

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.

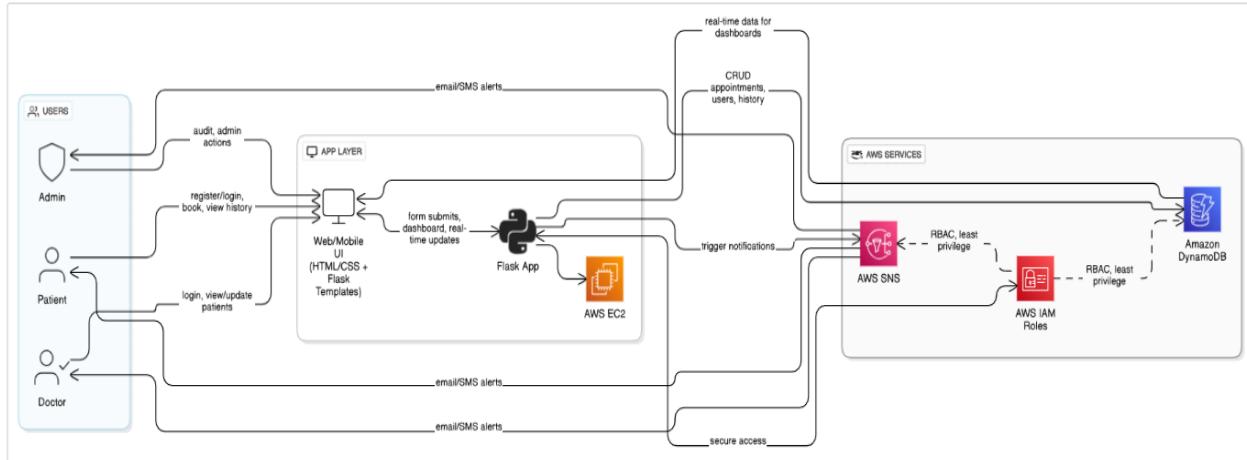
Scenario 2: Secure User Management with IAM

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.

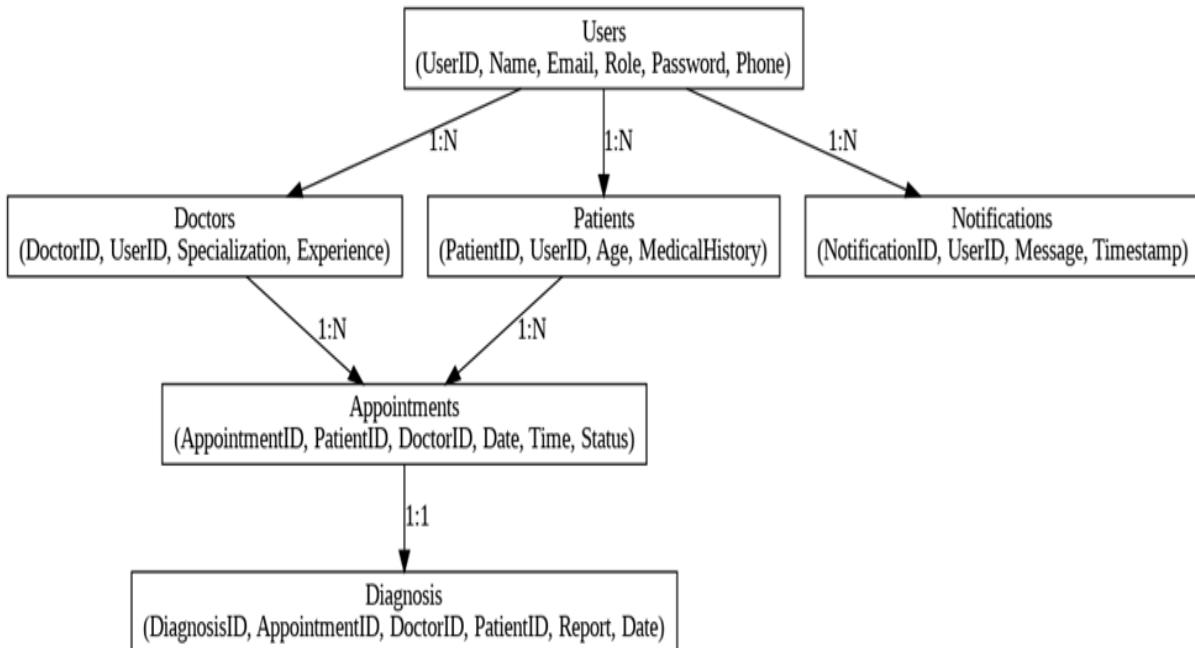
Scenario 3: Easy Access to Medical History and Resources

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

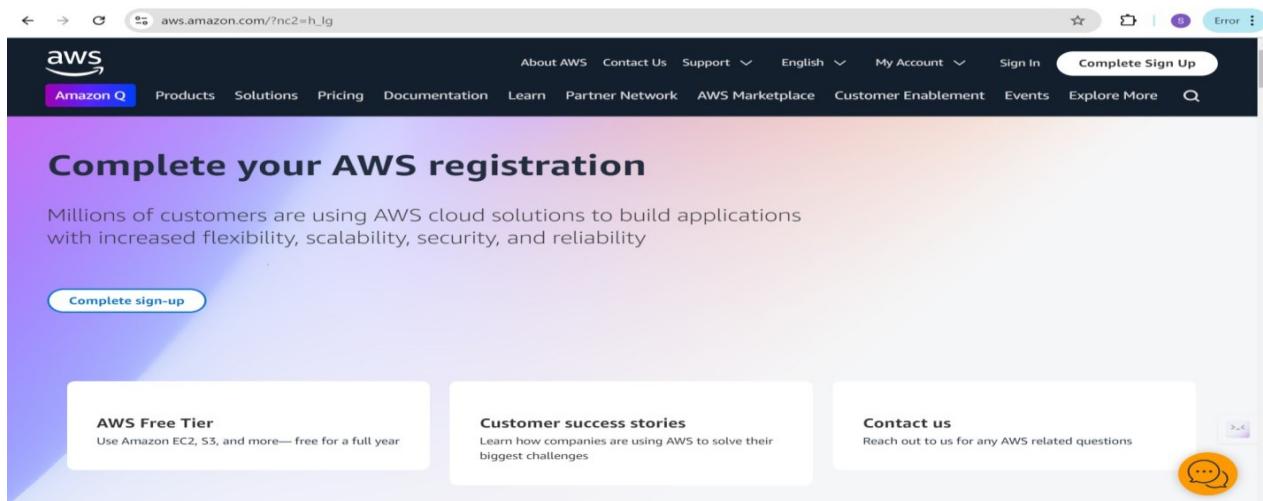
Activity 7.2: Run the Flask App

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](#).

A screenshot of the AWS sign-in interface. It features the AWS logo at the top left. Below it, there's a 'Sign in' section with two radio button options: 'Root user' (selected) and 'IAM user'. The 'Root user' option is described as an 'Account owner that performs tasks requiring unrestricted access'. Below this is a field for 'Root user email address' containing 'username@example.com' and a 'Next' button. At the bottom, there's a small note about cookie consent. To the right of the sign-in form is a dark purple sidebar with white text. The title is 'AI Use Case Explorer'. The sidebar text reads: '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 the text "dynamoDB". The left sidebar has a "Services" section with various links like Features, Resources (New), Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main content area is titled "Services" and lists several services: **DynamoDB** (Managed NoSQL Database), **Amazon DocumentDB** (Fully-managed MongoDB-compatible database service), **CloudFront** (Global Content Delivery Network), and **Athena** (Serverless interactive analytics service). Below this is another section titled "Features" with "Settings" and "Clusters" listed under "DynamoDB feature".

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, Clusters, Subnet groups, Parameter groups, and Events. The main dashboard has two main sections: "Alarms (0) Info" and "DAX clusters (0) Info". Both sections have search bars and "View details" buttons. To the right, there is a "Create resources" section with a "Create table" button and a note about the Amazon DynamoDB Accelerator (DAX). Below that is a "What's new" section with a note about AWS Cost Management providing purchase recommendations for Amazon DynamoDB.

The screenshot shows the AWS DynamoDB 'Tables' page. On the left, there's a sidebar with options like 'Dashboard', 'Tables', 'Explore items', 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', and 'Integrations'. The main area has a heading 'Tables (0) Info' with a search bar and filters for 'Any tag key' and 'Any tag value'. A message says 'You have no tables in this account in this AWS Region.' At the bottom right is a large orange 'Create table' button.

- **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.

The screenshot shows the 'Create table' wizard in the AWS DynamoDB console. The top navigation bar shows 'DynamoDB > Tables > Create table'. The main section is titled 'Create table' with a sub-section 'Table details'. It says 'DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.' Under 'Table name', it says 'This will be used to identify your table.' with a field containing 'Users'. Below it says 'Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).'. Under 'Partition key', it says '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.' with a field containing 'email' and a dropdown set to 'String'. Below it says '1 to 255 characters and case sensitive.' Under 'Sort key - optional', it says '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.' with a field containing 'Enter the sort key name' and a dropdown set to 'String'. Below it says '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 X Tables

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.

= [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.

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 settings

Default settings
This feature is to create a new table. You can modify these settings later to make DynamoDB work for you.

Customize settings
Use these advanced features to make DynamoDB work for you.

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 Requests table was created successfully.

DynamoDB > Tables

Tables (2) Info

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'.

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

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

- 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

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.

► **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**.

Amazon SNS

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

Topic BookRequestNotifications created successfully.
 You can create subscriptions and send messages to them from this topic.

Amazon SNS > Topics > BookRequestNotifications

[Edit](#) [Delete](#) [Publish message](#)

BookRequestNotifications

Details

Name	BookRequestNotifications	Display name	-
ARN	arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications	Topic owner	557690616836
Type	Standard		

[Subscriptions](#) [Access policy](#) [Data protection policy](#) [Delivery policy \(HTTP/S\)](#) [Delivery status logging](#) [Encryption](#) [Tags](#) [Integrations](#)

Subscriptions (0)

ID	Endpoint	Status	Protocol
No subscriptions found You don't have any subscriptions to this topic.			

[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.

ⓘ 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.

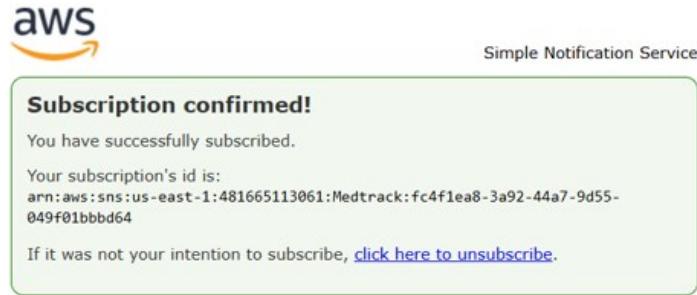
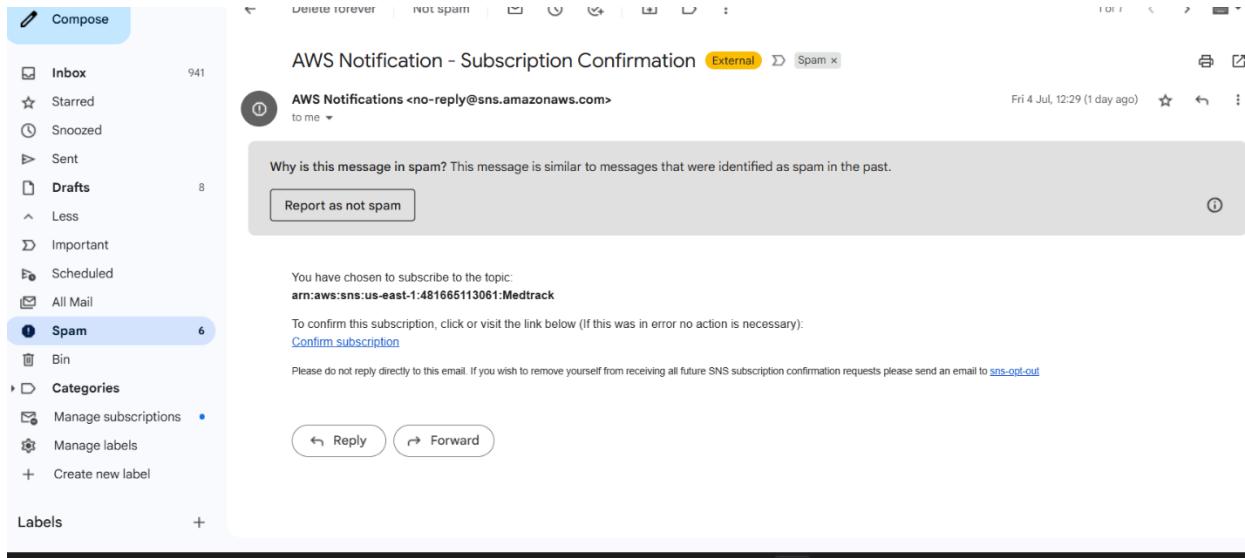
Cancel Create subscription

The screenshot shows the AWS SNS console. A green banner at the top indicates a new feature: "Amazon SNS now supports in-place message archiving and replay for FIFO topics. Learn more." Below this, a success message says "Subscription to BookRequestNotifications created successfully." The ARN of the subscription is listed as arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4. The main section shows the subscription details for "Subscription: d78e0371-9235-404d-952c-85c2743607c4". It includes fields for ARN, Endpoint (instantlibrary2@gmail.com), Topic (BookRequestNotifications), and Subscription Principal (arn:aws:iam::557690616836:root). The status is "Pending confirmation" and the protocol is "EMAIL". Below this, there are tabs for "Subscription filter policy" and "Redrive policy (dead-letter queue)". The "Subscription filter policy" section notes "No filter policy configured for this subscription. To apply a filter policy, edit this subscription." An "Edit" button is available.

- After subscription request for the mail confirmation

The screenshot shows the AWS SNS console. A green banner at the top indicates a new feature: "Amazon SNS now supports in-place message archiving and replay for FIFO topics. Learn more." Below this, a success message says "Confirmation request was sent successfully." The ARN of the subscription is listed as arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:7ae5d16-12ad-4f31-9f76-c342c2215ad5. The main section shows the topic "BookRequestNotifications" with its details: Name (BookRequestNotifications), Display name (-), ARN (arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications), Topic owner (557690616836), and Type (Standard). Below this, there are tabs for "Subscriptions", "Access policy", "Data protection policy", "Delivery policy (HTTP/S)", "Delivery status logging", "Encryption", "Tags", and "Integrations". The "Subscriptions" tab shows a table with one row: ID (Pending confirmation), Endpoint (instantlibrary2@gmail.com), Status (Pending confirmation), and Protocol (EMAIL). Buttons for "Edit", "Delete", "Request confirmation", "Confirm subscription", and "Create subscription" are available above the table. Navigation controls (< 1 > and a refresh icon) are also present.

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



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

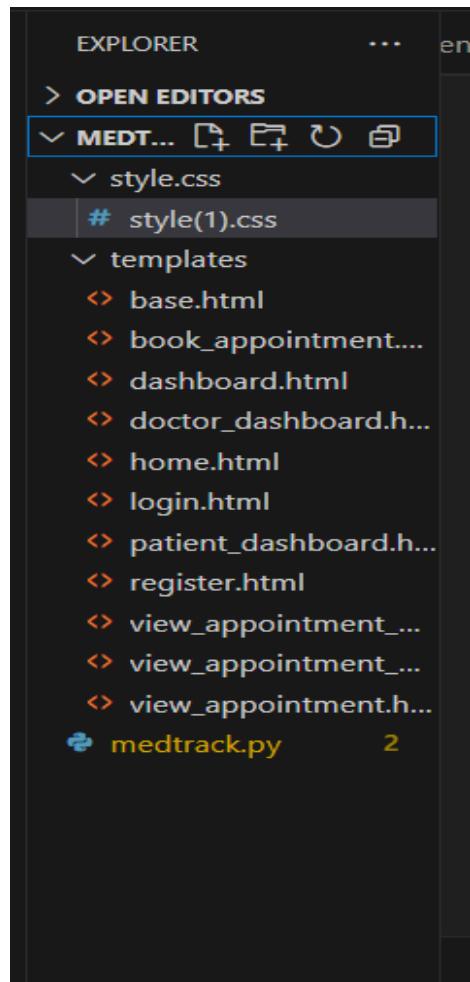
The screenshot shows the Amazon SNS console with the following details:

- Topic:** BookRequestNotifications
- Name:** BookRequestNotifications
- ARN:** arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications
- Type:** Standard
- Subscriptions:** 2 (instantlibrary2@gmail.com - confirmed via EMAIL)

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 'Reques
```

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:

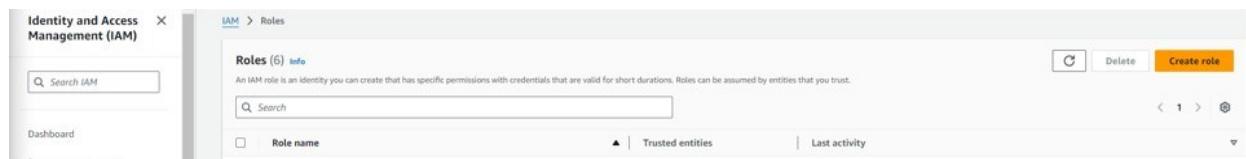
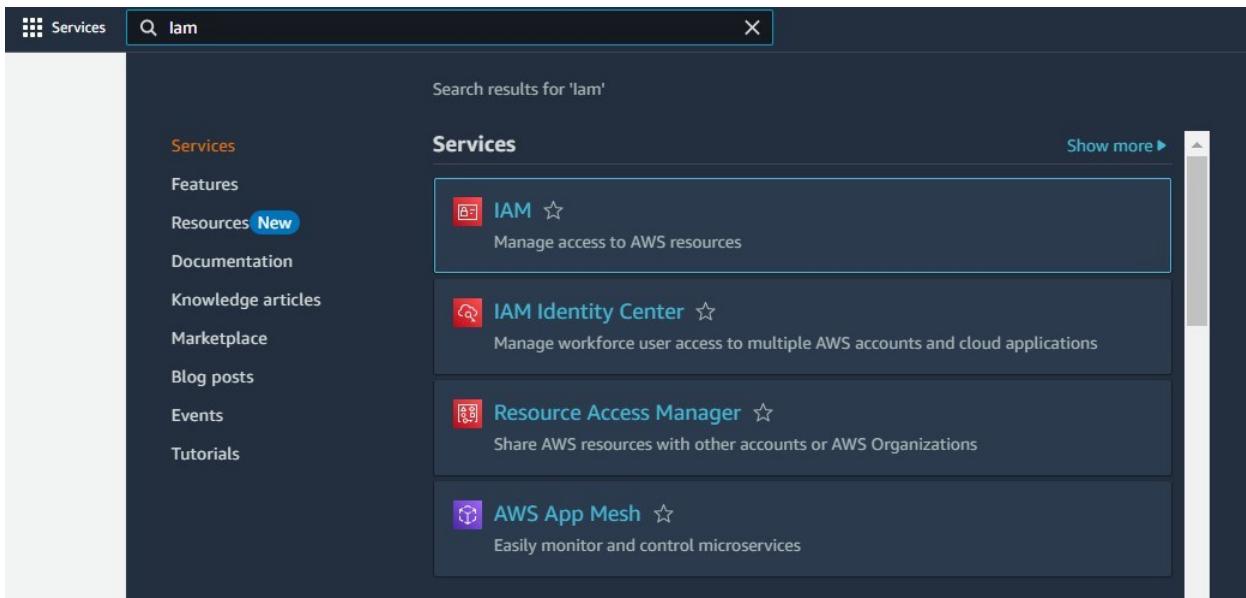
```
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=True)
```

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.



The image contains two screenshots of the AWS IAM 'Create role' wizard.

Top Screenshot (Step 1: Select trusted entity):

- Trusted entity type:**
 - AWS service: Allows AWS services like EC2, Lambda, or others to perform actions in this account.
 - AWS account: Allows access in other AWS accounts belonging to you or a 3rd party to perform actions in the account.
 - Web identity: Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- User case:** Allows an AWS service like EC2, Lambda, or others to perform actions in this account.
 - Service or use case:** EC2
 - User case:**
 - EC2: Allows EC2 instances to call AWS services on your behalf.
 - EC2 Role for AWS Systems Manager: Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.
 - EC2 Spot Fleet: Allows EC2 Spot Fleet to request and terminates Spot Instances on your behalf.
 - EC2 - Spot Fleet Auto Scaling: Allows EC2 Spot Fleet Auto Scaling and Update EC2 spot fleets on your behalf.
 - EC2 - Spot Fleet Tagging: Allows EC2 to search spot instances and attach tags to the launched instances on your behalf.
 - EC2 - Spot Fleet Termination Protection: Allows EC2 Spot Instances to launch and manage user fleet instances on your behalf.
 - EC2 - Spot Fleet
 - EC2 - Scheduled Instances: Allows EC2 Scheduled Instances to manage instances on your behalf.

Bottom Screenshot (Step 2: Add permissions):

- Permissions policies (1/955):** Choose one or more policies to attach to your new role.
 - Policy name:** AmazonDynamoDBFullAccess
 - Type:** AWS managed
 - AmazonDynamoDBReadOnlyAccess**: AWS managed
- Set permissions boundary - optional**

● 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

Step 2 Add permissions

Step 3 Name, review, and create

Add permissions Info

Permissions policies (2/955) Info

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

Filter by Type All types \$ matches

Policy name	Type
<input checked="" type="checkbox"/>  AmazonSNSFullAccess	AWS managed
<input type="checkbox"/>  AmazonSNSReadOnlyAccess	AWS managed
<input type="checkbox"/>  AmazonSNSRole	AWS managed
<input type="checkbox"/>  AWSLambdaBasicExecutionRole	AWS managed
<input type="checkbox"/>  AWSIoTDeviceDefenderPublishFindingsToSNSIntegrationAction	AWS managed

Set permissions boundary - optional

Cancel Previous Next

Name, review, and create

Role details

Role name

Description

Trust policy

```

1: {
2:   "Version": "2012-10-17",
3:   "Statement": [
4:     {
5:       "Effect": "Allow",
6:       "Principal": "*",
7:       "Action": "sts:AssumeRole"
8:     }
9:   ]
10: }
11: }
12: }
13: }
14: }
15: }
16: }
17: }
18: }
19: }
20: }
21: }
22: }
23: }
24: }
25: }
26: }
27: }
28: }
29: }
30: }
31: }
32: }
33: }
34: }
35: }
36: }

```

Step 2: Add permission summary

Permissions policy summary

Policy name	Type	Attached as
<input type="checkbox"/>  AmazonSNSFullAccess	AWS managed	Permissions policy
<input type="checkbox"/>  AmazonSNSReadOnlyAccess	AWS managed	Permissions policy

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

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 <small>6 days ago</small>	Maximum session duration 1 hour	

Permissions Edit

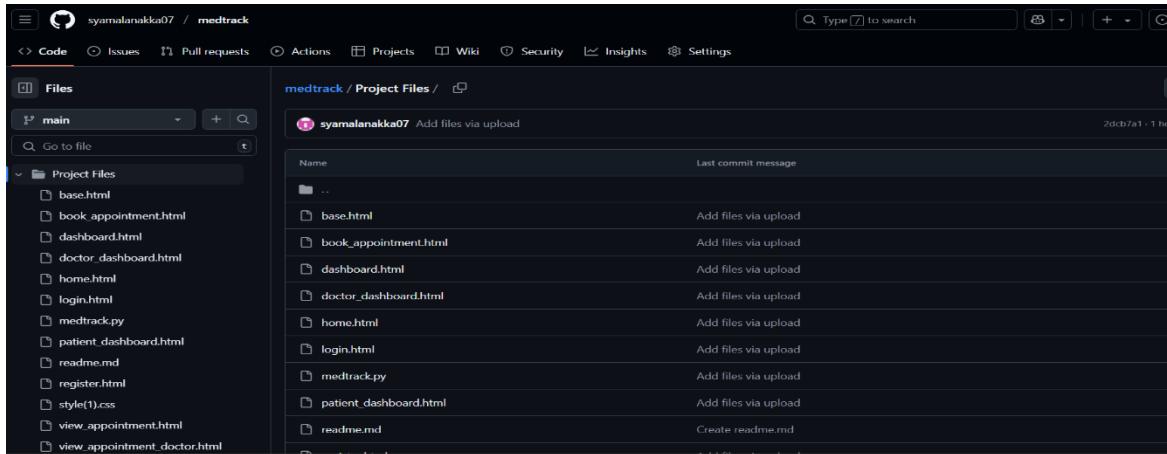
Permissions policies (2) Info

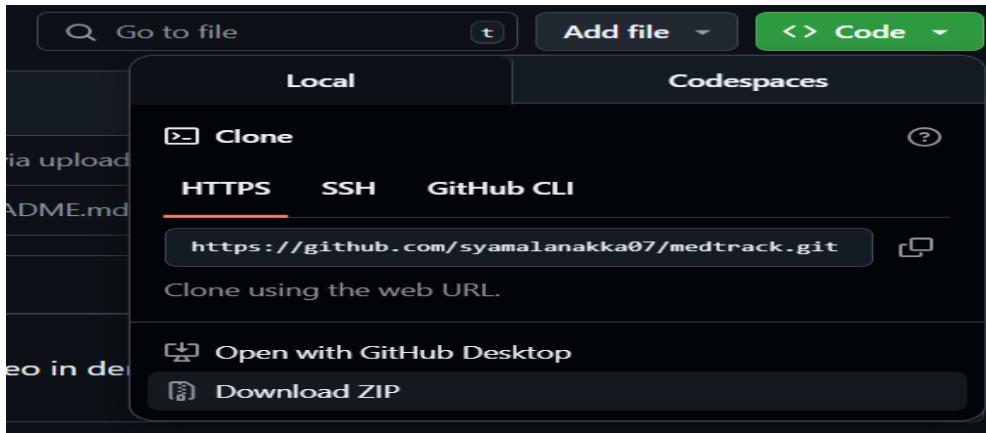
You can attach up to 10 managed policies.

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.

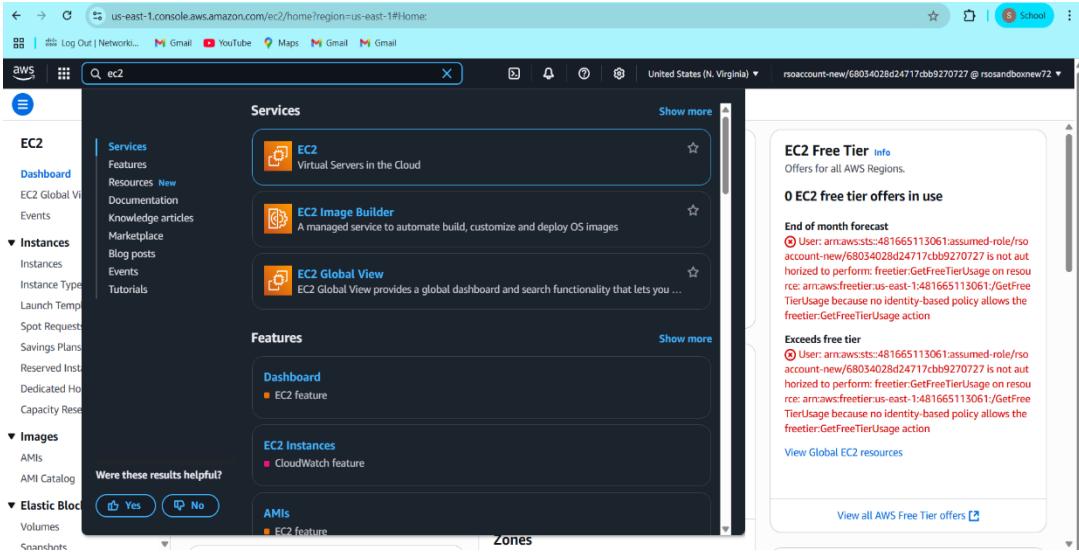




- **Activity 6.1: Launch an EC2 instance to host the Flask application.**

- **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. The left sidebar has a navigation menu with sections like EC2, Instances, Images, and Elastic Block Store. The main content area features a large banner for 'Amazon Elastic Compute Cloud (EC2)' with the subtext 'Create, manage, and monitor virtual servers in the cloud.' Below the banner, there's a section titled 'Benefits and features' with a sub-section 'EC2 offers ultimate scalability and control'. To the right, there's a 'Launch a virtual server' box with 'Launch instance' and 'View dashboard' buttons, and a 'Get started' box with 'Get started walkthroughs' and 'Get started tutorial' buttons.

This screenshot shows the 'Launch an instance' wizard. Step 1 is 'Set instance details'. It includes fields for 'Name and tags' (with a 'Name' input field containing 'e.g. My Web Server'), 'Application and OS Images (Amazon Machine Image)', and 'Quick Start' (with a search bar and a row of OS options: Amazon, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, Debian). On the right, there's a 'Summary' panel showing 'Number of instances' (1), 'Software Image (AMI)' (selected), 'Virtual server type (instance type)' (selected), 'Firewall (security group)' (selected), and 'Storage (volumes)' (selected). At the bottom 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 Marketplace interface for selecting an Amazon Machine Image (AMI). At the top, there are icons for Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and a placeholder for more options. To the right is a search icon and a link to "Browse more AMIs". Below this, a section titled "Amazon Machine Image (AMI)" displays the "Amazon Linux 2023 AMI". This section includes the AMI ID (ami-02b49a24cfb95941c), boot mode (uefi-preferred), and root device type (ebs). A "Free tier eligible" badge is present. A detailed description of the AMI follows, stating it is a modern, general purpose Linux-based OS with 5 years of support, optimized for AWS. Below the description are filters for Architecture (set to 64-bit (x86)), Boot mode (uefi-preferred), and AMI ID (ami-02b49a24cfb95941c), along with a "Verified provider" badge.

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI

Free tier eligible

ami-02b49a24cfb95941c (64-bit (x86), uefi-preferred) / ami-04ad8c7fcc828fad4 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

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-02b49a24cfb95941c Verified provider

- 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 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

Er

⚠ 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](#)



InstantLibrary.pem

The screenshot shows the AWS Launch Wizard interface for creating a new Amazon Linux 2023 instance. The configuration steps are as follows:

- 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)
- Instance type:** t2.micro (Family: t2)
Detailed description:
t2.micro
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
Additional costs apply for AMIs with pre-installed software
- Key pair (login):** InstantLibrary
- Summary:** Number of instances: 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 information: 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) [Edit](#)

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

Auto-assign public IP [Info](#)
 Enable [Edit](#)

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 [Edit](#) Select existing security group [Edit](#)

Security group name - required
 launch-wizard [Edit](#)

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 [Edit](#)

Inbound Security Group Rules

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

Type Info ssh Edit	Protocol Info TCP Edit	Port range Info 22 Edit
Source type Info Anywhere Edit	Source Info Add CIDR, prefix list or security	Description - optional Info e.g. SSH for admin desktop Edit
0.0.0.0/0 X		

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

Type Info HTTP Edit	Protocol Info TCP Edit	Port range Info 80 Edit
Source type Info Custom Edit	Source Info Add CIDR, prefix list or security	Description - optional Info e.g. SSH for admin desktop Edit
0.0.0.0/0 X		

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

Type Info Custom TCP Edit	Protocol Info TCP Edit	Port range Info 5000 Edit
Source type Info Custom Edit	Source Info Add CIDR, prefix list or security	Description - optional Info e.g. SSH for admin desktop Edit
0.0.0.0/0 X		

[Add security group rule](#)

- 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.

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

Actions ▾

- Connect
- Instance state ▾
- Actions ▾
- Connect
- Manage instance state
- Instance settings
- Networking
- Security**
- Get Windows password
- 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.

[▼](#) [C](#) [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.

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.

```
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'

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

```

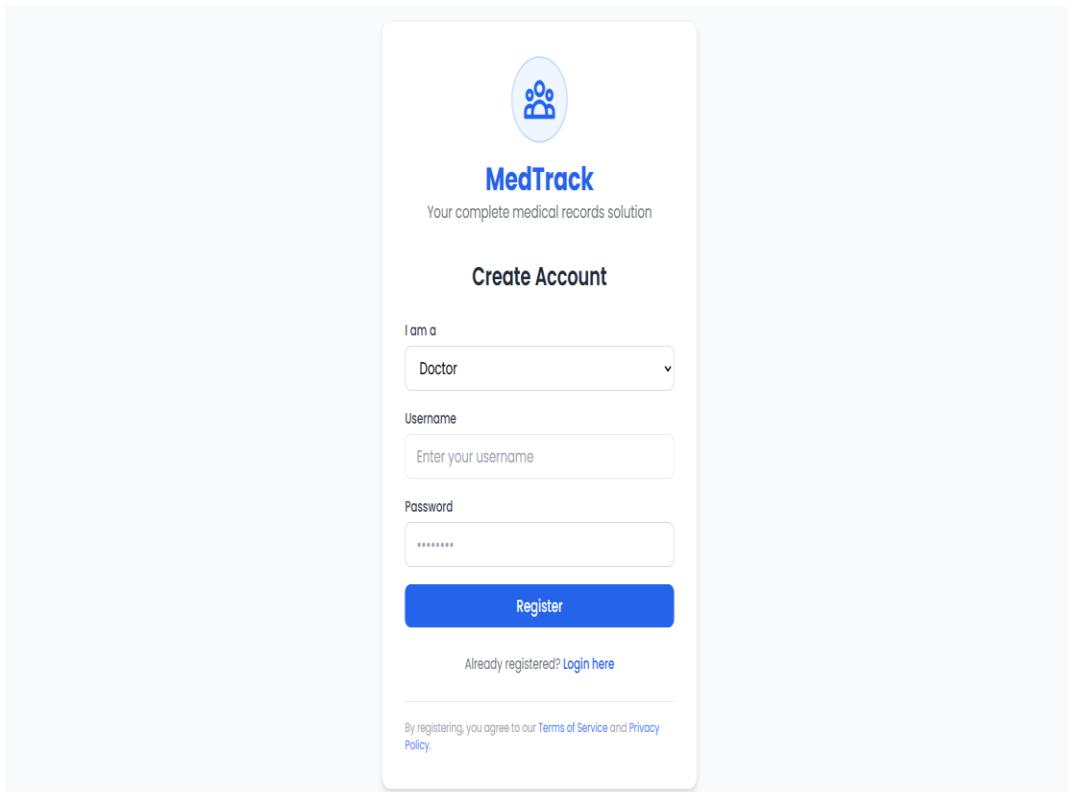
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running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

```

Milestone 8: Testing and Deployment

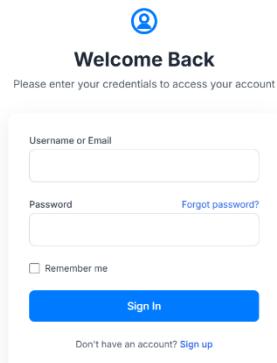
- **Activity 8.1: Conduct functional testing to verify user registration, login, book requests, and notifications.**

Register Page:



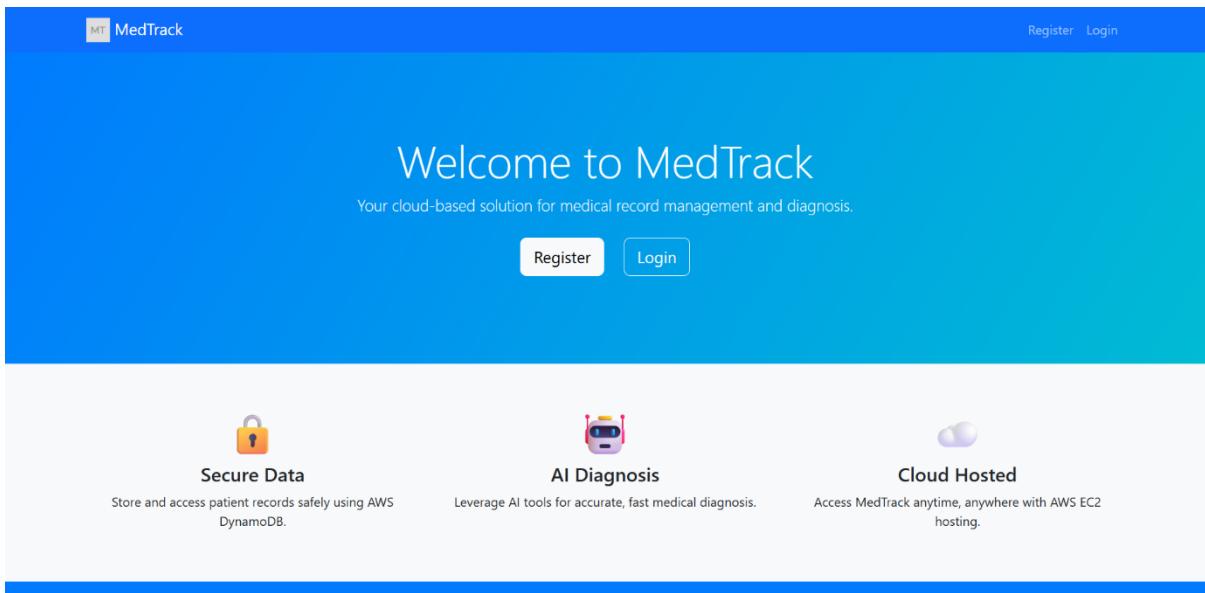
The image shows the 'Create Account' registration page for MedTrack. At the top, there is a logo consisting of three stylized human figures inside a blue circle. Below the logo, the word 'MedTrack' is written in a bold, blue, sans-serif font, followed by the tagline 'Your complete medical records solution' in a smaller, gray font. The main title 'Create Account' is centered above the form fields. The first field is a dropdown menu labeled 'I am a' with the option 'Doctor' selected. The next two fields are text input boxes: 'Username' containing the placeholder 'Enter your username' and 'Password' containing a series of five asterisks. A large blue 'Register' button is positioned below these fields. At the bottom of the form, there is a link 'Already registered? Login here' and a small note stating 'By registering, you agree to our [Terms of Service](#) and [Privacy Policy](#)'.

Login Page:



The image shows the MedTrack login page. At the top center is a blue circular logo with a white 'MT' monogram. Below it, the text "Welcome Back" is displayed in bold black font. Underneath that, a smaller line of text reads "Please enter your credentials to access your account". The main form area contains two input fields: "Username or Email" and "Password". To the right of the password field is a link "Forgot password?". Below the password field is a checkbox labeled "Remember me". A large blue "Sign In" button is centered at the bottom of the form. At the very bottom, there is a small link "Don't have an account? Sign up".

Home page:



The image shows the MedTrack home page. At the top, there is a blue header bar with the "MedTrack" logo on the left and "Register" and "Login" links on the right. The main content area has a teal background. Centered on the teal background is the text "Welcome to MedTrack" in large white font, followed by a smaller line of text "Your cloud-based solution for medical record management and diagnosis." Below this text are two buttons: "Register" and "Login", both in white text on a dark background. The bottom section of the page has a white background. It features three service icons: a lock icon for "Secure Data", a robot icon for "AI Diagnosis", and a cloud icon for "Cloud Hosted". Each service has a brief description below its icon.

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 Us page:

The screenshot shows the About Us page of the MedTrack website. At the top, there's a blue header bar. Below it, the main content area has a white background. It features three sections: 'Secure Data' with a lock icon, 'AI Diagnosis' with a robot icon, and 'Cloud Hosted' with a cloud icon. Each section includes a brief description. In the center, there's a large box with a rounded rectangle containing the heading 'About MedTrack' and a paragraph of text about the platform's purpose. At the bottom of the page, there's a blue footer bar with copyright information and links to social media.

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 screenshot shows the Doctor Dashboard. At the top, there's a green header bar with the 'MedTrack' logo on the left and a 'Logout' link on the right. Below the header, the main content area has a white background. It starts with a welcome message 'Welcome, Dr.' followed by a brief description of the dashboard's purpose. There are two main cards: 'Patient List' (with a 'View Patients' button) and 'Diagnosis Reviews' (with a 'Review Diagnoses' button). At the bottom of the page, there's a small copyright notice.

MedTrack

Logout

Welcome, Dr.

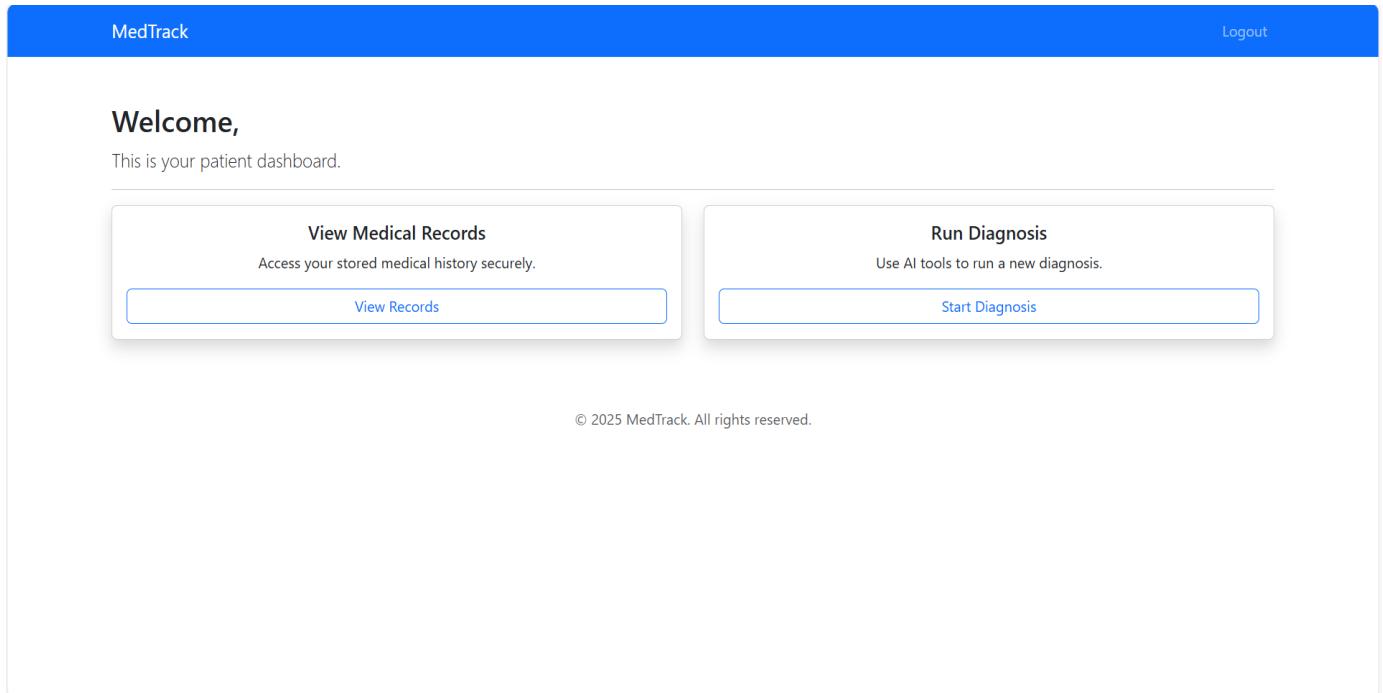
This is your doctor dashboard. You can access patient data and review diagnoses.

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:



A screenshot of the MedTrack patient dashboard. The top navigation bar is blue with the text "MedTrack" on the left and "Logout" on the right. Below the header, a welcome message reads "Welcome," followed by the text "This is your patient dashboard." Two main buttons are displayed: "View Medical Records" (with the sub-instruction "Access your stored medical history securely.") and "Run Diagnosis" (with the sub-instruction "Use AI tools to run a new diagnosis."). A copyright notice at the bottom center states "© 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.