name with your credentials

* This will download your project to the EC2 instance.

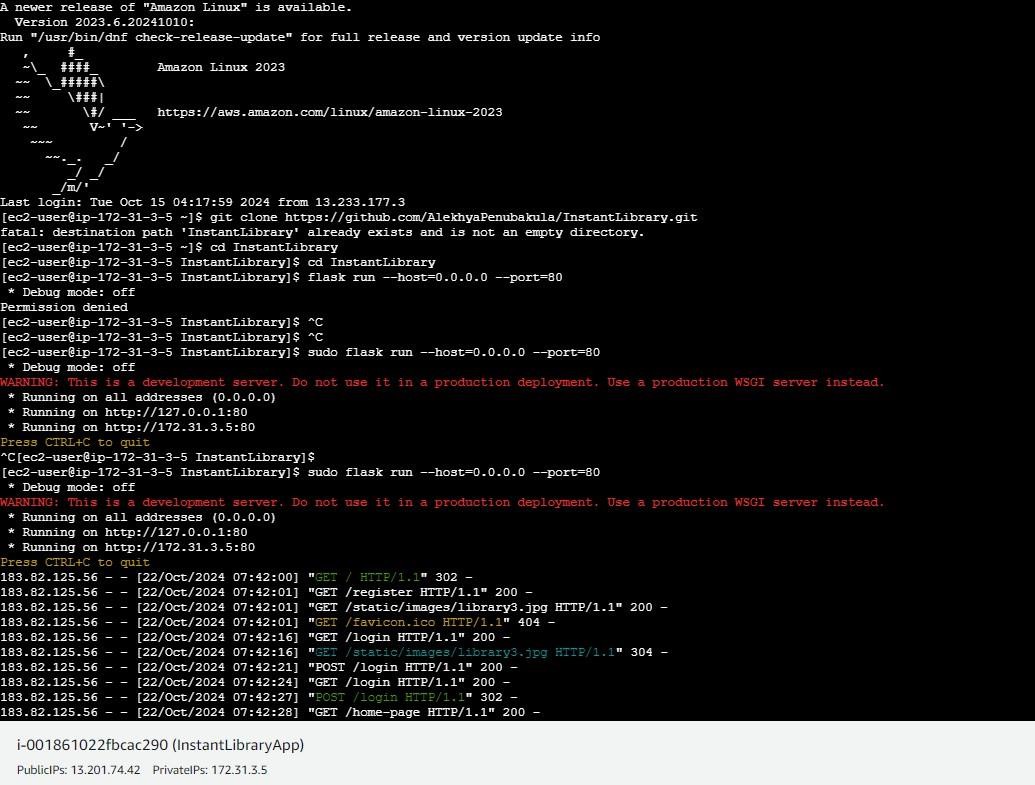
#### To navigate to the project directory, run the following command:

cd "C:\Users\user\OneDrive\Desktop\photography"

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

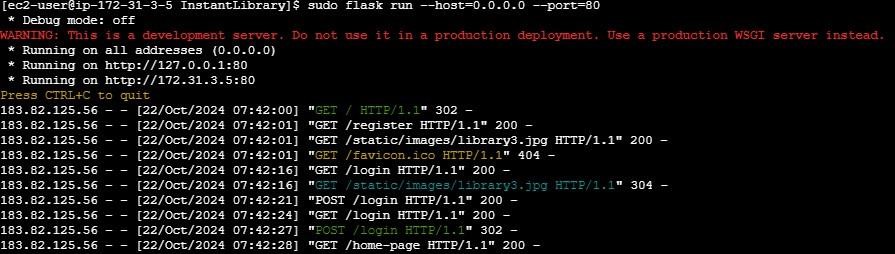
**Run the Flask Application**

Python app.py



**Verify the Flask app is running**: [http://your-ec2-public-ip](http://your-ec2-public-ip/)

* Run the Flask app on the EC2 instance



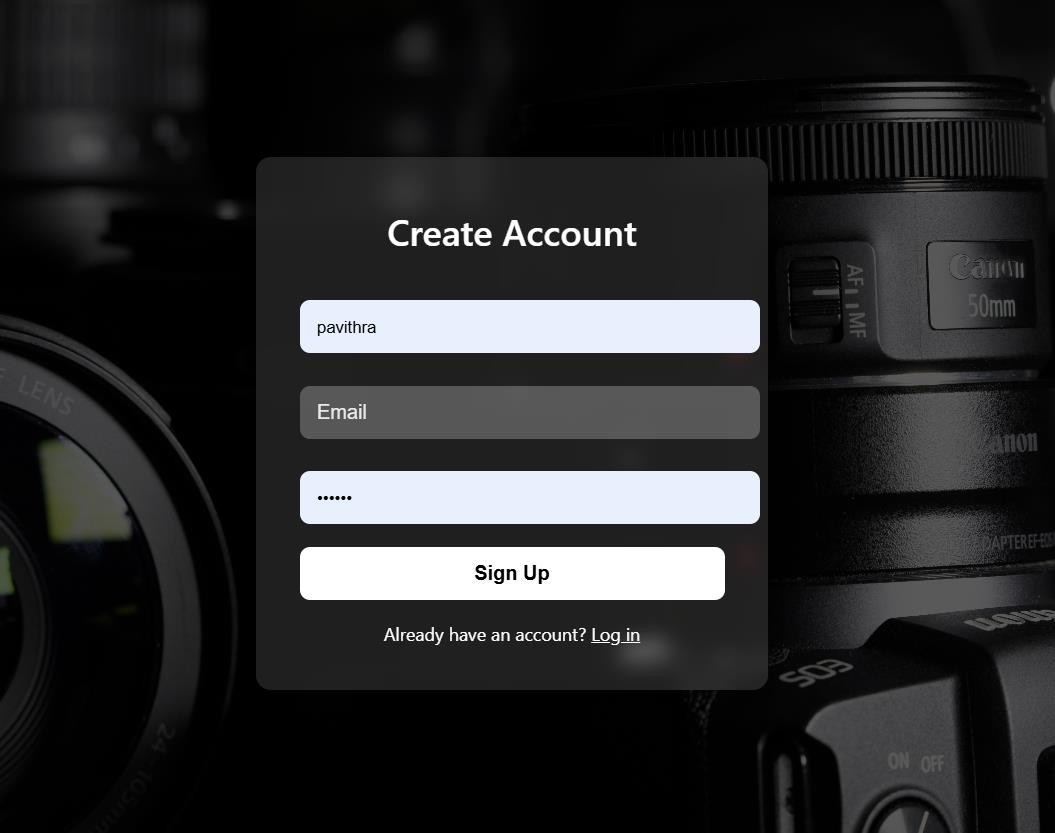
#### Access the website through:

**Public IPs:** [**http://54.161.227.172:5000**](http://54.161.227.172:5000/)

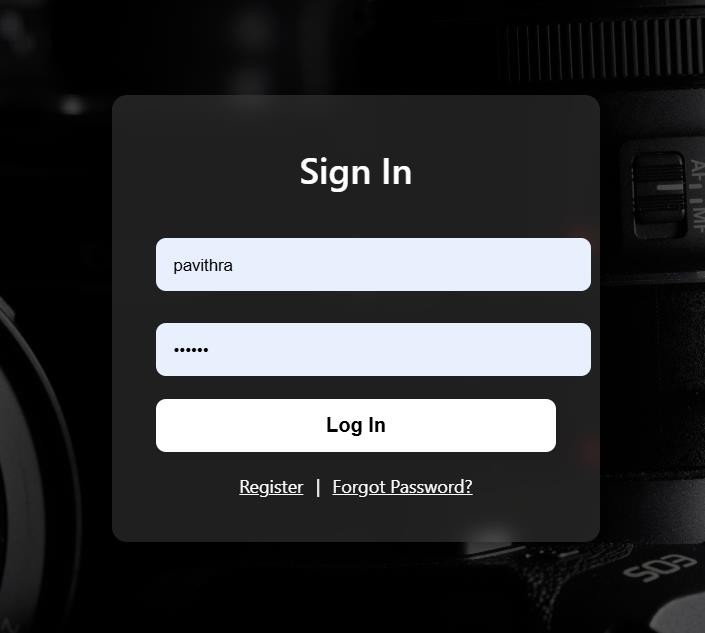
### Milestone 8: Testing and Deployment

#### Activity 8.1: Conduct functional testing to verify user registration, login, book requests, and notifications.

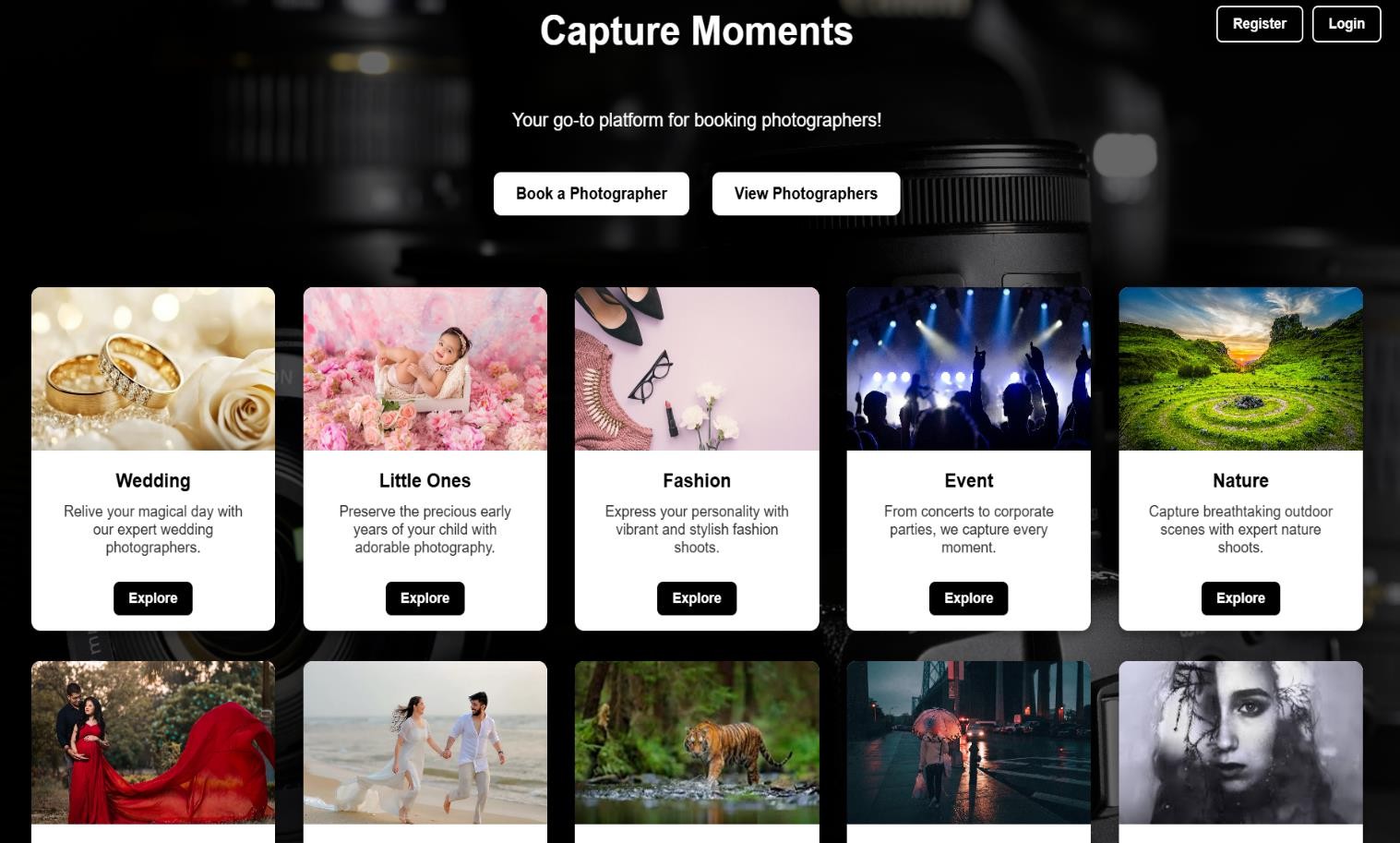
**Login Page:**

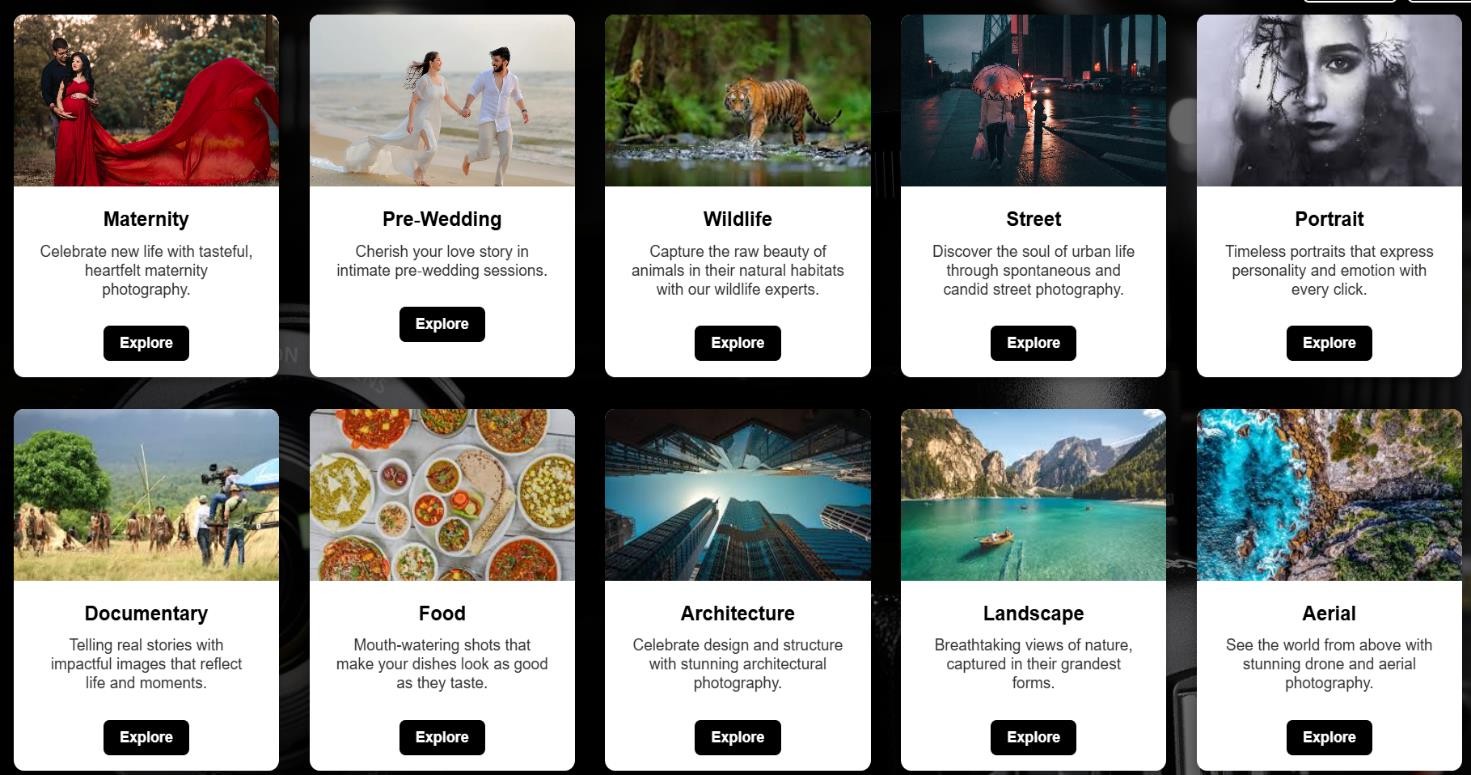
****

**Register Page:**



**Home page:**

****



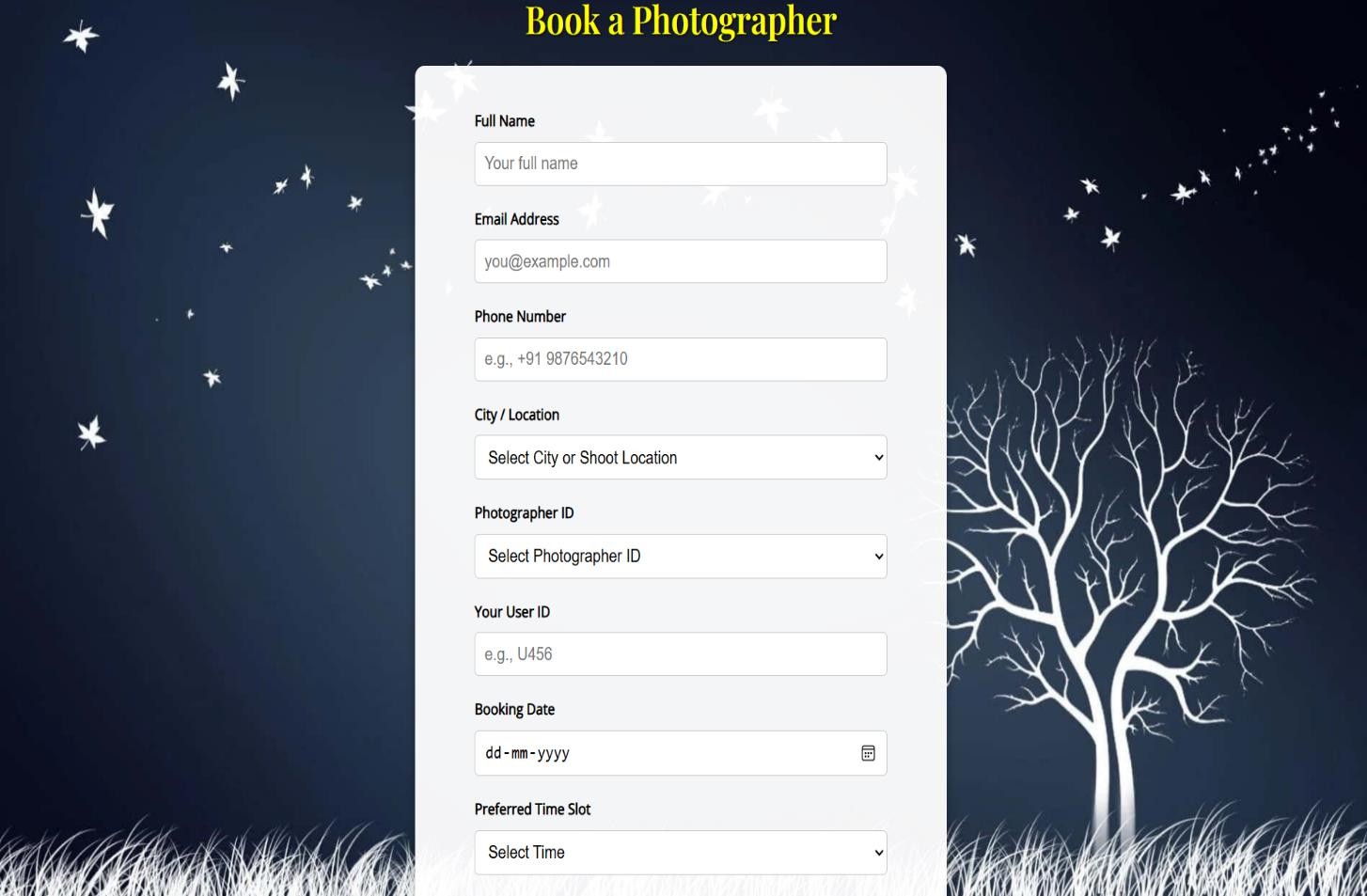
**About Us page:**

****

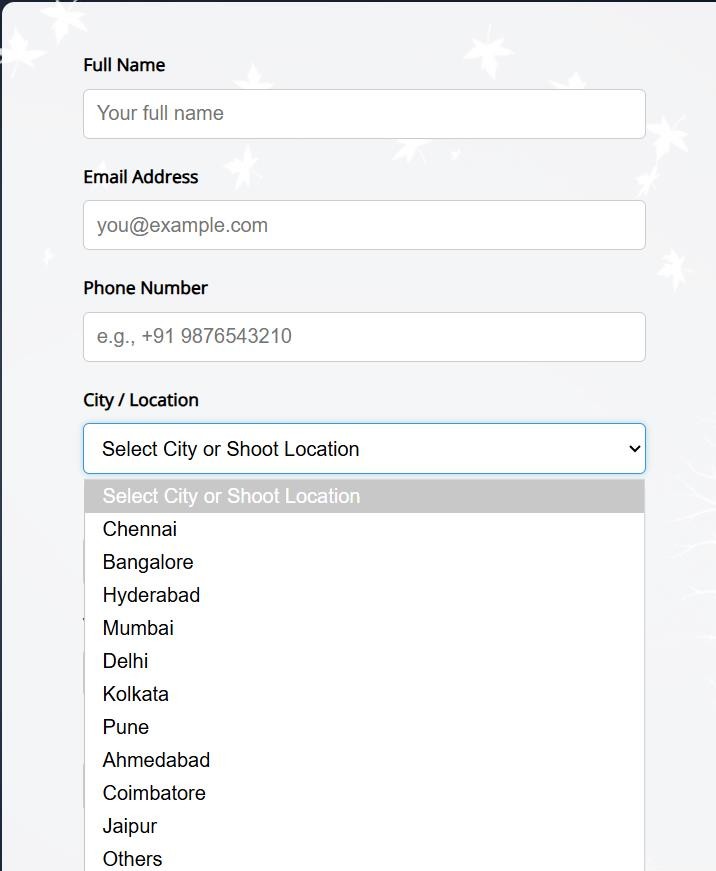
**Contact Page:**



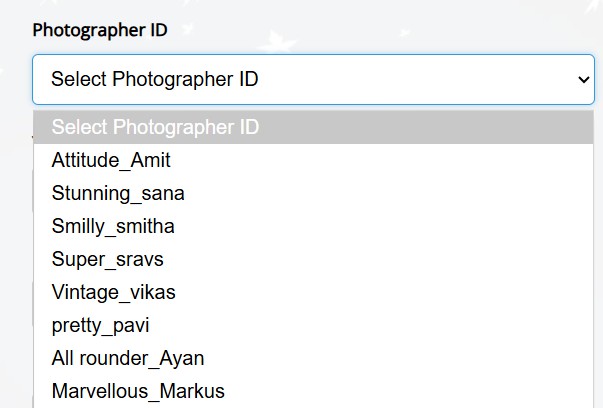
**Book a photographer page:**

****

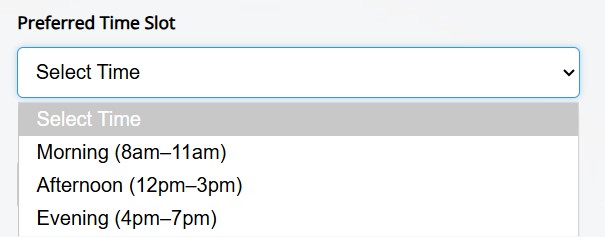
**Book a photographer location Page :**

****

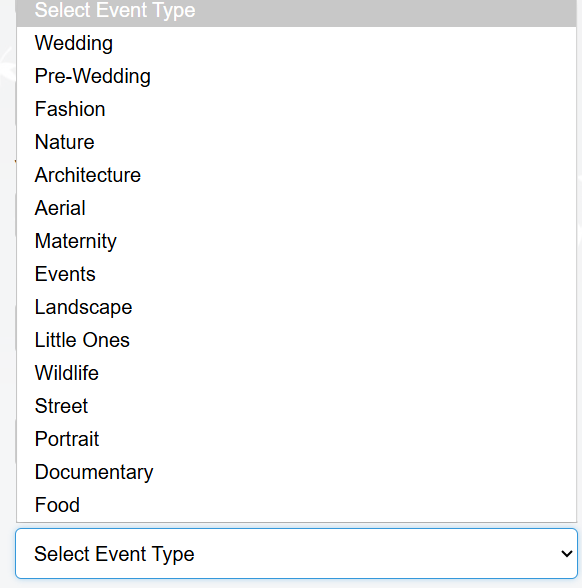
**Photographer ID page :**

****

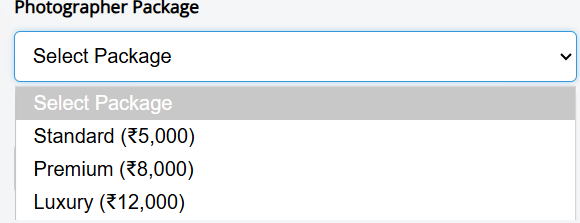
**Preferred Time Slot Page:**

****

**Event type Page:**

****

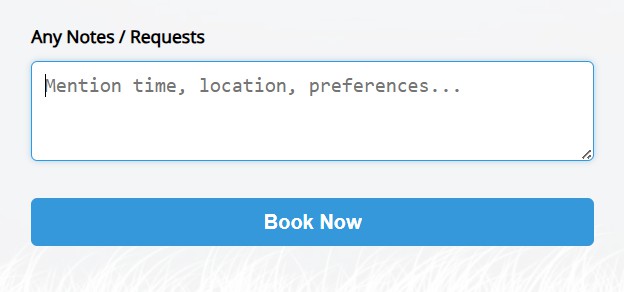
**Photographer package Page:**

****

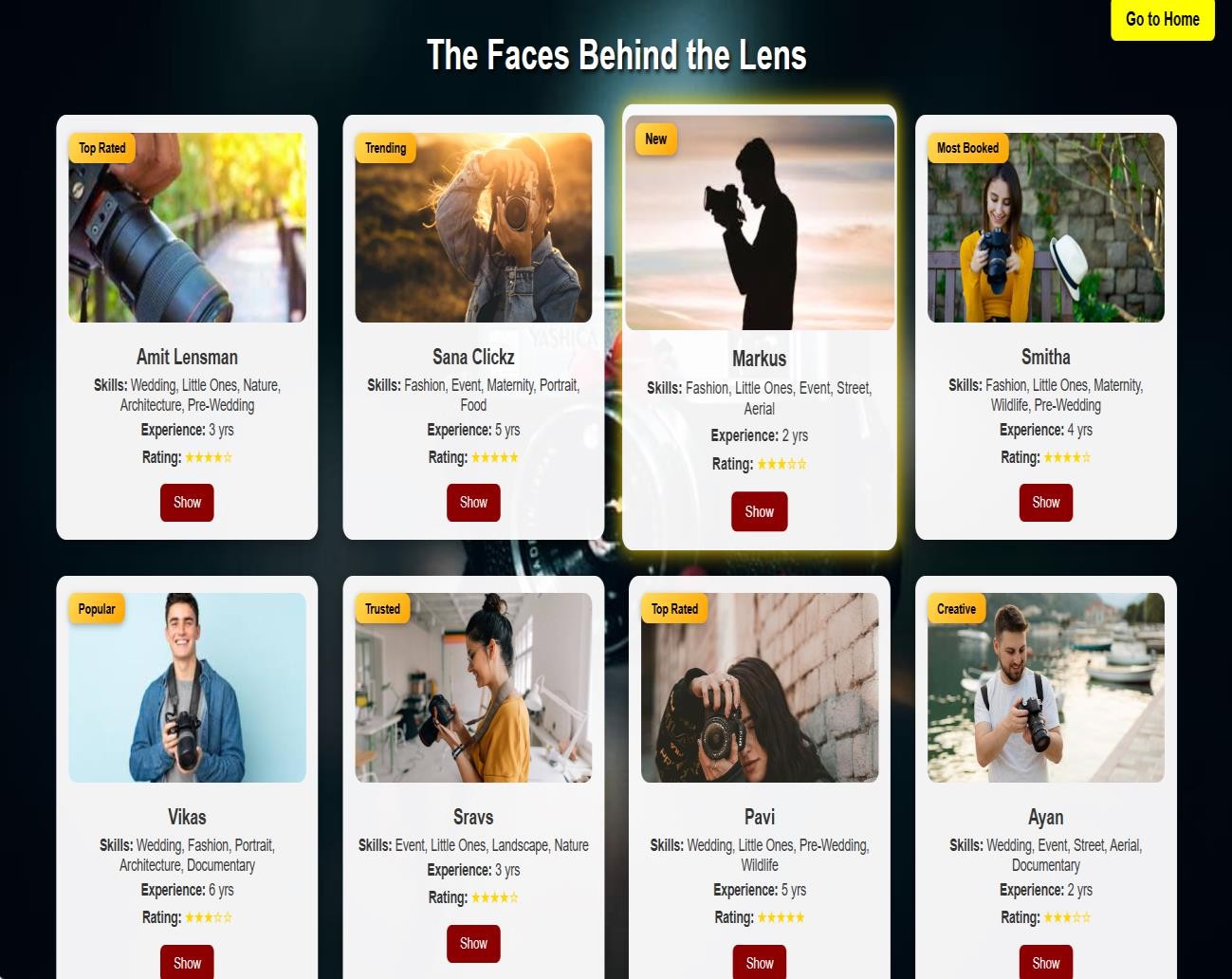
**Payment method page:**

****

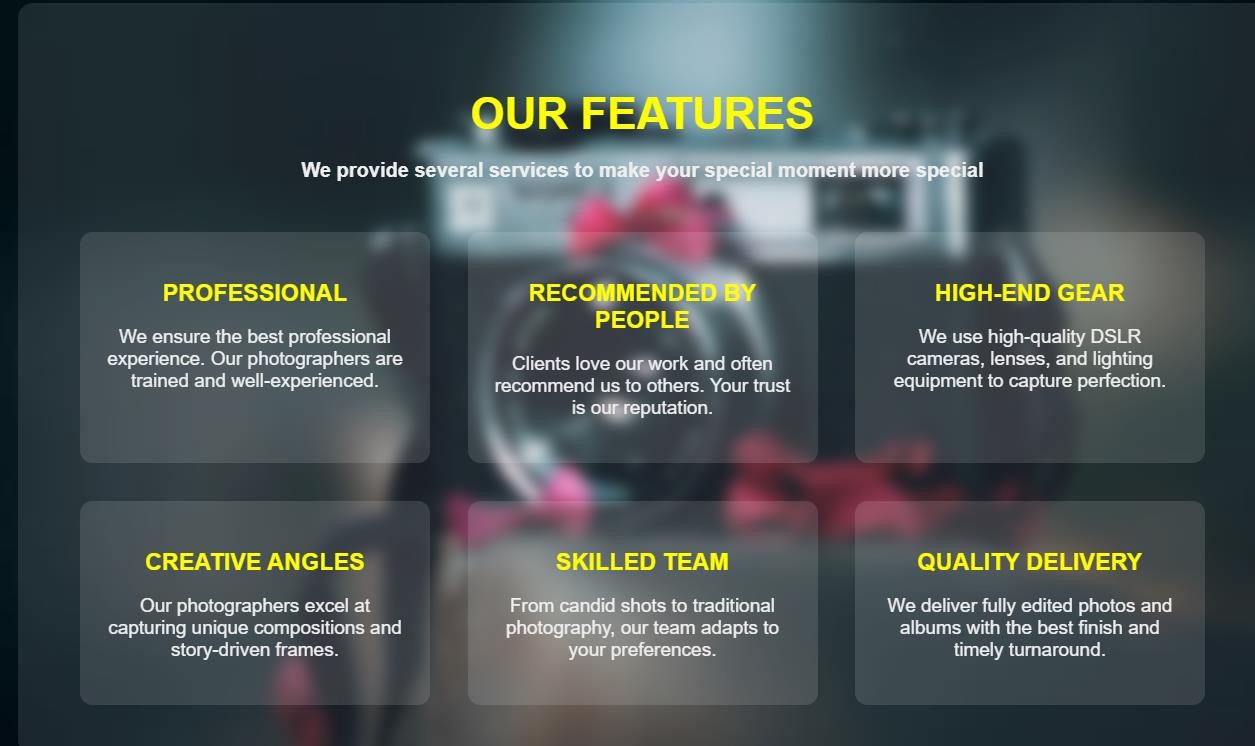
**Request Page:**

****

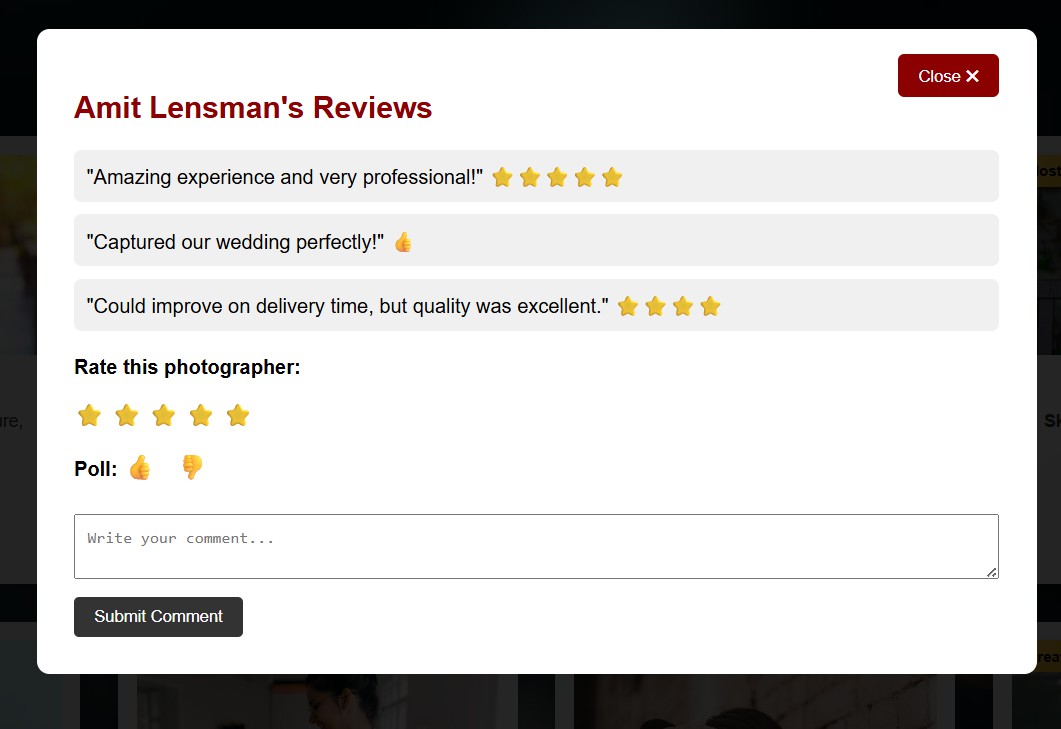
**VIEW PHOTOGRAPHERS PAGE:**

****

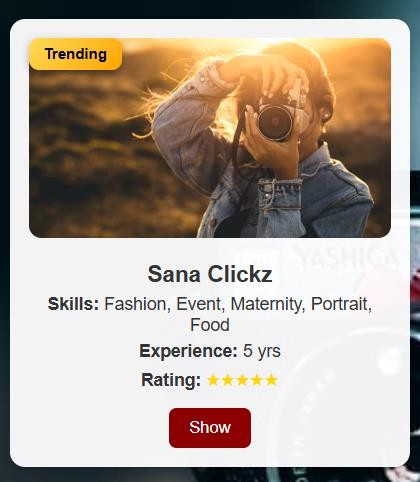
# Our Features page:

****

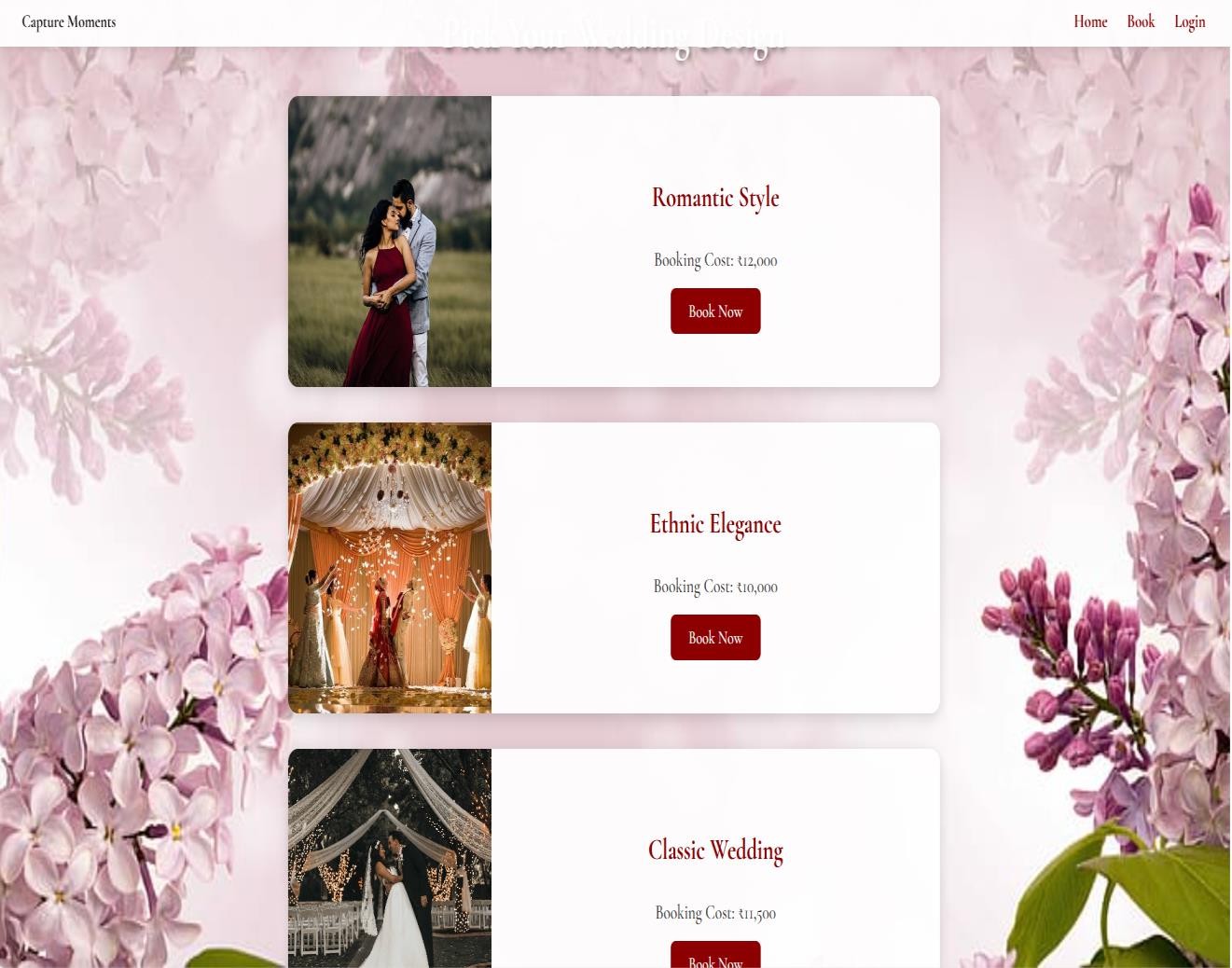
**Photographers Review page:**

****

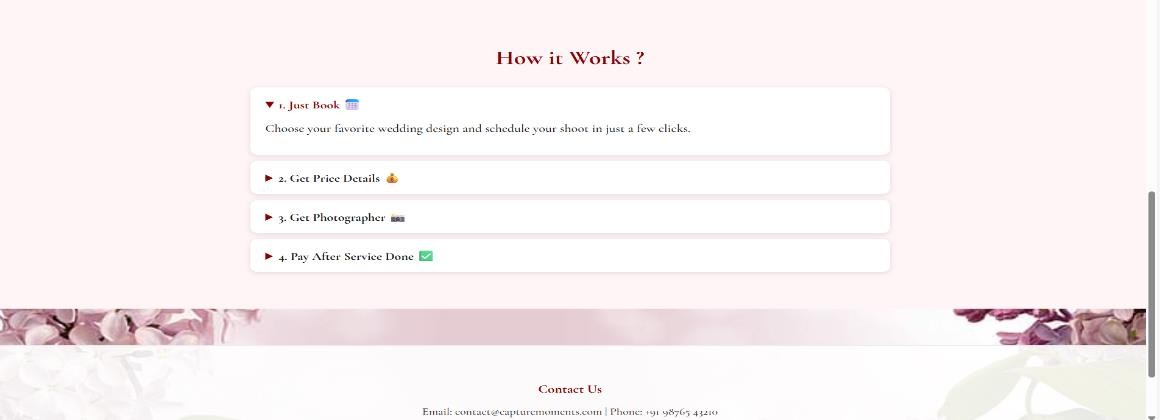
**Ratings showing:**

****

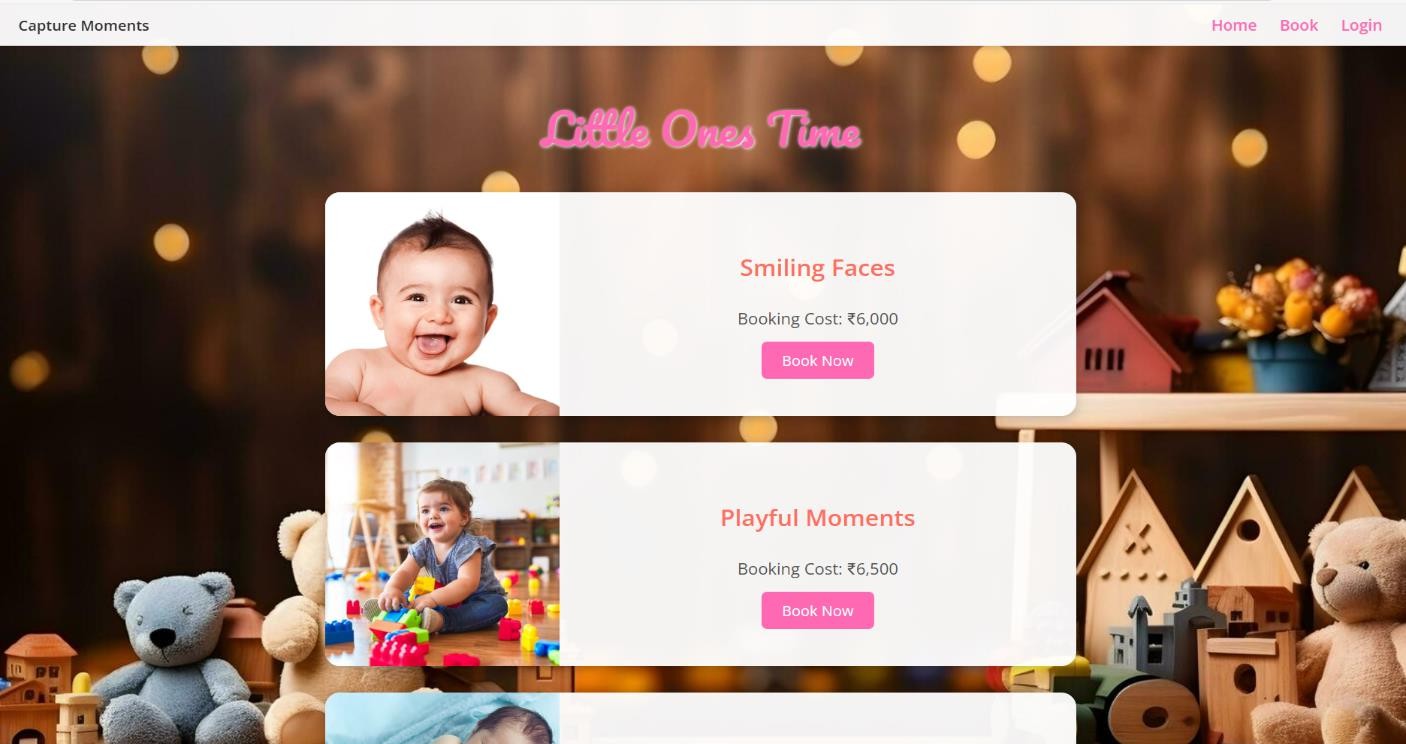
**Wedding page:**

****

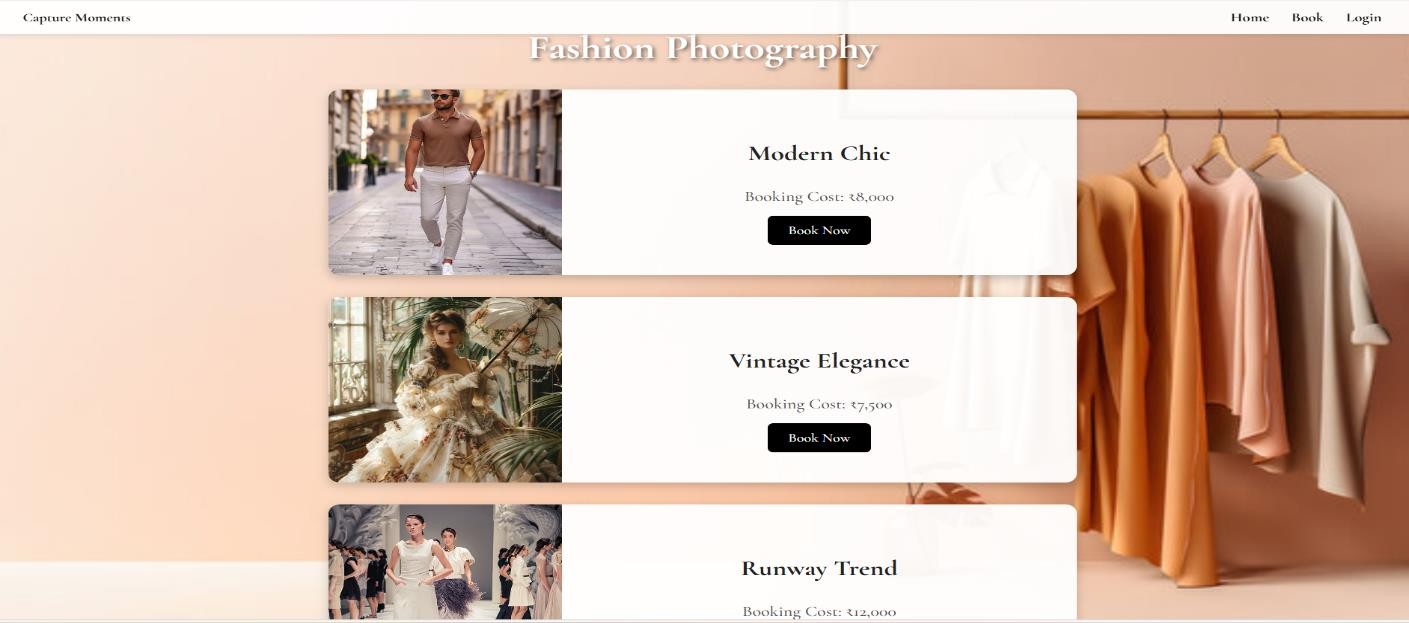


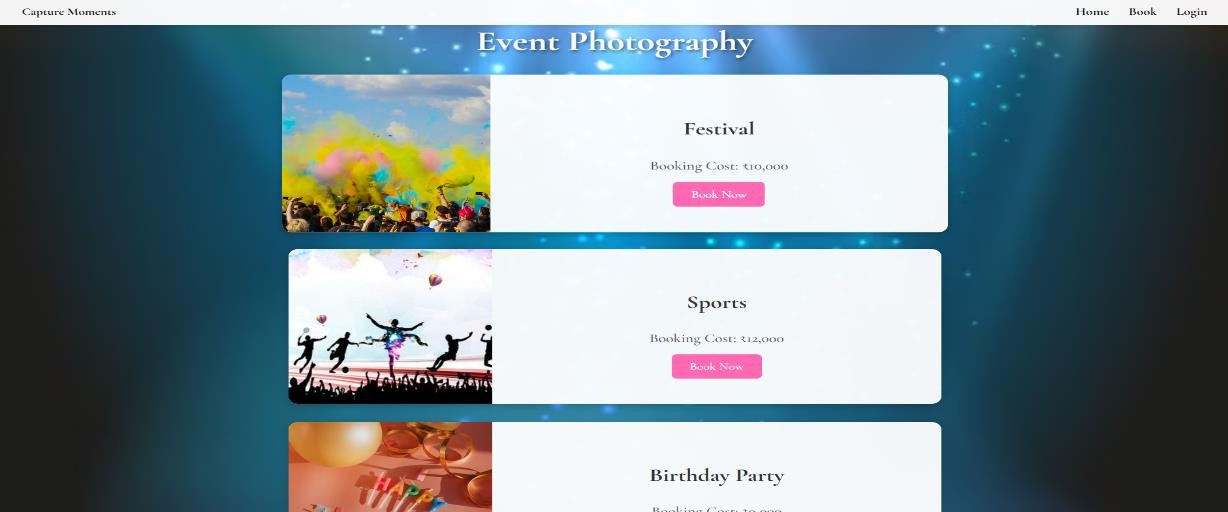


## Little ones page:

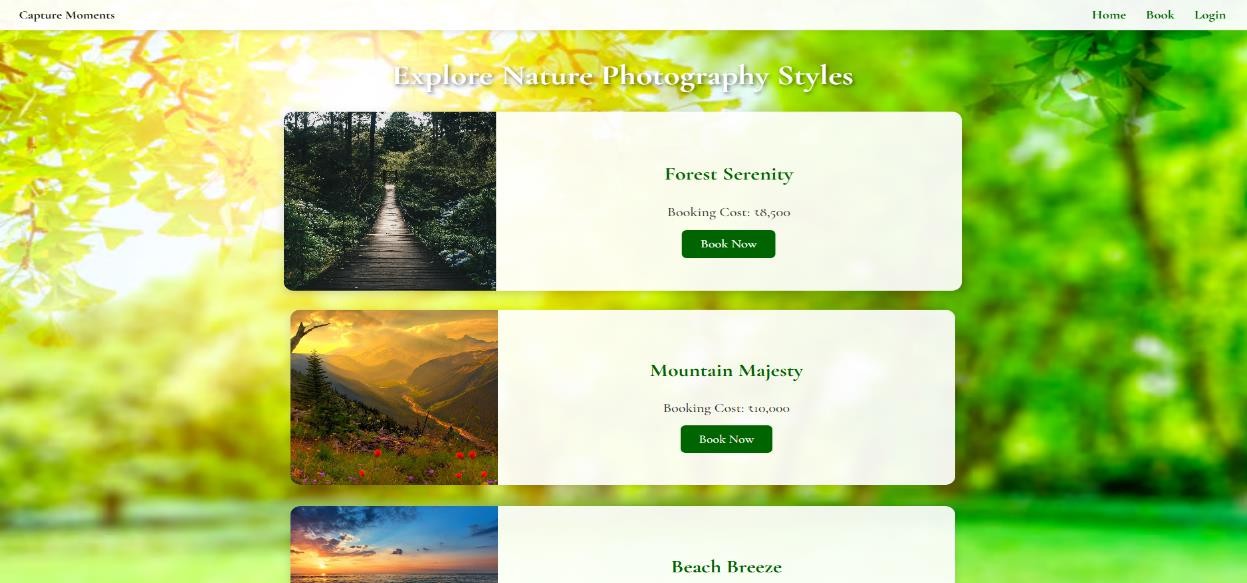
****

**Fashion page:**

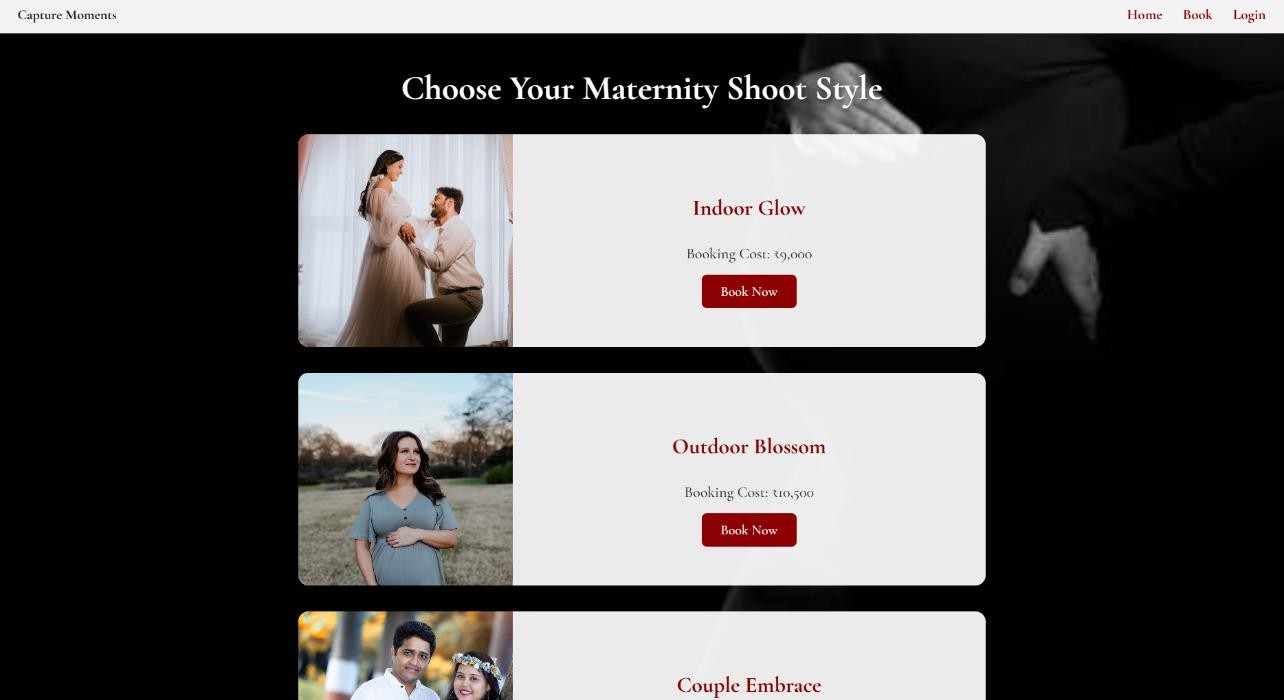
****



**Nature page:**

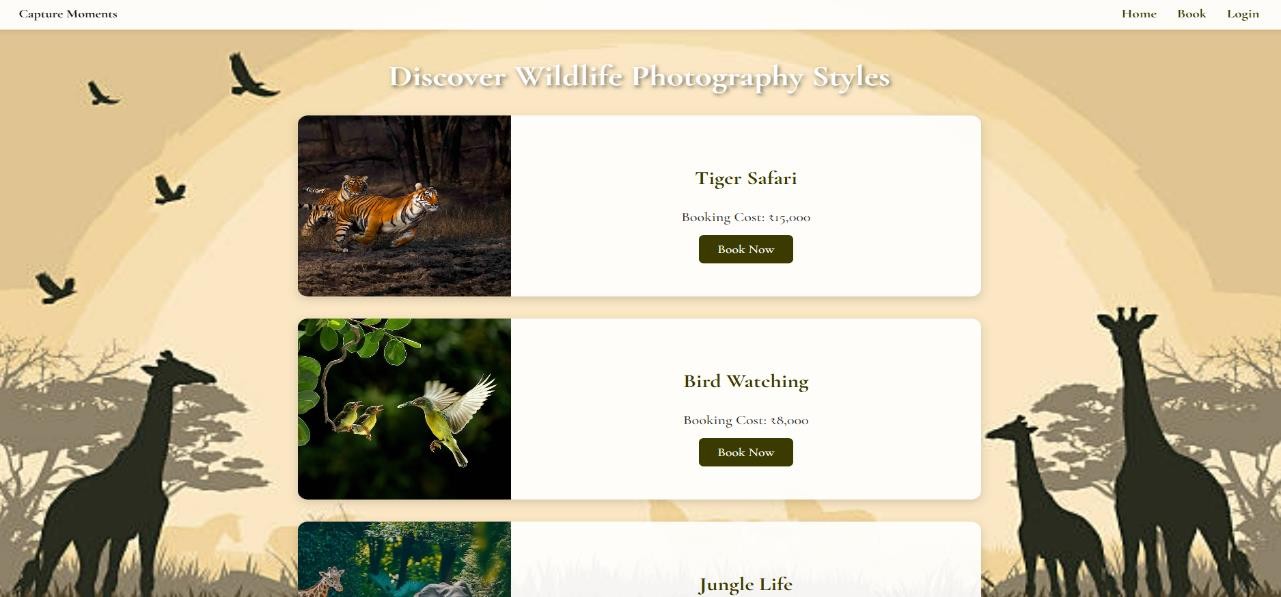
****

**Maternity page:**

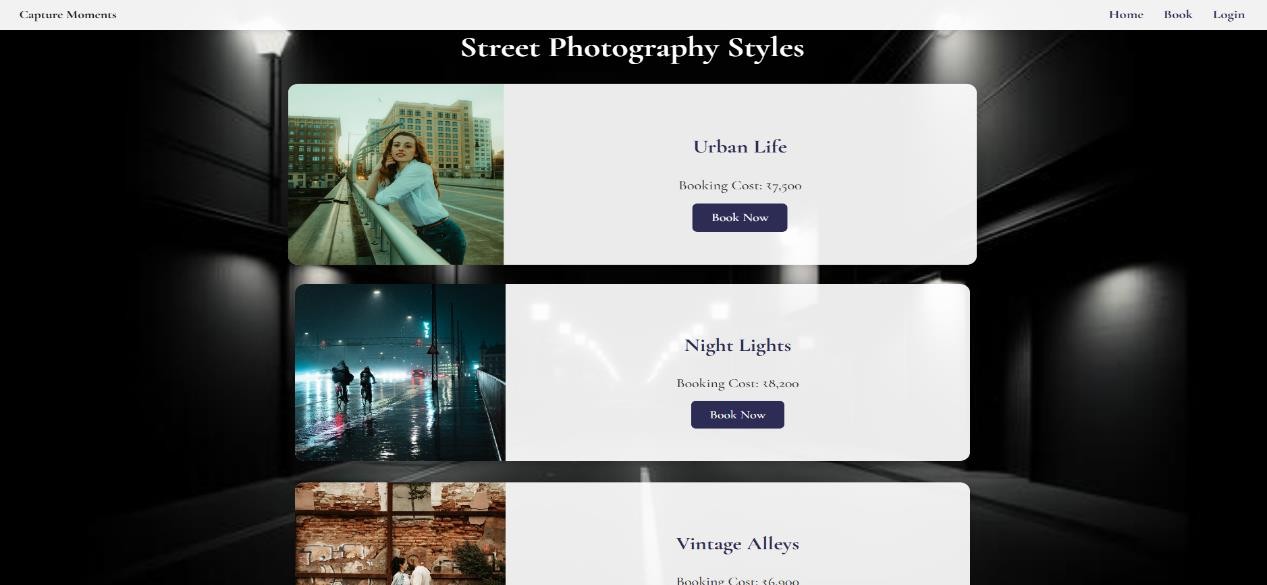
****

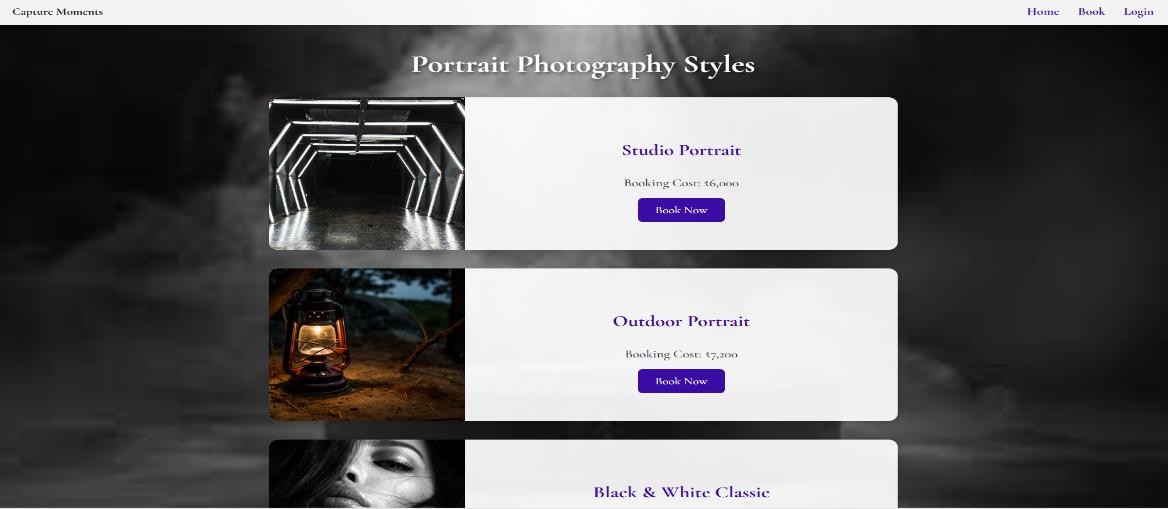


**Wild Life page:**

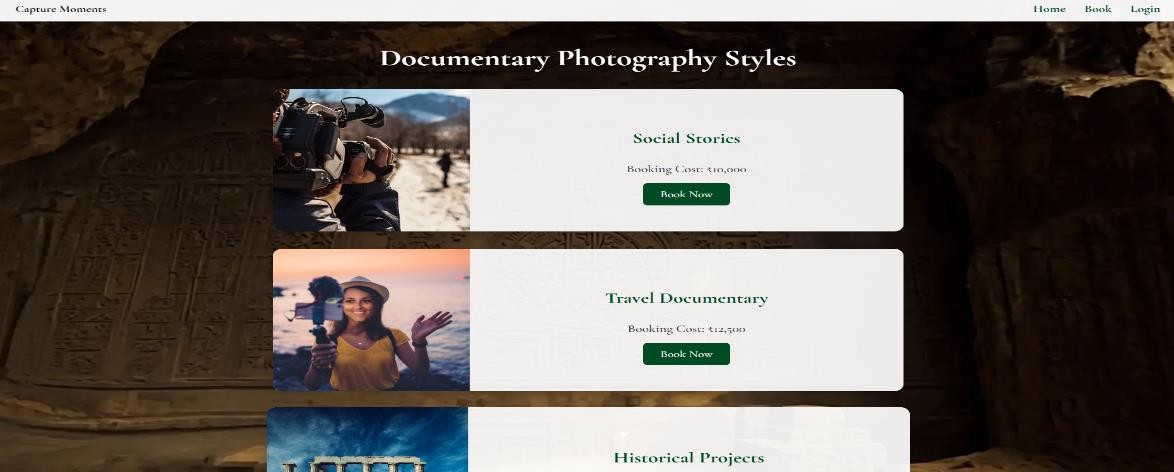


**Street Page:**

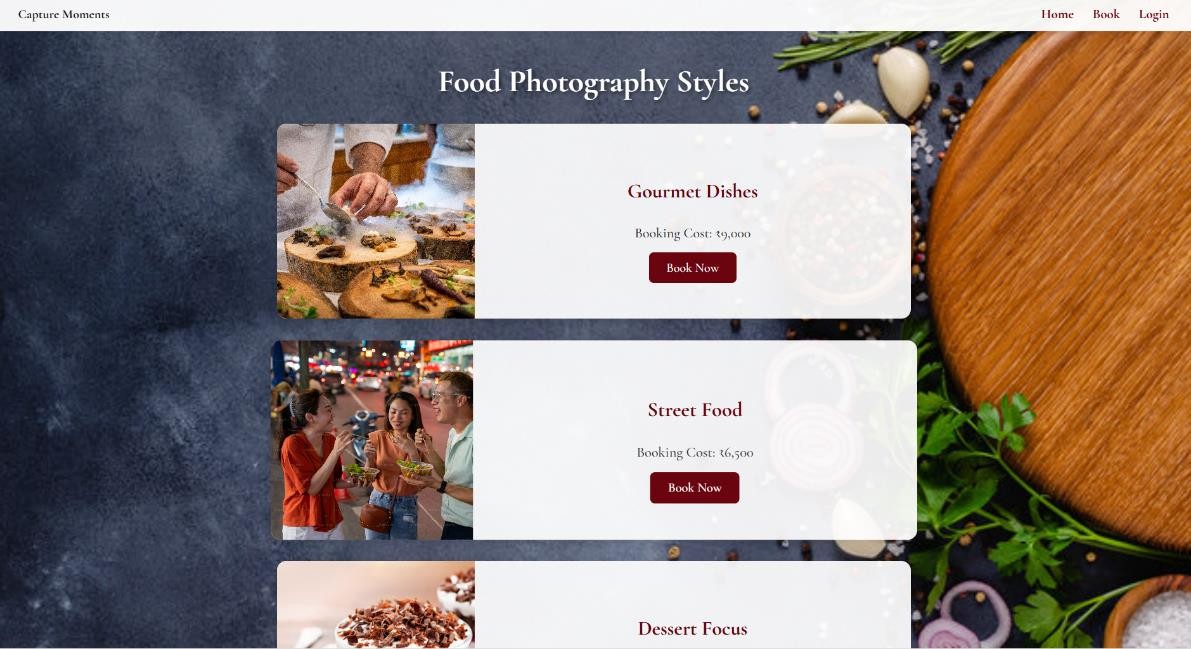
****



**Documentary page:**

****

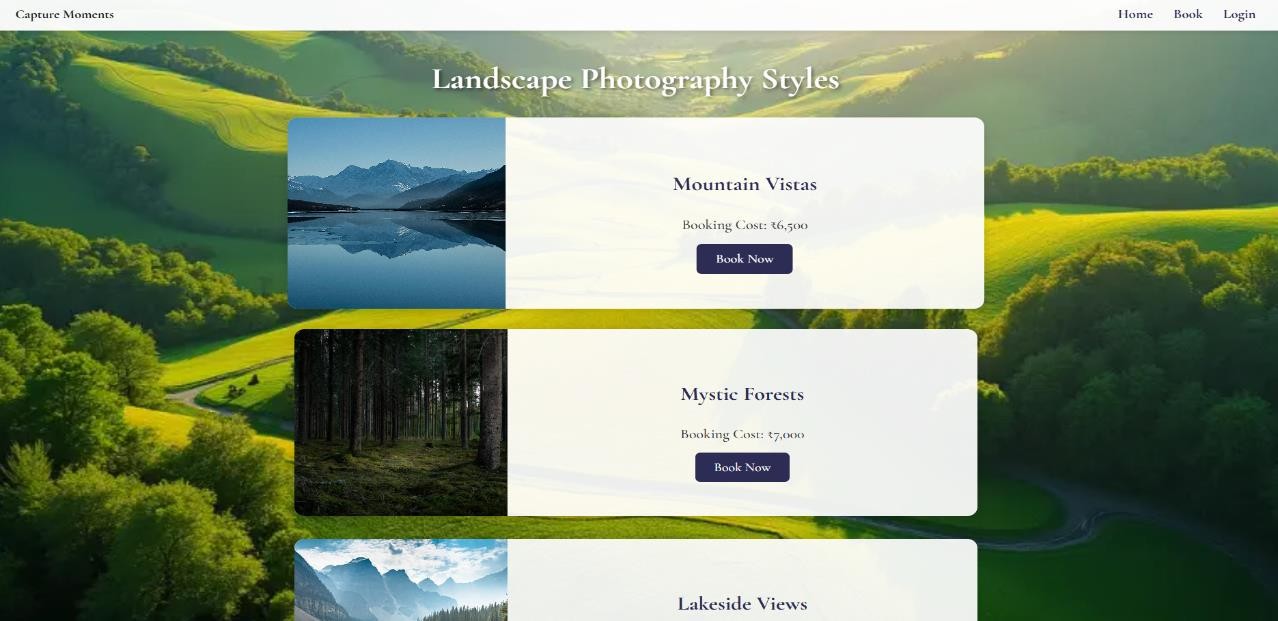
**Food page:**

****

**Architecture page:**

****

**Land Scape page:**

****

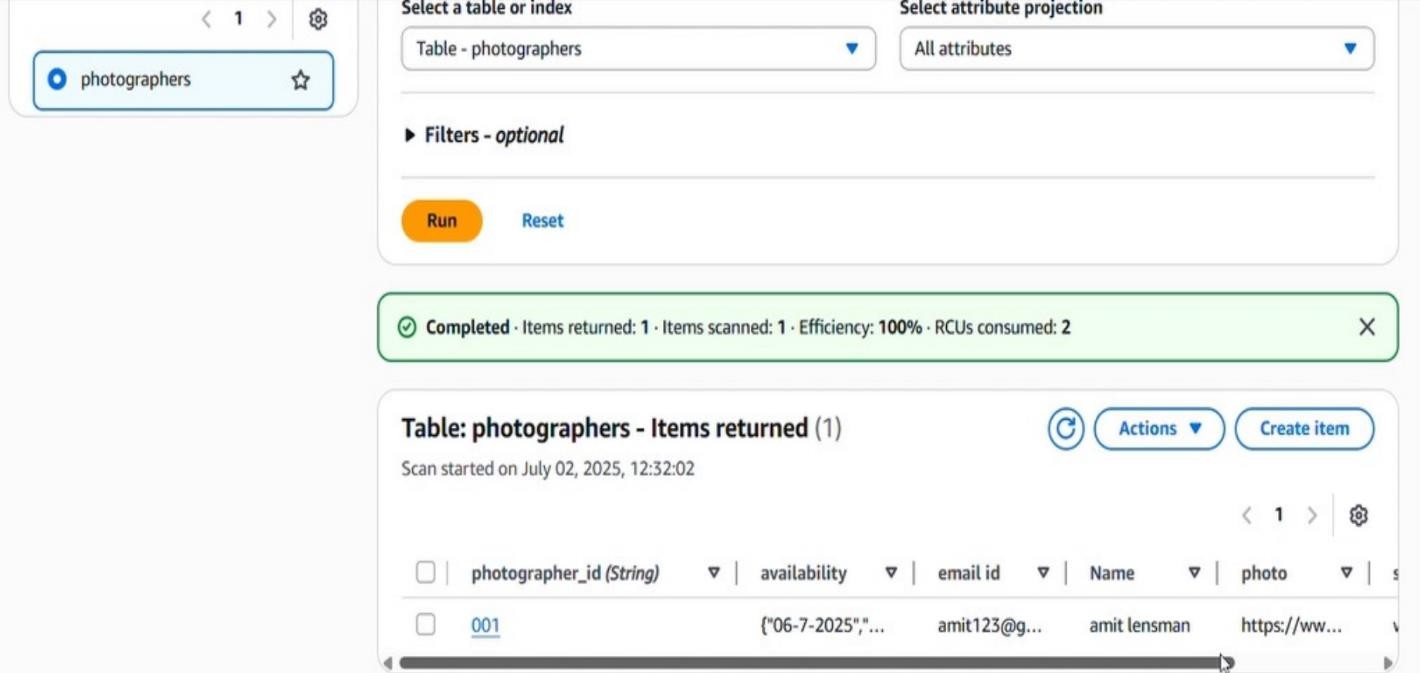
**Aerial page:**

****

# Book request submitted successfully!we will get back to you soon.

**Dynamodb Database updations :**

**1. Users table : 2.**

****

## Conclusion:

*Capture Moment* is a dynamic photography booking platform designed to showcase various photography styles including wedding, fashion, maternity, nature, and more. Built using **Flask** (Python) and HTML/CSS, the platform allows users to explore themed galleries and book professional photographers through an interactive web interface. The application is hosted on **Amazon Web Services (AWS)** for scalability and performance.

Key AWS integrations include **Amazon EC2** for deploying the Flask backend, **S3** for storing high-resolution photography portfolios, and **Route 53** for domain and DNS management. Optional services like **Amazon RDS** or **DynamoDB** can be used for managing photographer bookings and client data securely. The project ensures responsiveness, clean UI, and fast content delivery using AWS infrastructure.

This platform simplifies client-photographer interaction while leveraging cloud scalability to support growing content and user demands—ideal for creative professionals or photography businesses.

The application is hosted on **Amazon Web Services (AWS)** to ensure scalability, security, and high availability. The **Flask backend** runs on an **Amazon EC2 instance**, which provides flexible compute capacity to handle application traffic. All static assets including photographs, stylesheets, and scripts are stored in **Amazon S3**, offering durable and fast-access storage. **Amazon Route 53** is used for custom domain routing and DNS management, ensuring users can access the platform through a user-friendly URL.

The system can also be extended to use **Amazon RDS** for managing structured booking data (e.g., client information, schedule, preferred photography types), or **Amazon DynamoDB** for serverless and highly scalable NoSQL operations. To ensure secure access, **IAM roles and security groups** are configured to restrict unnecessary exposure.