

## 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 enabling diagnosis submissions. To address these challenges, the project utilizes Flask for backend development, AWS EC2 for hosting, and DynamoDB for managing data. MedTrack allows patients to register, log in, book appointments, and submit diagnosis reports online. The system ensures real-time notifications, enhancing communication between doctors and patients regarding appointments and medical submissions. Additionally, AWS Identity and Access Management (IAM) is employed to ensure secure access control to AWS resources, allowing only authorized users to access sensitive data. This cloud-based solution improves accessibility and efficiency in healthcare services for all users.

### Scenario 1: Efficient Appointment Booking System for Patients

In the MedTrack system, AWS EC2 provides a reliable infrastructure to manage multiple patients accessing the platform simultaneously. For example, a patient can log in, navigate to the appointment booking page, and easily submit a request for an appointment. Flask handles backend operations, efficiently retrieving and processing user data in real-time. The cloud-based architecture allows the platform to handle a high volume of appointment requests during peak periods, ensuring smooth operation without delays.

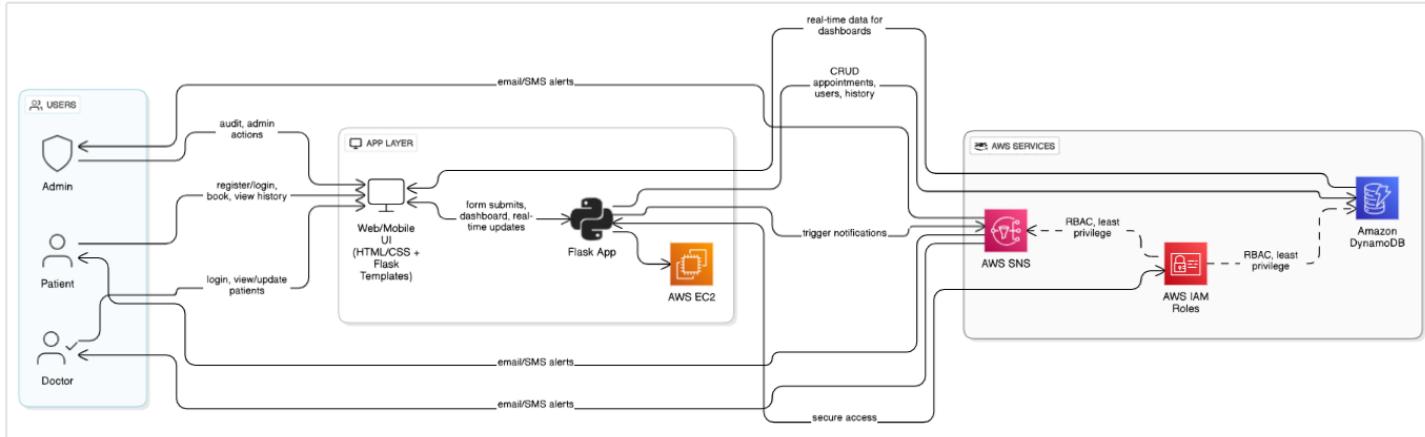
### Scenario 2: Secure User Management with IAM

MedTrack utilizes AWS IAM to manage user permissions and ensure secure access to the system. For instance, when a new patient registers, an IAM user is created with specific roles and permissions to access only the features relevant to them. Doctors have their own IAM configurations, allowing them access to patient records and appointment details while maintaining strict security protocols. This setup ensures that sensitive data is accessible only to authorized users.

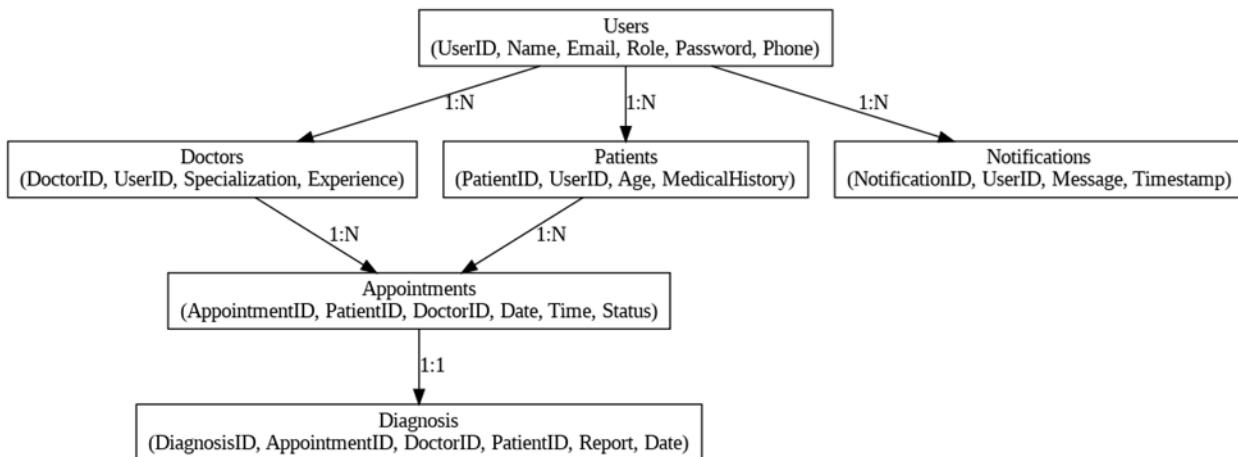
### Scenario 3: Easy Access to Medical History and Resources

The MedTrack system provides doctors and patients with easy access to medical histories and relevant resources. For example, a doctor logs in to view a patient's medical history and upcoming appointments. They can quickly access, and update records as needed. Flask manages real-time data fetching from DynamoDB, while EC2 hosting ensures the platform performs seamlessly even when multiple users access it simultaneously, offering a smooth and uninterrupted user experience.

## AWS ARCHITECTURE



## Entity Relationship (ER)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:**

### **1. AWS Account Setup 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

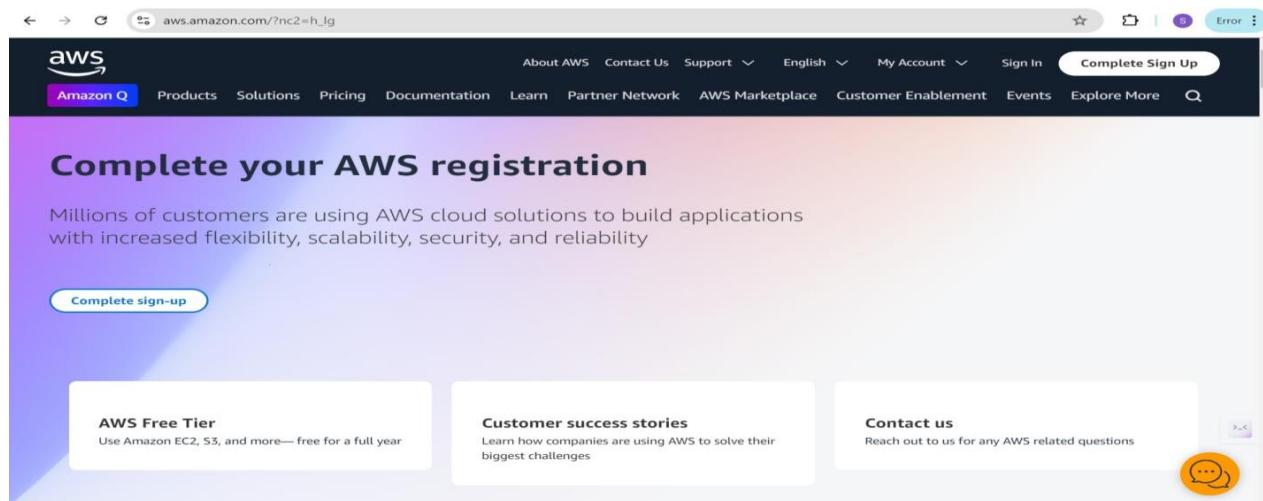
**Activity 7.2:** Run the Flask App

## 8. Testing and Deployment

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

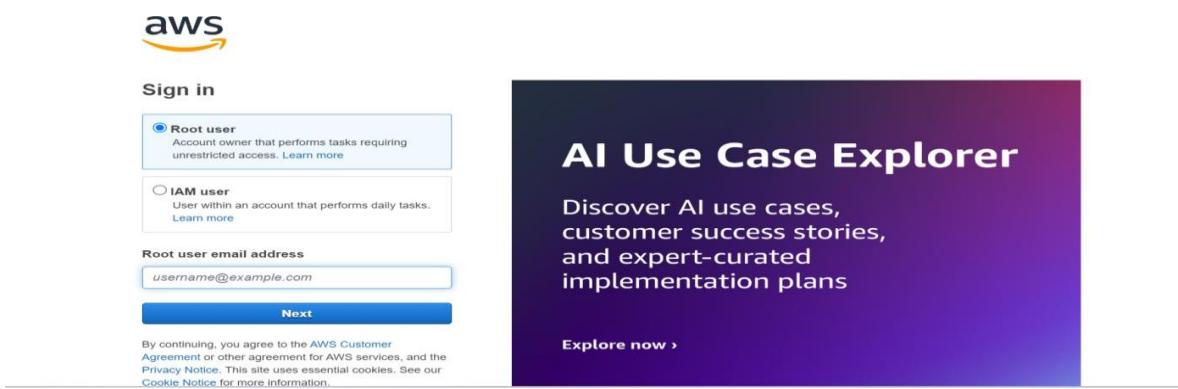
### Milestone 1: AWS Account Setup and Login

- **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](#).

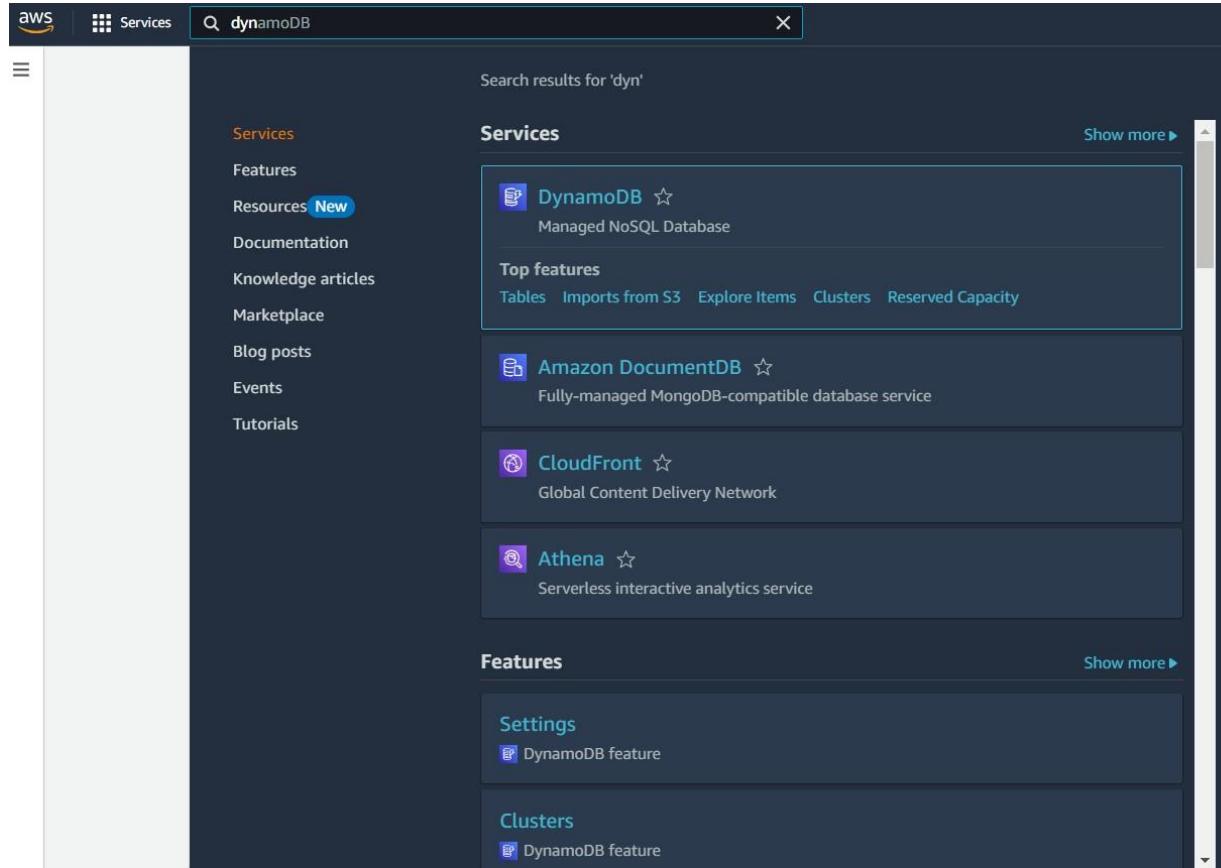


The screenshot shows the AWS sign-in interface. It features a 'Sign in' form with two radio button options: 'Root user' (selected) and 'IAM user'. Below the radio buttons is a field for 'Root user email address' containing 'username@example.com'. A 'Next' button is at the bottom of the form. To the right of the sign-in form is a dark sidebar with a purple-to-black gradient background. It has a title 'AI Use Case Explorer' and text 'Discover AI use cases, customer success stories, and expert-curated implementation plans'. At the bottom of the sidebar is a 'Explore now >' button.

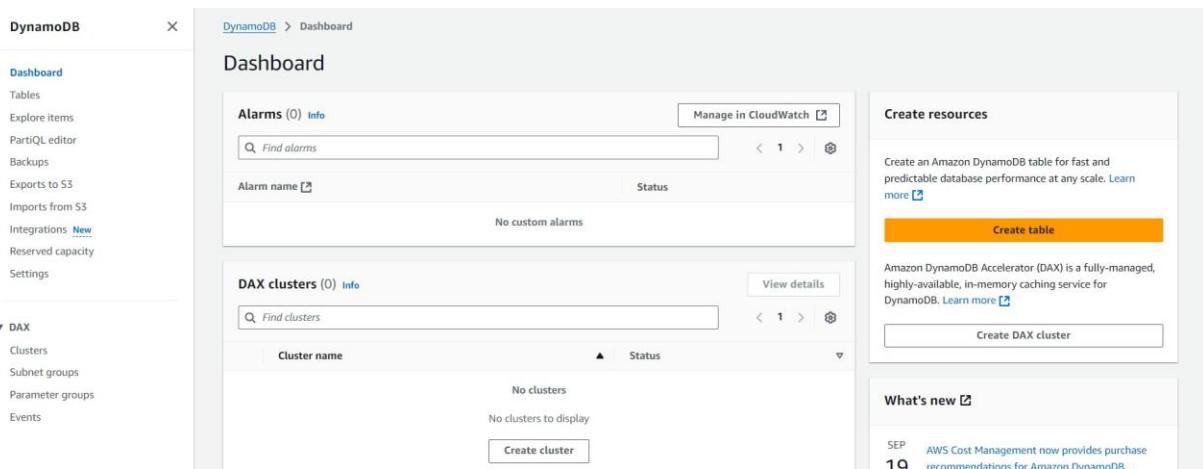
## Milestone 2: DynamoDB Database Creation and Setup

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

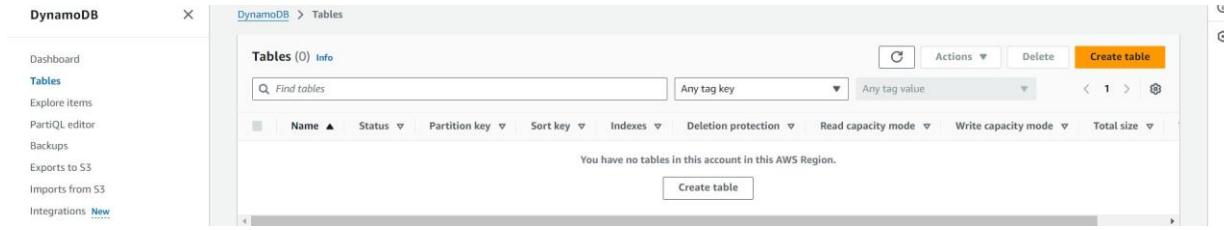
- In the AWS Console, navigate to DynamoDB and click on create tables.



The screenshot shows the AWS Services search results page. The search bar at the top contains 'dynamodb'. The left sidebar has links for Services, Features, Resources (New), Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main content area is titled 'Services' and shows a list of services. 'DynamoDB' is highlighted with a star icon and described as a 'Managed NoSQL Database'. Below it are 'Amazon DocumentDB' (Fully-managed MongoDB-compatible database service), 'CloudFront' (Global Content Delivery Network), and 'Athena' (Serverless interactive analytics service). A 'Show more ▶' link is at the top right of the services list. Below the services section is another 'Features' section with 'Settings' and 'Clusters' options, also with a 'Show more ▶' link.

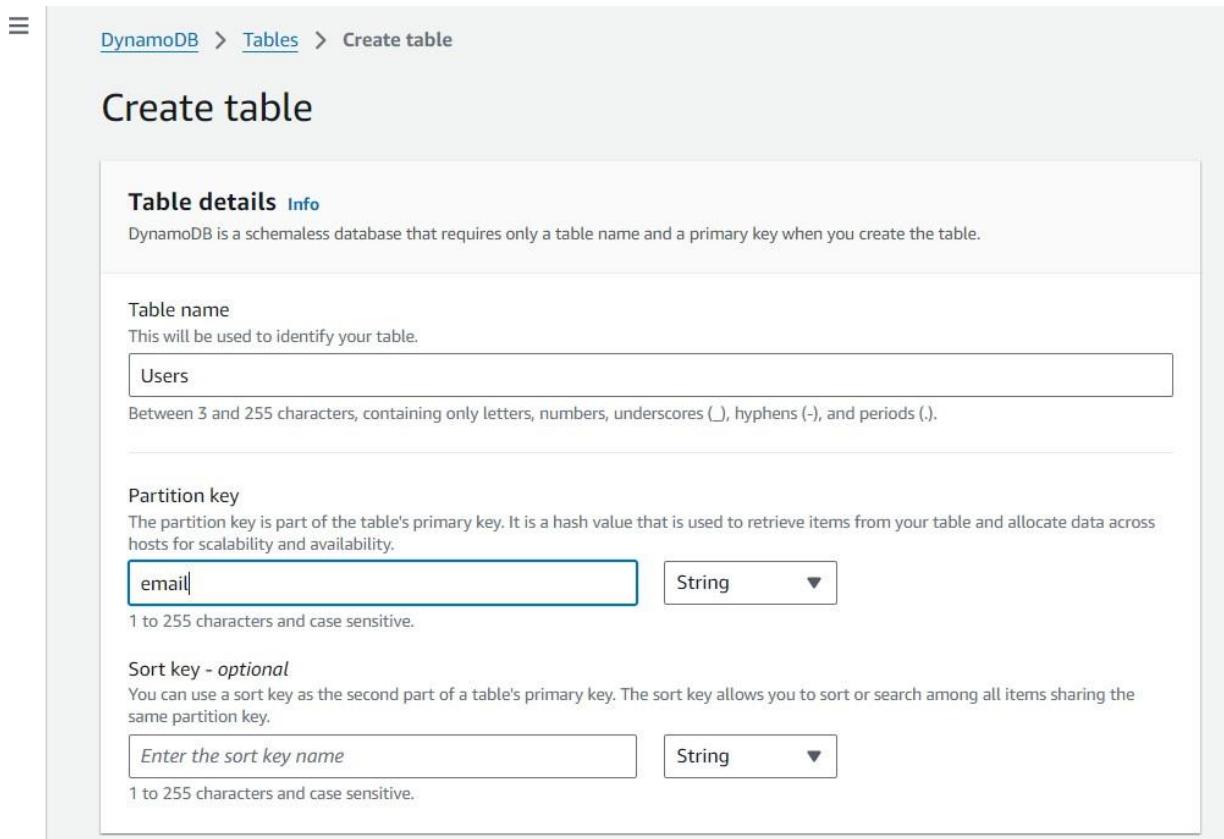


The screenshot shows the DynamoDB Dashboard. The left sidebar includes links for Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations (New), Reserved capacity, Settings, DAX (Clusters, Subnet groups, Parameter groups, Events), and a Create resources section. The main dashboard area has two sections: 'Alarms (0) Info' and 'DAX clusters (0) Info'. Both sections have search bars, status indicators, and buttons for 'View details' and 'Create cluster'. To the right is a 'Create resources' section for creating a new DynamoDB table, a 'What's new' section with a note about AWS Cost Management, and a news feed for September 1st.



- **Activity 2.2: Create a DynamoDB table for storing data**

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



**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 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 Users table was created successfully.

Tables (1) <a href="#">Info</a>										
	Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size	
<input type="checkbox"/>	Users	Active	email (\$)	-	0	Off	Provisioned (5)	Provisioned (5)	0 bytes	

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

aws |  Search [Alt+S] |    

DynamoDB > Tables > Create table

## 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.

AppointmentsTable

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.

appointment\_id

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.

Enter the sort key name

String ▾

1 to 255 characters and case sensitive.

DynamoDB > Tables

**DynamoDB**

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

**DAX**

- Clusters
- Subnet groups
- Parameter groups

CloudShell   [Feedback](#)

**Tables (7) [Info](#)**

<input type="checkbox"/>	Name	Status	Partition key	Sort key	Indexes	Replication Regions	Del
<input type="checkbox"/>	<a href="#">MedTrackAppointments</a>	<span>Active</span>	appointment_id (S)	-	0	0	
<input type="checkbox"/>	<a href="#">MedTrackDiagnosis</a>	<span>Active</span>	diagnosis_id (S)	-	0	0	
<input type="checkbox"/>	<a href="#">MedTrackDoctors</a>	<span>Active</span>	email (S)	-	0	0	
<input type="checkbox"/>	<a href="#">MedTrackNotifications</a>	<span>Active</span>	notification_id (S)	-	0	0	
<input type="checkbox"/>	<a href="#">MedTrackPatients</a>	<span>Active</span>	email (S)	-	0	0	
<input type="checkbox"/>	<a href="#">MedTrackUsers</a>	<span>Active</span>	email (S)	-	0	0	
<input type="checkbox"/>	<a href="#">MedTrackUsers1</a>	<span>Active</span>	email (S)	-	0	0	

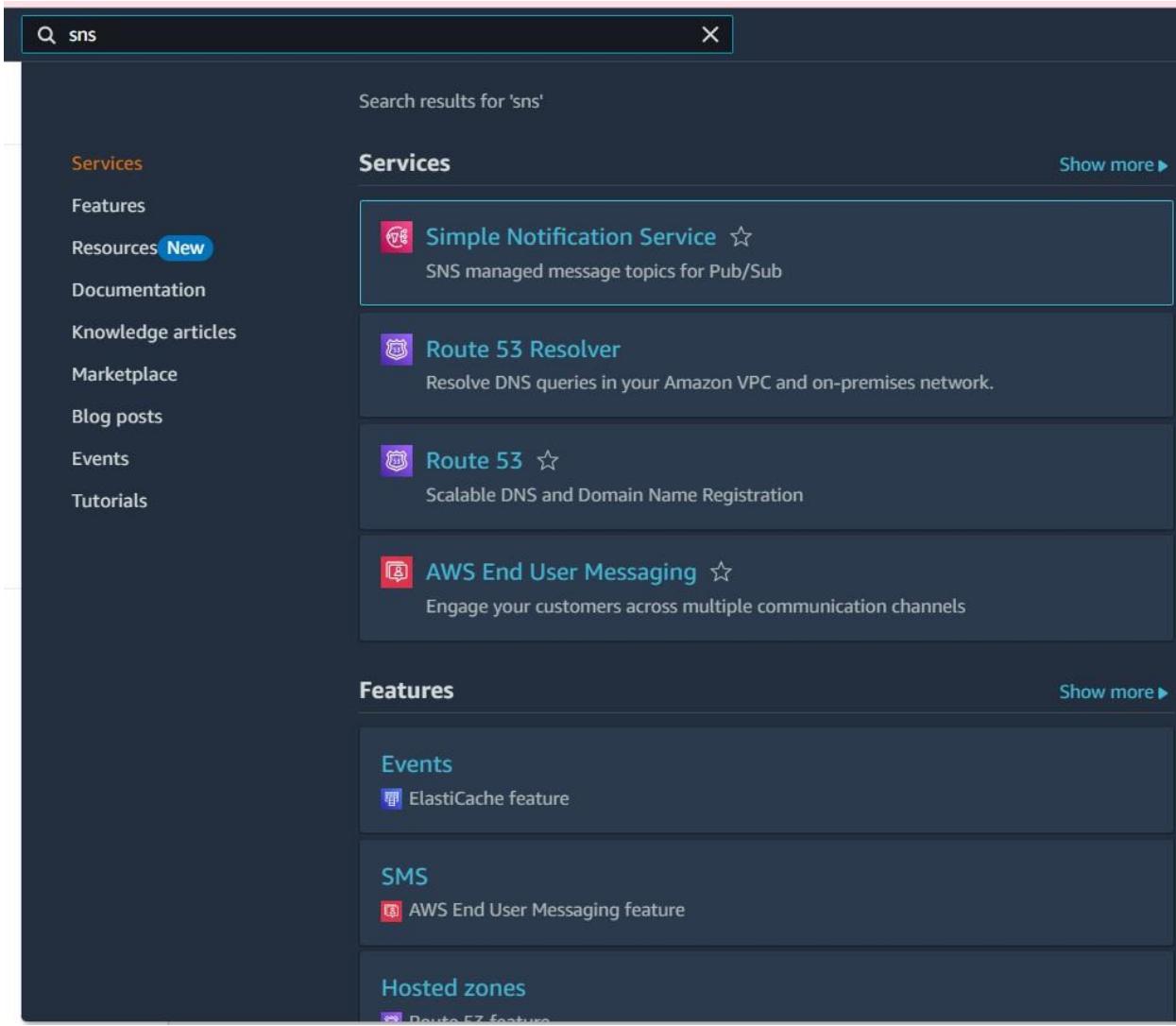
Activate Windows  
Go to Settings to activate Windows.

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

## Milestone 3: SNS Notification Setup

- **Activity 3.1: Create SNS topics for sending email notifications to users and Admin.**

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



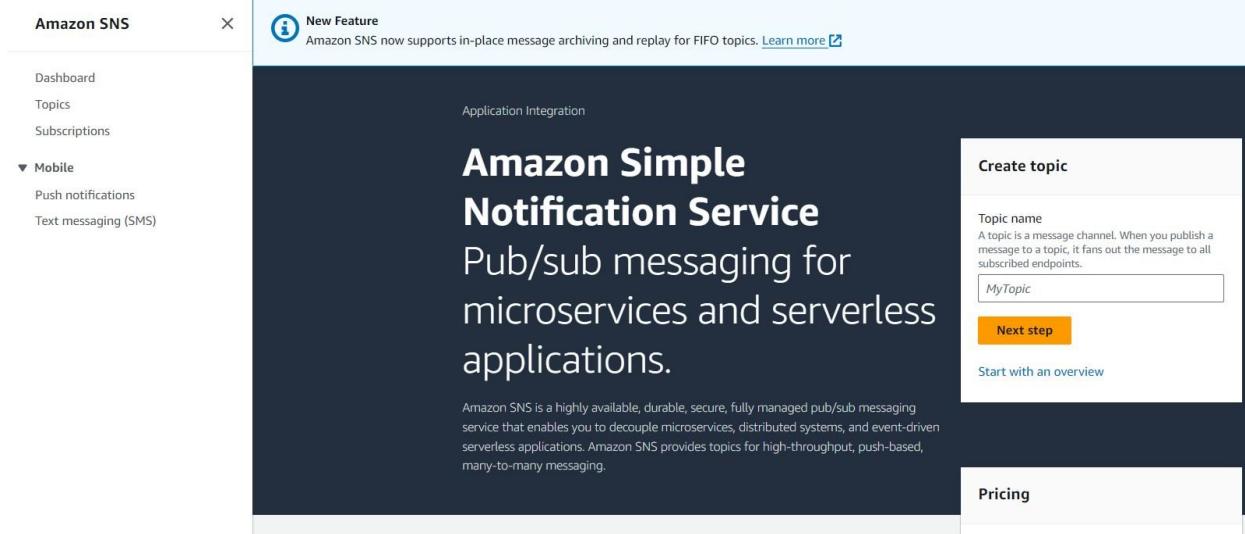
The screenshot shows the AWS search interface with the search term 'sns' entered. The results are categorized under 'Services' and 'Features'.

**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

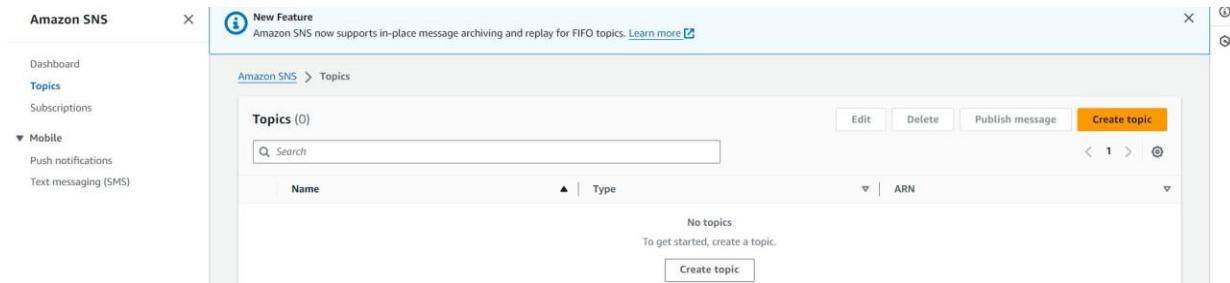
**Features**

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



The screenshot shows the 'Amazon Simple Notification Service' landing page. On the left, there's a sidebar with links like Dashboard, Topics, Subscriptions, Mobile (Push notifications, Text messaging (SMS)), and Application Integration. The main content area features a large heading 'Amazon Simple Notification Service' and a sub-section 'Pub/sub messaging for microservices and serverless applications.' Below this is a detailed description of Amazon SNS. A prominent 'Create topic' button is located on the right side of the main content area.

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



The screenshot shows the 'Topics' section of the Amazon SNS console. The left sidebar remains the same. The main area displays a table titled 'Topics (0)' with columns 'Name', 'Type', and 'ARN'. There is a note 'No topics' and a call-to-action 'To get started, create a topic.' with a 'Create topic' button. Navigation buttons for 'Edit', 'Delete', 'Publish message', and 'Create topic' are also visible.

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

Screenshot of the AWS SNS Create topic page.

The URL in the browser bar is: us-east-1.console.aws.amazon.com/sns/v3/home?region=us-east-1#/create-topic

The navigation path is: Amazon SNS > Topics > 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  
• Subscription protocols: SQS

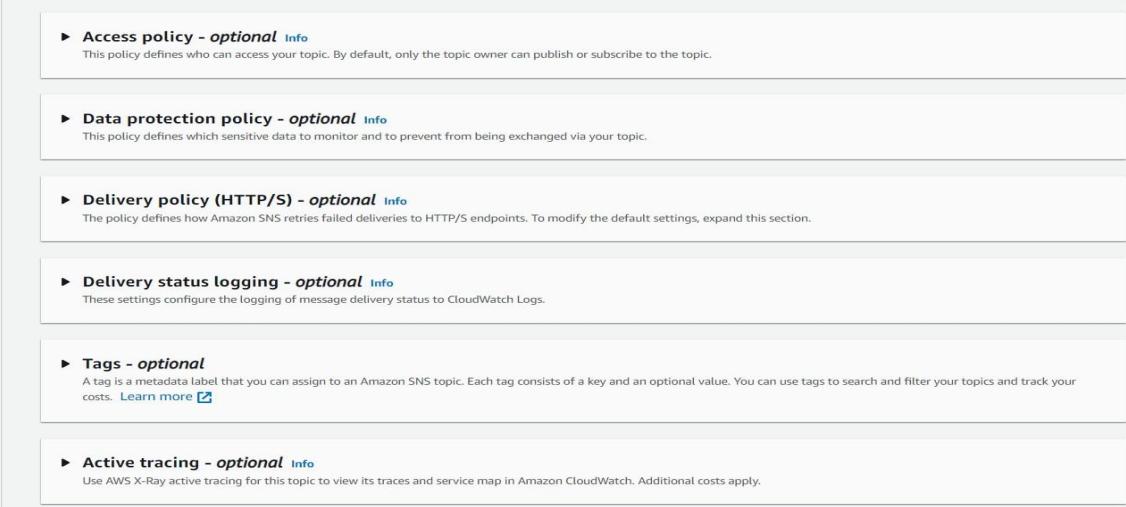
Standard  
• Best-effort message ordering  
• At-least once message delivery  
• Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SMS, email, mobile application endpoints

**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.

Activate Windows  
Go to Settings to activate Windows.

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

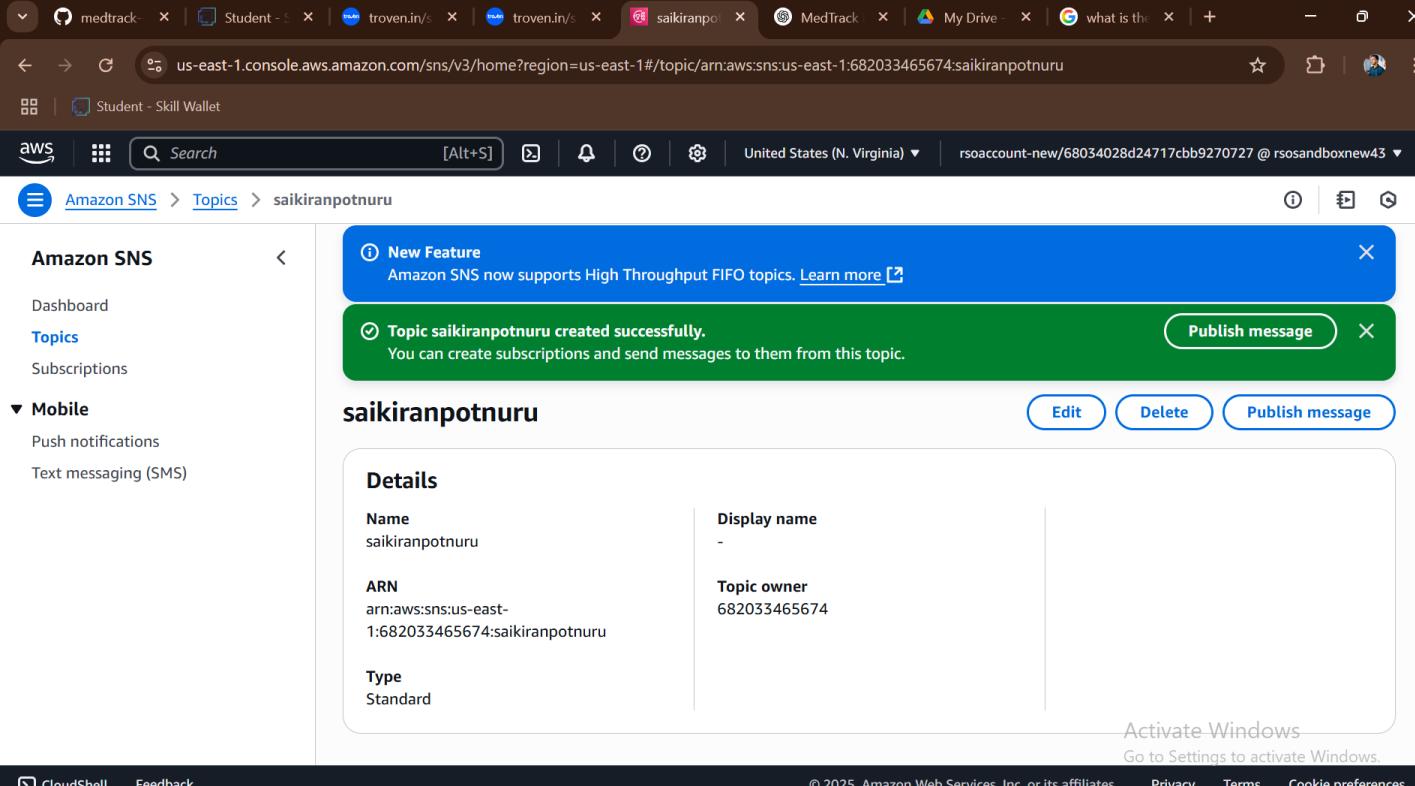


The screenshot shows the 'Create topic' wizard with several optional configuration sections:

- Access policy - optional**: 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**: This policy defines which sensitive data to monitor and to prevent from being exchanged via your topic.
- Delivery policy (HTTP/S) - optional**: 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**: 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**: Use AWS X-Ray active tracing for this topic to view its traces and service map in Amazon CloudWatch. Additional costs apply.

At the bottom right are 'Cancel' and 'Create topic' buttons.

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



The screenshot shows the AWS SNS Topics page with the following details for the topic 'saikiranpotnuru':

- Name:** saikiranpotnuru
- ARN:** arn:aws:sns:us-east-1:682033465674:saikiranpotnuru
- Type:** Standard
- Topic owner:** 682033465674

A green success message at the top right states: "Topic saikiranpotnuru created successfully. You can create subscriptions and send messages to them from this topic." There are 'Edit', 'Delete', and 'Publish message' buttons available for the topic.

At the bottom right, there is a message: "Activate Windows Go to Settings to activate Windows."

Page footer: CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- **Activity 3.2: Subscribe users and Admin.**
- Subscribe users to this topic via email. When an Appointment is made, notifications will be sent to the corresponding doctor or patient emails.

## Create subscription

### Details

#### Topic ARN

 arn:aws:sns:us-east-1:682033465674:saikiranpotnuru

#### Protocol

The type of endpoint to subscribe

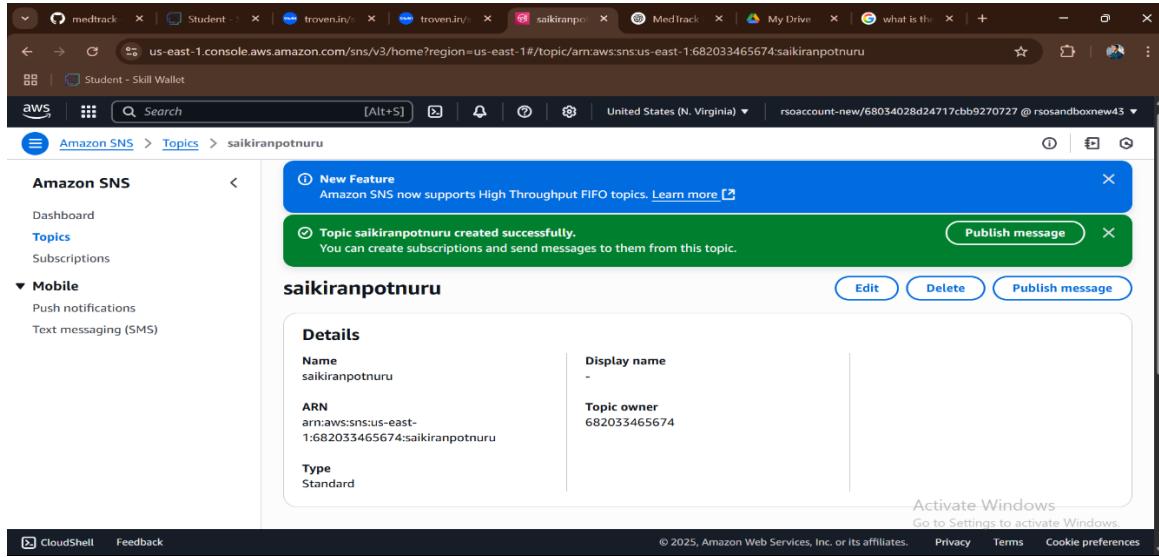
 Email

#### Endpoint

An email address that can receive notifications from Amazon SNS.

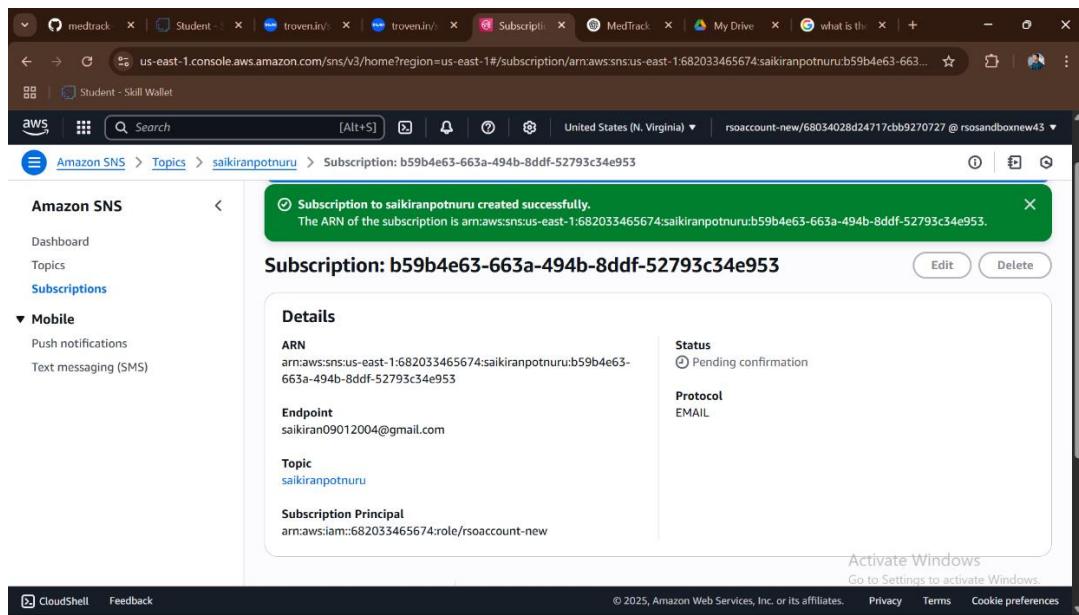
 saikiran09012004@gmail.com

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



The screenshot shows the AWS SNS console with the URL <https://us-east-1.console.aws.amazon.com/sns/v3/home?region=us-east-1#/topic/arm:aws:sns:us-east-1:682033465674:saikiranpotnuru>. A blue banner at the top indicates a 'New Feature' where Amazon SNS now supports High Throughput FIFO topics. Below it, a green banner says 'Topic saikiranpotnuru created successfully.' The main panel shows the topic details: Name (saikiranpotnuru), ARN (arn:aws:sns:us-east-1:682033465674:saikiranpotnuru), and Type (Standard). Buttons for Edit, Delete, and Publish message are visible.

- After subscription request for the mail confirmation



The screenshot shows the AWS SNS console with the URL <https://us-east-1.console.aws.amazon.com/sns/v3/home?region=us-east-1#/subscription/arm:aws:sns:us-east-1:682033465674:saikiranpotnuru:b59b4e63-663...>. A green banner at the top says 'Subscription to saikiranpotnuru created successfully.' The main panel shows the subscription details: ARN (arn:aws:sns:us-east-1:682033465674:saikiranpotnuru:b59b4e63-663a-494b-8ddf-52793c34e953), Endpoint (saikiran09012004@gmail.com), Topic (saikiranpotnuru), and Subscription Principal (arn:aws:iam::682033465674:role/rsoaccount-new). Buttons for Edit and Delete are visible.

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

AWS Notification - Subscription Confirmation Spam ×

AWS Notifications &lt;no-reply@sns.amazonaws.com&gt;

to me ▾

Fri 4 Jul, 12:22 (5 days ago)



Why is this message in spam? This message 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:us-east-1:682033465674:saikiranpotnuru**

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

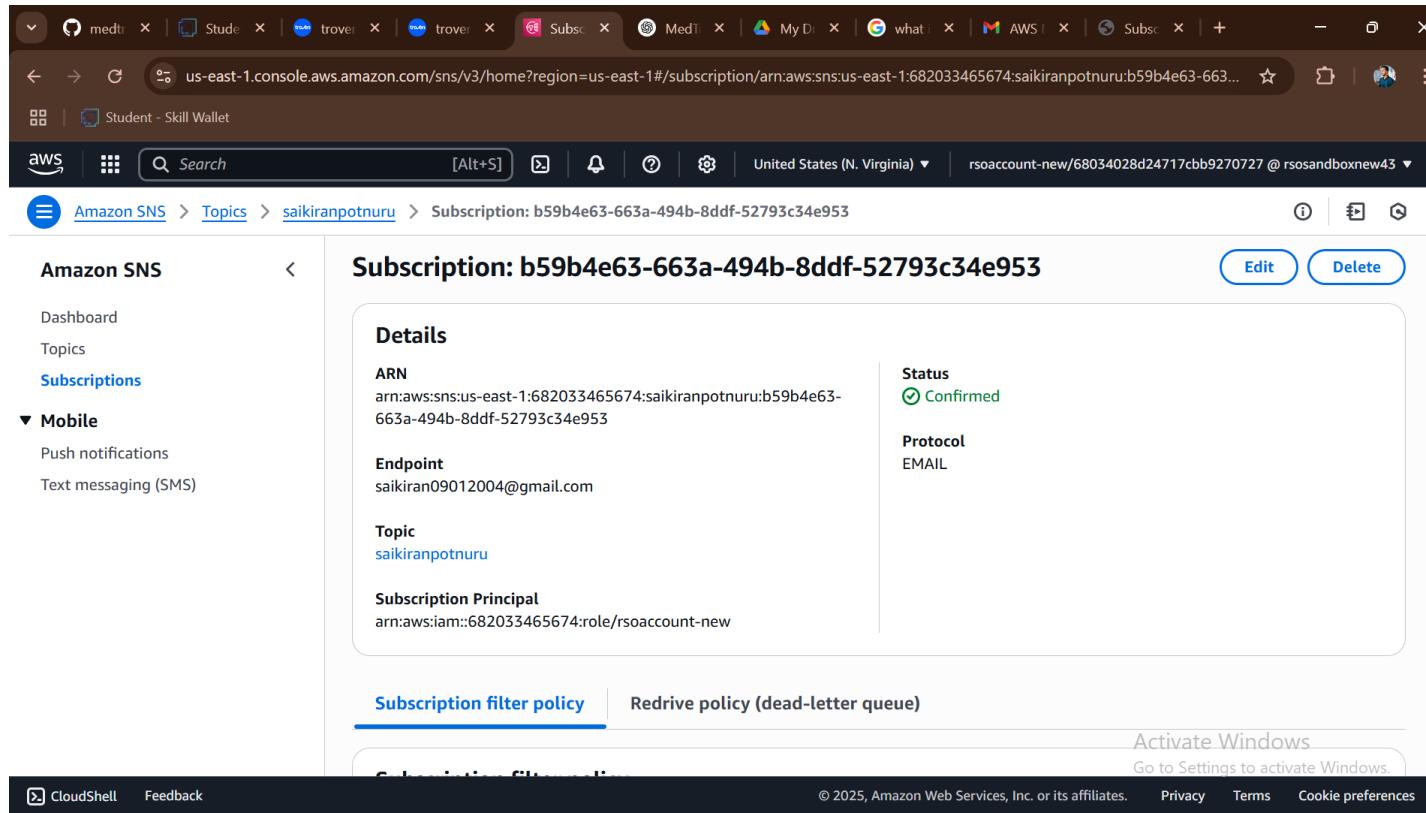
**Subscription confirmed!**

You have successfully subscribed.

Your subscription's id is:

**arn:aws:sns:us-east-1:682033465674:saikiranpotnuru:b59b4e63-663a-494b-8ddf-52793c34e953**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.



The screenshot shows the AWS SNS console with a subscription details page. The URL in the address bar is `us-east-1.console.aws.amazon.com/sns/v3/home?region=us-east-1#/subscription/arm:aws:sns:us-east-1:682033465674:saikiranpotnuru:b59b4e63-663a-494b-8ddf-52793c34e953`. The left sidebar shows navigation links for Amazon SNS, Dashboard, Topics, Subscriptions, Mobile (Push notifications, Text messaging (SMS)), and CloudWatch Metrics. The main content area is titled "Subscription: b59b4e63-663a-494b-8ddf-52793c34e953". It displays the following details:

- ARN:** arn:aws:sns:us-east-1:682033465674:saikiranpotnuru:b59b4e63-663a-494b-8ddf-52793c34e953
- Status:** Confirmed
- Endpoint:** saikiran09012004@gmail.com
- Protocol:** EMAIL
- Topic:** saikiranpotnuru
- Subscription Principal:** arn:aws:iam::682033465674:role/rsoaccount-new

Below the details, there are tabs for "Subscription filter policy" (which is selected) and "Redrive policy (dead-letter queue)". At the bottom of the page, there are links for CloudShell, Feedback, and various AWS terms like Activate Windows, Go to Settings to activate Windows, © 2025, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

## Milestone 4: Backend Development and Application Setup

- **Activity 4.1: Develop the backend using Flask**
  - File Explorer Structure

```
▽ MEDTRACK
  ▽ static
  ▽ templates
    ◄> about.html
    ◄> base.html
    ◄> book_appointment.html
    ◄> doctor_dashboard.html
    ◄> index.html
    ◄> login.html
    ◄> patient_dashboard.html
    ◄> search_results.html
    ◄> signup.html
    ◄> view_appointment_doctor.html
    ◄> view_appointment_patient.html
    ◄> view_appointment.html
  .env
  app.py
  • from flask import Flask, request, js...
```

**Description:** Backend Development and Application Setup focuses on establishing the core structure of the application. This includes configuring the backend framework, setting up routing, and integrating database connectivity. It lays the groundwork for handling user interactions, data management, and secure access.

### Description of the code :

- **Flask App Initialization**

```
from flask import Flask, render_template, request, redirect, url_for
import boto3
from boto3.dynamodb.conditions import Key
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from bcrypt import hashpw, gensalt, checkpw
```

**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 Table Names aligned with ER diagram
USERS_TABLE_NAME = os.environ.get('USERS_TABLE_NAME', 'MedTrackUsers')
DOCTORS_TABLE_NAME = os.environ.get('DOCTORS_TABLE_NAME', 'MedTrackDoctors')
PATIENTS_TABLE_NAME = os.environ.get('PATIENTS_TABLE_NAME', 'MedTrackPatients')
APPOINTMENTS_TABLE_NAME = os.environ.get('APPOINTMENTS_TABLE_NAME', 'MedTrackAppointments')
DIAGNOSIS_TABLE_NAME = os.environ.get('DIAGNOSIS_TABLE_NAME', 'MedTrackDiagnosis')
NOTIFICATIONS_TABLE_NAME = os.environ.get('NOTIFICATIONS_TABLE_NAME', 'MedTrackNotifications')
```

**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**

```
def send_email_notification(to_email, subject, body):
    if not ENABLE_EMAIL or not SENDER_EMAIL:
        logger.info(f"Email notification would be sent: {subject}")
        return True

    try:
        msg = MIMEText(body)
        msg['From'] = SENDER_EMAIL
        msg['To'] = to_email
        msg['Subject'] = subject
        msg.attach(MIMEText(body, 'plain'))

        server = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
        server.starttls()
        server.login(SENDER_EMAIL, SENDER_PASSWORD)
        server.send_message(msg)
        server.quit()

        logger.info(f"Email sent successfully to {to_email}")
        return True
    except Exception as e:
        logger.error(f"Failed to send email: {e}")
        return False
```

**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**

- **Index Route:**

```
# Route: Select role
@app.route('/')
def index():
    return render_template('index.html')
```

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

- **Signu Route:**

```

210     @app.route('/signup/<role>', methods=['GET', 'POST'])
211     def signup(role):
212         if role not in ('patient', 'doctor'):
213             flash('Invalid role selected.', 'danger')
214             return redirect(url_for('index'))
215
216         if request.method == 'POST':
217             name = request.form.get('name')
218             email = request.form.get('email')
219             password = request.form.get('password')
220
221             if not name or not email or not password:
222                 flash('All fields are required.', 'warning')
223                 return render_template('signup.html', role=role)
224
225             # Check if user already exists
226             if dynamodb:
227                 user_table = get_users_table()
228                 response = user_table.get_item(Key={'email': email})
229                 if 'Item' in response:
230                     flash('User already exists.', 'danger')
231                     return render_template('signup.html', role=role)
232                 else:
233                     #if email in local_db['users']:
234                         #flash('User already exists.', 'danger')
235                         #return render_template('signup.html', role=role)
236                     flash('User already exists.', 'danger')
237                     return render_template('signup.html', role=role)
238
239             # Create user
240             user_id = str(uuid.uuid4())
241             hashed_password = generate_password_hash(password)
242             user_data = {
243                 'user_id': user_id,
244                 'name': name,
245                 'email': email,
246                 'password_hash': hashed_password,
247                 'role': role,
248                 'created_at': datetime.now().isoformat(),
249                 'is_active': True
250             }
251
252             if dynamodb:
253                 user_table.put_item(Item=user_data)
254             else:
255                 #local_db['users'][email] = user_data
256                 flash('DynamoDB is required for production. Please enable it.', 'danger')
257                 return render_template('signup.html', role=role)
258
259
260             flash('Signup successful! Please log in.', 'success')
261             return redirect(url_for('login', role=role))
262
263             return render_template('signup.html', role=role)

```

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

- **login Route (GET/POST):**

```

266     login_attempts = {} # Rate-limiting support
267     @app.route('/login<role>', methods=['GET', 'POST'])
268     def login(role):
269         if role not in ('patient', 'doctor'):
270             flash("Invalid role.", "danger")
271             return redirect(url_for('index'))
272
273         if request.method == 'POST':
274             email = request.form.get('email')
275             password = request.form.get('password')
276
277             if not email or not password:
278                 flash('Email and password are required.', 'warning')
279                 return render_template('login.html', role=role)
280
281             # Rate-limiting check
282             client_ip = request.remote_addr
283             if client_ip in login_attempts:
284                 attempt_data = login_attempts[client_ip]
285                 if attempt_data['count'] >= 5:
286                     if datetime.now() - attempt_data['last_attempt'] < timedelta(minutes=15):
287                         flash('Too many login attempts. Try again later.', 'danger')
288                         return render_template('login.html', role=role)
289                     else:
290                         login_attempts[client_ip] = {'count': 0, 'last_attempt': datetime.now()}
291
292             # Fetch user data
293             user_data = None
294             if dynamodb:
295                 user_table = get_users_table()
296                 try:

```

**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

- **doctor and patient dashboards routes:**

```

""" def get_patient_dashboard_data(user_email): ...
def get_patient_dashboard_data(user_email): ...

"""def get_doctor_dashboard_data(user_email): ...

def get_doctor_dashboard_data(user_email): ...

@app.route('/patient_dashboard')
@login_required(role='patient')
def patient_dashboard(): ...

@app.route('/doctor_dashboard')
@login_required(role='doctor')
def doctor_dashboard(): ...
# ...existing code...

```

**Description:** define /home-page to render the main homepage, /ebook-buttons to handle subject selection and redirection, and /<subject>.html dynamic route to render subject-specific pages like Mathematics or English.

- Book Appointment Routes:

```
def book_appointment():
    if request.method == 'POST':
        doctor_email = request.form['doctor']
        date = request.form['date']
        time = request.form['time']
        title = request.form.get('title', 'Consultation')
        location = request.form.get('location', 'Office 203')
        color = request.form.get('color', '#3498db')
        if dynamodb:
            user_table = get_users_table()
            appointments_table = get_appointments_table()
            doctor_resp = user_table.get_item(Key={'email': doctor_email})
            doctor_name = doctor_resp.get('Item', {}).get('name', 'Doctor')
            patient_email = session['user']
            patient_resp = user_table.get_item(Key={'email': patient_email})
            patient_name = patient_resp.get('Item', {}).get('name', 'Patient')
            appointment_id = str(uuid.uuid4())
            appointment = {
                'appointment_id': appointment_id,
                'patient': patient_email,
                'patient_name': patient_name,
                'doctor': doctor_email,
                'doctor_name': doctor_name,
                'title': title,
                'date': date,
                'time': time,
                'location': location,
                'color': color
            }
            appointments_table.put_item(Item=appointment)
            flash('Appointment booked successfully!', 'success')
```

⑧ 0 △ 0

ϕ saikiranpo

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

### Logout Route:

```
@app.route('/api/logout', methods=['POST'])
@login_required(api=True) # Ensures API-style authentication response
def logout():
    try:
        user_email = session.get('email')
        user_name = session.get('name')

        # Clear session
        session.clear()

        logger.info(f"User logged out: {user_email}")

        # Optionally send notification (email or SNS)
        message = f"{user_name} has logged out from MediTrack."
        send sns_notification(message)
        # Optionally: send_email_notification(user_email, "Logout Alert", message)

        return jsonify({'message': 'Logged out successfully'}), 200

    except Exception as e:
        logger.error(f"Logout failed: {e}")
        return jsonify({'error': 'Logout failed'}), 500
```

**Description:** define /Logout route to render the exit.html page when the user chooses to leave or close the application.

### Deployment Code:

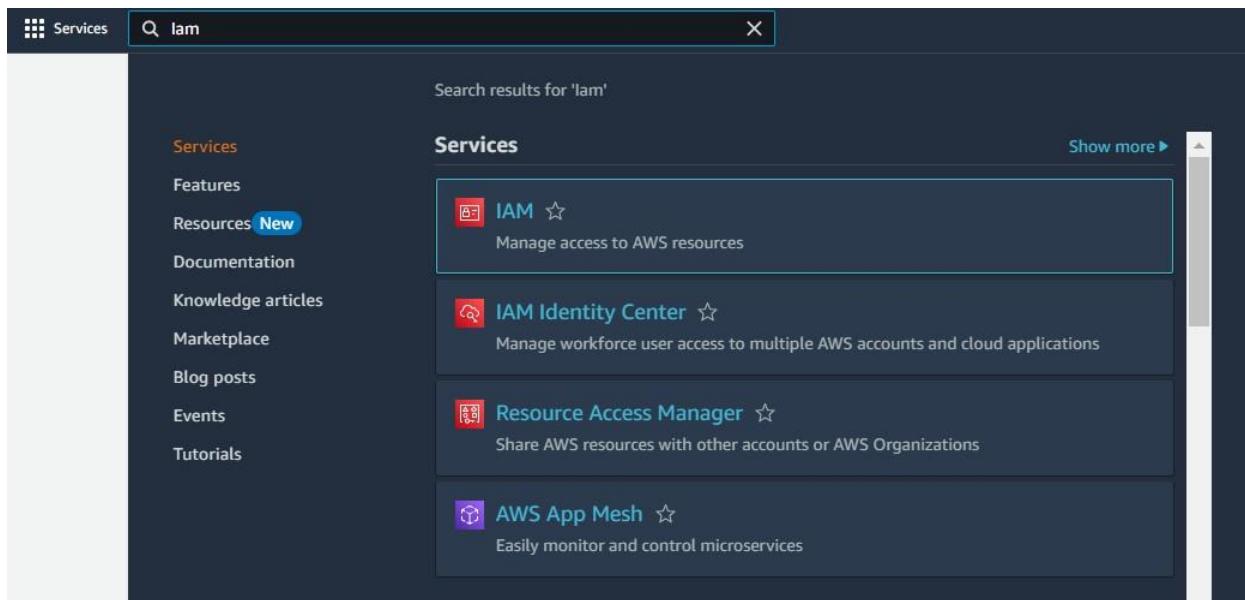
```
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=True)
```

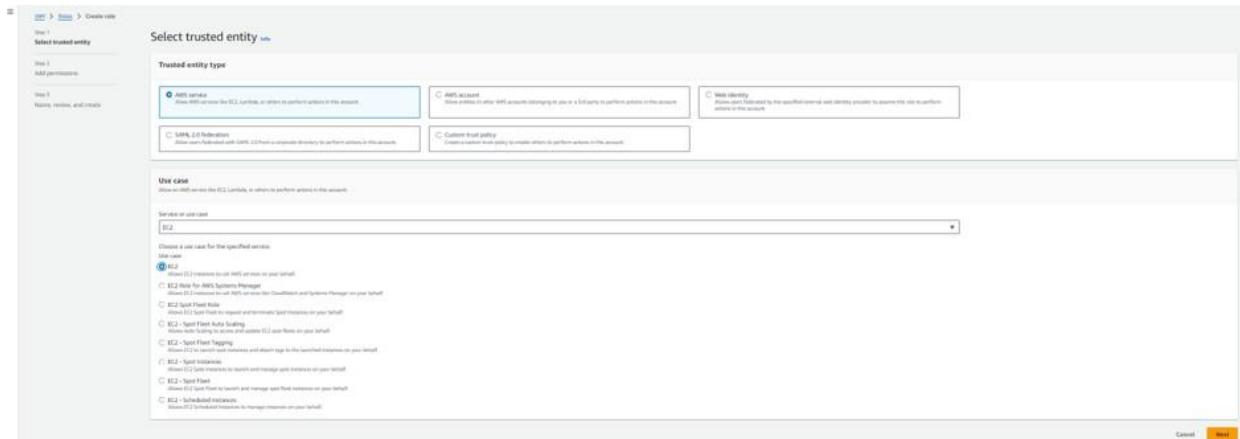
**Description:** start the Flask server to listen on all network interfaces (0 . 0 . 0 . 0) at port 80 with debug mode enabled for development and testing.

## 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.

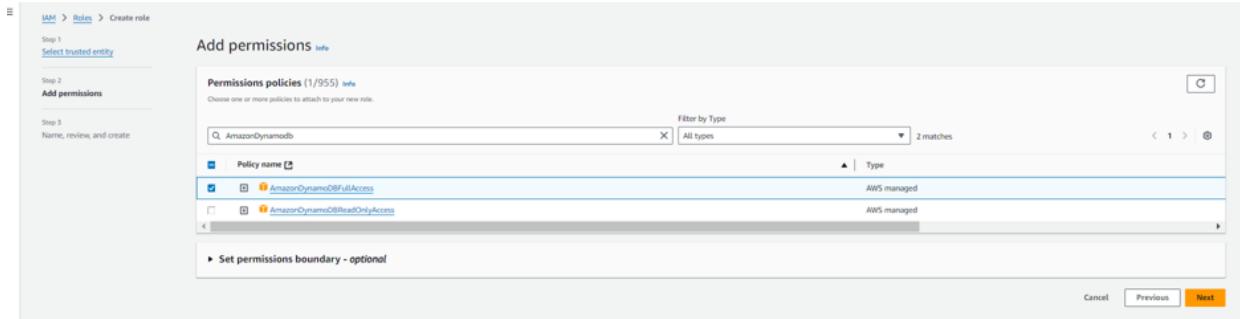




The screenshot shows the 'Select trusted entity' step of the 'Create role' wizard. It lists four options:

- AWS service**: Allows the specified AWS service to assume the role to perform actions in this account.
- AWS account**: Allows another AWS account to assume the role to perform actions in this account.
- IAM identity**: Allows users or roles in this account to assume the role to perform actions in this account.
- Custom trust policy**: Creates a custom JSON policy to define who can assume the role and what actions they can perform.

Below this, there is a 'User case' section with a dropdown set to 'EC2'. A list of EC2 use cases is provided, with 'EC2' (allow EC2 to assume to call AWS services on your behalf) selected. Other options include 'EC2 Role for AWS Systems Manager' (allow EC2 to assume to call AWS services from CloudWatch and Systems Manager on your behalf), 'EC2 Fleet Fleet Management' (allow EC2 Fleet to request or terminate 'Stand instances on your behalf'), 'EC2 - Spot Fleet Auto Scaling' (allow EC2 to search spot instances and adjust tags to the specified instances on your behalf), 'EC2 - Spot Fleet Tagging' (allow EC2 to search spot instances and adjust tags to the specified instances on your behalf), 'EC2 - SNS' (allow EC2 to publish messages to topics and manage subscriptions on your behalf), 'EC2 - SNS Fleet' (allow EC2 to publish messages to topics and manage subscriptions on your behalf), and 'EC2 - Scheduled Instances' (allow EC2 to scheduled instances to manage resources on your behalf).



The screenshot shows the 'Add permissions' step of the 'Create role' wizard. It displays a search bar for 'Permissions policies (1,955)' and a list of results. Two policies are listed: 'AmazonDynamoDBFullAccess' and 'AmazonDynamoDBReadOnlyAccess'. Both are described as 'AWS managed' policies.

## ● 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.

IAM > Roles > Create role

Step 1 Select trusted entity

### Add permissions Info

Permissions policies (2/955) Info

Choose one or more policies to attach to your new role.

Filter by Type  All types X 5 matches

Policy name	Type
<input checked="" type="checkbox"/>  AmazonSNSFullAccess	AWS managed
<input type="checkbox"/>  AmazonMSKFullAccess	AWS managed
<input type="checkbox"/>  AmazonMSISRole	AWS managed
<input type="checkbox"/>  AWSLambdaBasicExecutionRole	AWS managed
<input type="checkbox"/>  AWSIoTDeviceDefenderFullAuditPolicyExecutionRole	AWS managed

Set permissions boundary optional

Cancel Previous Next

Step 1: Select trusted entities

Name, review, and create

Role details

Role name

Description

Address EC2 instances for your AWS services on your behalf.

Maximum 1000 characters. Use letters A-Z and a-z, numbers 0-9, underscores, and hyphens.

Step 2: Add permission

Trust policy

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

Step 3: Add tags

Add tags - optional Info

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Cancel Previous Next

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 All types

Policy name	Type	Attached entities
<a href="#">AmazonDynamoDBFullAccess</a>	AWS managed	4
<a href="#">AmazonSNSFullAccess</a>	AWS managed	2

**C Simulate Remove Add permissions**

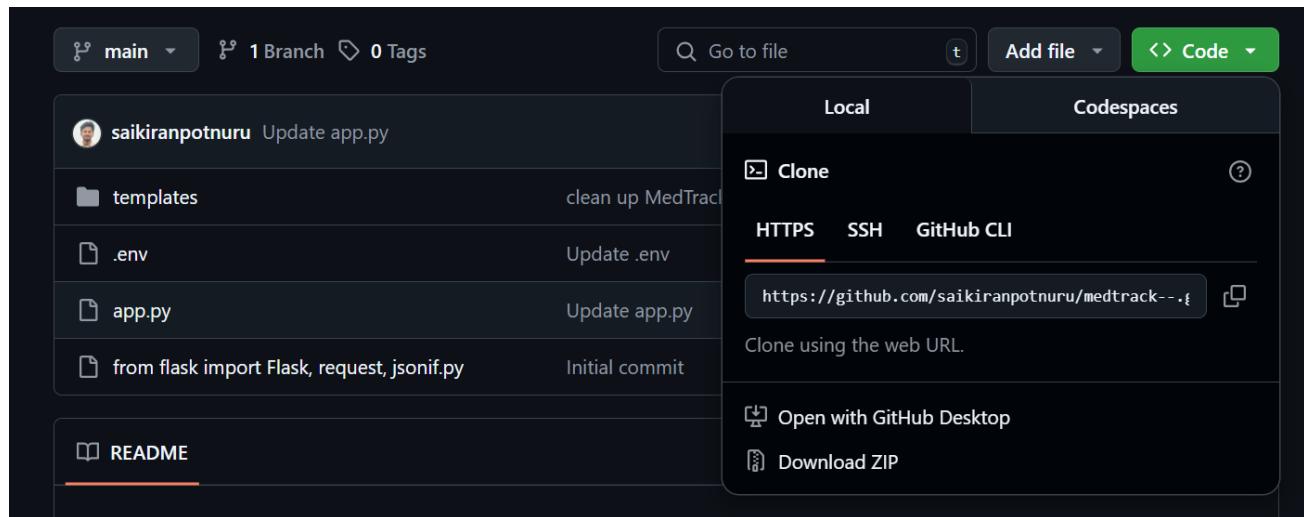
## Milestone 6: EC2 Instance Setup

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

---

	static	Initial commit
	templates	Update statistics.html
	app.py	Update app.py

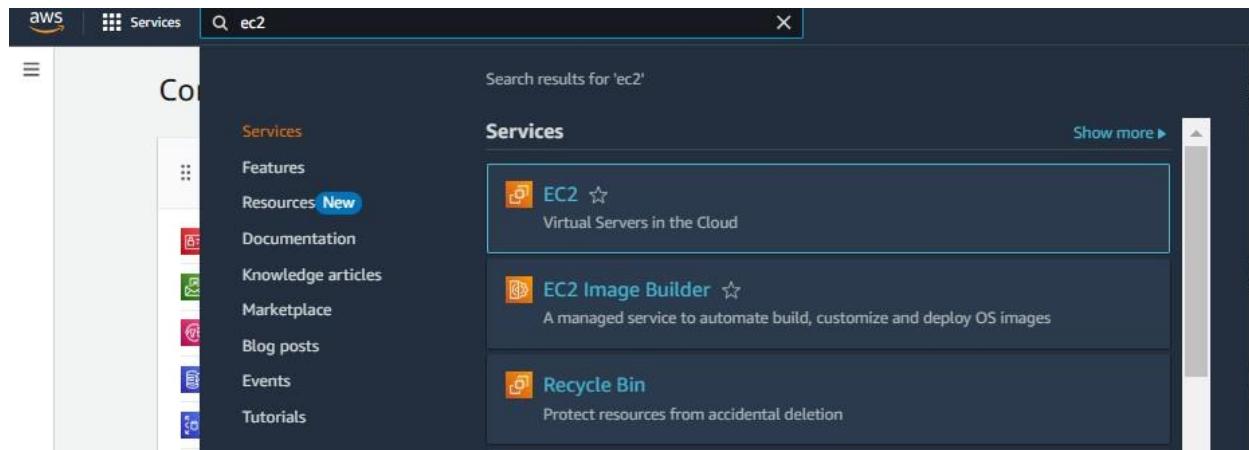
---



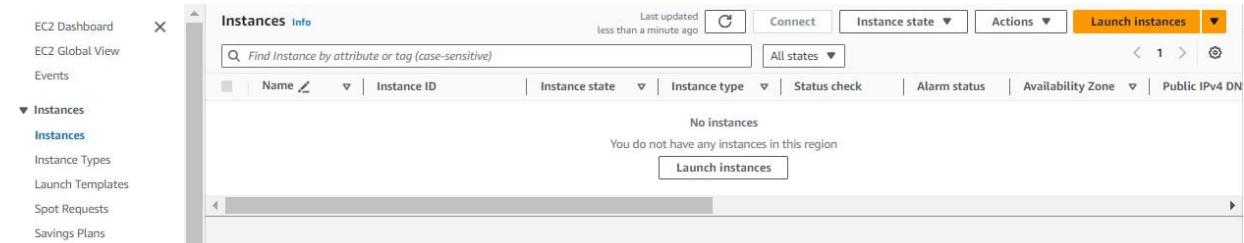
- **Activity 6.1: Launch an EC2 instance to host the Flask application.**

- **Launch EC2 Instance**

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



- Click on Launch instance to launch EC2 instance



The screenshot shows the AWS EC2 Instances page. The main header bar includes 'Instances Info', 'Last updated less than a minute ago', 'Connect', 'Instance state', 'Actions', and a prominent orange 'Launch instances' button. Below this is a search bar with placeholder 'Find Instance by attribute or tag (case-sensitive)' and dropdown filters for 'All states'. A message below the search bar states 'No instances' and 'You do not have any instances in this region'. At the bottom right of the main area is a large 'Launch instances' button.

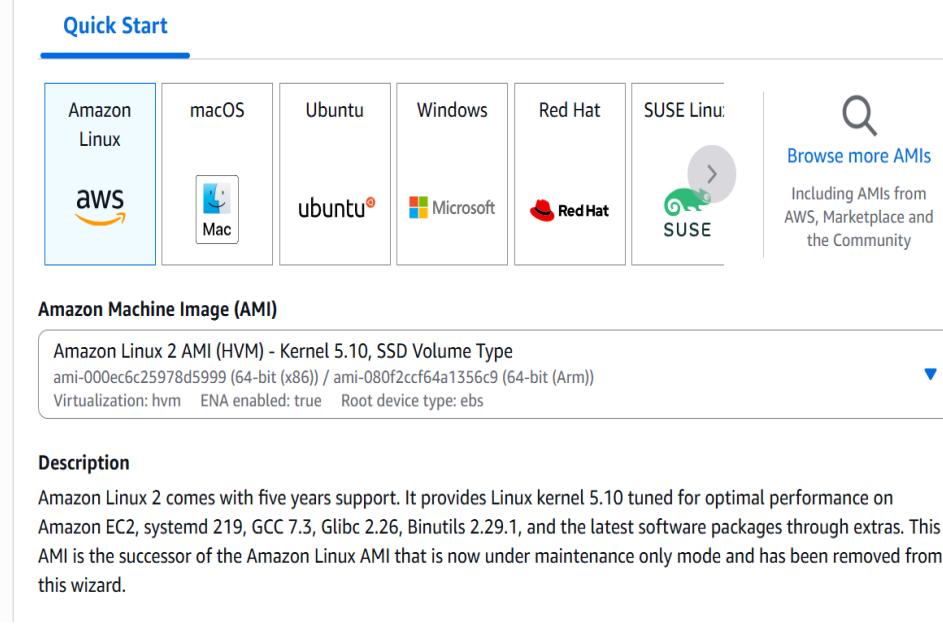
The browser navigation bar shows 'aws' and 'EC2 > Instances > Launch an instance'. A blue banner at the top of the content area says 'It seems like you may be new to launching instances in EC2. Take a walkthrough to learn about EC2, how to launch instances and about best practices' with 'Take a walkthrough' and 'Do not show me this message again.' buttons.

The main content area is titled 'Launch an instance' and contains several sections:

- Name and tags**: A field for 'Name' containing 'medtrack-server' and a link to 'Add additional tags'.
- Application and OS Images (Amazon Machine Image)**: A search bar with placeholder 'Search our full catalog including 1000s of application and OS images'.
- Summary**: Shows 'Number of instances' set to 1, 'Software Image (AMI)' as 'Amazon Linux 2 Kernel 5.10 AMI...', 'Virtual server type (instance type)' as 't2.micro', and 'Storage (volumes)' with an option to 'Activate Windows'.

At the bottom of the page are links for 'CloudShell', 'Feedback', and copyright information: '© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences'.

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



The screenshot shows the AWS Quick Start page under the 'Amazon Machine Image (AMI)' section. It features a grid of icons for various operating systems:

- Amazon Linux
- macOS
- Ubuntu
- Windows
- Red Hat
- SUSE Linux

Below the grid is a search icon and a link 'Browse more AMIs'. A note says 'Including AMIs from AWS, Marketplace and the Community'.

A detailed box for 'Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type' is shown, listing details: 'ami-000ec6c25978d5999 (64-bit (x86)) / ami-080f2ccf64a1356c9 (64-bit (Arm))', 'Virtualization: hvm', 'ENA enabled: true', and 'Root device type: ebs'.

The 'Description' section provides a summary: 'Amazon Linux 2 comes with five years support. It provides Linux kernel 5.10 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is now under maintenance only mode and has been removed from this wizard.'

- 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

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

Instance type

t2.micro	▼
Family: t2 1 vCPU 1 GiB Memory Current generation: true	

All generations

Compare instance types

▼ **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

medtrack-server1

**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	
64-bit (x86)	uefi-preferred	ami-078264b8ba71bc45e	ec2-user	

**Summary**

Number of instances: [Info](#)  
1

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

Virtual server type (instance type)  
t2.micro

Firewall (security group)  
New security group

Storage (volumes)  
1 volume(s) - 8 GiB

 **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 I/Os, 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)  
 172.31.0.0/16

Subnet [Info](#)  
 No preference [Create new subnet](#)

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

**Inbound Security Group Rules**

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0) [Remove](#)

Type <a href="#">Info</a> ssh	Protocol <a href="#">Info</a> TCP	Port range <a href="#">Info</a> 22
Source type <a href="#">Info</a> Anywhere	Source <a href="#">Info</a> <input type="text"/> Add CIDR, prefix list or security	Description - optional <a href="#">Info</a> e.g. SSH for admin desktop
<input type="text"/> 0.0.0.0/0 <a href="#">X</a>		

▼ Security group rule 2 (TCP, 80, 0.0.0.0/0) [Remove](#)

Type <a href="#">Info</a> HTTP	Protocol <a href="#">Info</a> TCP	Port range <a href="#">Info</a> 80
Source type <a href="#">Info</a> Custom	Source <a href="#">Info</a> <input type="text"/> Add CIDR, prefix list or security	Description - optional <a href="#">Info</a> e.g. SSH for admin desktop
<input type="text"/> 0.0.0.0/0 <a href="#">X</a>		

▼ Security group rule 3 (TCP, 5000, 0.0.0.0/0) [Remove](#)

Type <a href="#">Info</a> Custom TCP	Protocol <a href="#">Info</a> TCP	Port range <a href="#">Info</a> 5000
Source type <a href="#">Info</a> Custom	Source <a href="#">Info</a> <input type="text"/> Add CIDR, prefix list or security	Description - optional <a href="#">Info</a> e.g. SSH for admin desktop
<input type="text"/> 0.0.0.0/0 <a href="#">X</a>		

[Add security group rule](#)

EC2 > ... > Launch an Instance

**Success**  
Successfully initiated launch of instance i-001861022fbcac290

▶ Launch log

**Next Steps**

Q. What would you like to do next with this instance, for example "create alarm" or "create backup"?

< 1 2 3 4 >

Create billing and free tier usage alerts	Connect to your instance	Connect an RDS database	Create EBS snapshot policy	Manage detailed monitoring	Create Load Balancer
To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.	Once your instance is running, log into it from your local computer.	Configure the connection between an EC2 instance and a database to allow traffic flow between them.	Create a policy that automates the creation, retention, and deletion of EBS snapshots	Enable or disable detailed monitoring for the instance. If you enable detailed monitoring, the Amazon EC2 console displays monitoring graphs with a 1-minute period.	Create a application, network gateway or classic Elastic Load Balancer
<a href="#">Create billing alerts</a>	<a href="#">Connect to instance</a>	<a href="#">Connect an RDS database</a>	<a href="#">Create EBS snapshot policy</a>	<a href="#">Manage detailed monitoring</a>	<a href="#">Create Load Balancer</a>
<a href="#">Learn more</a>	<a href="#">Learn more</a>	<a href="#">Create a new RDS database</a>	<a href="#">Learn more</a>		
Create AWS budget	Manage CloudWatch alarms	Disaster recovery for your instances	Monitor for suspicious runtime activities	Get instance screenshot	Get system log
AWS Budgets allows you to create budgets, forecast spend, and take action on your costs and usage from a single location.	Create or update Amazon CloudWatch alarms for the instance.	Recover the instances you just launched into a different Availability Zone or a different Region using AWS Elastic Disaster Recovery (DRS).	Amazon GuardDuty enables you to continuously monitor for malicious runtime activity and unauthorized behavior, with near real-time visibility into on-host activities occurring across your Amazon EC2 workloads.	<a href="#">Get instance screenshot</a>	<a href="#">Get system log</a>
<a href="#">Create AWS budget</a>	<a href="#">Manage CloudWatch alarms</a>	<a href="#">Disaster recovery for your instances</a>	<a href="#">Monitor for suspicious runtime activities</a>		

[View all instances](#)

- 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.

aws [Alt+S] Search United States (N. Virginia) us-east-1 @ riosandsome@ws1

EC2 > Instances

Instances (1/1) [Info](#) Last updated less than a minute ago [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public
medtrack-server	i-08834150fd657c24e	<a href="#">Running</a>	t2.micro	<a href="#">2/2 checks passed</a>	<a href="#">View alarms</a> +	us-east-1c	ec2-44

[View all instances](#)

**i-08834150fd657c24e (medtrack-server)**

[Details](#) Status and alarms Monitoring Security Networking Storage Tags

**Instance summary** [Info](#) Updated less than a minute ago

Instance ID	<a href="#">i-08834150fd657c24e</a>	Public IPv4 address	<a href="#">44.220.130.196</a> [open address]	Private IPv4 addresses	<a href="#">172.51.22.7</a>
IPv6 address	-	Internet address	-	Activate Windows	Go to Settings to activate Windows.

EC2 > Instances > i-001861022fbcac290

Instance summary for i-001861022fbcac290 (InstantLibraryApp) [Info](#) Updated less than a minute ago

Instance ID	<a href="#">i-001861022fbcac290</a>	Public IPv4 address	-	Private IPv4 addresses	<a href="#">172.31.5.5</a>
IPv6 address	-	Instance state	<a href="#">Stopped</a>	Public IPv4 DNS	-
Hostname type	IP name: ip-172-31-5-ap-south-1.compute.internal	Private IP DNS name (IPv4 only)	<a href="#">ip-172-31-5-ap-south-1.compute.internal</a>	Elastic IP addresses	-
Answer private resource DNS name	IPv4 (A)	Instance type	t2.micro	AWS Compute Optimizer finding	<a href="#">Opt-in to AWS Compute Optimizer for recommendations.</a>   Learn more
Auto-assigned IP address	-	VPC ID	<a href="#">vpc-03cdc7b6f19dd7211</a>	Subnet ID	<a href="#">subnet-0d9fa5144480cc9a</a>
IAM Role	<a href="#">sns_Dynamodb_role</a>	Subnet	-	Instance ARN	<a href="#">arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290</a>
IMDSv2 Required	-	Tags	-	Auto Scaling Group name	-

[Details](#) Status and alarms Monitoring Security Networking Storage Tags

EC2 > Instances > i-001861022fbcac290

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	Change security groups	<a href="#">Get Windows password</a>
Answer private resource DNS name	–	Instance type	t2.micro	Elastic IP addresses	–
IPv4 (A)	–	VPC ID	vpc-03ccdc7b6f19dd7211	AWS Compute Optimizer finding	Opt-in to AWS Compute Optimizer for recommendations.   Learn more <a href="#">[?]</a>
Auto-assigned IP address	–	Subnet ID	subnet-0d9fa3144480cc9a9	Auto Scaling Group name	–
IAM Role	<a href="#">sns_Dynamodb_role</a>	Instance ARN	arn:aws:ec2:ap-south-1:557690616856:instance/i-001861022fbcac290	–	–
IMDSv2	Required				

EC2 > Instances > i-001861022fbcac290 > Modify IAM role

## Modify IAM role [Info](#)

Attach an IAM role to your instance.

Instance ID

[i-001861022fbcac290 \(InstantLibraryApp\)](#)

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.

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

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

- Now connect the EC2 with the files



## 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/your-github-username/your-repository-name.git>'

Note: change your-github-username and your-repository-name with your credentials

here: 'git clone https://github.com/AlekhyaPenubakula/InstantLibrary.git'

- This will download your project to the EC2 instance.

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

```
cd InstantLibrary
```

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
```

```
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
    /###\#
   \###|
  /##/
 V~-> https://aws.amazon.com/linux/amazon-linux-2023
  .-
  /|/
 /|/
/m.

Last login: Tue Oct 15 04:17:59 2024 from 13.233.177.3
[ec2-user@ip-172-31-3-5 ~]$ git clone https://github.com/AlekhyaPenubakula/InstantLibrary.git
fatal: destination path 'InstantLibrary' already exists and is not an empty directory.
[ec2-user@ip-172-31-3-5 ~]$ cd InstantLibrary
[ec2-user@ip-172-31-3-5 InstantLibrary]$ cd InstantLibrary
[ec2-user@ip-172-31-3-5 InstantLibrary]$ flask run --host=0.0.0.0 --port=80
 * Debug mode: off
Permission denied
[ec2-user@ip-172-31-3-5 InstantLibrary]$ ^C
[ec2-user@ip-172-31-3-5 InstantLibrary]$ ^C
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
 * Debug mode: off
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://127.0.0.1:80
 * Running on http://172.31.3.5:80
Press CTRL+C to quit
^C[ec2-user@ip-172-31-3-5 InstantLibrary]$ 
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
 * Debug mode: off
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://127.0.0.1:80
 * Running on http://172.31.3.5:80
Press CTRL+C to quit
183.82.125.56 - - [22/Oct/2024 07:42:00] "GET / HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /register HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /static/images/library3.jpg HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /favicon.ico HTTP/1.1" 404 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /static/images/library3.jpg HTTP/1.1" 304 -
183.82.125.56 - - [22/Oct/2024 07:42:21] "POST /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:24] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:27] "POST /login HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:28] "GET /home-page HTTP/1.1" 200 -
```

i-001861022fbcac290 (InstantLibraryApp)

PublicIPs: 13.201.74.42 PrivateIPs: 172.31.3.5

## Verify the Flask app is running:

<http://your-ec2-public-ip>

- Run the Flask app on the EC2 instance

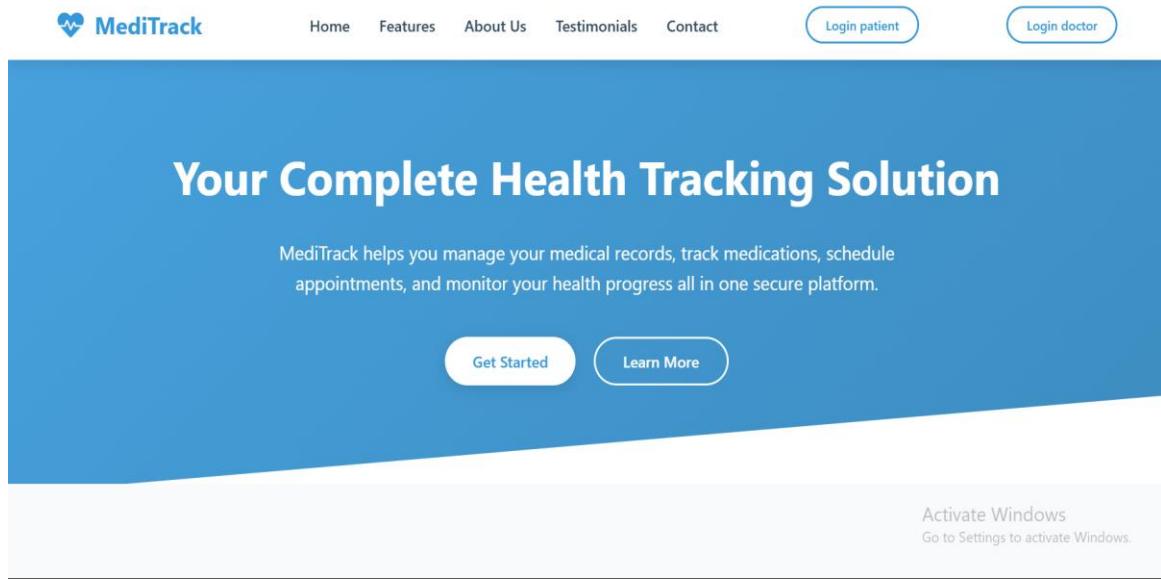
```
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://172.31.3.5:80
Press CTRL+C to quit
183.82.125.56 - - [22/Oct/2024 07:42:00] "GET / HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /register HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /static/images/library3.jpg HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /favicon.ico HTTP/1.1" 404 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /static/images/library3.jpg HTTP/1.1" 304 -
183.82.125.56 - - [22/Oct/2024 07:42:21] "POST /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:24] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:27] "POST /login HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:28] "GET /home-page HTTP/1.1" 200 -
```

**Access the website through:**  
Public IPs: <https://13.201.74.42/>

## Milestone 8: Testing and Deployment

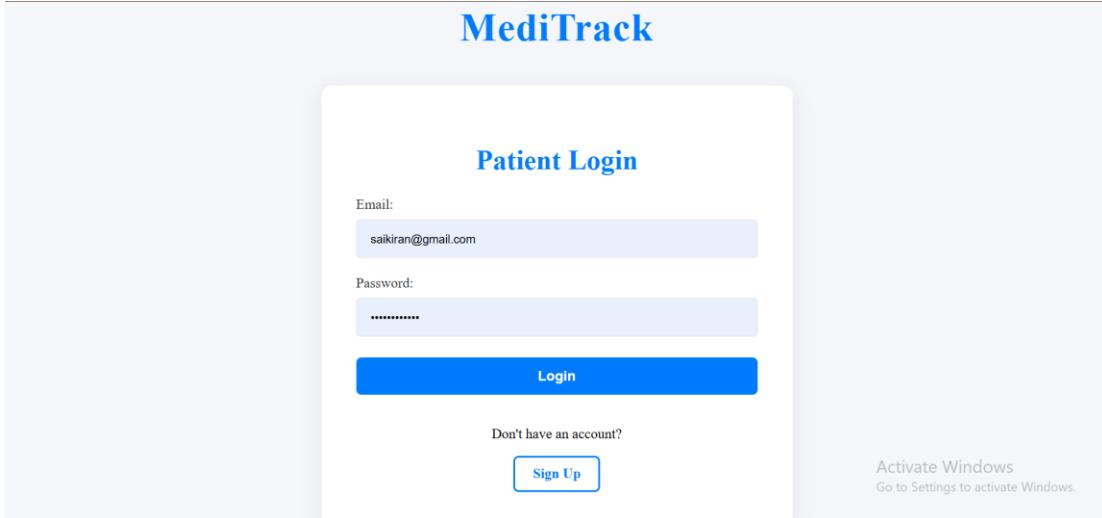
- **Activity 8.1: Conduct functional testing to verify user registration, login, book requests, and notifications.**

**Index page:**



The screenshot shows the homepage of MediTrack. At the top, there is a navigation bar with links for Home, Features, About Us, Testimonials, and Contact. To the right of these links are two buttons: "Login patient" and "Login doctor". Below the navigation bar, a large blue header section contains the text "Your Complete Health Tracking Solution". Underneath this, a subtext states: "MediTrack helps you manage your medical records, track medications, schedule appointments, and monitor your health progress all in one secure platform." At the bottom of the blue section are two buttons: "Get Started" and "Learn More". The footer of the page includes a link to activate Windows: "Activate Windows" and "Go to Settings to activate Windows".

**Patient Login Page:**



The screenshot shows the Patient Login page of MediTrack. The page features a large "MediTrack" logo at the top center. Below it is a "Patient Login" form. The form has two input fields: "Email:" containing "saikiran@gmail.com" and "Password:" containing a redacted password. A "Login" button is located below the password field. At the bottom of the form, there is a link "Don't have an account?". To the right of the form, there is a "Sign Up" button and a "Activate Windows" section with the text "Go to Settings to activate Windows".

**Sign up Page:**

# MedTrack

## Patient Signup

Name

Email

Password

**Sign Up**

Already have an account?

[Login](#)

A

G

**Patient dashboard page:**

🕒 **MediTrack**

- 🏠 [Dashboard](#)
- 💊 [Medications](#)
- 📅 [Appointments](#)
- 📄 [Prescriptions](#)
- 📞 [Video Consult](#)
- ↳ [Health Reports](#)
- 👤 [Profile](#)
- ⚙️ [Settings](#)

xyz  
Patient

[Logout](#)

🔍 
🔔 ✉️ ❓

## Patient Dashboard

Welcome back, xyz. Here's an overview of your health.

**Active Medications** 💊

**5**

2 need refill soon

**Upcoming Appointments** 📅

**0**

No upcoming appointments

**Health Score** 之心

**85%**

Last Month: 80% +5%

**Prescriptions** 📄

**4**

1 expires soon

Activate Windows  
Go to Settings to activate Windows.

**Doctor dashboard page:**


**MediTrack**

-  [Dashboard](#)
-  [Patients](#)
-  [Appointments](#)
-  [Video Consult](#)
-  [Prescriptions](#)
-  [Analytics](#)
-  [Profile](#)
-  [Settings](#)

Dr. Emily Johnson  
Endocrinologist

 [Logout](#)

4 7 ?

## Doctor Dashboard

Welcome back, Dr. Johnson. Here's your practice overview.

**Total Patients** 

**124**

+8 this month

**Today's Appointments** 

**7**

Next: John Doe at 10:00 AM

**Video Consultations** 

**3**

Next: 2:30 PM with Sarah Johnson

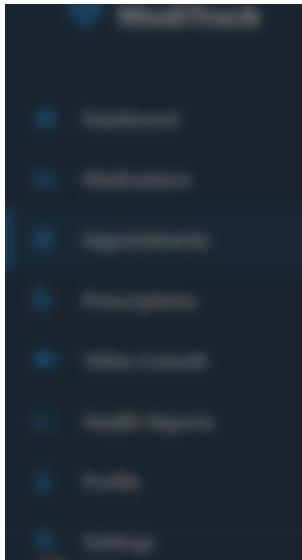
**Prescriptions** 

**15**

Issued this week

Activate Windows  
Go to Settings to activate Windows.

### Bookappointment Page:



**Schedule Appointment**

Doctor:

Date:  

Time:  

Title:

Location:

**Book Appointment**

### Doctor login :

## Doctor Login

Signup successful! Please log in.

Email:

Password:

**Login**

Don't have an account?

[Sign Up](#)

### Doctor dashboard Page:



- [!\[\]\(8bfad07bd7c7cb585d2bb309f2b80f96\_img.jpg\) Dashboard](#)
- [!\[\]\(44901738a580c3ee561a4e17d37db092\_img.jpg\) Patients](#)
- [!\[\]\(ab1df6ea0f41abac31256d1be6650d2a\_img.jpg\) Appointments](#)
- [!\[\]\(ce2cb50be2732a6370ade6466191e52f\_img.jpg\) Video Consult](#)
- [!\[\]\(6dff3266b643c898a4d91ae83882bf30\_img.jpg\) Prescriptions](#)
- [!\[\]\(73c53574b003488b1c8e760bf60e5a98\_img.jpg\) Analytics](#)
- [!\[\]\(6526d8aa6a3951535269570ec0dd4337\_img.jpg\) Profile](#)
- [!\[\]\(961e3caea857d9ded50235c50a63a356\_img.jpg\) Settings](#)

 Dr. sai  
Doctor

[Logout](#)

## Doctor Dashboard

Welcome back, Dr. sai. Here's your practice overview.

**Total Patients**



**0**

+8 this month

**Today's Appointments**



**1**

Next: xyz@gmail.com at 23:11

**Video Consultations**



**0**

No video consults today

**Prescriptions**



**0**

0 issued this week

Activate Windows  
Go to Settings to activate Windows.

## Upcoming Appointments Page:

### Recent Activity

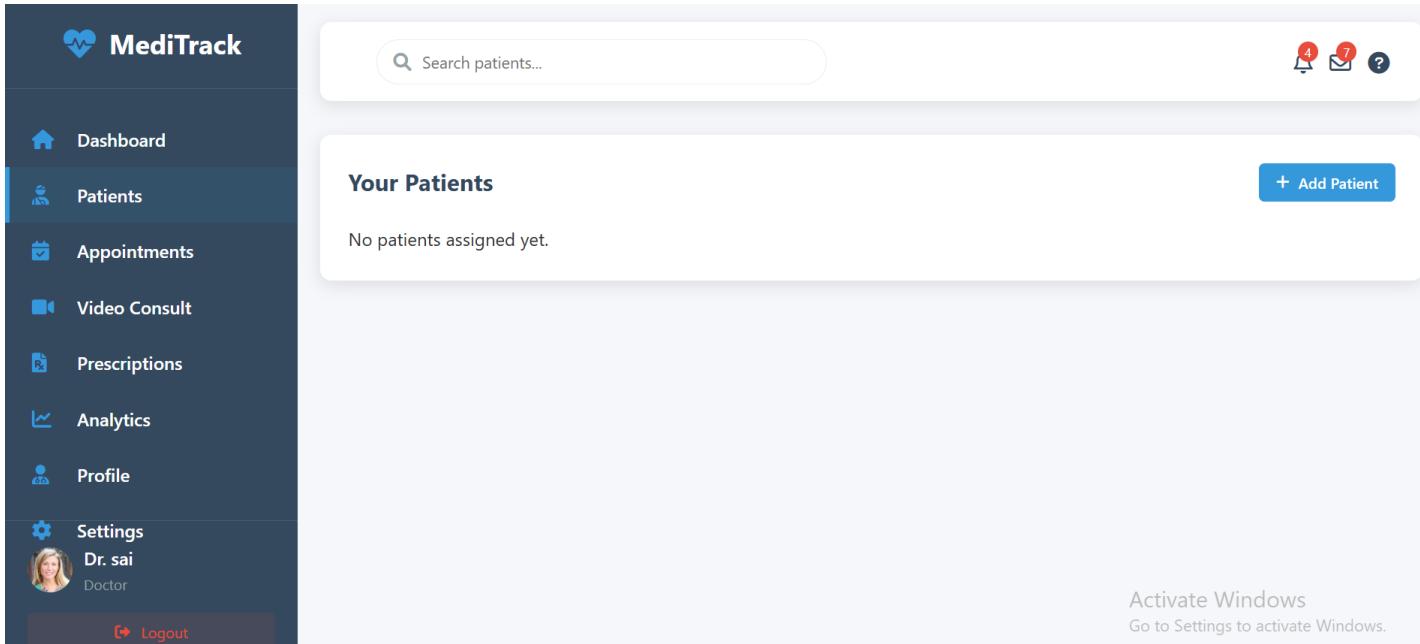
**Consultation**

⌚ 23:11 🚩 xyz@gmail.com

↻ Refresh

Activate Windows

## Add patient Page:



**MediTrack**

- Dashboard
- Patients**
- Appointments
- Video Consult
- Prescriptions
- Analytics
- Profile
- Settings

Dr. sai  
Doctor

Logout

Search patients...

Your Patients

No patients assigned yet.

+ Add Patient

4 7 ?

Activate Windows  
Go to Settings to activate Windows.

**Add patient with Email Page:**



**Patient Dash board Page:**

## Patient Dashboard

Welcome back, xyz. Here's an overview of your health.

### Active Medications



**5**

2 need refill soon

### Upcoming Appointments



**1**

Next: saikiran@gmail.com on 2025-07-09

### Health Score

**85%**

Last Month: 80%

+5

### Prescriptions



**4**

1 expires soon

Activate Windows  
Go to Settings to activate Windo

## Conclusion:

The **MedTrack application** has been successfully developed and deployed using a robust cloud-based architecture tailored for modern healthcare environments. Leveraging AWS services such as EC2 for hosting, DynamoDB for secure and scalable patient data management, and SNS for real-time alerts, the platform ensures reliable and efficient access to essential medical tracking services. This system addresses critical challenges in healthcare such as managing patient records, monitoring medication schedules, and ensuring timely communication between healthcare providers and patients.

The cloud-native approach enables seamless scalability, allowing MedTrack to support increasing numbers of users and data without compromising performance or reliability. The integration of Flask with AWS ensures smooth backend operations, including patient registration, medication reminders, and health updates. Thorough testing has validated all features—from user onboarding to alert notifications—function reliably and securely.

In conclusion, the MedTrack application delivers a smart, efficient solution for modernizing healthcare management, improving patient care, and streamlining communication between medical staff and patients. This project highlights the transformative power of cloud-based technologies in solving real-world challenges in the healthcare sector.

