

Movie Magic-Smart Movie Ticket Booking System

Project Description:

With the increasing demand for a seamless and modern movie-watching experience, traditional ticket booking methods often fall short due to long queues, limited availability, and inconsistent service. To address this, the development team introduced Movie Magic—a smart, cloud-based movie ticket booking system. Built using Flask for backend development, hosted on AWS EC2, and integrated with DynamoDB for dynamic data management, the platform allows users to register, log in, and book movie tickets online with ease. Users can search for movies and events based on location, view real-time seat availability, and complete their bookings in just a few clicks. Upon booking, AWS SNS sends instant email notifications confirming ticket details, enhancing user engagement and trust. This cloud-native solution streamlines the entire movie ticketing process, ensuring fast, scalable, and user-friendly access to entertainment for all.

Scenario 1: Efficient Booking system for users

In the Movie Magic System, AWS EC2 provides a reliable infrastructure capable of handling multiple users accessing the platform simultaneously. For example, a user can log in, navigate to the movie selection page, and seamlessly browse available shows and events in their city. They can then select a showtime, pick their preferred seats using an interactive layout, and confirm the booking—all in real-time. Flask manages backend processes, ensuring smooth data flow and quick response times even during high-traffic periods such as weekends or blockbuster releases.

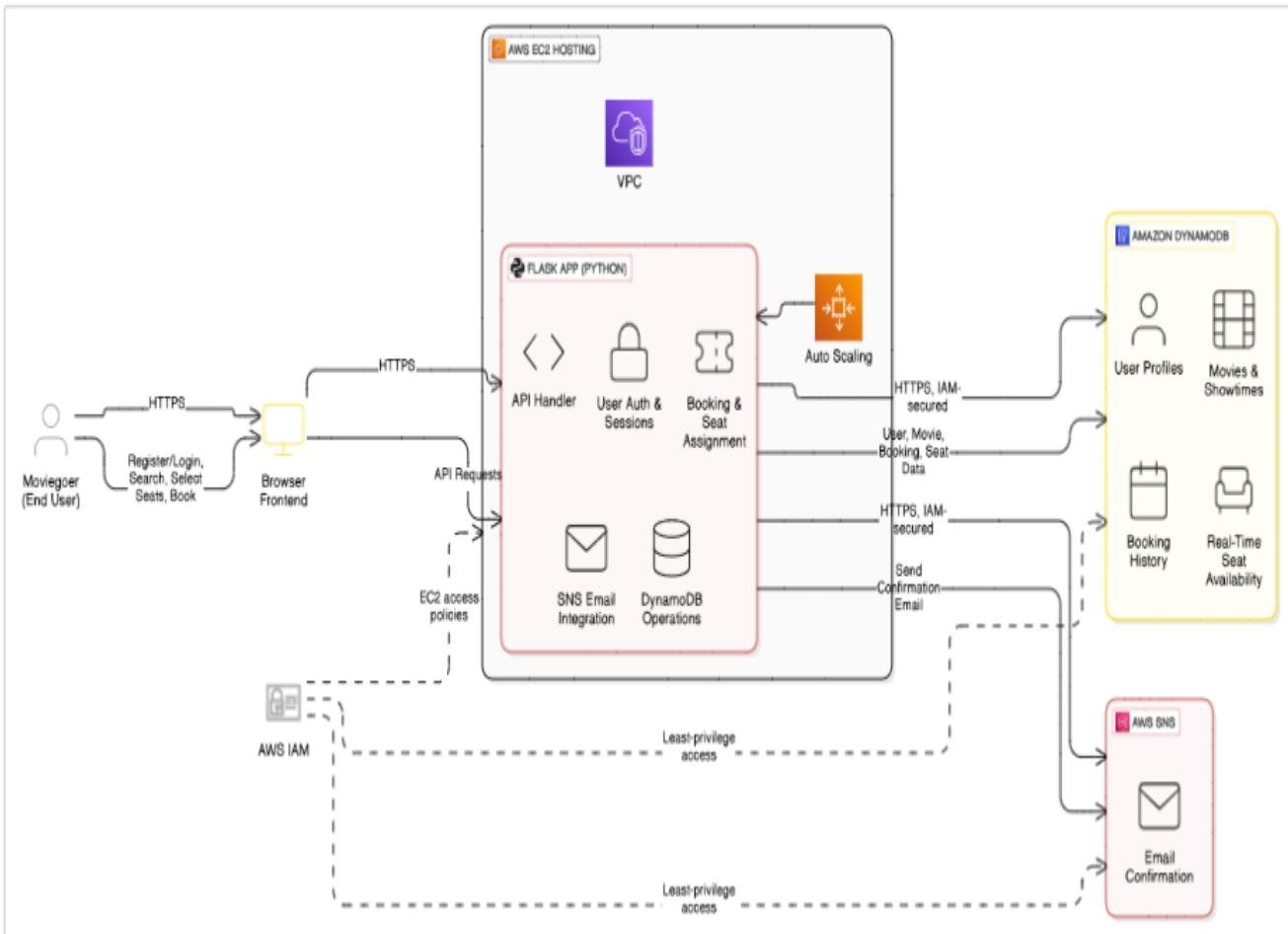
Scenario 2: Seamless Booking confirmation Notifications

When a user completes a ticket booking, the Movie Magic System leverages AWS SNS to send instant email notifications to confirm the booking. For instance, once the booking is submitted, Flask processes the transaction, and SNS sends a customized email to the user with all ticket details, including movie name, date, time, and seat numbers. This real-time notification system enhances the customer experience and reduces uncertainty, while DynamoDB securely stores the booking records for both users and admins to manage and track.

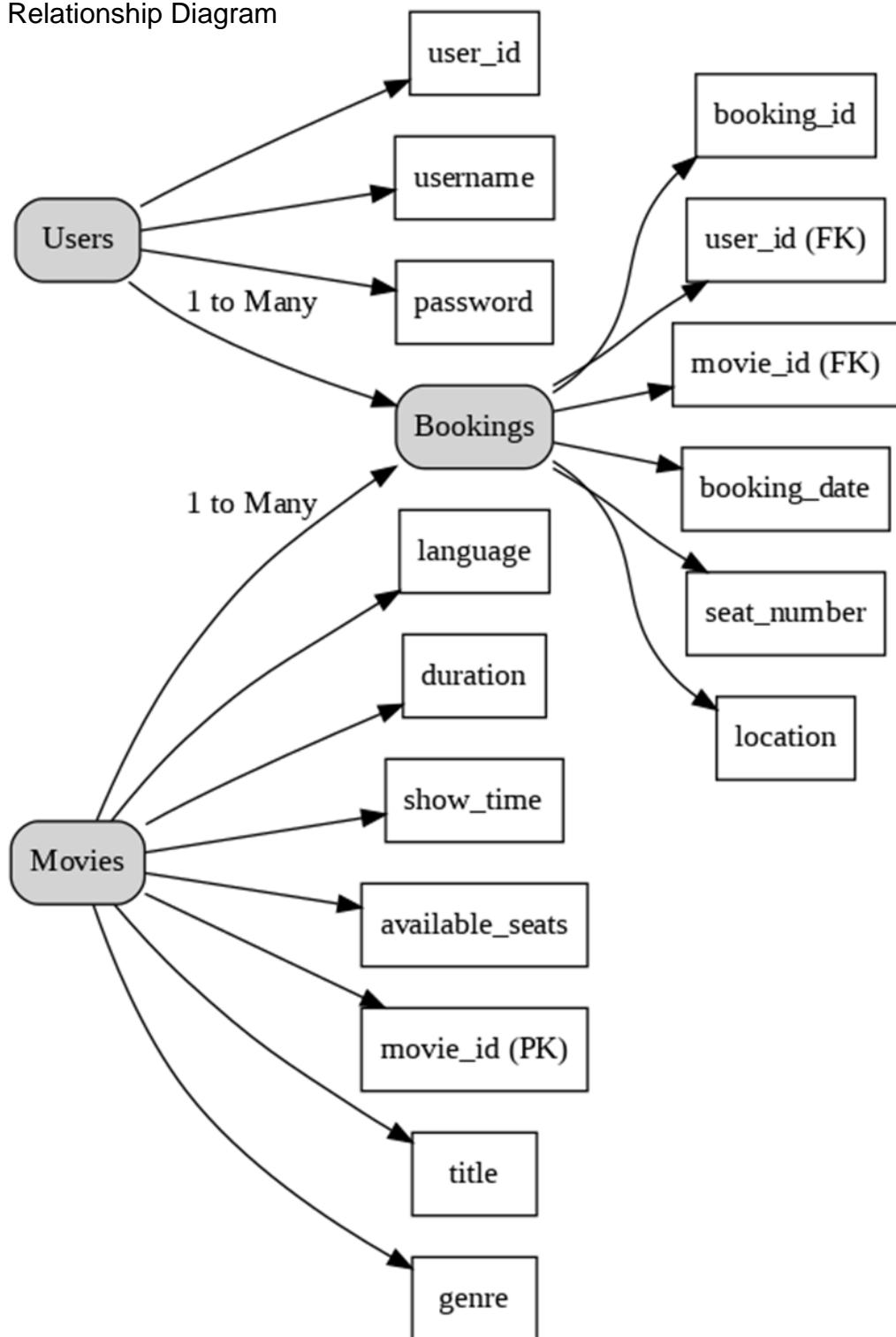
Scenario 3: Easy Access to Movies and Events

The Movie Magic platform offers users a seamless interface to explore currently running movies and upcoming live events. After logging in, a user can search by location or genre and instantly view listings with showtimes, ratings, and available seats. Flask dynamically fetches this data from DynamoDB, ensuring real-time updates on seat availability and event information. Meanwhile, the EC2-hosted application remains stable and responsive, even during traffic spikes, providing users with an uninterrupted and enjoyable booking experience.

AWS ARCHITECTURE



Entity Relationship Diagram



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 WorkFlow:

Milestone 1. Backend Development and Application Setup

Develop the Backend Using Flask.

Integrate AWS Services Using boto3.

Milestone 2. AWS Account Setup and Login

Set up an AWS account if not already done.

Log in to the AWS Management Console

Milestone 3. DynamoDB Database Creation and Setup

Create a DynamoDB Table.

Configure Attributes for User Data and Book Requests.

Milestone 4. SNS Notification Setup

Create SNS topics for book request notifications.

Subscribe users and library staff to SNS email notifications.

Milestone 5. IAM Role Setup

Create IAM Role

Attach Policies

Milestone 6. EC2 Instance Setup

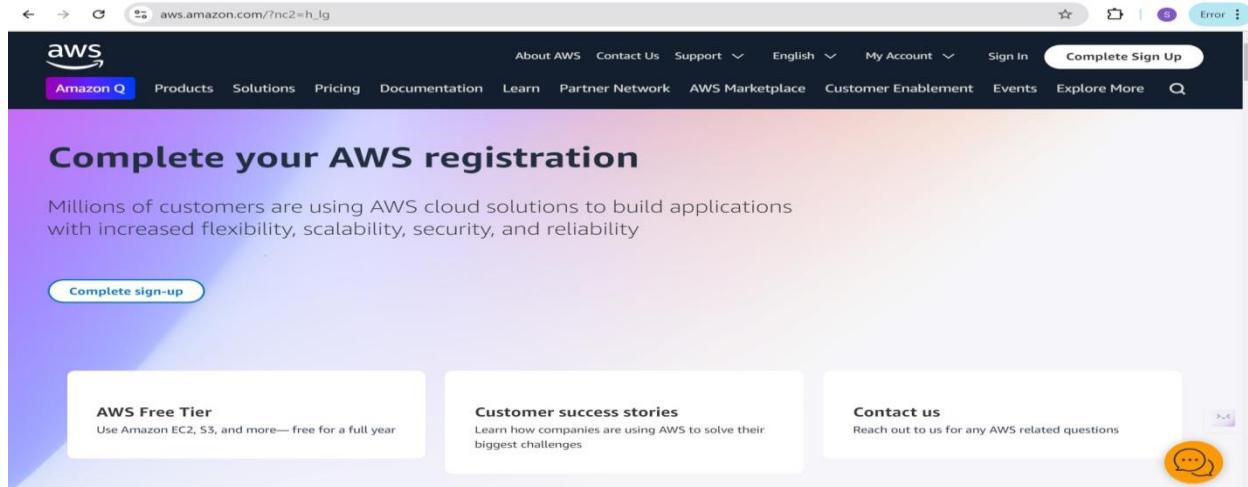
- Launch an EC2 instance to host the Flask application.
- Configure security groups for HTTP, and SSH access.

Milestone 7. Deployment on EC2

- Upload Flask Files
- Run the Flask App

Milestone 8. Testing and Deployment

- Conduct functional testing to verify user registration, login, book requests, and notifications.

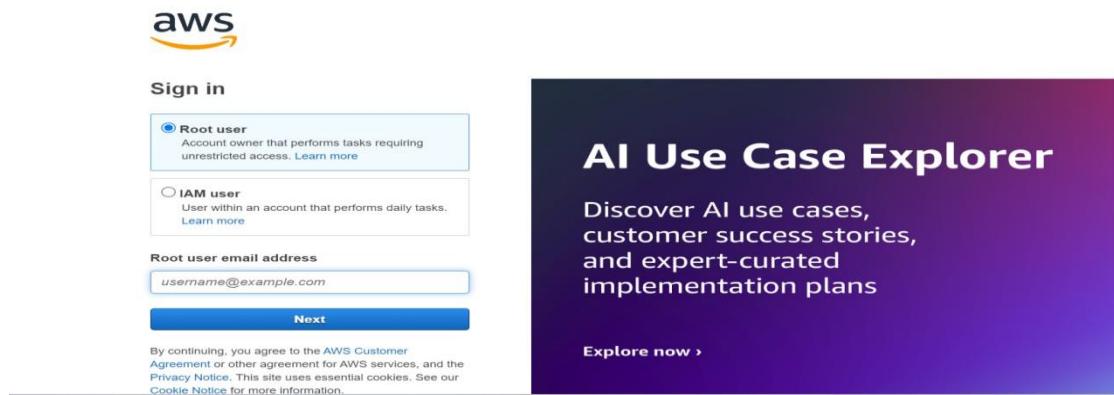


● Activity 1.2: Log in to the AWS Management Console

- After setting up your account, log in to the [AWS Management Console](#).

● Activity 1.2: Log in to the AWS Management Console

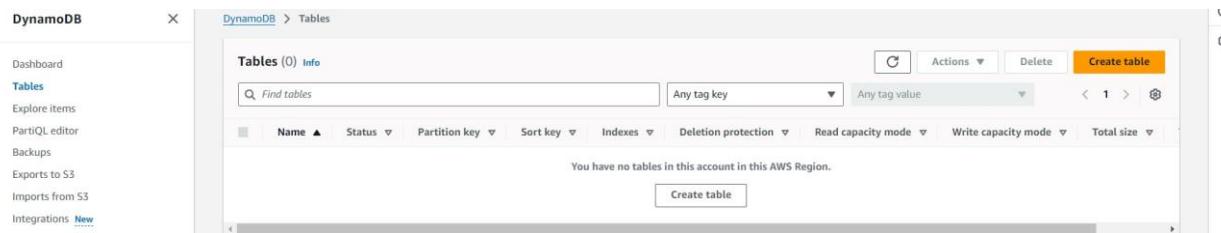
- After setting up your account, log in to the AWS Management Console.



The image shows two side-by-side screenshots. On the left is the AWS Sign-in page, featuring the AWS logo at the top, followed by a "Sign in" section. It includes two radio button options: "Root user" (selected) and "IAM user". Below each option is a brief description. A "Root user email address" input field contains "username@example.com", and a "Next" button is below it. At the bottom of the sign-in form is a small legal notice about cookie usage. On the right is a dark purple rectangular banner for the "AI Use Case Explorer". The text on the banner reads: "AI Use Case Explorer", "Discover AI use cases, customer success stories, and expert-curated implementation plans", and "Explore now >".

Milestone 2: DynamoDB Database Creation and Setup

- **Activity 2.1:** Navigate to the DynamoDB



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

- Create Users table with partition key "Email" with type String and click on create tables.

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.

Table settings

Default settings

The fastest way to create your table. You can modify most of these settings after your table has been created. To [modify these settings now, choose "Customize settings"](#)

Customize settings

Use these advanced features to make DynamoDB work better for your needs.



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

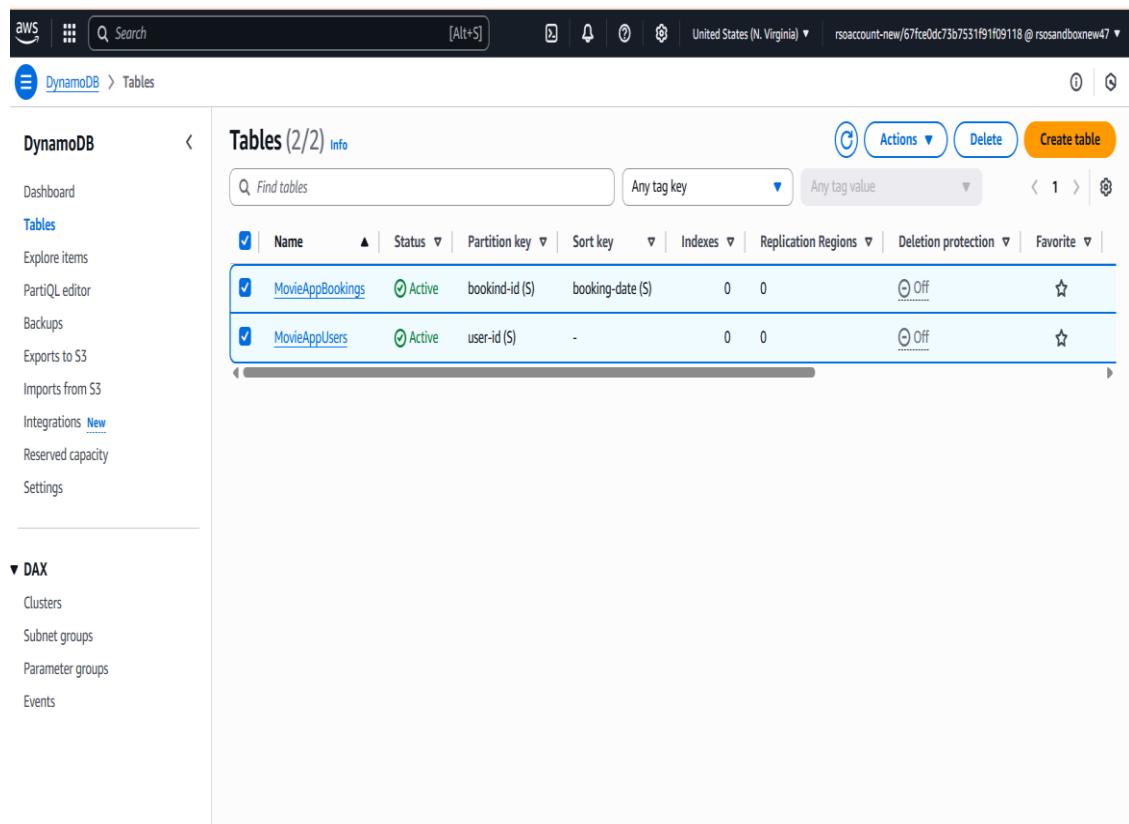
Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

[Add new tag](#)

You can add 50 more tags.

[Cancel](#)
[Create table](#)



The screenshot shows the AWS DynamoDB Tables page. On the left, there is a sidebar with the following navigation:

- Dashboard
- Tables** (selected)
- Explore items
- PartiQL editor
- Backups
- Exports to S3
- Imports from S3
- Integrations New
- Reserved capacity
- Settings

Below this is a section for DAX:

- Clusters
- Subnet groups
- Parameter groups
- Events

The main content area is titled "Tables (2/2)" and contains a table with the following data:

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite
MovieAppBookings	Active	bookid-id (\$)	booking-date (\$)	0	0	Off	☆
MovieAppUsers	Active	user-id (\$)	-	0	0	Off	☆

- Follow the same steps to create a requests table with Email as the primary key for book requests data.

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](#)

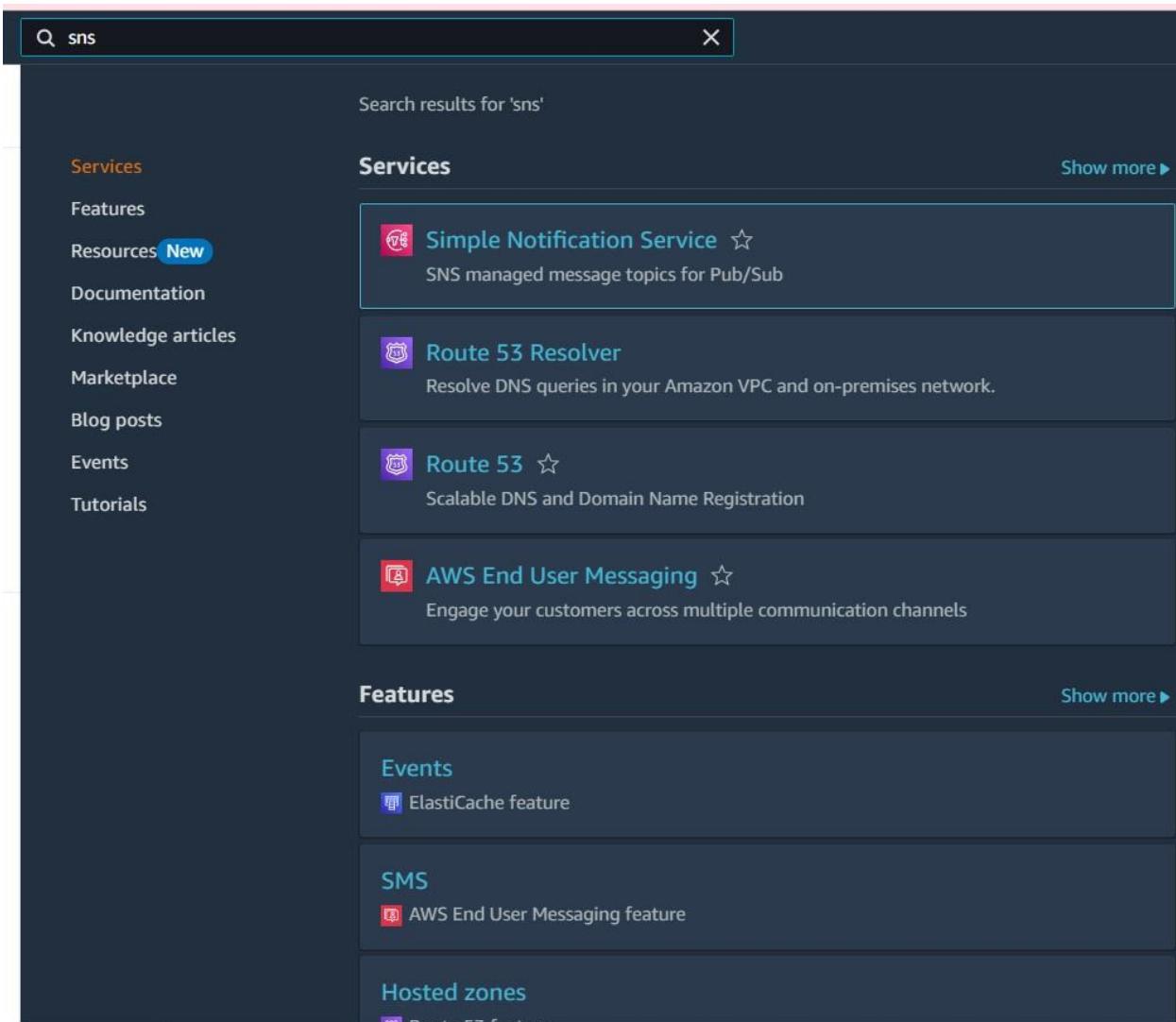
[Create table](#)

Milestone 3: SNS Notification Setup

- **Activity 3.1: Create SNS topics for sending email notifications to users and library staff.**
 - In the AWS Console, search for SNS and navigate to the SNS Dashboard.

Milestone 3: SNS Notification Setup

- Activity 3.1: Create SNS topics for sending email notifications to users and library staff.



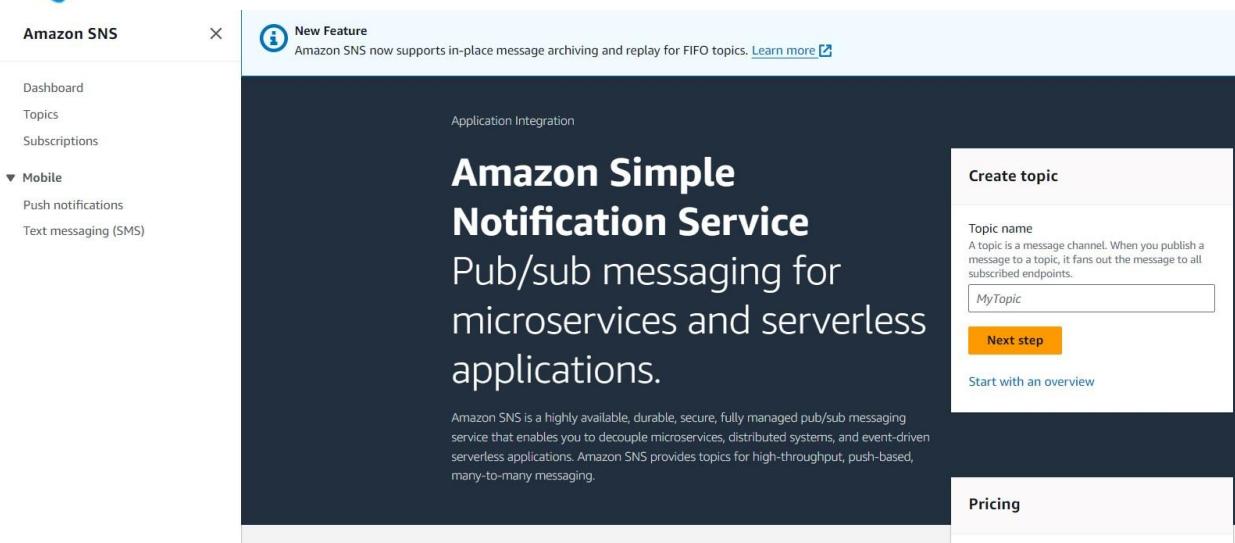
The screenshot shows the AWS search interface with the query 'sns' entered in the search bar. The results are categorized into 'Services' and 'Features'.

Services

- Simple Notification Service** ☆
SNS managed message topics for Pub/Sub
- Route 53 Resolver**
Resolve DNS queries in your Amazon VPC and on-premises network.
- Route 53** ☆
Scalable DNS and Domain Name Registration
- AWS End User Messaging** ☆
Engage your customers across multiple communication channels

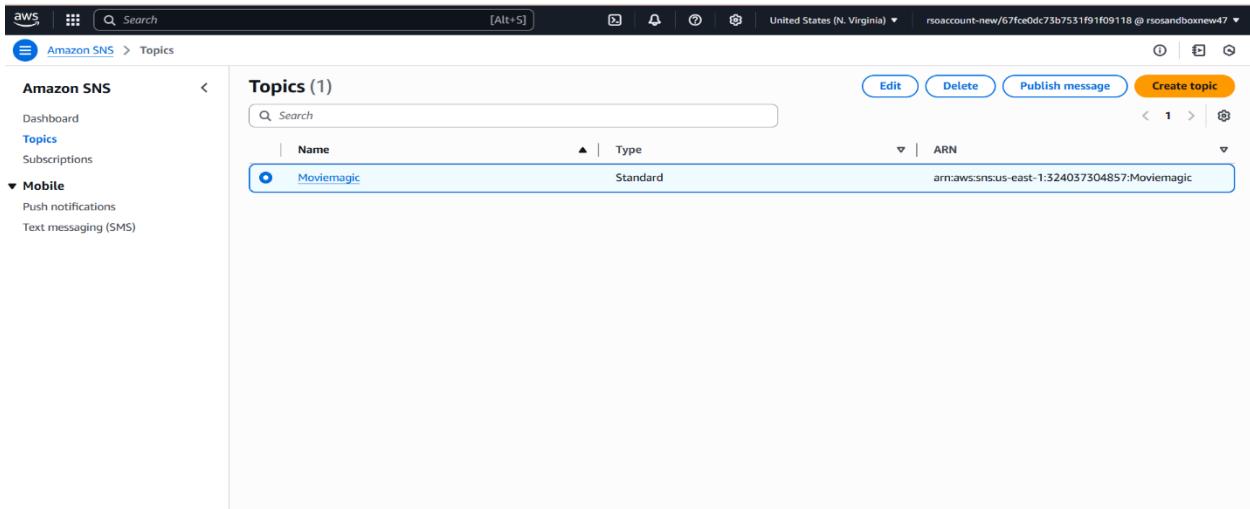
Features

- Events**
ElastiCache feature
- SMS**
AWS End User Messaging feature
- Hosted zones**
Route 53 feature



The screenshot shows the Amazon Simple Notification Service (SNS) home page. On the left, there is a sidebar with navigation links: Dashboard, Topics, Subscriptions, Mobile (Push notifications, Text messaging (SMS)), and Application Integration. A "New Feature" banner at the top right states: "Amazon SNS now supports in-place message archiving and replay for FIFO topics." Below the banner, the main content area features the title "Amazon Simple Notification Service" and the subtitle "Pub/sub messaging for microservices and serverless applications." A descriptive paragraph explains that Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service. To the right, there is a "Create topic" form with a "Topic name" field containing "MyTopic", a "Next step" button, and a link to "Start with an overview". At the bottom right, there is a "Pricing" section.

- Click on **Create Topic** and choose a name for the topic.



The screenshot shows the "Topics" page within the AWS Management Console. The left sidebar has the same navigation as the home page, with "Topics" being the active link. The main content area displays a table titled "Topics (1)". The table has columns for Name, Type, and ARN. One row is shown, labeled "Moviemagic" under the "Name" column, "Standard" under "Type", and "arn:aws:sns:us-east-1:324037304857:Moviemagic" under "ARN". Action buttons for "Edit", "Delete", "Publish message", and "Create topic" are located at the top right of the table. The ARN value is also displayed below the table.

- Choose Standard type for general notification use cases and Click on Create Topic.

Amazon SNS > Topics > Create topic

Create topic

Details

Type [Info](#)
Topic type cannot be modified after topic is created

FIFO (first-in, first-out)

- Strictly-preserved message ordering
- Exactly-once message delivery
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS

Standard

- Best-effort message ordering
- At-least once message delivery
- Highest throughput in publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Name

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - *optional* [Info](#)
To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.

Maximum 100 characters.

► **Access policy - optional** [Info](#)
 This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic.

► **Data protection policy - optional** [Info](#)
 This policy defines which sensitive data to monitor and to prevent from being exchanged via your topic.

► **Delivery policy (HTTP/S) - optional** [Info](#)
 The policy defines how Amazon SNS retries failed deliveries to HTTP/S endpoints. To modify the default settings, expand this section.

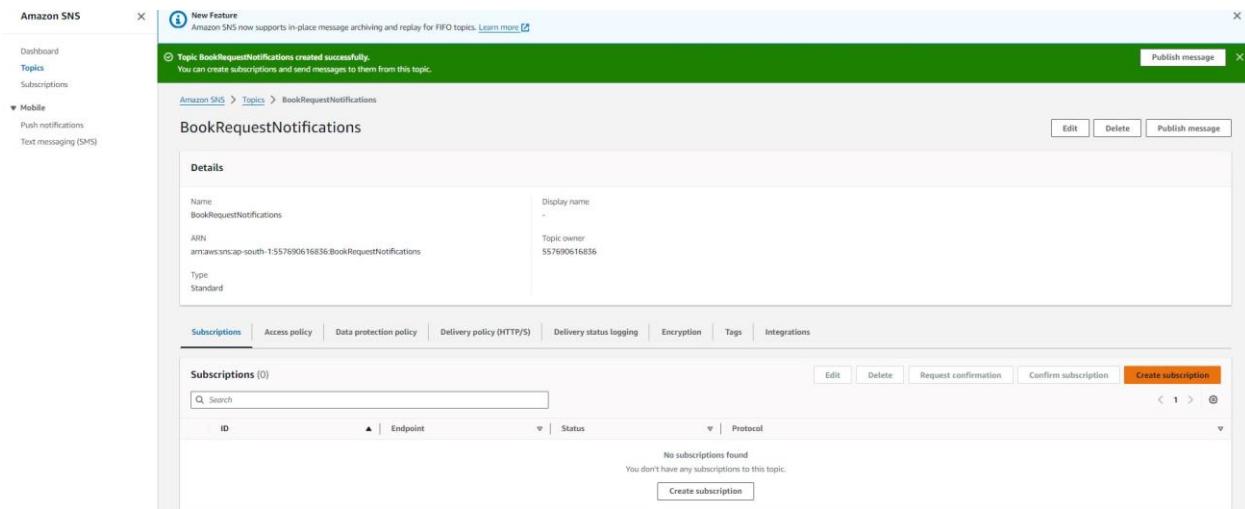
► **Delivery status logging - optional** [Info](#)
 These settings configure the logging of message delivery status to CloudWatch Logs.

► **Tags - optional**
 A tag is a metadata label that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track your costs. [Learn more](#)

► **Active tracing - optional** [Info](#)
 Use AWS X-Ray active tracing for this topic to view its traces and service map in Amazon CloudWatch. Additional costs apply.

[Cancel](#) [Create topic](#)

- Configure the SNS topic and note down the **Topic ARN**.



The screenshot shows the AWS SNS 'Topics' section. A green banner at the top indicates that the 'BookRequestNotifications' topic was created successfully. The main area displays the topic details:

- Name:** BookRequestNotifications
- ARN:** arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications
- Type:** Standard
- Display name:** (empty)
- Topic owner:** 557690616836

Below the details, there are tabs for Subscriptions, Access policy, Data protection policy, Delivery policy (HTTP/S), Delivery status logging, Encryption, Tags, and Integrations. The Subscriptions tab shows a table with one row:

ID	Endpoint	Status	Protocol
No subscriptions found. You don't have any subscriptions to this topic.			

At the bottom right of the table are buttons for Edit, Delete, Request confirmation, Confirm subscription, and Create subscription.

● **Activity 3.2: Subscribe users and staff to relevant SNS topics to receive real-time notifications when a movie booking is made.**

Subscribe users (or admin staff) to this topic via Email. When a book request is made, notifications will be sent to the subscribed emails

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN
arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications X

Protocol
The type of endpoint to subscribe
Email ▼

Endpoint
An email address that can receive notifications from Amazon SNS.
instantlibrary2@gmail.com

ⓘ After your subscription is created, you must confirm it. [Info](#)

► **Subscription filter policy - optional** [Info](#)
This policy filters the messages that a subscriber receives.

► **Redrive policy (dead-letter queue) - optional** [Info](#)
Send undeliverable messages to a dead-letter queue.

[Cancel](#) Create subscription



Amazon SNS > Topics > Moviemagic > Subscription: 06001d83-4d79-449c-b031-5e0e4d978688



Amazon SNS

Dashboard

Topics

Subscriptions

▼ Mobile

Push notifications

Text messaging (SMS)

New Feature

Amazon SNS now supports High Throughput FIFO topics. [Learn more](#)

Subscription to Moviemagic created successfully.

The ARN of the subscription is arn:aws:sns:us-east-1:324037304857:Moviemagic:06001d83-4d79-449c-b031-5e0e4d978688.

Subscription: 06001d83-4d79-449c-b031-5e0e4d978688

[Edit](#) [Delete](#)

Details

ARN

arn:aws:sns:us-east-1:324037304857:Moviemagic:06001d83-4d79-449c-b031-5e0e4d978688

Endpoint

akhilkuma2116@gmail.com

Topic

Moviemagic

Subscription Principal

arn:aws:iam::324037304857:role/rsoaccount-new

Status

Pending confirmation

Protocol

EMAIL

Subscription filter policy

Redrive policy (dead-letter queue)

- After subscription request for the mail confirmation

Amazon SNS

Dashboard
Topics
Subscriptions▼ Mobile
Push notifications
Text messaging (SMS)

New Feature

Amazon SNS now supports in-place message archiving and replay for FIFO topics. [Learn more](#)

Confirmation request was sent successfully.

The ARN of the subscription is arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:7ae5d15-12ad-4731-9f7d-347c2213a05.

Amazon SNS > Topics > BookRequestNotifications

[Edit](#) [Delete](#) [Publish message](#)

BookRequestNotifications

Details

Name

BookRequestNotifications

Display name

-

ARN

arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications

Topic owner

557690616836

Type

Standard

Subscriptions

Access policy Data protection policy Delivery policy (HTTP/S) Delivery status logging Encryption Tags Integrations

[Edit](#) [Delete](#) [Request confirmation](#) [Confirm subscription](#) [Create subscription](#)

Subscriptions (2)

 Q: Search

ID	Endpoint	Status	Protocol
Pending confirmation	instantlibrary2@gmail.com	Pending confirmation	EMAIL

- Navigate to the subscribed Email account and Click on the confirm subscription in the AWS Notification- Subscription Confirmation mail.

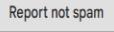
← Delete forever | Not spam |  

1 of 3 < >

AWS Notification - Subscription Confirmation Spam x

 **AWS Notifications <no-reply@sns.amazonaws.com>** to me ▾ Sun, Jul 6, 10:47 AM (4 days ago)    

Why is this message in spam? This message is similar to messages that were identified as spam in the past.

You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:975050261480:moviemagic

To confirm this subscription, click or visit the link below (if this was in error no action is necessary):
[Confirm subscription](#)

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to [sns-opt-out](#)



Simple Notification Service

Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:

arn:aws:sns:us-east-1:324037304857:Moviemagic:06001d83-4d79-449c-b031-5e0e4d978688

If it was not your intention to subscribe, [click here to unsubscribe](#).

- Successfully done with the SNS mail subscription and setup, now store the ARN link.

Amazon SNS > Topics > BookRequestNotifications

BookRequestNotifications

Details	
Name	BookRequestNotifications
Display name	-
ARN	arn:aws:sns:ap-south-1:155769061686:BookRequestNotifications
Type	Standard
Topic owner	557690616836

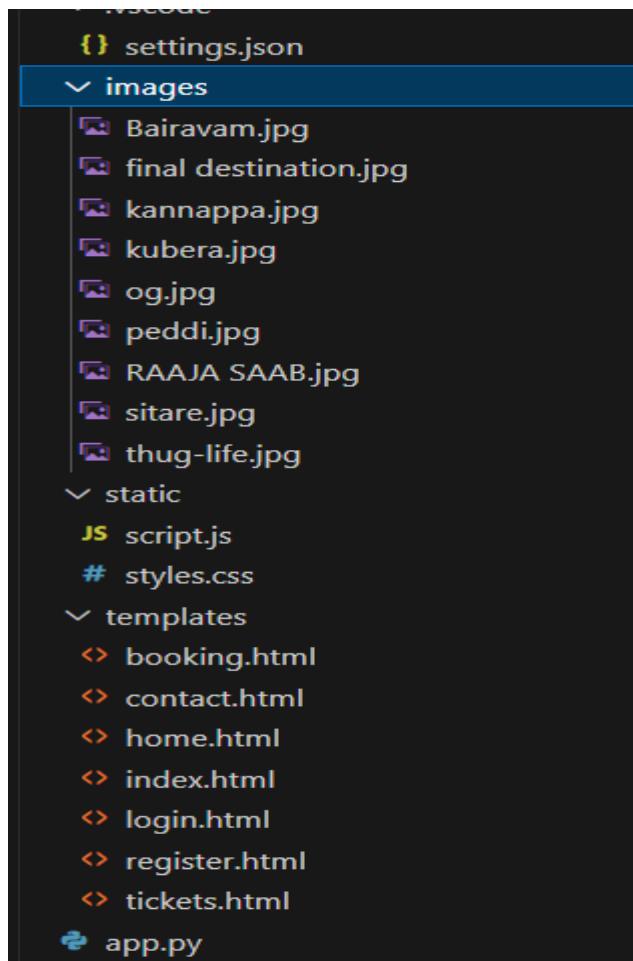
       

Subscriptions (2)						
ID	Endpoint	Status	Protocol			
d7fe0f571-9235-4040-952c-05c2743607c4	instantlibrary2@gmail.com	Confirmed	EMAIL			

Milestone 4:Backend Development and Application Setup

- **Activity 4.1: Develop the backend using Flask**

- File Explorer Structure



Description: set up the INSTANT LIBRARY project with an app.py file, a static/ folder for assets, and a templates/ directory containing all required HTML pages like home, login, register, booking,tickets,contact pages (e.g., home.html,login.html), and css and js files (e.g., styles.css,script.js etc).

Description of the code :

- **Flask App Initialization**

Description of the code :

● Flask App Initialization



```
from flask import Flask, render_template, request, redirect, url_for, session, flash, jsonify
from werkzeug.security import generate_password_hash, check_password_hash
import boto3
from botocore.exceptions import ClientError
import os
```

Description: import essential libraries including Flask utilities for routing, Boto3 for DynamoDB operations, SMTP and email modules for sending mails, and Bcrypt for password hashing and verification

```
app = Flask(__name__)
```

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

● Dynamodb Setup:

```
dynamodb = boto3.resource('dynamodb', region_name=AWS_REGION)
users_table = dynamodb.Table(DYNAMODB_USERS)
bookings_table = dynamodb.Table(DYNAMODB_BOOKINGS)
sns_client = boto3.client('sns', region_name=AWS_REGION)
```

Description: initialize the DynamoDB resource for the ap-south-1 region and set up access to the Users and Requests tables for storing user details and book requests.

● SNS Connection

```
# AWS Services setup
AWS_REGION = os.environ.get('AWS_REGION', 'us-east-1')
DYNAMODB_USERS = os.environ.get('DYNAMODB_USERS', 'MovieAppUsers')
DYNAMODB_BOOKINGS = os.environ.get('DYNAMODB_BOOKINGS', 'MovieAppBookings')
SNS_TOPIC_ARN = os.environ.get('SNS_TOPIC_ARN', 'arn:aws:sns:us-east-1:975050261480:moviemagic:84b8b')
```

Description:

Configure SNS to send notifications when a book request is submitted. Paste your stored ARN link in the sns_topic_arn space, along with the region_name where the SNS topic is created. Also, specify the chosen email service in SMTP_SERVER (e.g., Gmail, Yahoo, etc.) and enter the subscribed email in the SENDER_EMAIL section. Create an 'App password' for the email ID and store it in the SENDER_PASSWORD section.

- **Routes for Web Pages**

- **Home Route:**

```
# Home route redirects to Registration page
@app.route('/')
def home():
    return redirect(url_for('register'))
```

Description: define the home route / to automatically redirect users to the register page when they access the base URL.

- **Register Route:**

```
@app.route('/check-auth')
def check_auth():
    return jsonify({'authenticated': 'username' in session})

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        data = request.get_json()
        username = data['username']
        password = data['password']
        email = data['email']
        try:
            resp = users_table.get_item(Key={'username': username})
            if 'Item' in resp:
                return jsonify({'success': False, 'message': 'Username already exists'}), 400
            users_table.put_item(Item={
                'username': username,
                'password': generate_password_hash(password),
                'email': email
            })
            return jsonify({'success': True})
        except ClientError as e:
            return jsonify({'success': False, 'message': str(e)}), 500
    return render_template('register.html')
```

Description: define /register route to validate registration form fields, hash the user password using decrypt, store the new user in DynamoDB with a login count, and send an SNS notification on successful registration

● login Route (GET/POST):



```
def login():
    if request.method == 'POST':
        data = request.get_json()
        username = data['username']
        password = data['password']
        try:
            resp = users_table.get_item(Key={'username': username})
            user = resp.get('Item')
            if user and check_password_hash(user['password'], password):
                session['username'] = username
                return jsonify({'success': True})
            else:
                return jsonify({'success': False, 'message': 'Invalid username or password'}), 401
        except ClientError as e:
            return jsonify({'success': False, 'message': str(e)}), 500
    return render_template('login.html')
```

Description: define /login route to validate user credentials against DynamoDB, check the password using Bcrypt, update the login count on successful authentication, and redirect users to the home page

● Home Page

```
@app.route('/home')
def home():
    if 'username' not in session:
        flash('Please login first', 'error')
        return redirect(url_for('login'))
    return render_template('home.html', movies=movies)
```

Description: define /home-page to render the main homepage, / it will redirect to the homepage after the user logs in

● Booking Form:

```
@app.route('/booking', methods=['POST'])
def make_booking():
    if 'username' not in session:
        return jsonify({'success': False, 'message': 'Not authenticated'}), 401
    data = request.get_json()
    movie_id = int(data['movie_id'])
    movie = movies.get(movie_id)
    if not movie:
        return jsonify({'success': False, 'message': 'Movie not found'}), 404
    showtime = data['showtime']
    if showtime not in movie['showtimes']:
        return jsonify({'success': False, 'message': 'Invalid showtime'}), 400
    seats = data['seats']
    if not isinstance(seats, int) or seats <= 0:
        return jsonify({'success': False, 'message': 'Invalid number of seats'}), 400

    booking_data = {
        'username': session['username'],
        'movie_id': str(movie_id),
        'movie_title': movie['title'],
        'showtime': showtime,
        'seats': seats
    }

    try:
        # Store booking in DynamoDB
        bookings_table.put_item(item=booking_data)

        # Send SNS notification
        send_booking_confirmation(
            username=session['username'],
            movie_title=movie['title'],
            showtime=showtime,
            seats=seats
        )

        return jsonify({"success": True})
    except ClientError as e:
        return jsonify({'success': False, 'message': str(e)}), 500
●
```

Description: define /request-form route to capture movie bookings of users, store the request in DynamoDB, send a thank-you email to the user, notify the admin, and confirm submission with a success message.

Tickets Route:

```

@app.route('/tickets')
def tickets():
    if 'username' not in session:
        return redirect(url_for('login'))
    try:
        resp = bookings_table.scan(
            FilterExpression='username = :username',
            ExpressionAttributeValues={':username': session['username']})
    )
    user_bookings = resp.get('Items', [])
except Exception:
    user_bookings = []
return render_template('tickets.html', bookings=user_bookings)

@app.route('/contact')
def contact():
    return render_template('contact.html')

```

Description: it navigates to selected tickets section

Deployment Code:

```

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=True)

```

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.



Services Q Iam X

Search results for 'Iam'

Services

Features

Resources New

Documentation

Knowledge articles

Marketplace

Blog posts

Events

Tutorials

Services

Show more ▶

- 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
- AWS App Mesh** ☆ Easily monitor and control microservices

aws > Services > Create role

Select trusted entity

Trusted entity type

AWS service: Allow users in the EC2 Lambda, or others to perform actions in this account.

AWS account: Allow roles in other AWS accounts belonging to you or a third party to perform actions in this account.

IAM, ECR, or Lambda Federation: Allow users in IAM, ECR, or Lambda Federation accounts to perform actions in this account.

Custom trust policy: Create a trust policy to enable others to perform actions in this account.

Use case

Service or use case: EC2

Choose a use case for the specified service.

Use case:

EC2: Allow EC2 instances to be run on your behalf.

EC2 Role for AWS Systems Manager: Allow EC2 instances to run AWS Systems Manager on your behalf.

EC2 Lambda: Allow EC2 Lambda to request and terminate Lambda functions on your behalf.

EC2 - Spot Fleet Auto Scaling: Allow EC2 Spot Fleet Auto Scaling instances to run EC2 use cases on your behalf.

EC2 - Spot Fleet Tagging: Allow EC2 Spot Fleet instances and anything they do to be launched instances on your behalf.

EC2 - Spot Instances: Allow EC2 Spot Instances to be started and managed with Amazon on your behalf.

EC2 - Spot Fleet: Allow EC2 Spot Fleet instances to request and manage spot fleet instances on your behalf.

EC2 - Scheduled instances: Allow EC2 Scheduled instances to manage instances on your behalf.

Cancel Save

aws > Instances > i-08aebea5215699cb > Modify IAM role

Modify IAM role [Info](#)

Attach an IAM role to your instance.

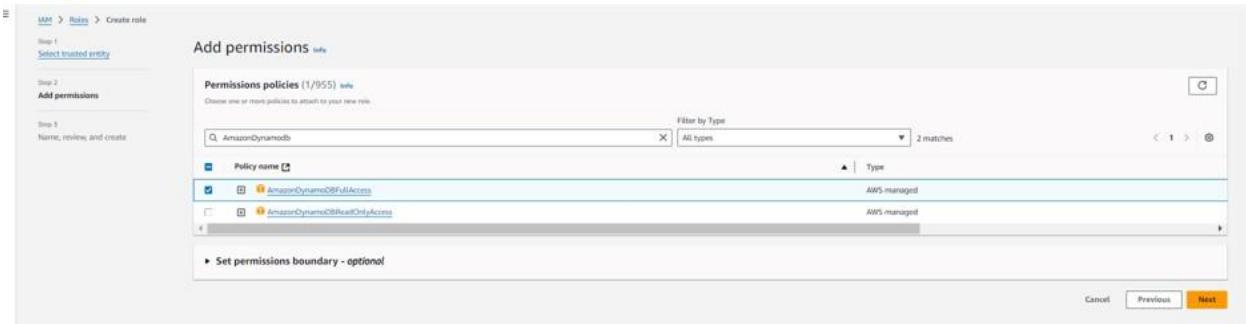
Instance ID: i-08aebea5215699cb (Moviemagic)

IAM role

Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

Create new IAM role [Create new IAM role](#)

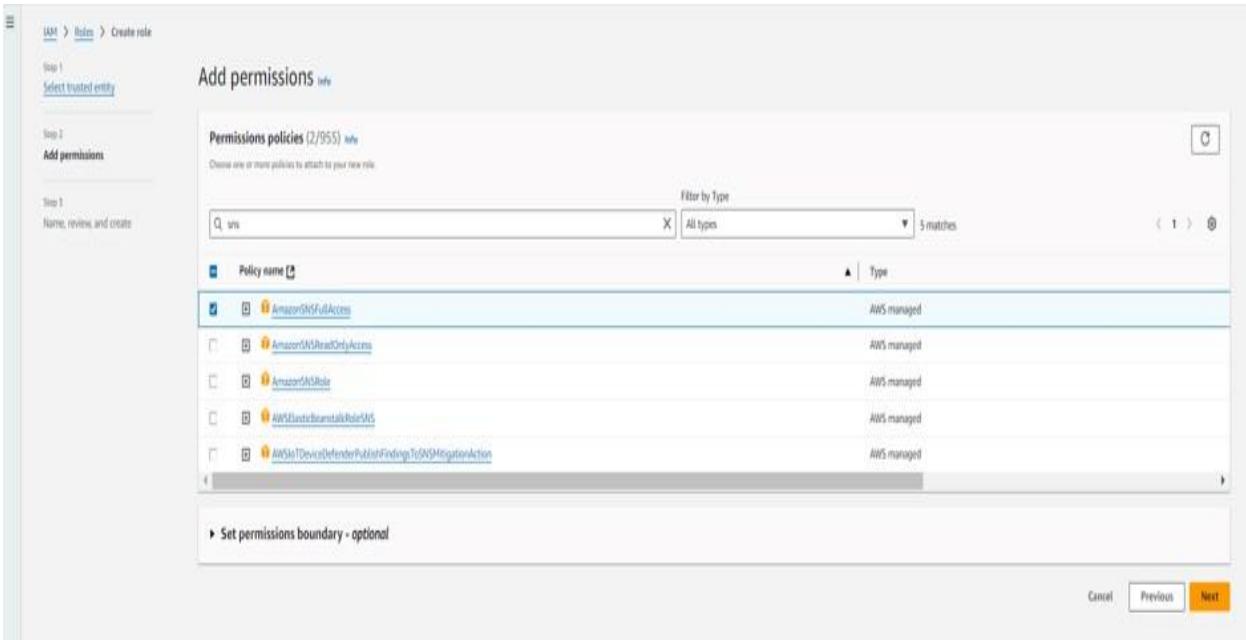
Cancel Update IAM role



- **Activity 5.2: Attach Policies.**

Attach the following policies to the role:

- AmazonDynamoDBFullAccess: Allows EC2 to perform read/write operations on DynamoDB.
- AmazonSNSFullAccess: Grants EC2 the ability to send notifications via SNS.





Name, review, and create

Role details

Role name: sns_Dynamodb_role

Description: Allows EC2 instances to call AWS services on your behalf.

Step 1: Select trusted entities

Trust policy:

```
1 | { "Version": "2012-10-17", 2 | "Statement": [ 3 | { "Effect": "Allow", 4 | "Principal": "AWS", 5 | "Action": "sts:AssumeRole" 6 | } ] } 7 | }
```

Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached to
AmazonDynamoDBFullAccess	AWS managed	Permissions policy
AmazonSNSFullAccess	AWS managed	Permissions policy

Step 3: Add tags

Add tags - optional Info

No tags associated with this resource.

Add new tag

Has not added any 50 more tags

Cancel | Previous | Create role

IAM > Roles > sns_Dynamodb_role

sns_Dynamodb_role Info

Allows EC2 Instances to call AWS services on your behalf.

Summary

Creation date October 13, 2024, 23:06 (UTC+05:30)	ARN arn:aws:iam::557690616836:role/sns_Dynamodb_role	Instance profile ARN arn:aws:iam::557690616836:instance-profile/sns_Dynamodb_role
Last activity 6 days ago	Maximum session duration 1 hour	

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Permissions policies (2) Info

You can attach up to 10 managed policies.

Filter by Type

Policy name	Type	Attached entities
AmazonDynamoDBFullAccess	AWS managed	4
AmazonSNSFullAccess	AWS managed	2

Search | All types | < 1 > | ⚙

Milestone 6: EC2 Instance Setup

- Load your Flask app and Html files into GitHub repository.



A screenshot of a GitHub repository page. At the top, it shows a commit from user 'Akhil000-ak' titled 'Update app.py'. Below the commit, there's a list of files with their status:

- images: Add files via upload
- static: Update script.js
- templates: Add files via upload
- README.md: Initial commit
- app.py: Update app.py

At the bottom of the list, there's a link to 'README'.

In the center of the page, there's a large banner with the text 'Movie-Magic-Smart-Movie-Ticket-Booking-Sys'.

A screenshot of a GitHub repository clone interface. It shows two tabs: 'Local' and 'Codespaces'. Below the tabs, there are three cloning options: 'Clone' (with icons for HTTPS, SSH, and GitHub CLI), 'HTTPS' (selected), 'SSH', and 'GitHub CLI'. A URL 'https://github.com/Akhil000-ak/MovieMagic.git' is displayed in a box with a copy icon. Below the URL, there's a note 'Clone using the web URL.' and two additional options: 'Open with GitHub Desktop' and 'Download ZIP'.

● 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.



A screenshot of the AWS search interface. The search bar at the top contains the text 'ec2'. Below the search bar, there is a sidebar with links like 'Services', 'Features', 'Resources New', 'Documentation', 'Knowledge articles', 'Marketplace', 'Blog posts', 'Events', and 'Tutorials'. The main content area displays search results for 'ec2', with the 'EC2' service card highlighted. The card has a blue header with the EC2 logo and the text 'Virtual Servers in the Cloud'. Below the header, there is a description: 'A managed service to automate build, customize and deploy OS images'. Other cards visible include 'EC2 Image Builder' and 'Recycle Bin'.

- Click on Launch instance to launch EC2 instance

A screenshot of the AWS EC2 Instances page. The left sidebar shows navigation options: Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, and Elastic Block Store. The main content area shows a table titled 'Instances (1/1)'. It lists one instance: 'Moviemagic' (Instance ID: i-08aebca32135699cb). The instance is shown as 'Running' with an 't2.micro' instance type, '2/2 checks passed' status, and 'us-east-1a' availability zone. A 'Launch instances' button is located at the top right of the table. Below the table, a detailed view for the instance 'i-08aebca32135699cb (Moviemagic)' is displayed. The 'Details' tab is selected, showing sections for Instance summary, Public IPv4 address (54.89.154.151), Instance state (Running), Private IPv4 addresses (172.31.27.116), and Public DNS (ec2-54-89-154-151.compute-1.amazonaws.com).

EC2 > Instances > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name	<input type="text" value="InstantLibraryApp"/>	Add additional tags
------	--	-------------------------------------

Summary

Number of instances [Info](#)

Software Image (AMI)
 Amazon Linux 2023 AMI 2023.5.2...[read more](#)
 ami-078264b8ba71bc45e

Virtual server type (instance type)
 t2.micro

Firewall (security group)

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



Amazon Linux



macOS



Ubuntu



Windows



Red Hat

...

🔍
[Browse more AMIs](#)

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI

ami-02b49a24cfb95941c (64-bit (x86), uefi-preferred) / ami-04ad8c7fcc828fad4 (64-bit (Arm), uefi)
 Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Description

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Architecture	Boot mode	AMI ID	Verified provider
<input type="text" value="64-bit (x86)"/>	<input type="text" value="uefi-preferred"/>	<input type="text" value="ami-02b49a24cfb95941c"/>	

- Create and download the key pair for Server access.

- Create and download the key pair for Server access.

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro	Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true	
On-Demand Linux base pricing: 0.0124 USD per Hour	
On-Demand Windows base pricing: 0.017 USD per Hour	
On-Demand RHEL base pricing: 0.0268 USD per Hour	
On-Demand SUSE base pricing: 0.0124 USD per Hour	

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

Select [Create new key pair](#)

Create key pair

Key pair name
Key pairs allow you to connect to your instance securely.

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

RSA
RSA encrypted private and public key pair

ED25519
ED25519 encrypted private and public key pair

Private key file format

.pem
For use with OpenSSH

.ppk
For use with PuTTY

⚠️ 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. [Learn more](#)

[Cancel](#) [Create key pair](#)


InstantLibrary.pem

Description

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Architecture	Boot mode	AMI ID	Username	 Verified provider
64-bit (x86)	uefi-preferred	ami-078264b8ba71bc45e	ec2-user	

Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro	Family: t2 1 vCPU 1 GiB Memory Current generation: true	Free tier eligible
	On-Demand Linux base pricing: 0.0124 USD per Hour	
	On-Demand Windows base pricing: 0.017 USD per Hour	Compare instance types
	On-Demand RHEL base pricing: 0.0268 USD per Hour	
	On-Demand SUSE base pricing: 0.0124 USD per Hour	

Additional costs apply for AMIs with pre-installed software

Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

InstantLibrary		Create new key pair
----------------	---	-------------------------------------

 **Free tier:** In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Cancel
 [Preview code](#)
 [Launch instance](#)

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

Network settings [Info](#)

VPC - required [Info](#)
 vpc-03cdc7b6f19dd7211 (default) ▾ 

Subnet [Info](#)
 No preference ▾ 

Auto-assign public IP [Info](#)
 Enable ▾

Additional charges apply when outside of free tier allowance

Firewall (security groups) [Info](#)
 A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group Select existing security group

Security group name - required
 launch-wizard

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and _-:/()#,@[]+=&{}!\$^

Description - required [Info](#)
 launch-wizard created 2024-10-13T17:49:56.622Z

EC2 > Security Groups > sg-096e7520d7a58d18 - launch-wizard-1 > Edit inbound rules



Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules [Info](#)

Security group rule ID	Type Info	Protocol Info	Port range	Source Info	Description - optional Info
Info					
sgr-021ec12b7e1217da6	SSH	TCP	22	Custom ▾	<input type="text"/> Q  0.0.0.0/X
sgr-09aed7f4d29b96410	HTTP	TCP	80	Custom ▾	<input type="text"/> Q  0.0.0.0/X
sgr-094b7c9c2e6d69b5b	HTTPS	TCP	443	Custom ▾	<input type="text"/> Q  0.0.0.0/X
-	Custom TCP	TCP	5000	Anywh... ▾	<input type="text"/> Q  0.0.0.0/X

[Add rule](#)

 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.

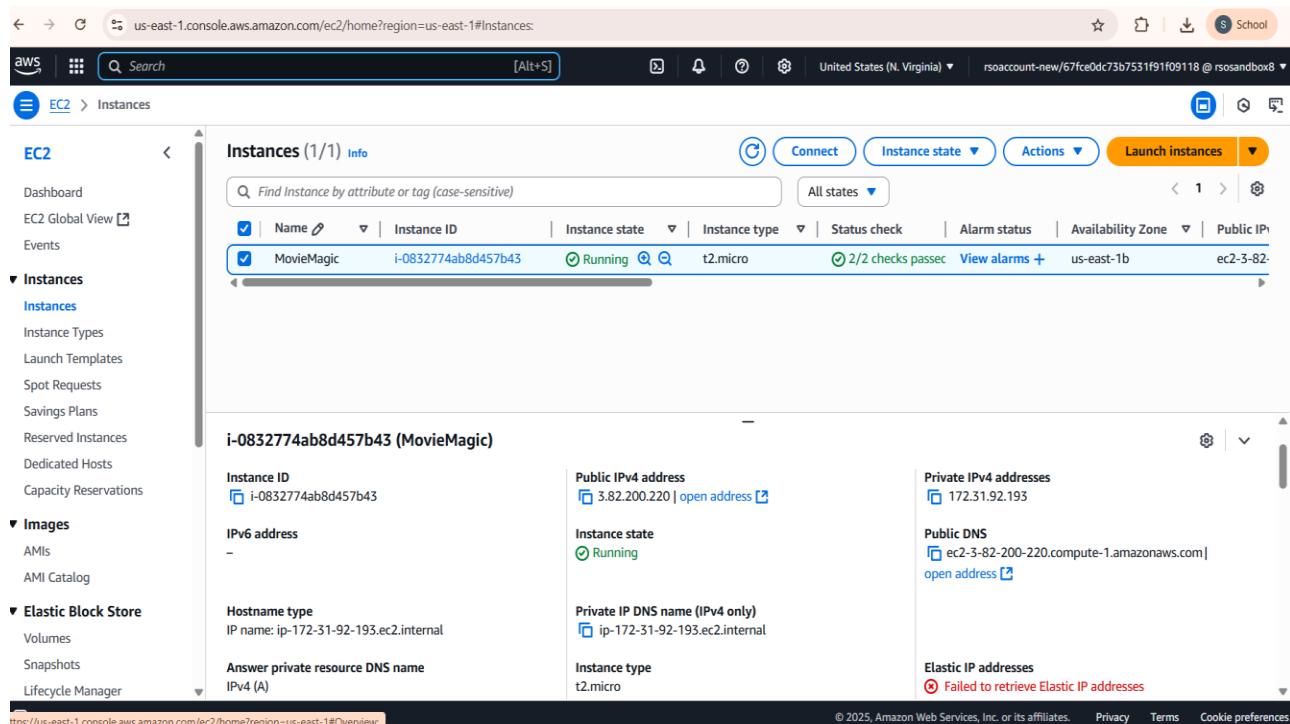


[Cancel](#)

[Preview changes](#)

[Save rules](#)

- 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.



Instances (1/1) [Info](#)

Find Instance by attribute or tag (case-sensitive)

All states [▼](#)

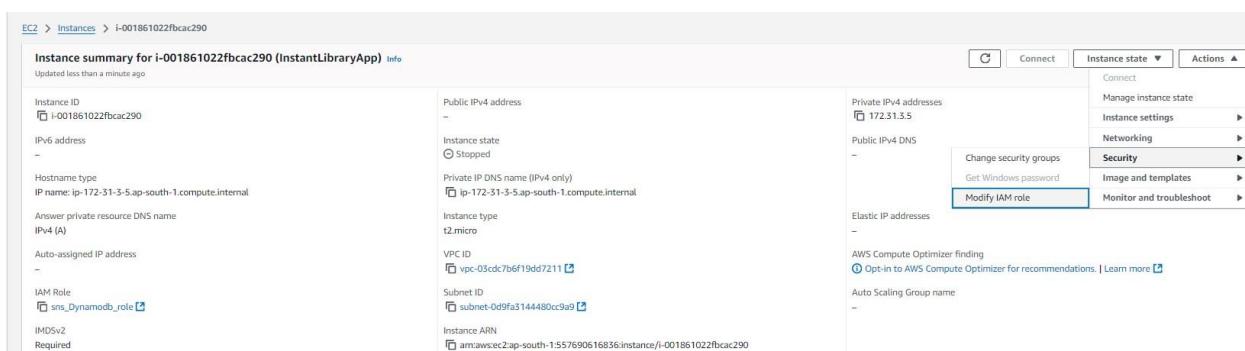
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
MovieMagic	i-0832774ab8d457b43	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-3-82-200-220.compute-1.amazonaws.com

i-0832774ab8d457b43 (MovieMagic)

Instance ID i-0832774ab8d457b43	Public IPv4 address 3.82.200.220 open address	Private IPv4 addresses 172.31.92.193
IPv6 address -	Instance state Running	Public DNS ec2-3-82-200-220.compute-1.amazonaws.com open address
Hostname type IP name: ip-172-31-92-193.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-92-193.ec2.internal	Elastic IP addresses Failed to retrieve Elastic IP addresses
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	

<https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Overview>

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)



Instance summary for i-001861022fbcac290 (InstantLibraryApp) [Info](#)

Updated less than a minute ago

Instance ID i-001861022fbcac290	Public IPv4 address -	Private IPv4 addresses 172.31.3.5
IPv6 address -	Instance state Stopped	Public IPv4 DNS -
Hostname type IP name: ip-172-31-3-5.ap-south-1.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-3-5.ap-south-1.compute.internal	Elastic IP addresses -
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto-assigned IP address -	VPC ID vpc-03cdc7b6f19dd7211	Auto Scaling Group name -
IAM Role sns_Dynamodb_role	Subnet ID subnet-0d9fa3144480cc9a9	Instance ARN arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290



AWS | Search [Alt+S] | United States (N. Virginia) ▾ rsoaccount-new/67fce0dc73b7531f91f09118 @ rsosandboxnew47 ▾

EC2 > Instances > i-08aebe32135699cb > Modify IAM role

Modify IAM role Info

Attach an IAM role to your instance.

Instance ID
 i-08aebe32135699cb (Moviemagic)

IAM role
Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

StudentUser Create new IAM role

[Cancel](#) [Update IAM role](#)

- Now connect the EC2 with the files

Connect to instance Info

Connect to your instance i-001861022fbcac290 (InstantLibraryApp) using any of these options

[EC2 Instance Connect](#) | [Session Manager](#) | [SSH client](#) | [EC2 serial console](#)

⚠ Port 22 (SSH) is open to all IPv4 addresses

Port 22 (SSH) is currently open to all IPv4 addresses, indicated by **0.0.0.0/0** in the inbound rule in [your security group](#). For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: 13.233.177.0/29. [Learn more](#).

Instance ID
 i-001861022fbcac290 (InstantLibraryApp)

Connection Type

Connect using EC2 Instance Connect
 Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.

Connect using EC2 Instance Connect Endpoint
 Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Public IPv4 address
 13.200.229.59

IPv6 address

Username
 Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.
 X

Note: In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

[Cancel](#) [Connect](#)

```
A newer release of "Amazon Linux" is available.
Version 2023.6.20241010:
Run "/usr/bin/dnf check-release-update" for full release and version update info
#                                     Amazon Linux 2023
#                                     https://aws.amazon.com/linux/amazon-linux-2023
Last login: Tue Oct 15 04:17:59 2024 from 13.233.177.3
[ec2-user@ip-172-31-3-5 ~]$
```

i-001861022fbcac290 (InstantLibraryApp)
 PublicIPs: 13.201.74.42 PrivateIPs: 172.31.3.5

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/Akhil000-ak/MovieMagic.git>

- This will download your project to the EC2 instance.

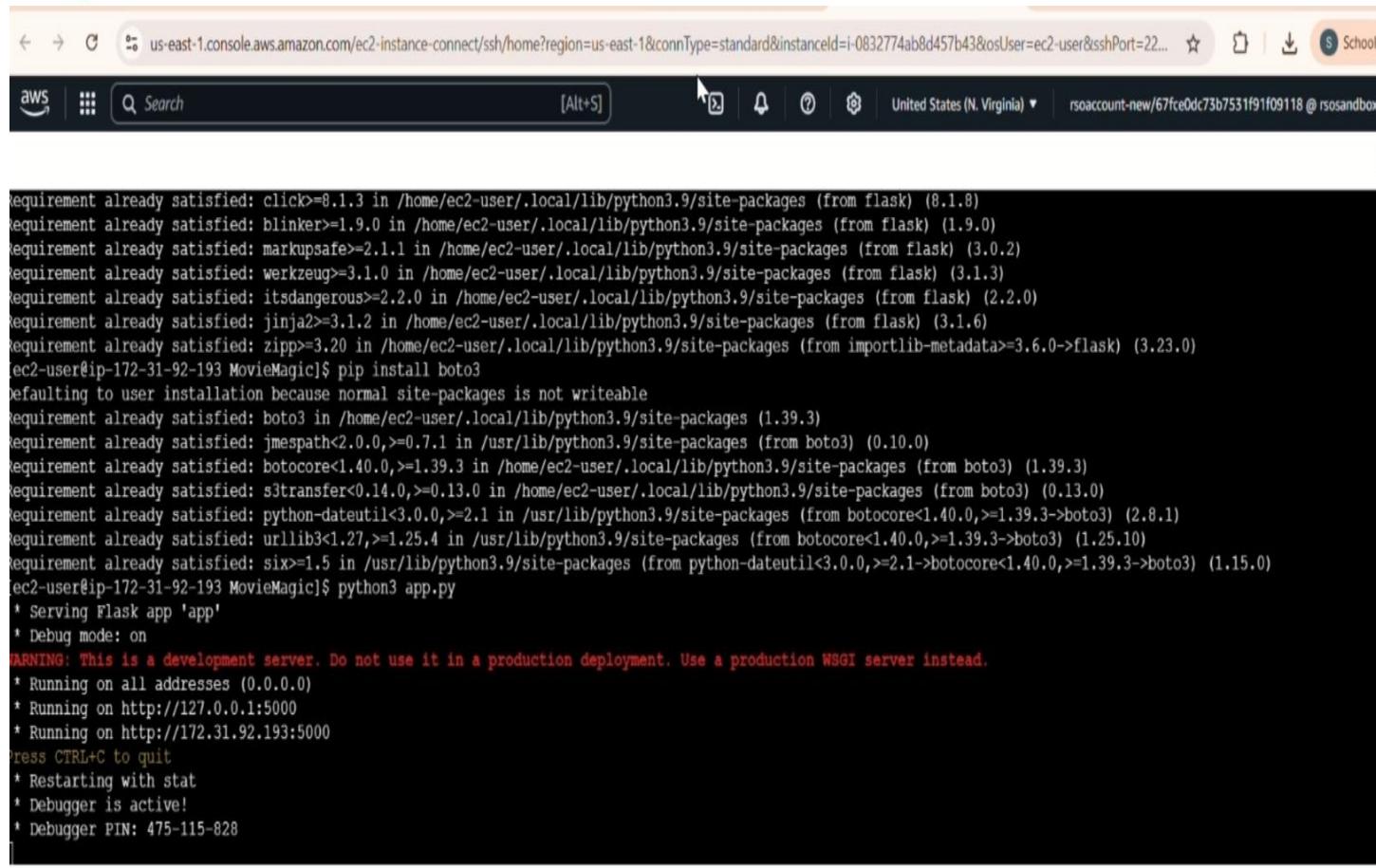
To navigate to the project directory, run the following command:

```
cd MovieMagic
```

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
```



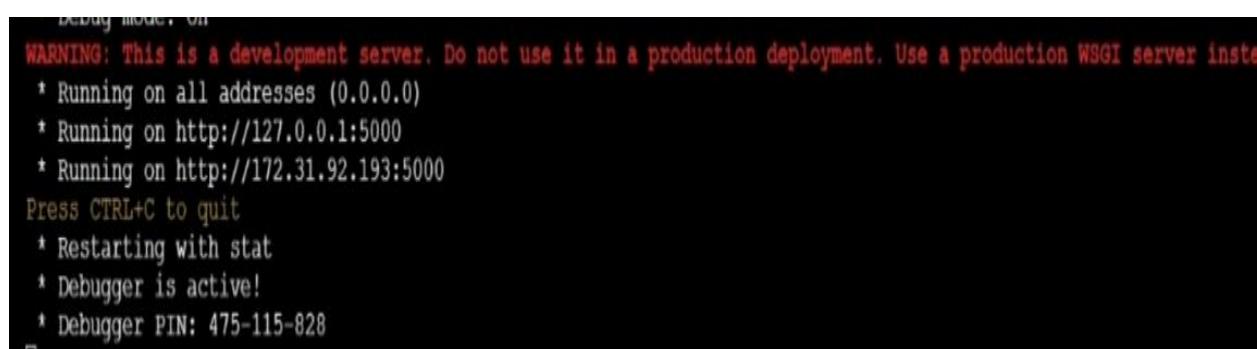
```

requirement already satisfied: click>=8.1.3 in /home/ec2-user/.local/lib/python3.9/site-packages (from flask) (8.1.8)
requirement already satisfied: blinker>=1.9.0 in /home/ec2-user/.local/lib/python3.9/site-packages (from flask) (1.9.0)
requirement already satisfied: markupsafe>=2.1.1 in /home/ec2-user/.local/lib/python3.9/site-packages (from flask) (3.0.2)
requirement already satisfied: werkzeug<3.1.0 in /home/ec2-user/.local/lib/python3.9/site-packages (from flask) (3.1.3)
requirement already satisfied: itsdangerous>=2.2.0 in /home/ec2-user/.local/lib/python3.9/site-packages (from flask) (2.2.0)
requirement already satisfied: jinja2>=3.1.2 in /home/ec2-user/.local/lib/python3.9/site-packages (from flask) (3.1.6)
requirement already satisfied: zipp>=3.20 in /home/ec2-user/.local/lib/python3.9/site-packages (from importlib-metadata>=3.6.0->flask) (3.23.0)
[ec2-user@ip-172-31-92-193 MovieMagic]$ pip install boto3
defaulting to user installation because normal site-packages is not writeable
requirement already satisfied: boto3 in /home/ec2-user/.local/lib/python3.9/site-packages (1.39.3)
requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3.9/site-packages (from boto3) (0.10.0)
requirement already satisfied: botocore<1.40.0,>=1.39.3 in /home/ec2-user/.local/lib/python3.9/site-packages (from boto3) (1.39.3)
requirement already satisfied: s3transfer<0.14.0,>=0.13.0 in /home/ec2-user/.local/lib/python3.9/site-packages (from boto3) (0.13.0)
requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.3->boto3) (2.8.1)
requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.3->boto3) (1.25.10)
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,>=1.39.3->boto3) (1.15.0)
[ec2-user@ip-172-31-92-193 MovieMagic]$ 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://172.31.92.193:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 475-115-828

```

Verify the Flask app is running: <http://your-ec2-public-ip>

- Run the Flask app on the EC2 instance



```

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://172.31.92.193:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 475-115-828

```

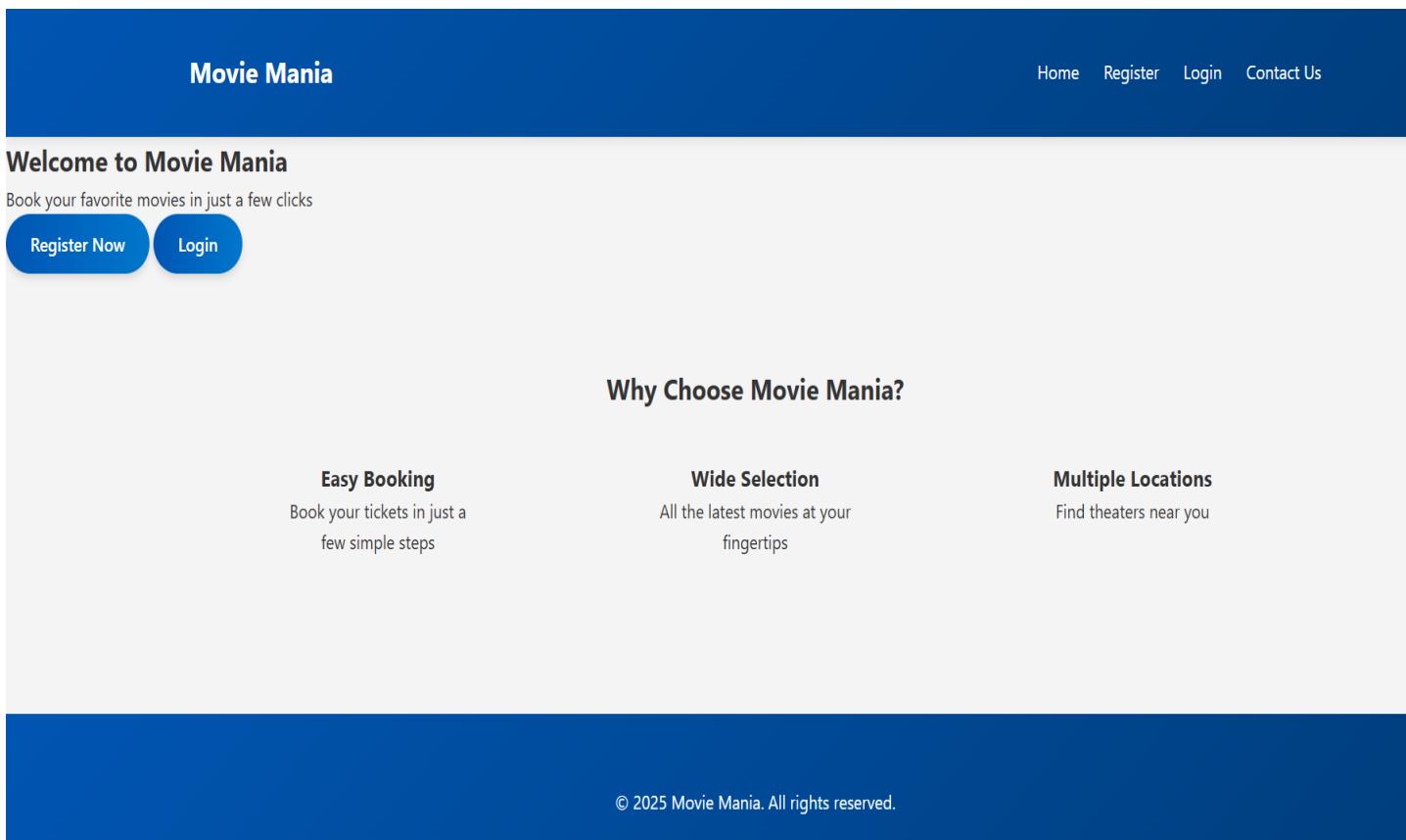
Access the website through:

Public ip: <https://3.82.200.220:5000>

Milestone 8: Testing and Deployment

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

Index Page:



The screenshot shows the homepage of Movie Mania. At the top, there is a dark blue header bar with the title "Movie Mania" on the left and navigation links "Home", "Register", "Login", and "Contact Us" on the right. Below the header, the main content area has a white background. It features a section titled "Welcome to Movie Mania" with the subtext "Book your favorite movies in just a few clicks". Two blue rounded rectangular buttons are present: "Register Now" on the left and "Login" on the right. Further down, there is a section titled "Why Choose Movie Mania?" containing three items: "Easy Booking" (described as booking tickets in simple steps), "Wide Selection" (described as having the latest movies at fingertips), and "Multiple Locations" (described as finding theaters near you). At the bottom of the page, a dark blue footer bar contains the copyright notice "© 2025 Movie Mania. All rights reserved."

Login page



Movie Mania

Home Register Login Contact Us

Login to Your Account

Email Address

Password

Login

Don't have an account? [Register](#)

Registration page:

Movie Mania

Home Register Login Contact Us

Create Your Account

Full Name

Email Address

Phone Number

Password

Confirm Password

Register

Already have an account? [Login](#)

Home page:

Movie Mania

Select City

Now Showing



Kubeera
Action, Drama, Thriller



Kannappa
Action, Drama, Fantasy



Sitaare Zameen Par
Comedy, Drama, Sports



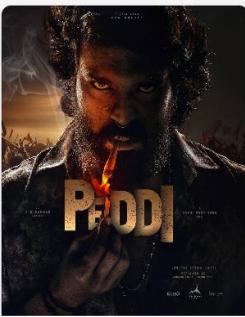
**Final Destination:
Bloodlines**
Horror, Supernatural, Thriller

Movie Mania

Coming Soon



They Call Him OG
Action, Drama, Violence



Peddi
Action, Thriller



Raaja Saab
Horror, Comedy

Booking Page:

Kubeera



Genre: Action, Drama, Thriller
Duration: 150 min
Rating: 8.5/10
Synopsis: A gripping tale of power and redemption.
Base Price: ₹180

Book Your Tickets

Date:

Time:

SCREEN									
A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
E1	E2	E3	E4	E5	E6	E7	E8	E9	E10
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
G1	G2	G3	G4	G5	G6	G7	G8	G9	G10

■ Available ■ Selected ■ Occupied

Selected Seats: 3
 Total Price: ₹810

Confirm Booking

Movie Mania

[Home](#) [Register](#) [Login](#) [Contact Us](#)

Your Movie Ticket

Thank you for booking with Movie Mania



Movie: Kubeera

Date: 2025-07-12

Time: 4:00 PM

Seats: A1 (₹270), A2 (₹270), A3 (₹270)

Price: ₹810

[Download Ticket](#)

Download Ticket:

Movie Mania - Movie Ticket

Movie: Kubeera

Date: 2025-07-12

Time: 1:30 PM

Seats: C6 (₹270), C7 (₹270), C8 (₹270)

Price: ₹810

Thank you for booking with Movie Mania!

Contact Page



CineBook

Home Register Login Contact Us

Contact Us

Email Us
support@moviemania.com

Call Us
+91 8464373223

Visit Us
5-110, Zinna Tower
Guntur, Andhra Pradesh

Send Us a Message

Your Name

Your Email

Your Message

Send Message

Dynamodb Database updatons :

1. Users table :

Users						
Scan or query items						
Expand to query or scan items.						
Completed. Read capacity units consumed: 0.5						
Items returned (3)						
<input type="checkbox"/>	email (String)	▼	login_count	▼	name	▼
<input type="checkbox"/>	penubakulaalekhy...@...	1	Alekhya		\$2b\$12\$eECDhD5je.nTmUf9DFSKouBvOxvGxd13MrI1T3rSgmzAlplyF/6T6	
<input type="checkbox"/>	alekhya080228@gma...	1	Alekhya		\$2b\$12\$5mlJnhDAh1OMlt7leSKQlau4ytsOCjPXifdQ6HdvYg6qU16G0kRhd2	
<input type="checkbox"/>	sirichakkala@gmail.com	1	Siri Chakkala		\$2b\$12\$IVMRmgzb0tUp8U3KdDHrOUh5w5/ApXg5LoF7k63cvWSWqOdgrs6	

2. Request Table

Movie Booking Requests				
movie_name	description	name	booking_date	status
Kubeera	booking id 1	Alekhya	2012-06-13	Pending
Annappa	booking id 2	Alekhya	2017-04-24	Completed
Hug Life	booking id 3	Siri Chakkala	2024-05-19	Pending

3. Mail to the User:

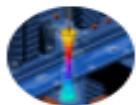
[Back to Inbox](#) **(no subject)** [Inbox](#)

 **Akhil Kumar**
 to me ▾
 Dear alekhya,
 Thank you for booking movie tickets through movie mania
 Movie Mania - Movie Ticket
 Movie: Kubeera
 Date: 2025-07-12
 Time: 4:00 PM
 Seats: A1 (₹270), A2 (₹270), A3 (₹270)
 Price: ₹810
 Thank you for booking with Movie Mania!

4. Mail to the admin:

New booking

◆ Summarize



Akhil Kumar <akhillovely471@gmail.com>

to alekhya325@gmail.com ▾

Movie: Kubeera

Date: 2025-07-12

Time: 4:00 PM

Seats: A1 (₹270), A2 (₹270), A3 (₹270)

Price: ₹810

Conclusion:

The Movie Magic website has been successfully developed and deployed using a robust, cloud-native architecture. By leveraging key AWS services such as EC2 for hosting, DynamoDB for real-time data management, and SNS for instant booking confirmations, the platform delivers a scalable and user-friendly movie ticketing experience. This solution addresses the limitations of traditional ticket booking systems by enabling users to seamlessly search for movies, select seats, and receive digital tickets—all from a single platform.

The cloud-based infrastructure ensures that the system can handle a high volume of users and transactions, especially during peak movie release times, without compromising speed or reliability. The integration of Flask with AWS services enables smooth backend operations, including user authentication, movie/event browsing, seat availability checks, and secure ticket booking.

Comprehensive testing has confirmed that all core functionalities—from user registration and login to booking confirmation via email—operate seamlessly. The platform's modern interface and responsive design further enhance the user experience, making movie booking quick, efficient, and enjoyable.

In conclusion, Movie Magic stands as a powerful demonstration of how cloud technologies can modernize traditional services. It provides an efficient, scalable, and intuitive solution for movie ticket booking, showcasing the potential of full-stack cloud applications in solving real-world user experience challenges in the entertainment industry.