

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.

Scenarios:

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.

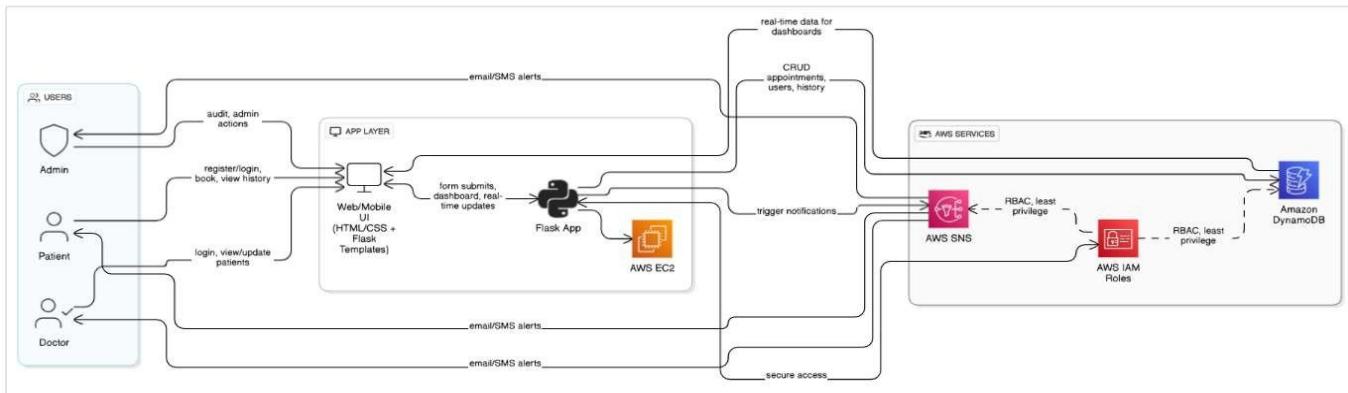
Scenario 2: Secure User Management with IAM

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.

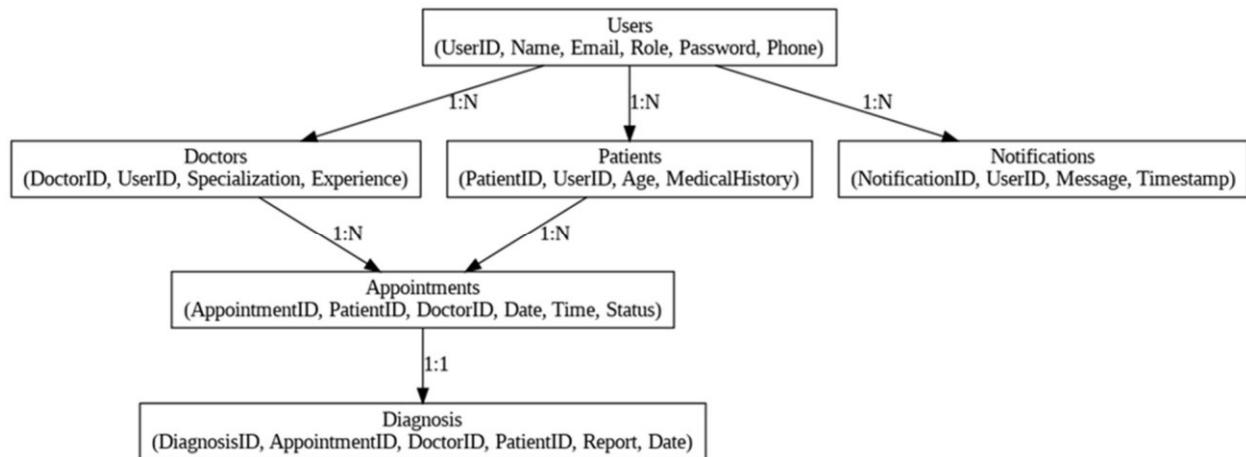
Scenario 3: Easy Access to Medical History and Resources

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**

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

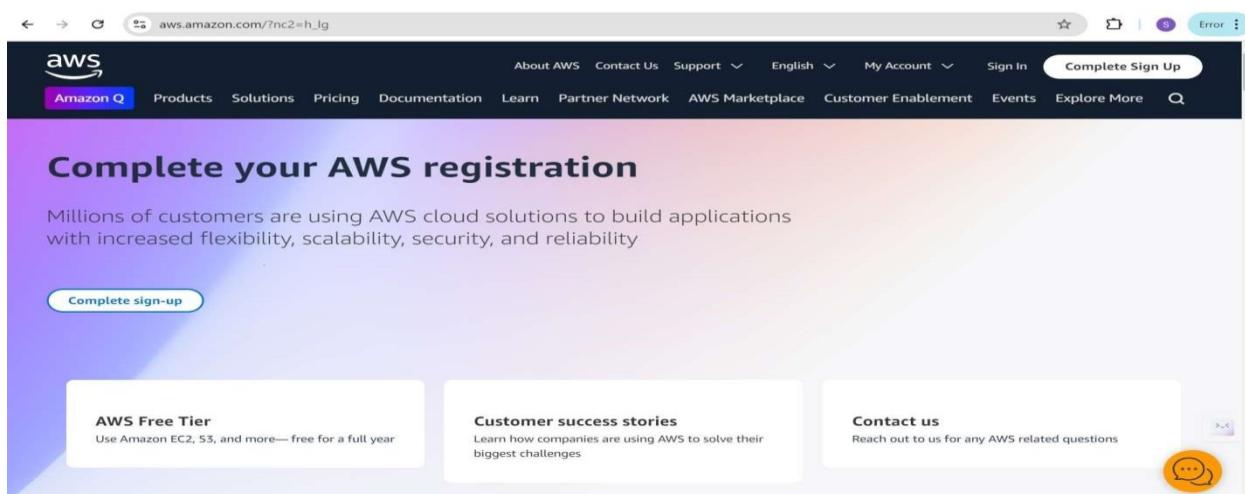
- Upload Flask Files
- Run the Flask App

Milestone 8. Testing and Deployment

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

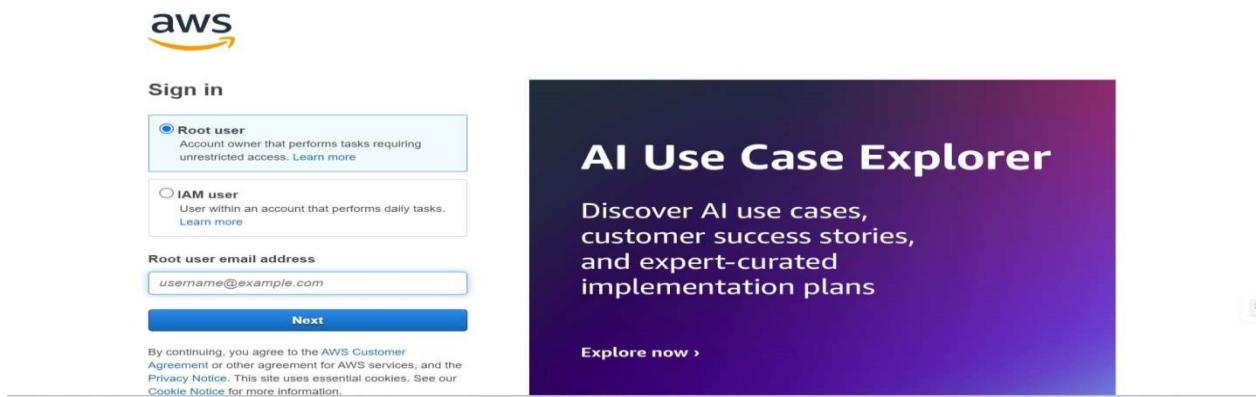
Milestone 1: Web Application Development and Setup

- Activity 1.1: Set up an AWS account if not already done.
 - Sign up for an AWS account and configure billing settings.



- Activity 1.2: Log in to the AWS Management Console

- After setting up your account, log in to the [AWS Management Console](#).

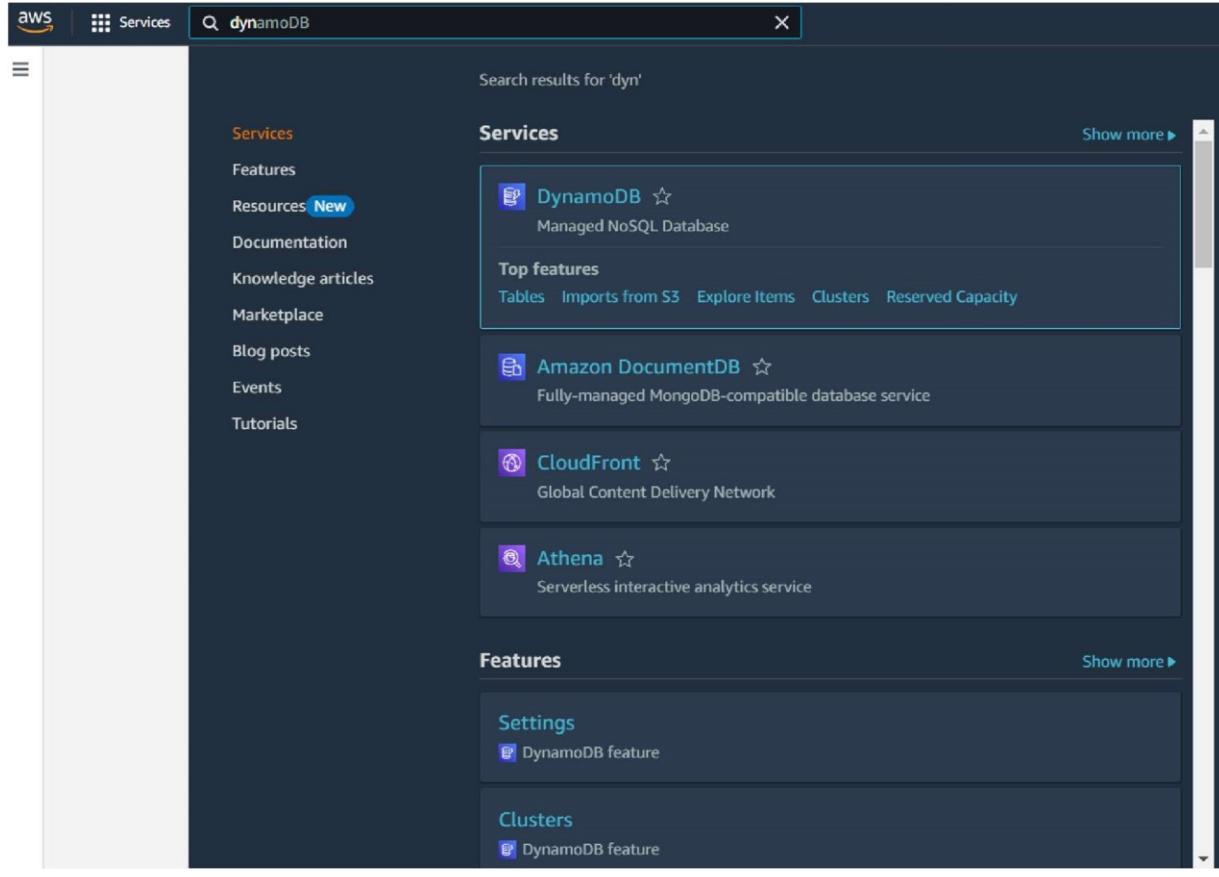


The left screenshot shows the AWS sign-in interface. It has a 'Sign in' header, two radio button options ('Root user' and 'IAM user'), a 'Root user email address' input field containing 'username@example.com', and a 'Next' button. Below the input field is a small note about cookie consent. The right screenshot shows the 'AI Use Case Explorer' landing page with a dark purple gradient background. The title 'AI Use Case Explorer' is at the top, followed by the text 'Discover AI use cases, customer success stories, and expert-curated implementation plans'. At the bottom is a 'Explore now >' button.

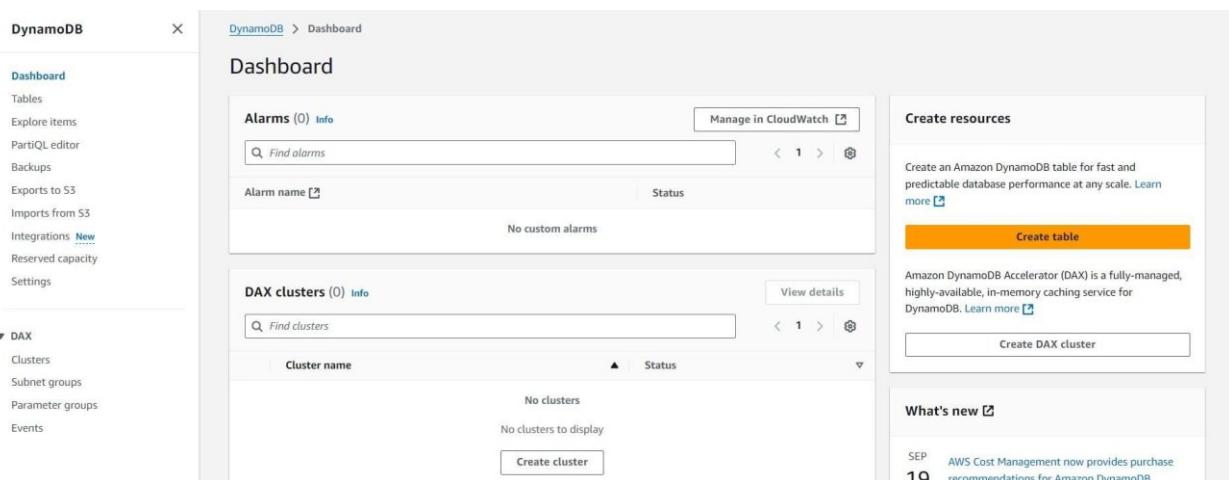
Milestone 2: DynamoDB Database Creation and Setup

- Activity 2.1: Navigate to the DynamoDB

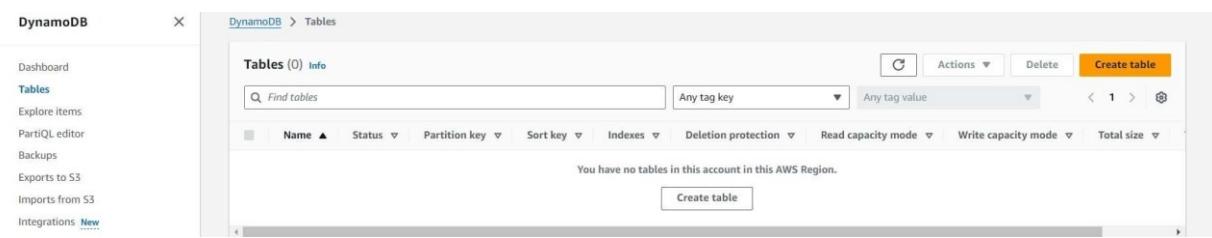
- In the AWS Console, navigate to DynamoDB and click on create tables.



The screenshot shows the AWS Services search results page. The search bar at the top contains 'dynamodb'. On the left, there is a sidebar with links to Services, Features, Resources (New), Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main area displays a list of services under the heading 'Services'. The first item in the list is 'DynamoDB' with a star icon, described as a 'Managed NoSQL Database'. Below it is 'Amazon DocumentDB' with a star icon, described as a 'Fully-managed MongoDB-compatible database service'. Other visible services include CloudFront, Athena, and Lambda. At the bottom, there are sections for 'Features' (Settings, Clusters) and 'Show more ▶' buttons.

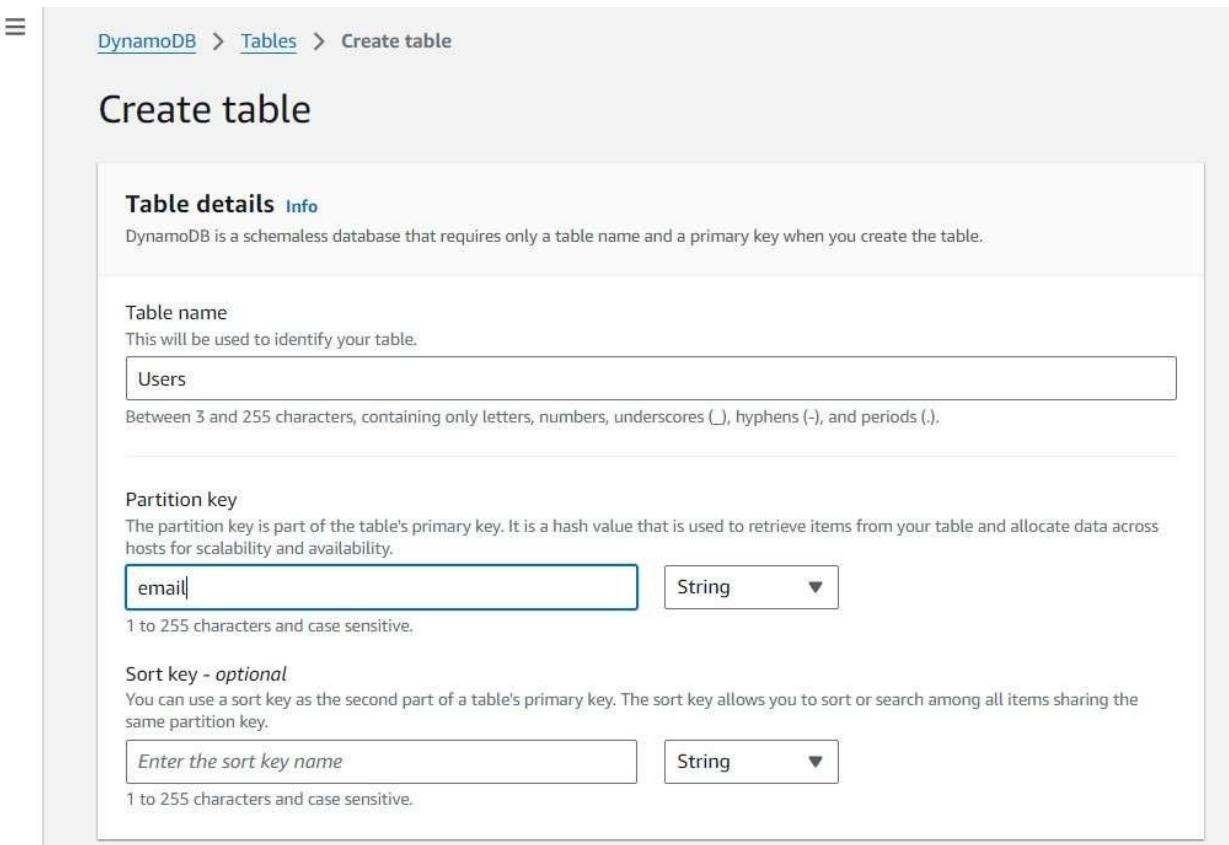


The screenshot shows the DynamoDB Dashboard. The left sidebar has a 'Dashboard' section with links to Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations (New), Reserved capacity, and Settings. It also has a 'DAX' section with links to Clusters, Subnet groups,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



The screenshot shows the AWS DynamoDB 'Tables' page. On the left, there is a sidebar with links like 'Dashboard', 'Tables', 'Explore items', 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', and 'Integrations'. The main area has a header 'DynamoDB > Tables' and a sub-header 'Tables (0) Info'. It features a search bar 'Find tables', a dropdown for 'Any tag key', and a 'Create table' button. Below the header, it says 'You have no tables in this account in this AWS Region.' and has a 'Create table' button.

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



The screenshot shows the 'Create table' wizard. At the top, the breadcrumb navigation shows 'DynamoDB > Tables > Create table'. The main title is 'Create table'. Under 'Table details', it says 'DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.' The 'Table name' field is set to 'Users'. Below it, a note says 'Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).'. Under 'Partition key', it says 'The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.' The 'email' field is selected as the partition key, and its type is set to 'String'. A note below says '1 to 255 characters and case sensitive.'. Under 'Sort key - optional', it says 'You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.' The 'Enter the sort key name' field is empty, and its type is set to 'String'. A note below says '1 to 255 characters and case sensitive.'

Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes
Resource-based policy	Not active	Yes

Tags

Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

[Add new tag](#)

You can add 50 more tags.

[Cancel](#)

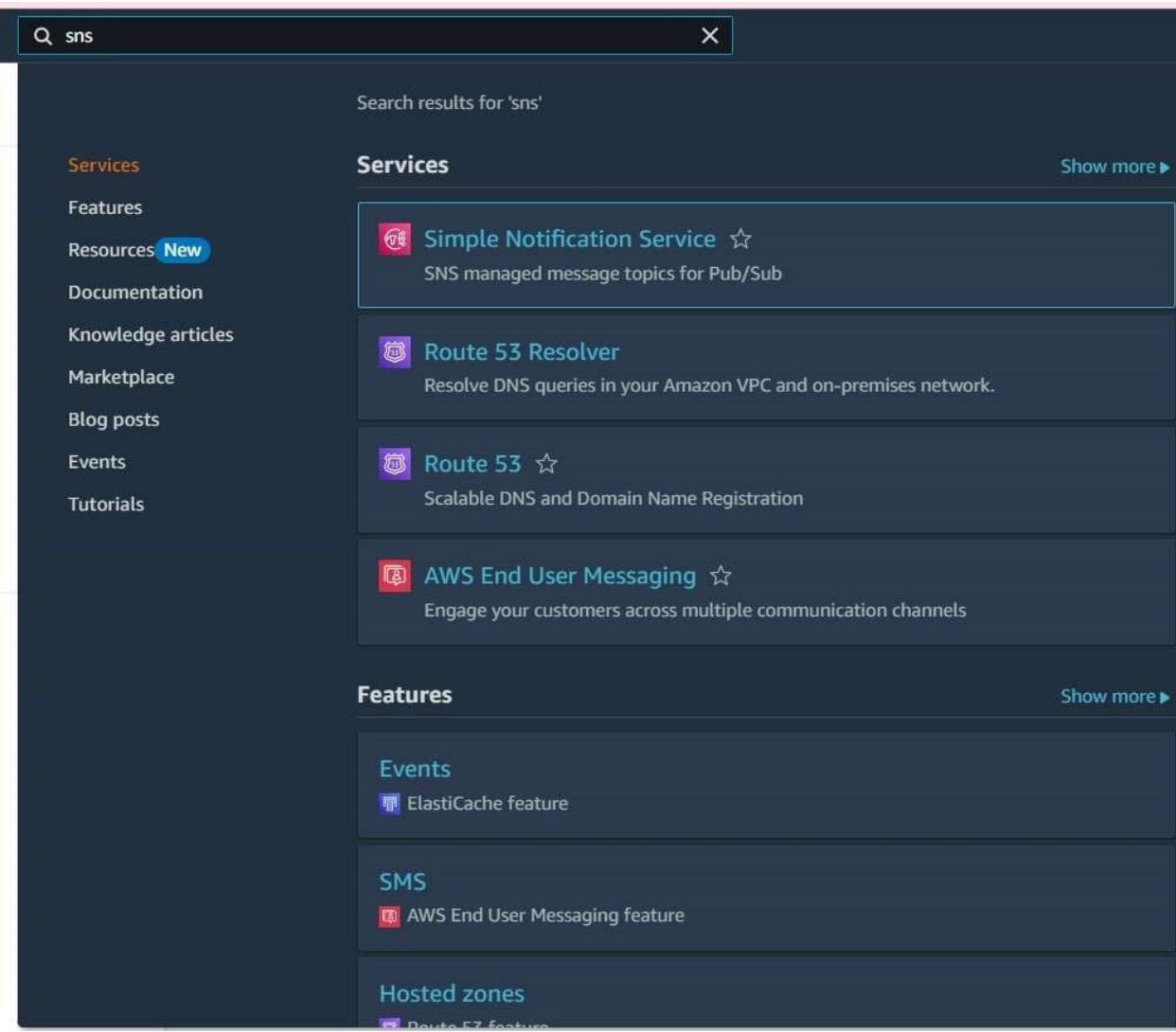
[Create table](#)

Tables (2) Info								
	Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favor
<input type="checkbox"/>	Appointments	<input checked="" type="radio"/> Active	appointment_id (\$)	-	0	0	<input checked="" type="radio"/> Off	
<input type="checkbox"/>	users	<input checked="" type="radio"/> Active	email (\$)	-	0	0	<input checked="" type="radio"/> Off	

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

Milestone 3: SNS Noti ication 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.



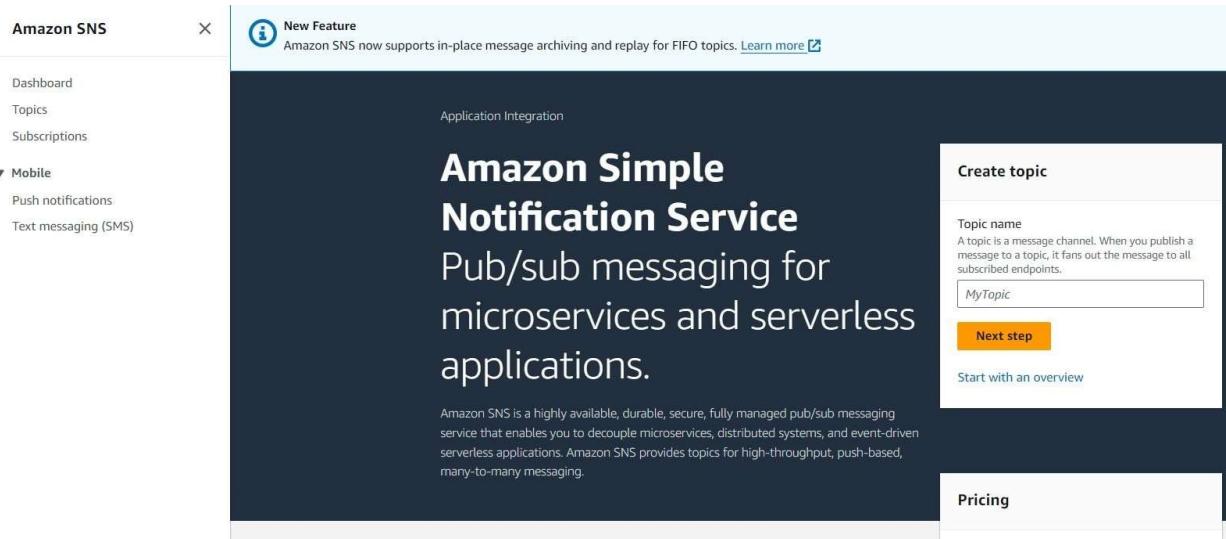
The screenshot shows the AWS search interface with the query 'sns' entered in the search bar. The search results page displays a list of services and features related to Simple Notification Service (SNS).

Services

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

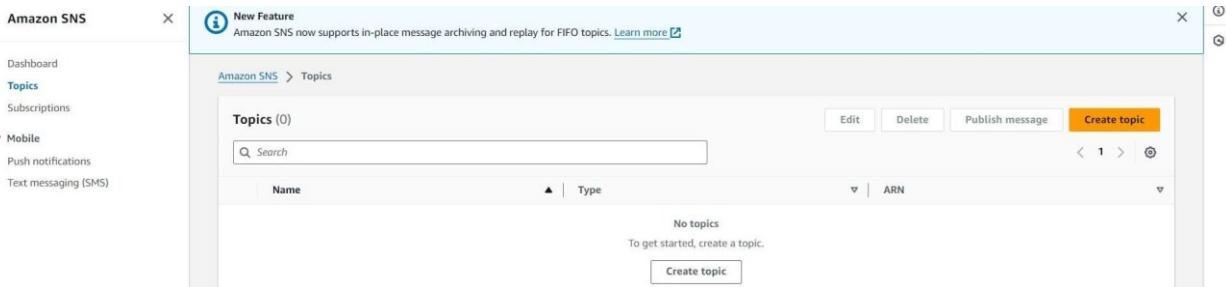
Features

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



The screenshot shows the Amazon Simple Notification Service (SNS) home page. On the left, there is a navigation sidebar with links for Dashboard, Topics, Subscriptions, Mobile (Push notifications, Text messaging (SMS)), and a New Feature notice about in-place message archiving and replay for FIFO topics. The main content area features a dark header "Application Integration" and a large title "Amazon Simple Notification Service" followed by a subtitle "Pub/sub messaging for microservices and serverless applications." Below the title, a paragraph describes Amazon SNS as a highly available, durable, secure, fully managed pub/sub messaging service. A "Create topic" button is prominently displayed on the right side of the main content area.

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



The screenshot shows the "Topics" page within the Amazon SNS service. The left sidebar includes links for Dashboard, Topics (which is selected), Subscriptions, and Mobile (Push notifications, Text messaging (SMS)). The main content area displays a table titled "Topics (0)" with columns for Name, Type, and ARN. A search bar is at the top of the table. A "Create topic" button is located at the bottom of the table. A message at the bottom of the page says "No topics. To get started, create a topic." and features a "Create topic" button.

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

Amazon SNS > Topics > Create topic

New Feature

Amazon SNS now supports High Throughput FIFO topics. [Learn more](#)

Create topic

Details

Type | [Info](#)

Topic type cannot be modified after topic is created

FIFO (first-in, first-out)

- Strictly-preserved message ordering
- Exactly-once message delivery
- Subscription protocols: SQS

Standard

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

Name

Medtrack

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 > Topics > Medtrack

New Feature

Amazon SNS now supports High Throughput FIFO topics. [Learn more](#)

Topic Medtrack created successfully.

You can create subscriptions and send messages to them from this topic.

Medtrack

Details

Name	Medtrack
ARN	arn:aws:sns:ap-south-1:940482422578:Medtrack
Type	Standard

Display name

-

Topic owner

940482422578

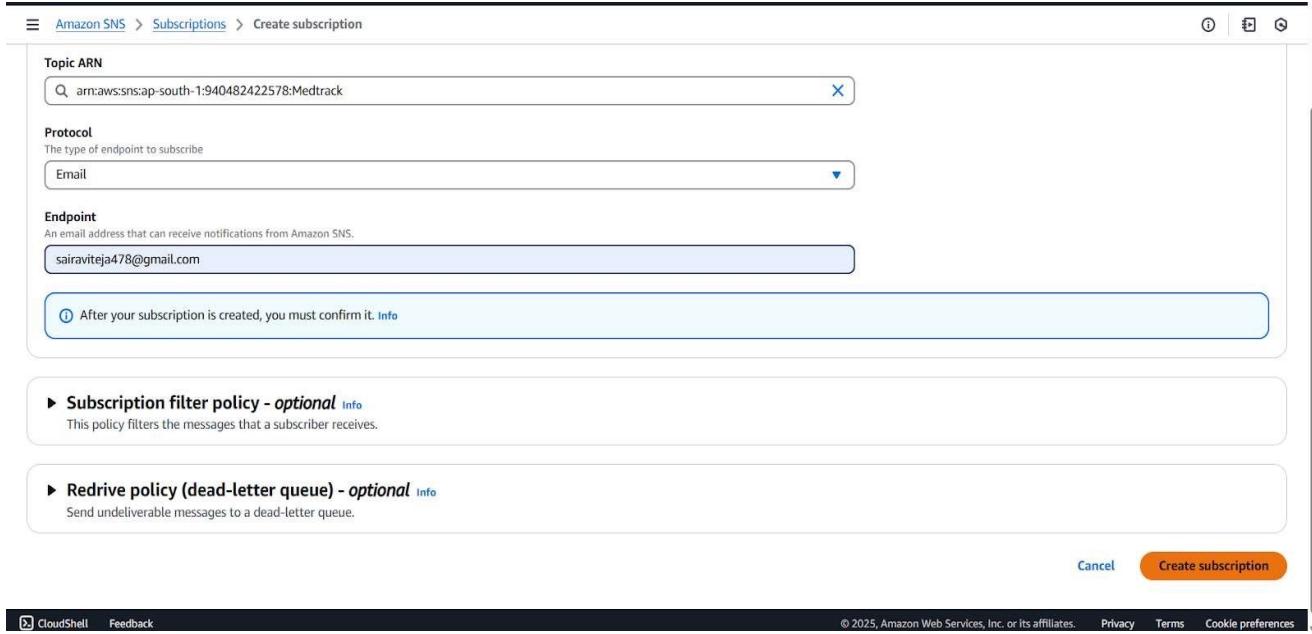
Subscriptions (1)

[Edit](#) [Delete](#) [Request confirmation](#) [Confirm subscription](#) [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

Topic ARN
 X

Protocol
 The type of endpoint to subscribe
 ▼

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

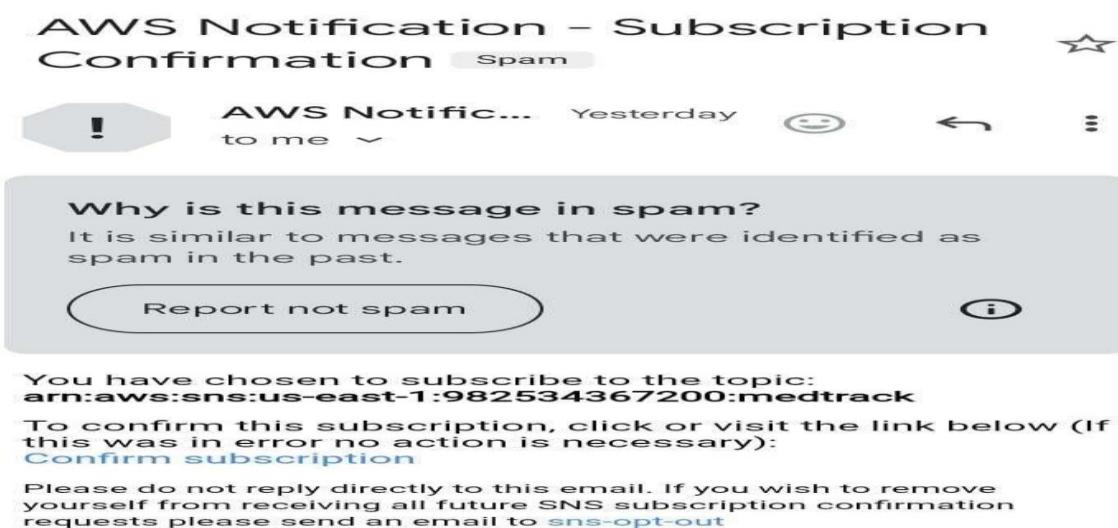
ⓘ After your subscription is created, you must confirm it. [Info](#)

► **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

- After subscription request for the mail confirmation



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



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.

Milestone 4: Backend Development and Application Setup

- Activity 4.1: Develop the backend using Flask

- File Explorer Structure

```
myapp/
    __init__.py
    config.py
    models.py
    routes.py
    static/
        styles.css
    templates/
        index.html
        aboutus.html
```

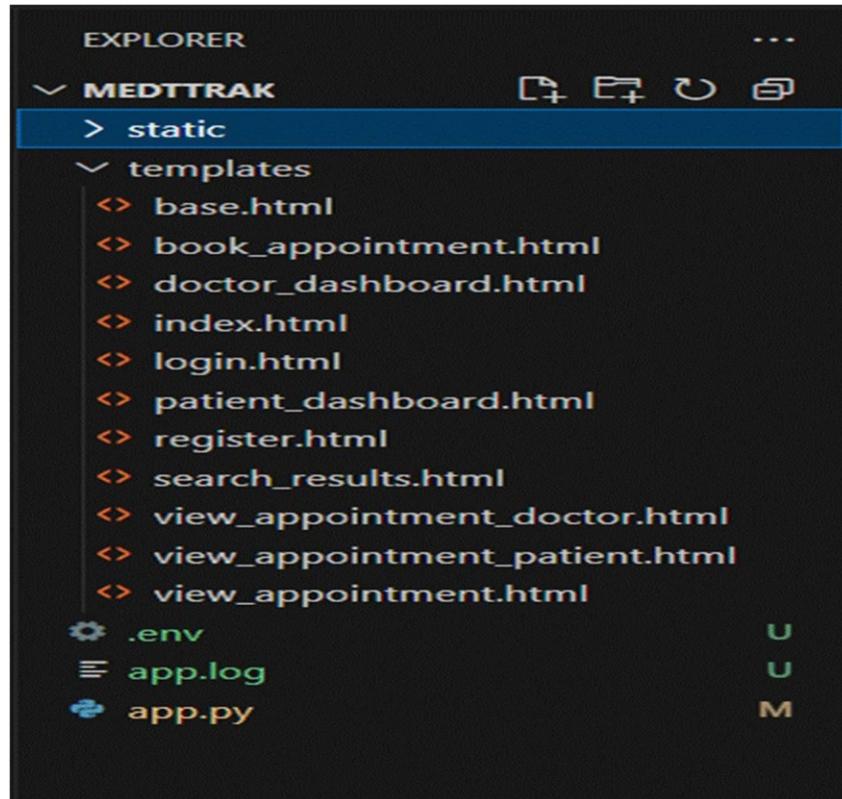
```
from flask import Flask, render_template, request, redirect, url_for, session, flash
from werkzeug.utils import secure_filename
import boto3
import uuid
import os
app = Flask(__name__)
app.secret_key = 'your_secret_key'
users = []
IS_DEV = True # Change to False when AWS credentials are provided in lab
if not IS_DEV:
    import boto3
    dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
    table = dynamodb.Table('Appointments')
    sns = boto3.client('sns', region_name='us-east-1')
else:
    print("Running in development mode. AWS services are disabled.")

# AWS Configuration
dynamodb = boto3.resource('dynamodb', region_name='us-east-1') # e.g., 'us-east-1'
table = dynamodb.Table('Appointments')

# AWS SNS (will be configured next)
#sns_client = sns.Client('sns', region_name='us-east-1')

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

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



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 :

```

app.py / ...
from flask import Flask, render_template, request, redirect, url_for, session, flash
from werkzeug.utils import secure_filename
import boto3
import uuid
import os
    
```

- Flask App Initialization

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:

```
# AWS Configuration
dynamodb = boto3.resource('dynamodb', region_name='us-east-1') # e.g., 'us-east-1'
table = dynamodb.Table('Appointments')
```

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

```
# AWS SNS (will be configured next)
sns = boto3.client('sns', region_name='us-east-1')
sns.create_topic()
```

Description: Configure SNS to send notifications when a book request is submitted. Paste your stored ARN link in the sns_topic_arn space, along with the region name where the SNS topic is created. Also, specify the chosen email service in SMTP_SERVER (e.g., Gmail, Yahoo, etc.) and enter the subscribed email in the SENDER_EMAIL section. Create an ‘App password’ for the email ID and store it in the SENDER_PASSWORD section.

- Routes for Web Pages
- index Route:

```
app.route('/')
def index():
    return render_template('index.html')
```

Description: define the index route / to automatically redirect users to the signup or login or aboutus or contactus page based on button we have chosen when they access the base URL.

□ Sign up Route:

```

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

        users_table = dynamodb.Table('users') # Connect to DynamoDB users table

        try:
            # ■■■ check if user already exists
            response = users_table.get_item(Key={'email': email})
            if 'Item' in response:
                return render_template('signup.html', error="Email already registered. Please use a different one.")

            # ■■■ optional: Hash the password
            hashed_password = generate_password_hash(password)

            # ■■■ Save user to DynamoDB
            users_table.put_item(
                Item={
                    'email': email,
                    'name': username,
                    'password': hashed_password,
                    'role': role,
                    'phone_no': phone_no
                }
            )

            return redirect('/login')
        except Exception as e:
            return render_template('signup.html', error="Signup failed: " + str(e))
    return render_template('signup.html')
    
```

Description: define signup route to validate create account form fields, hash the user password using Bcrypt, store the new user in DynamoDB with a login count, and redirect to login page.

● login Route (GET/POST):

```

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

        if not email or not password:
            return render_template('login.html', error='Please enter email and password')
        try:
            users_table = dynamodb.Table('users') # Connect to DynamoDB users table
            response = users_table.get_item(Key={'email': email})
            user = response.get('Item')

            if user and password == user['password']: # For plain-text passwords
                # If using hashed passwords, replace the above with:
                # If user and check_password_hash(user['password'], password):
                session['user_email'] = email
                session['username'] = user.get('name', '')
                session['role'] = user.get('role')

                print(session)

                if session['role'] == 'patient':
                    return redirect(url_for('patient_dashboard'))
                elif session['role'] == 'doctor':
                    session['specialization'] = user.get('specialization', '')
                    return redirect(url_for('doctor_dashboard'))
                else:
                    return render_template('login.html', error='Invalid role')
            else:
                return render_template('login.html', error='Invalid credentials')
        except Exception as e:
            return render_template('login.html', error=f"Login failed: {str(e)}")
    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 patient dashboard or doctor dashboard based on button they have chosen.

Aboutus and contactus routes:

```

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

@app.route('/contactus')
def contactus():
    return render_template('contactus.html')
from werkzeug.security import generate_password_hash
    
```

Description: define /aboutus and contactus page to render from the index page based on buttons to handle redirection to aboutus page and contactus page and /<aboutus>.html and <contactus>.html dynamic route to render from/to specific pages.

- 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.

Logout Route:

```

1
2     @app.route('/logout')
3     def logout():
4         session.clear()
5         return redirect('/')

```

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.

Book Appointment route:

```

@app.route('/book_appointment', methods=['GET', 'POST'])
def appointment():
    if request.method == 'POST':
        # Extract form data
        name = request.form['name']
        email = request.form['email']
        date = request.form['date']
        time = request.form['time']
        age = request.form['age']
        gender = request.form['gender']
        problem = request.form['problem']
        specialization = request.form['specialization']
        filename = None

        # Handle file upload
        file = request.files.get('report')
        if file and file.filename != '':
            filename = secure_filename(file.filename)
            file.save(os.path.join("static/uploads", filename))

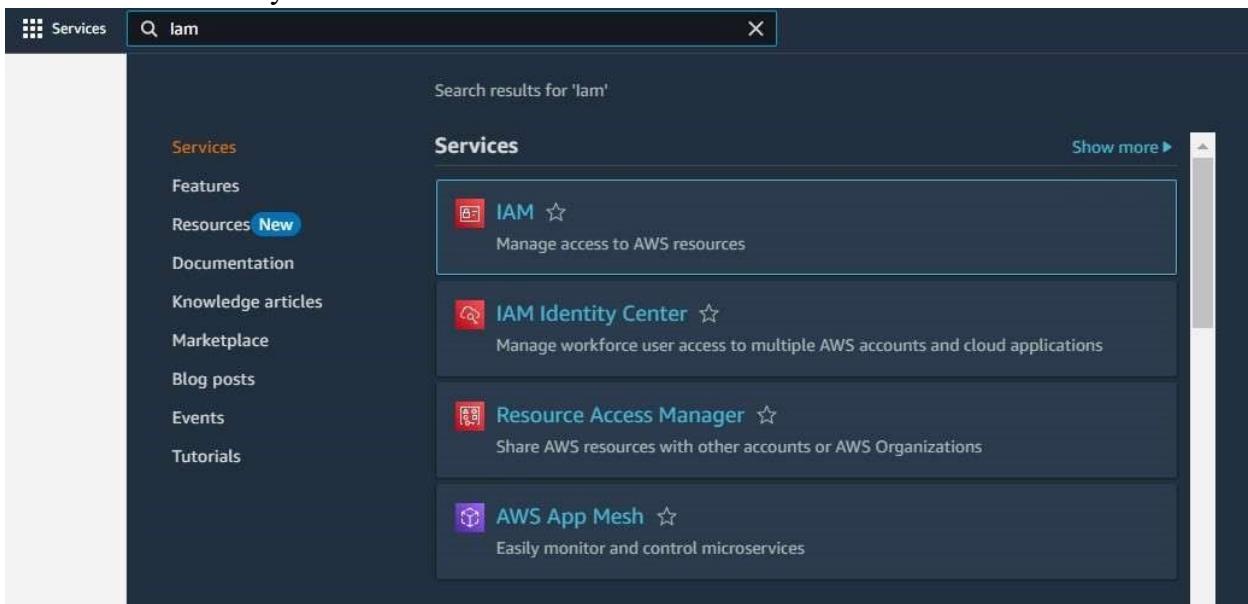
        appointment_id = str(uuid.uuid4())

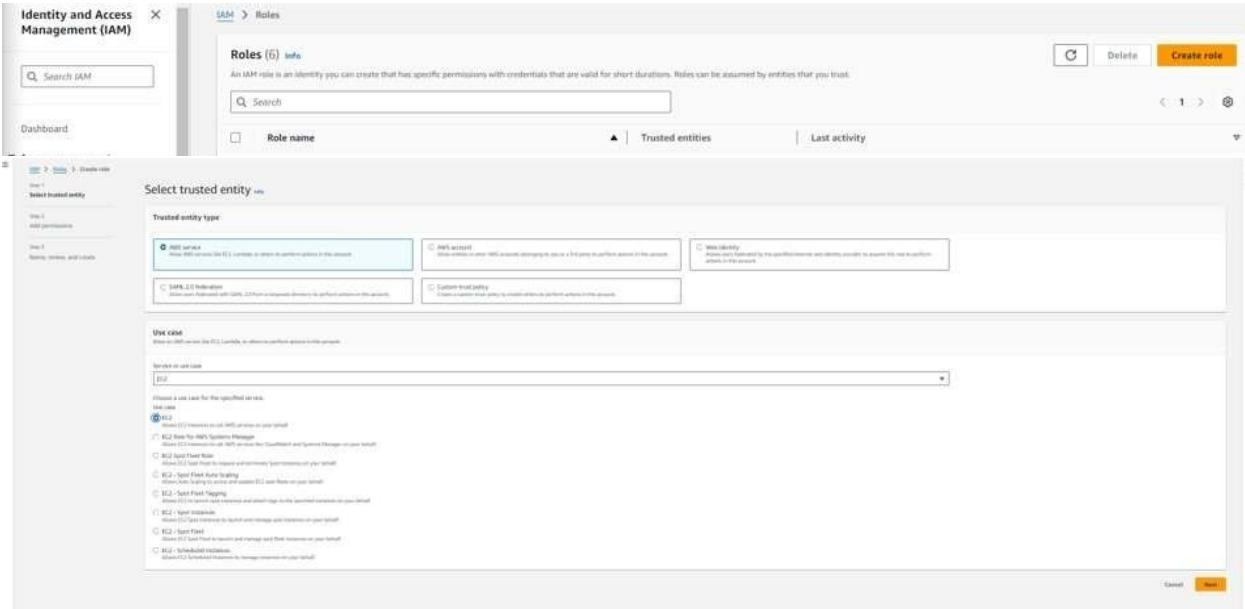
```

Milestone 5: IAM Role Setup

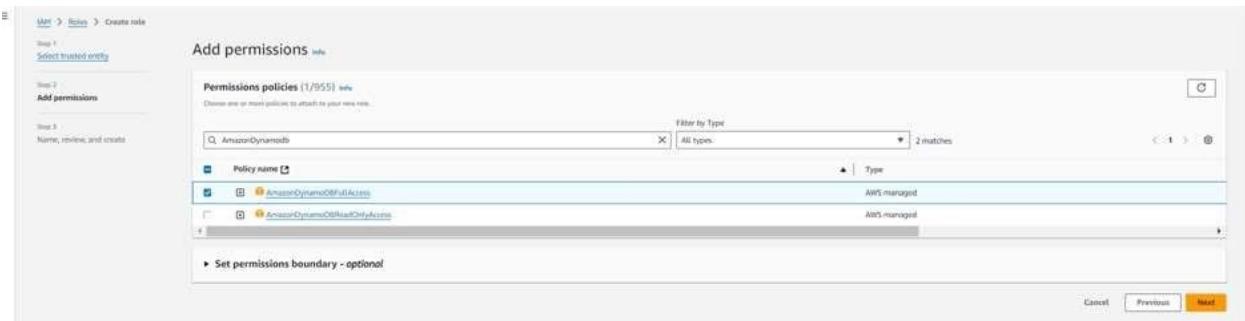
- Activity 5.1: Create IAM Role.

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





The screenshot shows the AWS IAM 'Create role' wizard at Step 1: Select trusted entity. The user has selected 'Amazon EC2' as the trusted entity type. Under 'Use case', 'EC2' is chosen as the service or use case. A list of specific EC2 permissions is shown, with 'AmazonEC2FullAccess' being selected.



The screenshot shows the 'Add permissions' step of the IAM role creation wizard. The user has searched for 'AmazonDynamoDB' and found two results: 'AmazonDynamoDBFullAccess' and 'AmazonDynamoDBReadOnlyAccess'. Both are listed under the 'AWS-managed' category.

- **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.

Step 1: Select trusted entity

Add permissions

Permissions policies (1/955) Info

Create one or more policies to attach to your role.

Filter by Type

Q: uni X All types 3 matches

Policy name	Type
<input checked="" type="checkbox"/>  AmazonSNSFullAccess	AWS managed
<input type="checkbox"/>  AmazonSQSReadOnlyAccess	AWS managed
<input type="checkbox"/>  AmazonVPCFullAccess	AWS managed
<input type="checkbox"/>  AWSLambdaRunAsRolePolicy	AWS managed
<input type="checkbox"/>  AWSDeviceDefenderPolicyForLambdaExecution	AWS managed

Set permissions boundary - optional

[Create new policy](#)

[Cancel](#) [Previous](#) **Next**

Step 1: Select trusted entities

Name, review, and create

Rate details

Role name:

Description:

Step 2: Add permissions

Permissions policy summary

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

Step 3: Add tags

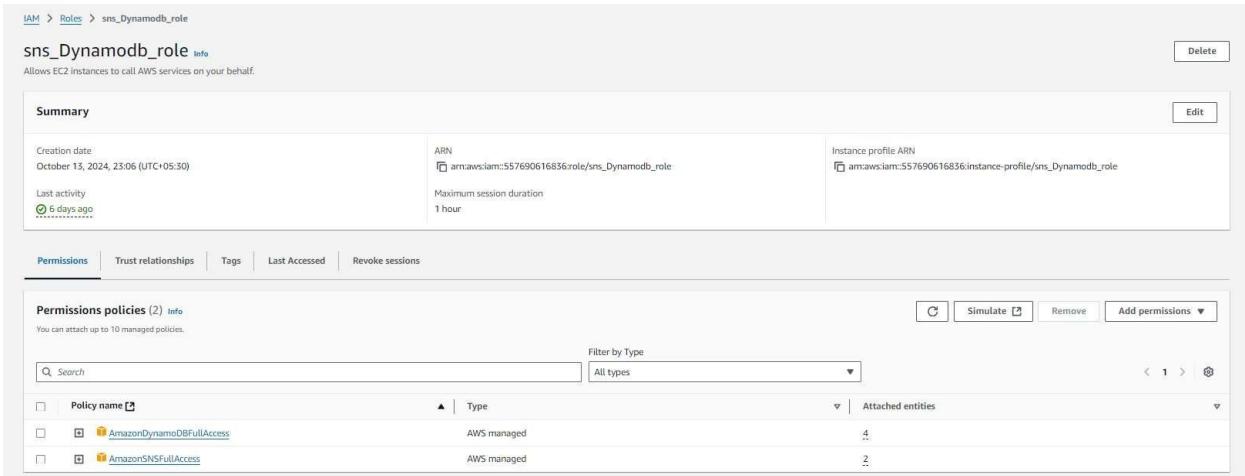
Add tags - optional. Tags are key-value pairs that you can use to identify, organize, or search for resources.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 tags.

[Cancel](#) [Previous](#) **Create role**



IAM > Roles > sns_Dynamodb_role

sns_Dynamodb_role [Info](#)

Allows EC2 instances to call AWS services on your behalf.

Summary

Creation date: October 15, 2024, 23:06 (UTC+05:30)
 Last activity: 6 days ago

ARN: arn:aws:iam::557690616836:role/sns_Dynamodb_role
 Maximum session duration: 1 hour

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

Edit **Delete**

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

Permissions policies (2) [Info](#)

You can attach up to 10 managed policies.

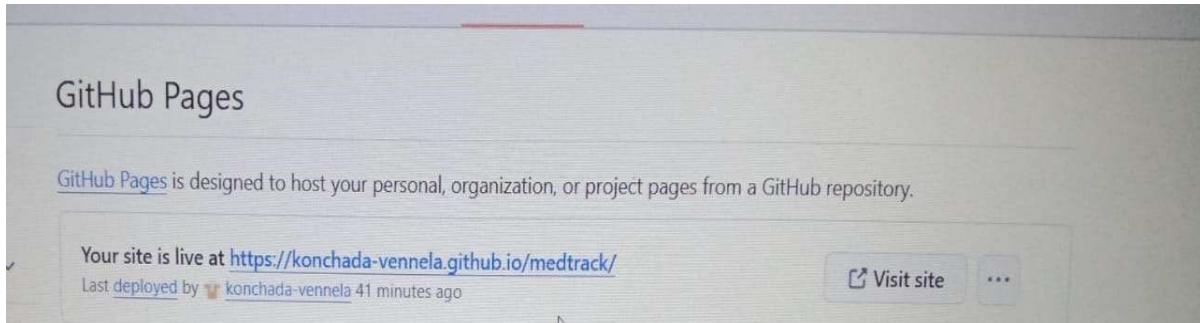
Policy name	Type	Attached entities
AmazonDynamoDBFullAccess	AWS managed	1
AmazonSNSFullAccess	AWS managed	2

Filter by Type: All types

Milestone 6: EC2 Instance Setup

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

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



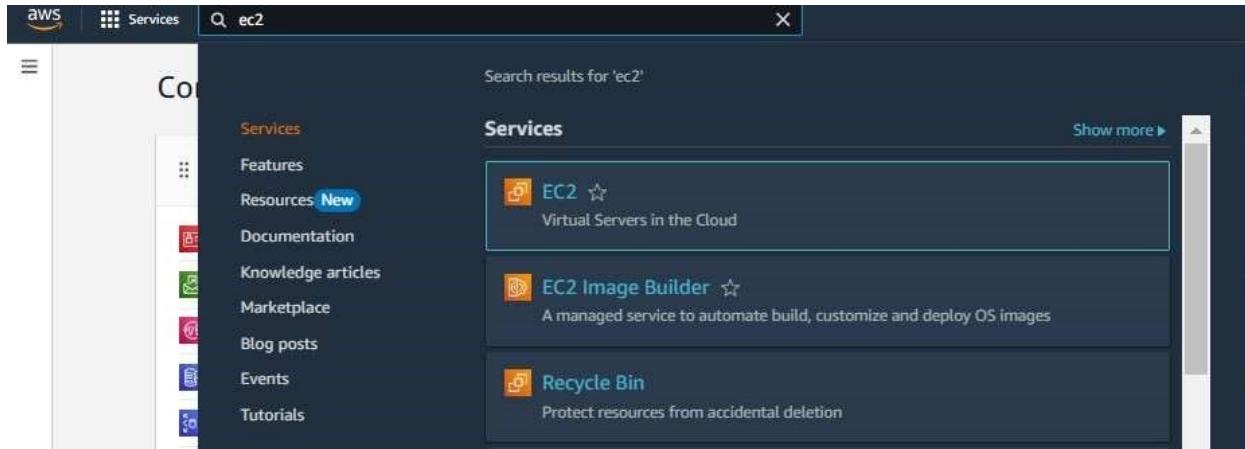
GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is live at <https://konchada-vennela.github.io/medtrack/>
 Last deployed by  konchada-vennela 41 minutes ago

[Visit site](#) [...](#)

- Activity 6.1: Launch an EC2 instance to host the Flask application.
- Launch EC2 Instance
 - In the AWS Console, navigate to EC2 and launch a new instance.



- Click on Launch instance to launch EC2 instance

The screenshot shows the AWS EC2 Instances page. The main area displays a message: "No instances" and "You do not have any instances in this region". Below this is a "Launch instances" button. On the left, there is a sidebar with navigation links: EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, and Savings Plans. The main content area has a breadcrumb trail: EC2 > Instances > Launch an instance. The "Launch an instance" step is currently active, showing fields for Name and tags (Name: InstantLibraryApp) and Summary (Number of instances: 1, Software Image (AMI): Amazon Linux 2023 AMI 2023.5.2, Virtual server type (instance type): t2.micro, Firewall (security group)).

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


[Browse more AMIs](#)

 Including AMIs from
 AWS, Marketplace and
 the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI

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

[Free tier eligible](#)

Description

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

Architecture

64-bit (x86)

Boot mode

uefi-preferred

AMI ID

ami-02b49a24cfb95941c

[Verified provider](#)

- Create and download the key pair for Server access.

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

Instance type

t2.micro

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

[Free tier eligible](#)
 All generations

[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) [Info](#)

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

Key pair name - required

Select

[Create new key pair](#)

Create key pair

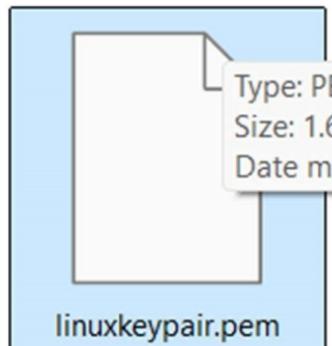
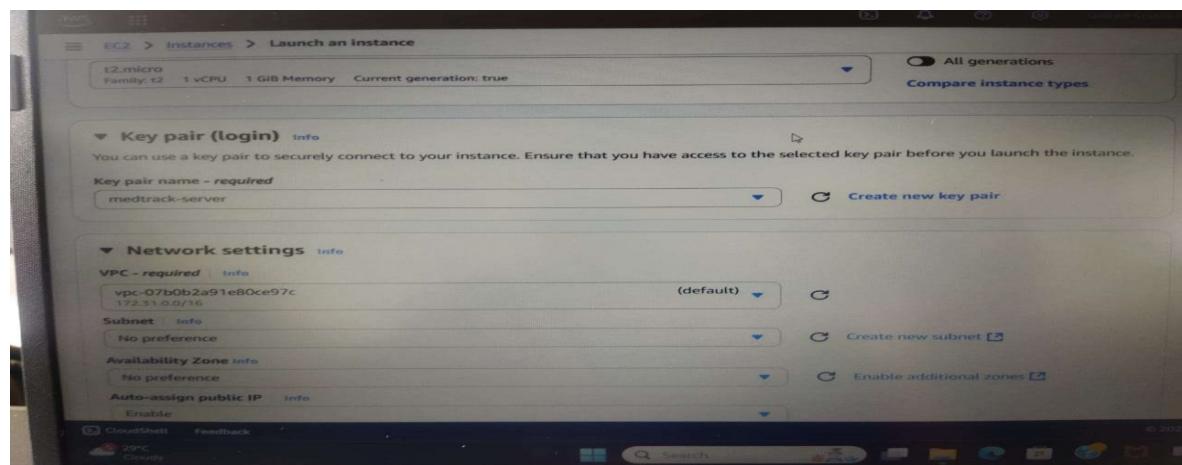
Key pair name
 Key pairs allow you to connect to your instance securely.
 InstantLibrary

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

Key pair type
 RSA RSA encrypted private and public key pair
 ED25519 ED25519 encrypted private and public key pair

Private key file format
 .pem For use with OpenSSH
 .ppk For use with PuTTY

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

[Cancel](#)
[Create key pair](#)



EC2 > Instances > Launch an Instance

I2.micro Family: I2 1 vCPU 1 GiB Memory Current generation: true

All generations Compare instance types

Key pair (login) [Info](#)
 You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required [Create new key pair](#)

Network settings [Info](#)

VPC - required [Info](#)
 vpc-07b0b2a91e80ce97c (default) [Change](#)

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

Availability Zone [Info](#)
 No preference [Enable additional zones](#)

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

CloudShell Feedback 29°C Cloudy

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

▼ Network settings [Info](#)

VPC - required [Info](#)
 vpc-03cdc7b6f19dd7211 (default) [G](#)
 172.31.0.0/16

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

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

Additional charges apply when outside of free tier allowance

Firewall (security groups) [Info](#)
 A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group Select existing security group

Security group name - required
 launch-wizard

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

Description - required [Info](#)
 launch-wizard created 2024-10-13T17:49:56.622Z

Inbound Security Group Rules

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

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

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

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

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

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

Add security group rule

EC2 > ... > Launch an instance

Success
Successfully initiated launch of instance i-001861022fbac290

▶ Launch log

Next Steps

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

< 1 2 3 4 >

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

[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 (1/2) Info

Last updated less than a minute ago

Find Instance by attribute or tag (case-sensitive)

All states ▾

Connect Instance state Actions Launch instances

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring	Security
InstantLibrary...	i-001861022fbac290	Stopped	t2.micro	-	View alarms	ap-south-1b	-	-	-	-	disabled	launch-wit



EC2 > Instances > i-001861022fbcac290

Instance summary for i-001861022fbcac290 (InstantLibraryApp) [Info](#)

Updated less than a minute ago

Instance ID i-001861022fbcac290	Public IPv4 address -	Private IPv4 addresses 172.31.3.5
IPv6 address -	Instance state Stopped	Public IPv4 DNS -
Hostname type IP name: ip-172-31-3-5.ap-south-1.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-3-5.ap-south-1.compute.internal	Elastic IP addresses -
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto-assigned IP address -	VPC ID vpc-03cdc7b6f19dd7211	Auto Scaling Group name -
IAM Role sns_Dynamodb_role	Subnet ID subnet-0d9fa3144480cc9a9	
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

Instance summary for i-001861022fbcac290 (InstantLibraryApp) [Info](#)

Updated less than a minute ago

Instance ID i-001861022fbcac290	Public IPv4 address -	Private IPv4 addresses 172.31.3.5
IPv6 address -	Instance state Stopped	Public IPv4 DNS -
Hostname type IP name: ip-172-31-3-5.ap-south-1.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-3-5.ap-south-1.compute.internal	Elastic IP addresses -
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto-assigned IP address -	VPC ID vpc-03cdc7b6f19dd7211	Auto Scaling Group name -
IAM Role sns_Dynamodb_role	Subnet ID subnet-0d9fa3144480cc9a9	
IMDSv2 Required	Instance ARN arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290	

[C](#) [Connect](#) [Instance state](#) [Actions](#)

[Connect](#) [Manage instance state](#) [Instance settings](#) [Networking](#) [Security](#) [Get Windows password](#) [Image and templates](#) [Monitor and troubleshoot](#)

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

Modify IAM role [Info](#)

Attach an IAM role to your instance.

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

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

[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.

Public IPv4 address
 13.200.229.59

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.

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

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

Cancel
Connect

```
A newer release of "Amazon Linux" is available.
Version 2023.6.20241010.
Run "/usr/bin/dnf check-release-update" for full release and version update info
#
#          Amazon Linux 2023
#
#          https://aws.amazon.com/linux/amazon-linux-2023
#
#          Last login: Tue Oct 15 04:17:59 2024 from 13.233.177.3
[ec2-user@ip-172-31-3-5 ~]$ 
```

i-001861022fbcac290 (InstantLibraryApp)
 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:

```
lask --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: '<https://github.com/hemasundar2005/medtrack>' Note: change your-githubusername and your-repository-name with your credentials here: 'git clone '<https://github.com/hemasundar2005/medtrack> '

- 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 lask 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
      #
      #
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]$ ^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>

ip

- Run the Flask app on the EC2 instance

```

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

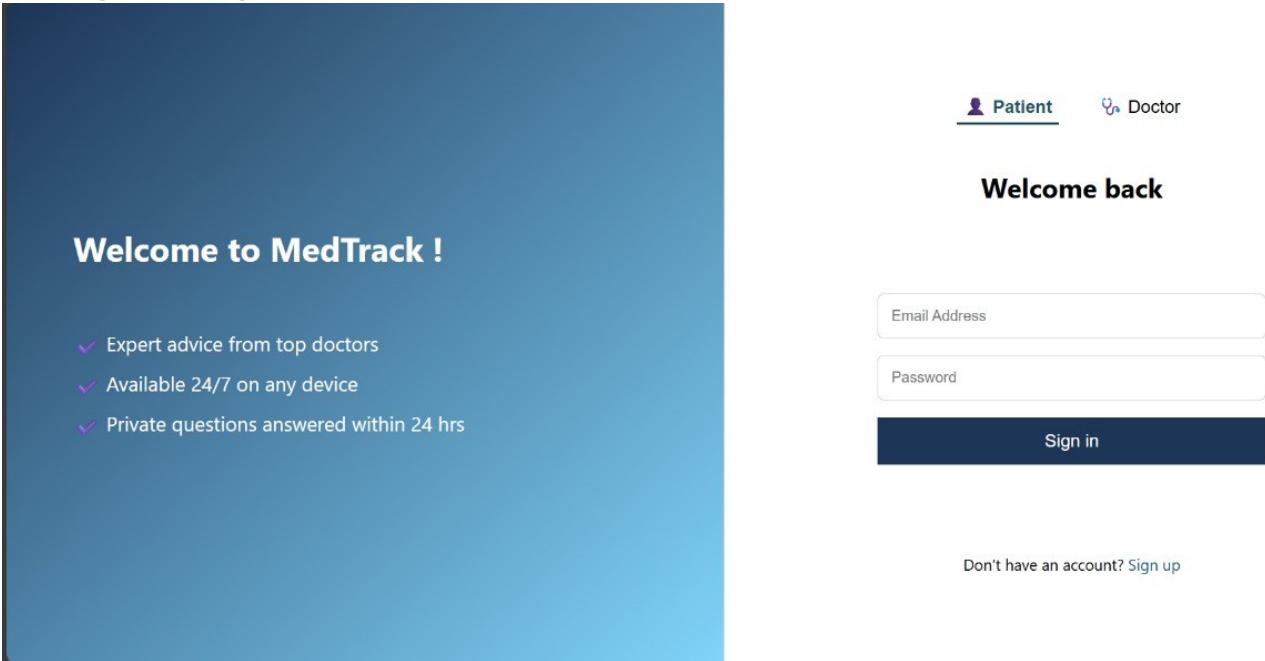
```

Access the website through:
Public IPs: [https://54.81.119.53 /](https://54.81.119.53/)

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 the MedTrack registration page. It features a blue header with the text "Welcome back" and navigation links for "Patient" and "Doctor". The main content area has a teal background with the heading "Welcome to MedTrack !". Below it, there is a list of three benefits with checkmarks: "Expert advice from top doctors", "Available 24/7 on any device", and "Private questions answered within 24 hrs". To the right, there are input fields for "Email Address" and "Password", and a "Sign in" button. At the bottom right, there is a link "Don't have an account? Sign up".

Welcome back

Patient Doctor

Welcome to MedTrack !

- ✓ Expert advice from top doctors
- ✓ Available 24/7 on any device
- ✓ Private questions answered within 24 hrs

Email Address

Password

Sign in

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

Login Page:

Sign Up

Patient

Full Name

Email

Password

Age

Gender

Register

Already have an account? [Login](#)

Patient Dashboard page:

Health Care

[Dashboard](#)

[My Appointments](#)

[Logout](#)

Welcome, Arasavilli dhanunjay

Find a Doctor

sateesh chandra
Dentist
sateeshchandra@gmail.com
 Book Appointment

vasu dev
ENT
v@gmail.com
 Book Appointment

Geetha Rani
Dermatologist
g@gmail.com
 Book Appointment

Book Appointment page:

Book Appointment with Dr. sateesh chandra

Symptoms

Description

Submit

Book Appointment with Dr. sateesh chandra

Tooth decay and loss of tooth

.....|

Submit

Health Care

- [Dashboard](#)
- [My Appointments](#)
- [Logout](#)

Welcome, Arasavili dhanunjay

My Appointments

Doctor	Specialization	Symptoms	Description	Booked On	Status	Doctor's Response	Responded Time
sateesh chandra		Tooth decay and loss of tooth		2025-07-09 19:10:39	Pending	—	—

Doctor login:

Sign Up

Doctor

Full Name

Email

Password

Specialization

Register

Already have an account? [Login](#)

Doctor Dashboard:

Doctor Panel

- [Dashboard](#)
- [Logout](#)

Welcome,
Dr. sateesh chandra

Appointments

Logged in as **Dr. sateesh chandra**

Patient	Symptoms	Description	Booked On	Status	Response	Responded Time
Reddy Hemasundara Rao	loss of tooth and tooth decay	bleeding in tooth,pain and feeling sensitivity	2025-07-17 14:08:08	Pending	<div style="border: 1px solid #ccc; padding: 5px; width: 150px; height: 40px; margin-bottom: 5px;">Write your prescription here...</div> <input style="background-color: #1a233a; color: white; border: 1px solid #ccc; padding: 2px 10px;" type="button" value="Submit"/>	—

Doctor Response:

Doctor Panel

- [Dashboard](#)
- [Logout](#)

Welcome,
Dr. sateesh chandra

Appointments

Logged in as **Dr. sateesh chandra**

Patient	Symptoms	Description	Booked On	Status	Response	Responded Time
Reddy Hemasundara Rao	loss of tooth and tooth decay	bleeding in tooth,pain and feeling sensitivity	2025-07-17 14:08:08	Replied	<div style="border: 1px solid #ccc; padding: 5px; background-color: #e0f2e0; width: 150px; height: 40px; margin-bottom: 5px;">use the medicine-clean your teeth atleast twice a day and use sensodyne tooth paste</div>	2025-07-17 14:11:12

Showing doctor response in patient dashboard:

Health Care
 Dashboard
[My Appointments](#)
[Logout](#)

My Appointments

Doctor	Specialization	Symptoms	Description	Booked On	Status	Doctor's Response	Responded Time
sateesh chandra		Tooth decay and loss of tooth	2025-07-09 19:10:39	Replied	There are steps that you can take to help prevent tooth decay. These include things like brushing your teeth at least twice a day, avoiding sweet foods, and making sure to visit your dentist regularly.	2025-07-09 19:14:51

Dynamodb Database Updations:

1. Users table and Appointments table :

Tables (2) Info						
	Name	Status	Partition key	Sort key	Indexes	Replication Regions
<input type="checkbox"/>	Appointments	<input checked="" type="radio"/> Active	appointment_id (\$)	-	0	0
<input type="checkbox"/>	users	<input checked="" type="radio"/> Active	email (\$)	-	0	0

Conclusion:

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

The cloud-native approach enables seamless scalability, allowing MedTrack to support increasing numbers of users and data without compromising performance or reliability. The integration of



Flask with AWS ensures smooth backend operations, including patient registration, medication reminders, and health updates. Thorough testing has validated that all features—from user onboarding to alert notifications—function reliably and securely.

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