

# **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.

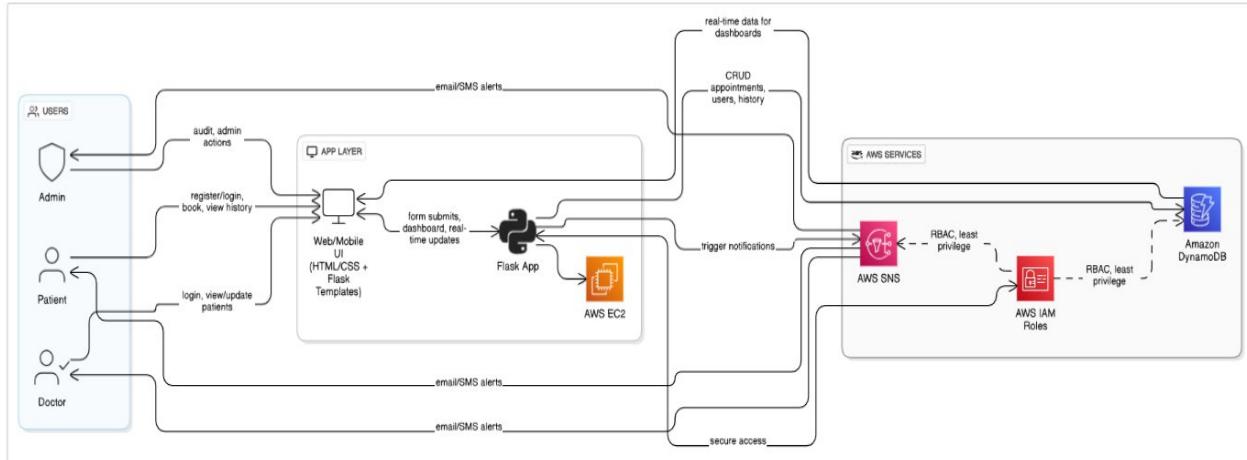
## **Scenario 2: Secure User Management with IAM**

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.

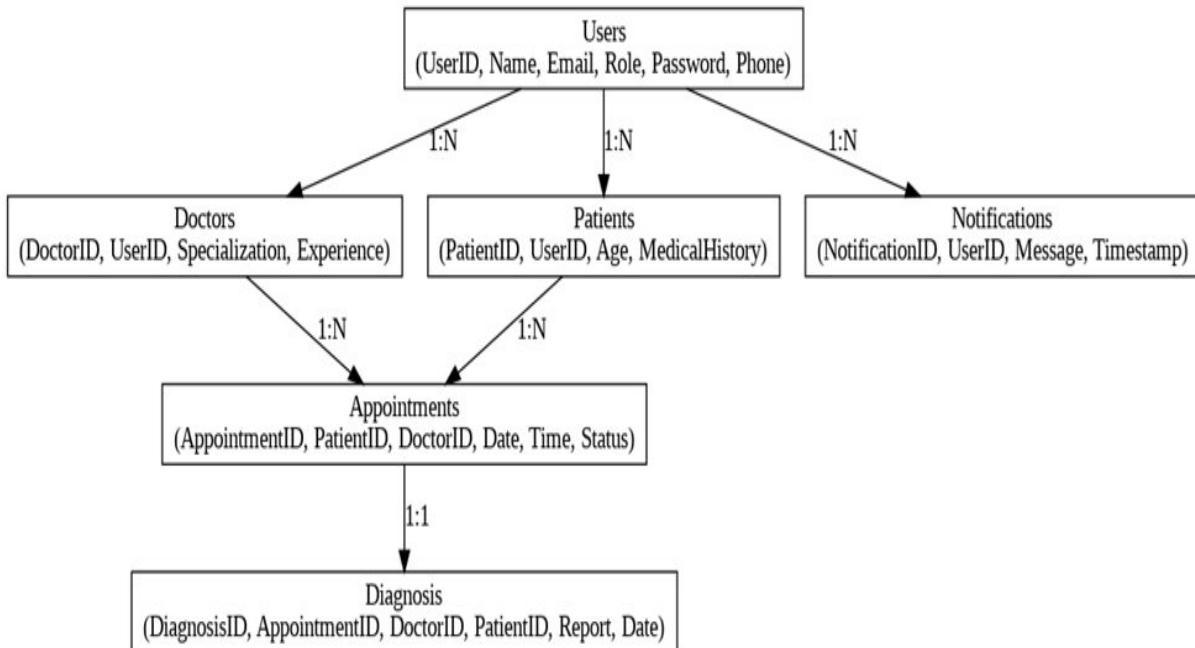
## **Scenario 3: Easy Access to Medical History and Resources**

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

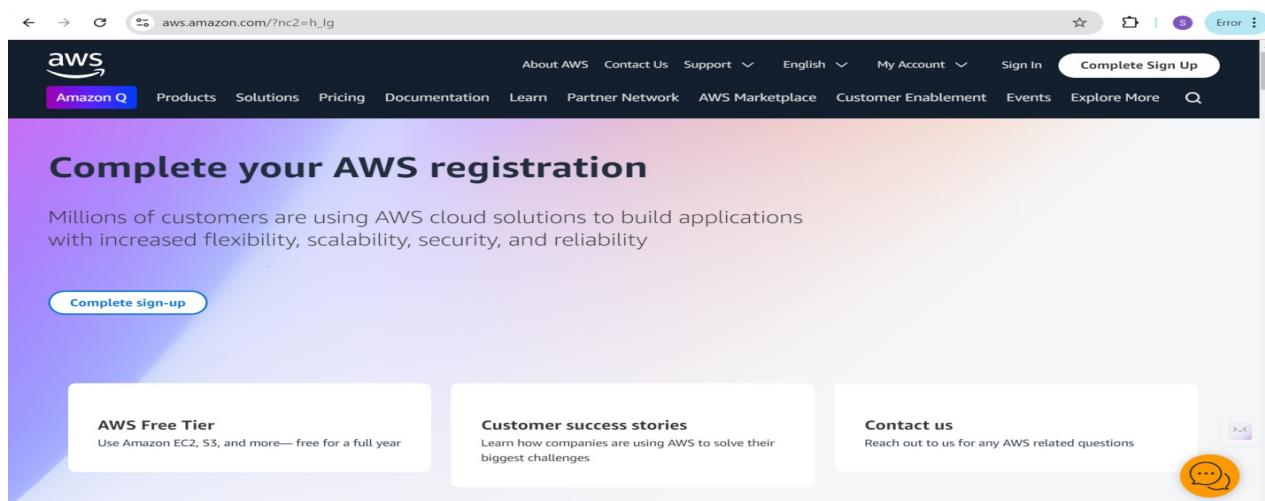
**Activity 7.2:** Run the Flask App

## 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 showing the AWS sign-in interface on the left and an AI Use Case Explorer sidebar on the right. The sign-in page features a "Sign in" header, two radio button options ("Root user" and "IAM user"), a "Root user email address" input field containing "username@example.com", and a "Next" button. A small note at the bottom states: "By continuing, you agree to the AWS Customer Agreement or other agreement for AWS services, and the Privacy Notice. This site uses essential cookies. See our Cookie Notice for more information." To the right is a dark purple sidebar with the title "AI Use Case Explorer" and the text "Discover AI use cases, customer success stories, and expert-curated implementation plans". At the bottom of the sidebar is a "Explore now &gt;" 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 search interface. The search bar at the top contains the text 'dynamoDB'. Below the search bar, the 'Services' section is expanded, showing a list of related services: Services, Features, Resources (New), Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The 'DynamoDB' service is highlighted with a blue border. To the right of the search bar, there is a search results summary: 'Search results for 'dyn'' followed by a list of services: **DynamoDB** (Managed NoSQL Database), **Amazon DocumentDB** (Fully-managed MongoDB-compatible database service), **CloudFront** (Global Content Delivery Network), and **Athena** (Serverless interactive analytics service). Each service entry includes a star icon indicating popularity.

The screenshot shows the DynamoDB Dashboard. On the left, there is a navigation sidebar with links for Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations (New), Reserved capacity, and Settings. A 'DAX' section is also present with links for Clusters, Subnet groups, Parameter groups, and Events. The main dashboard area has two sections: 'Alarms (0) Info' and 'DAX clusters (0) Info'. Both sections show a table with columns for Cluster name and Status. The 'Create resources' section on the right features a large orange 'Create table' button. Below it, there is information about Amazon DynamoDB Accelerator (DAX). The 'What's new' section at the bottom right indicates that AWS Cost Management now provides purchase recommendations for Amazon DynamoDB, dated SEP 1Q.

- **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.

**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. String ▾

**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. String ▾

|                            |                          |     |
|----------------------------|--------------------------|-----|
| 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** Create table

Tables (1) Actions ▾ Delete

| 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.

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 is the easiest way to create your table. You can modify it later if needed.

#### 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** X

**Tables**

- Dashboard
- Explore items
- PartiQL editor
- Backups
- Exports to S3
- Imports from S3
- Integrations **New**
- Reserved capacity
- Settings

**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.

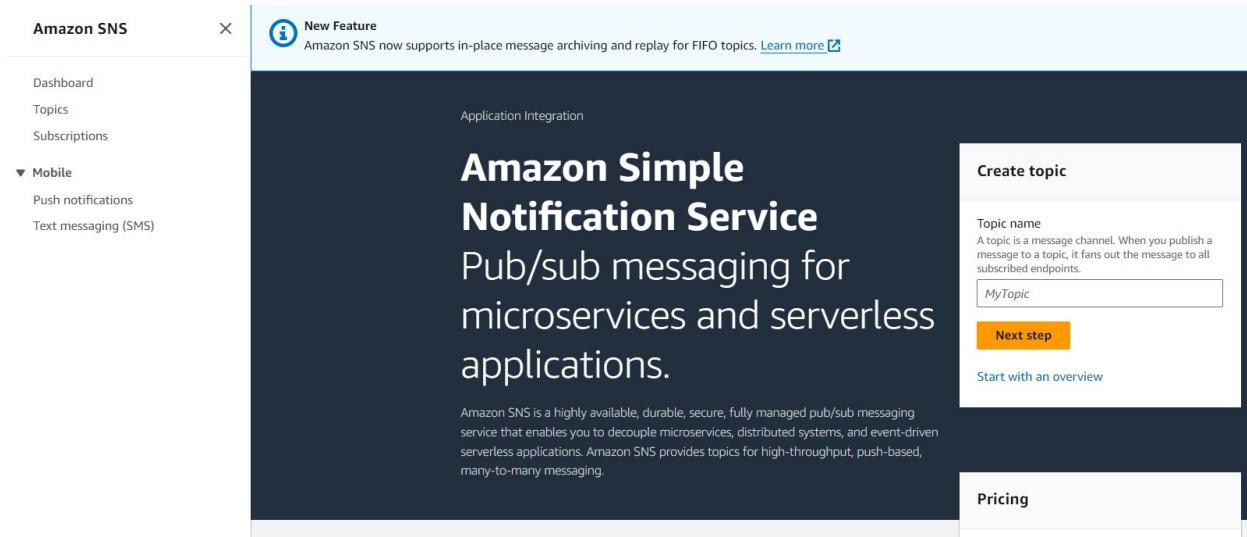
The screenshot shows the AWS search interface with the query 'sns' entered in the search bar. The results are categorized under 'Services' and 'Features'.

**Services**

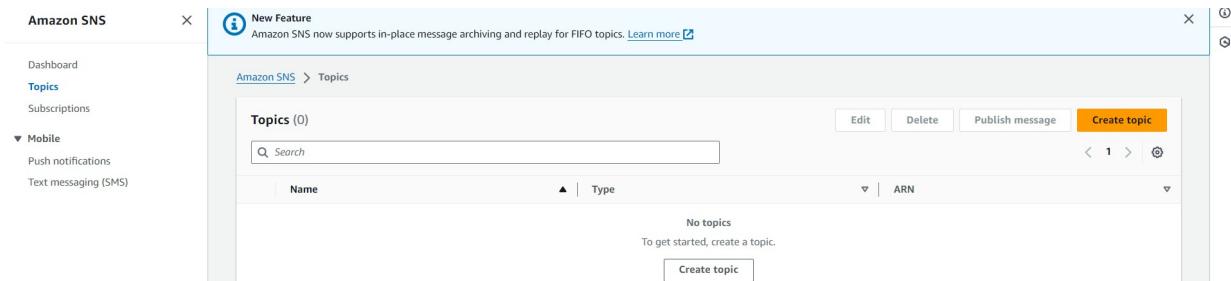
- Simple Notification Service** (star icon)  
SNS managed message topics for Pub/Sub
- Route 53 Resolver**  
Resolve DNS queries in your Amazon VPC and on-premises network.
- Route 53** (star icon)  
Scalable DNS and Domain Name Registration
- AWS End User Messaging** (star icon)  
Engage your customers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

**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

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**.

**Amazon SNS** X

**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-1:1557690616836: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<br>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 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". There are "Edit" and "Delete" buttons at the top right. Below the details, there are sections for "Subscription filter policy" (with a note about a dead-letter queue) and "Subscription filter policy" (info). A note states: "This policy filters the messages that a subscriber receives." It also mentions "No filter policy configured for this subscription. To apply a filter policy, edit this subscription." An "Edit" button is present here as well.

- After subscription request for the mail confirmation

The screenshot shows the AWS SNS console. A green banner at the top indicates a successful confirmation: "Confirmation request was sent successfully." The ARN of the subscription is listed as arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:7ae5d16-12ad-4731-9f7d-c342c2213ad5. Below this, the "BookRequestNotifications" topic configuration page is shown. The "Details" section lists the topic name (BookRequestNotifications), ARN (arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications), and type (Standard). The "Display name" field is empty. The "Topic owner" is listed as 557690616836. The "Subscriptions" tab is selected, showing a table of subscriptions. The table has columns for ID, Endpoint, Status, and Protocol. One row is visible: "Pending confirmation" (instantlibrary2@gmail.com), "Pending confirmation", EMAIL. At the bottom of the table are buttons for "Edit", "Delete", "Request confirmation" (highlighted in orange), "Confirm subscription", and "Create subscription". There are also navigation arrows and a search bar above the table.

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

## AWS Notification - Subscription Confirmation

Inbox ×

**AWS Notifications** <no-reply@sns.amazonaws.com>  
to me ▾

9

You have chosen to subscribe to the topic:  
**arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications**

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):  
[Confirm subscription](#)

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to [sns-opt-out](#)

---

**AWS Notifications** <no-reply@sns.amazonaws.com>  
to me ▾

You have chosen to subscribe to the topic:  
**arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications**

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):  
[Confirm subscription](#)

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to [sns-opt-out](#)



Simple Notification Service

### Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:

**arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4**

If it was not your intention to subscribe, [click here to unsubscribe](#).

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

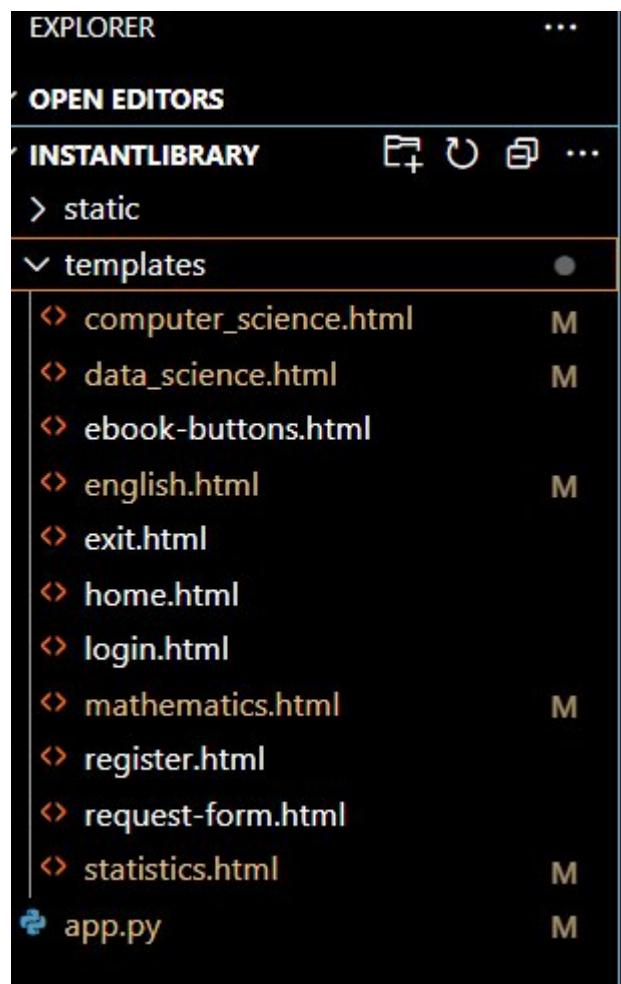
The screenshot shows the AWS SNS console with the following details:

- Topic:** BookRequestNotifications
- Name:** BookRequestNotifications
- ARN:** arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications
- Type:** Standard
- Subscriptions:** 2 (instantlibrary2@gmail.com - confirmed via EMAIL)
- Access policy:** Not explicitly shown but likely defined in the policy tab.
- Data protection policy:** Not explicitly shown but likely defined in the policy tab.
- Delivery policy (HTTP/S):** Not explicitly shown but likely defined in the policy tab.
- Delivery status logging:** Not explicitly shown but likely defined in the policy tab.
- Encryption:** Not explicitly shown but likely defined in the policy tab.
- Tags:** None
- Integrations:** None

## 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:

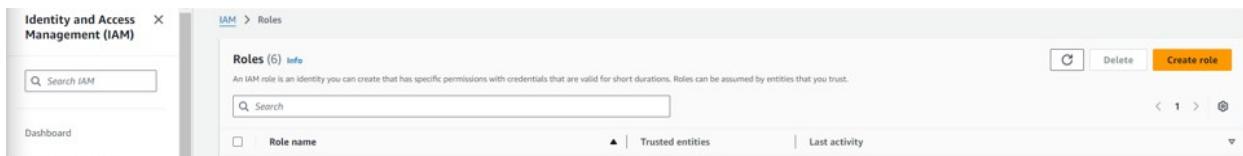
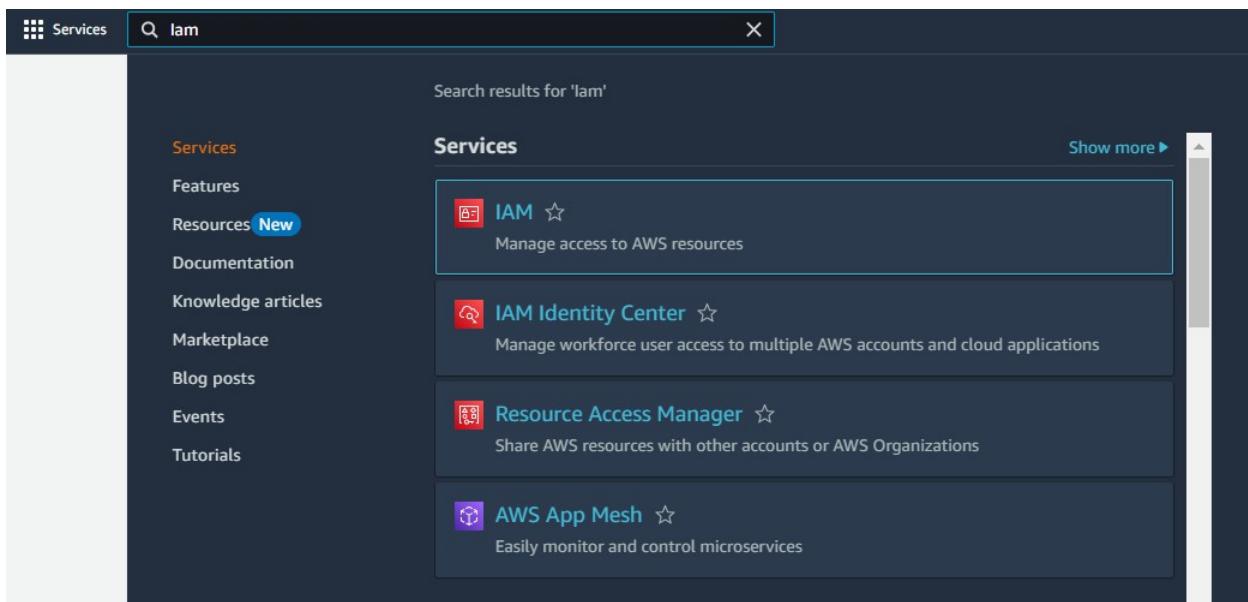
```
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=True)
```

**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.



**Select trusted entity**

**Trusted entity type**

- AWS service Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account Allow actions in other AWS accounts belonging to you or a 3rd party to perform actions in the account.
- Web identity Allow users federated by the specified external web identity provider to assume this role to perform actions in this account.

**Use case**  
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case  
EC2

Choose a use case for the specified service.

Use case  
 EC2      Allow EC2 instances to call AWS services on your behalf.  
 EC2 Role for AWS Systems Manager      Allow EC2 Instances to call AWS services like CloudWatch and Systems Manager on your behalf.  
 EC2 Spot Fleet      Allow EC2 Spot Fleet to request and terminates Spot Instances on your behalf.  
 EC2 - Spot Fleet Auto Scaling      Allow EC2 Spot Fleet Auto Scaling to update EC2 spot rates on your behalf.  
 EC2 - Spot Fleet Tagging      Allow EC2 to launch spot instances and attach tags to the launched instances on your behalf.  
 EC2 - Scheduled Instances      Allow EC2 Scheduled Instances to launch and manage spot instances on your behalf.  
 EC2 - Spot Fleet      Allow EC2 Spot Instances to launch and manage spot fleet instances on your behalf.  
 EC2 - Scheduled Instances      Allow EC2 Scheduled Instances to manage instances on your behalf.

**Add permissions**

**Permissions policies (1/955)**

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

Filter by Type  
All types

Policy name  
 AmazonDynamoDBFullAccess      AWS managed  
 AmazonDynamoDBReadOnlyAccess      AWS managed

Set permissions boundary - optional

**Next Step**

## ● 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  5 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"/>  AWSIoTDeviceDefenderPublishFindingsToSNSMitigationAction | AWS managed |

▶ Set permissions boundary - optional

[Name](#) [Select trusted entities](#)

[Name](#) [Add permissions](#)

[Step 1](#) [Name, review, and create](#)

### Name, review, and create

[Role details](#)

**Role name**  
Enter a meaningful name to identify this role.  
**aws\_dynamodb\_rrole**  
Maximum 128 characters. Use alphanumeric and '-' characters.

**Description**  
Add a short description for this role.  
**Allow EC2 instances to call AWS services on your behalf!**  
Maximum 1000 characters. Use spaces on Z and a-z, numbers 0-9, tabs, new lines, or any of the following characters: ~!@#\$%^&\*\_=+`{|},.

[Step 1: Select trusted entities](#) [Edit](#)

**Trust policy**

```
1 1 "Version": "2012-10-17",
2 2 "Statement": [
3 3 "Effect": "Allow",
4 4 "Action": "sts:AssumeRole",
5 5 "Principal": [
6 6 "service-role",
7 7 "arn:aws:lambda:us-east-1:123456789012:lambda-role"
8 8 ]
9 9 ]
10 10 }
11 11 }
12 12 }
13 13 }
14 14 }
15 15 }
16 16 }
17 17 }
18 18 }
19 19 }
20 20 }
21 21 }
```

[Step 2: Add permissions](#) [Edit](#)

**Permissions policy summary**

| Policy name                                  | Type        | Attached as        |
|--|-------------|--------------------|
| <a href="#">AmazonDynamoDBFullAccess</a>     | AWS-managed | Permissions policy |
| <a href="#">AmazonDynamoDBReadOnlyAccess</a> | AWS-managed | Permissions policy |

[Step 3: Add tags](#)

**Add tags - optional** [Edit](#)

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

No tags associated with the resource.

[Add new tag](#)  
You can add up to 50 tags per resource.

[Cancel](#) [Preview](#) [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::557690616856: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                 |

**Actions** C Simulate Remove Add permissions

## Milestone 6: EC2 Instance Setup

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

---

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

---

Local Codespaces

Clone ?

HTTPS SSH GitHub CLI

<https://github.com/AlekhyaPenukula/InstantLit>

Clone using the web URL.

---

Open with GitHub Desktop

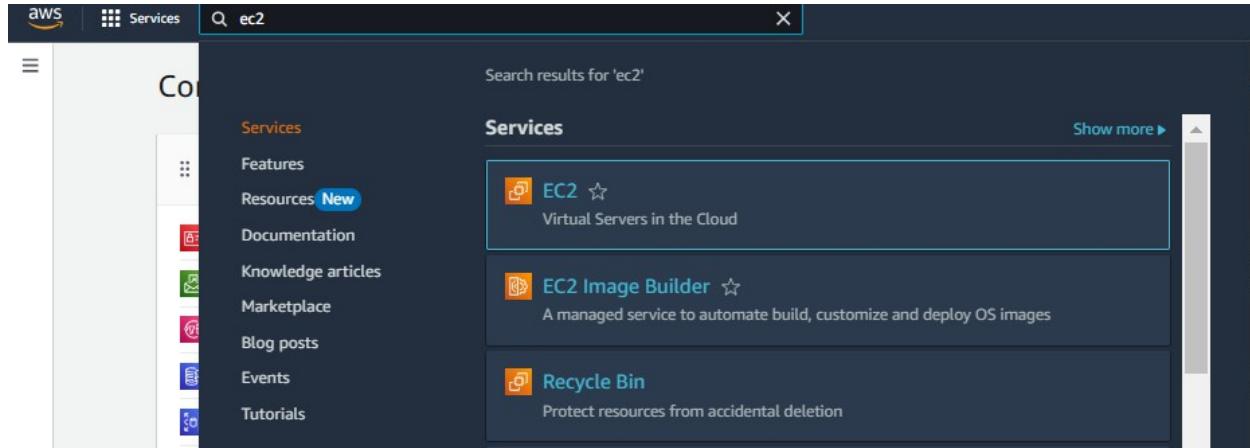
---

Download ZIP

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

- **Launch EC2 Instance**

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



- Click on Launch instance to launch EC2 instance

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with options like EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, and Savings Plans. The main area has a header with 'Instances Info' and filters for 'Find instance by attribute or tag (case-sensitive)', 'Last updated' (less than a minute ago), 'Connect', 'Instance state' (All states), 'Actions', and 'Launch instances'. Below this is a search bar and a table with columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. A message says 'No instances' and 'You do not have any instances in this region'. At the bottom right of the table is a large 'Launch instances' button.

The screenshot shows the 'Launch an instance' wizard. The current step is 'Name and tags'. It has a 'Name and tags' section with a 'Name' input field containing 'InstantLibraryApp' and a 'Add additional tags' link. To the right, under 'Summary', it shows 'Number of instances: 1'. Below that are sections for 'Software Image (AMI)', 'Virtual server type (instance type)', and 'Firewall (security group)'. A 'Next Step' button is at the bottom right.

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

The screenshot shows the 'Amazon Machine Image (AMI)' selection screen. On the left, there are cards for Amazon Linux, macOS, Ubuntu, Windows, and Red Hat. A 'More' button is next to the Red Hat card. On the right, there's a search icon and a link to 'Browse more AMIs'. Below this, it says 'Including AMIs from AWS, Marketplace and the Community'. Under 'Amazon Machine Image (AMI)', the 'Amazon Linux 2023 AMI' is selected, showing its details: 'Free tier eligible', 'ami-02b49a24cfb95941c (64-bit (x86), uefi-preferred) / ami-04ad8c7fcc828fad4 (64-bit (Arm), uefi)', 'Virtualization: hvm', 'ENA enabled: true', and 'Root device type: ebs'. Below this, there's a 'Description' section for Amazon Linux 2023, which is described as a modern, general purpose Linux-based OS with 5 years of long term support. It also lists 'Architecture: 64-bit (x86)', 'Boot mode: uefi-preferred', 'AMI ID: ami-02b49a24cfb95941c', and a '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                          |

**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.

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 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 (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
- Key pair (login):** InstantLibrary
- Summary:** Number of instances: 1
- Software Image (AMI):** Amazon Linux 2023 AMI 2023.5.2...read more
- 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:** Can cancel, Preview code, Launch instance (highlighted in orange).

- Activity 6.2: Configure security groups for HTTP, and SSH access.

**▼ Network settings [Info](#)**

VPC - required [Info](#)  
 (default) [▼](#) [C](#)

Subnet [Info](#)  
 [▼](#) [C](#) Create new subnet [D](#)

Auto-assign public IP [Info](#)  
 [▼](#)

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 [C](#)  Select existing security group [S](#)

Security group name - required

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](#)

**Inbound Security Group Rules**

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

|   |   |  |
|---|---|--|
| Type <a href="#">Info</a><br><input type="text" value="ssh"/> <a href="#">▼</a>             | Protocol <a href="#">Info</a><br><input type="text" value="TCP"/> <a href="#">▼</a>                             | Port range <a href="#">Info</a><br><input type="text" value="22"/> <a href="#">▼</a>                                     |
| Source type <a href="#">Info</a><br><input type="text" value="Anywhere"/> <a href="#">▼</a> | Source <a href="#">Info</a><br><input type="text" value="Add CIDR, prefix list or security"/> <a href="#">▼</a> | Description - optional <a href="#">Info</a><br><input type="text" value="e.g. SSH for admin desktop"/> <a href="#">▼</a> |
| <input type="text" value="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><br><input type="text" value="HTTP"/> <a href="#">▼</a>          | Protocol <a href="#">Info</a><br><input type="text" value="TCP"/> <a href="#">▼</a>                             | Port range <a href="#">Info</a><br><input type="text" value="80"/> <a href="#">▼</a>                                     |
| Source type <a href="#">Info</a><br><input type="text" value="Custom"/> <a href="#">▼</a> | Source <a href="#">Info</a><br><input type="text" value="Add CIDR, prefix list or security"/> <a href="#">▼</a> | Description - optional <a href="#">Info</a><br><input type="text" value="e.g. SSH for admin desktop"/> <a href="#">▼</a> |
| <input type="text" value="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><br><input type="text" value="Custom TCP"/> <a href="#">▼</a>    | Protocol <a href="#">Info</a><br><input type="text" value="TCP"/> <a href="#">▼</a>                             | Port range <a href="#">Info</a><br><input type="text" value="5000"/> <a href="#">▼</a>                                   |
| Source type <a href="#">Info</a><br><input type="text" value="Custom"/> <a href="#">▼</a> | Source <a href="#">Info</a><br><input type="text" value="Add CIDR, prefix list or security"/> <a href="#">▼</a> | Description - optional <a href="#">Info</a><br><input type="text" value="e.g. SSH for admin desktop"/> <a href="#">▼</a> |
| <input type="text" value="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, it says 'Successfully initiated launch of instance i-001861022fbcac290'. Below that is a 'Launch log' button. The main area is titled 'Next Steps' with a search bar. It lists several actions in a grid:

- 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. Buttons: 'Create billing alerts'.
- Connect to your instance**: Once your instance is running, log into it from your local computer. Buttons: 'Connect to instance', 'Learn more'.
- Connect an RDS database**: Configure the connection between an EC2 instance and a database to allow traffic flow between them. Buttons: 'Connect an RDS database', 'Create a new RDS database', 'Learn more'.
- Create EBS snapshot policy**: Create a policy that automates the creation, retention, and deletion of EBS snapshots. Buttons: 'Create EBS snapshot policy'.
- 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. Buttons: 'Manage detailed monitoring'.
- Create Load Balancer**: Create a application, network gateway or classic Elastic Load Balancer. Button: 'Create Load Balancer'.
- Create AWS budget**: AWS Budgets allows you to create budgets, forecast spend, and take action on your costs and usage from a single location. Button: 'Create AWS budget'.
- Manage CloudWatch alarms**: Create or update Amazon CloudWatch alarms for this instance. Button: 'Manage CloudWatch alarms'.
- 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 (EDR). Button: 'Disaster recovery for your instances'.
- 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. Button: 'Monitor for suspicious runtime activities'.
- Get instance screenshot**: Capture a screenshot from the instance and view it as an image. This is useful for troubleshooting an unreachable instance. Button: 'Get instance screenshot'.
- Get system log**: View the instance's system log to troubleshoot issues. Button: 'Get system log'.

At the bottom right is 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)' page. It lists one instance:

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

Below the table is the 'Instance summary for i-001861022fbcac290 (InstantLibraryApp)' page. It shows the following details:

- Instance ID:** i-001861022fbcac290
- IPV6 address:** -
- Hostname type:** IP name: ip-172-31-3-5.ap-south-1.compute.internal
- Answer private resource DNS name:** IPv4 (A)
- Auto-assigned IP address:** -
- IAM Role:** arn:aws:iam::aws:lambda:role/DynamoDBRole
- IMDSv2 Required:** -
- Public IPv4 address:** -
- Private IP DNS name (IPv4 only):** ip-172-31-3-5.ap-south-1.compute.internal
- Instance state:** Stopped
- Instance type:** t2.micro
- VPC ID:** vpc-03cdc7b6f19dd7211
- Subnet ID:** subnet-0d9fa3144480cc9a9
- Instance ARN:** arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290
- Private IPv4 addresses:** 172.31.3.5
- Public IPv4 DNS:** -
- Elastic IP addresses:** -
- AWS Compute Optimizer finding:** Opt-in to AWS Compute Optimizer for recommendations. | Learn more
- Auto Scaling Group name:** -

At the bottom 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<br><a href="#">i-001861022fbcac290</a>                  | Public IPv4 address<br>-   | Private IPv4 addresses<br><a href="#">172.31.3.5</a>   |
| IPv6 address<br>-   | Instance state<br><a href="#">Stopped</a>  | Public IPv4 DNS<br>-   |
| Hostname type<br>IP name: ip-172-31-3-5.ap-south-1.compute.internal | Private IP DNS name (IPv4 only)<br><a href="#">ip-172-31-3-5.ap-south-1.compute.internal</a>     | Change security groups   |
| Answer private resource DNS name<br>IPv4 (A)                        | Instance type<br>t2.micro  | Get Windows password   |
| Auto-assigned IP address<br>-                                       | VPC ID<br><a href="#">vpc-03cdc7b6f19dd7211</a>  | Modify IAM role  |
| IAM Role<br><a href="#">sns_Dynamodb_role</a>                       | Subnet ID<br><a href="#">subnet-0d9fa3144480cc9a9</a>  | Elastic IP addresses<br>-  |
| IMDSv2<br>Required  | Instance ARN<br><a href="#">arn:aws:ec2:ap-south-1:557690616856:instance/i-001861022fbcac290</a> | AWS Compute Optimizer finding<br><a href="#">Opt-in to AWS Compute Optimizer for recommendations.   Learn more</a> |

Connect Instance state Actions ▾

Connect  
Manage instance state  
Instance settings  
Networking  
**Security**  
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.

X

**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
   _#
  /~\_\ #####
 /~~ \_\####\_
~~  \_\#\#\|_
~~  \_\#/ \
~~  V~'__-> https://aws.amazon.com/linux/amazon-linux-2023
~~~ .-. /
~~~ /`_/
~~~ /m, -/
Last login: Tue Oct 15 04:17:59 2024 from 13.233.177.3
[ec2-user@ip-172-31-3-5 ~]$ █
```

i-001861022fbcac290 (InstantLibraryApp)  
PublicIPs: 13.201.74.42 PrivateIPs: 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/gorantlahemanth/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
      \###\
      \#/
      V--> https://aws.amazon.com/linux/amazon-linux-2023
      /
      /m/
Last login: Tue Oct 15 04:17:59 2024 from 13.233.177.3
[ec2-user@ip-172-31-3-5 ~]$ git clone https://github.com/AlekhyaPenubakula/InstantLibrary.git
fatal: destination path 'InstantLibrary' already exists and is not an empty directory.
[ec2-user@ip-172-31-3-5 ~]$ cd InstantLibrary
[ec2-user@ip-172-31-3-5 InstantLibrary]$ cd InstantLibrary
[ec2-user@ip-172-31-3-5 InstantLibrary]$ flask run --host=0.0.0.0 --port=80
 * Debug mode: off
Permission denied
[ec2-user@ip-172-31-3-5 InstantLibrary]$ ^C
[ec2-user@ip-172-31-3-5 InstantLibrary]$ ^C
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://172.31.3.5:80
Press CTRL+C to quit
^C[ec2-user@ip-172-31-3-5 InstantLibrary]$ 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 -
```

#### 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

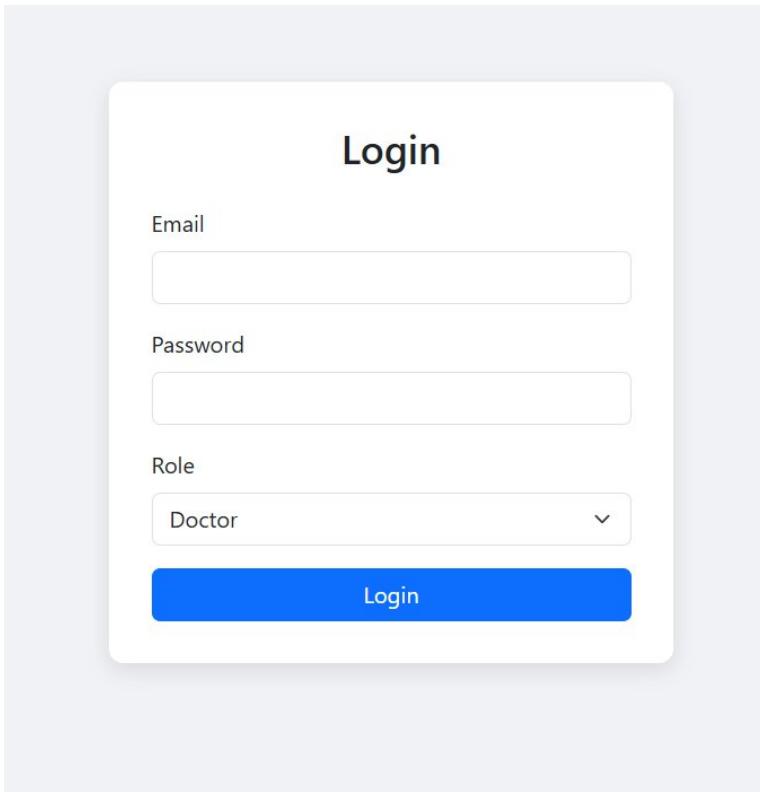
- **Activity 8.1: Conduct functional testing to verify user registration, login, book requests, and notifications.**

**Register Page:**

The screenshot shows a 'Register' form with the following fields:

- Name: An input field for entering a name.
- Email: An input field for entering an email address.
- Password: An input field for entering a password.
- Role: A dropdown menu set to 'Patient'.
- Age: An input field for entering age.
- Gender: A dropdown menu set to 'Male'.
- Register: A blue button labeled 'Register' at the bottom.

**Login Page:**



### Home page:

The home page for MedTrack. It features a blue header with the "MedTrack" logo. The main content area has a teal background with the text "Welcome to MedTrack" and "Your cloud-based solution for medical record management and diagnosis." Below this are three buttons: "Register" (white), "Login" (light gray), and "About Us" (dark gray).

Welcome to MedTrack

Your cloud-based solution for medical record management and diagnosis.

Register   Login   About Us

#### Secure Data

Store and access patient records safely using AWS DynamoDB.

#### AI Diagnosis

Leverage AI tools for accurate, fast medical diagnosis.

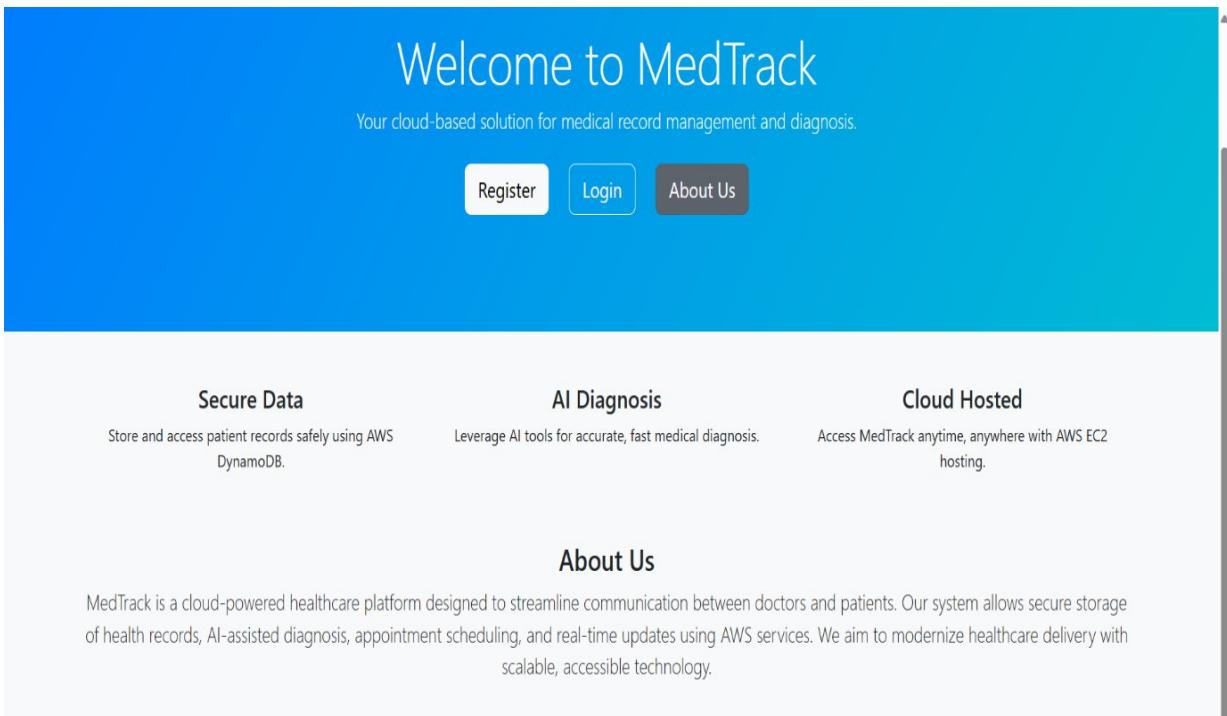
#### Cloud Hosted

Access MedTrack anytime, anywhere with AWS EC2 hosting.

#### About Us

MedTrack is a cloud-powered healthcare platform designed to streamline communication between doctors and patients. Our system allows secure storage of health records, AI-assisted diagnosis, appointment scheduling, and real-time updates using AWS services. We aim to modernize healthcare delivery with

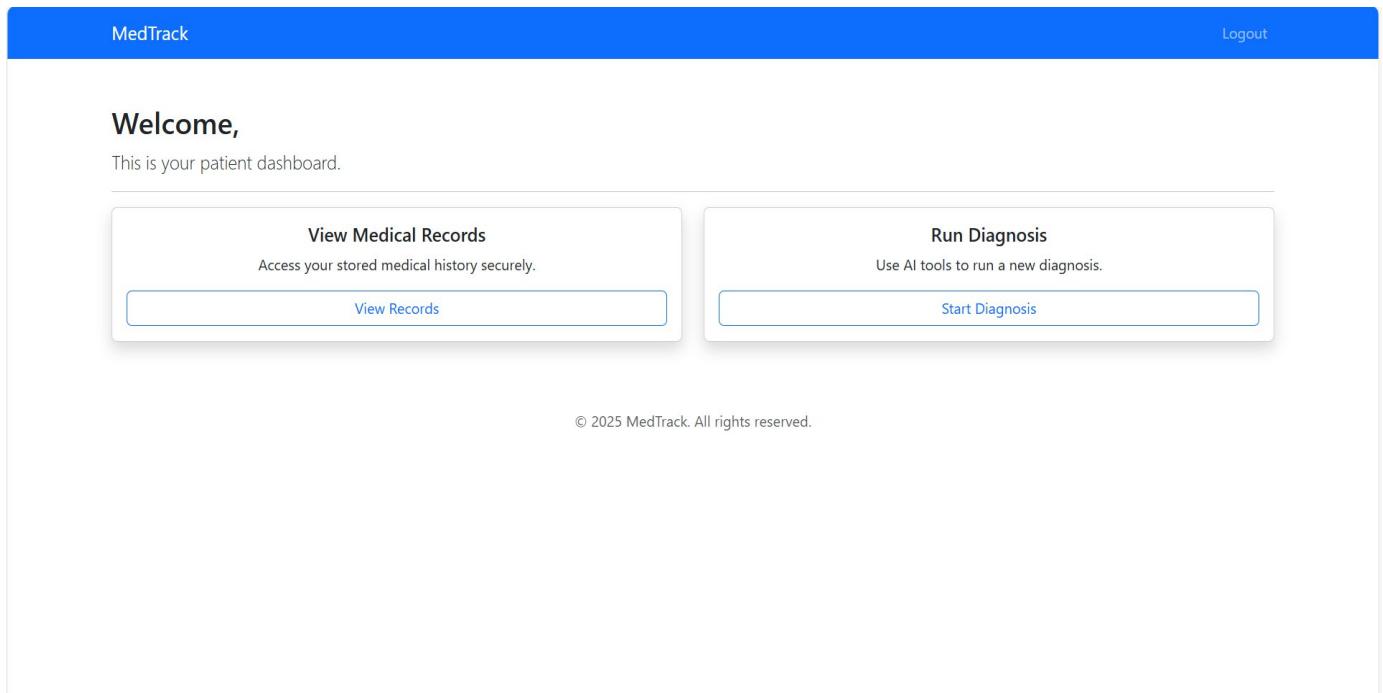
### About Us page:



## Doctor DashBoard:

The image shows the Doctor Dashboard interface. It has a green header bar with the "MedTrack" logo on the left and a "Logout" link on the right. The main content area is white and starts with a "Welcome, Dr." message. Below it is a "Patient List" section with a "View Patients" button. To the right is a "Diagnosis Reviews" section with a "Review Diagnoses" button. At the bottom center is a copyright notice: "© 2025 MedTrack. All rights reserved.".

## Patient DashBoard:



**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.