



## 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 Ticket 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

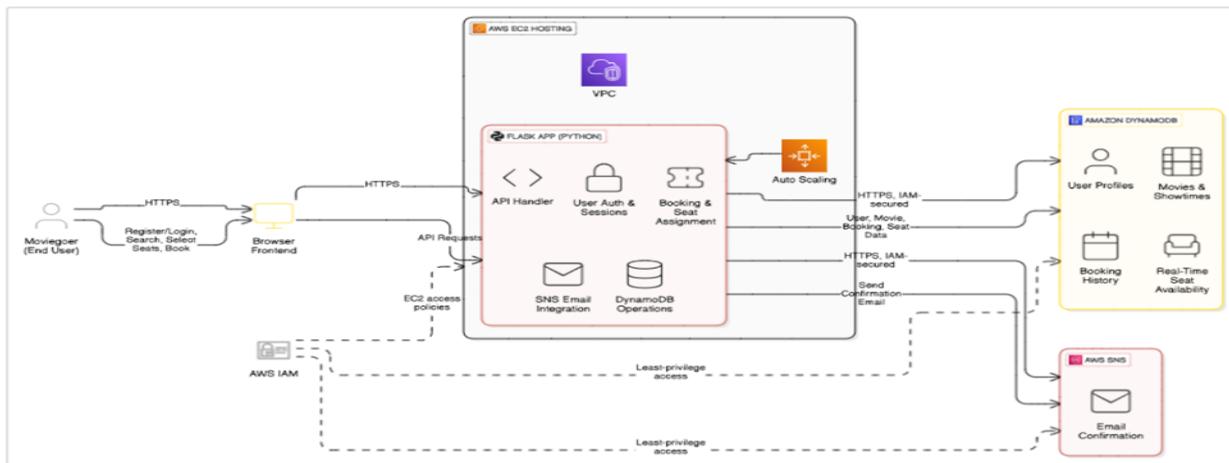
When a user completes a ticket booking, the Movie Magic System leverages AWS SNS to send instant email notifications to confirm the booking. For instance, once the booking is submitted, Flask processes the transaction, and SNS sends a customized email to the

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.

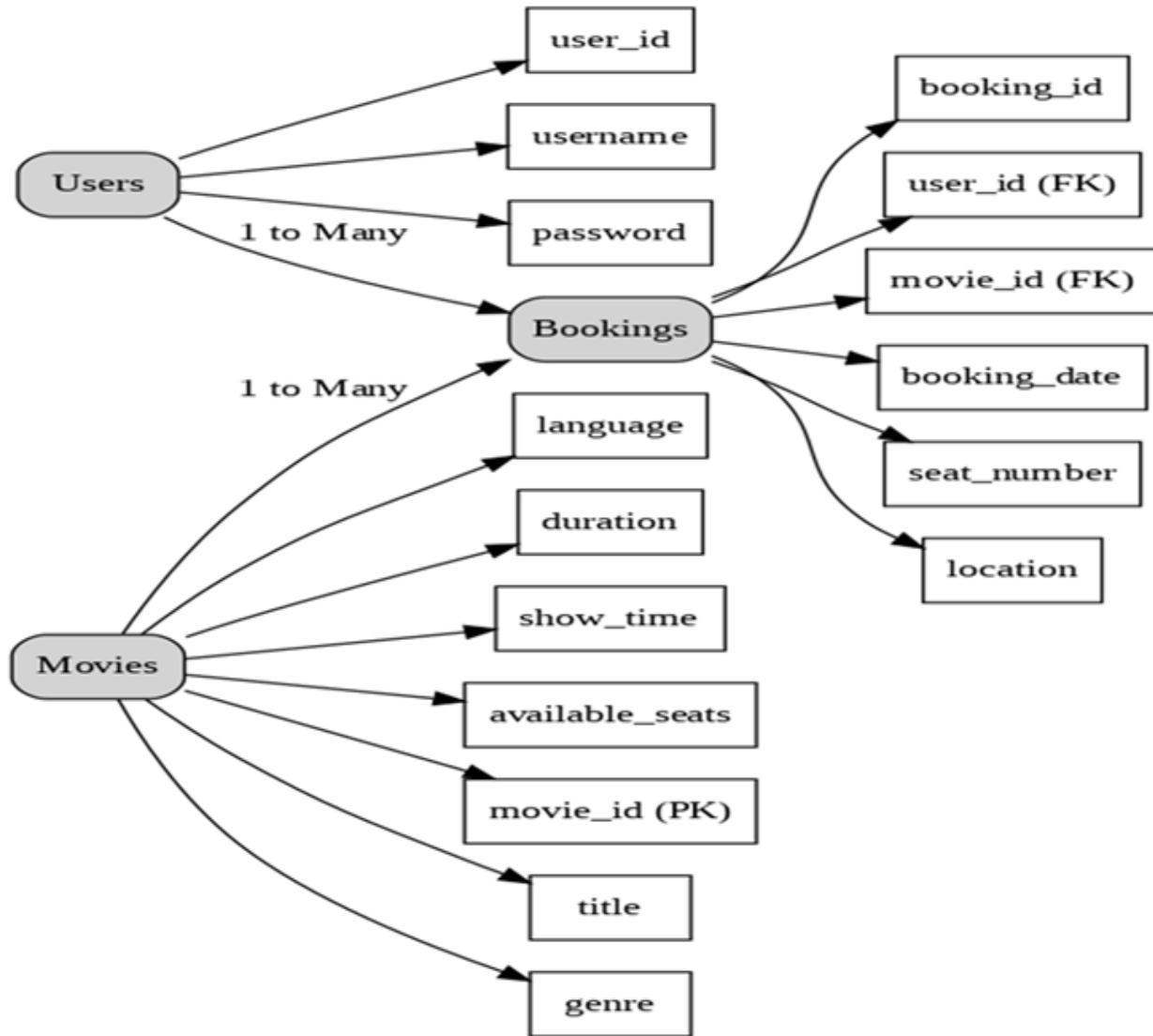
### Scenario 3: Easy Access to Movies and Events

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 (ER)Diagram:



## Pre-requisites:

### 1. AWS Account Setup:

<https://docs.aws.amazon.com/accounts/latest/reference/getting-started.html>

### 2. AWS IAM (Identity and Access Management) :

<https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html>

### 3. AWS EC2 (Elastic Compute Cloud) :

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>

### 4. AWS DynamoDB :

5. <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html>

### 6. Amazon SNS :

<https://docs.aws.amazon.com/sns/latest/dg/welcome.html>

### 7. Git Documentation :

<https://git-scm.com/doc>

### 8. VS Code Installation : (download the VS Code using the below link or you can get that in Microsoft store): <https://code.visualstudio.com/download>

## **Project WorkFlow:**

### **1. AWS AccountSetup and Login :**

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

**Activity 1.2:** Log in to the AWS Management Console.

### **2. DynamoDB Database Creation and Setup :**

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

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

### **3. SNS Notification Setup :**

**Activity 3.1:** Create SNS topics for book request notifications.

**Activity 3.2:** Subscribe users and library staff to SNS email notifications.

### **4. Backend Development and Application Setup :**

**Activity 4.1:** Develop the Backend Using Flask.

**Activity 4.2:** Integrate AWS Services Using boto3.

### **5. IAM Role Setup :**

**Activity 5.1:** Create IAM Role

**Activity 5.2:** Attach Policies

## 6. EC2 Instance Setup :

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

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

## 7. Deployment on EC2 :

**Activity 7.1:** Upload Flask Files

**Activity 7.2:** Run the Flask App

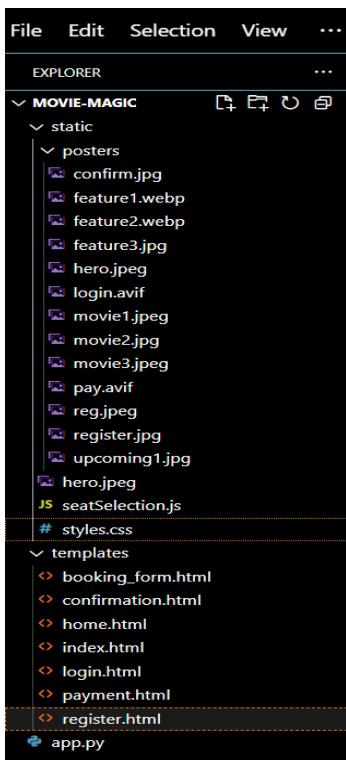
## 8. Testing and Deployment :

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

## Milestone 1: Web Application Development and Setup

- Local Deployment :

File Explorer Structure



### Description of the code :

- Flask App Initialization

Imports and Configuration:

```
app.py > ...
1  from flask import Flask, render_template, request, redirect, session, flash, url_for
2  import hashlib
3  import uuid
4  import boto3
```

**Description:** This project uses Flask for routing, session management, and user authentication with secure password hashing. It integrates AWS services via Boto3 for handling data storage, notifications, and unique user operations.

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

**Description:** A new Flask application instance is initialized, and a secret key is set to securely manage user sessions and protect against cookie tampering.

- **Dynamodb and SNS Setup :**

```
app = Flask(__name__)
app.secret_key = 'super-secret-key'
dynamodb = boto3.resource('dynamodb',region_name='us-east-1') # e.g., 'ap-south-1'

users_table = dynamodb.Table("user")

bookings_table = dynamodb.Table("moviebooking")
```

**Description:** Use **boto3** to connect to **DynamoDB** for handling user registration, movie bookings database operations and also mention **region\_name** where Dynamodb tables are created.

- **SNS Connection :**

**Description:** Configure **SNS** to send notifications when a movie ticket is booked. 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.

```
sns = boto3.client('sns', region_name='us-east-1')

sns_topic_arn = 'arn:aws:sns:us-east-1:216989138822:MovieTicketNotifications fifo'
```

- **Function to send the Notifications:**

**Description:** This function sends a booking confirmation email using AWS SNS. It formats the booking details into a message and publishes it to a specified SNS topic, notifying the user via email about their successful movie ticket booking.

```
# ----- Helper Functions -----
def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

def send_mock_email(email, booking_info):
    print(f"[MOCK EMAIL] Sent to {email}:\nBooking confirmed for {booking_info['movie']}\n"
          f"Seat: {booking_info['seat']}, Date: {booking_info['date']}, Time: {booking_info['time']}\n"
          f"Booking ID: {booking_info['id']}")
```

- **Routes for Web Pages**

**Register User:** Collecting registration data, hashes the password, and stores user details in the database.

```
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']

        if email in mock_users:
            flash("Account already exists.")
            return redirect(url_for('login'))

        mock_users[email] = hash_password(password)
        flash("Account created! Please login.")
        return redirect(url_for('login'))
    return render_template('register.html')
```

- **login Route (GET/POST):** Verifies user credentials, increments login count, and redirects to the dashboard on success

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        hashed = hash_password(password)

        print("Submitted email:", email)
        print("Submitted hash:", hashed)
        print("Registered users:", mock_users)

        if email in mock_users and mock_users[email] == hashed:
            session['user'] = email
            print("Login successful!")
            return redirect(url_for('home'))
        else:
            flash("Invalid email or password.")
            print("Login failed.")
            return render_template('login.html')

    return render_template('login.html')
```

These Flask routes handle key navigation in the app: `/logout` logs out the user by clearing the session and showing a flash message; `/home1` is a protected route accessible only to logged-in users.

```
@app.route('/logout')
def logout():
    session.clear()
    flash("You have been logged out.")
    return redirect(url_for('index'))
```

```
@app.route('/home')
def home():
    if 'user' not in session:
        |   return redirect(url_for('login'))

    now_showing = [
        {"title": "The Grand Premiere", "genre": "Drama", "poster": "posters/movie1.jpeg", "duration": "2h 10m", "rating": "4.5", "synopsis": "A hea..."}
        {"title": "Engaging", "genre": "Drama", "poster": "posters/movie2.jpg", "duration": "1h 45m", "rating": "4.2", "synopsis": "A hilarious ride..."}
    ]
    coming_soon = [
        {"title": "Future Flick", "genre": "Sci-Fi", "poster": "posters/upcoming1.jpg", "duration": "2h 20m", "rating": "N/A", "synopsis": "A mind-b..."}
    ]
    top_rated = [
        {"title": "Edge of Tomorrow", "genre": "Action", "poster": "posters/movie3.jpeg", "duration": "2h", "rating": "4.8", "synopsis": "A soldier..."}
    ]
    return render_template('home.html', now_showing=now_showing, coming_soon=coming_soon, top_rated=top_rated)
```

- **Booking Page Route:**

**Description:** This route displays the booking page (`b1.html`) with movie, theater, address, and

price details passed as query parameters. It ensures only logged-in users can access the page.

```
@app.route('/booking', methods=['GET', 'POST'])
def booking():
    if 'user' not in session:
        return redirect(url_for('login'))

    if request.method == 'POST':
        # Temporarily store booking form in session
        session['pending_booking'] = {
            'movie': 'Example Movie',
            'seat': request.form['seat'],
            'date': request.form['date'],
            'time': request.form['time']
        }
        return redirect(url_for('payment'))

    return render_template('booking_form.html', movie='Example Movie')
```

- **Tickets Page Route:**

**Description:** This route processes movie ticket bookings by collecting form data, generating a unique booking ID, storing details in DynamoDB, and sending a confirmation email via AWS SNS. It then displays the booking details on the tickets page.

```
@app.route('/payment', methods=['GET', 'POST'])
def payment():
    if 'user' not in session or 'pending_booking' not in session:
        return redirect(url_for('login'))

    if request.method == 'POST':
        # Pretend to process the fake payment
        booking_info = session['pending_booking']
        booking_info['user'] = session['user']
        booking_info['id'] = str(uuid.uuid4())[:8]

        mock_bookings.append(booking_info)
        session['last_booking'] = booking_info
        send_mock_email(session['user'], booking_info)
        session.pop('pending_booking', None)
        flash("Payment successful. Ticket booked!")

    return redirect(url_for('confirmation'))

    return render_template('payment.html')
```

- **Confirmation page for movie ticket booking :**

```
@app.route('/confirmation')
def confirmation():
    if 'user' not in session or 'last_booking' not in session:
        return redirect(url_for('login'))

    booking = session['last_booking']
    return render_template('confirmation.html', booking=booking)
```

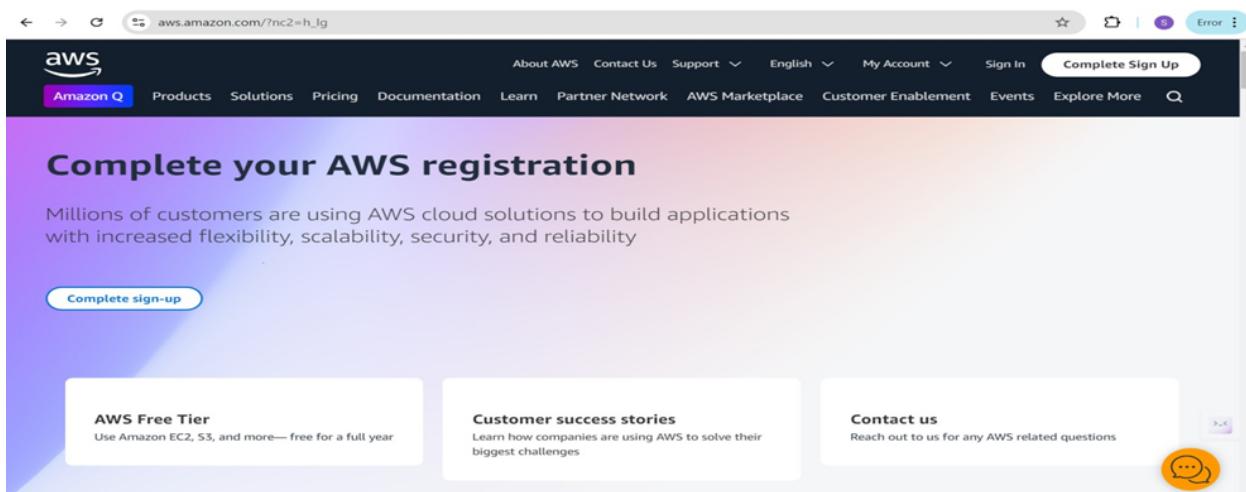
### Application Entry point:

```
if __name__ == '__main__':
    print("Mock MovieMagic running at http://127.0.0.1:5000")
    app.run(debug=True)
```

**Description:** This block starts the Flask application using the built-in development server, setting the host, port, and enabling debug mode for easier development and testing.

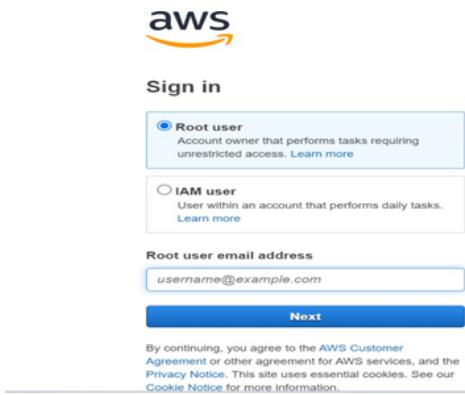
## Milestone 2: AWS AccountSetup and Login :

- **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](#).



The image shows the AWS Sign In page. It features the AWS logo at the top left. Below it is a 'Sign in' section with two radio button options: 'Root user' (selected) and 'IAM user'. Under 'Root user', there is a note about being the account owner and performing tasks requiring unrestricted access. Under 'IAM user', there is a note about being a user within an account. A 'Root user email address' input field contains 'username@example.com'. A blue 'Next' button is below the input field. At the bottom, a small note states: 'By continuing, you agree to the AWS Customer Agreement or other agreement for AWS services, and the Privacy Notice. This site uses essential cookies. See our Cookie Notice for more information.'

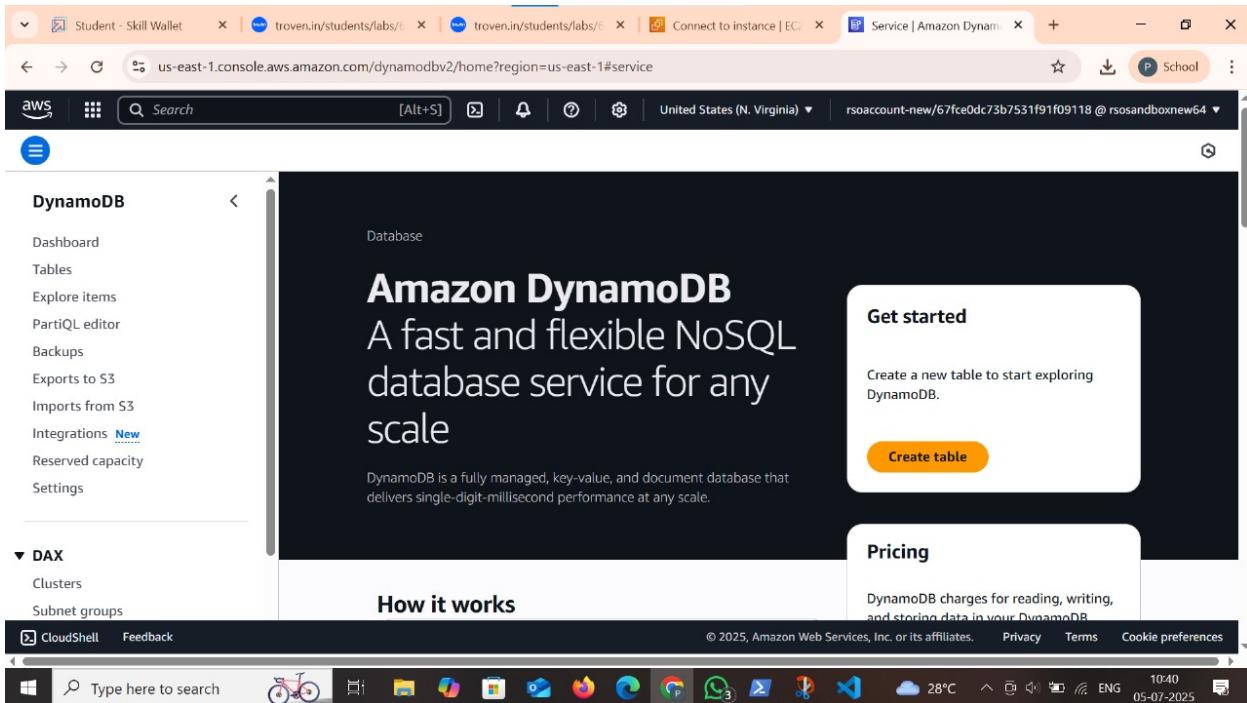


The image shows a dark-themed promotional card for the 'AI Use Case Explorer'. The title 'AI Use Case Explorer' is at the top in white. Below it, the text reads: 'Discover AI use cases, customer success stories, and expert-curated implementation plans'. At the bottom, a blue 'Explore now >' button is visible.

## Milestone 3: DynamoDB Database Creation and Setup

- **Activity 3.1: Navigate to the DynamoDB**

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



The image shows the Amazon DynamoDB dashboard. The browser address bar indicates the URL is 'us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#service'. The main content area has a dark background with the heading 'Amazon DynamoDB' and the subtext 'A fast and flexible NoSQL database service for any scale'. A 'Get started' call-to-action button is visible. On the left, a sidebar menu includes 'Dashboard', 'Tables', 'Explore items', 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', 'Integrations', 'Reserved capacity', and 'Settings'. A 'How it works' section is also present. The bottom of the screen shows the Windows taskbar with various pinned icons like File Explorer, Edge, and File History.



● **Activity 3.2: Create a DynamoDB table for storing registration details and book requests.**



Screenshot of the AWS DynamoDB 'Create table' wizard.

**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**

Screenshot of the AWS DynamoDB 'Tables' page.

**DynamoDB**

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

**DAX**

- Clusters
- Subnet groups

**Tables (2/2)** Info

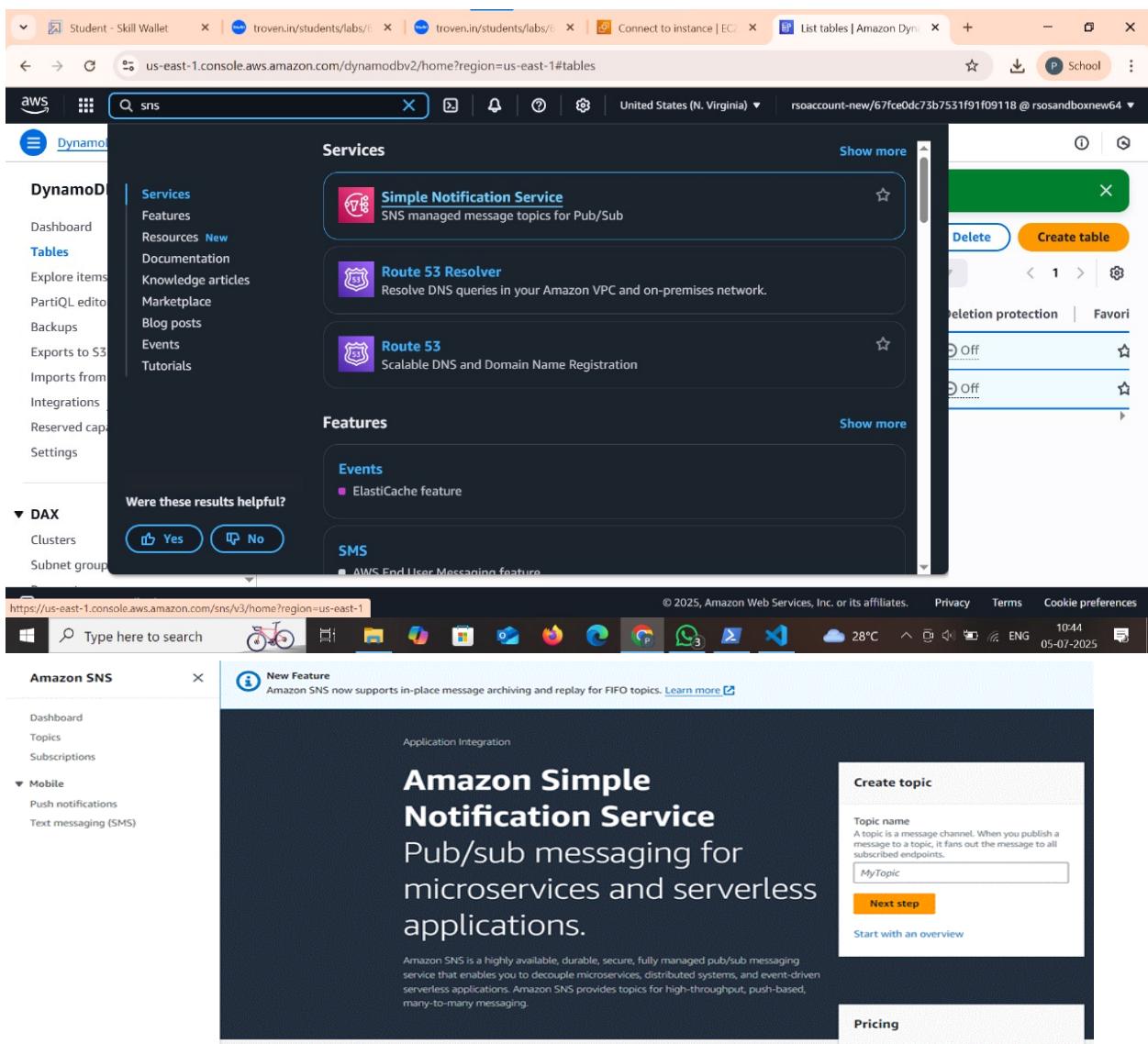
The bookings table was created successfully.

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favori
<a href="#">bookings</a>	Active	Booking_id (S)	-	0	0	Off	★
<a href="#">users</a>	Active	Email (S)	-	0	0	Off	★

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## Milestone 4 : SNS Notification Setup :

- **Activity 4.1: 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.



The screenshot shows the AWS Console interface. The search bar at the top contains the text "sns". The left sidebar shows the "DynamoDB" section with "Tables" selected. The main content area displays the "Simple Notification Service" and "Route 53 Resolver" services. A sidebar on the right shows a table with two items, both set to "Off". At the bottom, the Amazon SNS dashboard is shown with a "Create topic" button and a "Pricing" section.

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



The screenshot shows the 'Amazon SNS' interface. On the left, there's a sidebar with links like 'Dashboard', 'Topics', 'Subscriptions', 'Mobile' (with 'Push notifications' and 'Text messaging (SMS)'), and 'Create topic'. The main area is titled 'Topics (0)' and includes a search bar, a table header with columns 'Name' and 'Type', and a note 'No topics. To get started, create a topic.' A prominent orange button at the bottom right says 'Create topic'.

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

The screenshot shows the 'Create topic' page for Amazon SNS. It has two main sections: 'FIFO (first-in, first-out)' (selected) and 'Standard'. The FIFO section lists: 'Strictly-preserved message ordering', 'Exactly-once message delivery', and 'Subscription protocols: SQS'. The Standard section lists: 'Best-effort message ordering', 'At-least once message delivery', and 'Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SMS, email, mobile application endpoints'. Below these is a 'Name' input field containing 'MovieTicketNotifications.fifo'. Underneath it, a note says 'Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (\_). FIFO topic names must end with ".fifo".' There's also a 'High throughput' section with an info link and a note about configuring for maximum throughput. At the bottom, there are sections for 'Message group scope' (selected) and 'Topic scope', each with its own bullet points. The page footer includes standard AWS links like CloudShell, Feedback, and a search bar, along with system status information like temperature (28°C), time (10:46), and date (05-07-2025).

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



Amazon SNS > Topics > MovieTicket fifo

**MovieTicket fifo**

**Details**

Name	Display name	Retention policy
MovieTicket fifo	Movie_Ticket_Notifications	Inactive

**ARN**: arn:aws:sns:us-east-1:975050261480:MovieTicket fifo

**Type**: FIFO

**Topic owner**: 975050261480

**Content-based message deduplication**: Disabled

**Throughput scope**: MessageGroup

Subscriptions | Access policy | Archive policy | Delivery status logging | Encryption | Tags

## ● Subscribe Users And Admin

Subscribe users (or admin staff) to this topic via Email. When a movie ticket is booked, notifications will be sent to the user's emails.

Create subscription | Subscriptions

Topic ARN: arn:aws:sns:us-east-1:975050261480:MovieTicketNotifications

Protocol: Email

Endpoint: deekshithapotti2002@gmail.com

After your subscription is created, you must confirm it.

Subscription filter policy - optional

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



## AWS Notification - Subscription Confirmation Spam x

AWS Notifications <no-reply@sns.amazonaws.com>  
to me ▾

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

Report as not spam

You have chosen to subscribe to the topic:

**arn:aws:sns:ap-south-1:605134430972:MovieTicketNotifications**

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](#)

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

## 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.

The screenshot shows the AWS IAM console with the 'Modify IAM role' wizard. The 'IAM' service is selected. The role name is 'I-068d390c27a3360b'. The 'Description' field is empty. The 'Permissions boundary' dropdown is set to 'None'. The 'Trust relationship' section is collapsed. The 'Next Step' button is visible at the bottom right.

The screenshot shows a browser window with multiple tabs open. The tabs include 'Student - Skill Wall', 'troven.in/students', 'troven.in/students', 'Modify IAM role | x', 'Create role | IAM | x', 'Roles | IAM | Global | x', and 'School'. The main content area is titled 'Create role' under 'IAM > Roles > Create role'. It shows three options for trust policies: 'AWS Lambda', 'AWS Single Sign-On', and 'Custom trust policy'. Below this is a 'Use case' section for EC2, with 'EC2' selected. A sidebar on the left lists steps: 'Select trusted entity', 'Step 2', 'Add permissions', and 'Name, review, and create' (which is currently selected). The bottom of the screen shows a Windows taskbar with various icons and system status.

## ● Activity 5.2: Attach Policies.

Attach the following policies to the role:

The screenshot shows the 'Name, review, and create' step of the IAM role creation wizard. The 'Role name' field is set to 'movie\_role'. The 'Description' field contains the text 'Allows EC2 instances to call AWS services on your behalf.' In the 'Step 1: Select trusted entities' section, there is a 'Trust policy' field containing the JSON policy shown below. The bottom of the screen shows a Windows taskbar with various icons and system status.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "sts:AssumeRole"
        }
    ]
}
```

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



Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached as
<a href="#">AmazonDynamoDBFullAccess</a>	AWS managed	Permissions policy
<a href="#">AmazonSNSFullAccess</a>	AWS managed	Permissions policy

Step 3: Add tags

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

## Milestone 6: EC2 InstanceSetup :

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

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

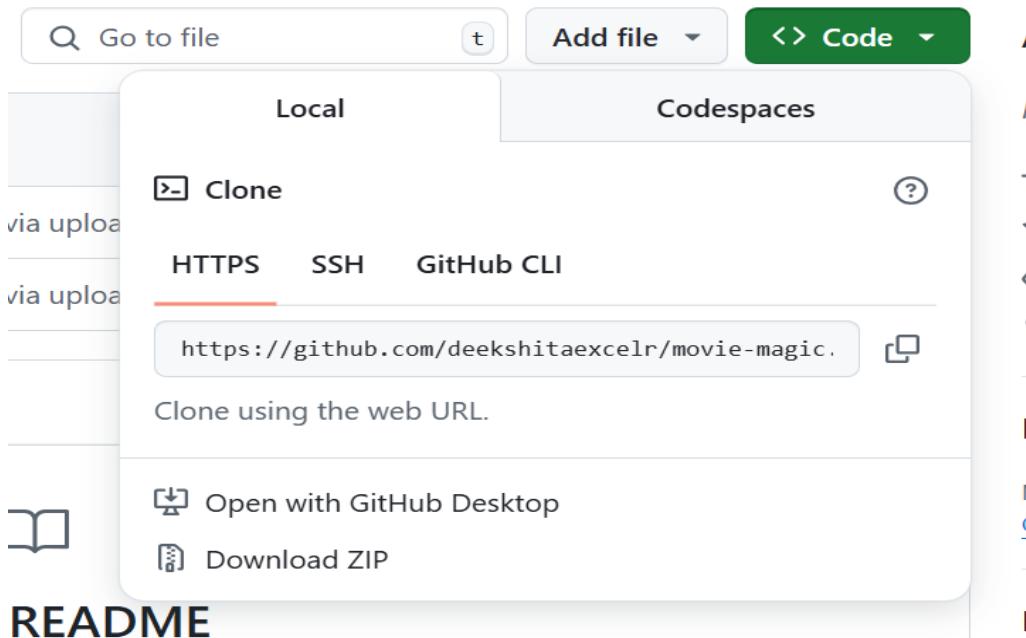
Files

main .. static templates app.py

movie-magic / movie-magic /

deekshitaexcelr Add files via upload d90ed7d · 4 days ago History

Name	Last commit message	Last commit date
..		
static	Add files via upload	4 days ago
templates	Add files via upload	4 days ago
app.py	Add files via upload	4 days ago



## README

- **Launch an EC2 instance to host the Flask**

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



- Click on Launch instance to launch EC2 instance



The screenshot shows the AWS EC2 'Launch an instance' wizard. In the 'Name and tags' step, the name 'movid' is entered. The 'Software Image (AMI)' section shows 'Amazon Linux 2023 AMI 2023.7.2...' selected. The 'Virtual server type (instance type)' is set to 't2.micro'. The 'Launch instance' button is highlighted.

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

The screenshot shows the 'Amazon Machine Image (AMI)' selection screen. It lists several options: Amazon Linux (selected), macOS, Ubuntu, Windows, Red Hat, and others. A 'Browse more AMIs' link is visible. Below the list, the 'Amazon Linux 2023 AMI' is detailed, including its ID, boot mode (uefi-preferred), and architecture (64-bit x86). A 'Verified provider' badge is present.

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: 64-bit (x86)

Boot mode: uefi-preferred

AMI ID: ami-02b49a24cfb95941c

Verified provider



- 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	

All generations [Compare instance types](#)

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](#)

Student - Skill Wallet | troven.in/students/labs/67fc... | troven.in/students/labs/67fc... | Launch an instance | EC2 | us-e... +

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:

aws Search [Alt+S] United States (N. Virginia) rsoaccount-new/67fce0dc73b7531f91f09118 @ rsosandboxnew64

EC2 > Instances > Launch an instance

Create key pair

Key pair name: movie-magic

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 (selected) ED25519

RSA RSA encrypted private and public key pair

ED25519 ED25519 encrypted private and public key pair

Private key file format: .pem (selected) .ppk

.pem For use with OpenSSH

.ppk For use with PuTTY

Cancel Create key pair

Summary

Image (AMI)

Server type (instance type)

Security group

Volume

Launch instance Preview code

CloudShell Feedback

Type here to search

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 10:02 ENG 05-07-2025



The screenshot shows the AWS EC2 'Launch an instance' wizard. On the left, under 'Auto-assign public IP', the 'Enable' option is selected. Below it, 'Additional charges apply when outside of free tier allowance'. Under 'Firewall (security groups)', there are two buttons: 'Create security group' (selected) and 'Select existing security group'. A note says: 'We'll create a new security group called 'launch-wizard-1' with the following rules:'. Three checkboxes are checked: 'Allow SSH traffic from Anywhere' (0.0.0.0/0), 'Allow HTTPS traffic from the internet' (To set up an endpoint, for example when creating a web server), and 'Allow HTTP traffic from the internet' (To set up an endpoint, for example when creating a web server). A warning message in a yellow box says: '⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.' On the right, the 'Summary' section shows 'Number of instances' (1), 'Software Image (AMI)' (Amazon Linux 2023.7.2...), 'Virtual server type (instance type)' (t2.micro), and 'Firewall (security group)' (New security group). At the bottom are 'Cancel', 'Launch Instance' (highlighted in orange), and 'Preview code'.

## ● Configure security groups for HTTP and SSH for client:

The screenshot shows the AWS EC2 'Connect to instance' wizard for instance i-068d3902c27a3360b. The 'EC2 Instance Connect' tab is selected. It shows the 'Instance ID' (i-068d3902c27a3360b (movie)) and two connection options: 'Connect using a Public IP' (selected) and 'Connect using a Private IP'. Below that, 'Public IPv4 address' is listed as 3.90.210.33. There is also an 'IPv6 address' section with a minus sign. Under 'Username', it says '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.' A search bar contains 'ec2-user'. On the right, the 'Summary' section shows 'Number of instances' (1), 'Software Image (AMI)' (Amazon Linux 2023.7.2...), 'Virtual server type (instance type)' (t2.micro), and 'Firewall (security group)' (New security group). At the bottom are 'Cancel', 'Launch Instance' (highlighted in orange), and 'Preview code'.



- Now open windows posershell and navigate to the key pair download pem file path and type the command as "ssh -i "path of pem file" ec2-user@copy the 4 th point in that above image"

```
PS C:\Users\deekshi> ssh -i "C:\Users\deekshi\Downloads\movie-magic.pem" ec2-user@ec2-3-90-210-33.compute-1.amazonaws.com
The authenticity of host 'ec2-3-90-210-33.compute-1.amazonaws.com (3.90.210.33)' can't be established.
ED25519 key fingerprint is SHA256:U0Mmrtvb19j0sLMjra9183b8120CI5y7Dd0lXPvOs.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

```
[ec2-user@ip-172-31-25-120 ~]$
```

```
PS C:\Users\deekshi> ssh -i "C:\Users\deekshi\Downloads\movie-magic.pem" ec2-user@ec2-3-90-210-33.compute-1.amazonaws.com
The authenticity of host 'ec2-3-90-210-33.compute-1.amazonaws.com (3.90.210.33)' can't be established.
ED25519 key fingerprint is SHA256:U0Mmrtvb19j0sLMjra9183b8120CI5y7Dd0lXPvOs.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-90-210-33.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Connection closed by 3.90.210.33 port 22
```

```
PS C:\Users\deekshi> ssh -i "movie-magic.pem" ec2-user@ec2-3-90-210-33.compute-1.amazonaws.com
Warning: Identity file movie-magic.pem not accessible: No such file or directory.
```

```
ec2-user@ec2-3-90-210-33.compute-1.amazonaws.com: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
```

```
PS C:\Users\deekshi> ssh -i "C:\Users\deekshi\Downloads\movie-magic.pem" ec2-user@ec2-3-90-210-33.compute-1.amazonaws.com
.
.
.
Amazon Linux 2023
.
.
.
https://aws.amazon.com/linux/amazon-linux-2023
```

## 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
```

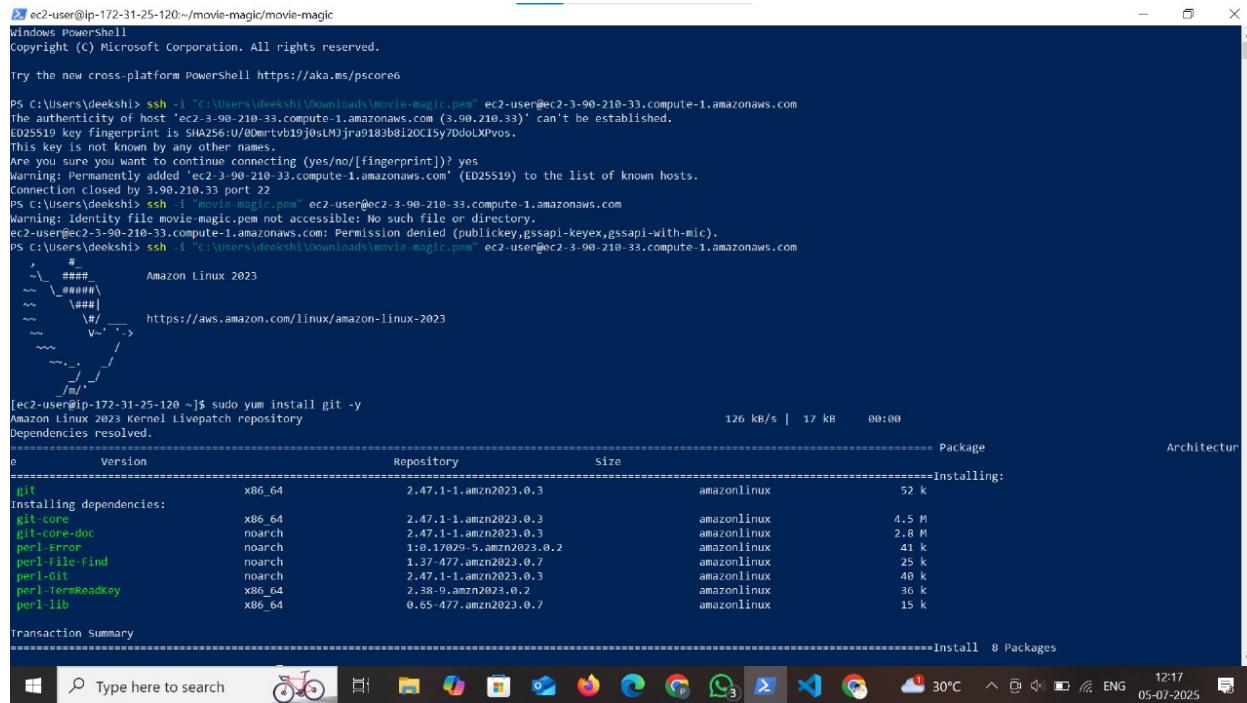
```
sudo yum install python3 git
```

```
sudo pip3 install flask boto3
```

- Verify Installations:

```
flask --version
```

```
git --version
```



```

PS C:\Users\deekshi> ssh -l "C:\Users\deekshi\Downloads\movie-magic.pem" ec2-user@ec2-3-90-210-33.compute-1.amazonaws.com
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\deekshi> ssh -l "C:\Users\deekshi\Downloads\movie-magic.pem" ec2-user@ec2-3-90-210-33.compute-1.amazonaws.com
The authenticity of host 'ec2-3-90-210-33.compute-1.amazonaws.com (3.90.210.33)' can't be established.
ED25519 key fingerprint is SHA256:U/0Mrmtvb19j0sLMjra9183b8i2OC15y/DdoLXPv09.
This key is not known by any other name.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-90-210-33.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Connection closed by 3.90.210.33 port 22
PS C:\Users\deekshi> ssh -l "C:\Users\deekshi\Downloads\movie-magic.pem" ec2-user@ec2-3-90-210-33.compute-1.amazonaws.com
Warning: identity file movie-magic.pem accessible: No such file or directory.
ec2-user@ec2-3-90-210-33.compute-1.amazonaws.com: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
PS C:\Users\deekshi> ssh -l "C:\Users\deekshi\Downloads\movie-magic.pem" ec2-user@ec2-3-90-210-33.compute-1.amazonaws.com
      #          Amazon Linux 2023
      #          \####
      #          \####
      #          \#/
      #          \#/
https://aws.amazon.com/linux/amazon-linux-2023
      #          \#/
      #          \#/
      #          \#/
      #          \#/
      #          \#/
[ec2-user@ip-172-31-25-120 ~]$ sudo yum install git -y
Amazon Linux 2023 Kernel Livepatch repository
Dependencies resolved.
=====
|e   Version           Repository      Size
+---+-----+-----+-----+
|   |     |       |       |
|git| x86_64          2.47.1-1.amzn2023.0.3 |amazonlinux | 52 k |Installing:
|   |     |       |       |
|git| x86_64          2.47.1-1.amzn2023.0.3 |amazonlinux | 4.5 M |
|git-core| x86_64         noarch        1:0.17029-5.amzn2023.0.2 |amazonlinux | 2.8 M |
|git-core-doc| x86_64        noarch        1:37-477.amzn2023.0.7 |amazonlinux | 41 k  |
|perl-Error| x86_64         noarch        2.37-477.amzn2023.0.3 |amazonlinux | 25 k  |
|perl-File-Find| x86_64        noarch        2.47.1-1.amzn2023.0.3 |amazonlinux | 40 k  |
|perl-Git| x86_64         noarch        2.38-9.amzn2023.0.2 |amazonlinux | 36 k  |
|perl-TermReadKey| x86_64        noarch        0.65-477.amzn2023.0.7 |amazonlinux | 15 k  |
=====
Transaction Summary
=====
Install 8 Packages

```



```
ec2-user@ip-172-31-25-120:~/movie-magic/movie-magic
$ pip install flask
Collecting flask>=1.1.1-py3-none-any.whl (103 kB)
  Downloading flask-3.1.1-py3-none-any.whl (103 kB) 19.9 MB/s
Collecting itsdangerous>=2.2.0
  Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Collecting werkzeug>=3.1.0
  Downloading werkzeug-3.1.3-py3-none-any.whl (224 kB) 45.0 MB/s
Collecting blinker>=1.9.0
  Downloading blinker-1.9.0-py3-none-any.whl (8.5 kB)
Collecting markupsafe>=2.1.1
  Downloading MarkupSafe-3.0.2-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (20 kB)
Collecting click>8.1.2
  Downloading click-8.1.2-py3-none-any.whl (98 kB) 13.5 MB/s
Collecting importlib-metadata>=3.6.0
  Downloading importlib_metadata-3.7.0-py3-none-any.whl (27 kB)
Collecting jinja2>=3.1.2
  Downloading jinja2-3.1.6-py3-none-any.whl (134 kB) 74.9 MB/s
Collecting zipp>3.20
  Downloading zipp-3.23.0-py3-none-any.whl (10 kB)
Installing collected packages: zipp, markupsafe, werkzeug, jinja2, itsdangerous, importlib-metadata, click, blinker, flask
Successfully installed blinker-1.9.0 click-8.1.2 flask-3.1.1 importlib-metadata-3.7.0 itsdangerous-2.2.0 jinja2-3.1.6 markupsafe-3.0.2 werkzeug-3.1.3 zipp-3.23.0
[ec2-user@ip-172-31-25-120 movie-magic]$ pip install boto3
Defaulting to user installation because normal site-packages is not writable
Collecting boto3
  Downloading boto3-1.39.3-py3-none-any.whl (139 kB)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3.9/site-packages (from boto3) (0.10.0)
Collecting s3transfer<0.14.0,>=0.13.0
  Downloading s3transfer-0.13.0-py3-none-any.whl (85 kB) 6.3 MB/s
Collecting botocore<1.40.0,>=1.39.3
  Downloading botocore-1.39.3-py3-none-any.whl (13.8 MB) 51.3 MB/s
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.3->boto3) (1.25.10)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.3->boto3) (2.8.1)
Requirement already satisfied: six<1.5 in /usr/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.40.0,>=1.39.3->boto3) (1.15.0)
```

Windows taskbar showing search bar, pinned icons (File Explorer, Task View, Start), and system status (30°C, ENG, 05-07-2025).

```
ec2-user@ip-172-31-25-120:~/movie-magic/movie-magic
$ python3 app.py
13.8 MB 51.3 MB/s
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.3->boto3) (1.25.10)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.3->boto3) (2.8.1)
Requirement already satisfied: six<1.5 in /usr/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.40.0,>=1.39.3->boto3) (1.15.0)
Installing collected packages: botocore, s3transfer, boto3
Successfully installed boto3-1.39.3 botocore-1.39.3 s3transfer-0.13.0
[ec2-user@ip-172-31-25-120 movie-magic]$ python3 app.py
python3: can't open file '/home/ec2-user/movie-magic/app.py': [Errno 2] No such file or directory
[ec2-user@ip-172-31-25-120 movie-magic]$ python3 app.py
python3: can't open file '/home/ec2-user/movie-magic/app.py': [Errno 2] No such file or directory
[ec2-user@ip-172-31-25-120 movie-magic]$ python3 app.py
python3: can't open file '/home/ec2-user/movie-magic/app.py': [Errno 2] No such file or directory
[ec2-user@ip-172-31-25-120 movie-magic]$ python3 app.py
python3: can't open file '/home/ec2-user/movie-magic/app.py': [Errno 2] No such file or directory
[ec2-user@ip-172-31-25-120 movie-magic]$ py app.py
:bash: py: command not found
[ec2-user@ip-172-31-25-120 movie-magic]$ python app.py
python3: can't open file '/home/ec2-user/movie-magic/app.py': [Errno 2] No such file or directory
[ec2-user@ip-172-31-25-120 movie-magic]$ python app.py
:bash: python: command not found
[ec2-user@ip-172-31-25-120 movie-magic]$ python3 app.py
python3: can't open file '/home/ec2-user/movie-magic/app.py': [Errno 2] No such file or directory
[ec2-user@ip-172-31-25-120 movie-magic]$ cd movie-magic
[ec2-user@ip-172-31-25-120 movie-magic]$ python3 app.py
Mock MovieMagic running at http://127.0.0.1:5000
 * 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 http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
Mock MovieMagic running at http://127.0.0.1:5000
 * Debugger is active!
 * Debugger PIN: 714-832-264
```

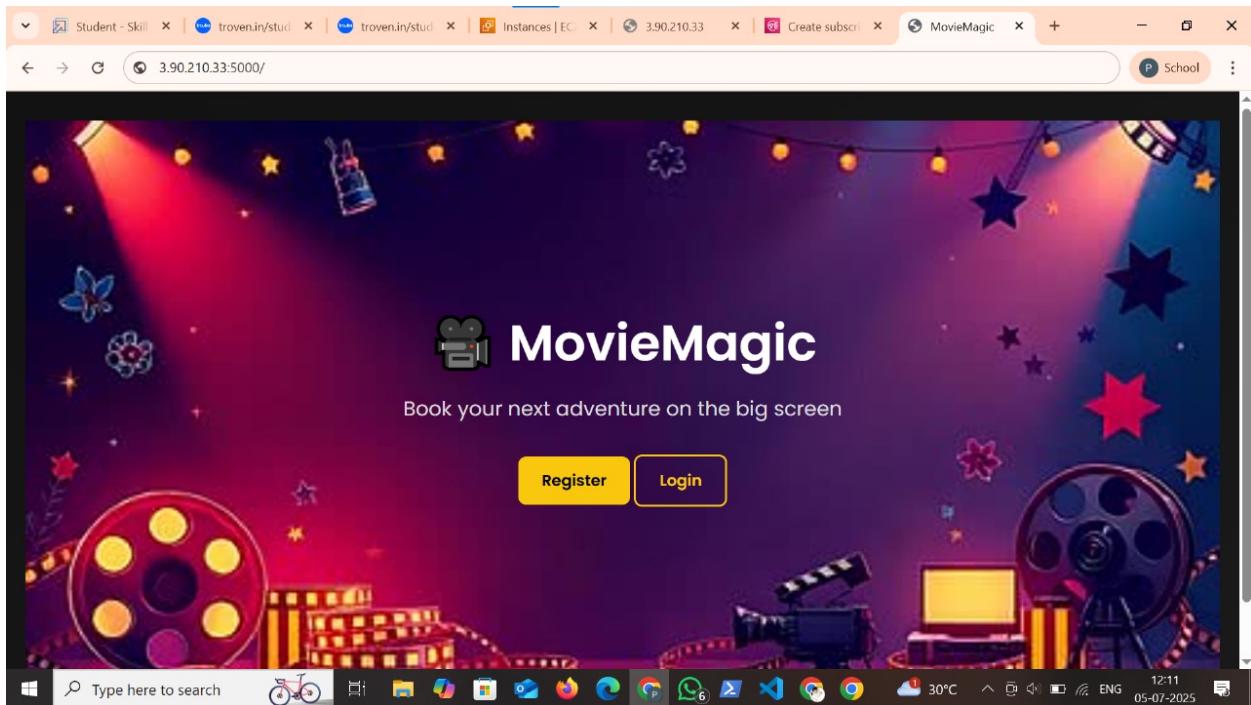
Windows taskbar showing search bar, pinned icons (File Explorer, Task View, Start), and system status (30°C, ENG, 05-07-2025).

- Access the website through: your-ec2-public-ip
- Public IPs : <http://3.90.210.33:5000/>

## Milestone 8: Testing and Deployment

Functional testing to verify the Project :

**Index Page :**





## Registration Page :

website design bg images with Register

127.0.0.1:5000/register

All Bookmarks

Create an Account 🎉

deekshithapotti2002@gmail.com

.....

Register

Already a member? Login

Type here to search

21:42 07-07-2025

## Login Page:

website design bg images with Login

127.0.0.1:5000/login

All Bookmarks

Welcome Back 🎉

deekshithapotti2002@gmail.com

.....

Login

Don't have an account? Register

Type here to search

21:42 07-07-2025



## Home Page :

The screenshot shows a web browser window displaying the 'MovieMagic | Home' page at 127.0.0.1:5000/home. The page has a dark theme with a red header bar. At the top left is a 'Logout' button. The top navigation bar includes 'LOREM' and the date 'Friday Feb 21'. The main content area features a large movie card for 'MOVIE NAME' from 20XX, directed by JOHN SMITH, with 14 episodes. Below the movie card are two buttons: 'STREAM NOW' and 'ALL EPISODES'. To the right of the movie card is a sidebar titled 'NETFLIX' featuring a play button icon and a grid of movie thumbnails. Below the sidebar is a 'POPULAR SHOW THIS WEEK' section. The bottom of the page has a footer with a search bar and various system icons.

The screenshot shows the same web browser window at 127.0.0.1:5000/home. The page now displays a large 'Welcome to MovieMagic' message and the subtext 'Discover, book, and experience the magic of cinema'. A prominent yellow 'Book Tickets' button is centered below the welcome message. The rest of the page is mostly blank, indicating a loading state or a different section of the application. The footer and system icons are visible at the bottom.

The screenshot shows the same web browser window at 127.0.0.1:5000/home. The page now displays a 'Now Showing' section with a blurred image of a movie scene. The rest of the page is mostly blank, indicating a loading state or a different section of the application. The footer and system icons are visible at the bottom.

deekshitaexcelr/movie-magic   MovieMagic | Home

127.0.0.1:5000/home

All Bookmarks

### Now Showing



**Majili**

The Grand Premiere

Drama [Book Now](#)



**Trivikram**

ALLU ARAVIND, RAJDEVA KRISHNA  
THAMARAI, PEEYONI, VAIBHAVI, RAKESH  
POO PHAAD, RAM LAXMAN, YUGANDHAR T

Engaging

Drama [Book Now](#)

Type here to search

website design bg images with

MovieMagic | Home

127.0.0.1:5000/home

All Bookmarks

### Coming Soon



**THE LAST WITCH HUNTER**



website design bg images with

MovieMagic | Home

127.0.0.1:5000/home

All Bookmarks

YouTube Maps Gmail Frontend Developer... Adobe Acrobat

THE LAST WITCH HUNTER  
LIVE FOREVER  
OCTOBER 23

Future Flick

Sci-Fi Coming Soon

28°C ENG 07-07-2025

## Booking\_form Page :

deekshitaexcelr/movie-magic

Book Tickets

127.0.0.1:5000/booking

All Bookmarks

YouTube Maps Gmail Frontend Developer... Adobe Acrobat

### Book for Example Movie

Date

11 - 07 - 2025

Time

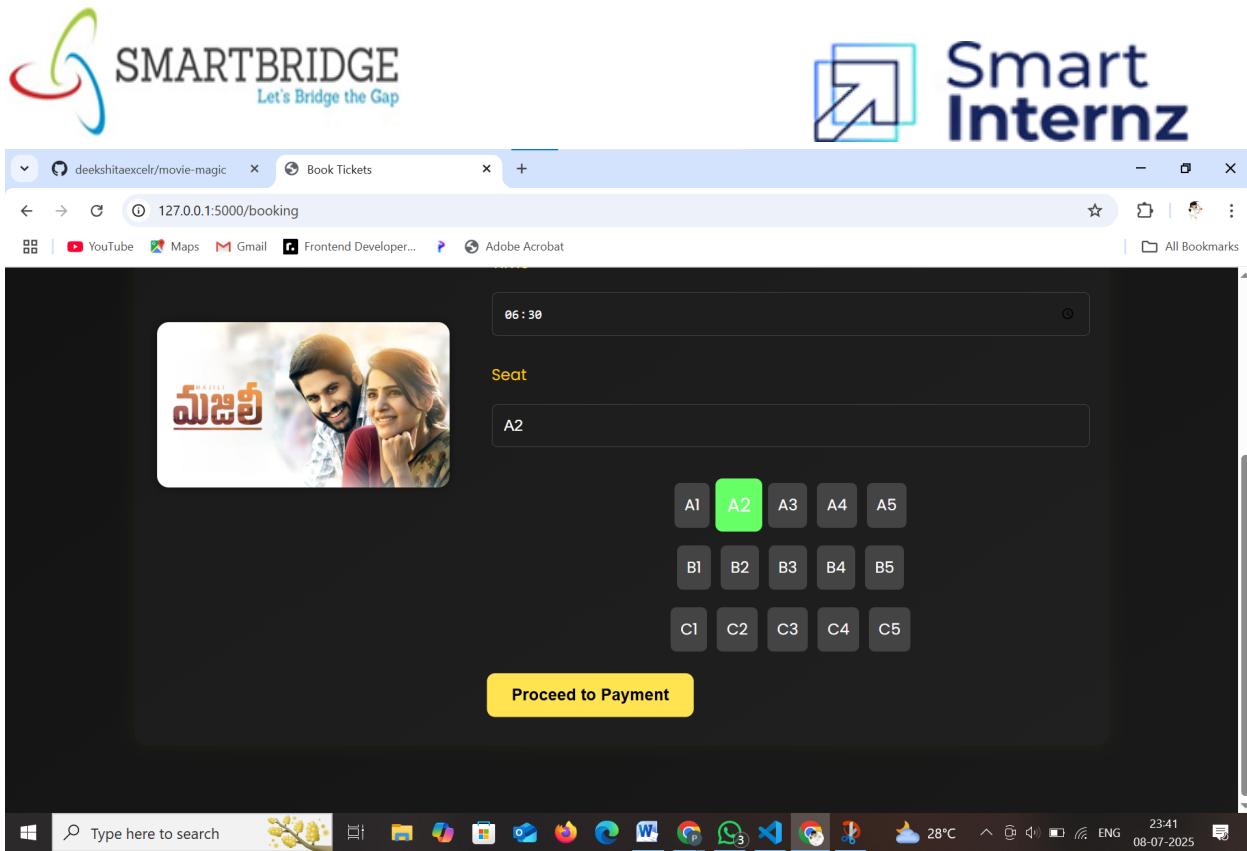
06 : 30

Seat

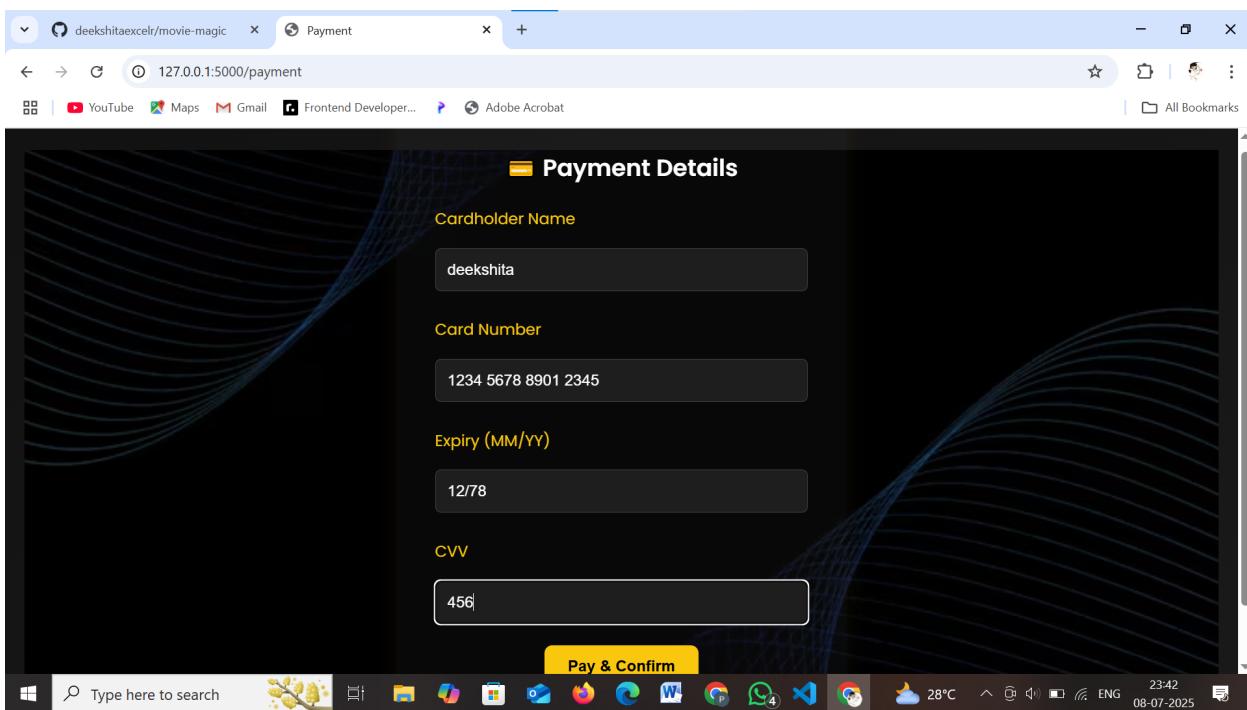
A2

A1 A2 A3 A4 A5  
B1 B2 B3 B4 B5

28°C ENG 08-07-2025

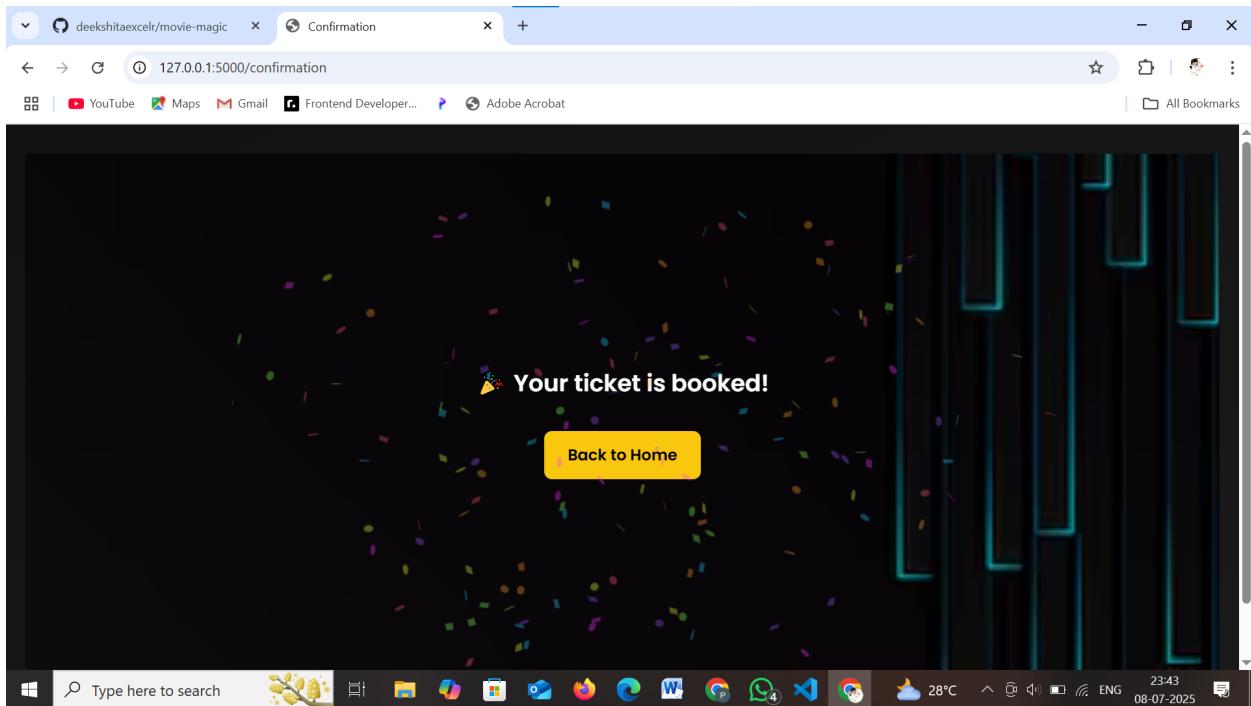


## **Payment Page :**



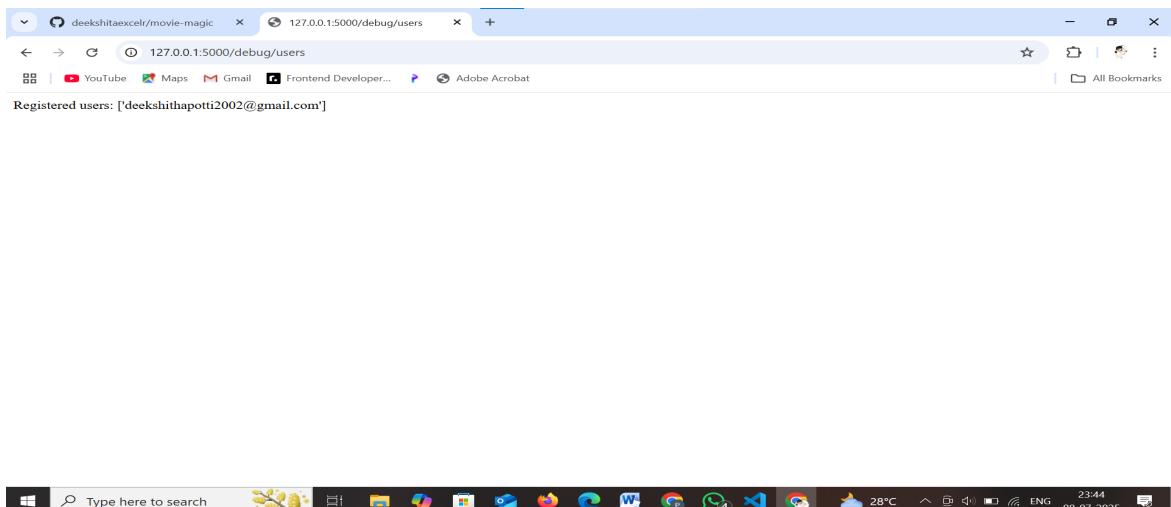


## confirmation Page :



**It stores the data in the backend :**

**a.users data :**





## b. booking data :

← → ⌂ ⓘ 127.0.0.1:5000/debug/bookings

YouTube Maps Gmail Frontend Developer... ↗ Adobe Acrobat

### All Bookings

- User: deekshithapotti2002@gmail.com, Movie: Example Movie, Seat: A2, Date: 2025-07-11, Time: 06:30, ID: 4c386d92

