

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.

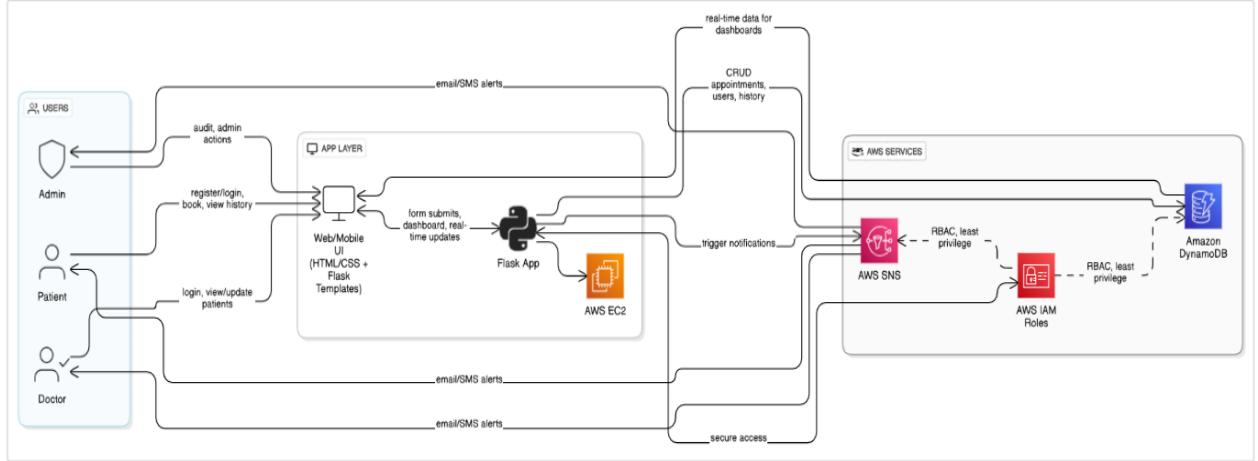
Scenario 2: Secure User Management with IAM

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.

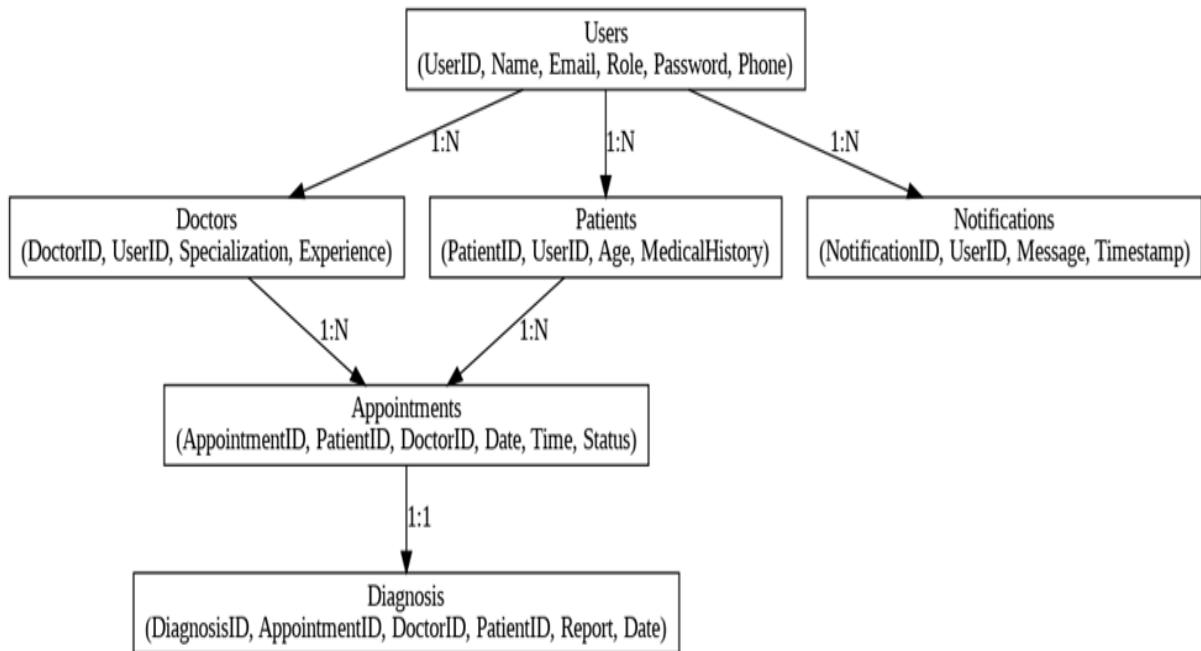
Scenario 3: Easy Access to Medical History and Resources

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:

- AWS Account Setup:
<https://docs.aws.amazon.com/accounts/latest/reference/getting-started.html>
- AWS IAM (Identity and Access Management):
<https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html>

- AWS EC2 (Elastic Compute Cloud):
<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>
- AWS DynamoDB:
<https://docs.aws.amazon.com/amazondynamodb/Introduction.html>
- Amazon SNS:
<https://docs.aws.amazon.com/sns/latest/dg/welcome.html>
- Git Documentation:
<https://git-scm.com/doc>
- VS Code Installation: (download the VS Code using the below link or you can get that in Microsoft store)
<https://code.visualstudio.com/download>

Project WorkFlow:

Milestone 1. Web Application Development and Setup

- Develop the Backend Using Flask.
- Integrate AWS Services Using boto3.

Milestone 2. AWS Account Setup and Login

- Set up an AWS account if not already done.
- Login to AWS Management Console.

Milestone 3. DynamoDB Database Creation and Setup

- Create a DynamoDB Table.
- Configure Attributes for User Data and Book Requests.

Milestone 4. SNS Notification Setup

- Create SNS topics for book request notifications.
- Subscribe users and library staff to SNS email notifications.

Milestone 5. IAM Role Setup

- Create IAM Role
- Attach Policies

Milestone 6. EC2 Instance Setup

- Launch an EC2 instance to host the Flask application.

- Configure security groups for HTTP, and SSH access.

Milestone 7. Deployment using EC2

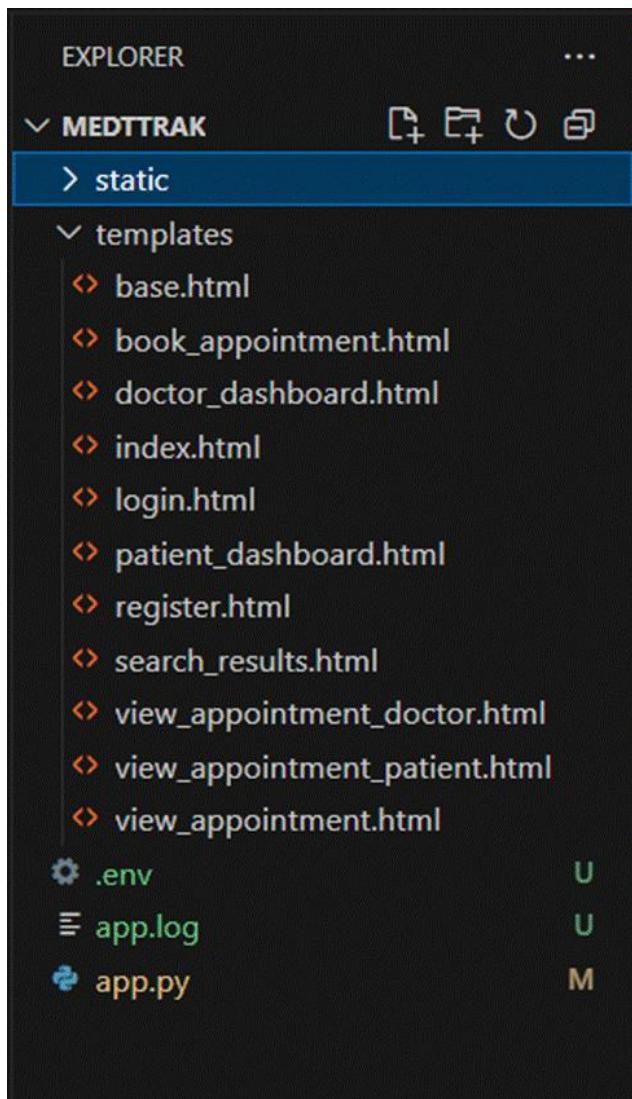
- Upload Flask Files
- Run the Flask App

Milestone 8. Testing and Deployment

- Conduct functional testing to verify user registration, login, book requests, and notifications.

1. Develop the backend using Flask

- File Explorer Structure



Description of the code :

1.Imports:

```
from flask import Flask, render_template, request, redirect, url_for, session, flash
from werkzeug.security import generate_password_hash, check_password_hash
from datetime import datetime
import json
```

2. Flask App Initialization:

```
app = Flask(__name__)
app.secret_key = 'your_secret_key_here'
```

- app = Flask(__name__): Starts the Flask application.
- app.secret_key: Required to safely use sessions and flashes.

3. Temporary In-Memory Storage:

```
# Temporary in-memory storage
users = []
bookings = []
booking_counter = 1
```

- users: List to store registered users.
- bookings: List to store all ticket bookings.
- booking_counter: A simple ID counter for bookings.

4. Authentication Routes:

4.1. Homepage /:

```
# Authentication Routes
@app.route('/')
def index():
    return render_template('index.html')
```

4.2. Login /login:

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

        user = next((u for u in users if u['email'] == email), None)

        if user and check_password_hash(user['password'], password):
            session['user'] = {
                'id': user['id'],
                'name': user['name'],
                'email': user['email']
            }
            print("👤 Logged In User Session:", session['user'])
            return redirect(url_for('home1'))
        else:
            flash('Invalid email or password', 'danger')
    return render_template('login.html')
```

- If GET: Show the login page.
- If POST: Validate user login.
- Check if the email exists and password is correct.
- If success, save user info in session.
- If wrong, flash an "Invalid" message.

4.3. Signup /signup:

```
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        password = generate_password_hash(request.form['password'])

        if any(u['email'] == email for u in users):
            flash('Email already registered!', 'danger')
            return redirect(url_for('signup'))

        new_user = {
            'id': len(users) + 1,
            'name': name,
            'email': email,
            'password': password
        }
        users.append(new_user)

        print("✓ Current Users List:", users)

        flash('Registration successful! Please login.', 'success')
        return redirect(url_for('login'))

    return render_template('signup.html')
```

- If GET: Show signup form.
- If POST:
 - Collect user details.
 - Check if email already exists.
 - Save the user with hashed password into users.
 - Redirect to login after successful registration.

4.4. Logout /logout:

```
@app.route('/logout')
def logout():
    session.pop('user', None)
    flash('You have been logged out!', 'info')
    return redirect(url_for('index'))
```

- Remove the user from session (log out).
- Flash a message.
- Redirect to homepage.

5. Main Application Pages:

5.1. Home After Login /home1, About Page /about, Contact Page /contact_us

```
# Application Routes
@app.route('/home1')
def home1():
    if 'user' not in session:
        return redirect(url_for('login'))

    print("👤 Current Session User Info:", session['user'])

    return render_template('home1.html')

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/contact_us')
def contact():
    return render_template('contact_us.html')
```

- Check if the user is logged in.
- If yes, show home1.html.
- If no, redirect to login page.
- Show About page.

- Show Contact Us page.

6.Running the Flask App:

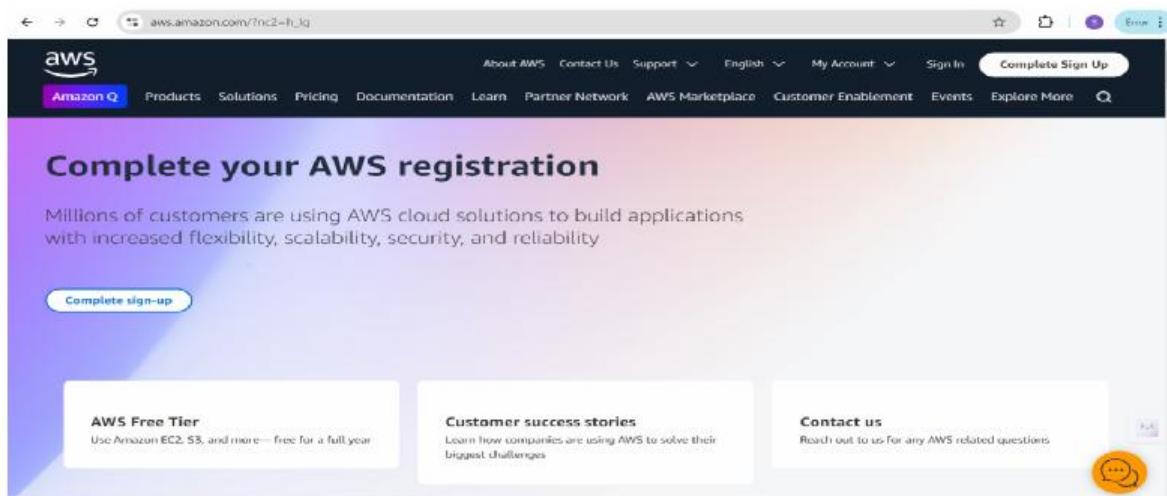
```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

- Starts the app on your machine at <http://localhost:5000>.
- debug=True means it auto-reloads when you change the code.

2. AWS Account Setup and Login

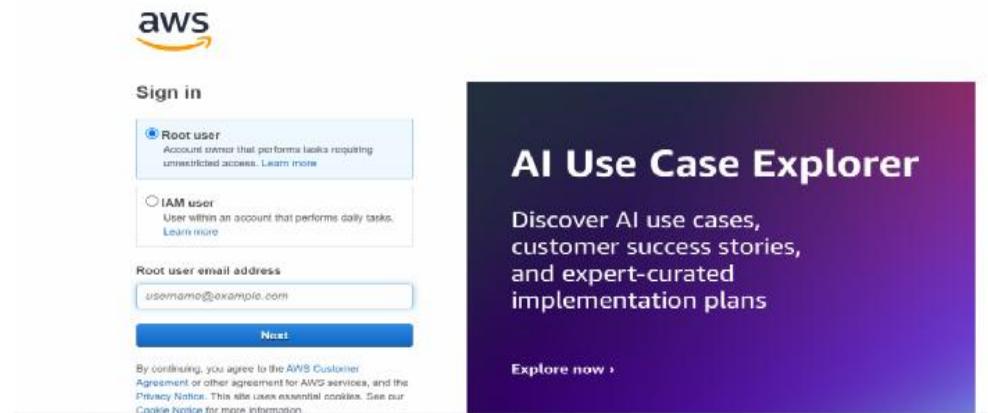
- **Activity 1.1: Set up an AWS account if not already done.**

- 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.



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 displays the AWS Services search results for 'dynamoDB'. The search bar at the top shows 'dynamoDB'. The left sidebar lists various navigation options like Services, Features, Resources (New), Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main content area shows a list of services under '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, under 'Features', there are sections for **Settings** (DynamoDB feature) and **Clusters** (DynamoDB feature).

The screenshot shows the AWS DynamoDB Dashboard. On the left, there's a sidebar with links like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations (New), Reserved capacity, and Settings. Under the 'DAX' section, there are links for Clusters, Subnet groups, Parameter groups, and Events.

The main area has two sections: 'Alarms (0) info' and 'DAX clusters (0) info'. Both sections have search bars, pagination, and status columns. The 'Create resources' sidebar on the right has a 'Create table' button and information about Amazon DynamoDB Accelerator (DAX). A 'What's new' section at the bottom right shows a recent update about AWS Cost Management.

This screenshot shows the 'Tables' section of the AWS DynamoDB Tables page. The sidebar on the left is identical to the one in the previous screenshot. The main area shows a table with columns: Name, Status, Partition key, Sort key, Indexes, Deletion protection, Read capacity mode, Write capacity mode, and Total size. A message says 'You have no tables in this account in this AWS Region.' There is a 'Create table' button at the bottom.

- **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.

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.

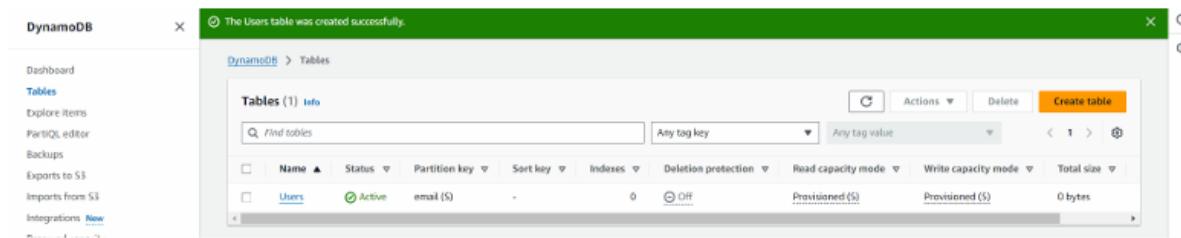
Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes
Resource-based 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.

Create table



- o 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

Customize settings

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
- Tables
- Explore items
- Partition editor
- Backups
- Exports to S3
- Imports from S3
- Integrations
- 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

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.

Searched for 'sns'

Services

- Simple Notification Service New
- Route 53 Resolver
- Route 53
- AWS End User Messaging

Features

- Events
- SMS
- Hosted zones

Resources New

Documentation

Knowledge articles

Marketplace

Blog posts

Events

Tutorials

Amazon SNS

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

Application Integration

Amazon Simple Notification Service

Pub/sub messaging for microservices and serverless applications.

Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and event-driven serverless applications. Amazon SNS provides topics for high-throughput, push-based, many-to-many messaging.

Create topic

Topic name
A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.

Next step

Pricing

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

The screenshot shows the Amazon SNS Topics page. On the left, there's a sidebar with links to Dashboard, Topics (which is selected), Subscriptions, Mobile, Push notifications, and Text messaging (SMS). The main area has a header with a 'New Feature' banner about FIFO message archiving. Below it is a search bar and a table with columns for Name, Type, and ARN. A button labeled 'Create topic' is at the bottom right of the table area.

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

The screenshot shows the 'Create topic' wizard. The top navigation bar includes 'Amazon SNS > Topics > Create topic'. The main section is titled 'Create topic' and has a 'Details' tab selected. Under 'Type', there are two options: 'FIFO (first-in, first-out)' (unchecked) and 'Standard' (checked). The 'Standard' option is highlighted with a blue border. Below the type selection is a 'Name' field containing 'BookRequestNotifications' with a note about character limits. There is also a 'Display name - optional' field containing 'My Topic' with a note about character limits.

The screenshot shows the configuration page for an AWS SNS topic named 'BookRequestNotifications'. The page is divided into several sections:

- Access policy - optional**: This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic.
- Data protection policy - optional**: This policy defines which sensitive data to monitor and to prevent from being exchanged via your topic.
- Delivery policy (HTTP/S) - optional**: The policy defines how Amazon SNS retries failed deliveries to HTTP/S endpoints. To modify the default settings, expand this section.
- Delivery status logging - optional**: These settings configure the logging of message delivery status to CloudWatch Logs.
- Tags - optional**: A tag is a metadata label that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track your costs. [Learn more](#)
- Active tracing - optional**: Use AWS X-Ray active tracing for this topic to view its traces and service map in Amazon CloudWatch. Additional costs apply.

At the bottom right of the configuration page are two buttons: 'Cancel' and 'Create topic'.

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

The screenshot shows the AWS SNS 'Topics' list. One topic, 'BookRequestNotifications', is selected. The topic details are displayed:

- Name:** BookRequestNotifications
- ARN:** arn:aws:sns:ap-southeast-1:517690416014:BookRequestNotifications
- Type:** Standard

The 'Subscriptions' tab shows the following table:

ID	Endpoint	Status	Protocol
No subscriptions found.			

At the top of the page, a green banner indicates: 'Topic BookRequestNotifications created successfully. You can create subscribers and send messages to them from this topic.'

- **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
arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications

Protocol
The type of endpoint to subscribe
Email

Endpoint
An email address that can receive notifications from Amazon SNS.
instantlibrary2@gmail.com

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

Subscription filter policy - optional [Info](#)
This policy filters the messages that a subscriber receives.

Redrive policy (dead-letter queue) - optional [Info](#)
Send undeliverable messages to a dead-letter queue.

[Cancel](#) [Create subscription](#)

Amazon SNS X

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

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

Amazon SNS > Topics > BookRequestNotifications > Subscription: d78e0371-9235-404d-952c-85c2743607c4 Edit Delete

Details	
ARN	arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4
Endpoint	instantlibrary2@gmail.com
Topic	BookRequestNotifications
Subscription Principal	arn:aws:iam:557690616836::root
Status	Pending confirmation
Protocol	Email

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

Subscription filter policy [Info](#)
This policy filters the messages that a subscriber receives.

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

- o After subscription request for the mail confirmation

The screenshot shows the AWS SNS console with the 'BookRequestNotifications' topic selected. The topic details are shown, including the ARN (arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications) and Type (Standard). A table lists a single subscription:

ID	Endpoint	Status	Protocol
Pending confirmation	instantlibrary@gmail.com	Pending confirmation	EMAIL

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

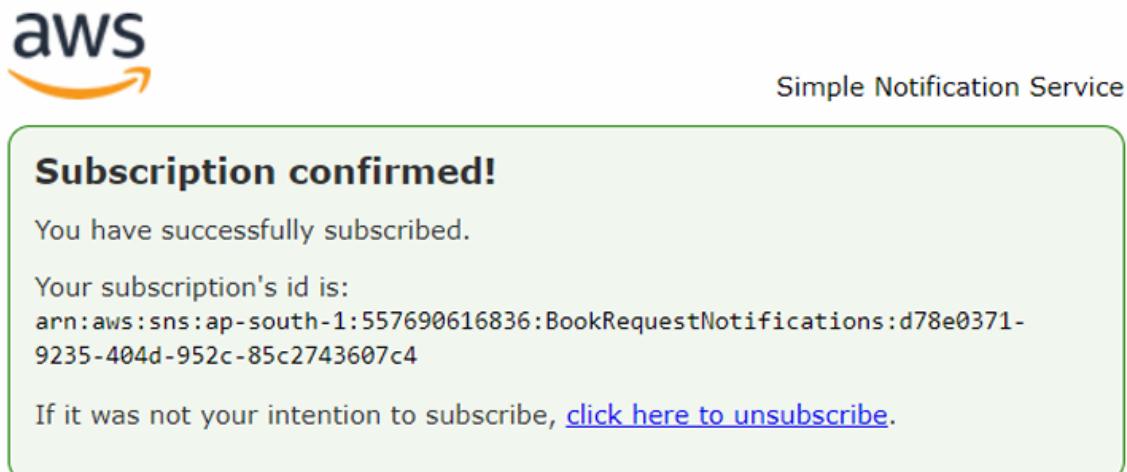
AWS Notification - Subscription Confirmation Inbox ×

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

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](#)



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

The screenshot shows the AWS SNS console with the topic 'BookRequestNotifications'. The 'Details' section displays the topic's name, ARN, and type. A single subscription is listed under the 'Subscriptions' tab, which is currently selected. The subscription is to the email address 'instantslibrary2@gmail.com' and is marked as 'Confirmed'.

5: IAM Role Setup

- **Activity 5.1: Create IAM Role.**

- In the AWS Console, go to IAM and create a new IAM Role for EC2 to interact with DynamoDB and SNS.

The screenshot shows the AWS IAM service page with a search bar at the top. The search results for 'iam' are displayed, including links to 'IAM', 'IAM Identity Center', 'Resource Access Manager', and 'AWS App Mesh'.

The screenshot shows the 'Roles' page within the IAM service. It lists five existing roles. A 'Create role' button is visible in the top right corner.

The screenshots show the AWS IAM 'Create role' wizard. Step 1: Select trusted entity. Step 2: Add permissions.

Step 1: Select trusted entity

- Trusted entity type:**
 - AWS service:** Allows EC2 to access the EC2 Lambda, or others to perform actions in this account.
 - AWS account:** Allows entities in other AWS accounts belonging to you or third-party to perform actions in this account.
 - IAM identity:** Allows entities controlled by the specified central web identity provider to assume the role to perform actions in this account.
 - Lambda function:** Allows entities associated with Lambda to have a separate identity to perform actions in this account.
 - Custom trust policy:** Create a custom trust policy to enable others to perform actions in this account.
- User case:** Allows EC2 to access the EC2 Lambda, or others to perform actions in this account.
- Service or user case:** EC2
- User case details:**
 - AmazonEC2 instance to call AWS services on your behalf:** Selected.
 - EC2 Role for AWS Systems Manager:** Allows EC2 to assume the role and execute AWS Systems Manager on your behalf.
 - EC2 Spot Fleet Role:** Allows EC2 to assume the role and terminate Spot Instances on your behalf.
 - EC2 Auto Scaling Role:** Allows access to actions and instances EC2 can perform on your behalf.
 - EC2 - Spot Fleet Tagging:** Allows EC2 to assume the role and tag instances on your behalf.
 - EC2 - Spot Instances:** Allows EC2 to search and manage spot fleet instances on your behalf.
 - EC2 - Scheduled Instances:** Allows EC2 to manage instances on your behalf.

Step 2: Add permissions

Permissions policies (1/955)

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

Filter by Type: All types | 2 matches

Policy name	Type
AmazonDynamoDBFullAccess	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.

The screenshot shows the 'Add permissions' step of the IAM role creation wizard. On the left, a sidebar lists 'Step 1: Select trusted entities', 'Step 2: Add permissions' (which is selected), and 'Step 3: Name, review, and create'. The main area is titled 'Add permissions' with a sub-section 'Permissions policies (2/955)'. It says 'Choose one or more policies to attach to your new role.' A search bar 'Q: am' and a filter 'All types' are at the top. Below is a table with columns 'Policy name', 'Type', and 'Description'. The table shows five policies:

Policy name	Type	Description
AmazonSNSFullAccess	AWS managed	Full access to Amazon SNS
AmazonSNSReadOnlyAccess	AWS managed	Read-only access to Amazon SNS
AmazonSQSFullAccess	AWS managed	Full access to Amazon SQS
AmazonSSMFullAccess	AWS managed	Full access to AWS Systems Manager
AWSLambdaBasicExecutionRole	AWS managed	Full access to AWS Lambda

At the bottom, there's a link 'Set permissions boundary - optional' and a note about IAM roles having a maximum of 256 permissions. The bottom right has 'Create', 'Previous', and 'Next Step' buttons.

The screenshot shows the 'Name, review, and create' step of the IAM role creation wizard. The left sidebar shows 'Step 1: Select trusted entities', 'Step 2: Add permissions' (selected), and 'Step 3: Name, review, and create'. The main area is titled 'Name, review, and create' with a sub-section 'Role details'. It shows the role name 'aws_lambda_function' and a detailed description of its permissions. Below is the 'Permissions policy summary' section, which includes:

- Policy name:** [AmazonSNSFullAccess](#) (Type: AWS managed)
- Policy name:** [AmazonSQSFullAccess](#) (Type: AWS managed)

At the bottom, there's a note about IAM roles having a maximum of 256 permissions and a 'Create' button.

sns_Dynamodb_role Info

Allows EC2 instances to call AWS services on your behalf.

Summary

Creation date: October 13, 2024, 23:06 (UTC+05:30)

Last activity: 6 days ago

ARN: arn:aws:iam::557690616836:role/sns_Dynamodb_role

Instance profile ARN: arn:aws:iam::557690616836:instance-profile/sns_Dynamodb_role

Maximum session duration: 1 hour

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

Permissions policies (2) Info

You can attach up to 10 managed policies.

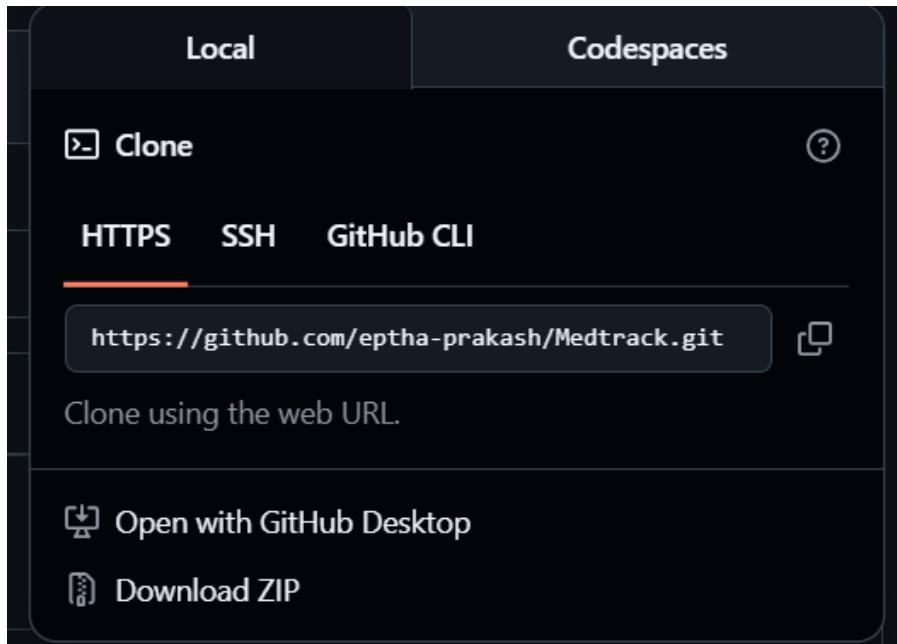
Policy name	Type	Attached entities
AmazonDynamoDBFullAccess	AWS managed	4
AmazonSNSFullAccess	AWS managed	2

Filter by Type: All types

Milestone 6: EC2 Instance Setup

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

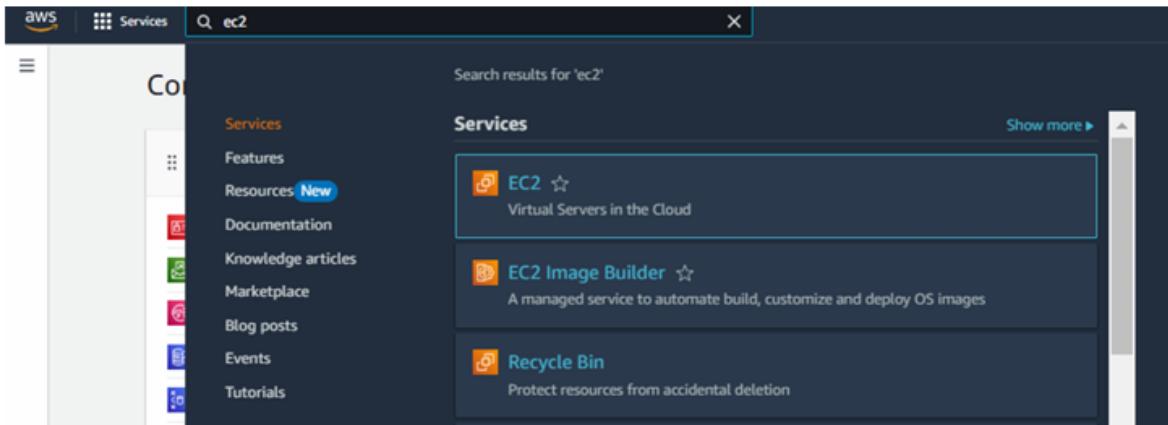
Name	Last commit message	Last commit date
...		
templates	Delete projectfiles/templates/Activities.html	last week
medtrack.py	Add files via upload	last week
readme.md	Create readme.md	last week



- **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.

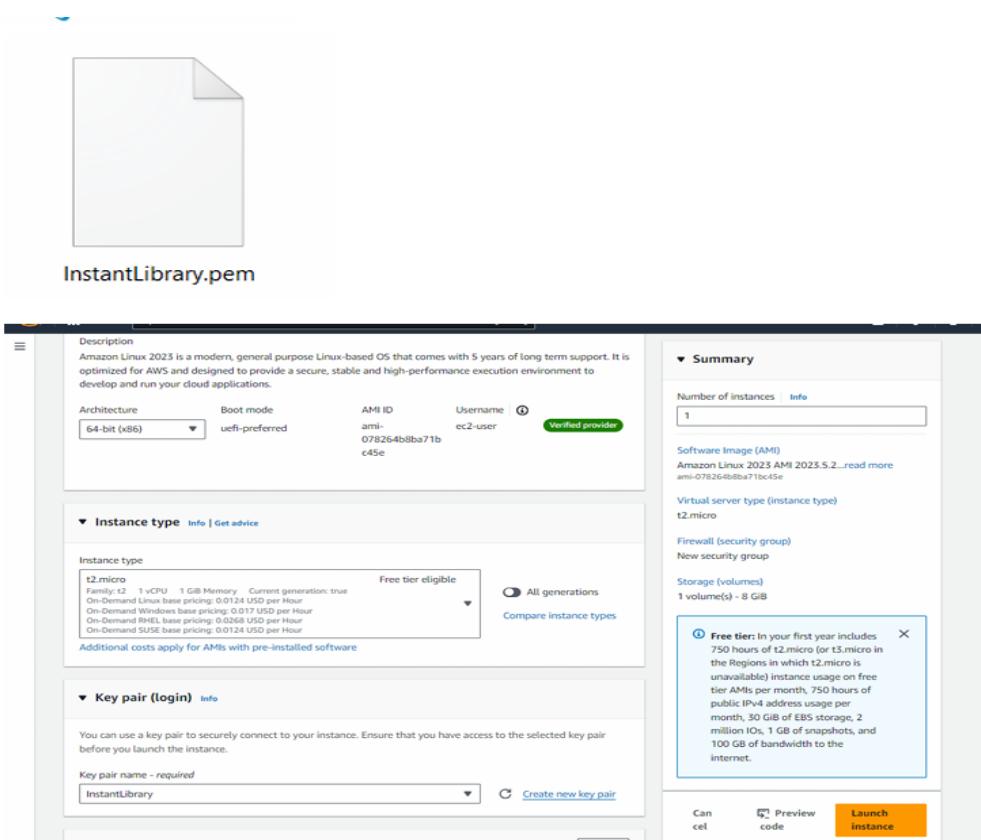


- Click on Launch instance to launch EC2 instance

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with options like EC2 Dashboard, Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, and Savings Plans. The main area is titled 'Instances info' and shows a table with one row: 'No instances'. A 'Launch instances' button is visible. Below this, the 'Launch an instance' wizard is open. It has a 'Name and tags' section where the 'Name' field contains 'InstantLibraryApp'. To the right, there's a 'Summary' section showing 'Number of instances: 1' and details about the selected AMI (Amazon Linux 2023 AMI) and instance type (t2.micro).

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

The screenshot shows the AWS AMI selection page. At the top, there are icons for Amazon Linux, macOS, Ubuntu, Windows, and Red Hat. A search bar on the right says 'Browse more AMIs'. Below, a section for 'Amazon Machine Image (AMI)' shows 'Amazon Linux 2023 AMI' with the ID 'ami-02b49a24cfb95941c'. This AMI is labeled as 'Free tier eligible'. A detailed description of the AMI follows, mentioning it's a modern, general purpose Linux-based OS with 5 years of support. At the bottom, there are dropdowns for 'Architecture' (64-bit (x86)), 'Boot mode' (uefi-preferred), and 'AMI ID' (ami-02b49a24cfb95941c), and a 'Verified provider' badge.



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

Network settings

VPC – required

vpc-03cdc7b6f19dd7211 (default)

Subnet

No preference

Create new subnet

Auto-assign public IP

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group

Select existing security group

Security group name – required

launch-wizard

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and _-:/()#,@[]+=&;!\$^

Description – required

launch-wizard created 2024-10-13T17:49:56.622Z

Inbound Security Group Rules

- Security group rule 1 (TCP, 22, 0.0.0.0/0)

Type: ssh	Protocol: TCP	Port range: 22
Source type: Anywhere	Source: 0.0.0.0/0	Description - optional: e.g. SSH for admin desktop
- Security group rule 2 (TCP, 80, 0.0.0.0/0)

Type: HTTP	Protocol: TCP	Port range: 80
Source type: Custom	Source: 0.0.0.0/0	Description - optional: e.g. SSH for admin desktop
- Security group rule 3 (TCP, 5000, 0.0.0.0/0)

Type: Custom TCP	Protocol: TCP	Port range: 5000
Source type: Custom	Source: 0.0.0.0/0	Description - optional: e.g. SSH for admin desktop

Add security group rule

Success
Successfully started launch of instance i-0019611227bc1299

Launch log

Next Steps

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

Create billing and free tier usage alerts	Connect to your instance	Connect an RDS database	Create EBS snapshot policy	Manage detailed monitoring	Create Load Balancer
To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.	Get your instance's IP address and log into it from your local computer.	Configure the connection between an EC2 instance and a database to allow traffic flow between them.	Create a policy that automates the creation, retention, and deletion of EBS snapshots.	Enable or disable detailed monitoring for the instance. If you enable detailed monitoring, the Amazon EC2 console displays monitoring graphs with a 1 minute period.	Create a application, network gateway or classic Load Balancer.
Create billing alerts	Connect to instance	Connect an RDS database	Create EBS snapshot policy	Manage detailed monitoring	Create Load Balancer
Learn more	Learn more	Create a new RDS database	Learn more	Get instance screenshot	Get system log
Create AWS Budget	Manage CloudWatch alarms	Disaster recovery for your instances	Monitor for suspicious runtime activities	Get instance screenshot	Get system log
AWS Budgets allows you to create budgets. Forecast spend, and take action on your costs and usage from a single location.	Create or update Amazon CloudWatch alarms for the instance.	Recover the instances you just launched into a different Availability Zone or a different Region using AWS Elastic Disaster Recovery (EDR).	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.	Capture a screenshot from the instance and save it as an image. This is useful for troubleshooting an unresponsive instance.	View the instance's system log to troubleshoot issues.
Create AWS Budget	Manage CloudWatch alarms	Disaster recovery for your instances	Monitor for suspicious runtime activities	Get instance screenshot	Get system log
Learn more	Learn more	Create a new RDS database	Learn more	Get instance screenshot	Get system log

View all instances

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

Instances [1/1] Info

Last updated 10 minutes ago (see last activity)

All status ▾

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Private IPv4	Elastic IP	Iv6 Ps	Monitors	Security
InstantLibraryApp	i-001861022fbcac290	Stopped	t2.micro	-	View details	ap-south-1b	-	-	-	-	-	disabled

EC2 > Instances > i-001861022fbcac290 [InstantLibraryApp] Info

Updated less than a minute ago

Instance ID i-001861022fbcac290	Public IPv4 address -	Private IPv4 address 172.31.35
IPv4 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	Auto Scaling Group name -
Assign private resource DNS name IPv4 (1)	Instance type t2.micro	Elastic IP addresses -
Auto-assigned IP address -	VPC ID vpc-03e0c7bf1f9d7211	AWS Compute Optimizer findings Opt-in to AWS Compute Optimizer for recommendations Learn more
IAM Role sns_Dynamodb_role	Subnet ID subnet-009fa144480cc369	Auto Scaling Group ARN arn:aws:autoscaling:ap-south-1:557690616836:instance/i-001861022fbcac290
IMDSv2 Required	Instance ARN arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290	

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

EC2 > Instances > i-001861022fbcac290 [InstantLibraryApp] Info

Updated less than a minute ago

Instance ID i-001861022fbcac290	Public IPv4 address -	Private IPv4 address 172.31.35
IPv4 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	Auto Scaling Group name -
Assign private resource DNS name IPv4 (1)	Instance type t2.micro	Elastic IP addresses -
Auto-assigned IP address -	VPC ID vpc-03e0c7bf1f9d7211	AWS Compute Optimizer findings Opt-in to AWS Compute Optimizer for recommendations Learn more
IAM Role sns_Dynamodb_role	Subnet ID subnet-009fa144480cc369	Auto Scaling Group ARN arn:aws:autoscaling:ap-south-1:557690616836:instance/i-001861022fbcac290
IMDSv2 Required	Instance ARN arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290	

EC2 > Instances > i-001861022fbcac290 [InstantLibraryApp] Info

Updated less than a minute ago

Instance ID i-001861022fbcac290	Public IPv4 address -	Private IPv4 address 172.31.35
IPv4 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	Auto Scaling Group name -
Assign private resource DNS name IPv4 (1)	Instance type t2.micro	Elastic IP addresses -
Auto-assigned IP address -	VPC ID vpc-03e0c7bf1f9d7211	AWS Compute Optimizer findings Opt-in to AWS Compute Optimizer for recommendations Learn more
IAM Role sns_Dynamodb_role	Subnet ID subnet-009fa144480cc369	Auto Scaling Group ARN arn:aws:autoscaling:ap-south-1:557690616836:instance/i-001861022fbcac290
IMDSv2 Required	Instance ARN arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290	

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

Modify IAM role Info

Attach an IAM role to your instance.

Instance ID
i-001861022fbcac290 (InstantLibraryApp)

IAM role

Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

sns_Dynamodb_role

Create new IAM role

Cancel **Update IAM role**

- Now connect the EC2 with the files

Connect to instance Info

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

[EC2 Instance Connect](#) | [Session Manager](#) | [SSH client](#) | [EC2 serial console](#)

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

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

Instance ID
 i-001861022fbcac290 (InstantLibraryApp)

Connection Type

[Connect using EC2 Instance Connect](#)
 Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.

[Connect using EC2 Instance Connect Endpoint](#)
 Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

[Public IPv4 address](#)
 13.200.229.59

[IPv6 address](#)
 —

Username
 Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.
 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
[ec2-user@ip-172-31-3-5 ~]$ Amazon Linux 2023
[ec2-user@ip-172-31-3-5 ~]$ https://aws.amazon.com/linux/amazon-linux-2023
[ec2-user@ip-172-31-3-5 ~]$ 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/eptha-prakash/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/AlekhyaPenukula/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
[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
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

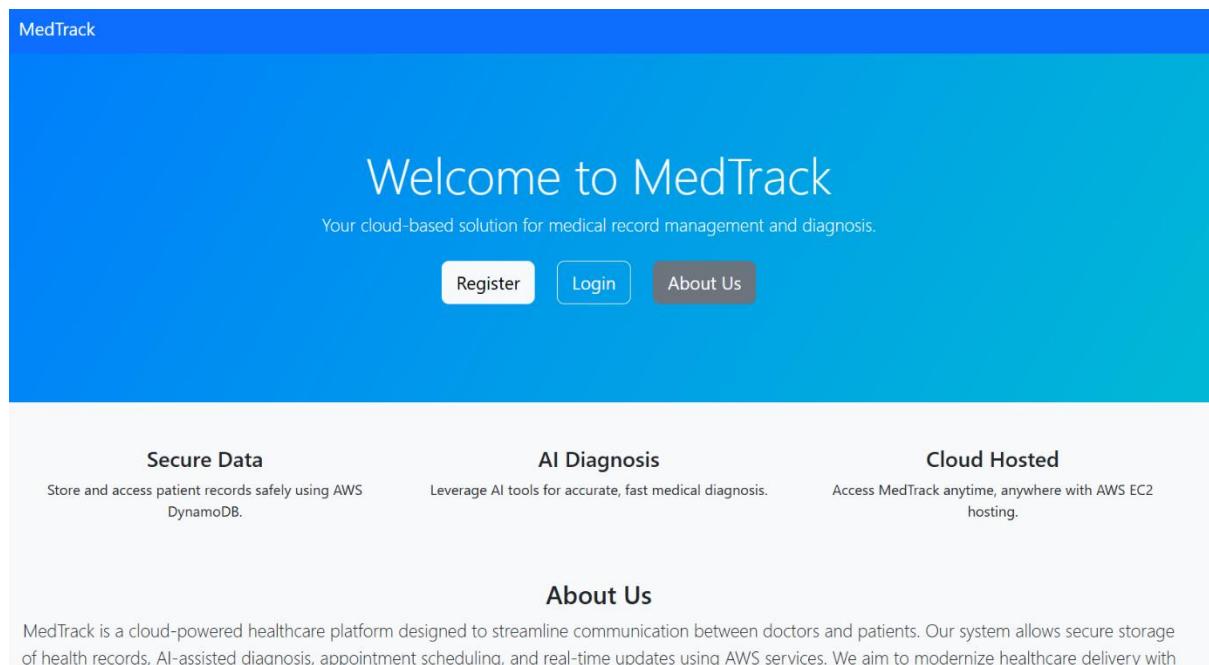
```
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:80
* Running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

Home Page:



Register page:

Register

Name

Email

Password

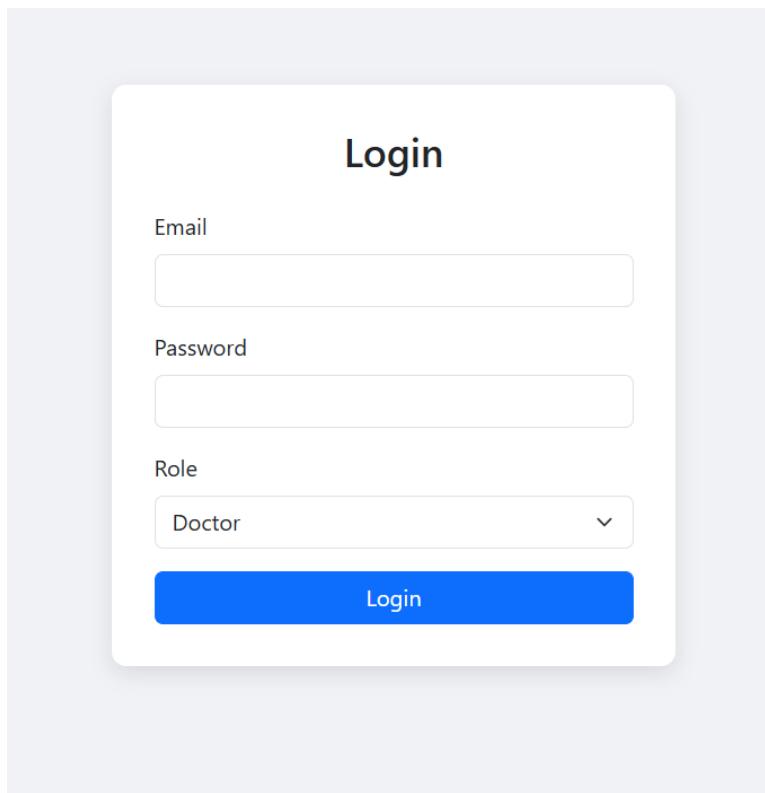
Role

Age

Gender

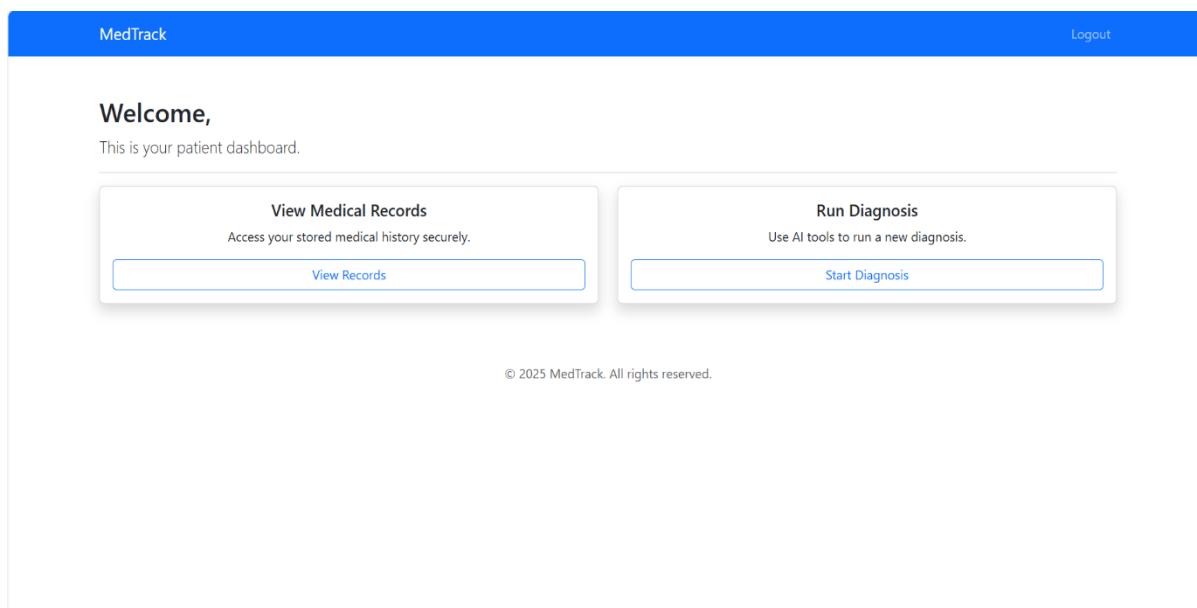
Register

Login Page:

**About Us Page:**

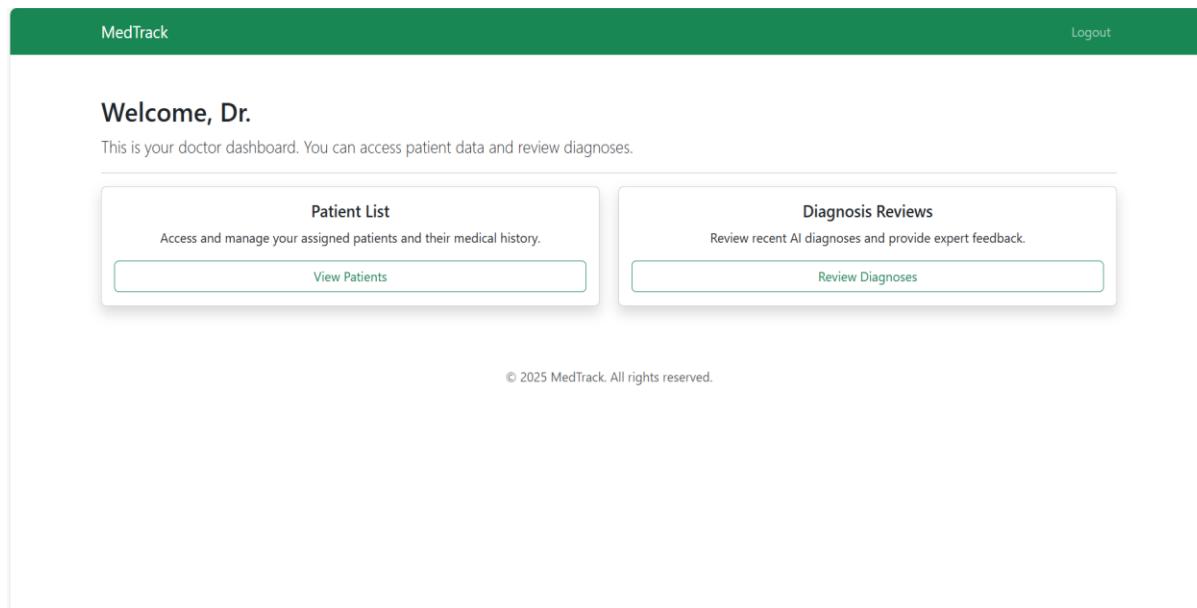
A screenshot of the "About Us" page for MedTrack. The top section has a teal header with the title "Welcome to MedTrack" and a subtitle "Your cloud-based solution for medical record management and diagnosis." Below the header are three buttons: "Register", "Login", and "About Us". The main content area is white and contains three sections: "Secure Data", "AI Diagnosis", and "Cloud Hosted". Each section has a brief description and a link to "About Us". At the bottom, there is a larger "About Us" section with a detailed description of the platform's purpose and technology.

Patient Dashboard:



The screenshot shows the MedTrack Patient Dashboard. At the top, a blue header bar contains the "MedTrack" logo on the left and a "Logout" link on the right. Below the header, a welcome message reads "Welcome," followed by the text "This is your patient dashboard." Two main action 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."). Each button has a corresponding "View Records" or "Start Diagnosis" link below it. At the bottom center of the page is a copyright notice: "© 2025 MedTrack. All rights reserved."

Doctor Dashboard:



The screenshot shows the MedTrack Doctor Dashboard. At the top, a dark green header bar contains the "MedTrack" logo on the left and a "Logout" link on the right. Below the header, a welcome message reads "Welcome, Dr." followed by the text "This is your doctor dashboard. You can access patient data and review diagnoses." Two main action buttons are displayed: "Patient List" (with the sub-instruction "Access and manage your assigned patients and their medical history.") and "Diagnosis Reviews" (with the sub-instruction "Review recent AI diagnoses and provide expert feedback."). Each button has a corresponding "View Patients" or "Review Diagnoses" link below it. At the bottom center of the page is a copyright notice: "© 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.

Reference :

Github : <https://github.com/eptha-prakash/Medtrack.git>

Video demo : https://drive.google.com/drive/folders/1NC7-2I_dYx8HPYgeG4sDNl4O-tZZSzxo?usp=sharing