

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. The system allows users to manage appointments, medical records, and diagnosis reports in one centralized platform, improving the overall healthcare experience for both patients and doctors. All patient and doctor data, including appointment details and medical histories, are stored in AWS DynamoDB.

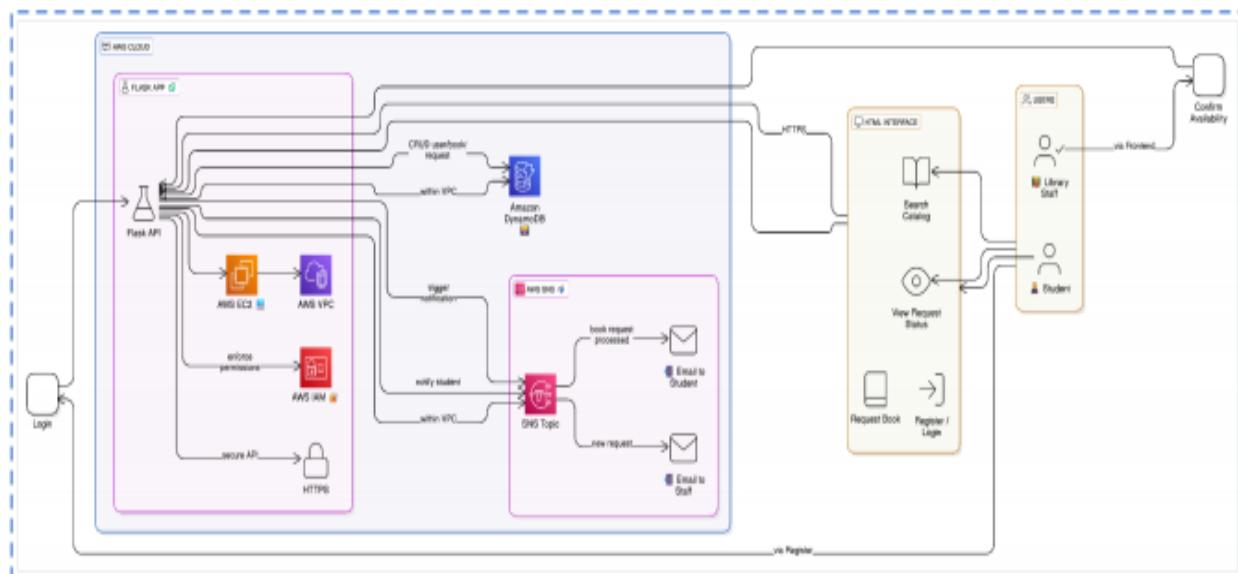
Scenario 2: Secure User Management with IAM

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.

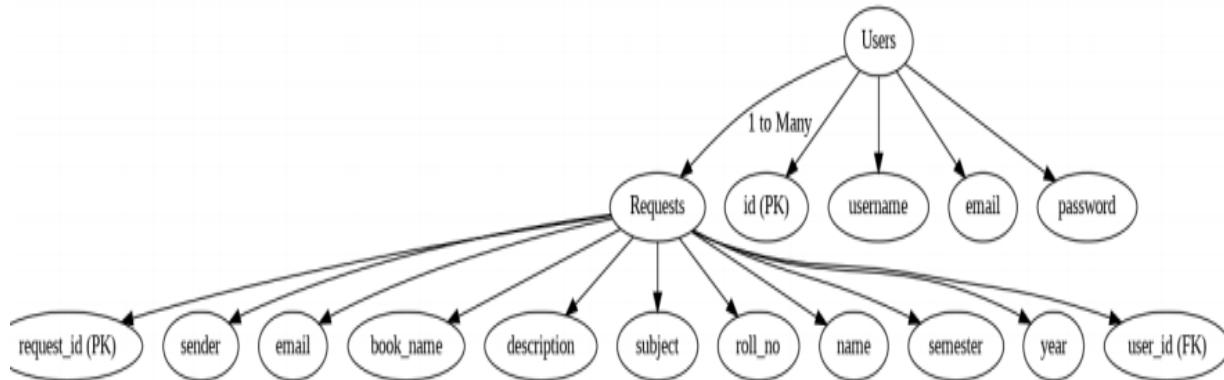
Scenario 3: Easy Access to Medical History and Resources

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



ER DIAGRAM



PROJECT WORK FLOW

1. AWS Account Setup and Login

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

Activity 1.2: Log into the AWS Management Console

2. DynamoDB Database Creation and Setup

Activity 2.1: Create a DynamoDB Table.

Activity 2.2: Configure Attributes for User Data and Book Requests.

3. SNS Notification Setup

Activity 3.1: Create SNS Topics for book request notifications.

Activity 3.2: Subscribe users and library staff to SNS email notifications.

4. Backend Development and Application Setup

Activity 4.1: Develop the Backend Using Flask.

Activity 4.2: Integrate AWS Services Using boto3.

5. IAM Role Setup

Activity 5.1: Create IAM Role

Activity 5.2: Attach Policies

6. EC2 Instance Setup

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

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

7.DeploymentonEC2

Activity7.1:UploadFlaskFiles

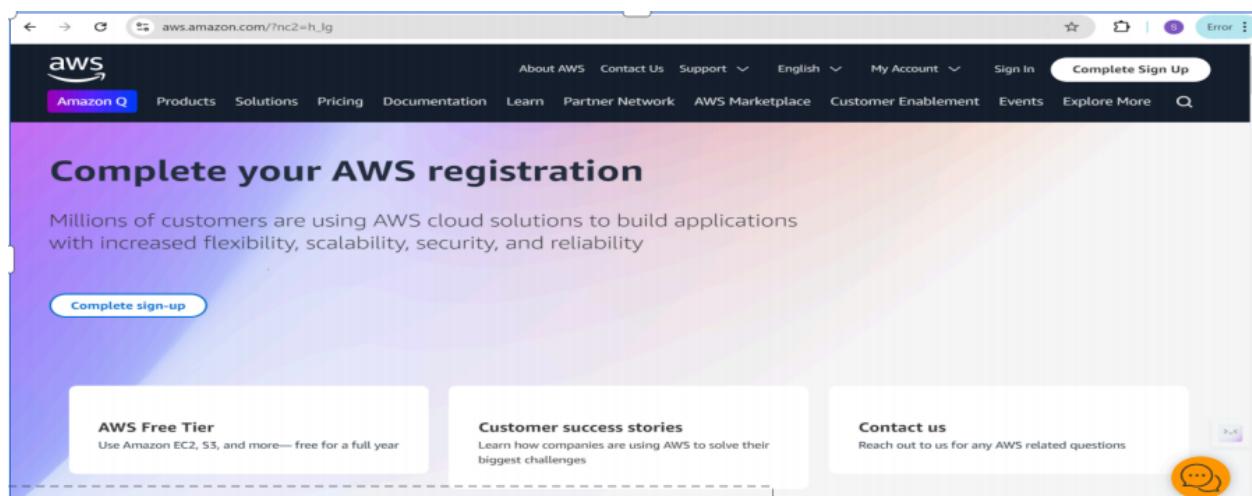
Activity7.2:RuntheFlaskApp

8. Testing and Deployment

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

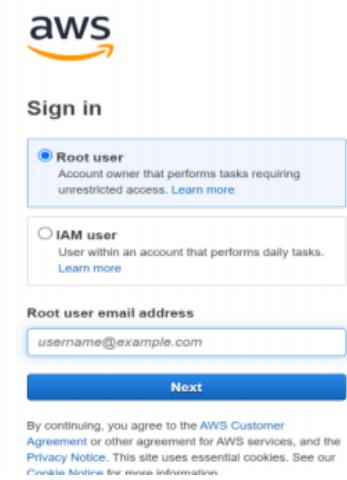
Milestone 1: AWS Account Setup and Login

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



Activity 1.2: Log in to the AWS Management Console

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



The image shows the AI Use Case Explorer landing page. The background is a gradient from dark purple to light blue. The title 'AI Use Case Explorer' is prominently displayed in white. Below the title, a large block of text reads: 'Discover AI use cases, customer success stories, and expert-curated implementation plans'. At the bottom of the main content area is a blue '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 image shows the AWS Services search results for 'dynamoDB'. The search bar at the top has 'dynamoDB' typed into it. Below the search bar is a 'Search results for "dyn"' placeholder. On the left, a sidebar lists various AWS services and features: Services, Features, Resources (New), Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main content area is titled 'Services' and shows a list of services: 'DynamoDB' (Managed NoSQL Database), 'Amazon DocumentDB' (Fully-managed MongoDB-compatible database service), 'CloudFront' (Global Content Delivery Network), and 'Athena' (Serverless interactive analytics service). Each service entry includes a star icon. Below this is another section titled 'Features' with 'Settings' and 'Clusters' subsections, each containing a single item: 'DynamoDB feature'. There are 'Show more >' links at the top and bottom of the service and feature sections.

DynamoDB Dashboard

Alarms (0) Info

Find alarms

Alarm name Status

No custom alarms

DAX clusters (0) Info

Find clusters

Cluster name Status

No clusters

No clusters to display

Create cluster

Create resources

Create an Amazon DynamoDB table for fast and predictable database performance at any scale. [Learn more](#)

Create table

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

Create DAX cluster

What's new

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

DynamoDB Tables

Tables (0) Info

Find tables Any tag key Any tag value

Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size
You have no tables in this account in this AWS Region.								

Create table

- **Activity 2.2: Create a DynamoDB table for storing registration details and book requests.**

- Create Users table with partition key “Email” with type String and click on create tables

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.

The screenshot shows the AWS DynamoDB console interface. The left sidebar has a 'Tables' section selected. The main area displays a table titled 'Tables (4)'. The table columns include Name, Status, Partition key, Sort key, Indexes, Replication Regions, and Deletion protection. All four tables listed are active and have 'Off' selected for deletion protection.

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion p
MedTrackAppointments	Active	appointment_id (\$)	-	0	0	Off
MedTrackDoctors	Active	email (\$)	-	0	0	Off
MedTrackPatients	Active	email (\$)	-	0	0	Off
MedTrackPrescriptions	Active	prescription_id (\$)	-	0	0	Off

Milestone 3: SNS Notification Setup

Activity 3.1

Create SNS topics for sending email notifications to users and

library staff.

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

Search results for '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

Show more ▶

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

Start with an overview

Pricing

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

The screenshot shows the Amazon SNS console. On the left, there's a sidebar with links to Dashboard, Topics, Subscriptions, Mobile (Push notifications, Text messaging (SMS)), and a New Feature banner about FIFO message archiving. The main area is titled 'Topics (0)' and has a search bar, sorting options (Name, Type, ARN), and a 'Create topic' button.

> Choose Standard type for general notification use cases and Click on Create

This screenshot shows the 'Create topic' wizard. In the 'Details' step, it compares two topic types: 'FIFO (first-in, first-out)' and 'Standard'. The 'Standard' type is selected. Other fields shown include 'Name' (BookRequestNotifications), 'Display name - optional' (My Topic), and several optional sections: Access policy, Data protection policy, Delivery policy (HTTP/S), Delivery status logging, Tags, and Active tracing. A 'Create topic' button is at the bottom right.

This screenshot continues the 'Create topic' wizard, focusing on optional settings. It includes sections for Access policy, Data protection policy, Delivery policy (HTTP/S), Delivery status logging, Tags, and Active tracing. Each section provides a brief description and a 'Learn more' link. At the bottom right is a 'Create topic' button.

> Configure the SNS topic and note down the **Topic ARN**

The screenshot shows the AWS SNS Topics page. On the left, there's a sidebar with links for Dashboard, Topics, Subscriptions, and Mobile (Push notifications, Text messaging (SMS)). The main area shows a blue banner at the top with a 'New Feature' message about High Throughput FIFO topics. Below it, a green banner says 'Topic project created successfully.' with the message 'You can create subscriptions and send messages to them from this topic.' There are 'Publish message' and 'X' buttons next to the green banner. The topic 'project' is listed with its details: Name (project), Display name (medtrack), ARN (arn:aws:sns:us-east-1:851725243544:project), Topic owner (851725243544), and Type (Standard). There are 'Edit', 'Delete', and 'Publish message' buttons for the 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

The screenshot shows the 'Create subscription' page for the 'project' topic. It has fields for 'Topic ARN' (arn:aws:sns:us-east-1:686255958327:project), 'Protocol' (Email selected), and 'Endpoint' (divyayerragalla@gmail.com). A note says 'After your subscription is created, you must confirm it.' Below this are sections for 'Subscription filter policy - optional' (with a note about filtering messages) and 'Redrive policy (dead-letter queue) - optional' (with a note about sending undeliverable messages to a dead-letter queue).

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



AWS Notification - Unsubscribe Confirmation Inbox



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

to me ▾

Your subscription to the topic below has been deactivated:

arn:aws:sns:us-east-1:686255958327:project

If this was in error or you wish to resubscribe, click or visit the link below:

[Resubscribe](#)

Please do not reply directly to this email. If you have any questions or comments regarding this email, please visit [AWS Support](#).

[Reply](#)

[Forward](#)



Simple Notification Service

Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:

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

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

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

The screenshot shows the AWS SNS console. In the top navigation bar, the path is: Amazon SNS > Topics > project > Subscription: 28a09dc5-264c-4e4c-ba73-62e13c8733fe. On the left sidebar, under 'Amazon SNS', the 'Subscriptions' option is selected. Under 'Mobile', there are links for 'Push notifications' and 'Text messaging (SMS)'. A blue banner at the top right says 'New Feature' with the text: 'Amazon SNS now supports High Throughput FIFO topics. Learn more' with a link icon. Below it, a green banner says 'Subscription to project created successfully.' with the ARN: arn:aws:sns:us-east-1:686255958327:project:28a09dc5-264c-4e4c-ba73-62e13c8733fe. The main content area is titled 'Subscription: 28a09dc5-264c-4e4c-ba73-62e13c8733fe'. It has a 'Details' section with the following information:

ARN
arn:aws:sns:us-east-1:686255958327:project:28a09dc5-264c-4e4c-ba73-62e13c8733fe

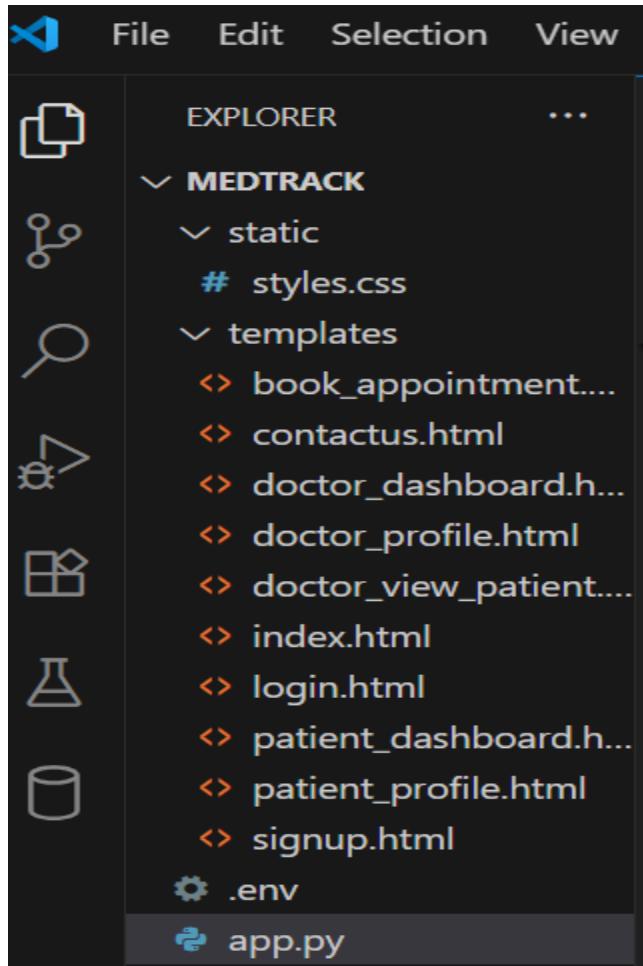
Endpoint
divyayerragalla@gmail.com

Topic
project

Subscription Principal
arn:aws:iam::686255958327:role/rsoaccount-new

Milestone 4:Backend Development and Application Setup

- **Activity 4.1: Develop the backend using Flask**
 - File Explorer Structure



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

Description of the code :

- Flask App Initialization

```
app.py ▾ 3 X .env # styles.css <> book_appointment.html <> contactus.html <> doctor_view_patient.html
app.py ...
1 1 ✓ from flask import Flask, render_template, request, redirect, url_for, sess
2 2 from werkzeug.security import generate_password_hash, check_password_hash
3 3 import boto3, uuid, os, logging
4 4 from dotenv import load_dotenv
```

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

```
# ----- AWS Setup -----
dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
sns = boto3.client('sns', region_name='us-east-1')
SNS_TOPIC_ARN = os.getenv("SNS_TOPIC_ARN")

# DynamoDB Tables
doctor_table = dynamodb.Table("MedTrackDoctors")
patient_table = dynamodb.Table("MedTrackPatients")
appointment_table = dynamodb.Table("MedTrackAppointments")
prescription_table = dynamodb.Table("MedTrackPrescriptions")
```

Description: initialize the DynamoDB resource for the ap-south-1 region and set up access to the Users and Requests tables for storing user details and book requests.

- **SNS Connection**

```

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

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

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

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

```

Description: Configure SNS to send notifications when a book request is submitted. Paste your stored ARN link in the sns_topic_arn space, along with the region_name where the SNS topic is created. Also, specify the chosen email service in SMTP_SERVER (e.g., Gmail, Yahoo, etc.) and enter the subscribed email in the SENDER_EMAIL section.

Routes for Web Pages

- **Home Route**

```

# ----- Routes -----
@app.route("/")
def index():
    return render_template("index.html")

@app.route("/signup", methods=["GET", "POST"])
def signup():
    if request.method == "POST":
        role = request.form.get("role")
        name = request.form.get("name")
        email = request.form.get("email")
        phone = request.form.get("phone")
        gender = request.form.get("gender")
        password = request.form.get("password")

        table = doctor_table if role == "doctor" else patient_table
        if table.get_item(Key={"email": email}).get("Item"):
            flash("Email already registered.")
            return redirect(url_for("signup"))

```

Description: define /register route to validate registration form fields, hash the user password using Bcrypt, store the new user in DynamoDB with a login count, and send an SNS notification on successful registration

Login Route (GET/POST):

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

        table = doctor_table if role == "doctor" else patient_table
        user = table.get_item(Key={"email": email}).get("Item")

        if not user or user["password"] != password:
            flash("Invalid credentials.")
            return redirect(url_for("login"))

        session.update({"name": user["name"], "email": user["email"], "role": role})
        return redirect(url_for("doctor_dashboard" if role == "doctor" else patient_dashboard"))
```

Doctor Dashboard route:

```
@app.route("/doctor_dashboard")
def doctor_dashboard():
    if session.get("role") != "doctor": return redirect(url_for("login"))

    name = session.get("name")
    write_mode = request.args.get("write_mode") == "yes"
    show_all = request.args.get("show_all") == "yes"

    upcoming, completed, patient_list = [], [], []
    total = 0

    items = appointment_table.scan().get("Items", [])
    for item in items:
        if item.get("doctor") != name: continue
        total += 1
        if item.get("prescription"):
```

Description: The doctor_dashboard route in app.py displays the doctor's main

dashboard after login. It fetches appointment data from DynamoDB and categorizes them as upcoming or completed. The route passes this data to doctor_dashboard.html, which shows a summary of appointments and patient interactions.

Patient dashboard Route

```
@app.route("/patient_dashboard", endpoint="patient_dash")
def patient_dashboard():
    if session.get("role") != "patient": return redirect(url_for("login"))
    name = session.get("name")

    show_all = request.args.get("show_all")
    prescription_success = request.args.get("prescription_success")
    (variable) appointments: Any
    appointments = appointment_table.scan()["Items"]
    prescriptions = prescription_table.scan()["Items"]

    upcoming = [a for a in appointments if a.get("patient") == name and a.
    completed = [a for a in appointments if a.get("patient") == name and a
    patient_prescriptions = [p for p in prescriptions if p.get("patient")]

    return render_template("patient_dashboard.html", name=name,
```

Description: The patient_dashboard route displays the logged-in patient's dashboard with upcoming and completed appointments. It retrieves appointment and prescription data from DynamoDB, filtering it based on the patient's name.

Doctor view Patients route:

```
@app.route("/doctor_view_patients")
def doctor_view_patients():
    if session.get("role") != "doctor": return redirect(url_for("login"))
        (variable) session: SessionMixin
    name = session.get("name")
    items = appointment_table.scan().get("Items", [])
    patients = [item for item in items if item.get("doctor") == name and it

    return render_template("doctor_view_patients.html", name=name, patients

@app.route("/submit_prescription", methods=[ "POST"])
def submit_prescription():
    if session.get("role") != "doctor": return redirect(url_for("login"))

    doctor = session.get("name")
    patient = request.form["patient"]
    prescription = request.form["prescription"]
```

Doctor Profile Route:

```
@app.route("/doctor_profile", methods=["GET", "POST"])
def doctor_profile():
    if session.get("role") != "doctor": return redirect(url_for("login"))

    email = session.get("email")

    if request.method == "POST":
        doctor_table.update_item(
            Key={"email": email},
            UpdateExpression="SET #n = :name, phone = :phone, gender = :gender",
            ExpressionAttributeNames={"#n": "name"},
            ExpressionAttributeValues={
                ":name": request.form["name"],
                ":phone": request.form["phone"],
                ":gender": request.form["gender"],
                ":pwd": generate_password_hash(request.form["password"])
            })
        session["name"] = request.form["name"]
```

Patient Profile Route:

```
@app.route("/patient_profile", methods=["GET", "POST"])
def patient_profile():
    if session.get("role") != "patient": return redirect(url_for("login"))

    email = session.get("email")
    if request.method == "POST":
        patient_table.put_item(Item={
            "email": email,
            "name": request.form["name"],
            "phone": request.form["phone"],
            "gender": request.form["gender"],
            "password": generate_password_hash(request.form["password"]),
            "role": "patient"
        })
        flash("Profile updated.")
        return redirect(url_for("patient_profile"))
```

Book Appointment Route:

```
@app.route("/book_appointment", methods=[ "GET", "POST"])
def book_appointment():
    if session.get("role") != "patient": return redirect(url_for("login"))

    name = session.get("name")
    if request.method == "POST":
        appointment = {
            "id": str(uuid.uuid4()),
            "patient": name,
            "doctor": request.form["doctor"],
            "date": request.form["date"],
            "time": request.form["time"],
            "problem": request.form["problem"],
            "status": "accepted",
            "prescription": ""
        }
```

Contact Route:

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

Deployment code

```
if __name__ == "__main__":
    app.run(debug=True)
```

Milestone 5: IAM Role Setup

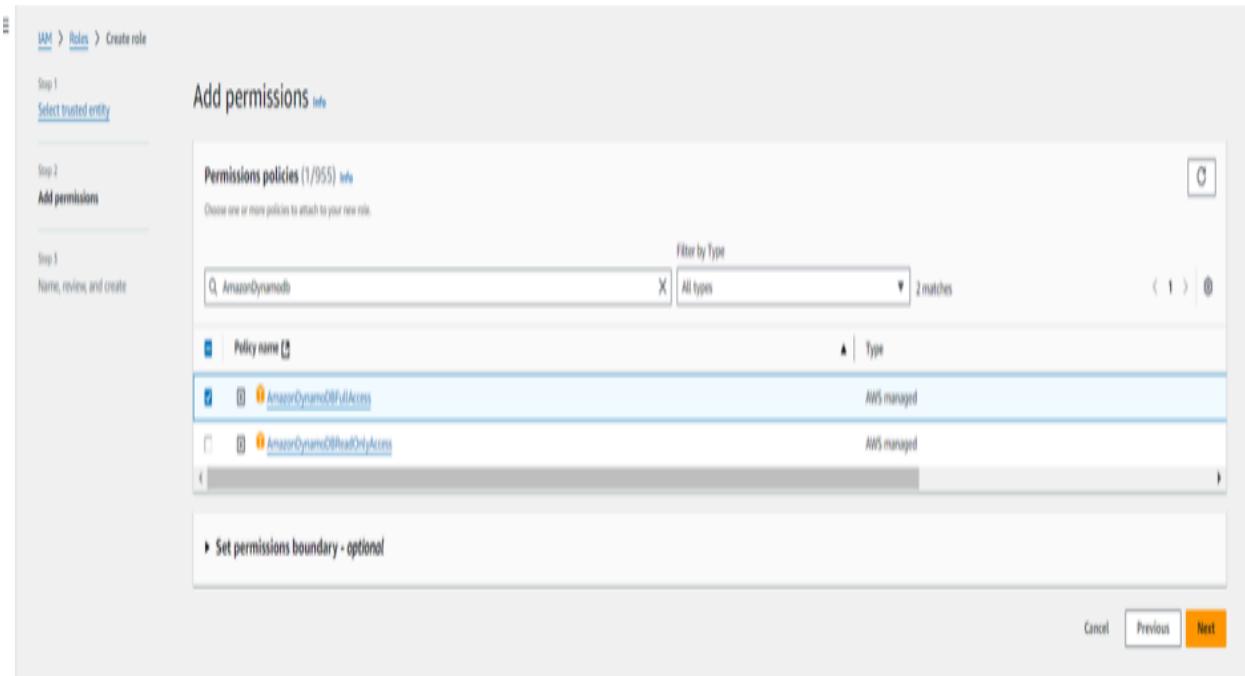
- Activity 5.1: Create IAM Role.

- 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 search interface with the query 'iam' entered in the search bar. The search results page displays several services under the 'Services' category. The 'IAM' service card is highlighted, showing its icon (a red square with a white key), name, and description: 'Manage access to AWS resources'. Other visible cards include 'IAM Identity Center', 'Resource Access Manager', and 'AWS App Mesh'.

The screenshot shows the 'Roles (6) info' section of the IAM service. It lists six roles with their names and last activity. A search bar and filters for 'Role name' and 'Trusted entities' are present. Action buttons for 'Create role' and 'Delete' are located at the top right.

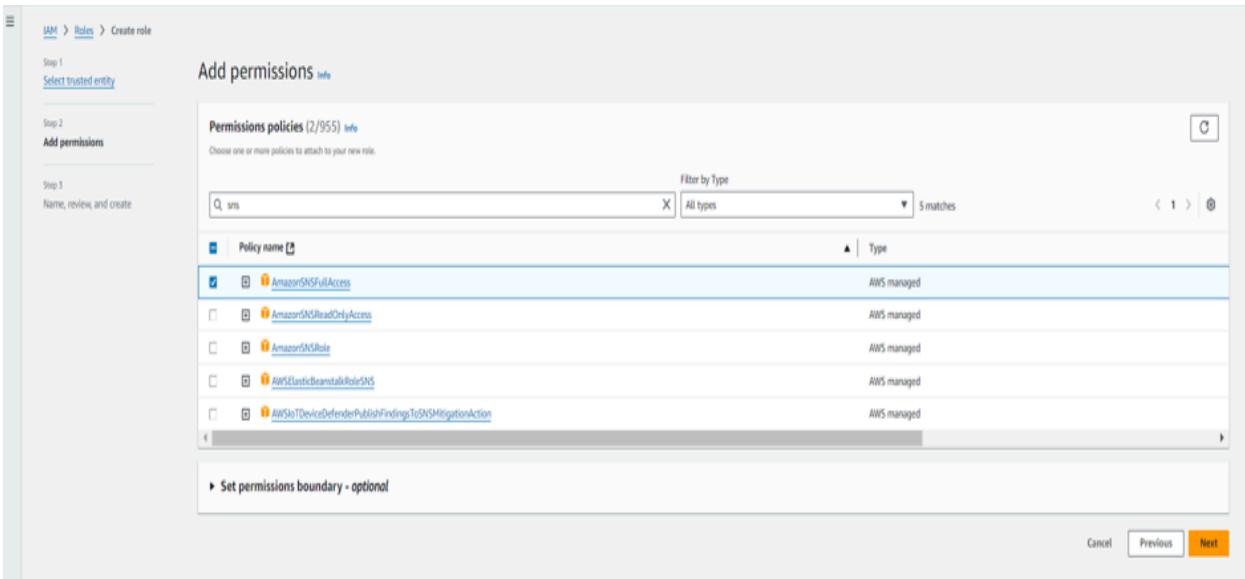
The screenshot shows the first step of the 'Create role' wizard, titled 'Select trusted entity'. It lists four options: 'AWS service' (selected), 'AWS account', 'SAML 2.0 Federation', and 'Web identity'. Below this, the 'Use case' section is shown, with 'EC2' selected from a dropdown. A detailed list of EC2 use cases follows, including 'EC2', 'EC2 - Spot Instances', 'EC2 - Spot Fleet', 'EC2 - Spot Fleet Auto Scaling', 'EC2 - Spot Fleet Tagging', 'EC2 - Spot Instances', 'EC2 - Spot Fleet', and 'EC2 - Scheduled Instances'. A 'Next Step' button is at the bottom right.



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.



Screenshot of the AWS IAM Roles page.

Identity and Access Management (IAM)

Roles (14) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
AWSServiceRoleForAccessAnalyzer	AWS Service: access-analyzer (Service-Linked Role)	373 days
AWSServiceRoleForAPIGateway	AWS Service: ops.apigateway (Service-Linked Role)	-
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable	AWS Service: dynamodb.application-autoscaling (Service-Linked Role)	373 days
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Linked Role)	400 days
AWSServiceRoleForCloudWatchEvents	AWS Service: events (Service-Linked Role)	390 days
AWSServiceRoleForECS	AWS Service: ecs (Service-Linked Role)	176 days ago
AWSServiceRoleForElasticLoadBalancing	AWS Service: elasticloadbalancing (Service-Linked Role)	400 days

Create role

Search IAM

Dashboard

Access management

- User groups
- Users
- Roles**
- Policies
- Identity providers
- Account settings
- Root access management New

Access reports

Screenshot of the AWS IAM Roles page showing a newly created role.

Identity and Access Management (IAM)

Roles (1/15) Info

Role EC2_MedTrack_Role created.

Create role

Search IAM

Dashboard

Access management

- User groups
- Users
- Roles**
- Policies
- Identity providers
- Account settings
- Root access management New

Access reports

Roles Anywhere Info

Authenticate your non AWS workloads and securely provide access to AWS services.

Manage

Access AWS from your non AWS

X.509 Standard

Temporary credentials

Milestone 6: EC2 Instance Setup

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

The screenshot shows two views of a GitHub repository named "medtrackproject". The top view displays the repository's main page with a list of files and their commit history. The bottom view shows the same repository with a "Clone" dialog box overlaid. The dialog box has tabs for "Local" and "Codespaces", with "Local" selected. It shows the HTTPS URL <https://github.com/Divya-yerragalla/medtrackpr>. Below the URL are options to "Clone using the web URL", "Open with GitHub Desktop", and "Download ZIP".

medtrackproject Public

main 1 Branch 0 Tags

Go to file Add file Code

Divya-yerragalla Update main.py 4a63002 · 3 days ago 5 Commits

static/images Add files via upload 3 days ago

templates Add files via upload 3 days ago

.env Update .env 3 days ago

main.py Update main.py 3 days ago

medtrackproject Public

main 1 Branch 0 Tags

Go to file Add file Code

Divya-yerragalla Update main.py

static/images Add files via upload

templates Add files via upload

.env Update .env

main.py Update main.py

README

Local Codespaces

Clone

HTTPS SSH GitHub CLI

<https://github.com/Divya-yerragalla/medtrackpr>

Clone using the web URL.

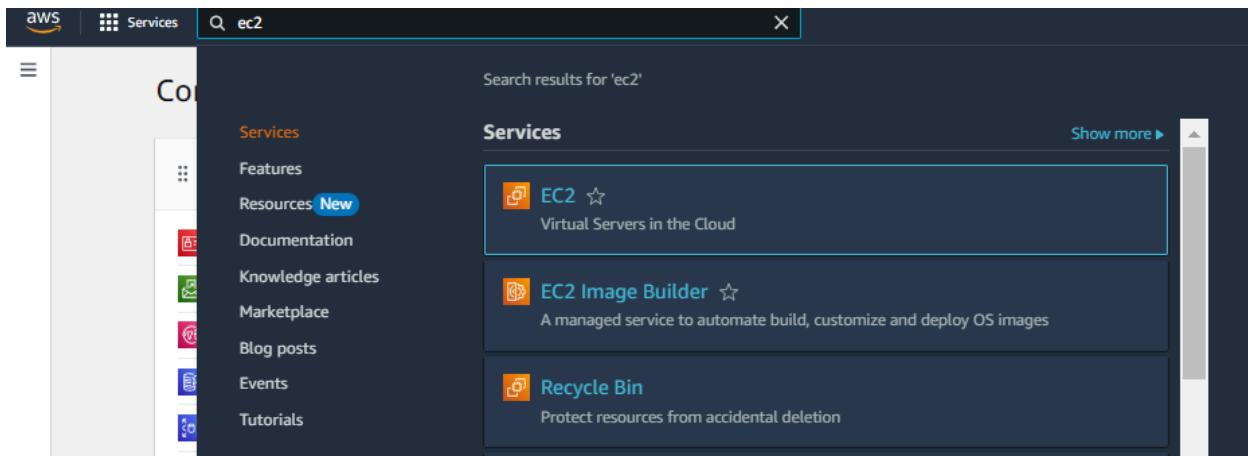
Open with GitHub Desktop

Download ZIP

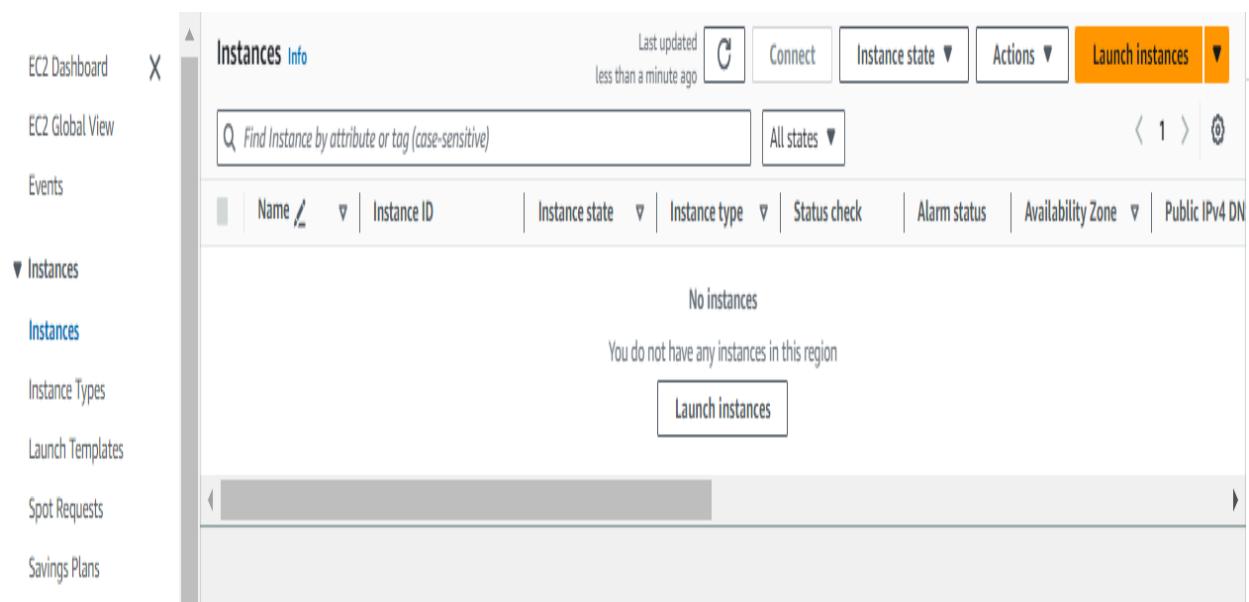
Activity 6.1: Launch an EC2 instance to host the Flask application.

- Launch EC2 Instance

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



>Click on Launch instance to launch EC2 instance



Screenshot of the AWS EC2 Launch an instance page:

Name and tags: Name: medtrack-server, Add additional tags

Application and OS Images (Amazon Machine Image): Search bar: Search our full catalog including 1000s of application and OS images, Quick Start button.

Summary: Number of instances: 1, Software Image (AMI): Amazon Linux 2 Kernel 5.10 AMI... (read more), ami-000ec6c25978d5999, Virtual server type (instance type): t2.micro, Firewall (security group): New security group, Launch instance button.

Amazon Machine Image (AMI)

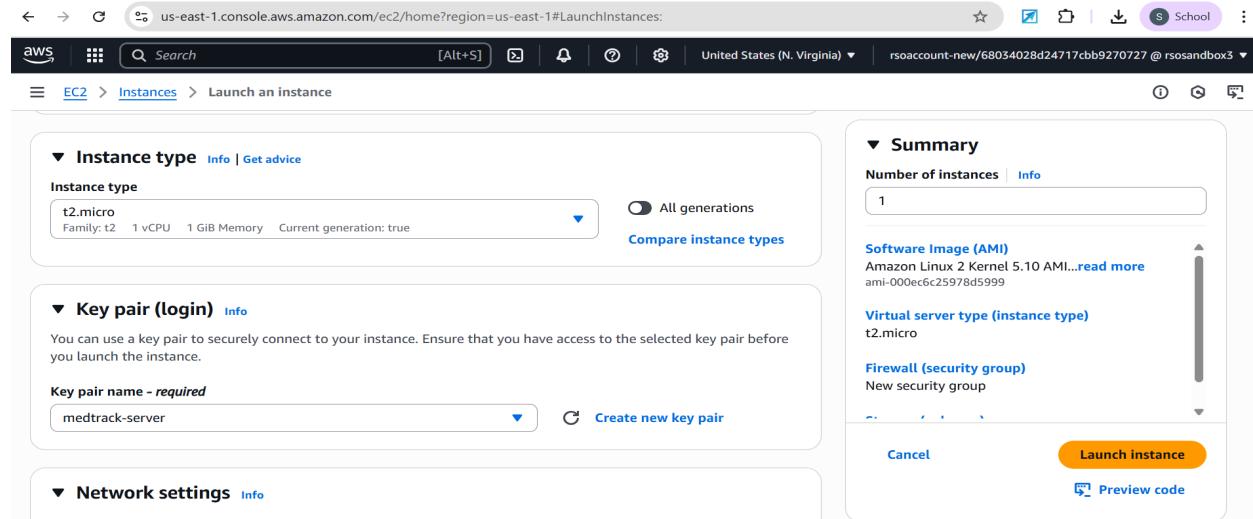
Amazon Linux 2023 AMI: Free tier eligible, ami-02b49a24cfb95941c (64-bit (x86), uefi-preferred) / ami-04ad8c7fcc828fad4 (64-bit (Arm), uefi), Virtualization: hvm, ENA enabled: true, Root device type: ebs.

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

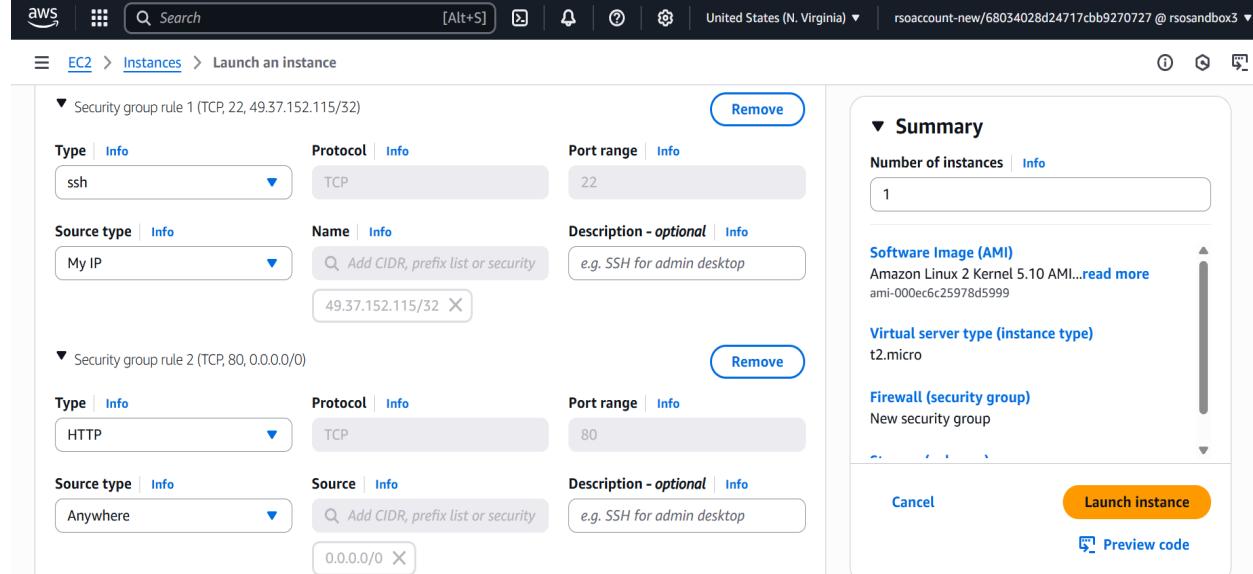
Architecture	Boot mode	AMI ID	Verified provider
64-bit (x86)	uefi-preferred	ami-02b49a24cfb95941c	Verified provider

Browse more AMIs: Including AMIs from AWS, Marketplace and the Community.

>Create and Download the keypair for server access:



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



The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with 'EC2' selected. Under 'Instances', it lists 'Instances', 'Instance Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances', 'Dedicated Hosts', and 'Capacity Reservations'. Under 'Images', it lists 'AMIs'. The main area shows 'Instances (1/1) info' with a table header: 'Name' (with a search icon), 'Instance ID', 'Instance state', 'Instance type', 'Status check', and 'Alarm status'. A single row is selected: 'medtrack-server' (i-01e8a55830cb0f801), which is 'Running' (t2.micro). Below this, the instance details for 'i-01e8a55830cb0f801 (medtrack-server)' are shown, including tabs for 'Details', 'Status and alarms', 'Monitoring', 'Security', 'Networking', 'Storage', and 'Tags'. Under 'Details', it shows 'Instance ID' (i-01e8a55830cb0f801), 'Public IPv4 address' (98.80.11.234), and 'Private IPv4 addresses' (172.31.24.250).

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

The screenshot shows the 'Instance summary for i-001861022fbcac290 (InstantLibraryApp) Info' page. The left sidebar lists instance details: Instance ID (i-001861022fbcac290), IPv6 address (-), Hostname type (IP name: ip-172-31-3-5.ap-south-1.compute.internal), Answer private resource DNS name (IPv4 (A)), Auto-assigned IP address (-), IAM Role (sns_Dynamodb_role), and IMDSv2 (Required). The main area shows instance details: Public IPv4 address (-), Instance state (Stopped), Private IP DNS name (IPv4 only) (ip-172-31-3-5.ap-south-1.compute.internal), Instance type (t2.micro), VPC ID (vpc-03cdc7b6f19dd7211), Subnet ID (subnet-0d9fa3144480cc9a9), and Instance ARN (arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290). On the right, the 'Actions' menu is open, showing options like 'Manage instance state', 'Instance settings', 'Networking', 'Security' (highlighted in blue), 'Image and templates', and 'Monitor and troubleshoot'. The 'Security' section includes 'Change security groups', 'Get Windows password', and 'Modify IAM role'.

The screenshot shows the AWS IAM Roles page. On the left, there's a sidebar with 'Identity and Access Management (IAM)' and a search bar. The main area has a title 'Roles (14) Info' and a description: 'An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.' There's a search bar at the top right of the table. The table has columns for 'Role name', 'Trusted entities', and 'Last activity'. The last activity column includes a red 'X' icon and a timestamp. A 'Create role' button is at the top right of the table.

Role name	Trusted entities	Last activity
AWSServiceRoleForAccessAnalyzer	AWS Service: access-analyzer (Service-Linked)	X 373 days
AWSServiceRoleForAPIGateway	AWS Service: ops.apigateway (Service-Linked)	-
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable	AWS Service: dynamodb.application	X 373 days
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Linked)	X 400 days
AWSServiceRoleForCloudWatchEvents	AWS Service: events (Service-Linked)	X 390 days
AWSServiceRoleForECS	AWS Service: ecs (Service-Linked Role)	176 days ago
AWSServiceRoleForElasticLoadBalancing	AWS Service: elasticloadbalancing (Service-Linked Role)	X 400 days

Now connect the EC2 with the files

The screenshot shows the 'Connect to instance' dialog for EC2 instance [i-001861022fbcac290 \(InstantLibraryApp\)](#). The top navigation tabs are 'EC2 Instance Connect', 'Session Manager', 'SSH client', and 'EC2 serial console'. The 'EC2 Instance Connect' tab is selected. A warning message box says: '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**: -

Username: 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.

Buttons at the bottom: 'Cancel' and 'Connect'.

The screenshot shows a terminal session in an AWS CloudShell window. The terminal output is as follows:

```

Amazon Linux 2
AL2 End of Life is 2026-06-30.
A newer version of Amazon Linux is available!
Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/
[ec2-user@ip-172-31-29-63 ~]$ sudo su
[root@ip-172-31-29-63 ec2-user]# sudo su
[root@ip-172-31-29-63 ec2-user]# yum install python3
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
Package python3-3.7.16-1.amzn2.0.17.x86_64 already installed and latest version
Nothing to do
[root@ip-172-31-29-63 ec2-user]# yum install git
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
-->> Package git.x86_64 0:2.47.1-1.amzn2.0.3 will be installed
-->> Processing Dependency: git-core = 2.47.1-1.amzn2.0.3 for package: git-2.47.1-1.amzn2.0.3.x86_64
-->> Processing Dependency: git-core-doc = 2.47.1-1.amzn2.0.3 for package: git-2.47.1-1.amzn2.0.3.x86_64
-->> Processing Dependency: perl-Git = 2.47.1-1.amzn2.0.3 for package: git-2.47.1-1.amzn2.0.3.x86_64
-->> Processing Dependency: perl(Git) for package: git-2.47.1-1.amzn2.0.3.x86_64
-->> Processing Dependency: perl(Term::ReadKey) for package: git-2.47.1-1.amzn2.0.3.x86_64

```

i-0fe9981734e6d7266 (medtrack-server)
PublicIPs: 54.224.221.69 PrivateIPs: 172.31.29.63

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/Divya-yerragalla/medtrackproject.git>

This downloaded my project to the EC2 instance.

To navigate to the project directory, run the following command:

```
cd InstantLibrary
```

```
[DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 in January 2021. More details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support
WARNING: Running pip install with root privileges is generally not a good idea. Try 'pip install --user' instead.
Requirement already satisfied: werkzeug in /usr/lib/python2.7/site-packages (1.0.1)
[root@ip-172-31-29-63 ec2-user]# pip install python-dotenv
[DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 in January 2021. More details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support
WARNING: Running pip install with root privileges is generally not a good idea. Try 'pip install --user' instead.
Collecting python_dotenv==0.18.0-py2.py3-none-any.whl (18 kB)
  Downloading typing-3.10.0.0-py2-none-any.whl (26 kB)
Installing collected packages: typing, python-dotenv
Successfully installed python-dotenv==0.18.0 typing-3.10.0.0
[root@ip-172-31-29-63 ec2-user]# git clone https://github.com/Divya-yerragalla/medtrackproject.git
Cloning into 'medtrackproject'...
remote: Enumerating objects: 28, done.
remote: Counting objects: 100% (28/28), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 28 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (28/28), 4.48 MiB | 29.61 MiB/s, done.
Resolving deltas: 100% (6/6), done.
[root@ip-172-31-29-63 ec2-user]# ls medtrackproject
[root@ip-172-31-29-63 medtrackproject]# ls
main.py static templates
[root@ip-172-31-29-63 medtrackproject]# ]
```

i-Ofef9981734e6d7266 (medtrack-server)
PublicIPs: 54.224.221.69 PrivateIPs: 172.31.29.63

Verify the Flask app is running

```
[root@ip-172-31-29-63 ec2-user]# pip install -r req.txt
[DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 in January 2021. More details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support
WARNING: Could not find a version that satisfies the requirement Flask<2.1.3 (from -r req.txt (line 1)) (from versions: 0.1, 0.2, 0.3, 0.3.1, 0.4, 0.5, 0.5.1, 0.5.2, 0.6, 0.6.1, 0.7, 0.7.1, 0.8, 0.8.1, 0.9, 0.10, 0.10.1, 0.11, 0.11.1, 0.12, 0.12.1, 0.12.2, 0.12.3, 0.12.4, 0.12.5, 1.0, 1.0.1, 1.0.2, 1.0.3, 1.0.4, 1.1.0, 1.1.1, 1.1.2, 1.1.3, 1.1.4)
[DEPRECATION: No matching distribution found for Flask<2.1.3 (from -r req.txt (line 1))
[root@ip-172-31-29-63 ec2-user]# pip install flask
[DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 in January 2021. More details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support
WARNING: Running pip install with root privileges is generally not a good idea. Try 'pip install --user' instead.
Collecting flask
  Downloading Flask-1.1.4-py2.py3-none-any.whl (94 kB)
Collecting click<8.0,>=5.1
  Downloading click-7.1.2-py2.py3-none-any.whl (82 kB) 82 kB 1.7 MB/s
Collecting Jinja<3.0,>=2.10.1
  Downloading Jinja2-2.11.3-py2.py3-none-any.whl (125 kB) 125 kB 61.8 MB/s
Collecting Werkzeug<2.0,>=0.15
  Downloading Werkzeug-1.0.1-py2.py3-none-any.whl (298 kB) 298 kB 57.2 MB/s
Collecting itsdangerous<2.0,>=0.24
  Downloading itsdangerous-1.1.0-py2.py3-none-any.whl (16 kB)
Collecting MarkupSafe<0.23
  Downloading MarkupSafe-1.1.1-cp27-cp27mu-manylinux1_x86_64.whl (24 kB)
```

i-Ofef9981734e6d7266 (medtrack-server)
PublicIPs: 54.224.221.69 PrivateIPs: 172.31.29.63

```

Installing collected packages: python-dotenv
  WARNING: The script dotenv is installed in '/usr/local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed python-dotenv-0.21.1
[root@ip-172-31-29-63 medtrackproject]# git clone https://github.com/Divya-yerragalla/medtrackproject.git
fatal: destination path 'medtrackproject' already exists and is not an empty directory.
[root@ip-172-31-29-63 medtrackproject]# ls
main.py  medtrackproject static templates
[root@ip-172-31-29-63 medtrackproject]# python3 main.py
/usr/local/lib/python3.7/site-packages/boto3/compat.py:82: PythonDeprecationWarning: Boto3 will no longer support Python 3.7 starting December 13, 2023. To continue receiving service updates, bug fixes, and security updates please upgrade to Python 3.8 or later. More information can be found here: https://aws.amazon.com/blogs/developer/python-support-policy-updates-for-aws-sdks-and-tools/
  warnings.warn(warning, PythonDeprecationWarning)
INFO:botocore.credentials:Found credentials in environment variables.
* Serving Flask app 'main'
* Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug:  Restarting with stat
/usr/local/lib/python3.7/site-packages/boto3/compat.py:82: PythonDeprecationWarning: Boto3 will no longer support Python 3.7 starting December 13, 2023. To continue receiving service updates, bug fixes, and security updates please upgrade to Python 3.8 or later. More information can be found here: https://aws.amazon.com/blogs/developer/python-support-policy-updates-for-aws-sdks-and-tools/
  warnings.warn(warning, PythonDeprecationWarning)
INFO:botocore.credentials:Found credentials in environment variables.
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 106-528-324
INFO:werkzeug: https://github.com/Divya-yerragalla/medtrackproject.git[]

```

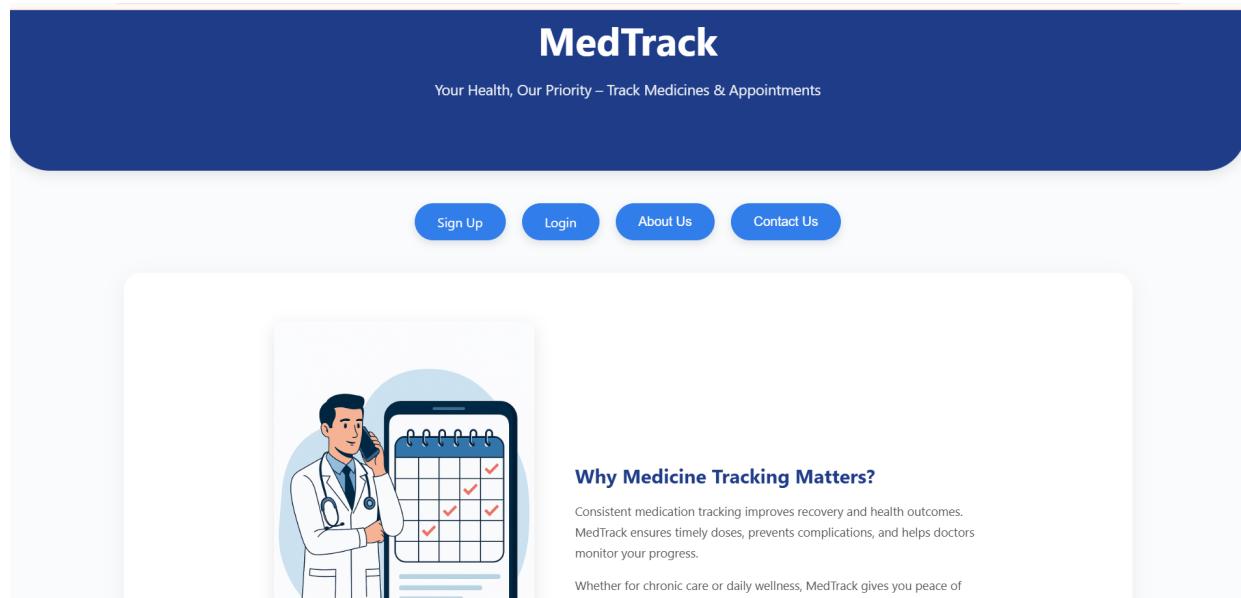
i-0fe9981734ed67266 (medtrack-server)

PublicIPs: 54.224.221.69 PrivateIPs: 172.31.29.63

Access the website through: <http://54.224.221.69:5000>

Milestone 8: Testing and Deployment

- **Activity 8.1: Conduct functional testing to verify user index, login,signup,doctor dashboard, patient dashboard,book appointment , prescription,about us, contact us pages.**



Register



Sign Up
Get started with your
MedTrack account

MedTrack

Role

Full Name
Yerragalla Divya

Email
divyayerragalla@gmail.com

Phone Number
9873216541

Gender
 Male Female

Password
.....

Already have an account? [Login here](#)

Login



Login
Welcome back
to MedTrack

MedTrack

Role

Email
tejaswi@gmail.com

Password
.....

[Create Account?](#)

[Dashboard](#) [Profile](#) [Logout](#)

Welcome, Dr. Yerragalla Divya

Manage your appointments with MedTrack

Doctor Dashboard

0

Pending Appointments

0

Completed Appointments

0

Total Appointments

Patients

[View Details](#)

Write Prescriptions

[Click to prescribe](#)

Doctor Profile Update

Full Name:
Yerragalla Divya

Email:
divyayerragalla@gmail.com

Phone:
9873216541

Gender:
Female

Password:

[Update Profile](#) [← Return to Dashboard](#)

Welcome, Tejaswi
Track your Appointments and Prescriptions with MedTrack

Patient Dashboard

[Book Appointment](#)

0 Upcoming Appointments

0 Completed Consultations

0 Prescriptions Available

Upcoming Appointments
No upcoming appointments.

Completed Consultations
No completed consultations.

Appointment with Dr. Yerragalla Divya on 2025-07-07 at 19:26 Confirmed!

Available Doctors for Appointment

Book Appointment with Dr. Yerragalla Divya ▾

Date:

07-07-2025



Time:

19:27



Your Problem:

fever

Confirm Appointment

← Back to Dashboard

Patient Appointment Details

Name	Gender	Date	Time	Problem	Status	Action
Tejaswi	female	2025-07-07	19:26	fever	Pending	<button>Accept</button> <button>Reject</button>

← Back to Dashboard

Patients

[View Details](#)

Write Prescriptions

[Click to prescribe](#)

[Book Appointment](#)

0
Upcoming Appointments

1
Completed Consultations

1
Prescriptions Available

Upcoming Appointments

No upcoming appointments.

Completed Consultations

Doctor Name	Date	Time
Yerragalla Divya	2025-07-07	19:26

Prescriptions

Doctor Name	Prescription
Yerragalla Divya	Take Dolo 650 tablet after lunch and dinner.

Patient Profile Update

Full Name:

Email:

Phone:

Gender:

Password:

[Update Profile](#) [Return to Dashboard](#)