**MedTrack: AWS Cloud-Enabled Healthcare Management System**

### **Project Description**

In today's fast-evolving healthcare landscape, efficient communication and coordination between doctors and patients are crucial. MedTrack is a cloud-based healthcare management system that streamlines patient-doctor interactions by providing a centralized platform for booking appointments, managing medical histories, and submitting diagnoses. The system leverages Flask for backend development, AWS EC2 for hosting, and DynamoDB for data management. It supports real-time notifications through AWS SNS and secure access control via AWS IAM, ensuring both accessibility and data integrity. MedTrack is designed to improve healthcare accessibility, efficiency, and real-time communication.

### **Scenarios**

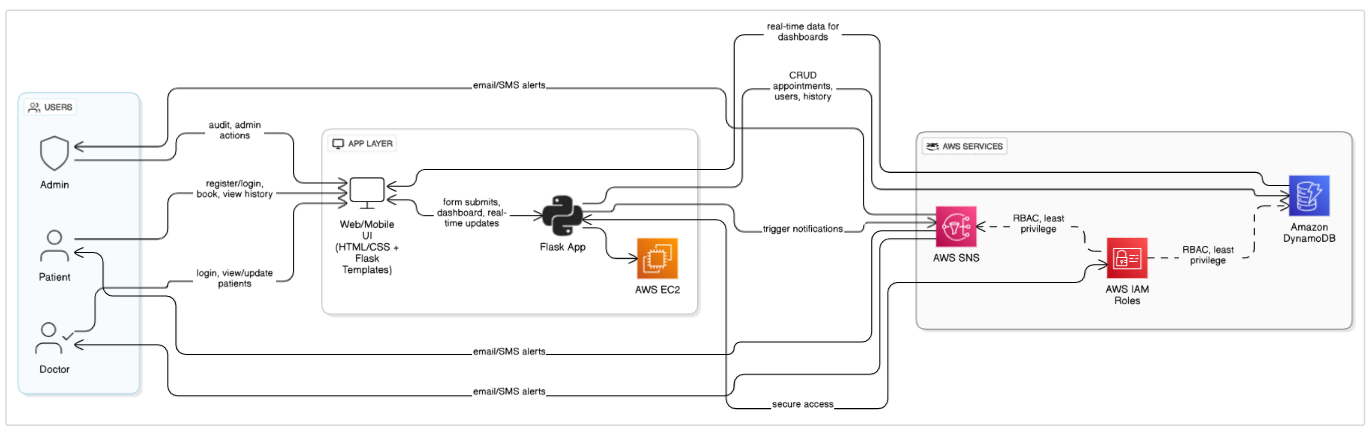
**Scenario 1: Efficient Appointment Booking System for Patients**AWS EC2 supports multiple concurrent users, allowing patients to log in and book appointments. Flask handles backend operations and integrates with DynamoDB to store real-time data efficiently.

**Scenario 2: Secure User Management with IAM**MedTrack uses IAM roles for secure, role-based access control. Patients and doctors receive permissions specific to their roles, ensuring secure access to sensitive data.

**Scenario 3: Easy Access to Medical History and Resources**Doctors can retrieve patient records and update diagnoses in real time. Flask and DynamoDB integration enables high-speed access and data updates, ensuring seamless operation during peak usage.

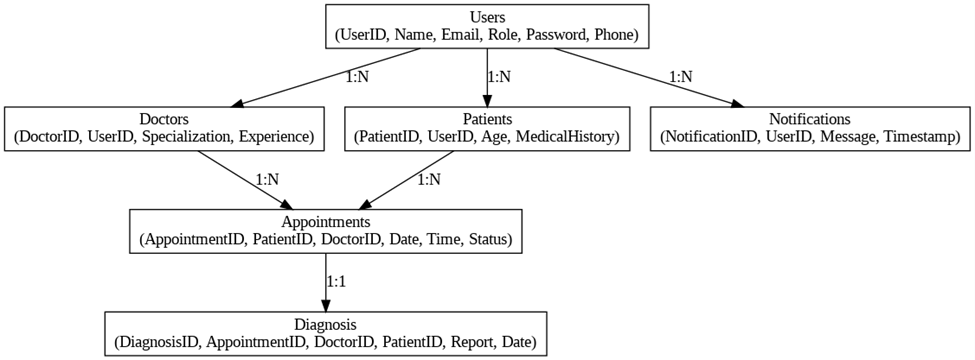
### 

### **AWS Architecture**



* **Flask (Python Framework):** Handles routing and backend logic.
* **Amazon EC2:** Hosts the Flask application.
* **Amazon DynamoDB:** Stores patient data, appointments, and records.
* **AWS SNS:** Sends real-time alerts and appointment confirmations.
* **AWS IAM:** Controls user access and permissions securely.

### **Entity Relationship (ER) Diagram**



The ER diagram illustrates entities such as Users (patients/doctors), Appointments, and their relationships. Key attributes include user ID, name, email, appointment ID, doctor ID, date, and status. This structure supports efficient query processing and normalization.

### **Pre-requisites**

* AWS Account Setup:<https://docs.aws.amazon.com/accounts/latest/reference/getting-started.html>
* IAM Overview:

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

* EC2 Tutorial:

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

* DynamoDB Introduction:<https://docs.aws.amazon.com/amazondynamodb/Introduction.html>
* SNS Documentation:

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

* Git Documentation:

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

* VS Code:

<https://code.visualstudio.com/download>

### **Project Workflow**

**Milestone 1: Web Application Development and Setup**

* Set up the Flask app with routing and templates.
* Use local Python lists/dictionaries for initial testing.
* Integrate AWS services (DynamoDB, SNS) using boto3.

**Milestone 2: AWS Account Setup**

* Access AWS via Troven Labs.
* Avoid personal AWS account usage to prevent billing issues.

**Milestone 3: DynamoDB Database Creation and Setup**

* Create a Users table (Primary key: Email).
* Create an Appointments table (Primary key: appointment\_id).

**Milestone 4: SNS Notification Setup**

* Create an SNS topic.
* Subscribe to users/admin via email.
* Confirm subscriptions and note Topic ARN.

**Milestone 5: IAM Role Setup**

* Create IAM Role (e.g., flask dynamodb sns).
* Attach policies: AmazonDynamoDBFullAccess, AmazonSNSFullAccess.

**Milestone 6: EC2 Instance Setup**

* Launch EC2 instance (Amazon Linux 2/Ubuntu, t2.micro).
* Assign IAM Role and key pair.
* Configure security groups for HTTP/SSH.

**Milestone 7: Deployment on EC2**

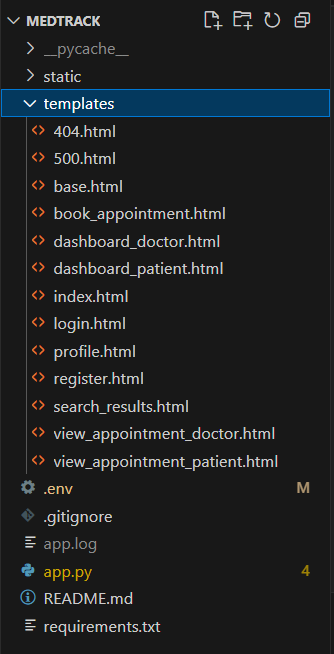
* Install Python3, Flask, Git.
* Clone the GitHub repo.
* Run Flask app: sudo flask run --host=0.0.0.0 --port=5000

**Milestone 8: Testing and Deployment**

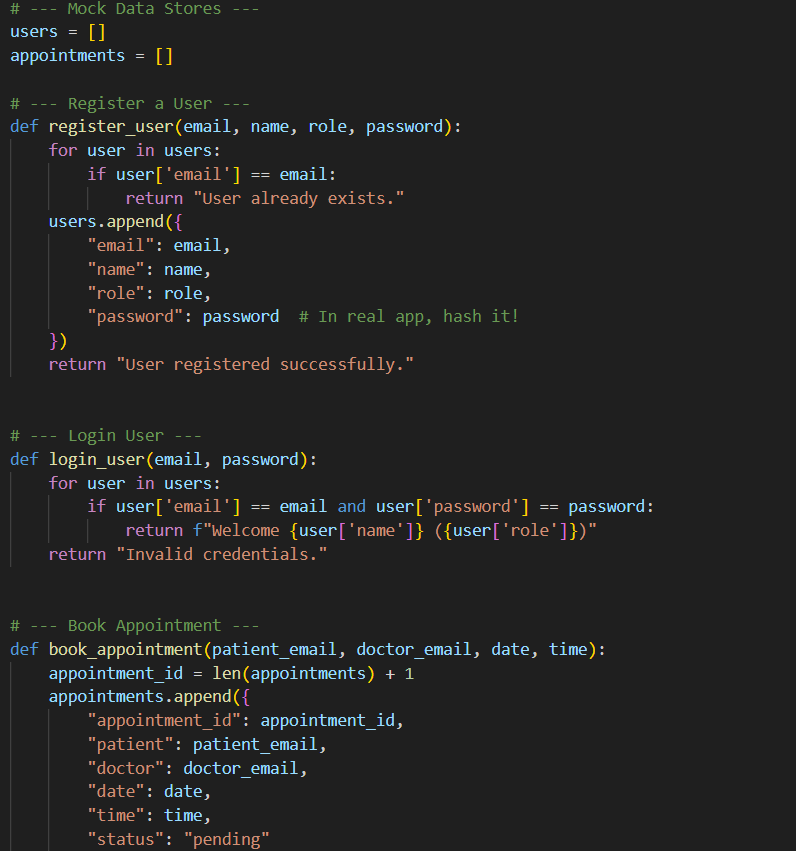
* Verify registration, login, appointment booking, and SNS notifications.

**Milestone 1: Web Application Development and Setup**

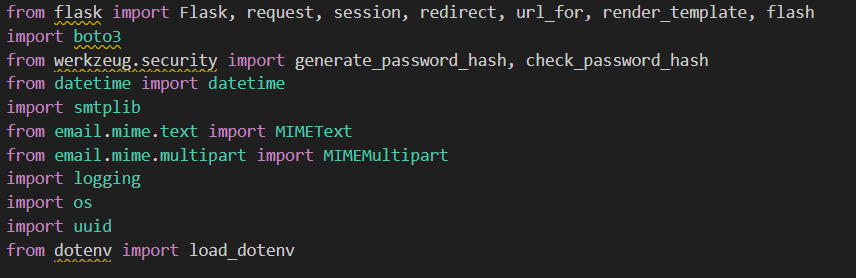
* Set up Flask app with routing and templates.



* Use local Python lists/dictionaries for initial testing.

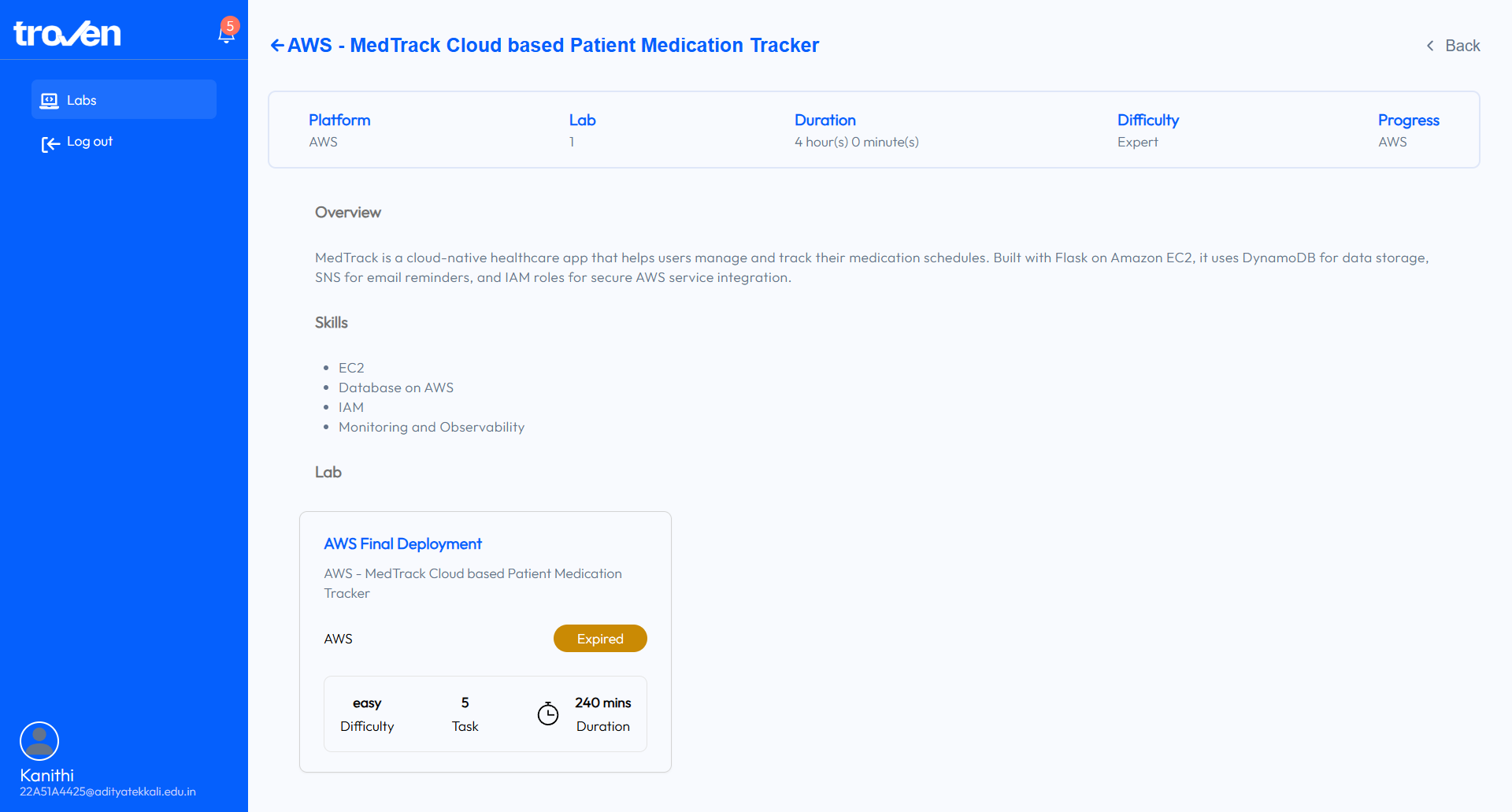


* Integrate AWS services (DynamoDB, SNS) using boto3.



**Milestone 2: AWS Account Setup**

* Access AWS via Troven Labs.

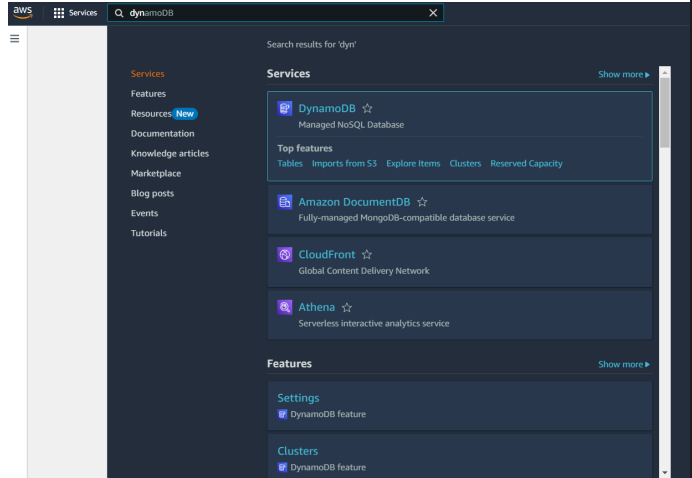


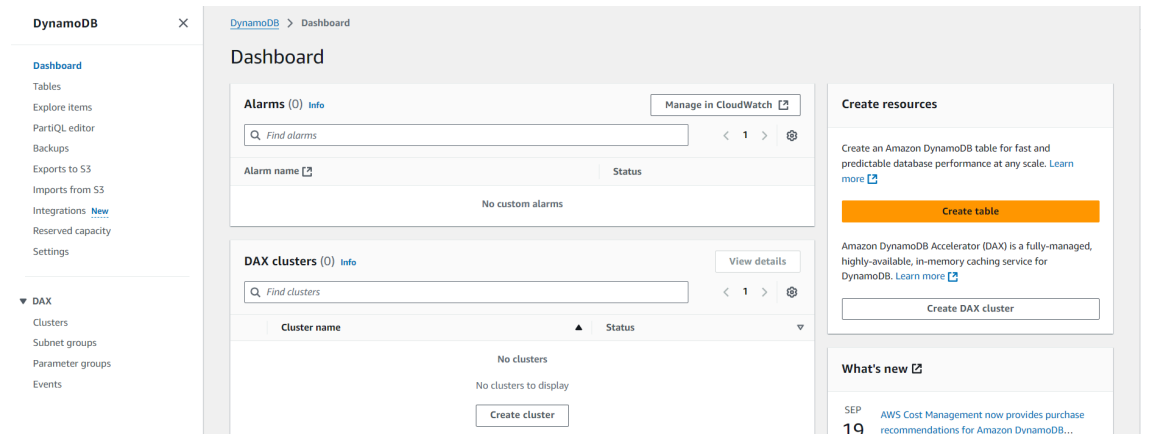
* Avoid personal AWS account usage to prevent billing issues.

**Milestone 3: DynamoDB Database Creation and Setup**

**Activity 3.1:Navigate to the DynamoDB**

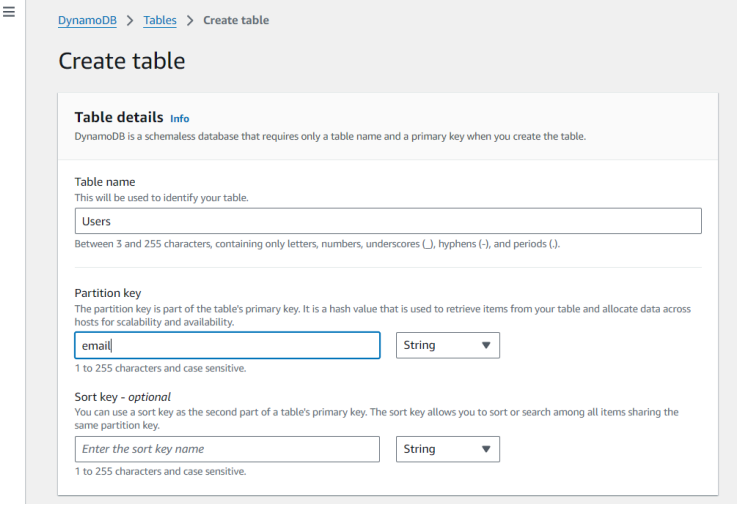
**○ In the AWS Console, navigate to DynamoDB and click on Create tables**

****

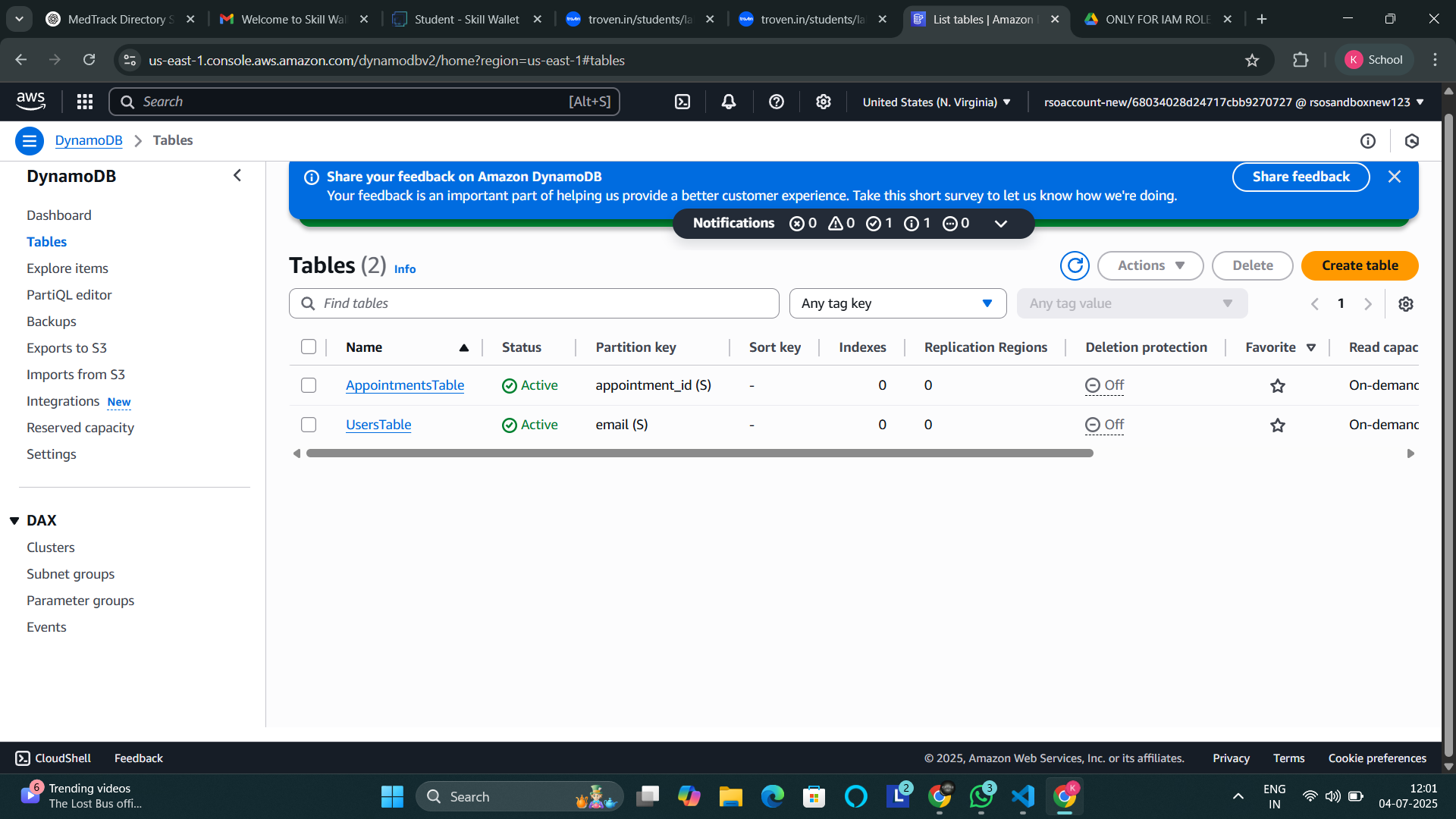
****

**Activity 3.2:Create a DynamoDB table for the Create Users table (Primary key: Email).**

○ **Create Users table with partition key “Email” with type String and click on create tables.**

****

**Activity 3.3:Create Appointments table (Primary key: appointment\_id).**

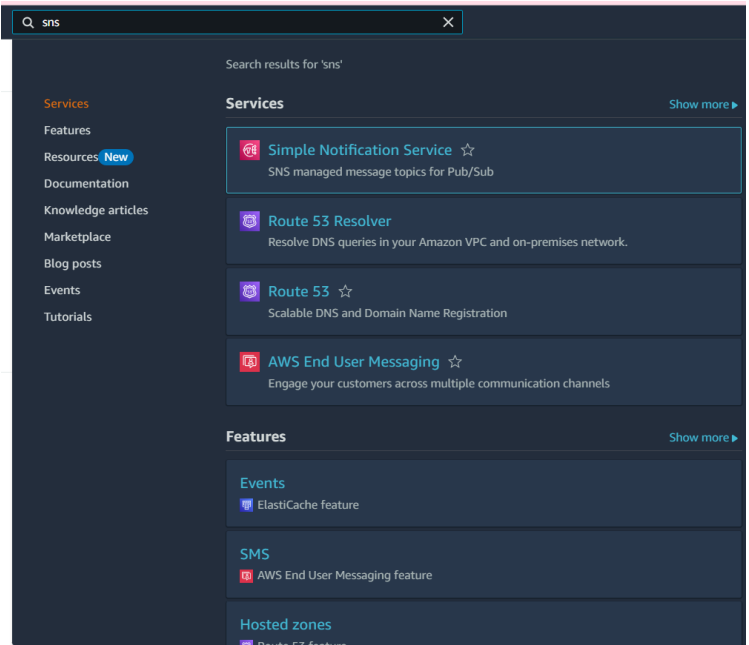


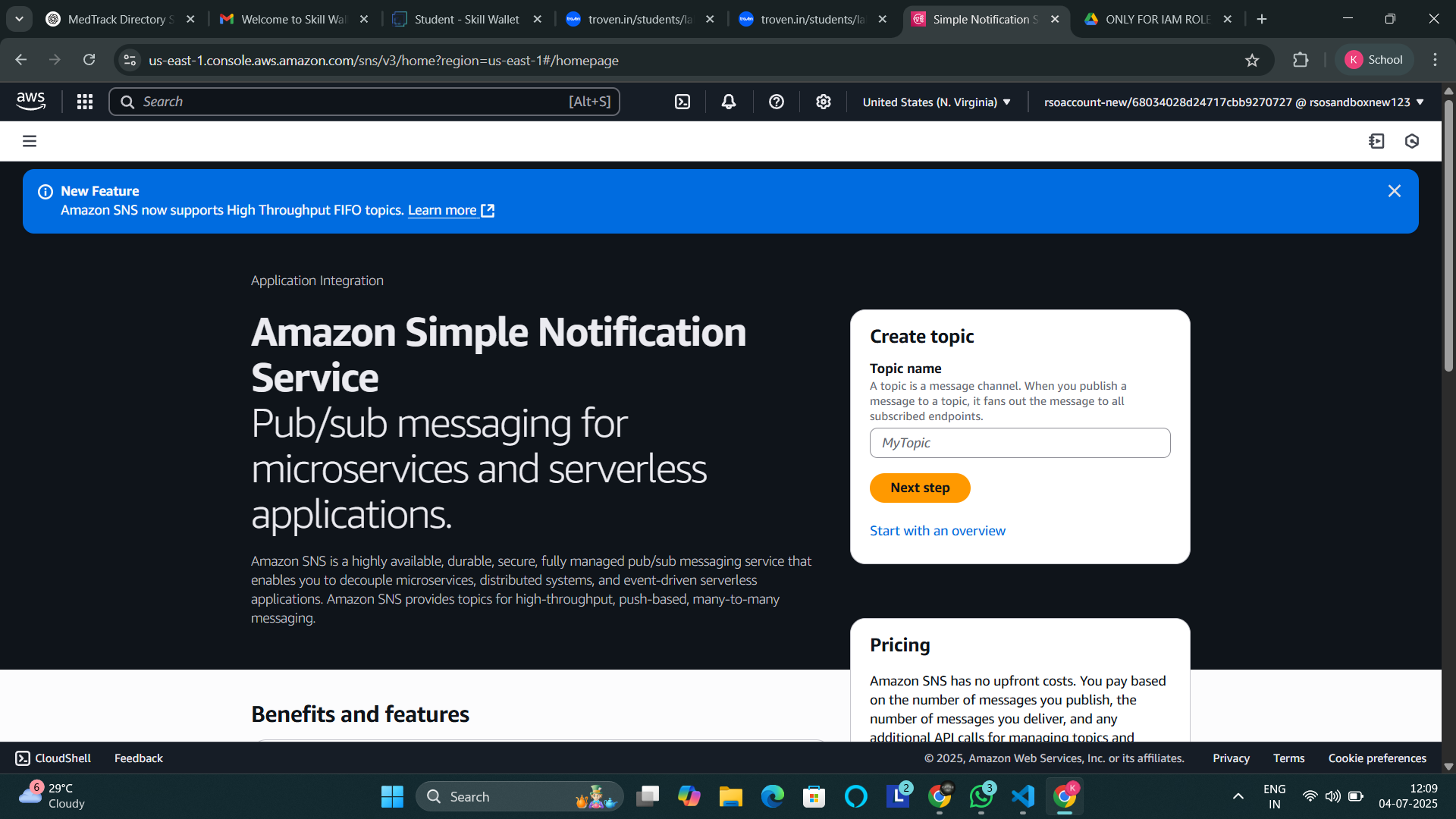
Follow the same steps to create an Appointments table with Email as the primary key for booking diagnosis data.

**Milestone 4: SNS Notification Setup**

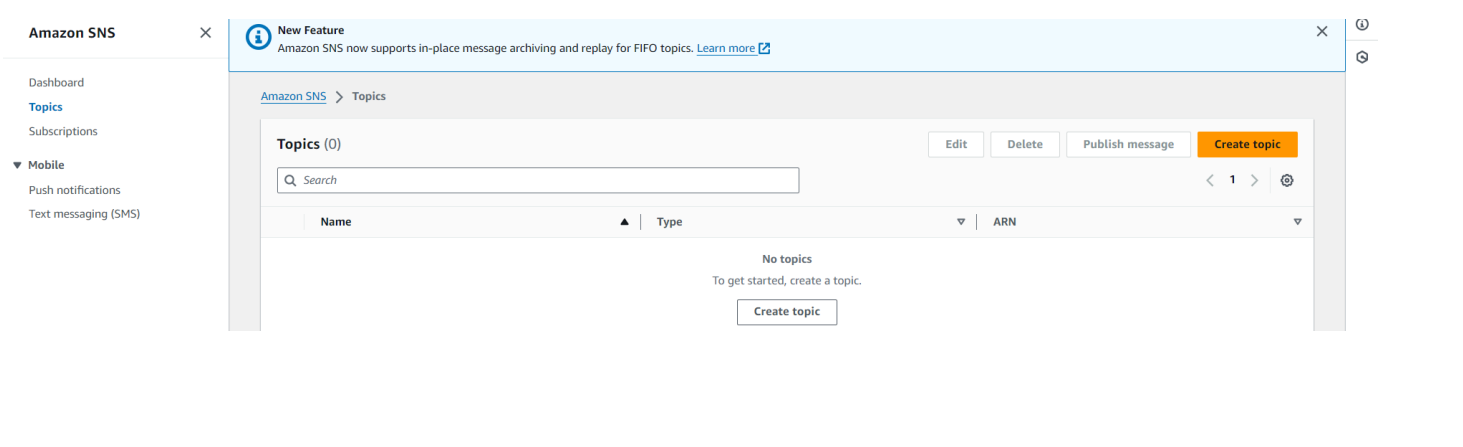
* **Activity 4.1: Create SNS topics for sending email notifications to users and doctor prescriptions.**

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

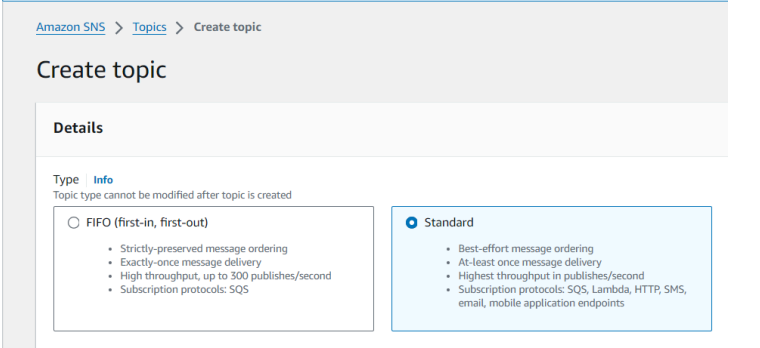




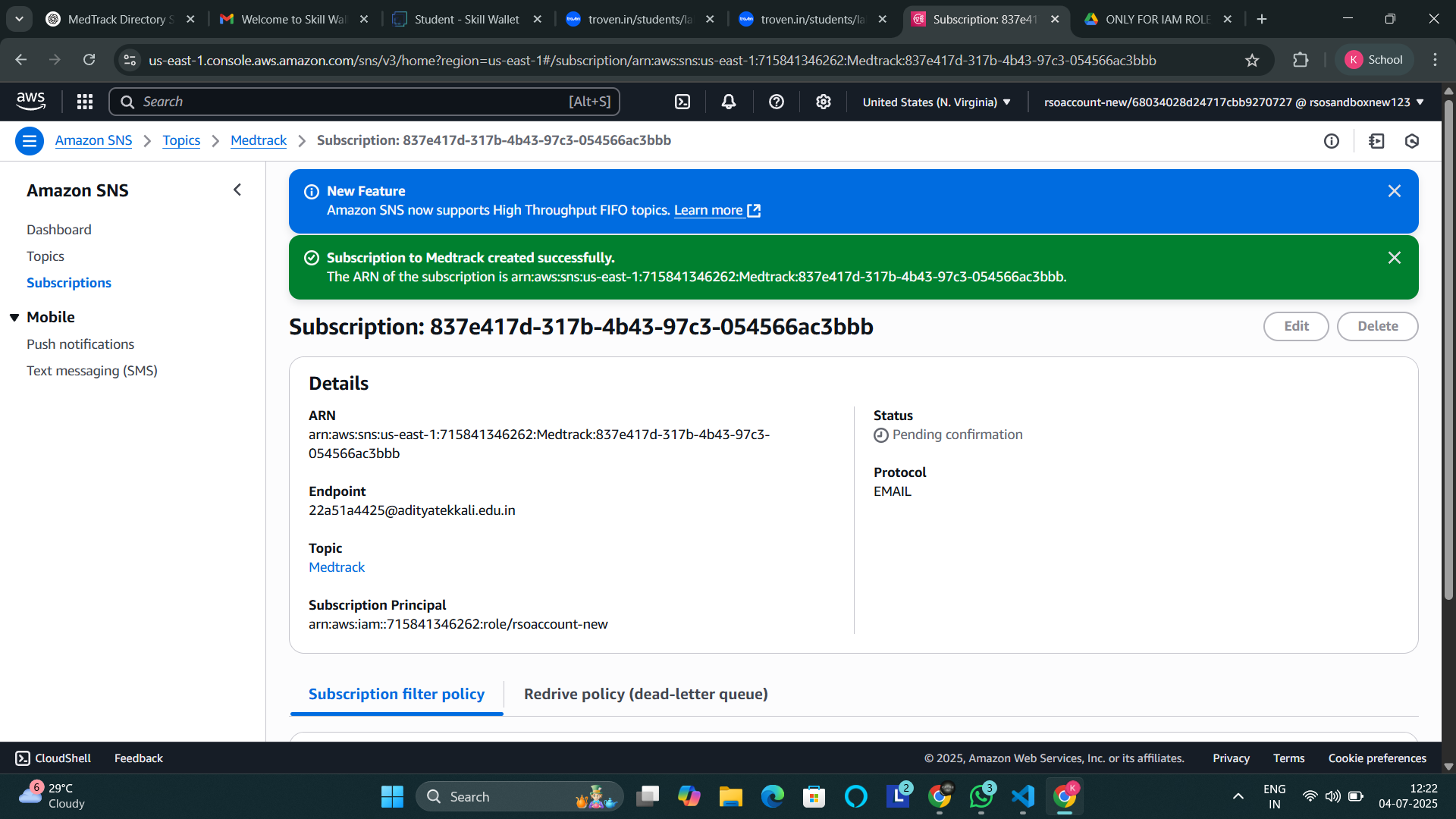
○ Click on Create Topic and choose a name for the topic.



○ Choose Standard type for general notification use cases, and click on Create Topic.

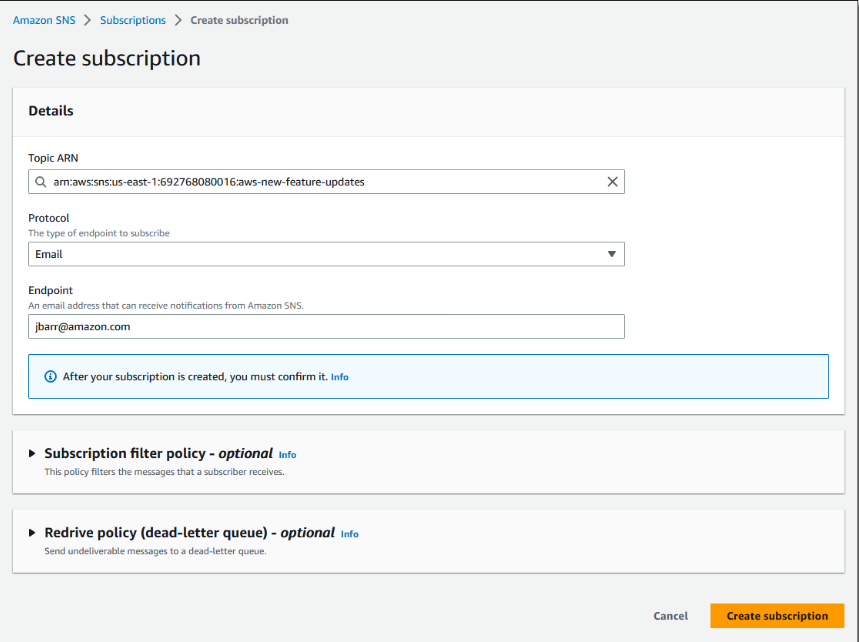


○ Configure the SNS topic and note down the Topic ARN.

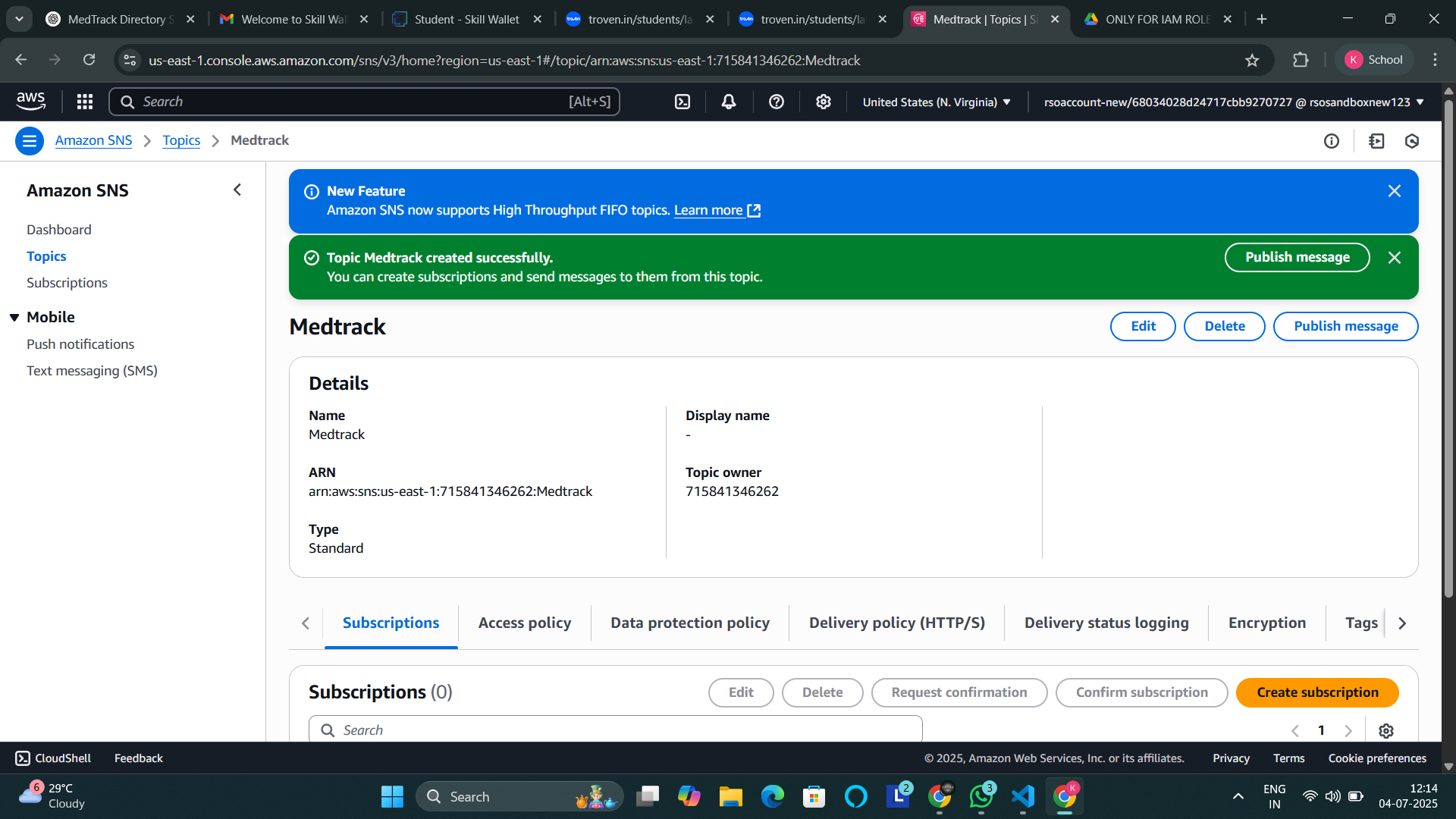


**● Activity 4.2: Subscribe users and Doctors to relevant SNS topics to receive real-time notifications when a book appointment is made.**

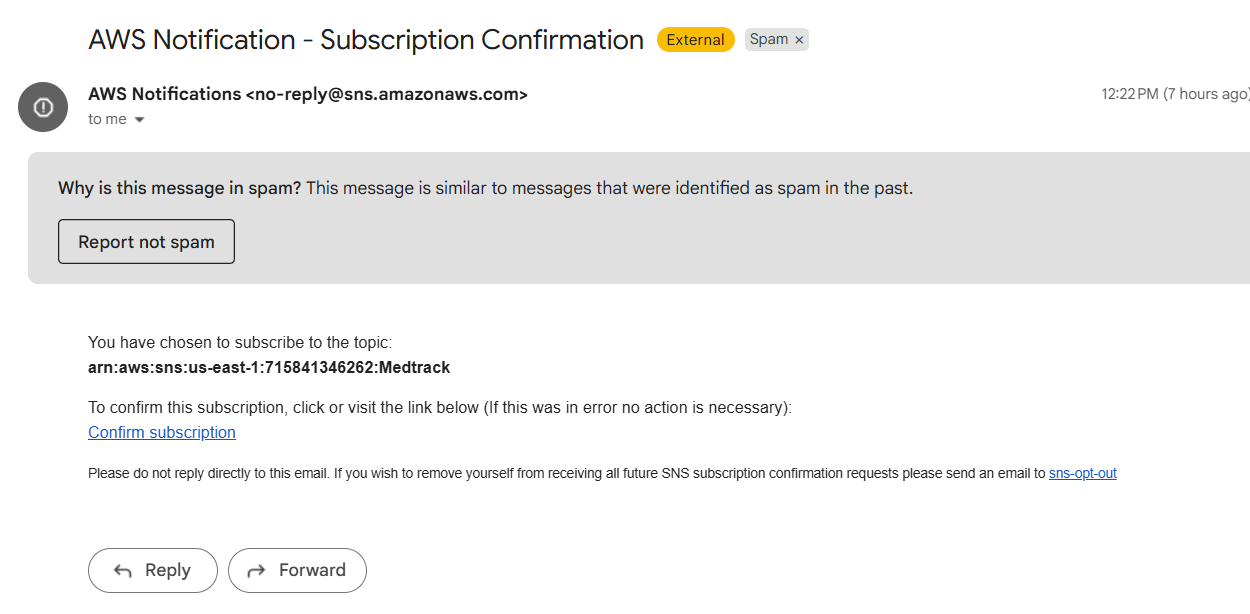
○ Subscribe users (or admin staff) to this topic via Email. When an appointment is made, notifications will be sent to the subscribed emails.

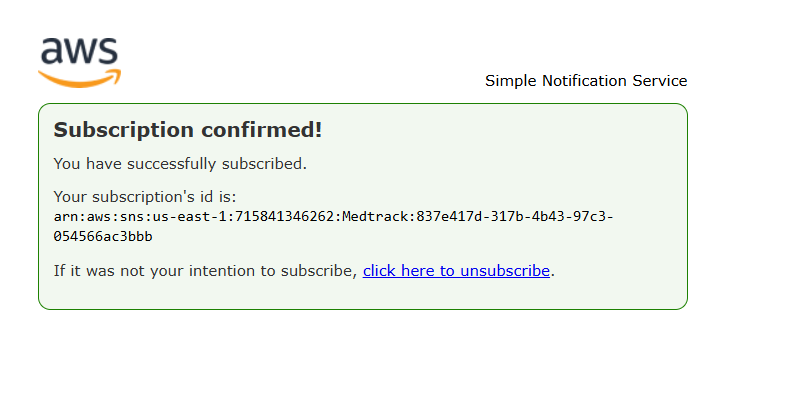


After the subscription request for the mail confirmation.

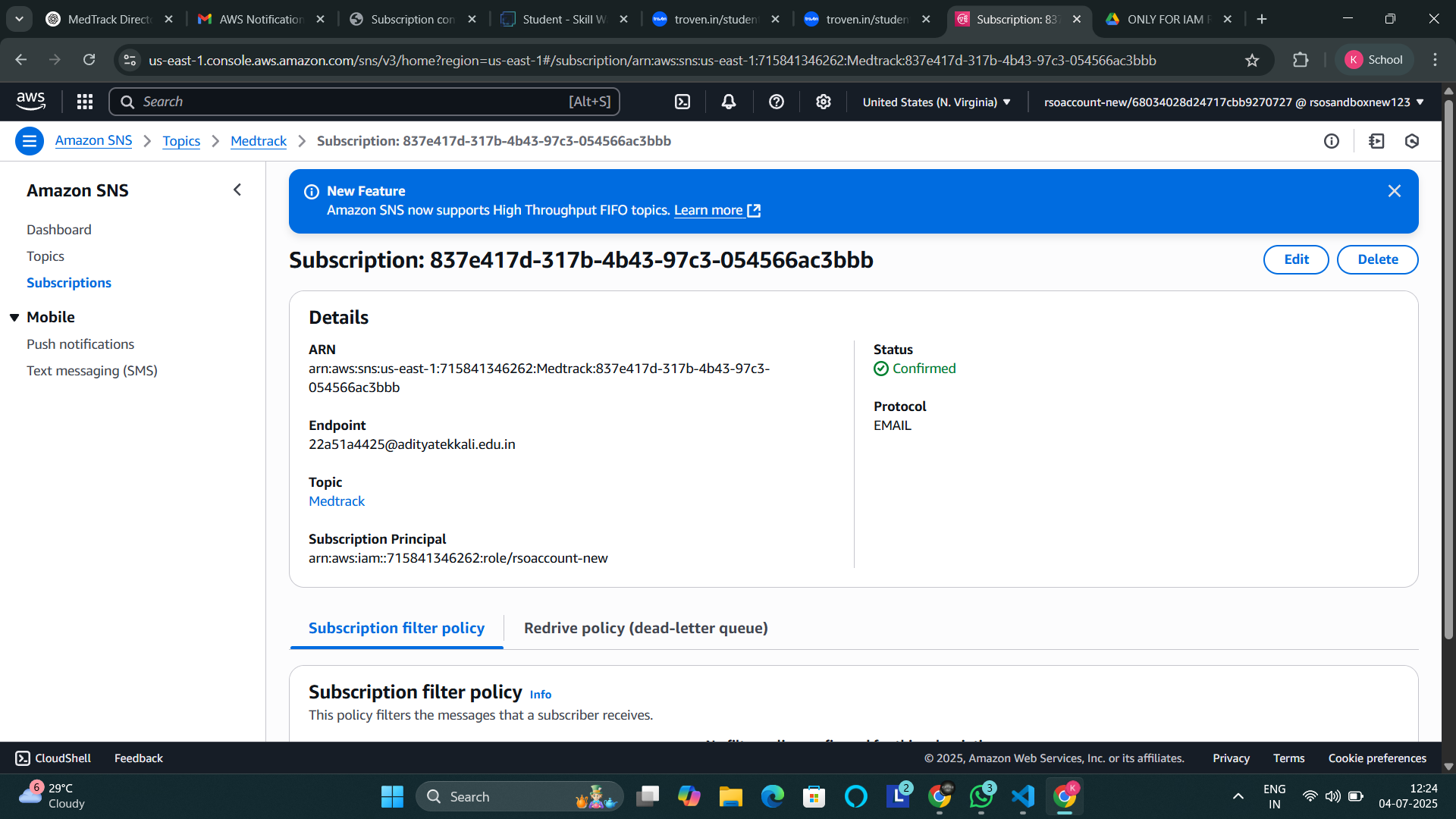


**Navigate to the subscribed Email account and click on the confirm subscription in the AWS Notification- Subscription Confirmation email.**

****

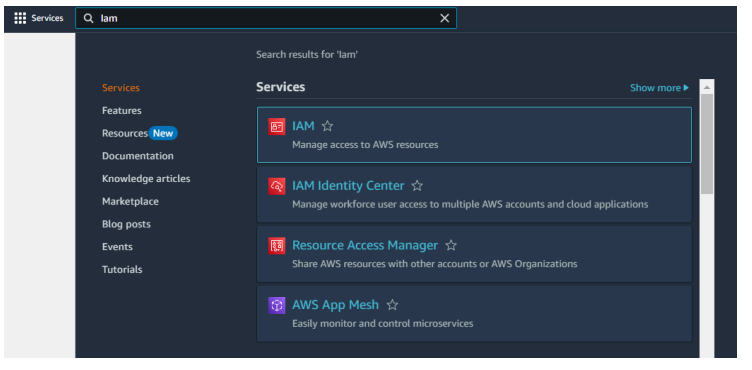
****

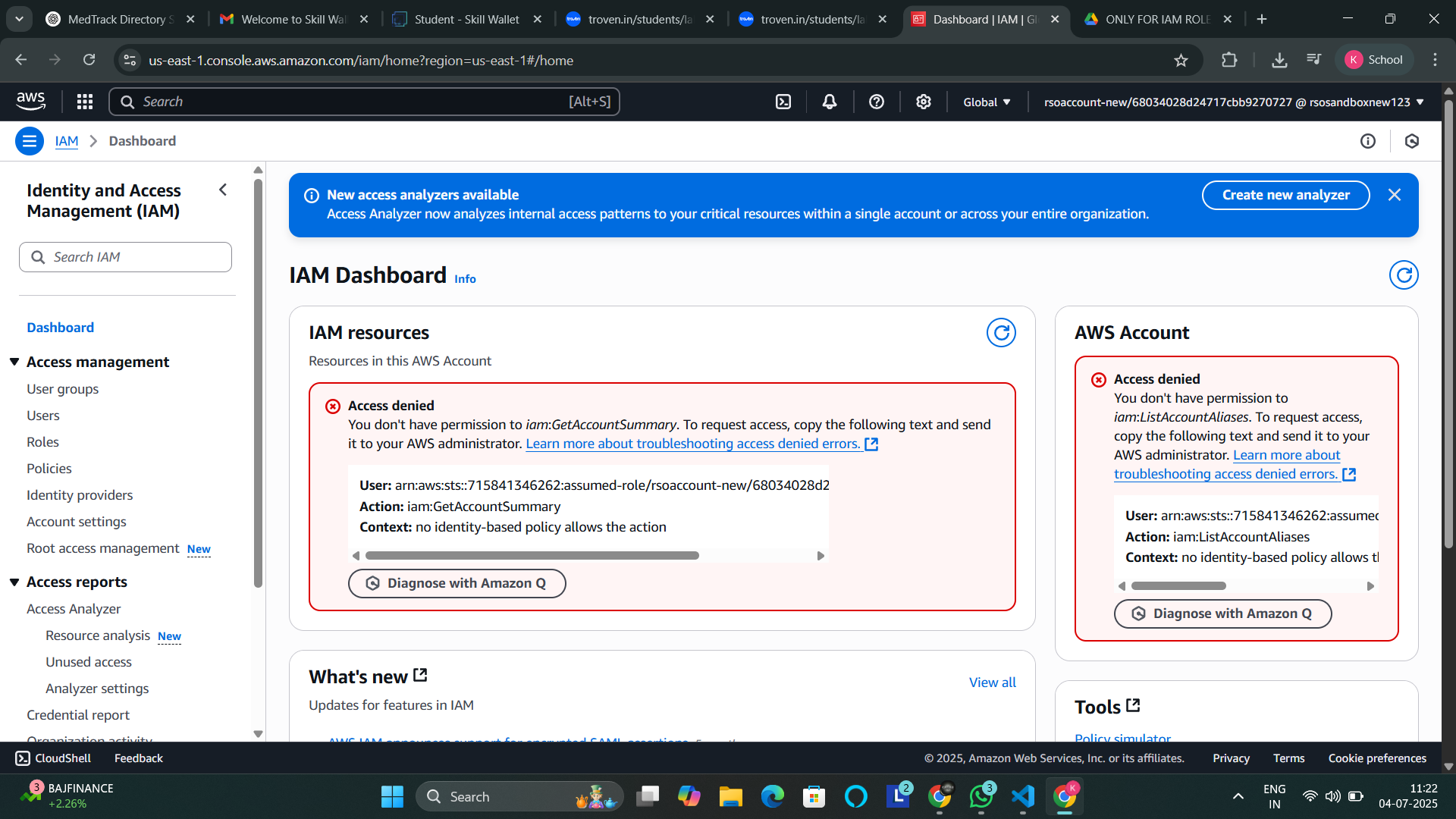
* Successfully done with the SNS mail subscription and setup, now store the ARN link.



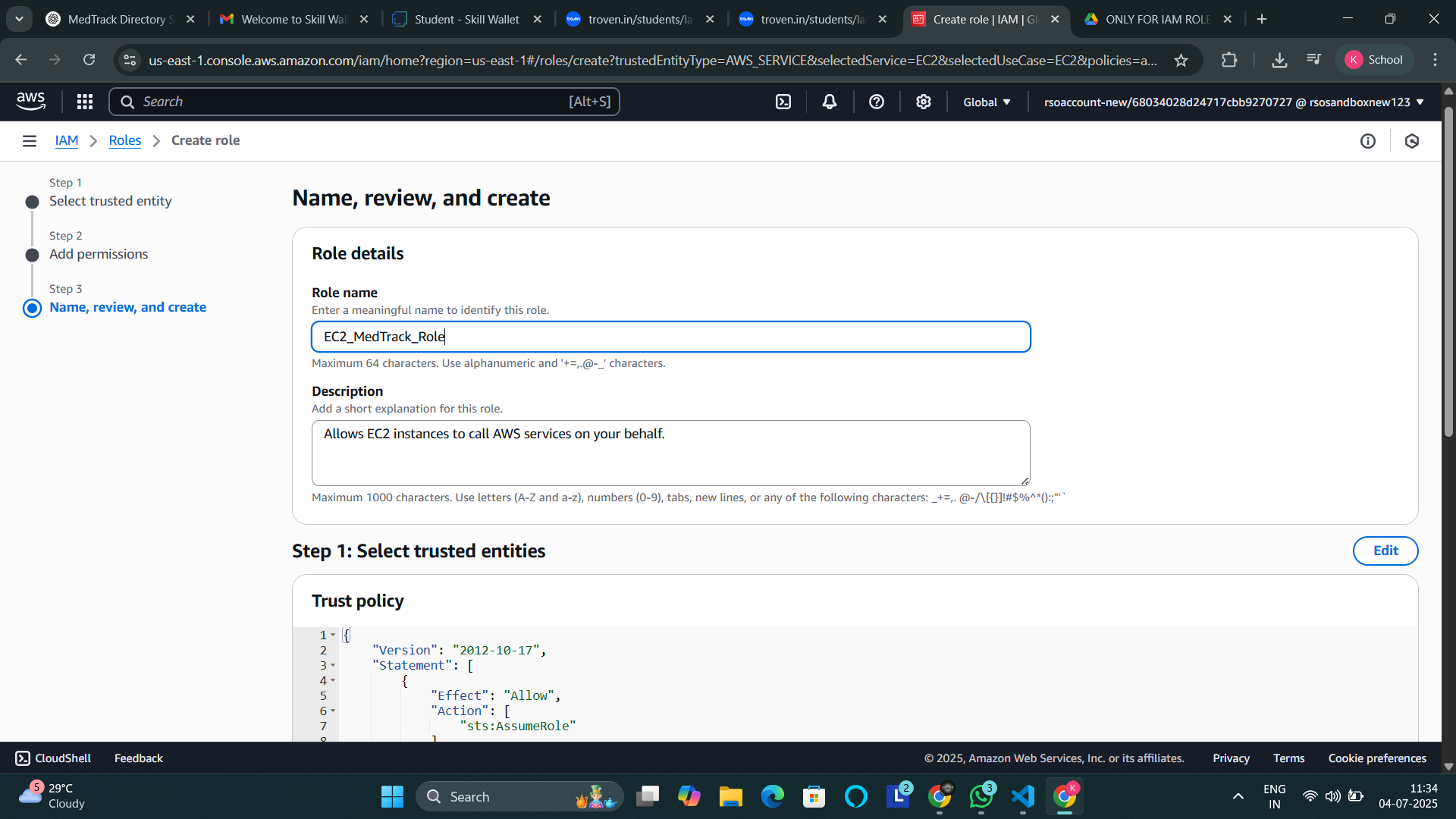
**Milestone 5: IAM Role Setup**

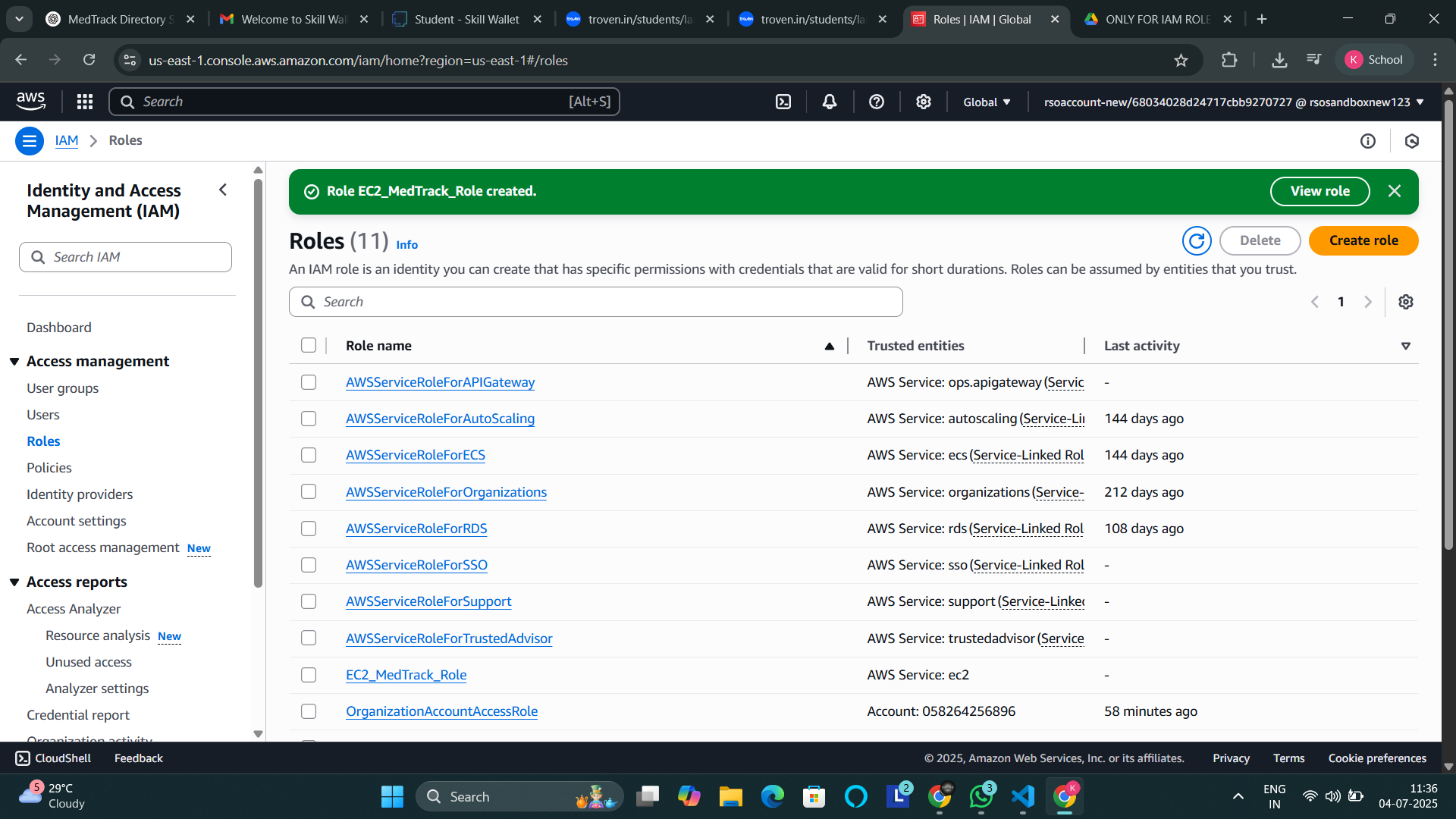
* **Activity 5.1:Create IAM Role. ○ In the AWS Console, go to IAM and create a new IAM Role for EC2 to interact with DynamoDB and SNS.**

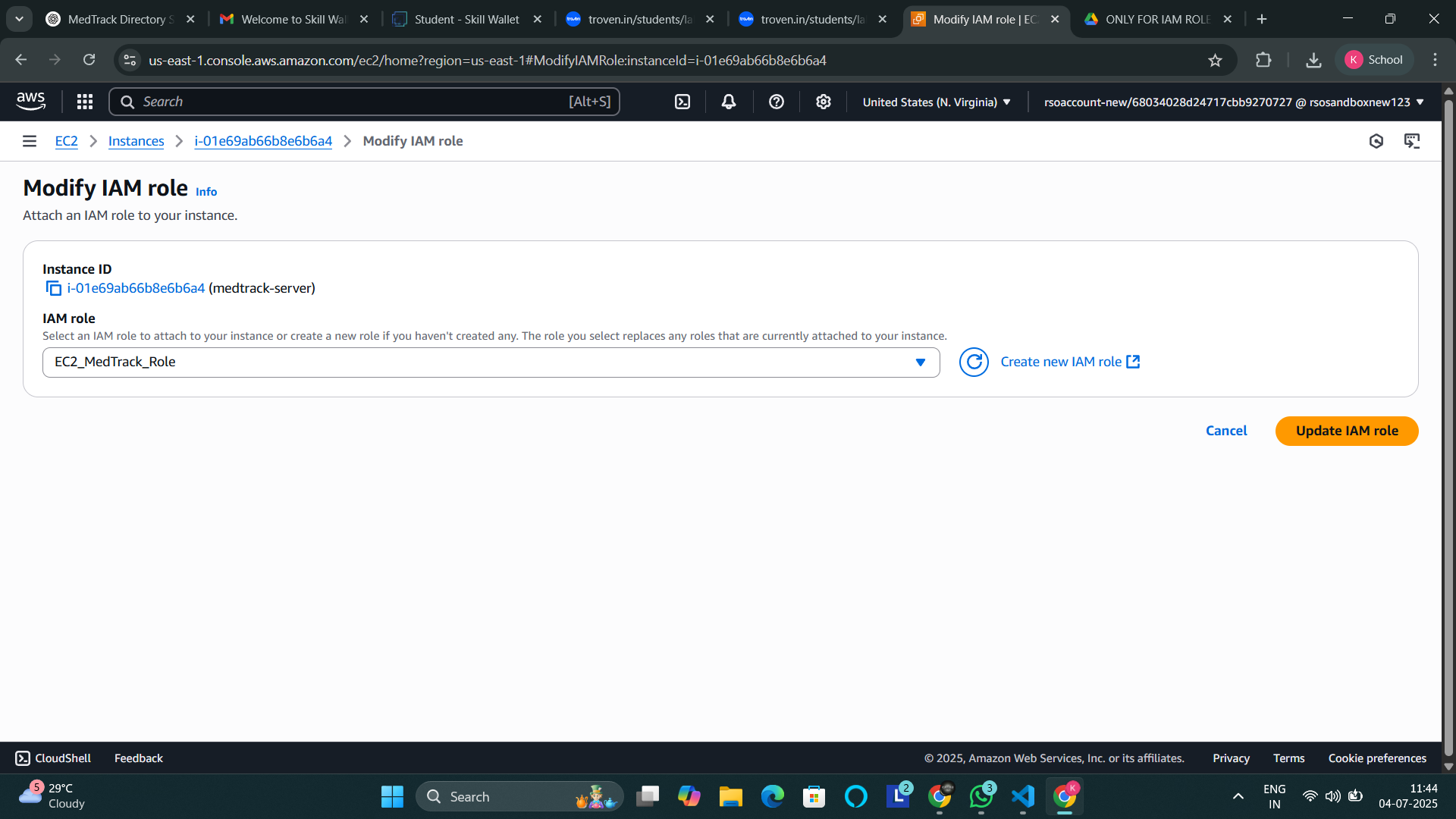
****

****

* **Create IAM Roles:**

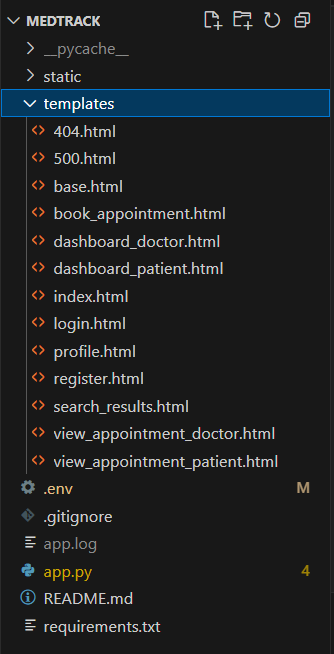
****

* **Attach policies: AmazonDynamoDBFullAccess, AmazonSNSFullAccess.**



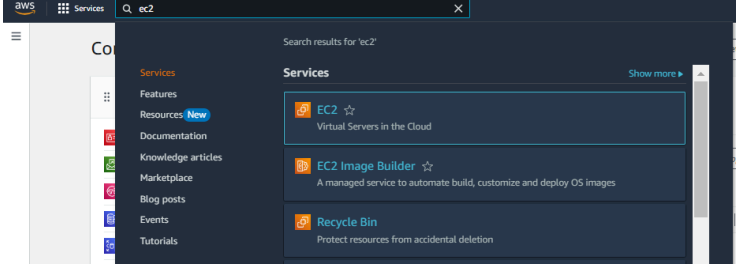
**Milestone 6: EC2 Instance Setup**

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

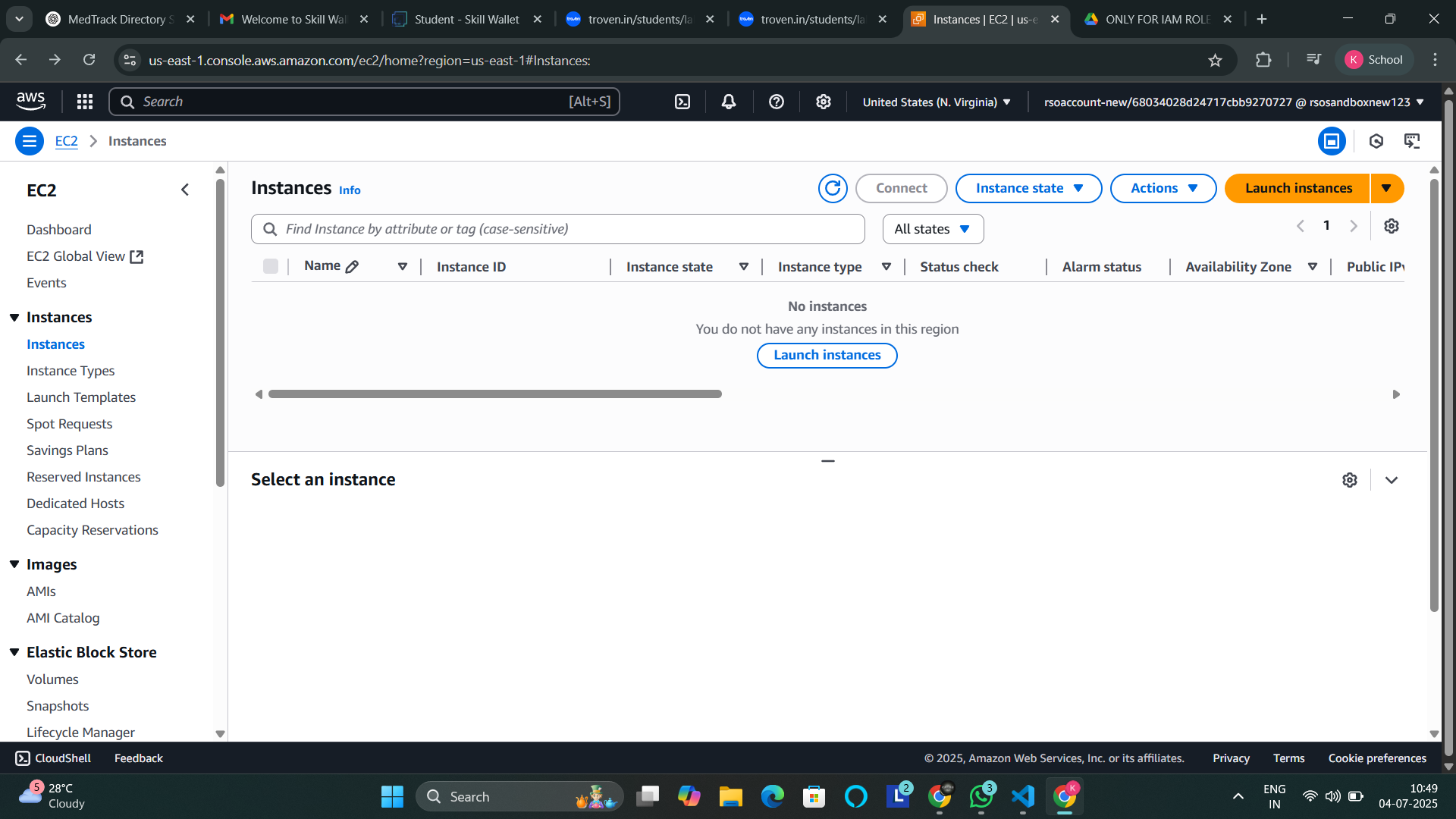
****

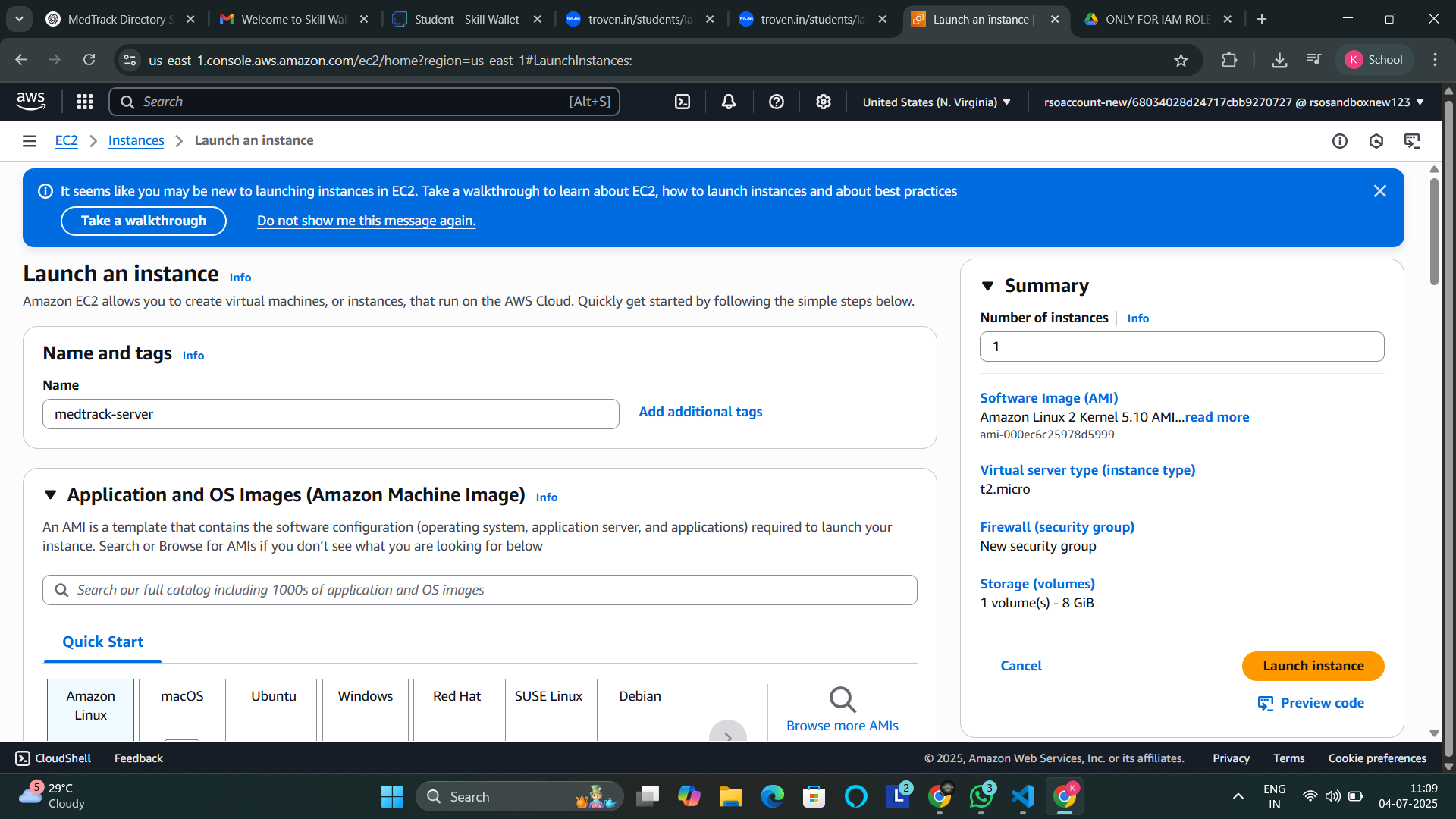
**Launch an EC2 instance to host the Flask application.**

**○ In the AWS Console, navigate to EC2 and launch a new instance.**

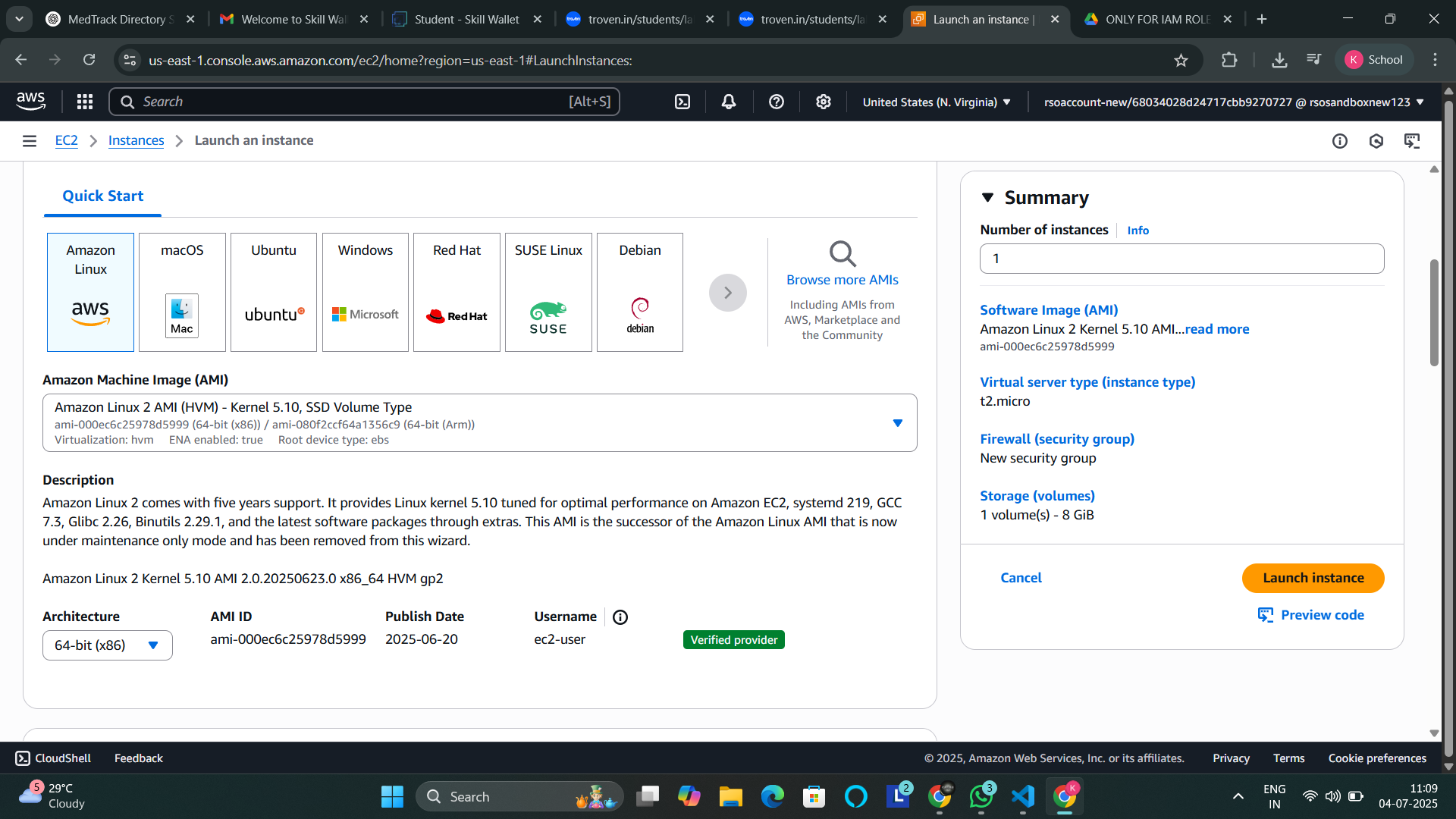
****

* **Click on Launch instance to launch EC2 instance**

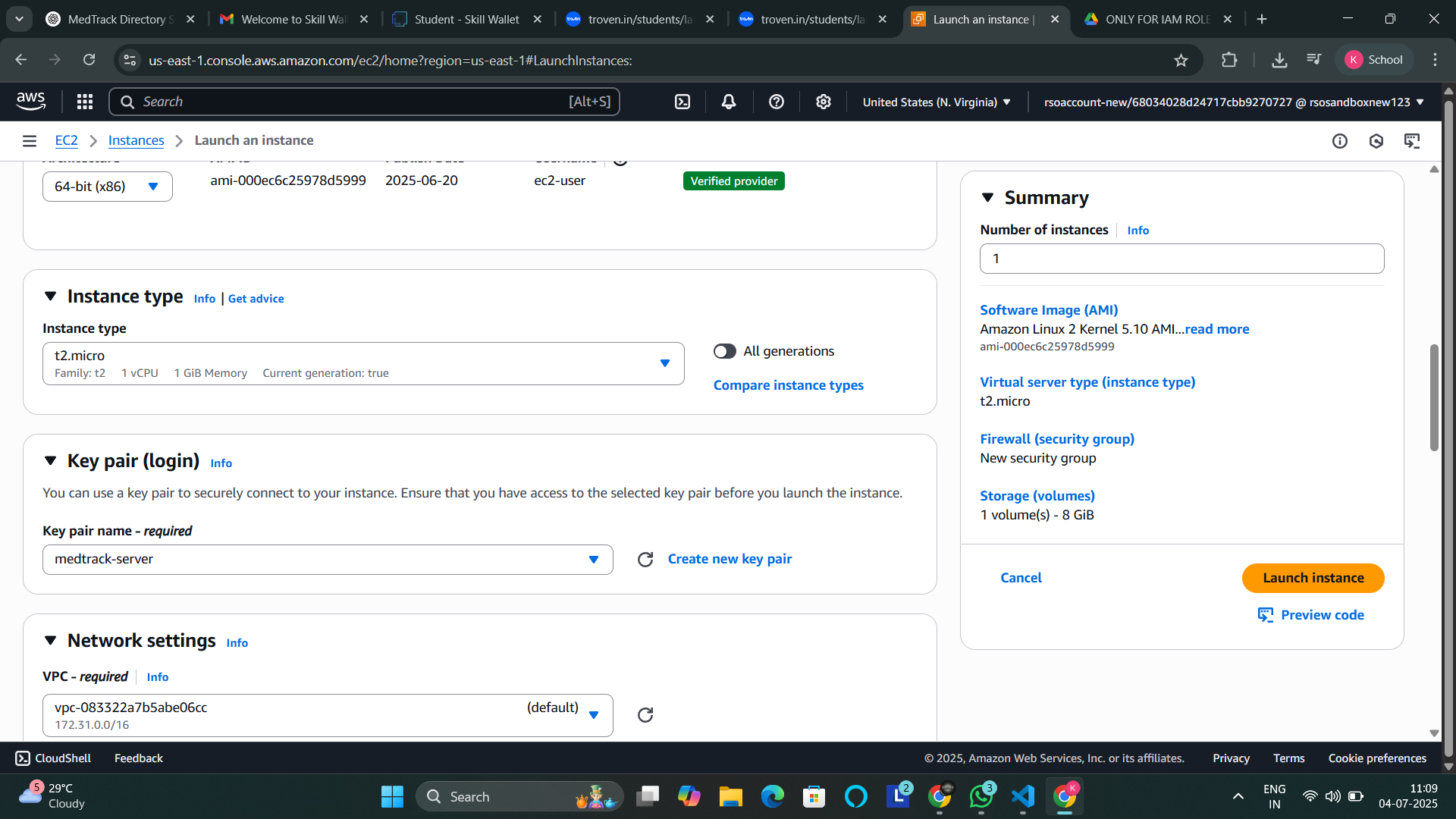
****

****

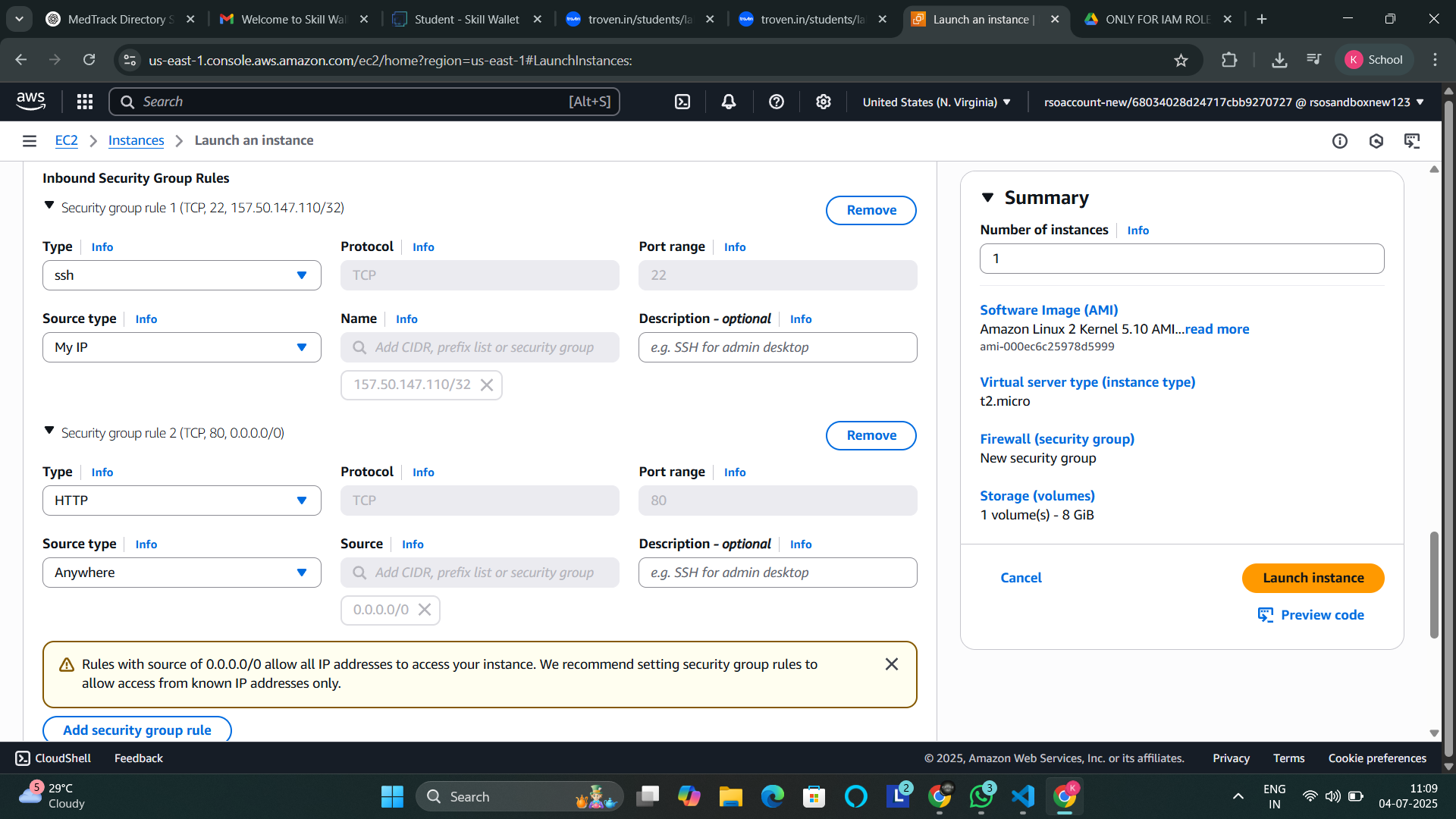
* **Launch EC2 instance (Amazon Linux 2/Ubuntu, t2.micro).**

****

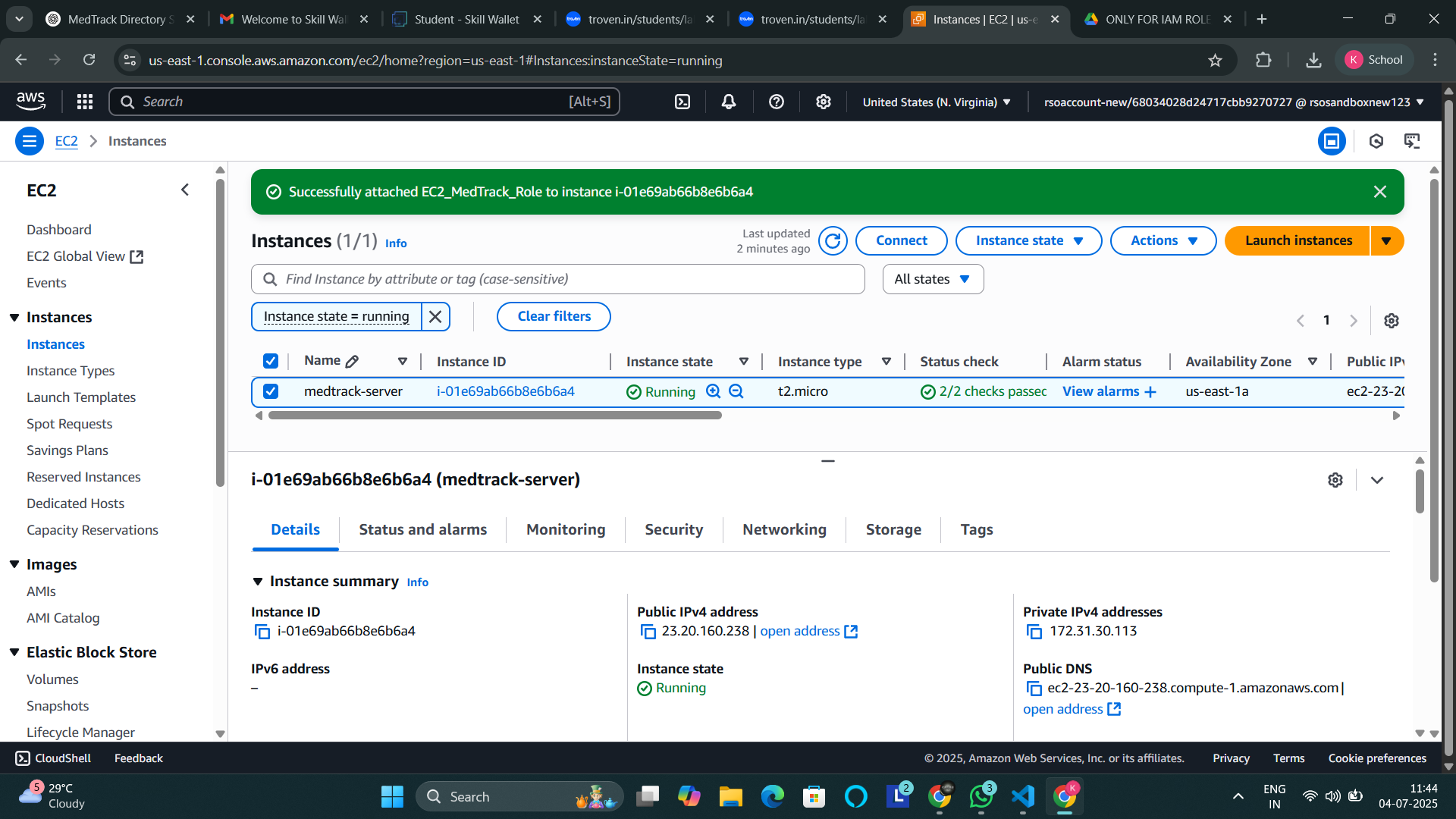
* **Assign an IAM Role and key pair.**

****

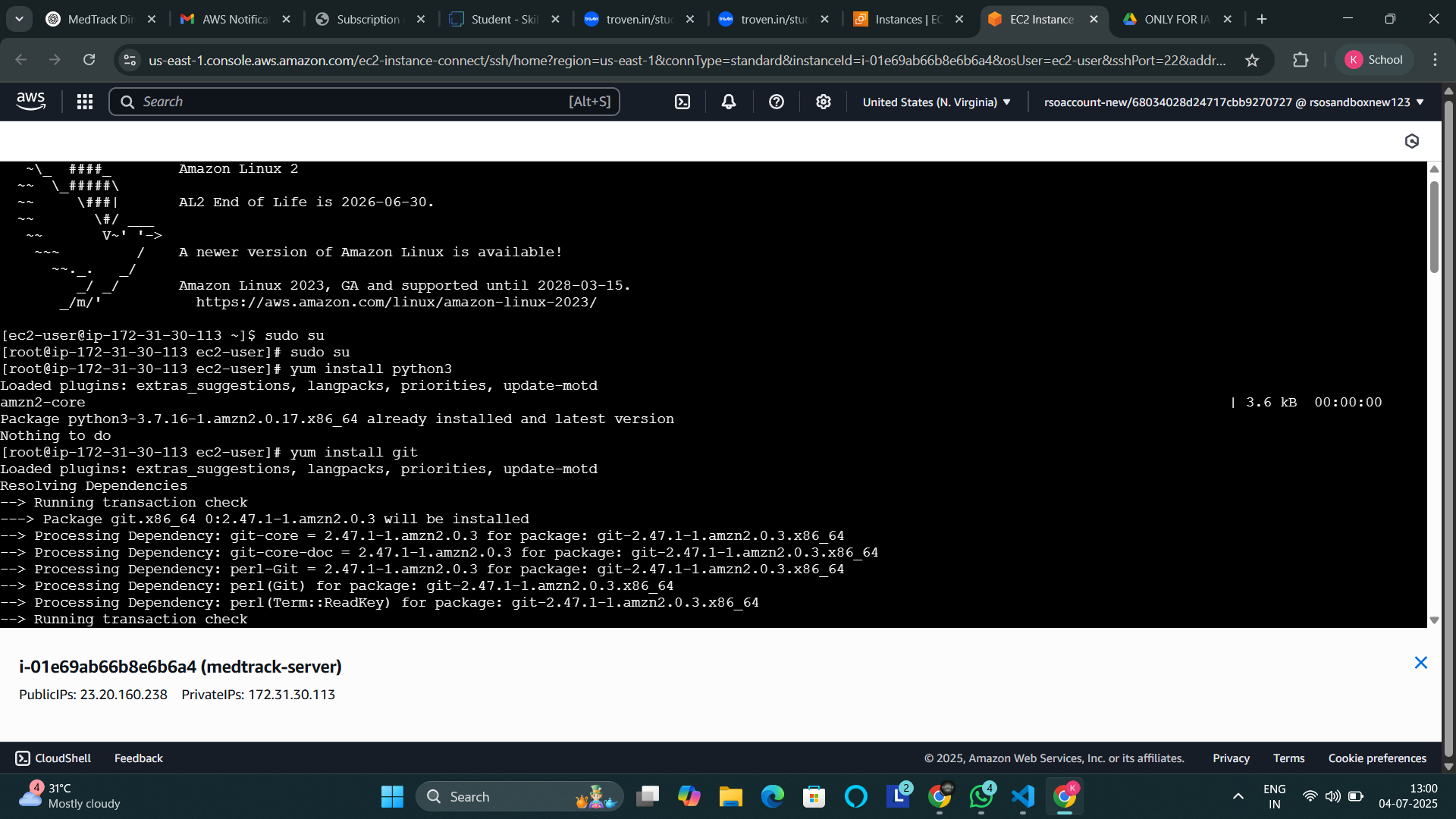
* **Configure security groups for HTTP/SSH.**

****

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

****

* **Now connect the EC2 with the files.**

****

**Milestone 7: Deployment on EC2**

**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

**Clone Your Flask Project from GitHub:**

Run: ‘git clone<https://github.com/saiganeshpadhy/medtrack.git>

Note: change your-github-username and your-repository-name with your credentials. here: ‘git clone<https://github.com/saiganeshpadhy/medtrack.git>

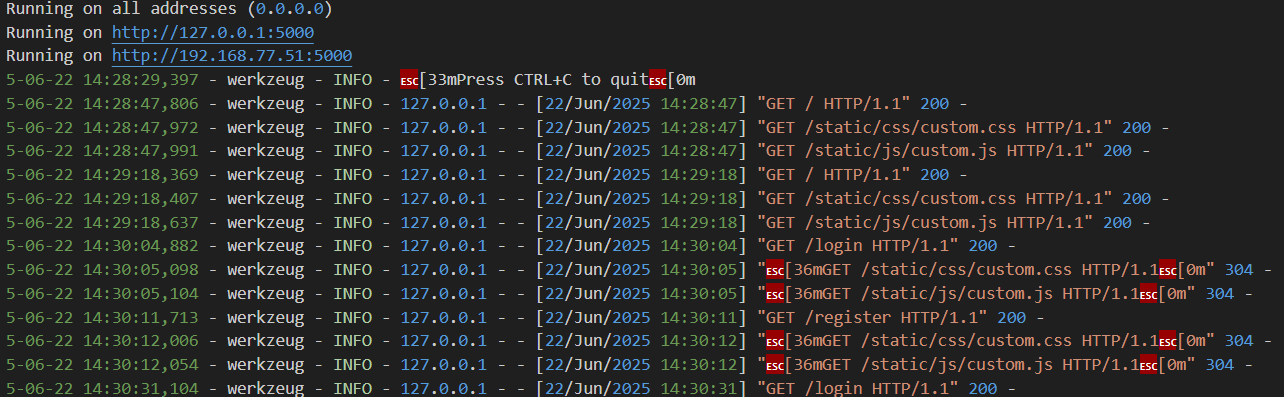
* This will download your project to the EC2 instance.
* To navigate to the project directory, run the following
* command: cd Medtrack.

**Once inside the project directory, configure and run the Flask application by executing the following command with elevated privileges: Run the Flask Application.**

* Run Flask app: sudo flask run --host=0.0.0.0 --port=5000

**Verify the Flask app is running:**

* http://your-ec2-public-ip
* Run the Flask app on the EC2 instance

****

**Access the website through:**

* PublicIPs: <http://127.0.0.1:5000>

**Milestone 8: Testing and Deployment**

* Verify registration, login, appointment booking, and SNS notifications.

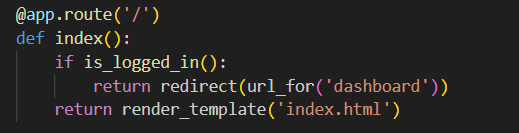
### 

### 

### 

### **Flask Application Structure & Code**

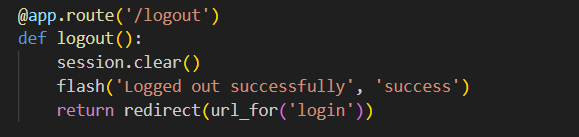
**App Initialization:**

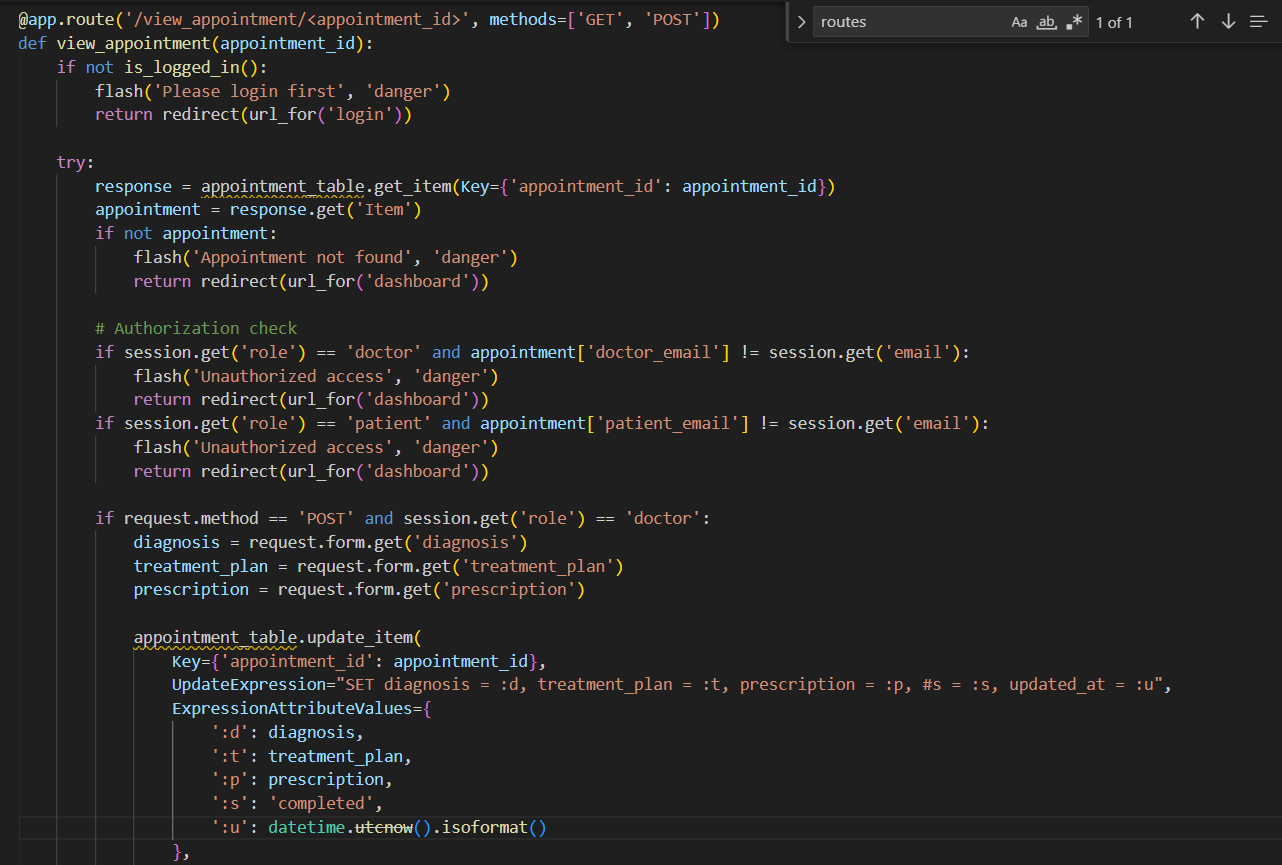
* Setup routes: register, login, dashboard, book appointment, view appointment, search, profile.  
  
* Connect to DynamoDB and SNS using boto3 with the correct region and ARN.

**Routes:**

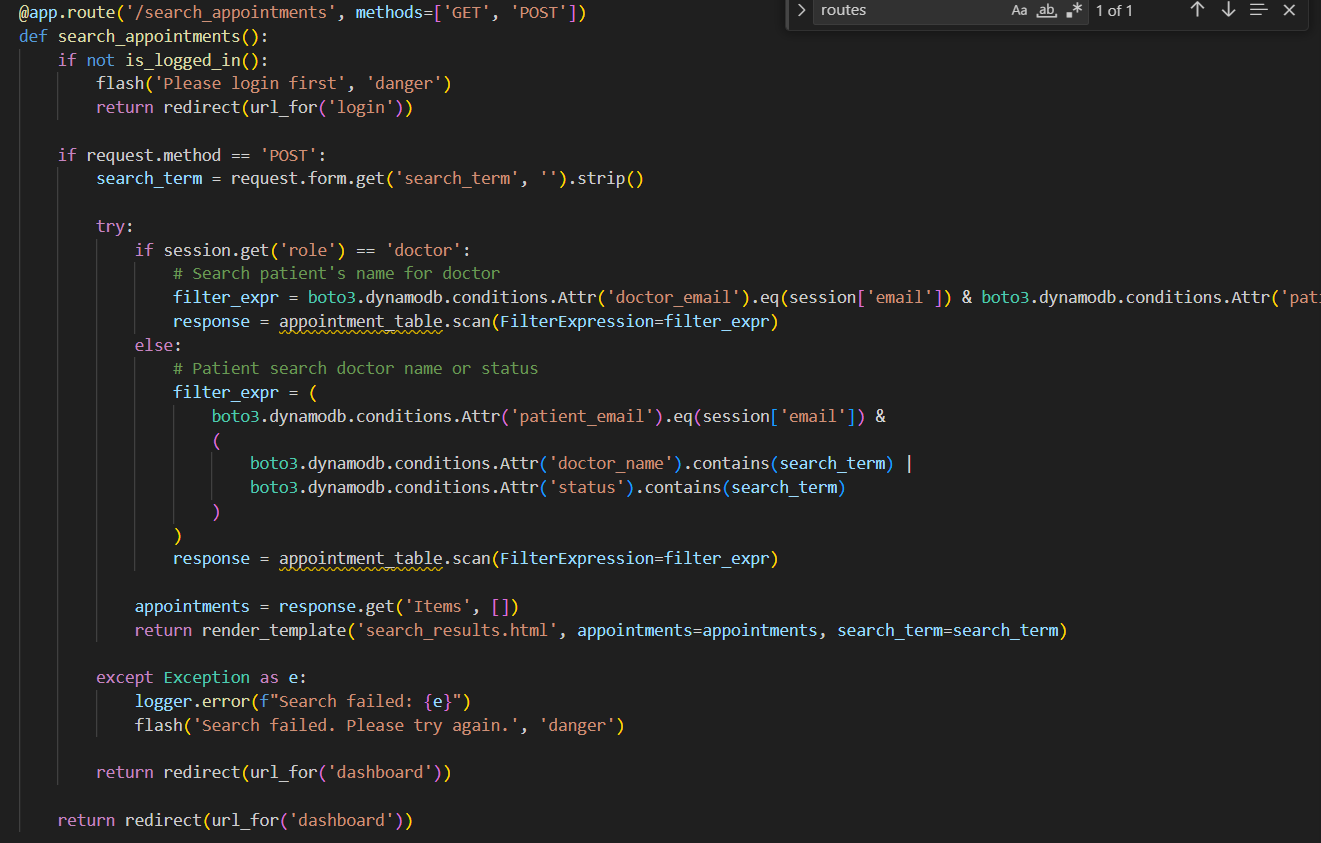
* **Register:** Register the user, hash the password, and store it in DynamoDB.



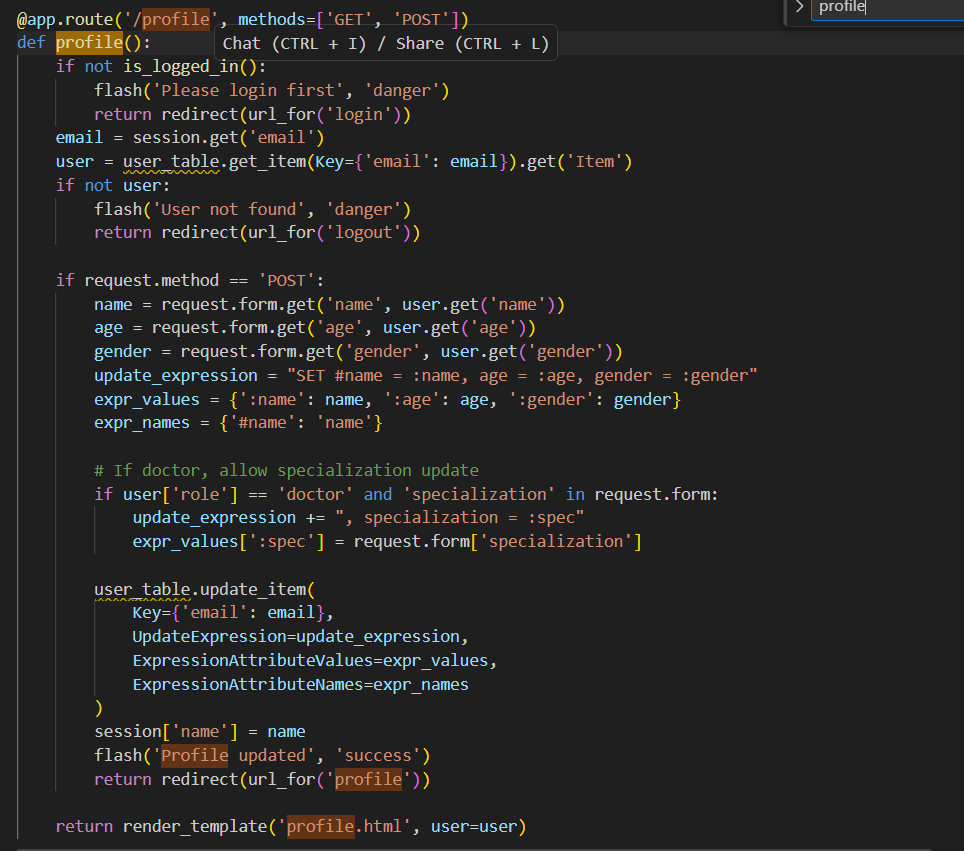
* **Login:** Authenticate and update login count.  
  
* **Logout:** End session.  
  
* **Book Appointment:** Collect and store appointment details, trigger SNS.
* **View Appointments:** Retrieve data from DynamoDB.



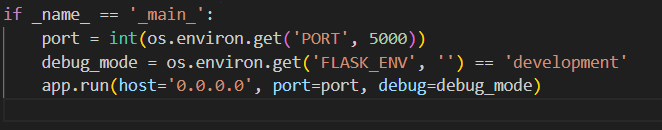
* **Search:** Filter appointments.



**Profile:** **View/edit personal data:**

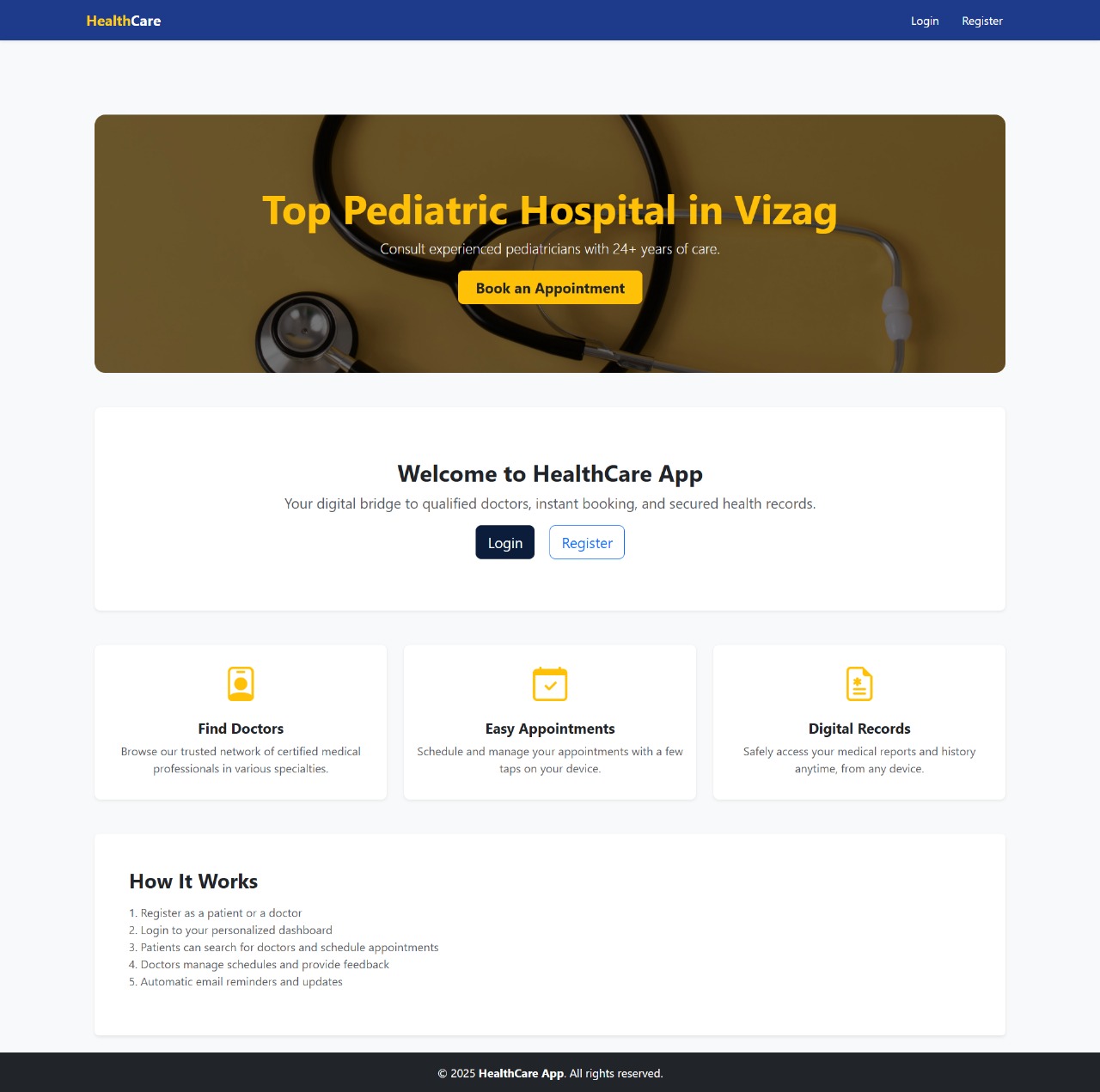


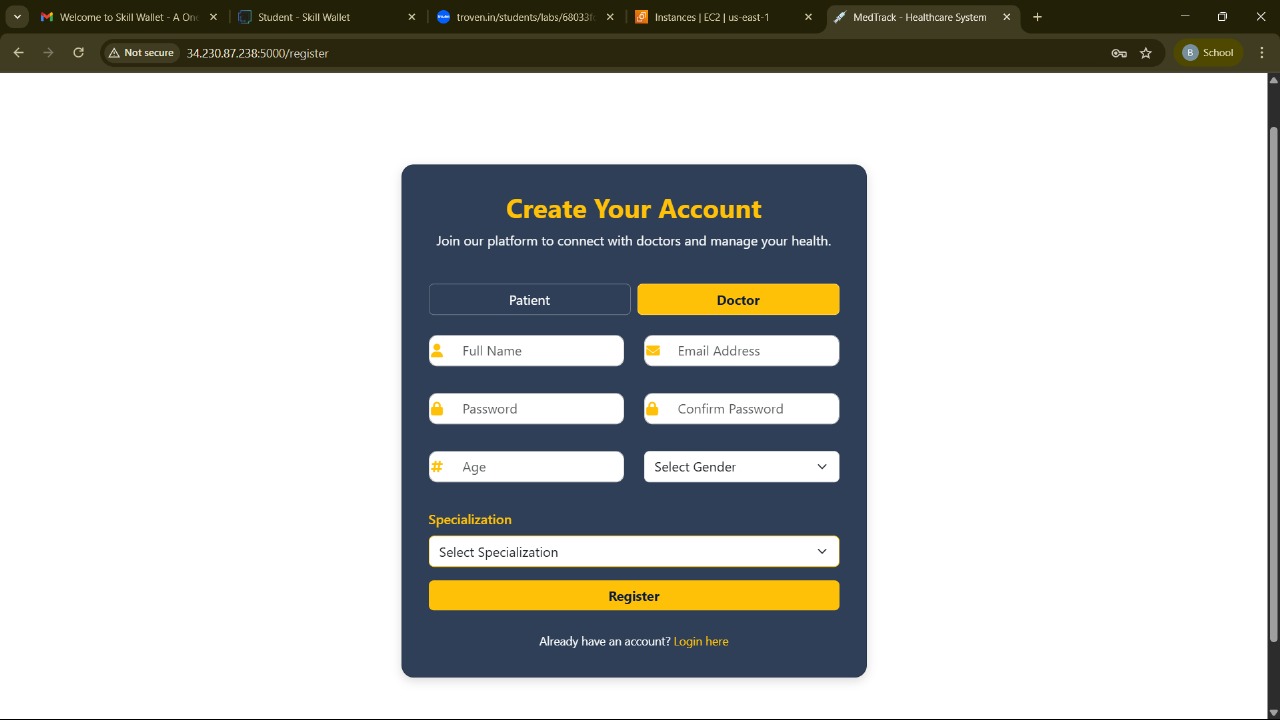
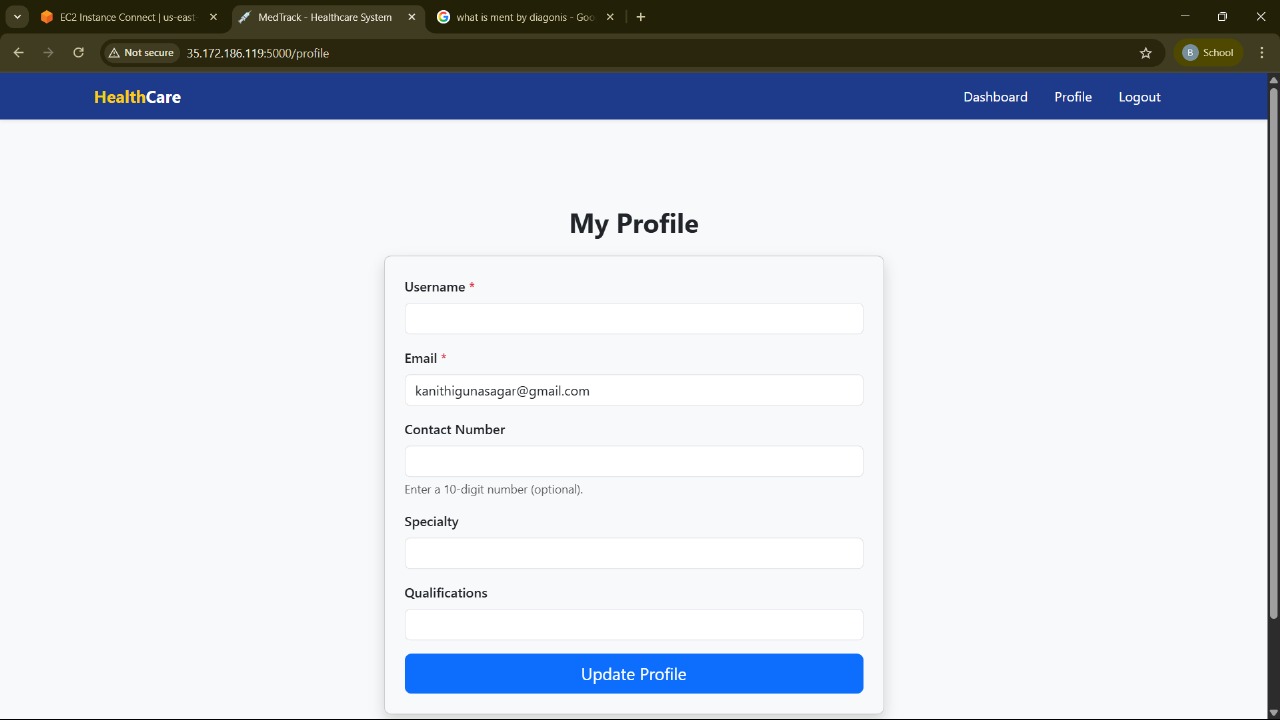
**Deployment Code:**

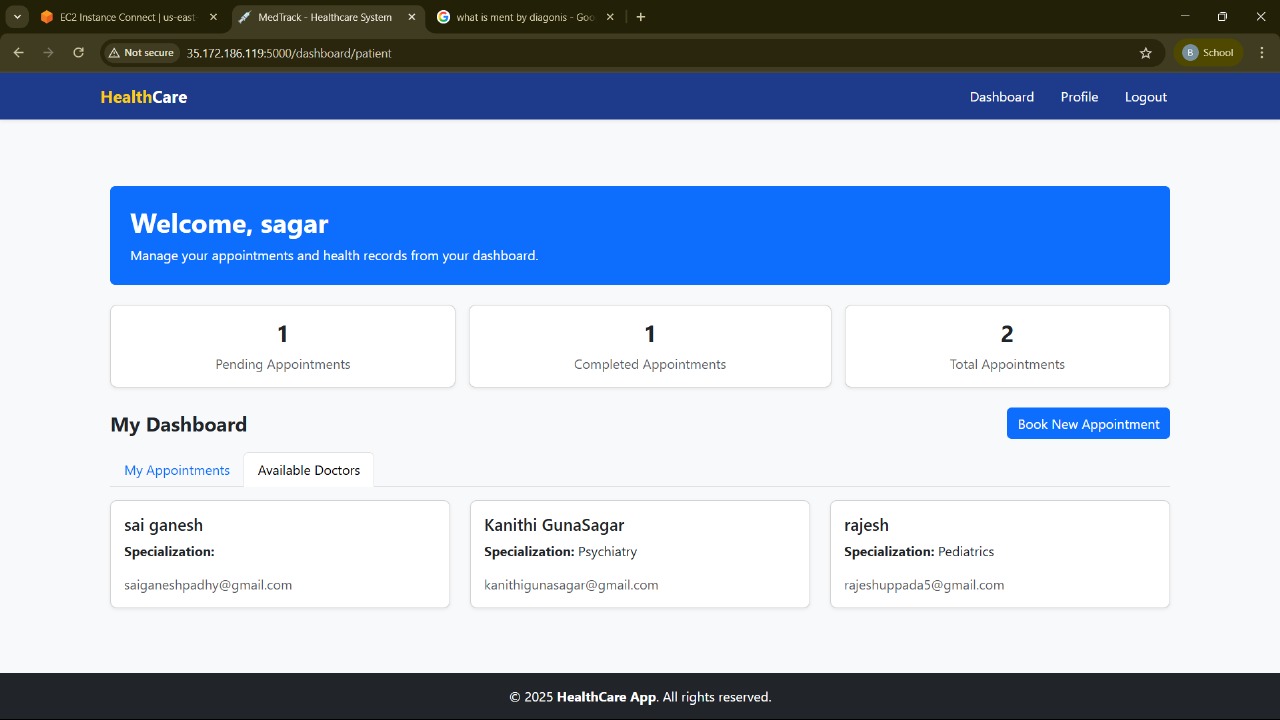


### **Functional Testing Summary**

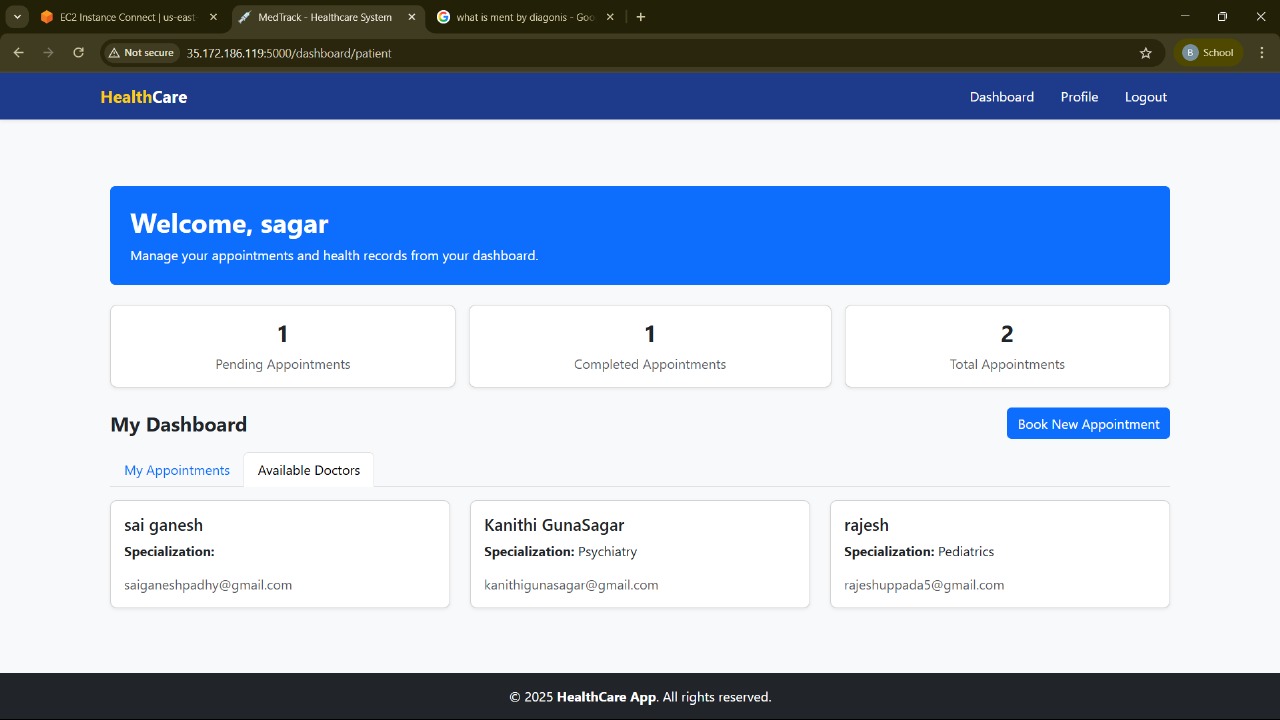
* **Home Page:** Entry point with navigation and responsive design.



* **Doctor & Patient Registration:** Collects and validates credentials.
* 
* **Login Pages:** Secures access and redirects to dashboards.
* **Dashboards:** Role-based UIs for managing appointments.
* **Doctor Dashboard**



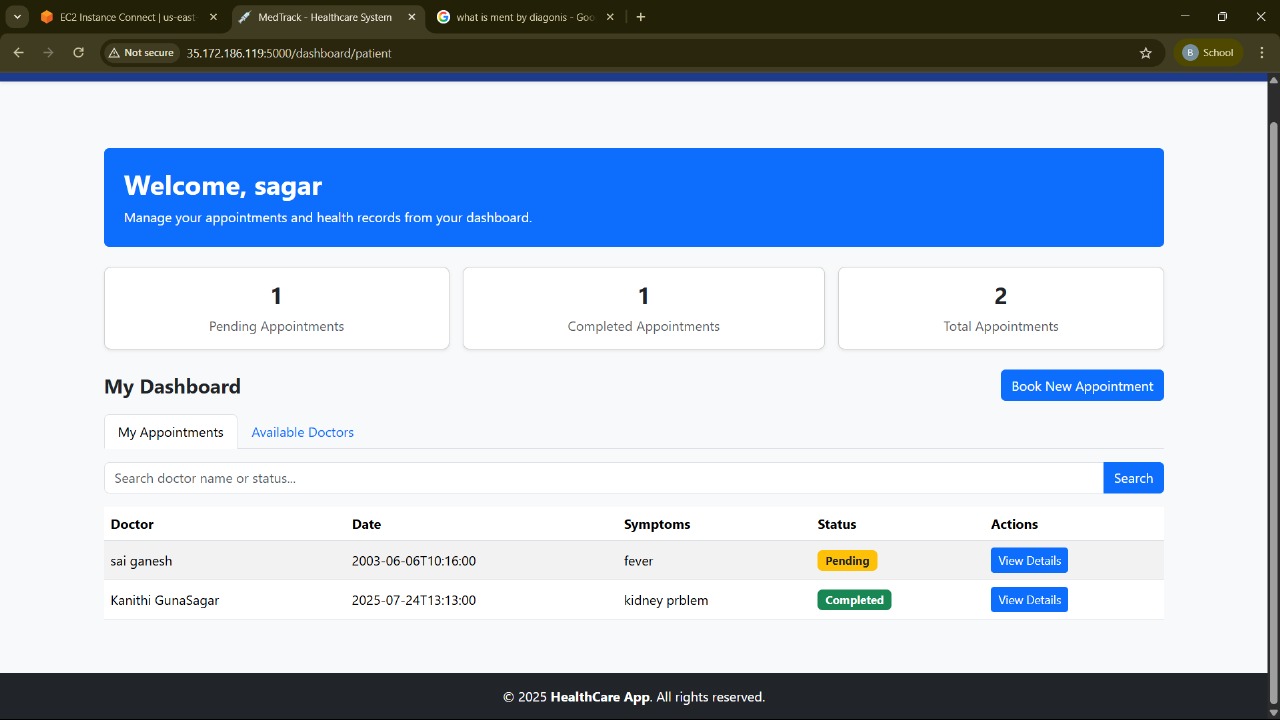
* **Patient Dashboard**

**Search Feature:** Enables real-time filtering by status/date.

### **Database Updates**

**Users Table:**

* Add a new user (doctor/patient)
* Update Profile
* Track active/inactive status.



### 

### **Conclusion**

MedTrack successfully demonstrates how cloud-native technologies can modernize healthcare delivery. Integrating Flask with AWS services such as EC2, DynamoDB, SNS, and IAM ensures secure, scalable, and responsive operations. MedTrack enhances patient care, optimizes appointment workflows, and supports reliable doctor-patient communication. This project stands as a powerful model of how technology can bridge operational gaps in real-world healthcare systems.