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

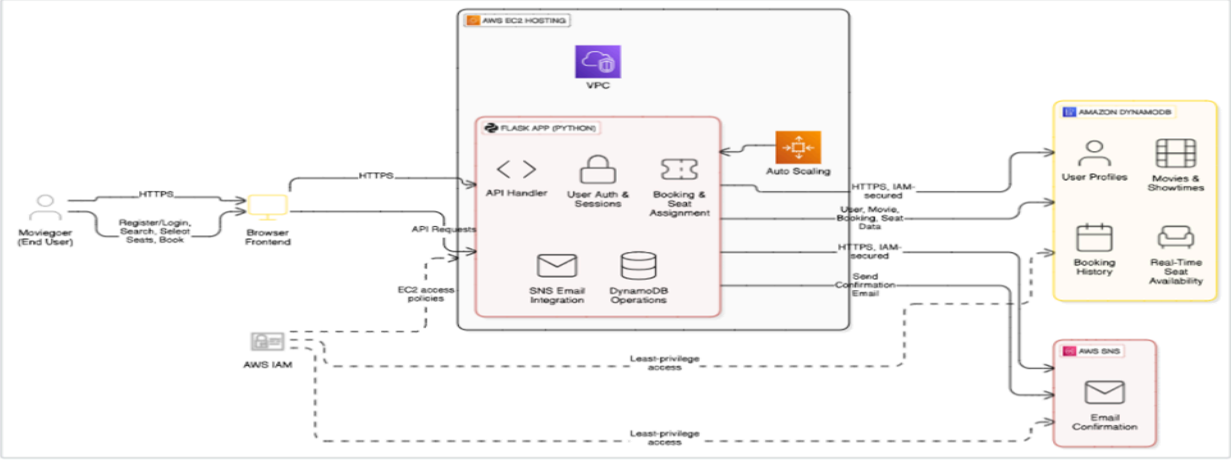
When a user completes a ticket booking, the Movie Magic System leverages AWS SNS to send instant email notifications to confirm the booking. For instance, once the booking is submitted, Flask processes the transaction, and SNS sends a customized email to the

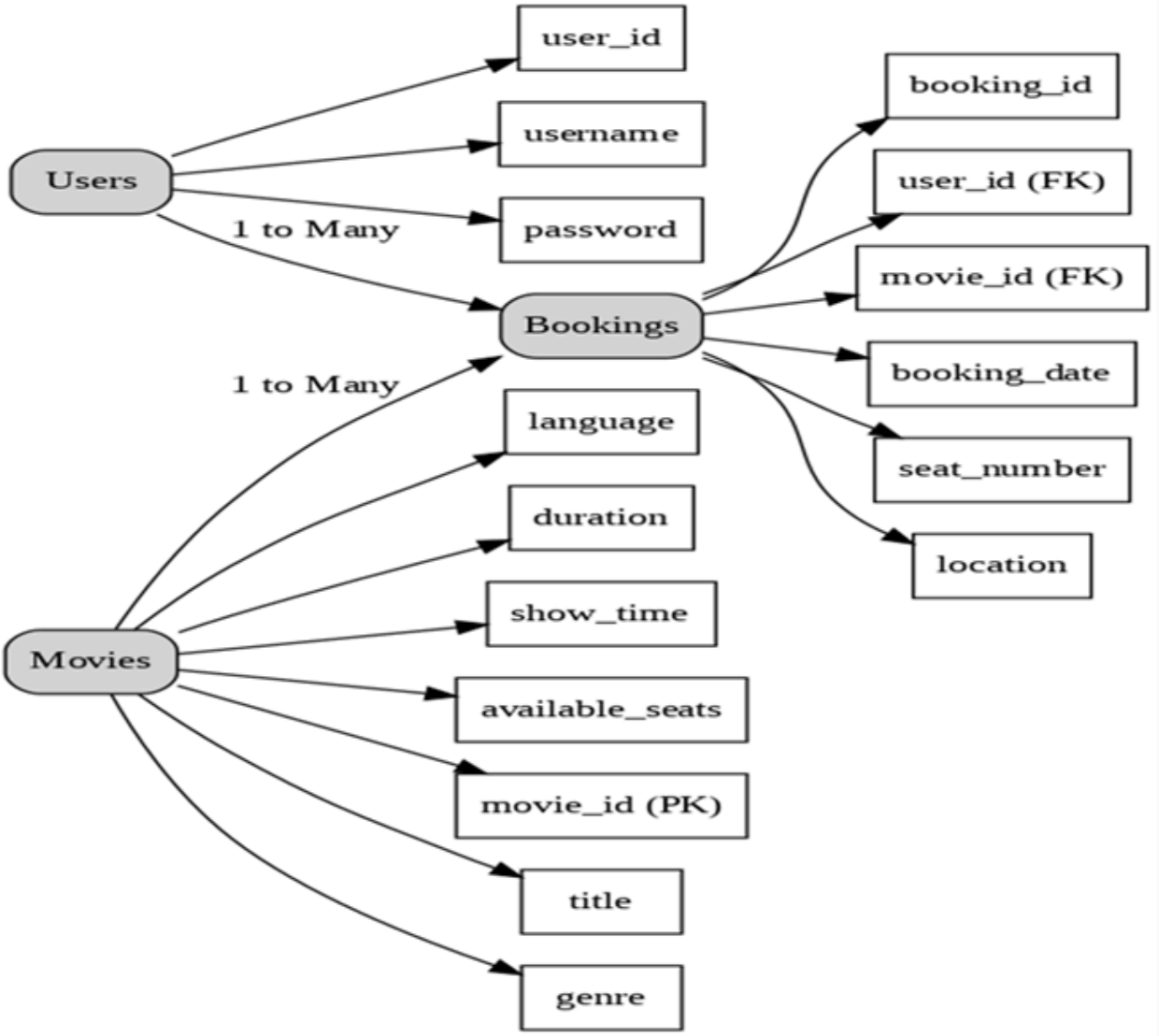
user with all ticket details, including movie name, date, time, and seat numbers. This real-time notification system enhances the customer experience and reduces uncertainty, while DynamoDB securely stores the booking records for both users and admins to manage and track.

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 :





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** :
   https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html

5. **Amazon SNS :**

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

6. **Git Documentation :**
   https://git-scm.com/doc

7. **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 accountif 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 requestnotifications.

Activity 3.2: Subscribe users and librarystaff to SNS email notifications.

4. Backend Development and Application Setup :

Activity 4.1:Develop the BackendUsing 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 instanceto host the Flask application.

Activity 6.2: Configure securitygroups 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 :**

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.

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 :**

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.

- **Routes for Web Pages**

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

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

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.

- **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.

- **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.

Confirmation page for movie ticket booking :

Application Entry point:

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.

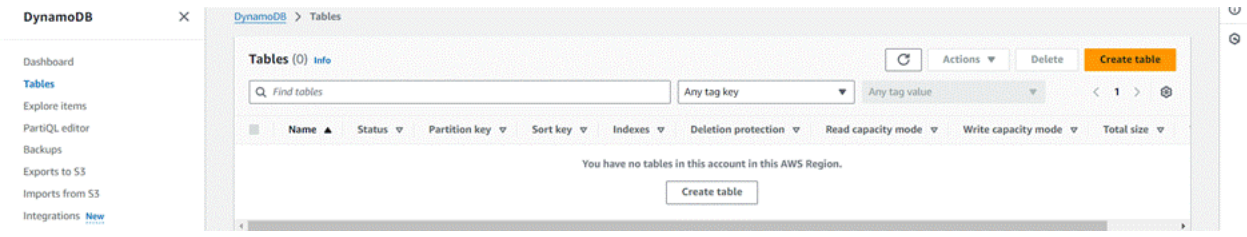
Milestone 2: AWS AccountSetup and Login :

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

- **Activity 1.2: Log in to the AWS Management Console :**
  - ➤ After setting up your account, log in to the AWS Management Console.

Milestone 3: DynamoDBDatabase Creation and Setup

- **Activity 3.1:Navigate to the DynamoDB**
  - In the AWS Console, navigate to DynamoDB and click on create tables.



- **Activity 3.2:Create a DynamoDBtable for storingregistration details and book requests.**


SMARTBRIDGE
Let's Bridge the Gap


Smart
Internz

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.

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

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

Configure the SNS topic and note down the Topic ARN.

- ## 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.

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

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 createa new IAM Role for EC2 to interact with DynamoDB and SNS.

- **Activity 5.2: AttachPolicies.**

Attach the following policiesto the role:

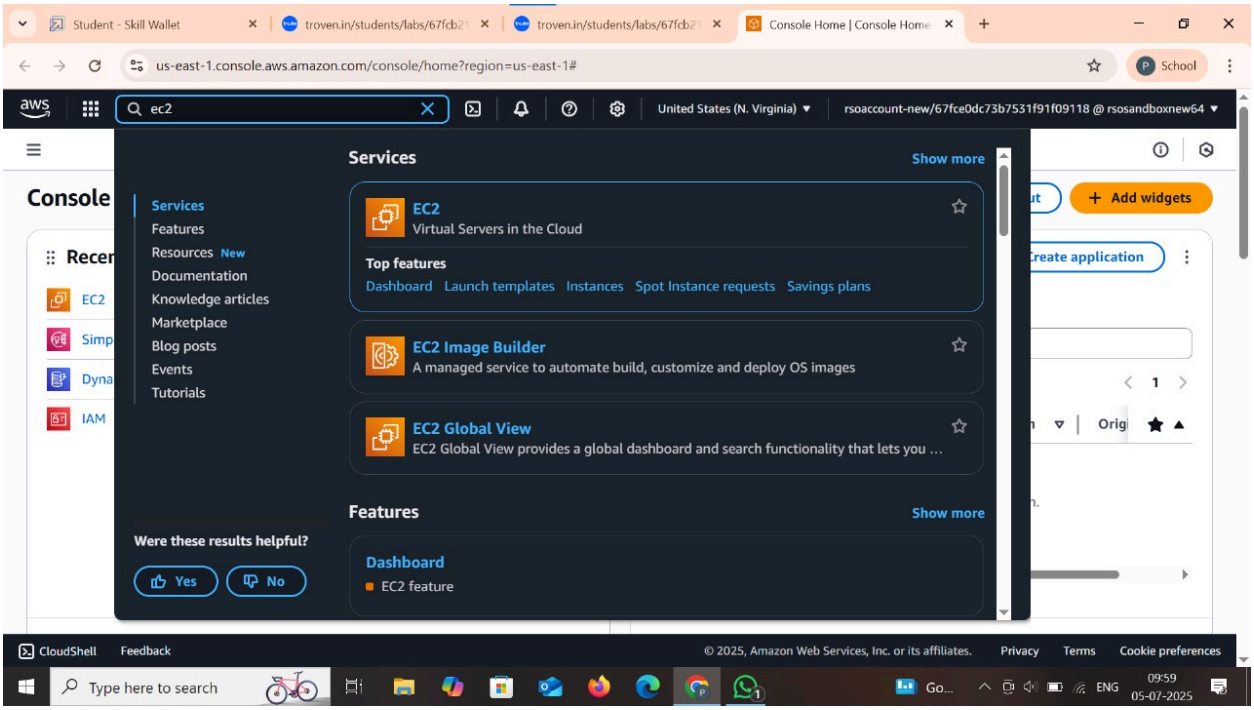
- ✓ AmazonDynamoDBFullAccess: Allows EC2 to perform read/write operations on DynamoDB.
- ✓ AmazonSNSFullAccess: Grants EC2 the ability to send notifications via SNS.

Milestone 6: EC2 InstanceSetup :

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

- **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

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

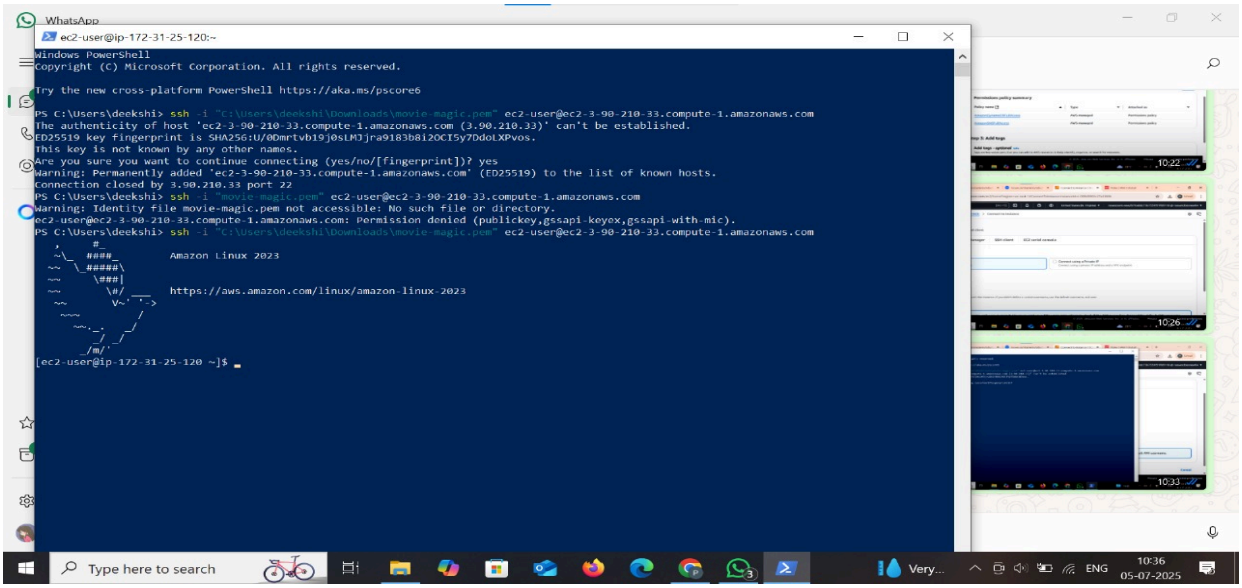
- Create and download the key pair for Server access.


SMARTBRIDGE
Let's Bridge the Gap


Smart
Internz

- **Configure security groups for HTTP and SSH for client:**

- 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".



Milestone 7: Deployment on EC2

Activity 7.1: Install Softwareon the EC2 Instance

Install Python3, Flask, and Git

- **On Amazon Linux 2:**

  sudo yum update -y

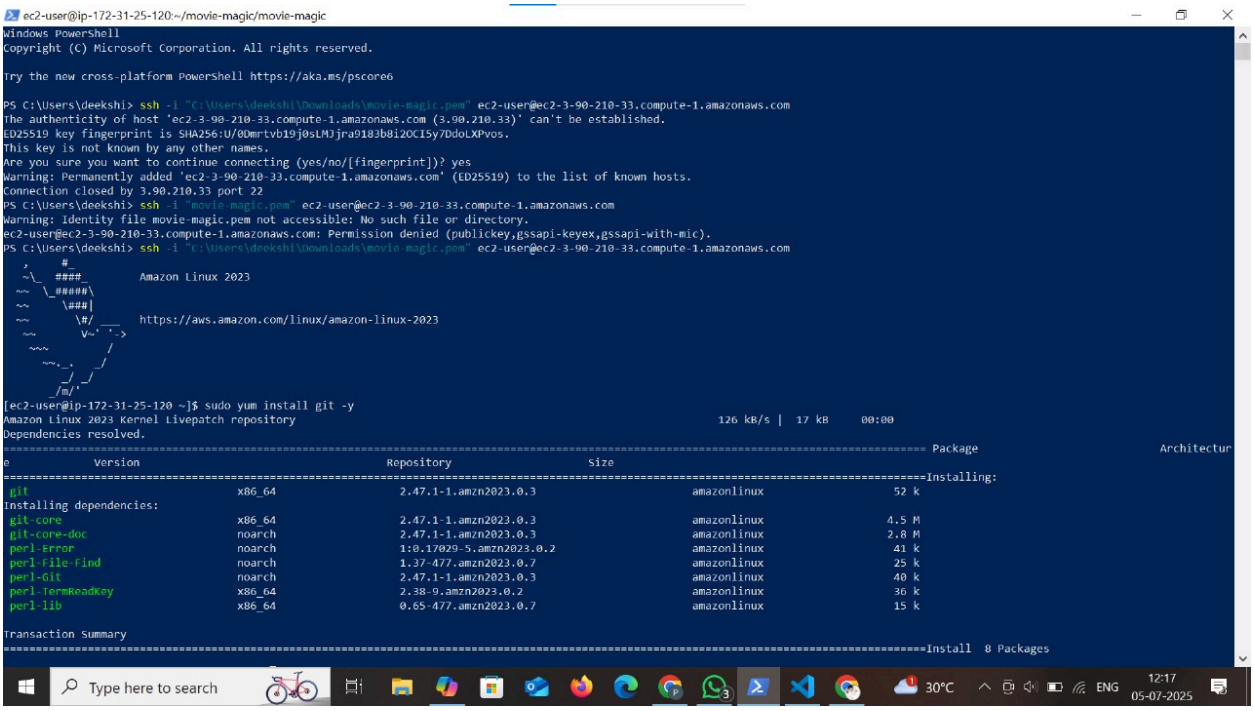
sudo yum install python3 git

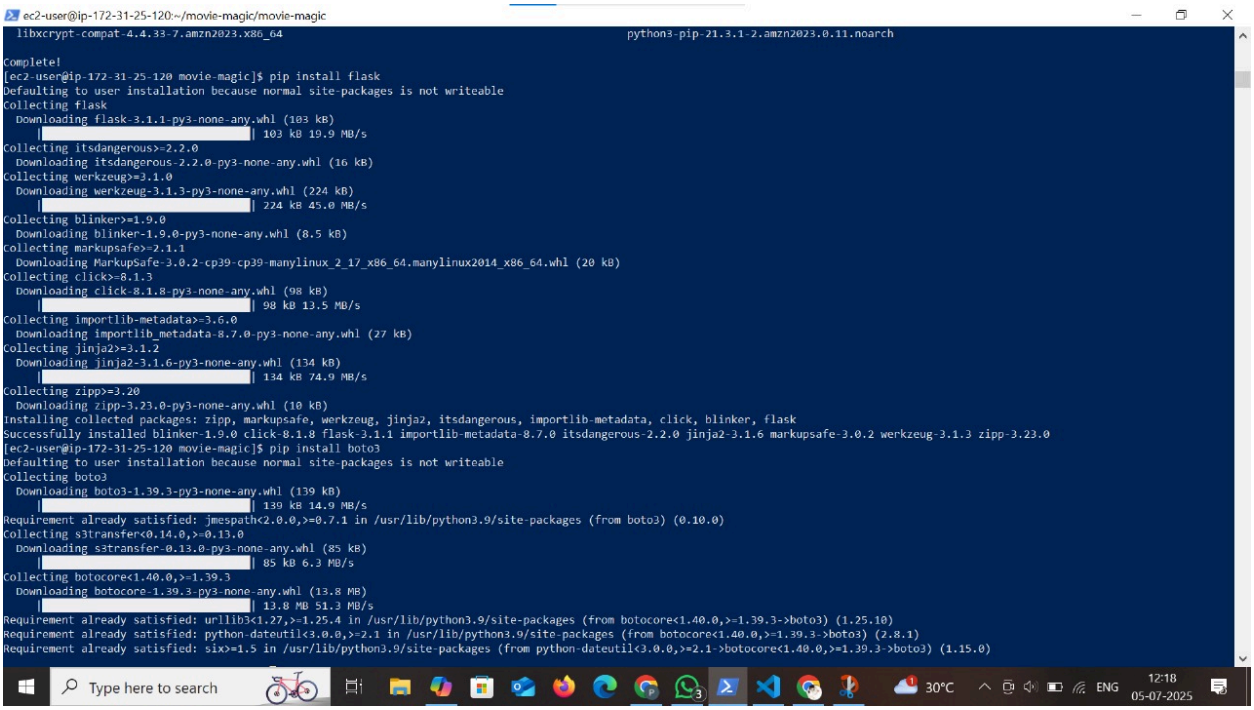
sudo pip3 install flask boto3

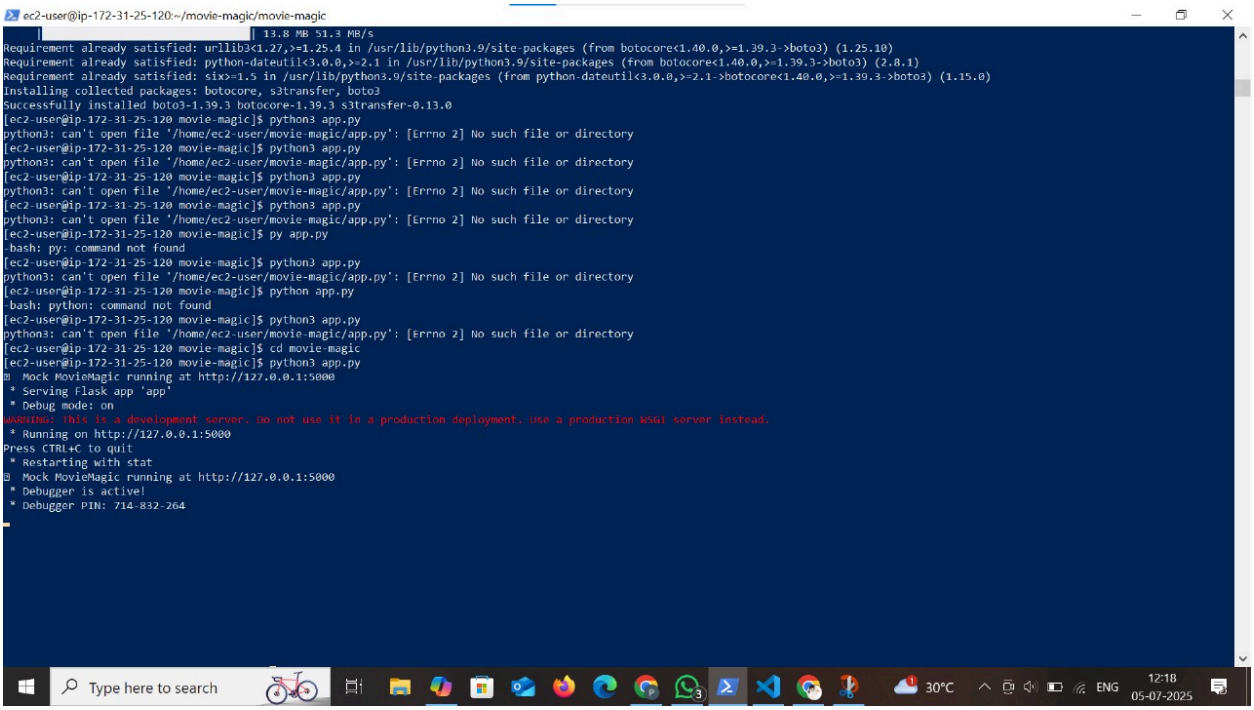
- **Verify Installations:**

flask --version

git --version





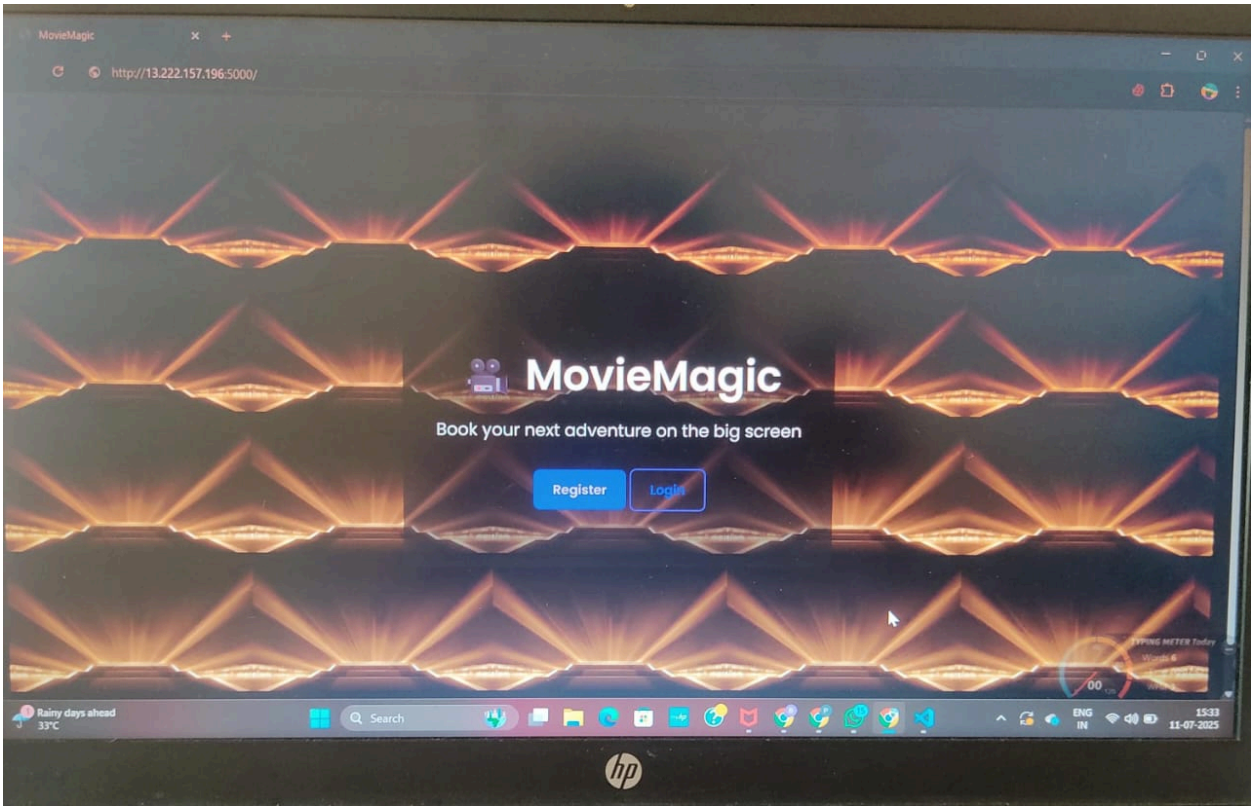


- Access the website through: your-ec2-public-ip
- Public IPs :http://127.0.0.1:5000

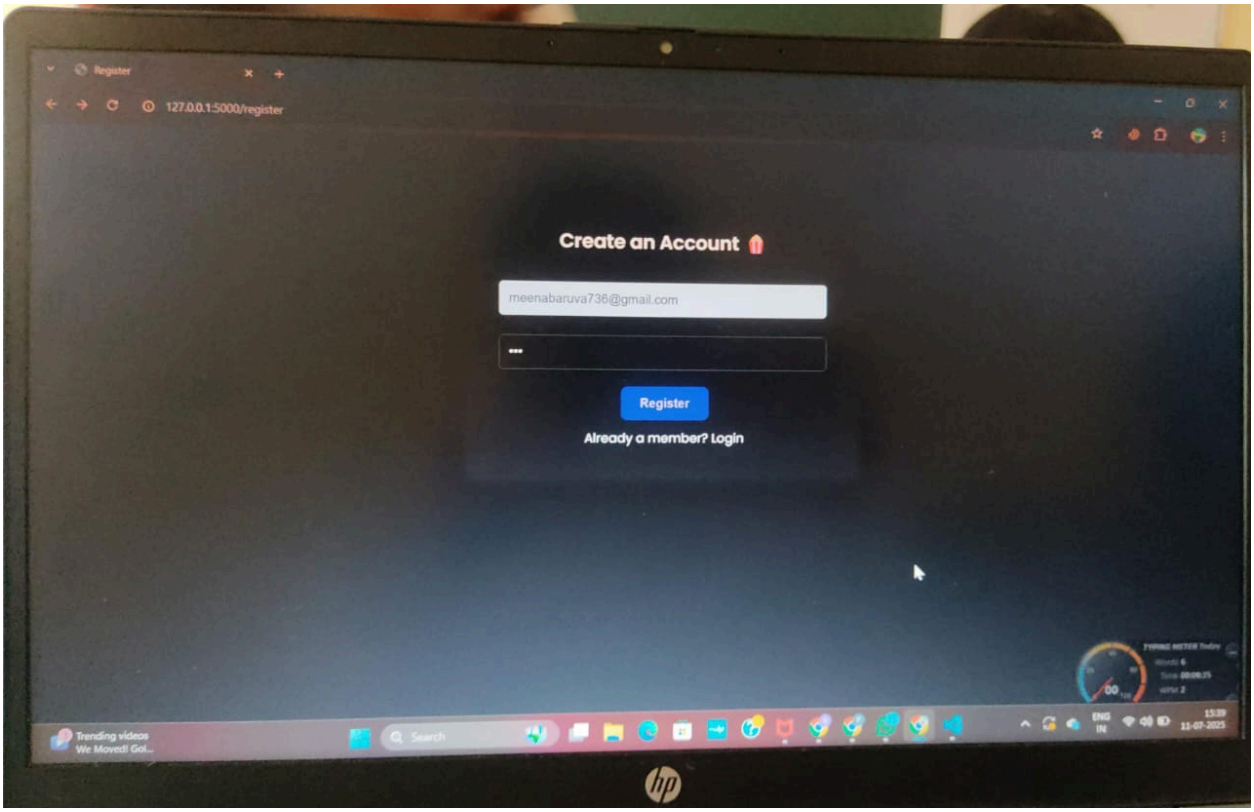
Milestone 8: Testing and Deployment

Functional testing to verify the Project :

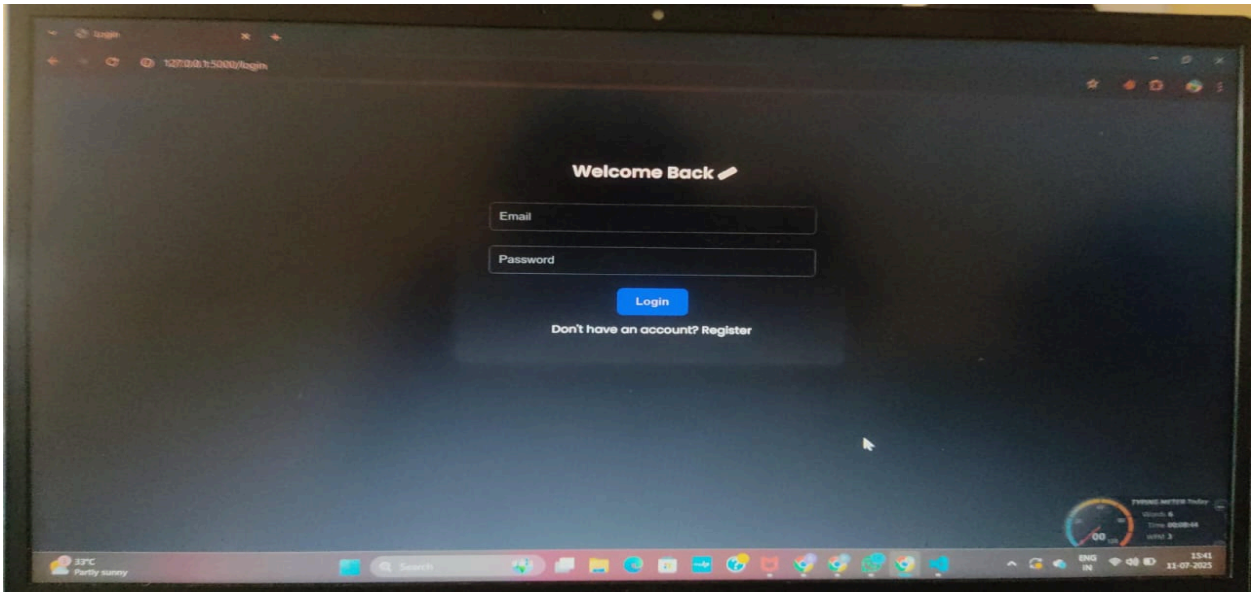
Index Page :



Registeration Page :



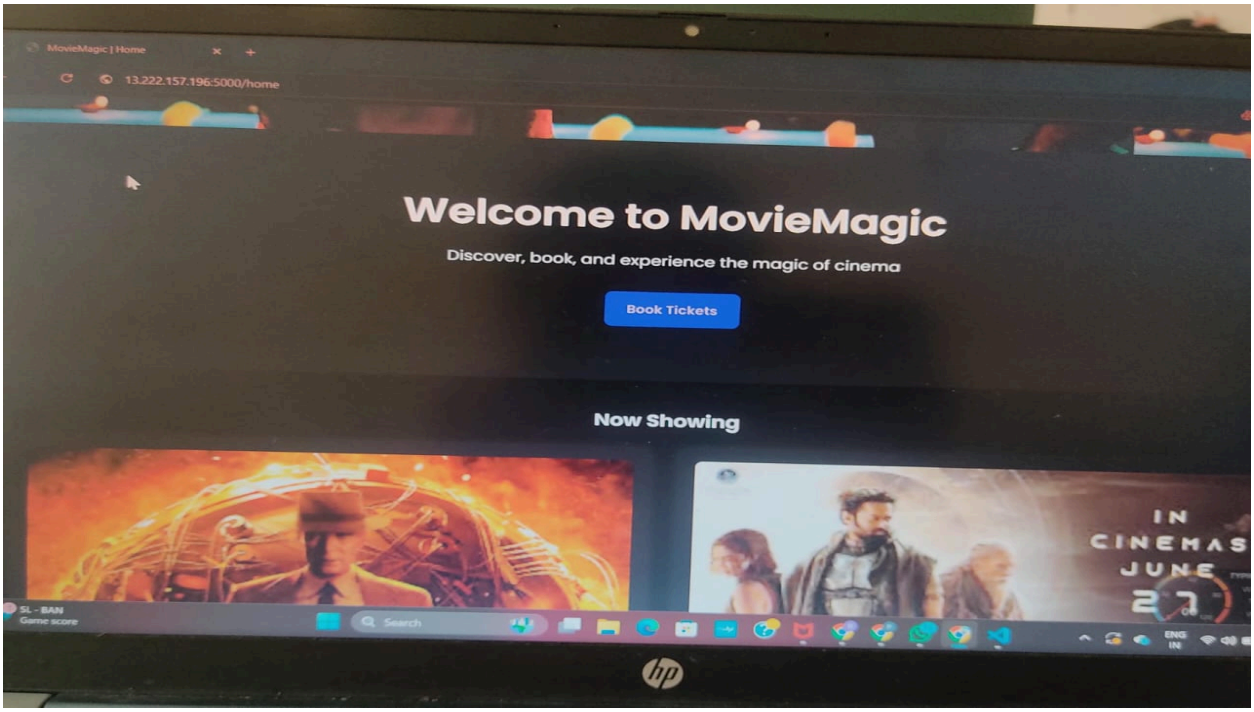
Login Page:

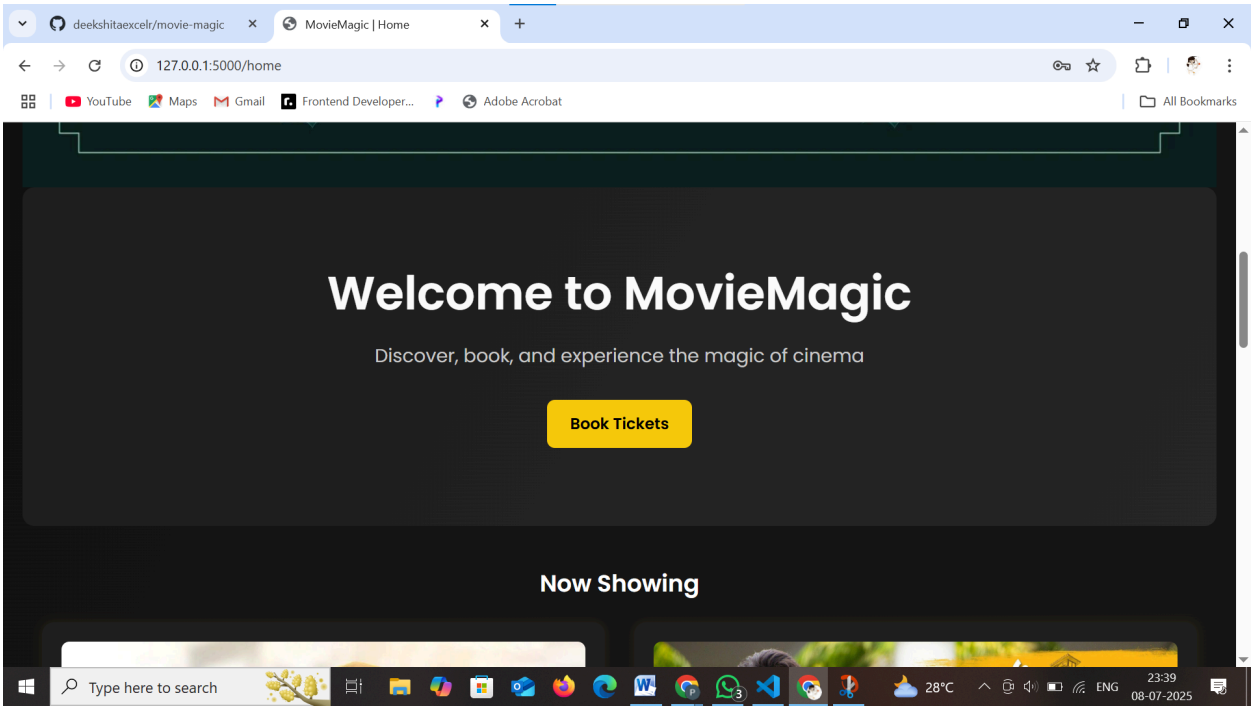


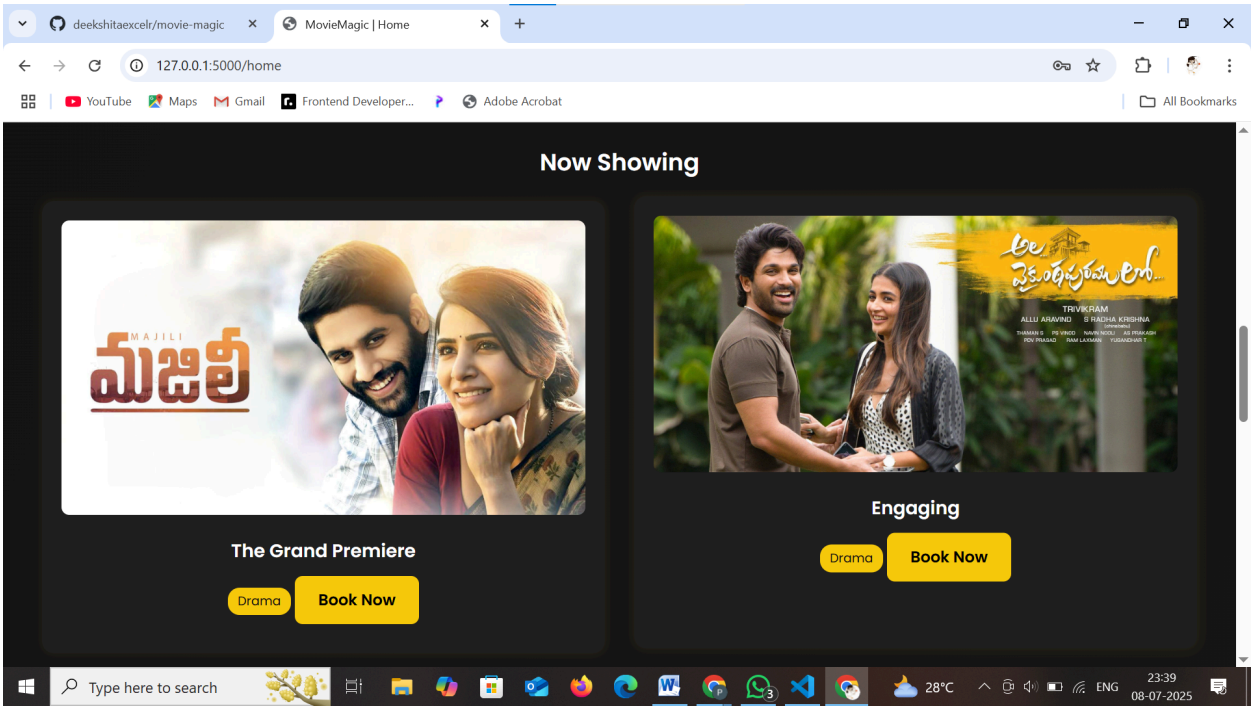
Home Page:

SMARTBRIDGE
Let's Bridge the Gap

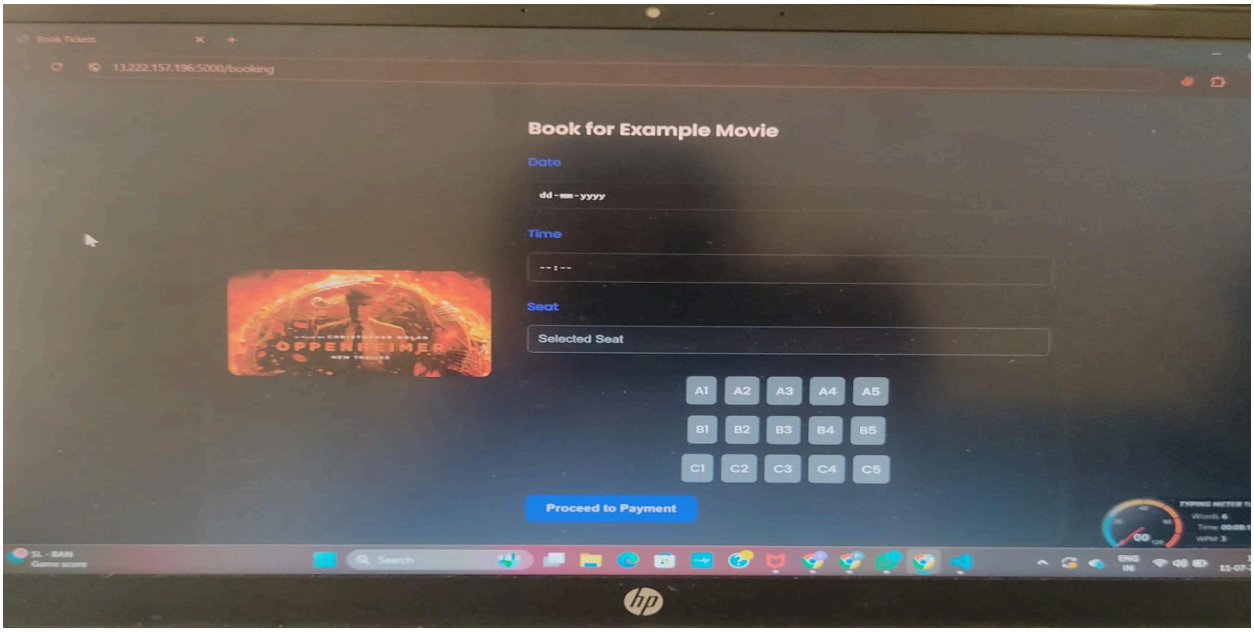
Smart Internz



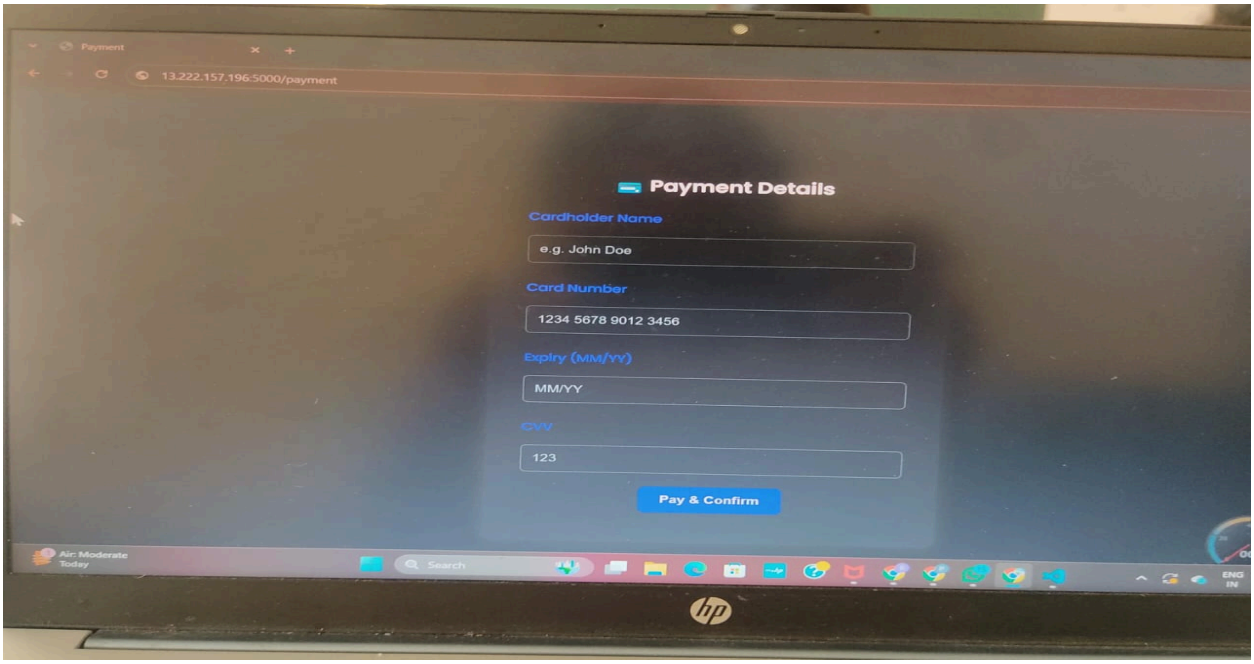




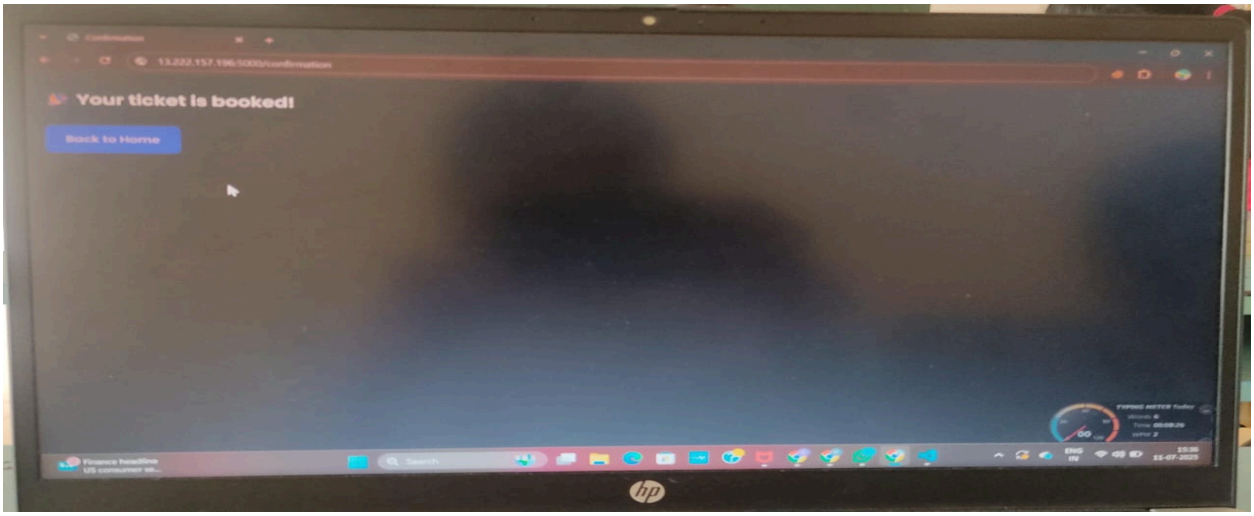
Booking_form Page :



Payment Page :

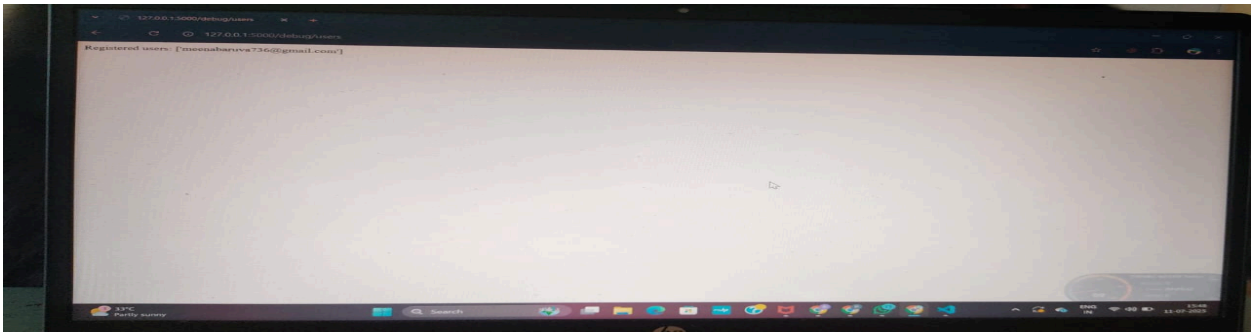


confirmation Page :



It stores the data in the backend :

a.users data :



b. booking data :