

# **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**

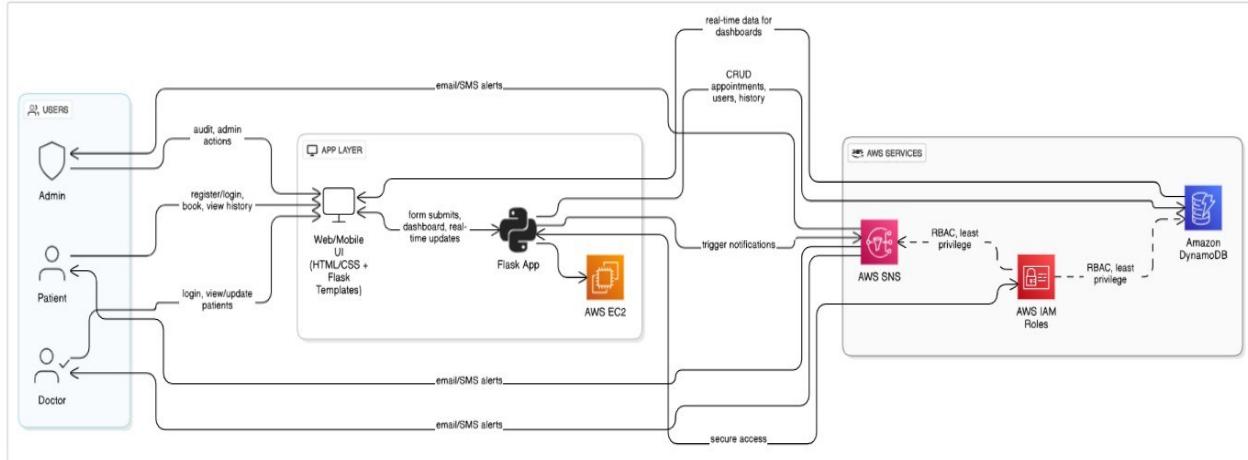
MedTrack utilizes AWS IAM to manage user permissions and ensure secure access to the system. For instance, when a new patient registers, an IAM user is created with specific roles and permissions to access only the features relevant to them. Doctors have their own IAM configurations, allowing them access 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**

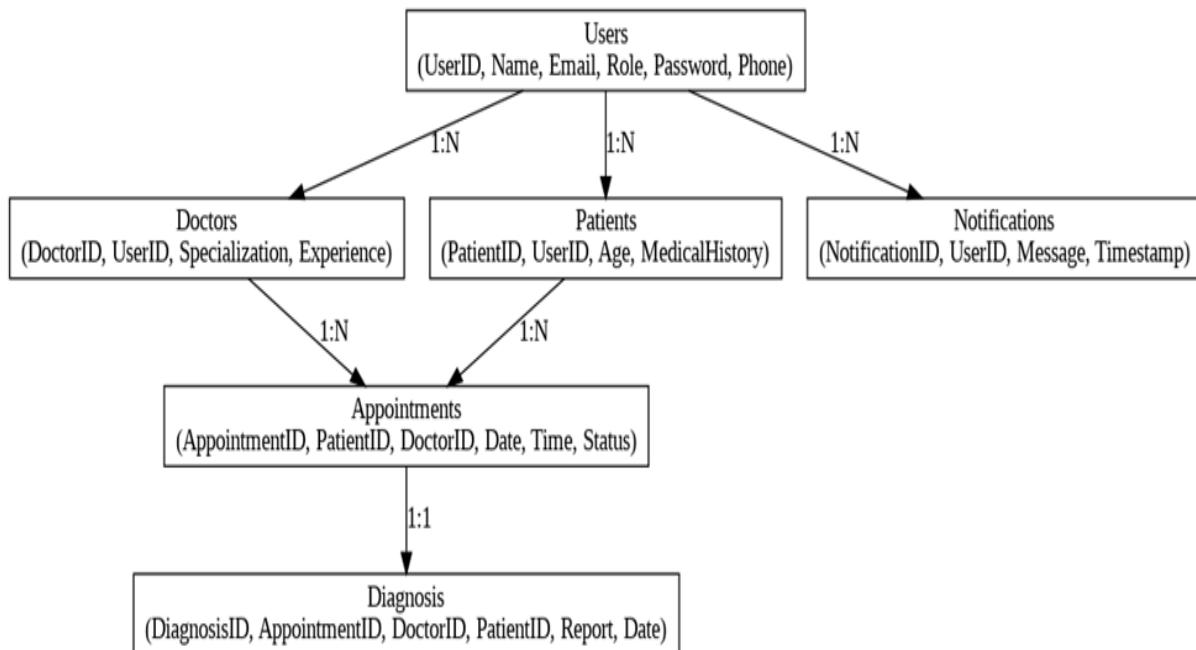
The MedTrack system provides doctors and patients with easy access to medical histories and relevant resources. For example, a doctor logs in to view a patient's medical history and upcoming appointments. They can quickly access, and update records as needed. Flask manages real-time data fetching from 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

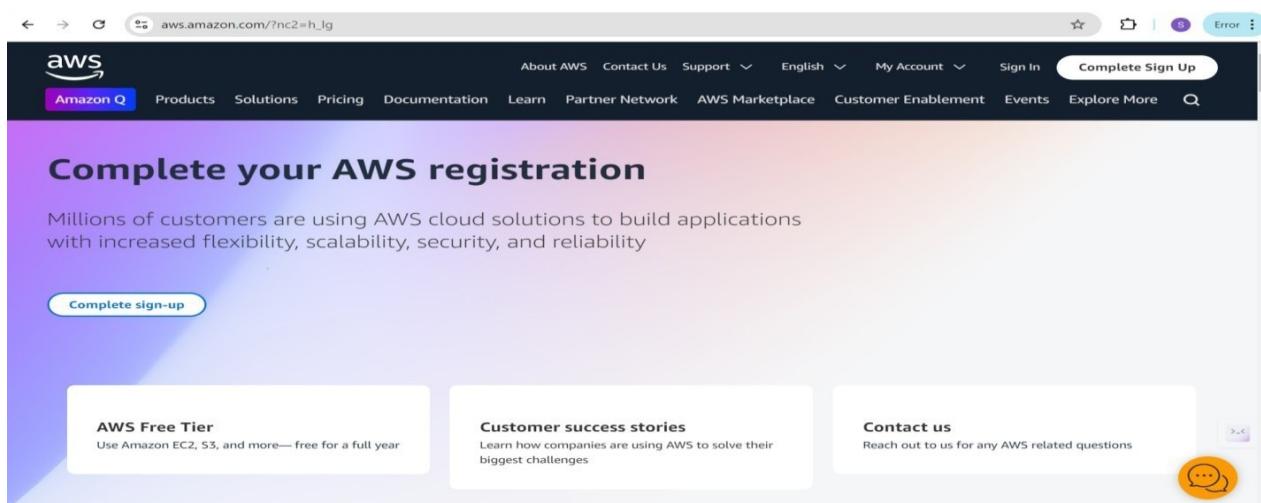
## Activity 7.2: Run the Flask App

## 8. Testing and Deployment

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

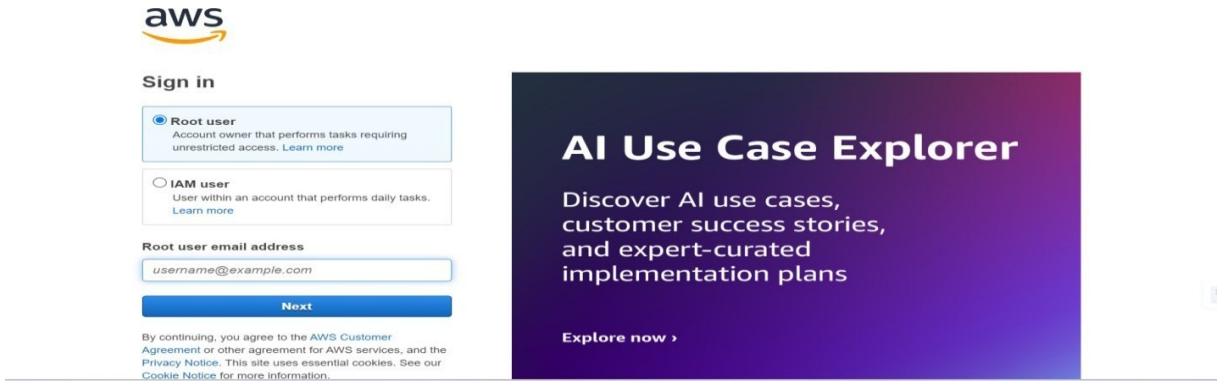
### Milestone 1: AWS Account Setup and Login

- **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](#).



## 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.

A screenshot of the AWS Services search interface. 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 area shows search results for 'dyn':

- DynamoDB** ☆ Managed NoSQL Database
- Amazon DocumentDB** ☆ Fully-managed MongoDB-compatible database service
- CloudFront** ☆ Global Content Delivery Network
- Athena** ☆ Serverless interactive analytics service

Below these are sections for **Features**, **Settings** (DynamoDB feature), and **Clusters** (DynamoDB feature).

DynamoDB Dashboard

**Alarms (0) Info**

Manage in CloudWatch

Find alarms

Alarm name Status

No custom alarms

**DAX clusters (0) Info**

View details

Find clusters

Cluster name Status

No clusters

No clusters to display

Create cluster

**Create resources**

Create an Amazon DynamoDB table for fast and predictable database performance at any scale. Learn more

Create table

Amazon DynamoDB Accelerator (DAX) is a fully-managed, highly-available, in-memory caching service for DynamoDB. Learn more

Create DAX cluster

**What's new**

SEP 14 AWS Cost Management now provides purchase recommendations for Amazon DynamoDB...

DynamoDB Tables

**Tables (0) Info**

Find tables

Any tag key

Any tag value

Actions Delete Create table

Name Status Partition key Sort key Indexes Deletion protection Read capacity mode Write capacity mode Total size

You have no tables in this account in this AWS Region.

Create table

- **Activity 2.2 : Create a DynamoDB table for storing registration details and book requests.**
  - Create Users table with partition key “Email” with type String and click on create tables.



## 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.

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.



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.



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.

**DynamoDB** ×  ⓘ ⟳

**Tables** (1) **Info**

Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size
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.

## 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.

Requests

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.

email

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.

### Table settings



Default settings

The easiest way to make your table. You can modify.



Customize settings

Use these advanced features to make DynamoDB work.

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.

The Requests table was created successfully.

**Create table**

Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size
Requests	Active	email (\$)	-	0	Off	Provisioned (5)	Provisioned (5)	0 bytes
Users	Active	email (\$)	-	0	Off	Provisioned (5)	Provisioned (5)	0 bytes

## Milestone 3: SNS Notification Setup

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

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

The screenshot shows the AWS search interface with the query 'sns' entered in the search bar. The results are categorized under 'Services' and 'Features'. The 'Services' section is expanded, showing:

- Simple Notification Service** (star icon): SNS managed message topics for Pub/Sub.
- Route 53 Resolver**: Resolve DNS queries in your Amazon VPC and on-premises network.
- Route 53** (star icon): Scalable DNS and Domain Name Registration.
- AWS End User Messaging** (star icon): Engage your customers across multiple communication channels.

The 'Features' section is also listed:

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

The screenshot shows the Amazon SNS landing page. On the left, there's a navigation sidebar with links like Dashboard, Topics, Subscriptions, Mobile (Push notifications, Text messaging (SMS)), and Application Integration. The main content area features a large title 'Amazon Simple Notification Service' with the subtitle 'Pub/sub messaging for microservices and serverless applications.' Below the title is a brief description of the service. To the right, there's a 'Create topic' form with a 'Topic name' field containing 'MyTopic', a 'Next step' button, and a link to 'Start with an overview'. At the bottom right of the main content area, there's a 'Pricing' link.

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

The screenshot shows the 'Topics' page within the Amazon SNS console. The left sidebar has the same navigation as the previous screen. The main area displays a table titled 'Topics (0)' with columns for Name, Type, and ARN. A search bar and buttons for Edit, Delete, Publish message, and Create topic are at the top of the table. Below the table, a message says 'No topics' and 'To get started, create a topic.' with a 'Create topic' button.

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

## 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
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS

Standard

- Best-effort message ordering
- At-least once message delivery
- Highest throughput in publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

#### Name

BookRequestNotifications

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.

My Topic

Maximum 100 characters.

#### Access policy - optional [Info](#)

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 [Info](#)

This policy defines which sensitive data to monitor and to prevent from being exchanged via your topic.

#### Delivery policy (HTTP/S) - optional [Info](#)

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 [Info](#)

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 [Info](#)

Use AWS X-Ray active tracing for this topic to view its traces and service map in Amazon CloudWatch. Additional costs apply.

Cancel

Create topic

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

The screenshot shows the Amazon SNS console. On the left, there's a sidebar with links like Dashboard, Topics, Subscriptions, Mobile, Push notifications, and Text messaging (SMS). The main area is titled 'BookRequestNotifications' and shows a success message: 'Topic BookRequestNotifications created successfully. You can create subscriptions and send messages to them from this topic.' Below this, there are tabs for Details, Subscriptions, Access policy, Data protection policy, Delivery policy (HTTP/S), Delivery status logging, Encryption, Tags, and Integrations. The Subscriptions tab is active, showing a table with a single row labeled 'Create subscription'. The table has columns for ID, Endpoint, Status, and Protocol. At the bottom of the table, it says 'No subscriptions found' and 'You don't have any subscriptions to this topic.' There are also buttons for Edit, Delete, Request confirmation, Confirm subscription, and Create subscription.

- **Activity 3.2: Subscribe users and staff to relevant SNS topics to receive real-time notifications when a book request is made.**
  - Subscribe users (or admin staff) to this topic via Email. When a book request is made, notifications will be sent to the subscribed emails.

Amazon SNS > Subscriptions > Create subscription

## Create subscription

**Details**

**Topic ARN**

X

**Protocol**  
The type of endpoint to subscribe

▼

**Endpoint**  
An email address that can receive notifications from Amazon SNS.

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

► **Subscription filter policy - optional** [Info](#)

This policy filters the messages that a subscriber receives.

► **Redrive policy (dead-letter queue) - optional** [Info](#)

Send undeliverable messages to a dead-letter queue.

Cancel
Create subscription

Amazon SNS

New Feature  
Amazon SNS now supports in-place message archiving and replay for FIFO topics. [Learn more](#)

Subscription to BookRequestNotifications created successfully.  
The ARN of the subscription is arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4

Subscription: d78e0371-9235-404d-952c-85c2743607c4

**Details**

ARN arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4	Status <span style="color: #337ab7;">i</span> Pending confirmation
Endpoint	Protocol
instantlibrary2@gmail.com	EMAIL
Topic	
BookRequestNotifications	
Subscription Principal	
arn:aws:iam::557690616836:root	

[Subscription Filter policy](#) [Redrive policy \(dead-letter queue\)](#)

**Subscription Filter policy** [Info](#)

This policy filters the messages that a subscriber receives.

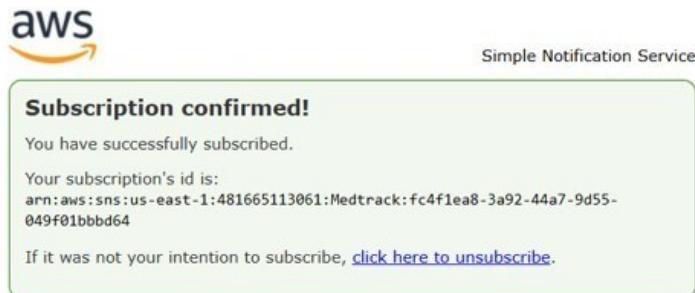
No filter policy configured for this subscription.  
To apply a filter policy, edit this subscription.

- After subscription request for the mail confirmation

The screenshot shows the Amazon SNS console. A banner at the top indicates a new feature: "Amazon SNS now supports in-place message archiving and replay for FIFO topics." Below this, a message says "Confirmation request was sent successfully." The main area displays a topic named "BookRequestNotifications". The "Details" section shows the name, ARN, and type. The "Subscriptions" section lists two entries, both marked as "Pending confirmation".

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

The screenshot shows a Gmail inbox with several messages. One message from "AWS Notifications <no-reply@sns.amazonaws.com>" is selected. The subject is "AWS Notification - Subscription Confirmation". The message body contains a note about being identified as spam and a button to "Report as not spam". It also includes a link to "Confirm subscription".



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

The screenshot shows the AWS SNS 'Topics' page. On the left, a sidebar lists navigation options: Dashboard, Topics (selected), Subscriptions, Mobile (Push notifications, Text messaging (SMS)), and a 'Create new topic' button. The main content area shows a topic named 'BookRequestNotifications'. The 'Details' section includes fields for Name (BookRequestNotifications), Display name (empty), ARN (arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications), Topic owner (557690616836), and Type (Standard). Below this, a 'Subscriptions' section shows two entries:

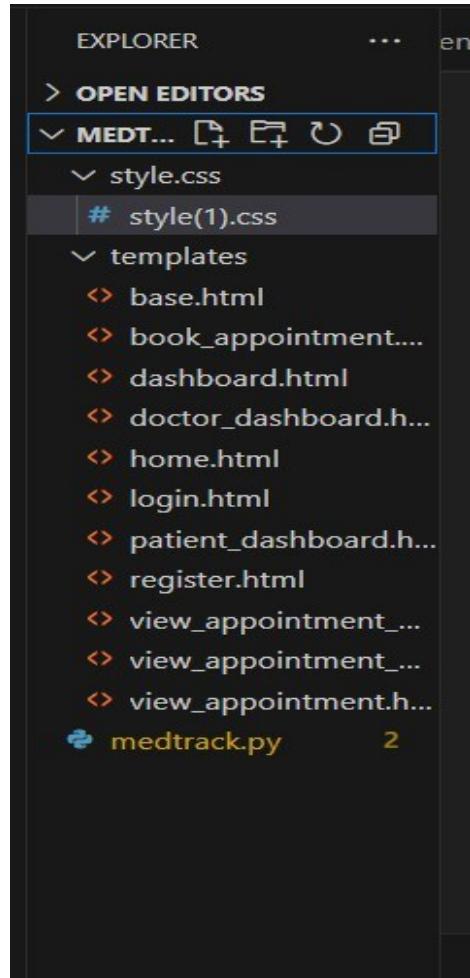
ID	Endpoint	Status	Protocol
d78e0371-9235-404d-952c-85c2743607c4	instantlibrary2@gmail.com	Confirmed	EMAIL

At the top right of the main content area, there are buttons for Edit, Delete, Publish message, Request confirmation, Confirm subscription, and Create subscription. The 'Create subscription' button is highlighted in orange.

## Milestone 4:Backend Development and Application Setup

- **Activity 4.1: Develop the backend using Flask**

- File Explorer Structure



**Description:** set up the INSTANT LIBRARY project with an app.py file, a static/ folder for assets, and a templates/ directory containing all required HTML pages like home, login, register, subject-specific pages (e.g., computer\_science.html, data\_science.html), and utility pages (e.g., request-form.html, statistics.html).

### 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:**

```
# Initialize DynamoDB resource
dynamodb = boto3.resource('dynamodb', region_name='ap-south-1')

# DynamoDB Tables
users_table = dynamodb.Table('Users') # Ensure the 'Users' table
requests_table = dynamodb.Table('Requests') # Ensure the 'Requests'
```

**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**

```

# SNS Topic ARN (create the SNS topic in AWS and provide the ARN here)
sns = boto3.client('sns', region_name='ap-south-1')
sns_topic_arn = 'arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications'

# Email settings (for sending emails)
SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 587
SENDER_EMAIL = "instantlibrary2@gmail.com"
SENDER_PASSWORD = "luut dsih nyvq dgzv" # Your app password

# Function to send email
def send_email(to_email, subject, body):
    msg = MIMEText(body)
    msg['From'] = SENDER_EMAIL
    msg['To'] = to_email
    msg['Subject'] = subject
    msg.attach(MIMEText(body, 'plain'))

    try:
        server = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
        server.starttls()
        server.login(SENDER_EMAIL, SENDER_PASSWORD)
        text = msg.as_string()
        server.sendmail(SENDER_EMAIL, to_email, text)
        server.quit()
        print("Email sent successfully")
    except Exception as e:
        print(f"Failed to send email: {e}")

```

**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**

- **Home Route:**

```

# Home route redirects to Registration page
@app.route('/')
def home():
    return redirect(url_for('register'))

```

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

- **Register Route:**

```

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

        # Basic Validation: Ensure all fields are filled
        if not name or not email or not password or not confirm_password:
            return "All fields are mandatory! Please fill out the entire form."
        if password != confirm_password:
            return "Passwords do not match! Please try again."

        # Check if user already exists
        response = users_table.get_item(Key={'email': email})
        if 'Item' in response:
            return "User already exists! Please log in."

        # Hash the password
        hashed_password = hashpw(password.encode('utf-8'), gensalt()).decode('utf-8')

        # Store user in DynamoDB with login_count initialized to 0
        users_table.put_item(
            Item={
                'email': email,
                'name': name,
                'password': hashed_password,
                'login_count': 0
            }
        )

        # Send SNS notification for new registration
        sns.publish(
            TopicArn=sns_topic_arn,
            Message=f'New user registered: {name} ({email})',
            Subject='New User Registration'
        )

    return redirect(url_for('login'))
return render_template('register.html')

```

**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):**

```

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

        # Basic Validation: Ensure both fields are filled
        if not email or not password:
            return "Please enter both email and password."

        # Fetch user data from DynamoDB
        response = users_table.get_item(Key={'email': email})
        user = response.get('Item')

        if not user or not checkpw(password.encode('utf-8'), user['password'].encode('utf-8')):
            return "Incorrect email or password! Please try again."

        # Update login count
        users_table.update_item(
            Key={'email': email},
            UpdateExpression='SET login_count = login_count + :inc',
            ExpressionAttributeValues={':inc': 1}
        )

        # Successful login
        return redirect(url_for('home_page'))
    return render_template('login.html')

```

**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

- Home, E-book buttons and subject routes:

```

# Home Page with E-Books, Request Books, and Exit
@app.route('/home-page')
def home_page():
    return render_template('home.html')

# E-Books Page (Dropdown Selection for Course and Subject)
@app.route('/ebook-buttons', methods=['GET', 'POST'])
def ebook_buttons():
    if request.method == 'POST':
        subject = request.form['subject']
        return redirect(url_for('subject_page', subject=subject))
    return render_template('ebook-buttons.html')

# Subject Page (Example with Mathematics)
@app.route('/<subject>.html')
def subject_page(subject):
    return render_template(f'{subject}.html')

```

**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.

- Request Routes:

```

# Book Request Form Page
@app.route('/request-form', methods=['GET', 'POST'])
def request_form():
    if request.method == 'POST':
        # Retrieve form data from the POST request
        email = request.form['email'] # Capture email to send thank-you note
        name = request.form['name']
        year = request.form['year']
        semester = request.form['semester']
        roll_no = request.form['roll-no']
        subject = request.form['subject']
        book_name = request.form['book-name']
        description = request.form['description']

        # Store book request in DynamoDB along with the user email
        requests_table.put_item(
            Item={
                'email': email,
                'roll_no': roll_no,
                'name': name,
                'year': year,
                'semester': semester,
                'subject': subject,
                'book_name': book_name,
                'description': description
            }
        )

        # Send a thank-you email to the requesting user
        thank_you_message = f"Dear {name},\n\nThank you for submitting a book request for '{book_name}'. We will"
        send_email(email, "Thank You for Your Book Request", thank_you_message)

        # Send an email to the Instant Library admin with the book request details
        admin_message = f"User {name} ({email}) has requested the book '{book_name}'.\n\nDetails:\nYear: {year}\n"
        send_email("instantlibrary2@gmail.com", "New Book Request", admin_message)

    return "<h3>Book request submitted successfully! We will get back to you soon.</h3>"

    # Render the request form for GET requests
    return render_template('request-form.html')

```

**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.

### Exit Route:

```

# Exit Page
@app.route('/exit')
def exit_page():
    return render_template('exit.html')

```

**Description:** define /exit 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)

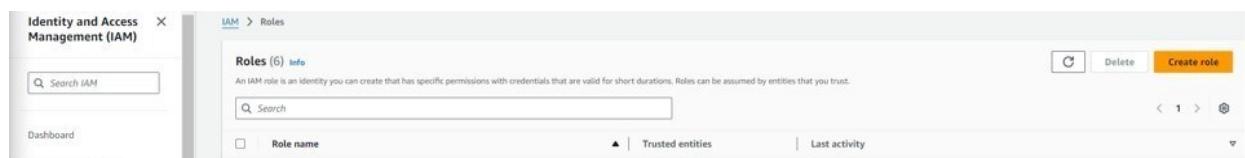
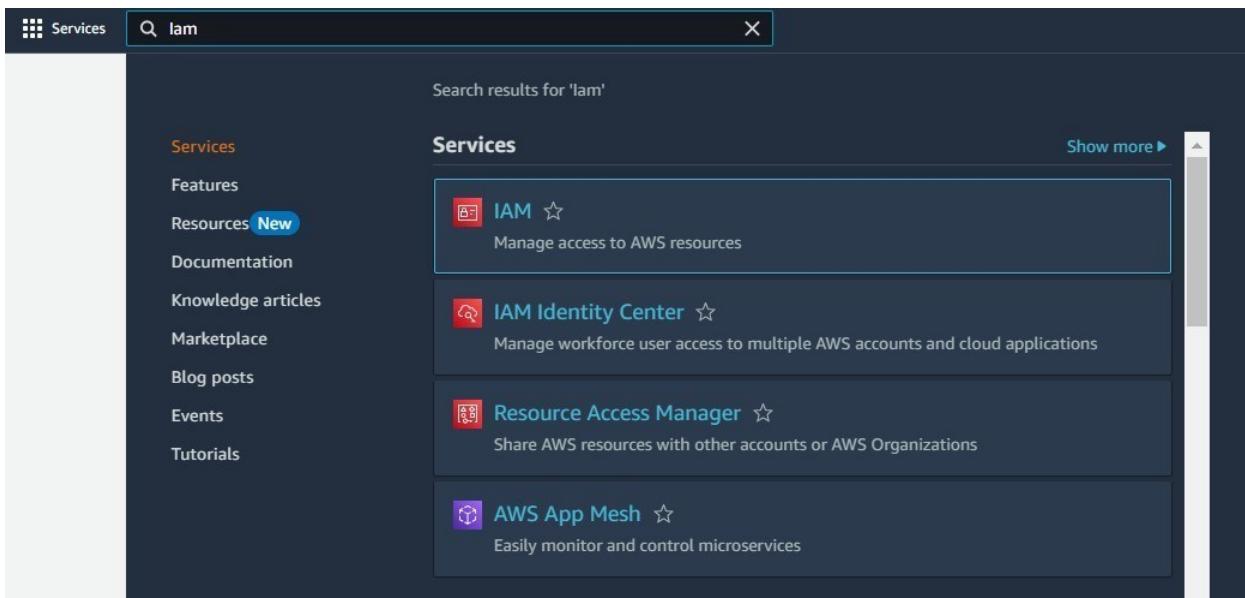
```

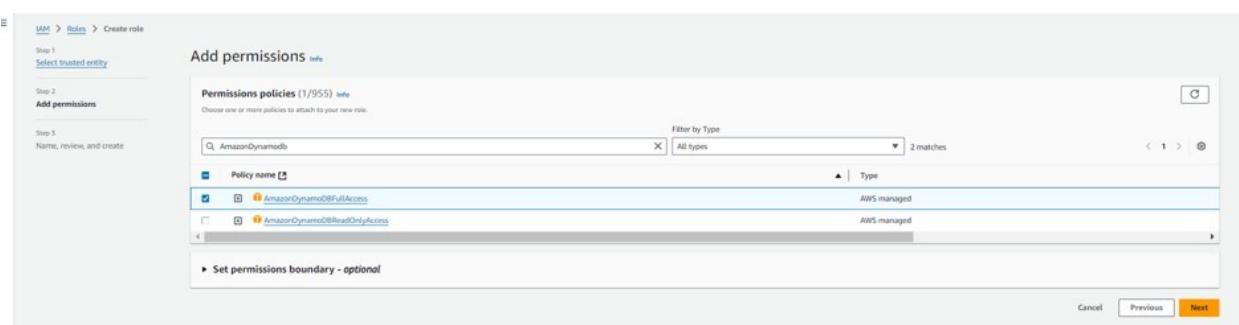
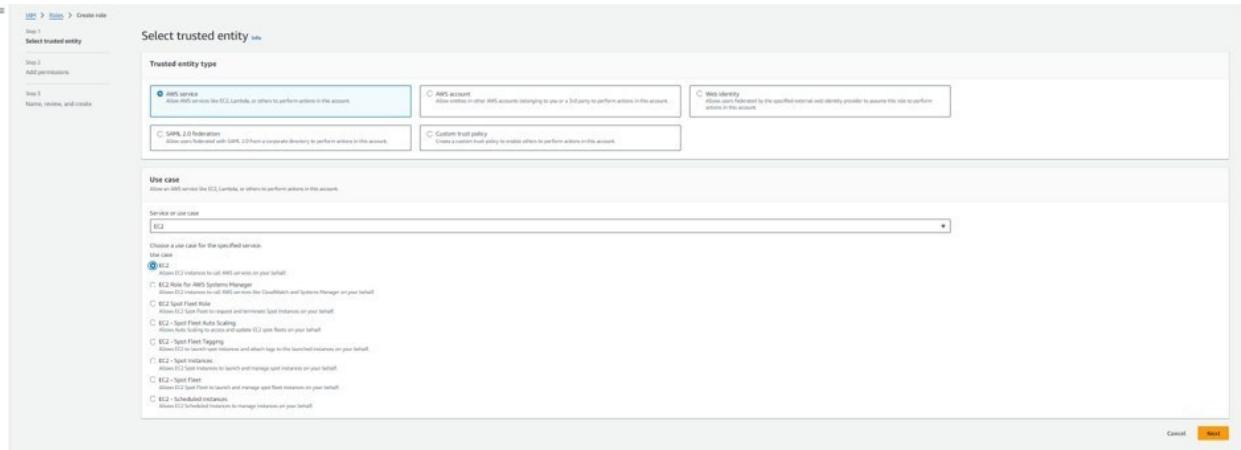
**Description:** start the Flask server to listen on all network interfaces (0.0.0.0) at port 80 with debug mode enabled for development and testing.

## 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.





## ● 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.

**Add permissions**

Permissions policies (2/955) Info

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

Filter by Type: All types

Policy name: sns

Policy name	Type
AmazonSNSFullAccess	AWS managed
AmazonSNSReadOnlyAccess	AWS managed
AmazonSNSRole	AWS managed
AWSLambdaBasicExecutionRole	AWS managed
AWSIoTDeviceDefenderPublishFindingsToSNSSignatureAction	AWS managed

Set permissions boundary - optional

Cancel Previous Next

**Name, review, and create**

**Role details**

Role name: sns\_DynamicallyRole

Description: Allows EC2 instances to call AWS services on your behalf.

Trust policy:

```

1: "Version": "2012-10-17",
2: "Statement": [
3:     {
4:         "Effect": "Allow",
5:         "Principal": "*",
6:         "Action": "sts:AssumeRole"
7:     }
8: ],
9: "Statement": [
10:     {
11:         "Effect": "Allow",
12:         "Action": "sns:Publish"
13:     }
14: ]
15: }

```

**Step 1: Select trusted entities**

**Step 2: Add permissions**

**Permissions policy summary**

Policy name	Type	Attached as
AmazonSNSFullAccess	AWS managed	Permissions policy
AmazonSNSRole	AWS managed	Permissions policy

**Step 3: Add tags**

Add tags - optional Info

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

No tags associated with this resource.

Add new tag

You can add up to 50 more tags.

Create role Cancel Previous Create 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)

Last activity: 6 days ago

ARN: arn:aws:iam::557690616836:role/sns\_Dynamodb\_role

Maximum session duration: 1 hour

Instance profile ARN: arn:aws:iam::557690616836:instance-profile/sns\_Dynamodb\_role

**Permissions** | Trust relationships | Tags | Last Accessed | Revoke sessions

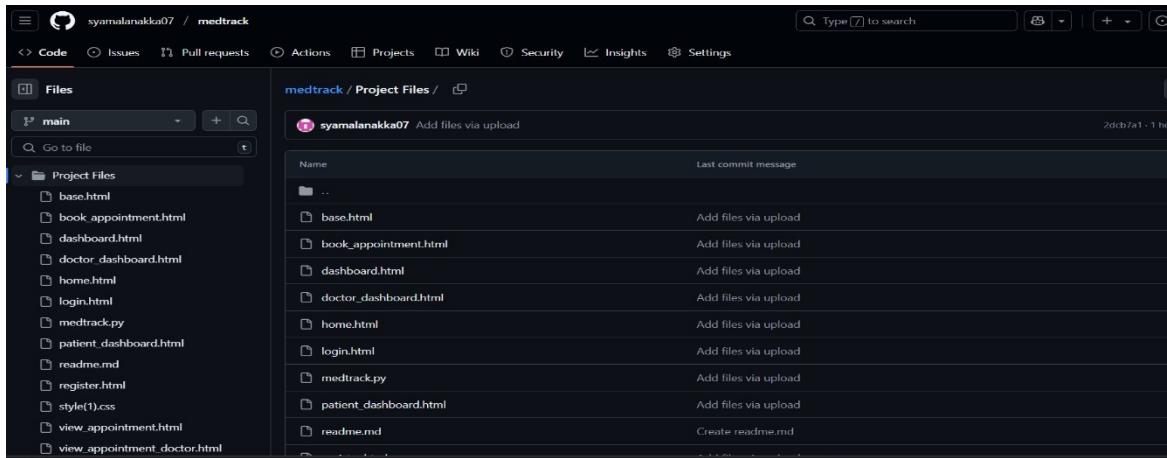
**Permissions policies (2) Info**

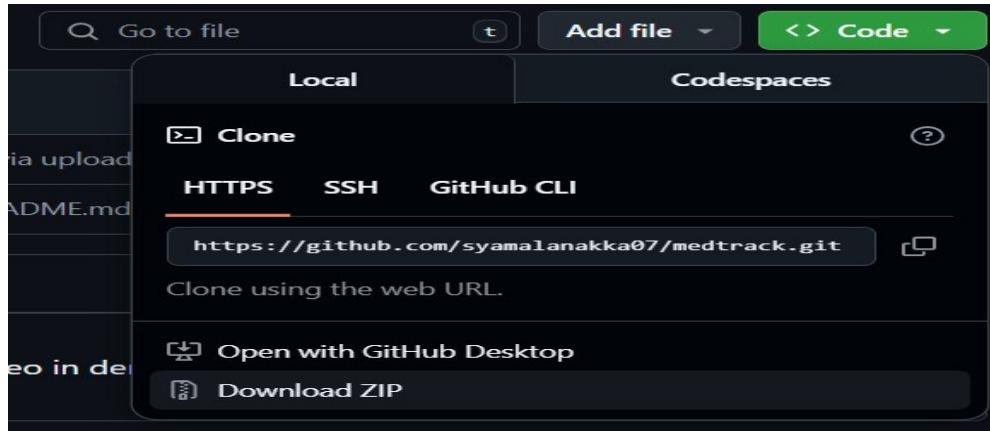
You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonDynamoDBFullAccess	AWS managed	4
AmazonSNSFullAccess	AWS managed	2

## Milestone 6: EC2 Instance Setup

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

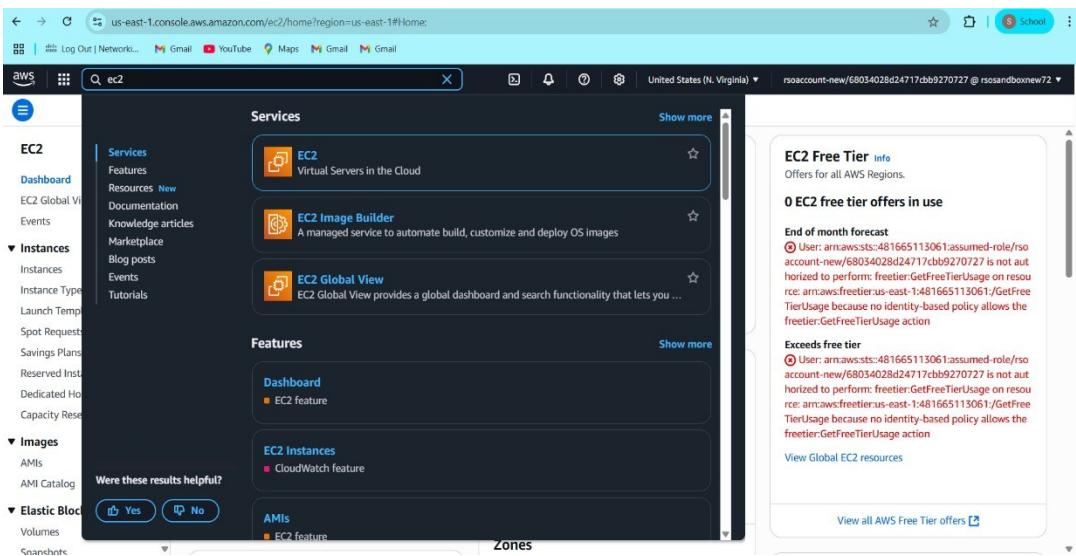




- **Activity 6.1: Launch an EC2 instance to host the Flask application.**

- **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 home page. On the left, there's a sidebar with navigation links: Dashboard, EC2 Global View, Events, Instances (with sub-links: Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), Images (AMIs, AMI Catalog), and Elastic Block Store (Volumes, Snapshots). The main content area has a dark header "Compute" and a large title "Amazon Elastic Compute Cloud (EC2)". Below it, a sub-section says "Create, manage, and monitor virtual servers in the cloud." A paragraph describes EC2's offerings. To the right, there's a callout box titled "Launch a virtual server" with "Launch instance" and "View dashboard" buttons. Another callout box titled "Get started" contains "Get started walkthroughs" and "Get started tutorial".

This screenshot shows the "Launch an instance" wizard, step 1: Set instance details. At the top, there's a message: "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 form has sections for "Name and tags" (with a "Name" input field containing "e.g. My Web Server" and an "Add additional tags" button), "Application and OS Images (Amazon Machine Image)" (with a search bar and a "Quick Start" dropdown menu showing options: Amazon, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, Debian), and "Summary" (with a "Number of instances" input field set to "1"). On the right, there are "Cancel", "Launch instance", and "Preview code" buttons.

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

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with links for AWS Lambda, AWS Lambda Functions, AWS Lambda Policies, AWS Lambda Metrics, AWS Lambda Traces, and AWS Lambda Logs. Below the navigation bar, there's a search bar and a 'Create Function' button. The main content area is titled 'Amazon Machine Image (AMI)' and shows a list of available AMIs:

- Amazon Linux (selected)
- macOS
- Ubuntu
- Windows
- Red Hat
- [More]

To the right of the AMI list is a search icon and a link to 'Browse more AMIs'. Below the AMI list, there's a note: 'Including AMIs from AWS, Marketplace and the Community'.

Below the AMI list, there's a detailed view for 'Amazon Linux 2023 AMI':

Amazon Linux 2023 AMI	Free tier eligible
ami-02b49a24cfb95941c (64-bit (x86), uefi-preferred) / ami-04ad8c7fcc828fad4 (64-bit (Arm), uefi)	▼
Virtualization: hvm	ENI enabled: true
	Root device type: ebs

Below this, there's a 'Description' section with the following text:

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.

At the bottom, there are filter options for 'Architecture' (set to '64-bit (x86)'), 'Boot mode' (set to 'uefi-preferred'), and 'AMI ID' (set to 'ami-02b49a24cfb95941c'). There's also a green 'Verified provider' badge.

- Create and download the key pair for Server access.

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

Instance type

t2.micro	Free tier eligible		
Family: t2	1 vCPU	1 GiB Memory	Current generation: true
On-Demand Linux base pricing: 0.0124 USD per Hour			
On-Demand Windows base pricing: 0.017 USD per Hour			
On-Demand RHEL base pricing: 0.0268 USD per Hour			
On-Demand SUSE base pricing: 0.0124 USD per Hour			

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

**▼ Key pair (login)** [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

Select [Create new key pair](#)

**Create key pair**

Key pair name

Key pairs allow you to connect to your instance securely.

InstantLibrary

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

RSA  
RSA encrypted private and public key pair

ED25519  
ED25519 encrypted private and public key pair

Private key file format

.pem  
For use with OpenSSH

.ppk  
For use with PuTTY

**⚠ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#)**

Cancel [Create key pair](#)



InstantLibrary.pem

The screenshot shows the AWS Launch Wizard interface for creating a new Amazon Linux 2023 instance. The process is at Step 2: Set instance details.

**Description:** Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

**Architecture:** 64-bit (x86) | **Boot mode:** uefi-preferred | **AMI ID:** ami-078264b8ba71bc45e | **Username:** ec2-user | **Verified provider:** Verified provider

**Instance type:** t2.micro | **Free tier eligible:** Yes | **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

**Key pair (login):** InstantLibrary | **Create new key pair:** Create new key pair

**Summary:** Number of instances: 1 | Software Image (AMI): Amazon Linux 2023 AMI 2023.5.2... | 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.

Buttons: 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) [▼](#) [C](#)

Subnet [Info](#)

No preference [▼](#) [C](#) 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>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>
ssh	TCP	22
Source type <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional <a href="#">Info</a>
Anywhere	Add CIDR, prefix list or security	e.g. SSH for admin desktop
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>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>
HTTP	TCP	80
Source type <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional <a href="#">Info</a>
Custom	Add CIDR, prefix list or security	e.g. SSH for admin desktop
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>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>
Custom TCP	TCP	5000
Source type <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional <a href="#">Info</a>
Custom	Add CIDR, prefix list or security	e.g. SSH for admin desktop
0.0.0.0/0 <a href="#">X</a>		

[Add security group rule](#)

The screenshot shows the AWS EC2 'Launch an instance' success page. At the top, a green banner displays the message 'Successfully initiated launch of instance i-001861022fbcac290'. Below the banner, there's a 'Launch log' button. The main area is titled 'Next Steps' with a sub-section 'What would you like to do next with this instance, for example "create alarm" or "create backup"'.

The 'Next Steps' section contains several cards:

- Create billing and free tier usage alerts**: To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds. Includes a 'Create billing alerts' button.
- Connect to your instance**: Once your instance is running, log into it from your local computer. Includes 'Connect to instance' and 'Learn more' buttons.
- Connect an RDS database**: Configure the connection between an EC2 instance and a database to allow traffic flow between them. Includes 'Connect an RDS database' and 'Learn more' buttons.
- Create EBS snapshot policy**: Create a policy that automates the creation, retention, and deletion of EBS snapshots. Includes a 'Create EBS snapshot policy' button.
- Manage detailed monitoring**: 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. Includes a 'Manage detailed monitoring' button.
- Create Load Balancer**: Create a application, network gateway or classic Elastic Load Balancer. Includes a 'Create Load Balancer' button.
- Create AWS budget**: AWS Budgets allows you to create budgets, forecast spend, and take action on your costs and usage from a single location. Includes a 'Create AWS budget' button.
- Manage CloudWatch alarms**: Create or update Amazon CloudWatch alarms for the instance. Includes a 'Manage CloudWatch alarms' button.
- Disaster recovery for your instances**: Recover the instances you just launched into a different Availability Zone or a different Region using AWS Elastic Disaster Recovery (DRS). Includes a 'Disaster recovery for your instances' button.
- Monitor for suspicious runtime activities**: 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. Includes a 'Monitor for suspicious runtime activities' button.
- Get instance screenshot**: Capture a screenshot from the instance and view it as an image. This is useful for troubleshooting an unreachable instance. Includes a 'Get instance screenshot' button.
- Get system log**: View the instance's system log to troubleshoot issues. Includes a 'Get system log' button.

At the bottom right, there's a 'View all instances' button.

- 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.

The screenshot shows the AWS EC2 'Instances (1/2)' list. The instance 'InstantLibraryApp' (i-001861022fbcac290) is listed with the following details:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring	Security
InstantLibrary...	i-001861022fbcac290	Stopped	t2.micro	-	View alarms +	ap-south-1b	-	-	-	-	disabled	launch-wit...

The screenshot shows the 'Instance summary for i-001861022fbcac290 (InstantLibraryApp)' page. The instance was updated less than a minute ago. The summary includes the following details:

Instance ID	Public IPv4 address	Private IPv4 addresses
i-001861022fbcac290	-	172.31.5
IPv6 address	Public IPv4 DNS	Elastic IP addresses
-	-	-
Hostname type	Private IP DNS name (IPv4 only)	AWS Compute Optimizer finding
IP name: ip-172-31-5-5.ap-south-1.compute.internal	ip-172-31-5-5.ap-south-1.compute.internal	Opt-in to AWS Compute Optimizer for recommendations.   Learn more
Answer private resource DNS name	Instance type	Auto Scaling Group name
IPv4 (A)	t2.micro	-
Auto-assigned IP address	VPC ID	
-	vpc-03cdc7b6f19dd7211	
IAM Role	Subnet ID	
sns_Dynamodb_role	subnet-0d9fa5144480cc9a9	
IMDSv2 Required	Instance ARN	
	arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290	

At the bottom, there are tabs for 'Details', 'Status and alarms', 'Monitoring', 'Security', 'Networking', 'Storage', and '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	172.31.3.5
IPv6 address	-	Instance state	Stopped
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
Answer private resource DNS name	IPV4 (A)	Instance type	t2.micro
Auto-assigned IP address	-	VPC ID	vpc-03cdc7b6f19dd7211
IAM Role	<a href="#">sns_Dynamodb_role</a>	Subnet ID	subnet-0d9fa3144480cc9a9
IMDsv2	Required	Instance ARN	arn:aws:ec2:ap-south-1:1557690616836:instance/i-001861022fbcac290

**Actions ▾**

- Connect
- Instance state ▾
- Actions ▾
- Manage instance state
- Instance settings
- Networking
- Security** ▾
- Change security groups
- Get Windows password
- Modify IAM role**
- Image and templates
- Monitor and troubleshoot

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.

[sns\\_Dynamodb\\_role](#) [▼](#) [Create new IAM role](#)

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

- Now connect the EC2 with the files

## Connect to instance Info

Connect to your instance i-001861022fbcac290 (InstantLibraryApp) using any of these options

[EC2 Instance Connect](#)

[Session Manager](#)

[SSH client](#)

[EC2 serial console](#)



### Port 22 (SSH) is open to all IPv4 addresses

Port 22 (SSH) is currently open to all IPv4 addresses, indicated by **0.0.0.0/0** in the inbound rule in [your security group](#). For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: 13.233.177.0/29. [Learn more.](#)

Instance ID

i-001861022fbcac290 (InstantLibraryApp)

Connection Type

Connect using EC2 Instance Connect

Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.

Connect using EC2 Instance Connect Endpoint

Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Public IPv4 address

13.200.229.59

IPv6 address

—

Username

Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.

ec2-user

**Note:** In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

[Cancel](#)

[Connect](#)

```
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
 https://aws.amazon.com/linux/amazon-linux-2023
Last login: Tue Oct 15 04:17:59 2024 from 13.233.177.3
[ec2-user@ip-172-31-3-5 ~]$
```

i-001861022fbcac290 (InstantLibraryApp)

Public IPs: 13.201.74.42 Private IPs: 172.31.3.5

## 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/syamalanakka07/medtrack.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
      +----- https://aws.amazon.com/linux/amazon-linux-2023
      V-' '--->
      /----- /
      /m'----- /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]$ 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

```

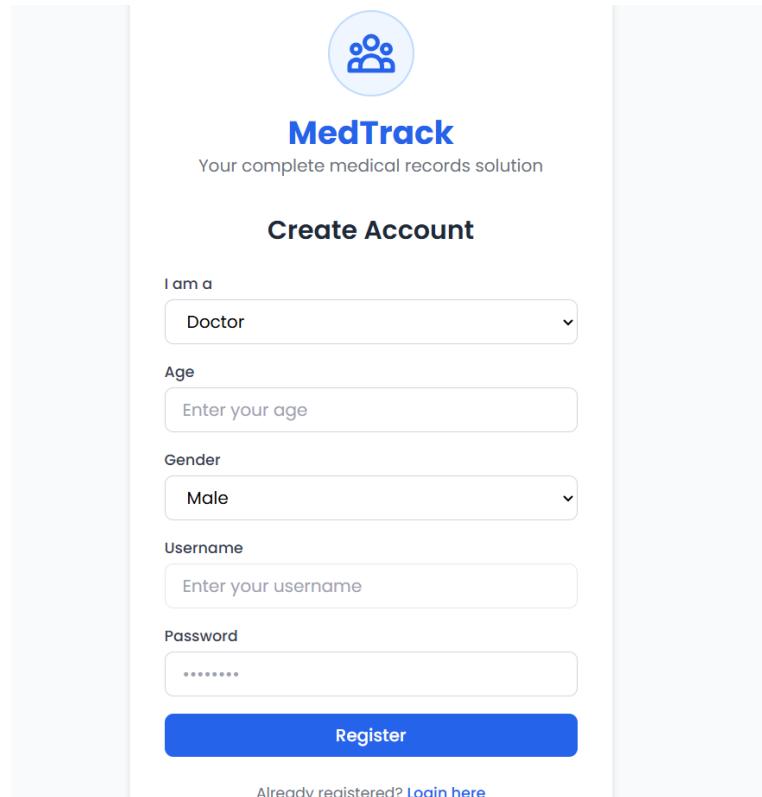
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running 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 -

```

## Milestone 8: Testing and Deployment

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

Register Page:



The image shows the 'Create Account' page for MedTrack. At the top right is a circular icon with three stylized human figures. Below it, the MedTrack logo is displayed with the tagline 'Your complete medical records solution'. A large 'Create Account' button is centered below the logo. The form consists of several input fields: 'I am a' (dropdown menu showing 'Doctor'), 'Age' (text input placeholder 'Enter your age'), 'Gender' (dropdown menu showing 'Male'), 'Username' (text input placeholder 'Enter your username'), and 'Password' (text input placeholder '\*\*\*\*\*'). At the bottom is a large blue 'Register' button. Below the button, a link says 'Already registered? [Login here](#)'.

I am a  
Doctor

Age  
Enter your age

Gender  
Male

Username  
Enter your username

Password  
\*\*\*\*\*

Register

Already registered? [Login here](#)

Login Page:



**Welcome Back**

Please enter your credentials to access your account

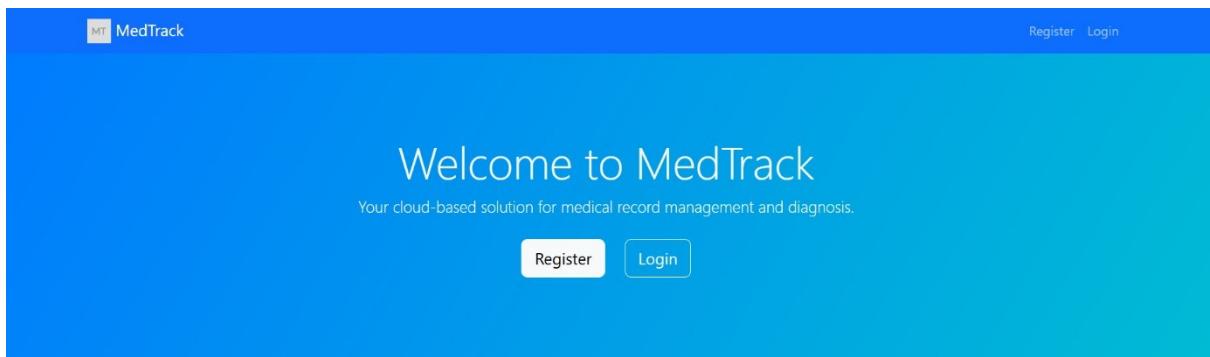
[Forgot password?](#)

Remember me

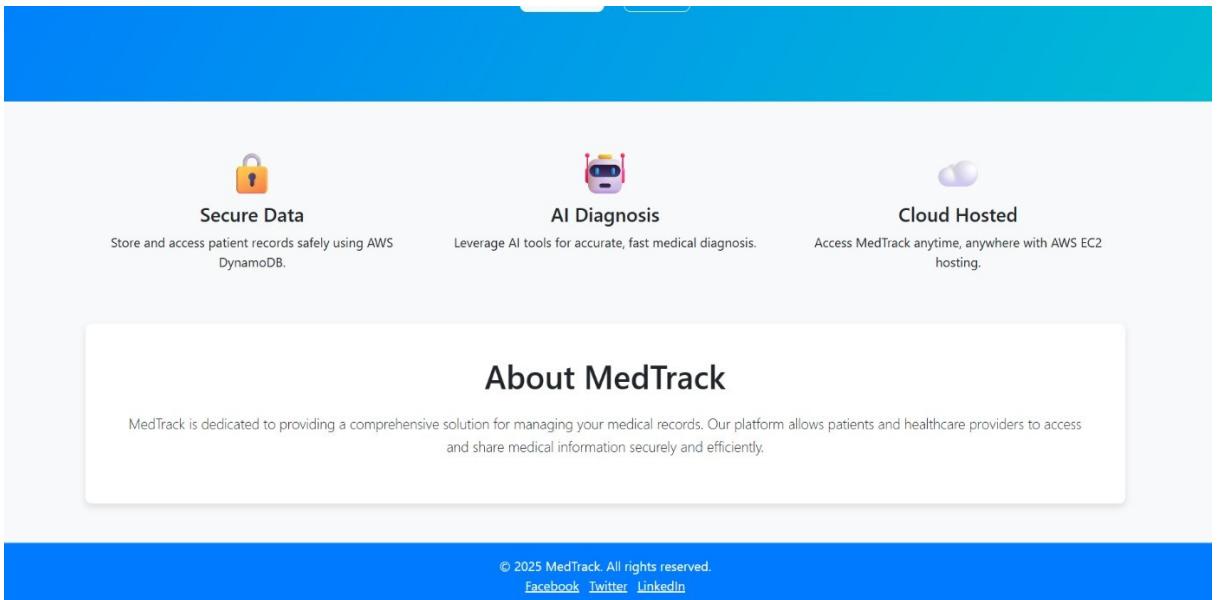
Sign In

Don't have an account? [Sign up](#)

### Home page:



### About Us page:



The image shows the MedTrack landing page. At the top, there's a blue header bar. Below it, a white section features three main service icons: 'Secure Data' (lock icon), 'AI Diagnosis' (robot icon), and 'Cloud Hosted' (cloud icon). Each service has a brief description below its icon. A central callout box is titled 'About MedTrack' and contains a paragraph about the platform's purpose. At the bottom of the page is a blue footer bar with copyright and social media links.

**Secure Data**  
Store and access patient records safely using AWS DynamoDB.

**AI Diagnosis**  
Leverage AI tools for accurate, fast medical diagnosis.

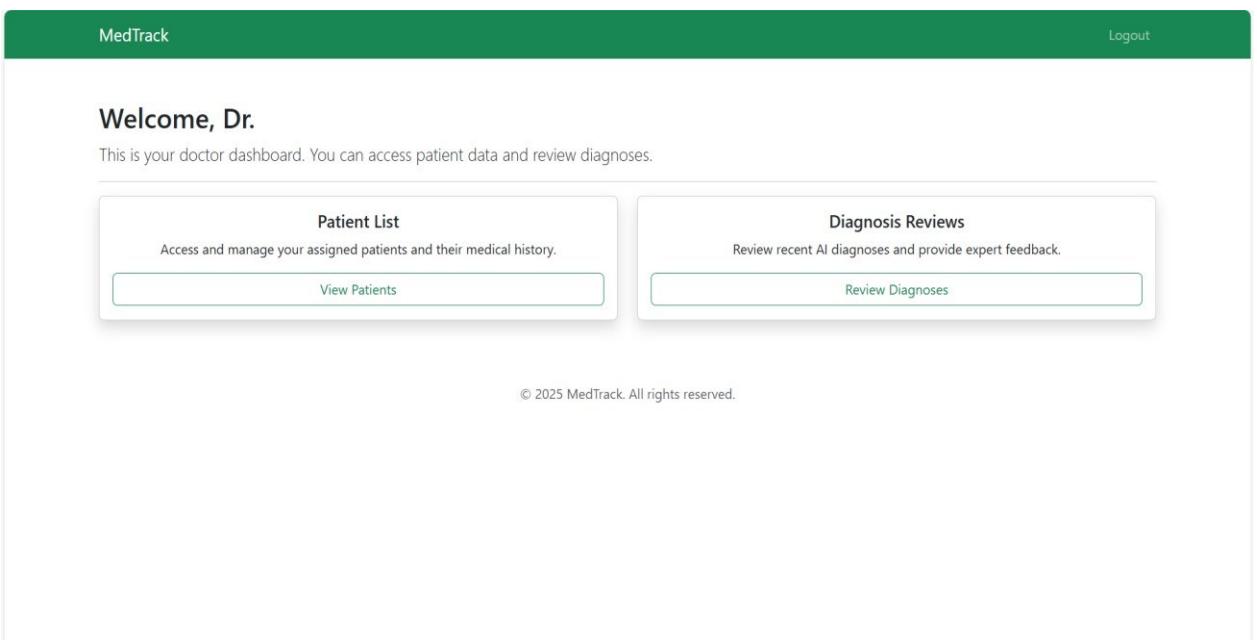
**Cloud Hosted**  
Access MedTrack anytime, anywhere with AWS EC2 hosting.

**About MedTrack**

MedTrack is dedicated to providing a comprehensive solution for managing your medical records. Our platform allows patients and healthcare providers to access and share medical information securely and efficiently.

© 2025 MedTrack. All rights reserved.  
[Facebook](#) [Twitter](#) [LinkedIn](#)

## Doctor Dashboard:



The image shows the Doctor Dashboard. It has a green header bar with the 'MedTrack' logo on the left and a 'Logout' link on the right. The main content area starts with a welcome message 'Welcome, Dr.' followed by a descriptive text. Below this are two cards: 'Patient List' (with a 'View Patients' button) and 'Diagnosis Reviews' (with a 'Review Diagnoses' button). At the bottom of the dashboard is a small copyright notice.

**Welcome, Dr.**

This is your doctor dashboard. You can access patient data and review diagnoses.

**Patient List**  
Access and manage your assigned patients and their medical history.  
[View Patients](#)

**Diagnosis Reviews**  
Review recent AI diagnoses and provide expert feedback.  
[Review Diagnoses](#)

© 2025 MedTrack. All rights reserved.

## Patient DashBoard:

MedTrack

Logout

Welcome,

This is your patient dashboard.

**View Medical Records**  
Access your stored medical history securely.  
[View Records](#)

**Run Diagnosis**  
Use AI tools to run a new diagnosis.  
[Start Diagnosis](#)

© 2025 MedTrack. All rights reserved.

## Exit:

## Session Ended

Please close this tab.

## **Conclusion:**

**MedTrack** is a web-based healthcare management system built using Flask that facilitates interaction between doctors and patients. It allows users to register and log in as either a doctor or a patient. Patients can view available doctors, book appointments, and view their scheduled or past visits. Doctors can access their assigned appointments, update them with a diagnosis, treatment plan, and prescription, and mark them as completed. The system uses in-memory storage for users and appointments, making it suitable for demonstration or development purposes without requiring a database. Although email and AWS SNS notification features are present in the code, they are disabled by default. MedTrack is designed with a clear role-based workflow and is easily extendable for real-world deployment with additional features like persistent databases, email alerts, password encryption, and a modern UI.