1. Title Page

Project Title: MedTrack – AWS Cloud Enabled Healthcare Management System

Category: AWS Cloud Foundation

Submitted By: [Patnala Lakshmi Jahnavi]

College/University: [Kallam HaranadhaReddy Institute of Technology]

Course: [CSE -data science]

Year: [2023-2026]

Date: [4july 2025]

1. Introduction:

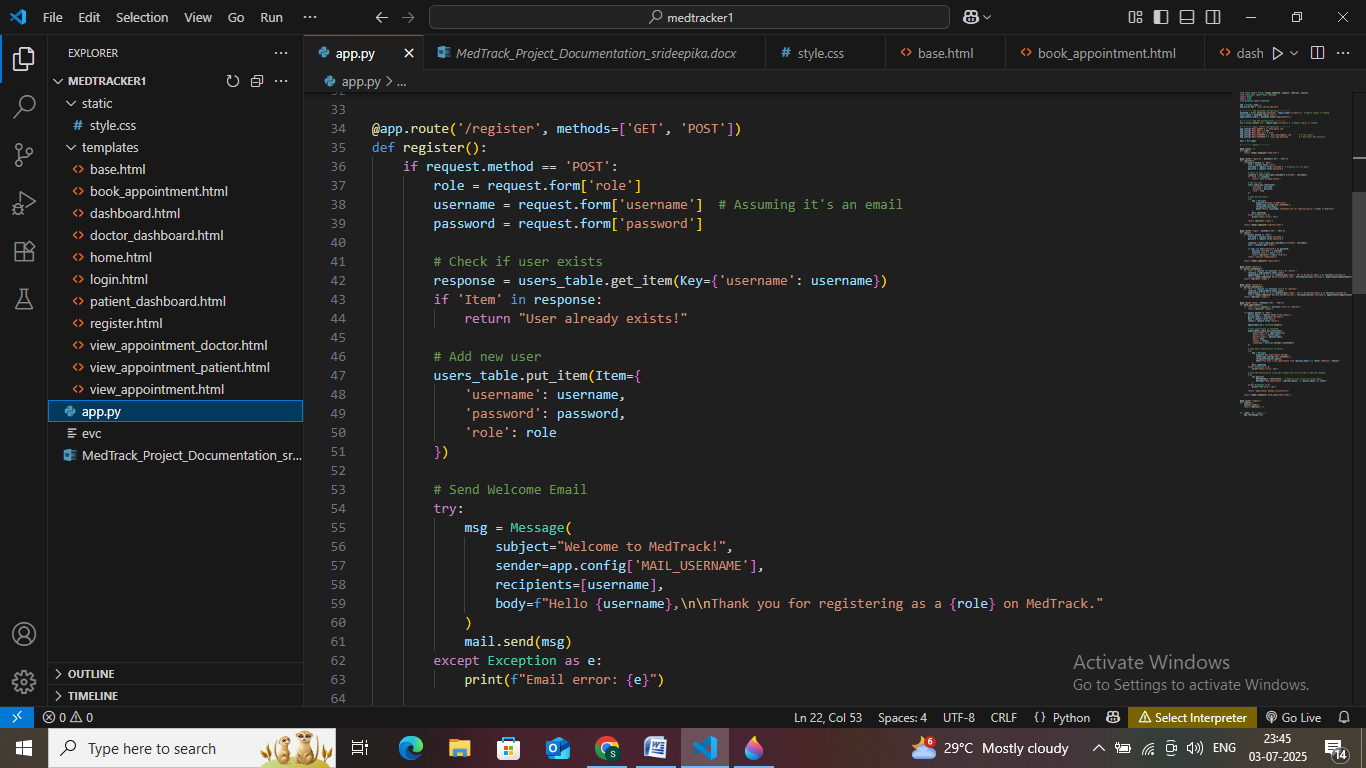
MedTrack is a cloud-based healthcare platform that simplifies patient-doctor interactions through secure, real-time appointment booking, diagnosis submission, and medical record access. Hosted on AWS EC2 and powered by DynamoDB and Flask, it ensures scalability and data integrity for both patients and doctors.

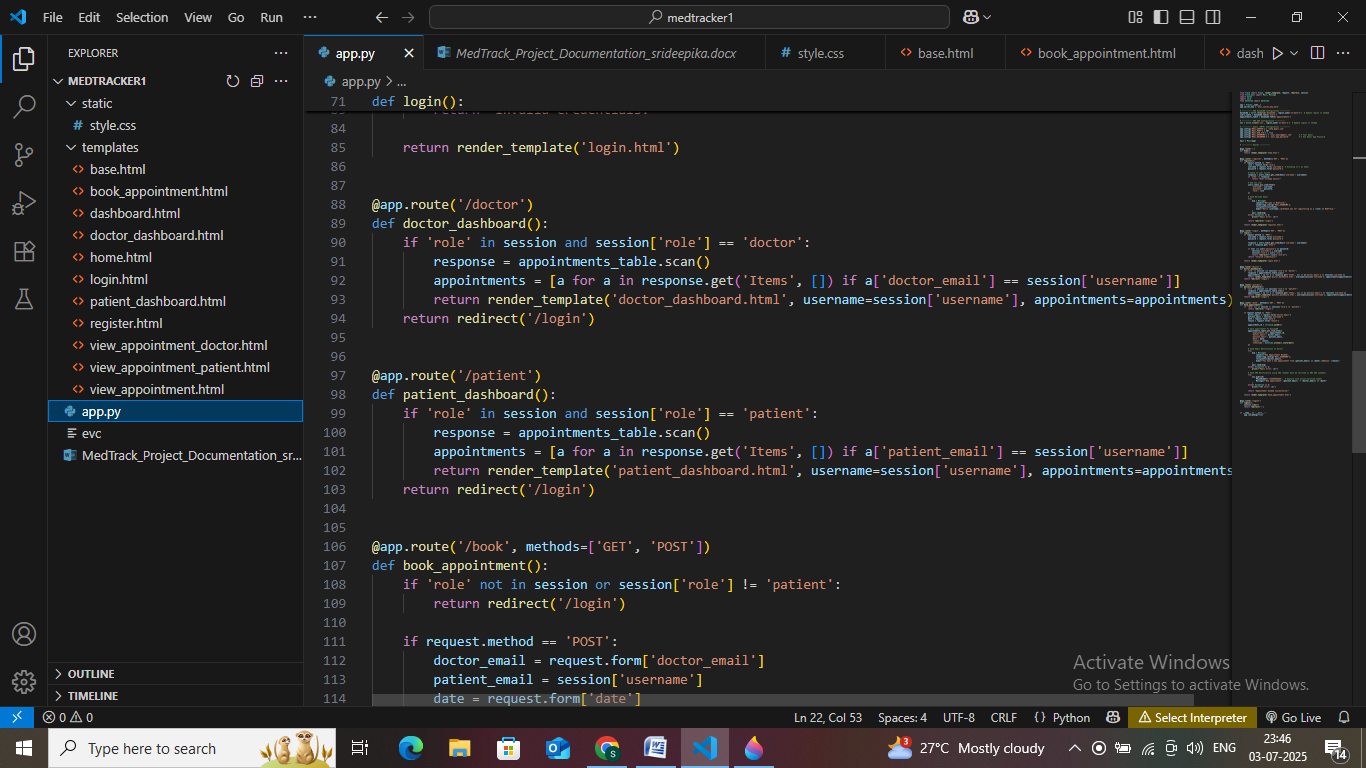
1. Objective:

The objective of MedTrack is to use AWS cloud services and Python Flask to build a secure and efficient online healthcare management system. It focuses on solving real-world issues like delayed communication, poor record-keeping, and access issues in traditional systems

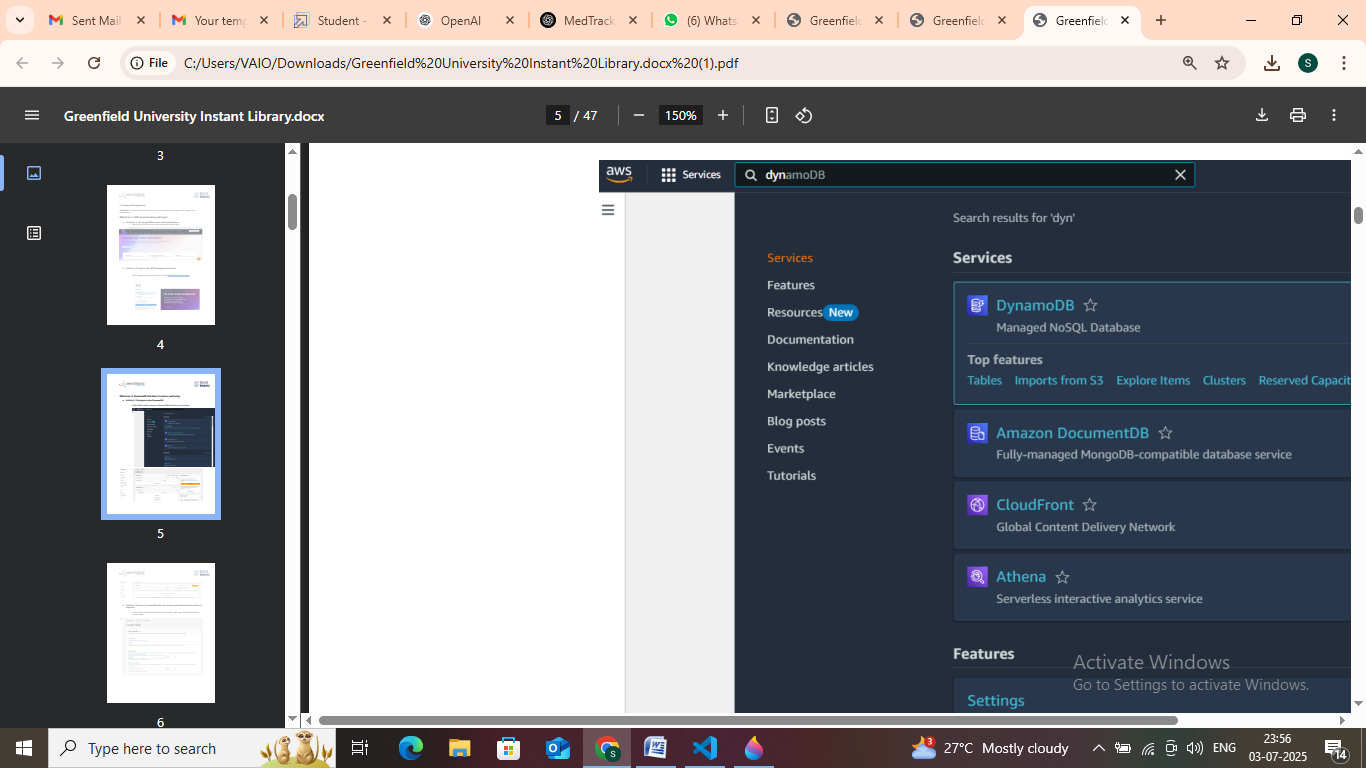
1. Technologies Used:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Technology** | **-Description** | | --- | --- |  |  |  | | --- | --- | | Python Flask | -Backend framework for server-side logic |  |  |  | | --- | --- | | AWS EC2 | -Cloud server to host the app | |  |  | | AWS DynamoDB | -NoSQL cloud database |  |  |  | | --- | --- | | Flask-Mail | -Sends email notifications |  |  |  | | --- | --- | | HTML/CSS | -Frontend design |  |  |  | | --- | --- | | 5.Modules and Features   1. User Module    Register/Login as Patient or Doctor   Email Notification on Register   1. Patient Dashboard    Book Appointment   View Appointment History |  | |  |  | |  |
| **c.Doctor Dashboard**   1. View All Patient Appointments 2. Access Patient Data   6.Working Scenarios  **Scenario 1: Efficient Appointment Booking System**  MedTrack simplifies the appointment process through a cloud-based interface where patients can easily book appointments with doctors. Here's how it works:   * **User Flow:**   + A registered patient logs into the system.   + Navigates to the appointment booking page.   + Selects a doctor, date, and time from the available slots.   + Submits the appointment request. * **Technical Implementation:**   + The backend, built with **Python Flask**, handles appointment logic and validation.   + Data is securely stored in **AWS DynamoDB**, ensuring each booking is stored instantly.   + Flask routes handle appointment creation and return confirmation feedback to the user.   + Hosted on **AWS EC2**, the system can scale to handle multiple booking requests simultaneously without performance issues. * **Benefits:**   + Reduces manual errors and delays.   + Offers 24/7 access for patients to book appointments.   + Can handle a large number of concurrent users during peak times.   **Scenario 2: Secure IAM Access Management**  Security and controlled access are critical in healthcare. MedTrack ensures that users only access what they’re authorized to see using **AWS Identity and Access Management (IAM)**:   * **Patient Access:**   + Can register and log in using their credentials.   + Allowed to book appointments and view their own medical data.   + Cannot access other users' data. * **Doctor Access:**   + Doctors log in with assigned credentials.   + Can view appointments made with them.   + Access and manage patient records that are relevant to their cases. * **IAM Role Setup (Optional Advanced Feature):**   + IAM policies can be defined on AWS to create roles like DoctorRole and PatientRole.   + Each role gets permissions for specific AWS services (like read-only for certain DynamoDB tables).   + Ensures sensitive operations (like data read/write) are restricted by role. * **Benefits:**   + Enhances data security and privacy.   + Meets compliance needs for access control.   + Ensures only authorized roles perform specific actions.7   **✅ Scenario 3: Medical Record Retrieval with DynamoDB**  Doctors and patients need fast access to medical history and appointments. MedTrack uses **AWS DynamoDB**, a fast and flexible NoSQL database, for real-time access:   * **How It Works:**   + Patient and appointment data are stored in DynamoDB tables like Users and Appointments.   + When a doctor logs in, a Flask route queries DynamoDB using the doctor’s name/email.   + All appointments related to that doctor are fetched and displayed.   + Patients can also view their own appointment history retrieved through query filters. * **Technical Advantage:**   + DynamoDB’s low-latency reads ensure fast response.   + Supports **querying by partition key**, which speeds up record retrieval.   + Works seamlessly with Flask and Boto3 SDK for Python. * **Benefits:**   + Real-time access to medical and appointment records.   + Reliable and always available due to DynamoDB’s managed infrastructure.   + Scalable for large-scale data without performance drops.   7.AWS Services Used  **Ec2-** Hosting the Flask application  **Dynamodb-**Storing user and appointment data  **IAM-**Managing secure access to AWS   | **8.folder structure**  **medtrack/**  **├── app.py**  **├── templates/**  **│ ├── home.html**  **│ ├── register.html**  **│ ├── login.html**  **│ ├── patient\_dashboard.html**  **│ └── doctor\_dashboard.html**  **├── static/**  **│ └── style.css**  9.Screenshots |  | | --- | --- | |  |  | |  |  | |  |  | |  |  | |  |
|  |  |
|  |  |
|  |  |

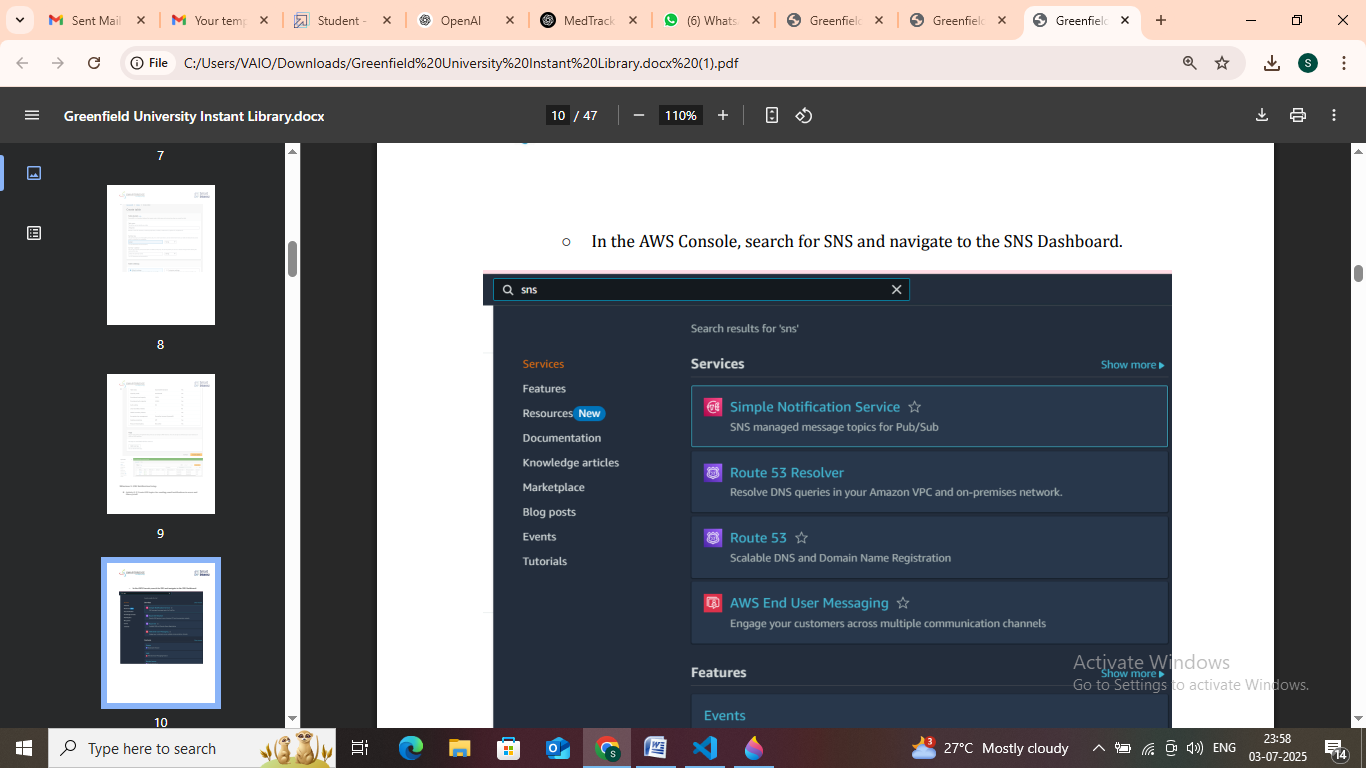
****

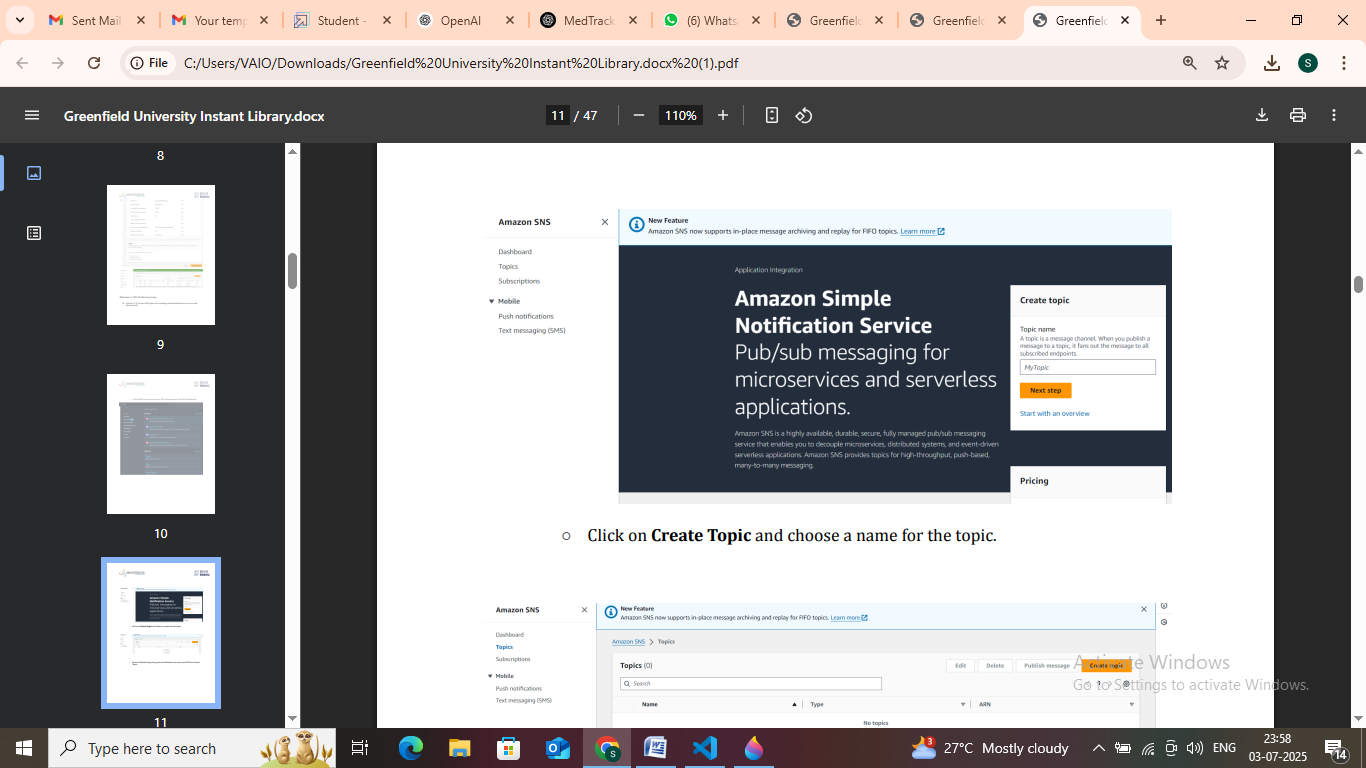


DynamoDB Database Creation and Setup

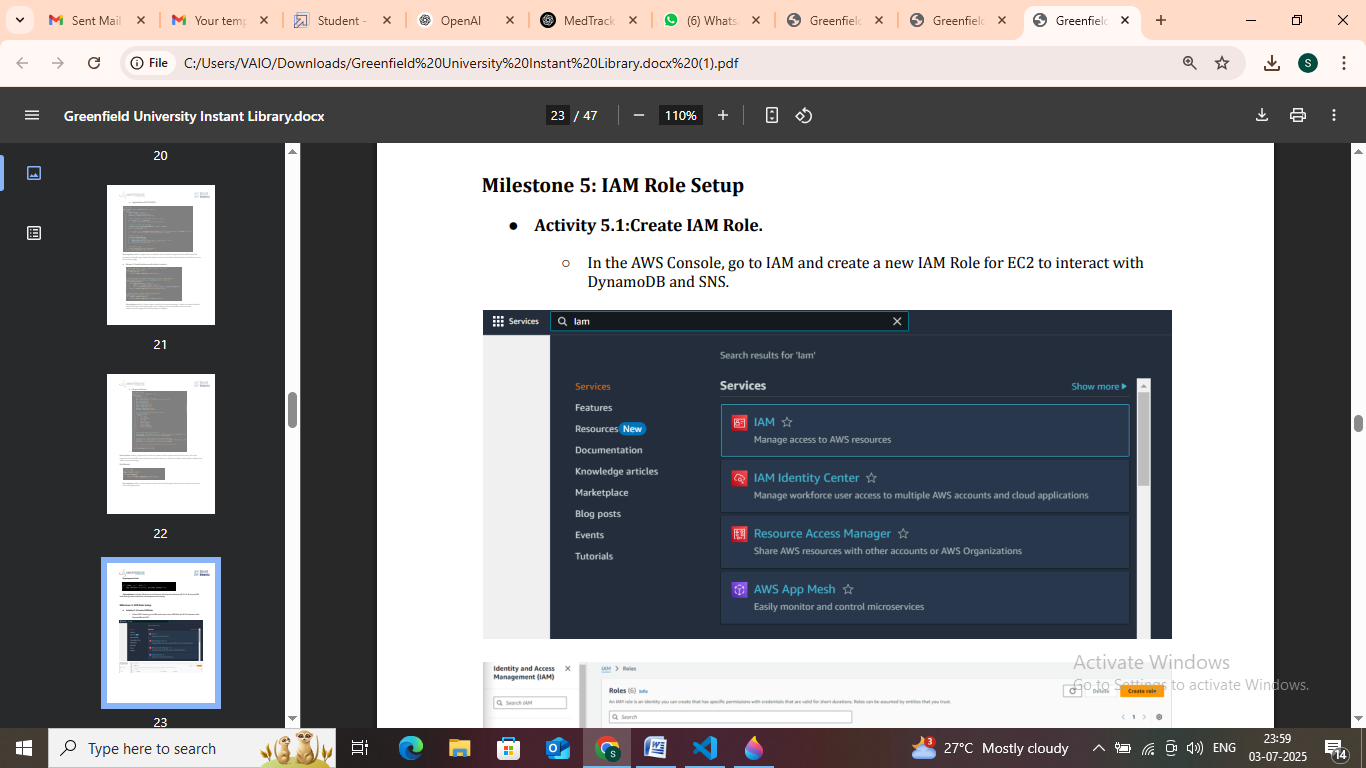


In the AWS Console, search for SNS and navigate to the SNS Dashboard.

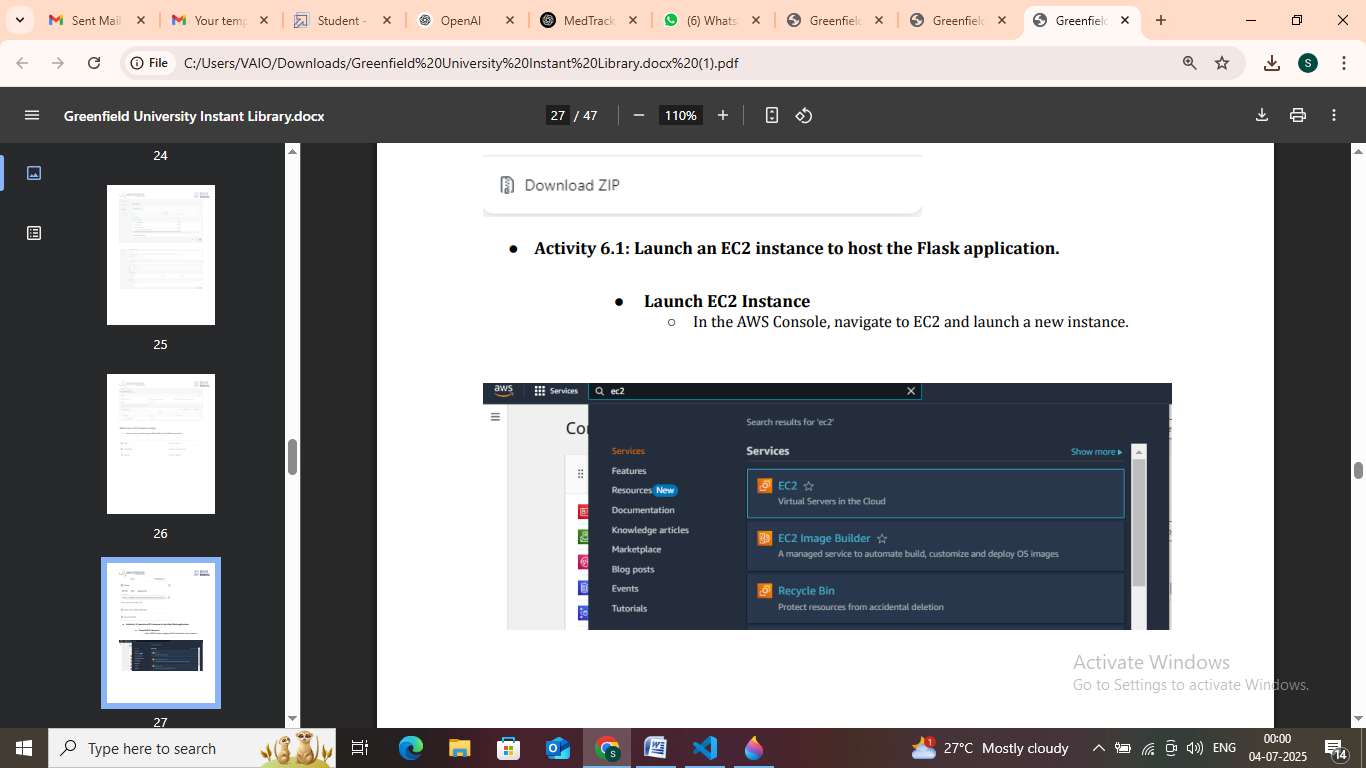




: IAM Role Setup



Launch an EC2 instance to host the Flask application



.10. Conclusion

MedTrack enhances healthcare services by providing cloud-powered solutions that allow easy and secure communication between patients and doctors. Its scalable architecture ensures it is ready for real-world deployment.

