

Med Track: AWS Cloud-Enabled Healthcare Management System

Project Description:

MedTrack is a secure, cloud-based healthcare management platform designed to streamline patient care and administrative workflows. Built on Amazon Web Services (AWS), it allows healthcare providers to manage electronic health records, schedule appointments, conduct telemedicine consultations, and issue e-prescriptions from a single system.

The platform uses **AWS EC2** for hosting applications, **Amazon RDS** for reliable database storage, **S3** for storing medical documents, and **Cognito** for secure user authentication and access control. Automated notifications and reminders are sent via **AWS SNS** to keep patients informed.

By leveraging AWS cloud infrastructure, MedTrack delivers high availability, scalability, and robust data protection, helping clinics and hospitals improve efficiency while maintaining compliance with healthcare data regulations like HIPAA.

Scenario 1 – Appointment Booking and Notifications

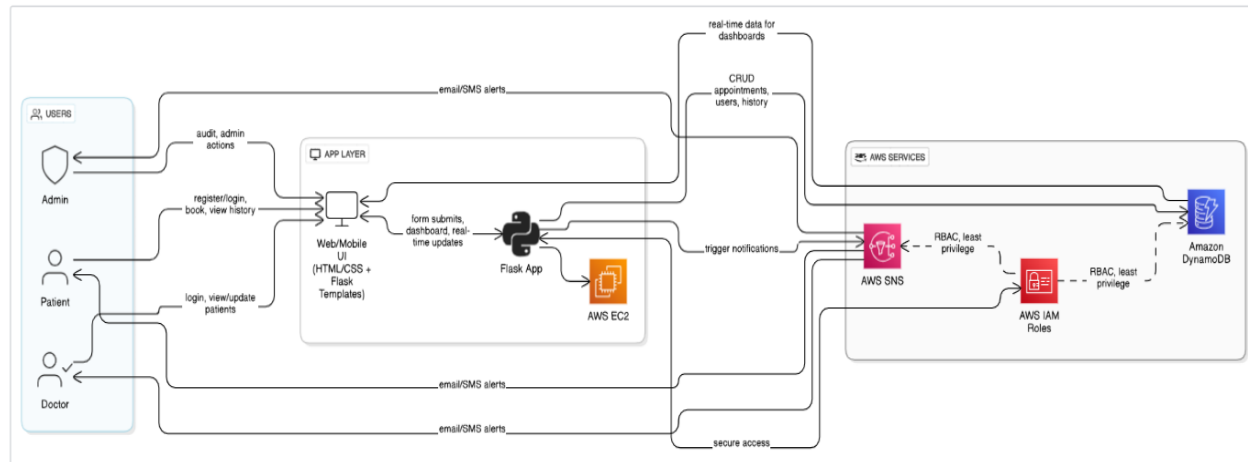
A patient logs into the MedTrack web app, which is hosted on **Amazon EC2** instances. They schedule an appointment, and the booking details are saved in **Amazon DynamoDB**, enabling fast retrieval and updates. Immediately, **AWS SNS** sends a confirmation SMS and email to the patient with appointment details.

Scenario 2 – Medical Record Update with Secure Access

A doctor logs in through the MedTrack portal running on **EC2**. Using permissions managed by **AWS IAM**, the doctor can securely access and update patient records stored in **DynamoDB**. Any changes are recorded in real time so all authorized staff see up-to-date information.

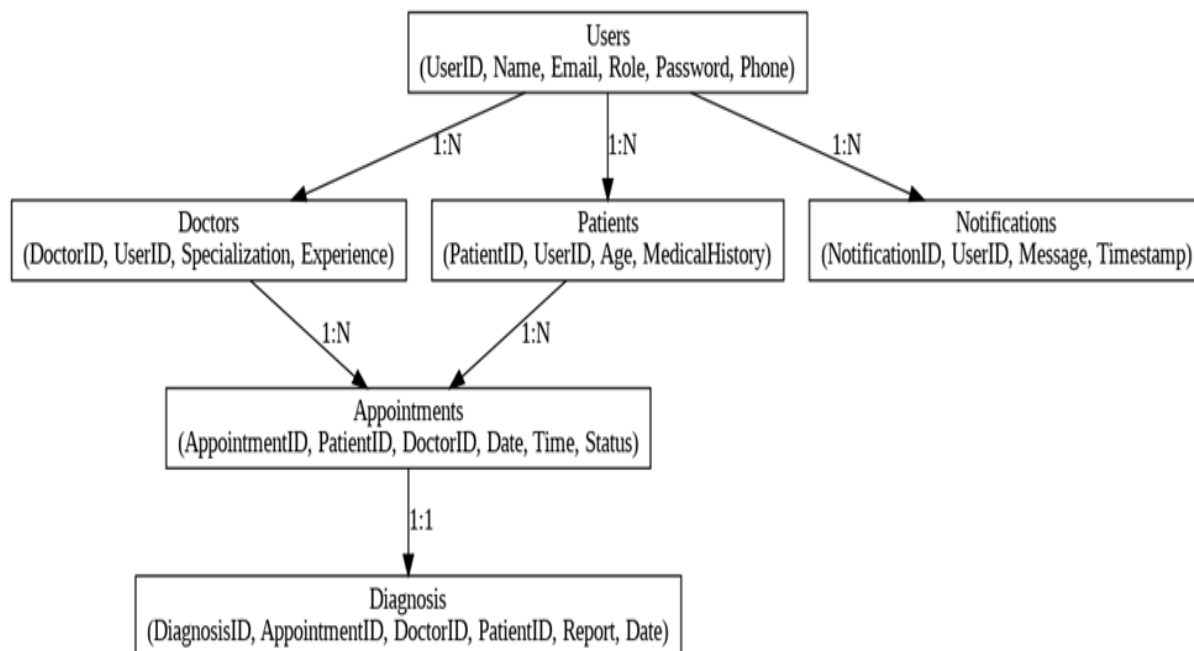
Scenario 3 – Emergency Alerts and Notifications

When critical lab results are entered into the system, MedTrack triggers an alert. A Lambda function scans the **DynamoDB** table for urgent flags and uses **AWS SNS** to send high-priority notifications to the assigned doctor's mobile device. **IAM** policies ensure only authorized medical staff can receive and act on these alerts.



AWS ARCHITECTURE

Entity Relationship (ER)Diagram:



Pre-requisites:

- AWS Account Setup:**
<https://docs.aws.amazon.com/accounts/latest/reference/getting-started.html>
- AWS IAM (Identity and Access Management):**
<https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html>
- AWS EC2 (Elastic Compute Cloud):**
<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>

4. AWS DynamoDB:

<https://docs.aws.amazon.com/amazondynamodb/Introduction.html>

5. Amazon SNS:

<https://docs.aws.amazon.com/sns/latest/dg/welcome.htm>

6. Git Documentation:

<https://git-scm.com/doc>

7. VS Code Installation: (download the VS Code using the below link or you can get that in Microsoft store)

<https://code.visualstudio.com/download>

Project WorkFlow:

1. AWS Account Setup and Login

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

Activity 1.2: Log in to the AWS Management Console

2. DynamoDB Database Creation and Setup

Activity 2.1: Create a DynamoDB Table.

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

3. SNS Notification Setup

Activity 3.1: Create SNS topics for book request notifications.

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

4. Backend Development and Application Setup

Activity 4.1: Develop the Backend Using JavaScript.

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 JavaScript application.

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

7. Deployment on EC2

Activity 7.1: Upload JavaScript

Files

8. Activity 7.2: Run the

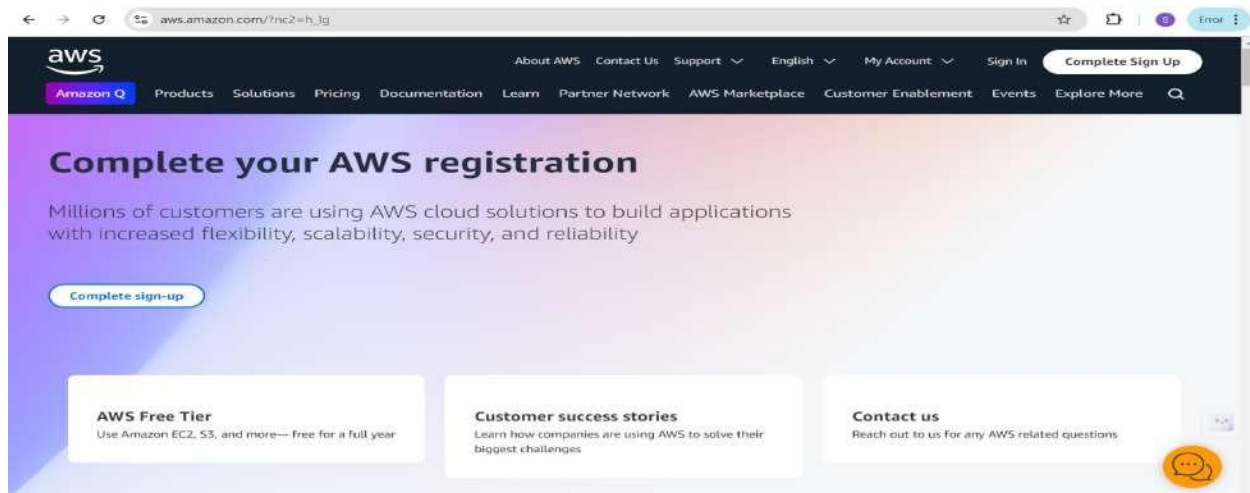
JavaScript App

9. Testing and Deployment

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

Milestone 1: AWS Account Setup and Login

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

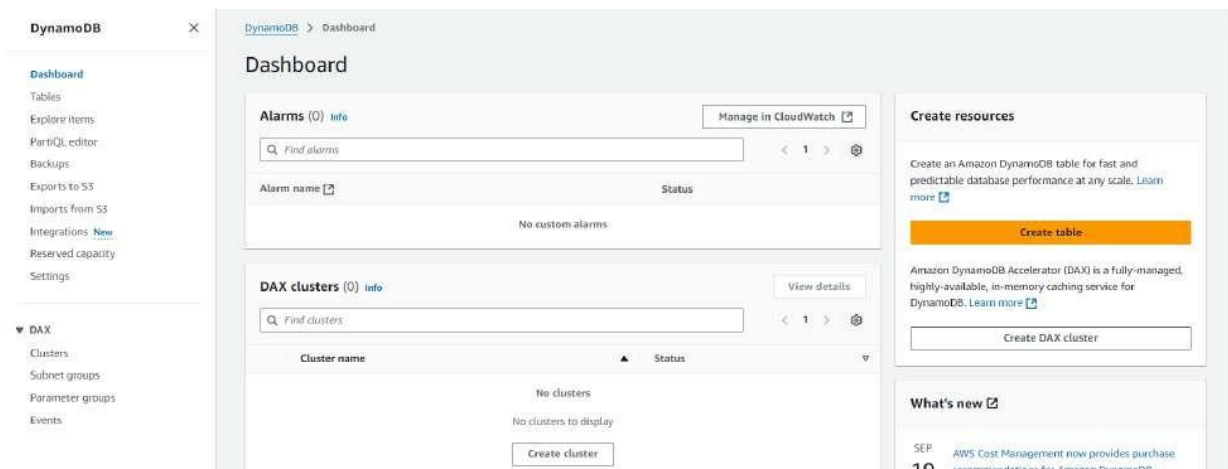
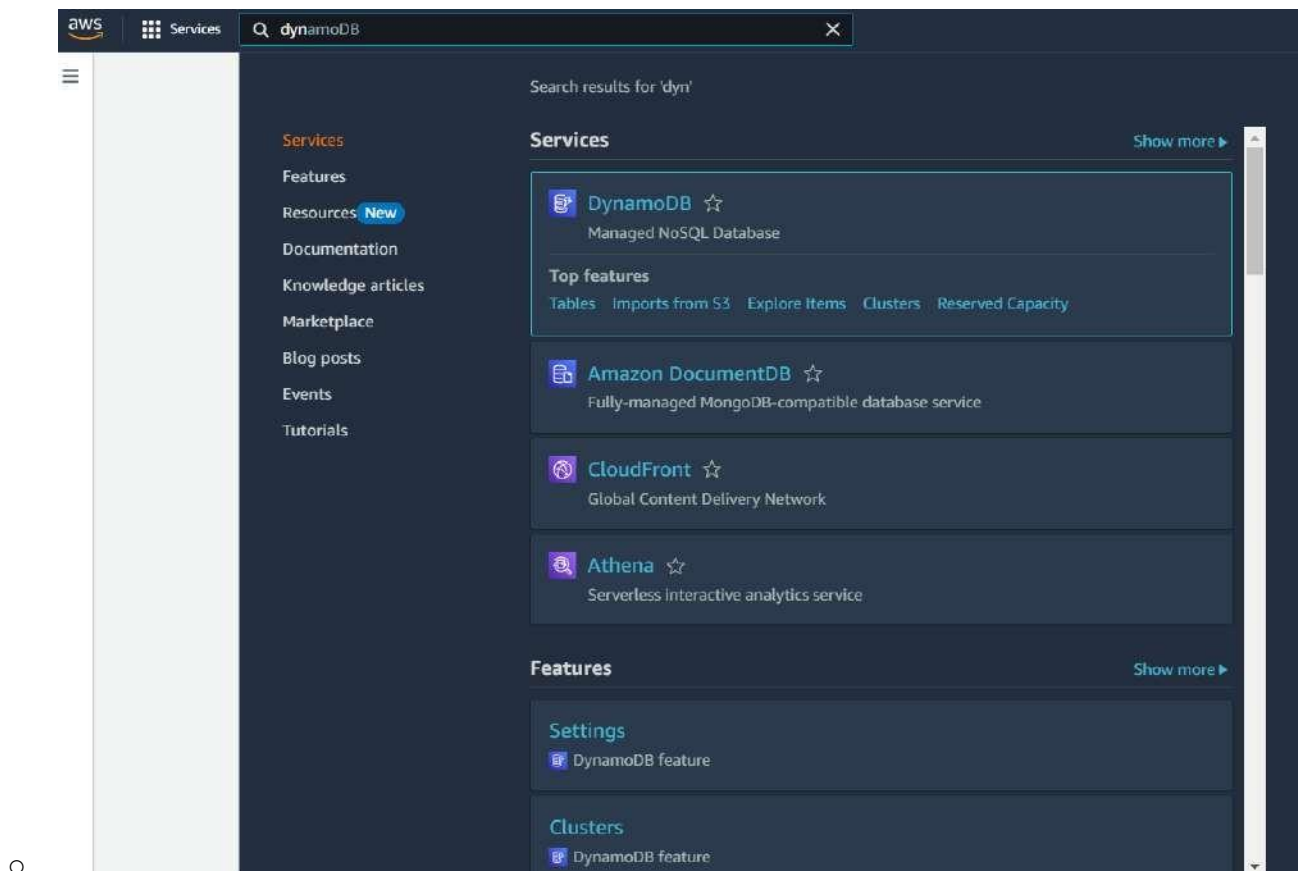


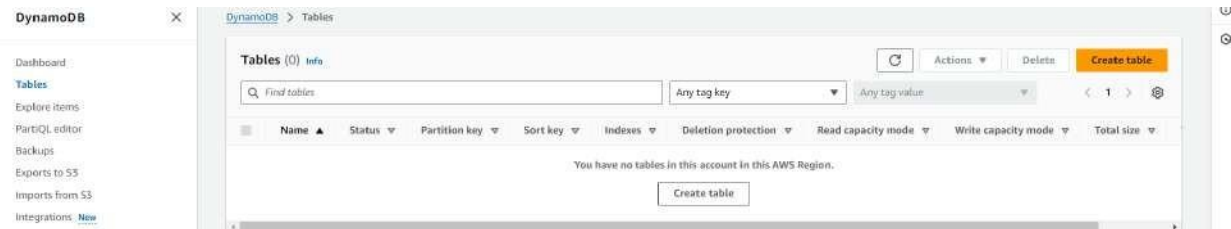
- **Activity 1.2: Log in to the AWS Management Console**
 - After setting up your account, log in to the [AWS Management Console](#).



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.





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

[DynamoDB](#) > [Tables](#) > Create table

Create table

Table details
[Info](#)

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

String ▼

1 to 255 characters and case sensitive.

Sort key - optional

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

String ▼

1 to 255 characters and case sensitive.

☰

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

Tags

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

No tags are associated with the resource.

Add new tag

You can add 50 more tags.

Cancel
Create table

DynamoDB
 Dashboard
Tables
 Explore items
 PartiQL editor
 Backups
 Exports to S3
 Imports from S3
 Integrations New

The Users table was created successfully.

DynamoDB > Tables
Refresh
Actions
Delete
Create table

Any tag key

Any tag value

<input type="checkbox"/>	Name ▲	Status ▼	Partition key ▼	Sort key ▼	Indexes ▼	Deletion protection ▼	Read capacity mode ▼	Write capacity mode ▼	Total size ▼
<input type="checkbox"/>	Users	Active	email (S)	-	0	Off	Provisioned (S)	Provisioned (S)	0 bytes

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

" [DynamoDB](#) \ [Tables](#) } Create table

Create table

Table details info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

Between 3 and 255 characters, curlyLaJning only letters, rJumLt s, uryderscores , hmheris (-), and p@iods (.).

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

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 ▼

4 to 255 characters and case sensitive.

Table settings

☒ Default settings

☐ Customize settings



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

Tags

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

No tags are associated with the resource.

Add new tag

You can add 50 more tags.

Cancel

Create table

DynamoDB

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

The Requests table was created successfully.

DynamoDB > Tables

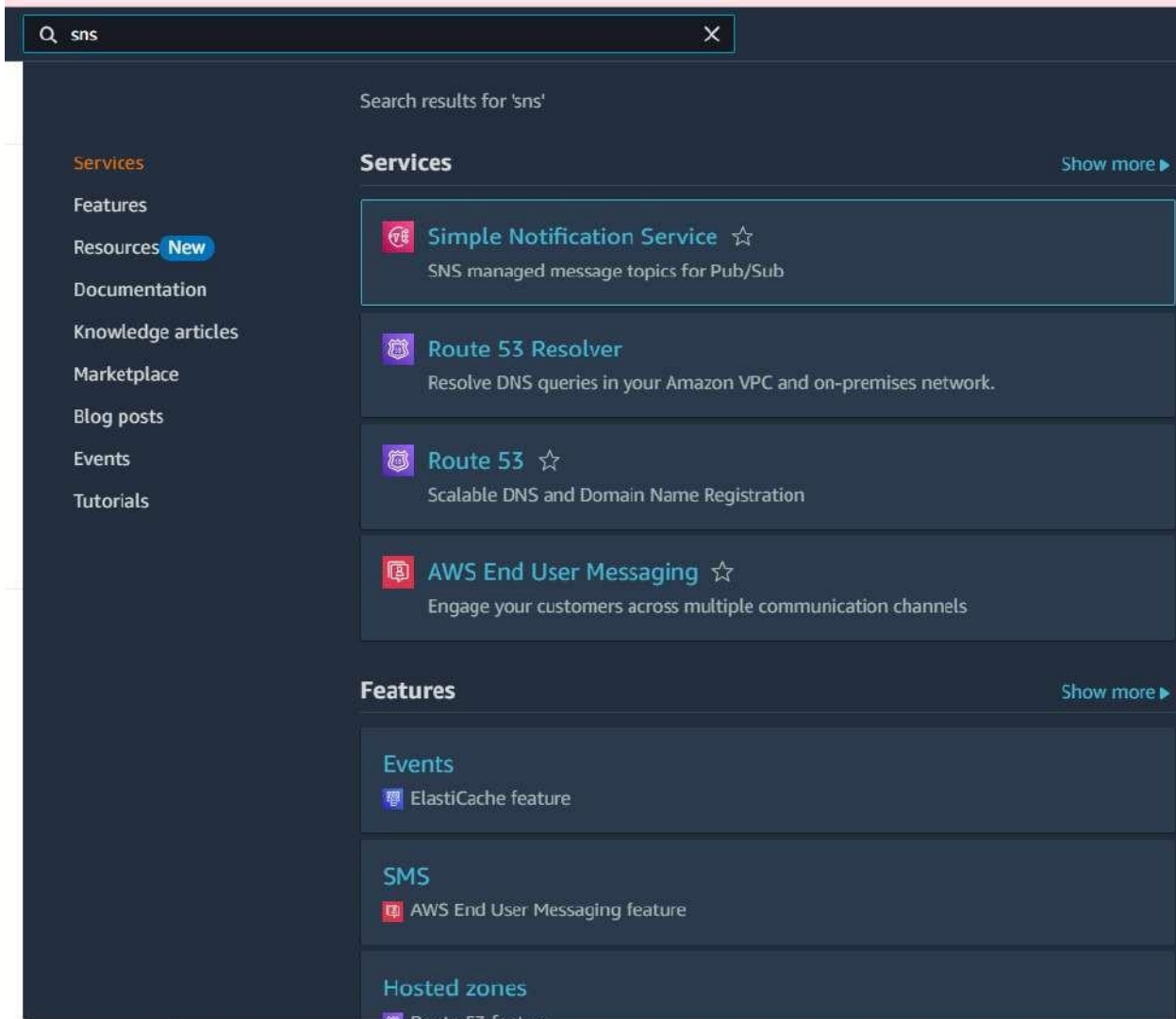
Tables (2) Info

Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size
Requests	Active	email (S)	-	0	Off	Provisioned (5)	Provisioned (5)	0 bytes
Users	Active	email (S)	-	0	Off	Provisioned (5)	Provisioned (5)	0 bytes

Milestone 3: SNS Notification Setup

- **Activity 3.1: Create SNS topics for sending email notifications to users and library staff.**





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






The screenshot shows the AWS Console search results for 'sns'. The search bar at the top contains 'sns' and a close button. Below the search bar, the text 'Search results for 'sns'' is displayed. On the left side, there is a navigation menu with links to Services, Features, Resources (marked as 'New'), Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main content area is divided into two sections: 'Services' and 'Features'. The 'Services' section lists four services: Simple Notification Service (with a star icon), Route 53 Resolver, Route 53 (with a star icon), and AWS End User Messaging (with a star icon). Each service entry includes a brief description. The 'Features' section lists three features: Events (with an ElastiCache feature sub-entry), SMS (with an AWS End User Messaging feature sub-entry), and Hosted zones (with a Route 53 feature sub-entry). Each feature entry includes a brief description. 'Show more' links are present at the end of both the Services and Features sections.

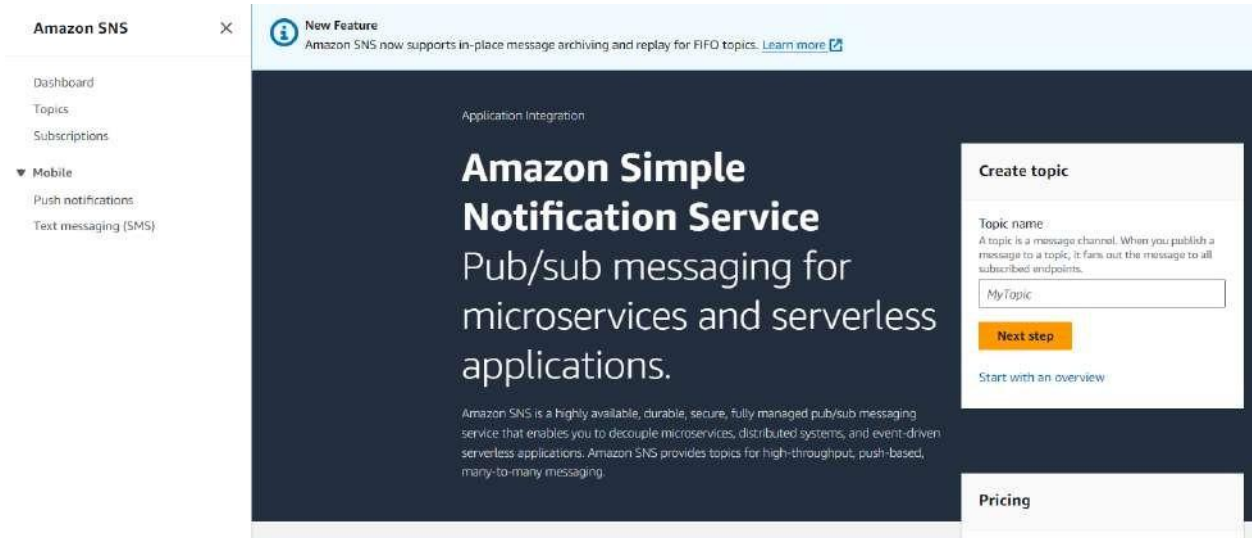
Search results for 'sns'

Services [Show more ►](#)

-  **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 [Show more ►](#)

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



Amazon SNS

Dashboard
Topics
Subscriptions

▼ Mobile
Push notifications
Text messaging (SMS)

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.

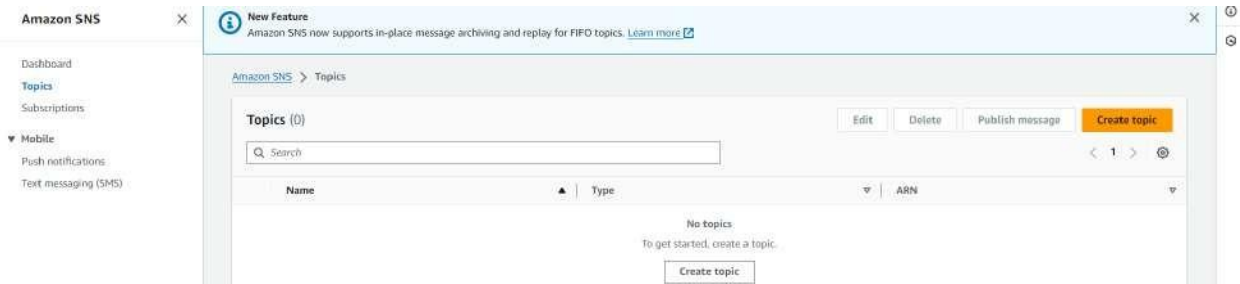
MyTopic

Next step

[Start with an overview](#)

Pricing

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



Amazon SNS

Dashboard
Topics
Subscriptions

▼ Mobile
Push notifications
Text messaging (SMS)

Topics (0)

Edit Delete Publish message **Create topic**

Search

Name	Type	ARN
No topics To get started, create a topic.		

Create topic

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

[Amazon SNS](#) \ [Topics](#) [Create topic](#)

Create topic

Details

Type **rrfo**

Topic type cannot be modified after topic is created

☐ **FIFO (first-in, first-out)**

- Strictly—preserved message ordering
- Exactly-once message delivery
- High throughput, up to 200 publishes/second
- Subscription protocol: SQS

☒ **Standard**

Best-effort message ordering
at-least once message delivery
Highest throughput in publishes/second
Subscription protocols: SQS, Lambda, HTTP, SNS,
email, mobile application endpoints

Name

BookRequestNotification

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - optional **rrfo**

To use this topic with SNS subscriptions, enter a display name. Only the first 10 characters are displayed in an SNS message.

Maximum 100 characters.

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

Amazon SNS

Dashboard

Topics

Subscriptions

Mobile

Push notifications

Text messaging (SMS)

New Feature

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

Topic: BookRequestNotifications created successfully.

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

Publish message

Amazon SNS > Topics > BookRequestNotifications

BookRequestNotifications

Edit

Delete

Publish message

Details

Name

BookRequestNotifications

Display name

-

ARN

arn:aws:sns:ap-south-1:557690616835:BookRequestNotifications

Topic owner

557690616835

Type

Standard

Subscriptions

Access policy

Data protection policy

Delivery policy (HTTP/S)

Delivery status logging

Encryption

Tags

Integrations

Subscriptions (0)

Q: Search

Edit

Delete

Request confirmation

Confirm subscription

Create subscription

ID

Endpoint

Status

Protocol

No subscriptions found.

You don't have any subscriptions to this topic.

Create subscription

- **Activity 3.2: Subscribe users and staff to relevant SNS topics to receive real-time notifications when a book request is made.**
 - Subscribe users (or admin staff) to this topic via Email. When a book request is made, notifications will be sent to the subscribed emails.

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN

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**

Amazon SNS x

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

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

Amazon SNS > Topics > BookRequestNotifications > Subscriptions: d78e0371-9235-404d-952c-85c2743607c4

Subscription: d78e0371-9235-404d-952c-85c2743607c4 [Edit] [Delete]

Details

ARN <code>arn:aws:sns:south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4</code>	Status Pending confirmation
Endpoint <code>instantlibrary2@gmail.com</code>	Protocol EMAIL
Topic BookRequestNotifications	
Subscription provided <code>arn:aws:sns:south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4</code>	

Subscription filter policy | Redrive policy (dead-letter queue)

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

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

[Edit]

- After subscription request for the mail confirmation

Amazon SNS x

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

Confirmation request was sent successfully.
The ARN of the subscription is `arn:aws:sns:south-1:557690616836:BookRequestNotifications:7ae5d516-12ad-4711-9f7d-34d2713ae5`.

Amazon SNS > Topics > BookRequestNotifications

BookRequestNotifications [Edit] [Delete] [Publish message]

Details

Name <code>BookRequestNotifications</code>	Display name -
ARN <code>arn:aws:sns:south-1:557690616836:BookRequestNotifications:7ae5d516-12ad-4711-9f7d-34d2713ae5</code>	Topic owner <code>557690616836</code>
Type Standard	

Subscriptions | Access policy | Data protection policy | Delivery policy (HTTP/S) | Delivery status logging | Encryption | Tags | Integrations

Subscriptions (2) [Edit] [Delete] [Request confirmation] [Confirm subscription] [Create subscription]

Search

ID	Endpoint	Status	Protocol
Pending confirmation	<code>instantlibrary2@gmail.com</code>	Pending confirmation	EMAIL

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

AWS Notification - Subscription Confirmation Inbox x

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

9

to me ▼

You have chosen to subscribe to the topic:

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

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

[Confirm subscription](#)

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

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

to me ▼

You have chosen to subscribe to the topic:

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

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

[Confirm subscription](#)

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



Simple Notification Service

Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:

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

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

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

Amazon SNS > Topics > BookRequestNotifications

BookRequestNotifications Edit Delete Publish message

Details

Name BookRequestNotifications	Display name -
ARN arn:aws:sns:ap-south-1:55760619636:BookRequestNotifications	Topic owner 55760619636
Type Standard	

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

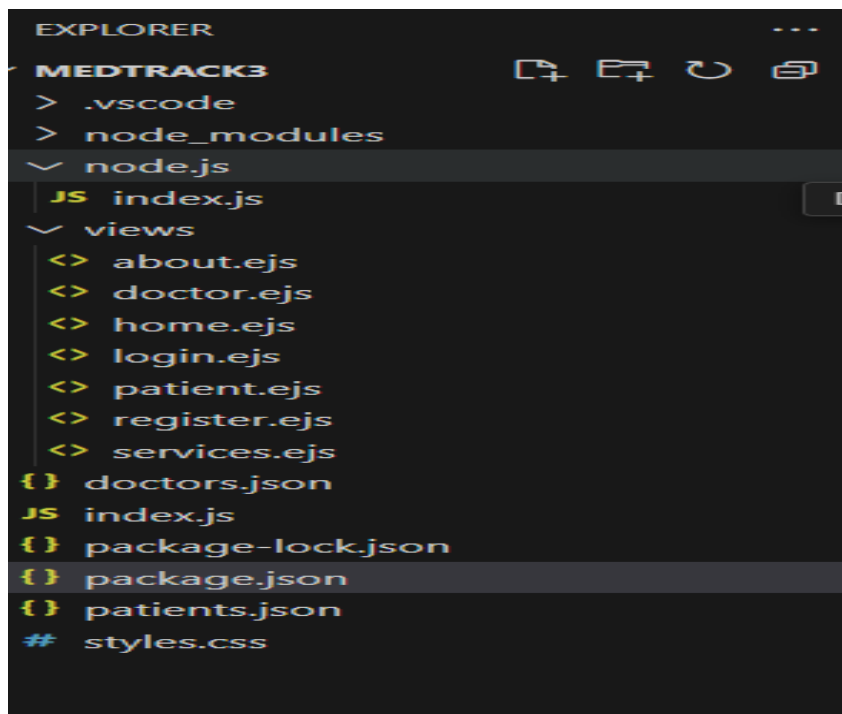
Subscriptions (2) Edit Delete Request confirmations Confirm subscription Create subscription

Q Search

ID	Endpoint	Status	Protocol
arn:aws:sns:ap-south-1:55760619636:BookRequestNotifications:123456789	instantlibrary2@gmail.com	Confirmed	EMAIL

Milestone 4: Backend Development and Application Setup

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





Description: set up the INSTANT LIBRARY project with an index.js file, a public/ folder for assets, and a views/ 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 :

- **Node js (Express) Initialization**

```
JS index.js X .env <% index.ejs
JS index.js > app.get("/") callback
1 import express from 'express';
2 import path from 'path';
3 import { fileURLToPath } from 'url';
4 import bcrypt from 'bcrypt';
5 import session from 'express-session';
6 import dotenv from 'dotenv';
7 import { DynamoDBClient } from '@aws-sdk/client-dynamodb';
8 import { SNSClient, PublishCommand } from '@aws-sdk/client-sns';
9 import { DynamoDBDocumentClient, GetCommand, PutCommand, QueryCommand, UpdateCommand } from '@aws-sdk/lib-dynamodb';
```

Description: import essential libraries including Express utilities for routing, Boto3 for DynamoDB operations, SMTP and email modules for sending mails, and Bcrypt for password hashing and verification

```
14 const __filename = fileURLToPath(import.meta.url);
15 const __dirname = path.dirname(__filename);
16 const app = express();
17 const port = process.env.PORT || 3000;
```

Description: initialize the Flask application instance using __filename to get the path of the directory and app is used to get express to start building the web app.

- **Dynamodb Setup:**

```
52 // Helper functions for DynamoDB
53 const getUserByEmail = async (email, role) => {
54   const tableName = role === 'doctor' ? DOCTORS_TABLE : PATIENTS_TABLE;
55   const command = new GetCommand({
56     TableName: tableName,
57     Key: { email }
58   });
59   const { Item } = await docClient.send(command);
60   return Item;
61 };
62
```

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**

```
214 // Send SNS notification
215 if (SNS_TOPIC_ARN) {
216   const snsCommand = new PublishCommand({
217     TopicArn: SNS_TOPIC_ARN,
218     Message: `New doctor registered: ${firstName} ${lastName} (${email}) - ${specialization}`,
219     Subject: 'MedTrack Registration'
220   });
221   await snsClient.send(snsCommand).catch(err => console.error('SNS Error:', err));
222 }
223
224 res.redirect('/login');
225 });
226
```

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

- **Routes for Web Pages**

- **Home Route:**

```
139
140 app.get("/", (req, res) => res.render("index"));
141 app.get("/contactus", (req, res) => res.render("contactus"));
142 app.get("/about", (req, res) => res.render("about"));
143 app.get("/register", (req, res) => res.render("register"));
144
```

Description: define the home route / to automatically redirect users to the register page when they access the base URL.

- Register Route:

```
61 app.post('/register/patient', async (req, res) => {
62   const { firstName, lastName, dob, gender, email, phone, address, password } = req.body;
63
64   // Check if patient exists
65   const existingPatient = await getUserByEmail(email, 'patient');
66   if (existingPatient) return res.send('Patient exists');
67
68   // Create new patient
69   await createUser({
70     id: Date.now().toString(),
71     name: `${firstName} ${lastName}`,
72     dob,
73     gender,
74     email,
75     phone,
76     address,
77     password: await bcrypt.hash(password, 10)
78   }, 'patient');
79
80   // Send SNS notification
81   if (SNS_TOPIC_ARN) {
82     const snsCommand = new PublishCommand({
83       TopicArn: SNS_TOPIC_ARN,
84       Message: `New patient registered: ${firstName} ${lastName} (${email})`,
85       Subject: 'MedTrack Registration'
86     });
87     await snsClient.send(snsCommand).catch(err => console.error('SNS Error:', err));
88   }
89
90   res.redirect('/login');
91 });
92
```

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.post('/check', async (req, res) => {
  const { email, password, role } = req.body;

  const user = await getUserByEmail(email, role);
  if (!user || !(await bcrypt.compare(password, user.password))) {
    return res.render('login', { message: 'Invalid credentials' });
  }

  req.session.user = {
    id: user.id,
    name: user.name,
    email: user.email,
    role: user.role
  };
  res.redirect(`/${role}`);
});
```

Description: define /login route to validate user credentials against DynamoDB, check the password using Bcrypt, update the login count on successful authentication, and redirect users to the home page

- **Home, E- book buttons and subject routes:**

```
// Appointment actions
app.post('/doctor/appointment/:id/precautions', requireDoctor, async (req, res) => {
  await updateAppointment(req.params.id, {
    precautions: req.body.precautions,
    status: 'Completed',
    updatedAt: new Date().toISOString()
  });
  res.redirect('/doctor');
});

app.post('/doctor/appointment/:id/reschedule', requireDoctor, async (req, res) => {
  await updateAppointment(req.params.id, {
    date: req.body.date,
    time: req.body.time,
    status: 'Rescheduled',
    updatedAt: new Date().toISOString()
  });
  res.redirect('/doctor');
});

app.post('/doctor/appointment/:id/cancel', requireDoctor, async (req, res) => {
  await updateAppointment(req.params.id, {
    status: 'Cancelled',
    updatedAt: new Date().toISOString()
  });
  res.redirect('/doctor');
});
```

Description: define /home-page to render the main homepage, /ebook-buttons to handle subject selection and redirection, and /<subject>.html dynamic route to render subject-specific pages.

- **Request Routes:**

```
// Create appointment
await createAppointment({
  doctorId,
  doctorName: doctor.name,
  specialty: doctor.specialization,
  patientId: req.session.user.id,
  patientName: req.session.user.name,
  date,
  time,
  reason
});

res.redirect('/patient');
```

Description: define /request-form route to capture book request details from users, store the request in DynamoDB, send a thank-you email to the user, notify the admin, and confirm submission with a success message.

Exit Route:

```
app.get('/logout', (req, res) => {
  req.session.destroy(() => res.redirect('/'));
});
```

Description: define /exit route to render the exit.html page when the user chooses to leave or close the application.

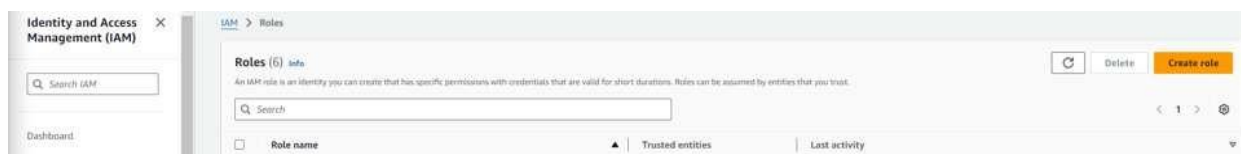
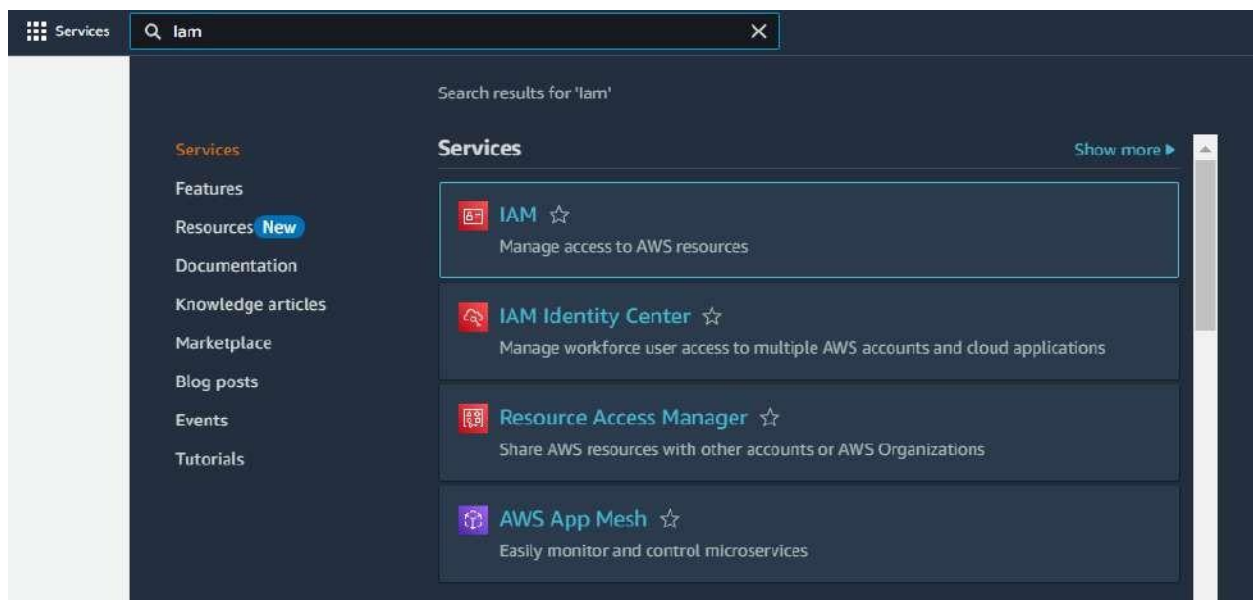
Deployment Code:

```
app.listen(3000, '0.0.0.0', () => {  
  console.log(`Server running at http://localhost:3000`);  
  console.log(`Using AWS Region: ${REGION}`);  
  console.log(`Patients Table: ${PATIENTS_TABLE}`);  
  console.log(`Doctors Table: ${DOCTORS_TABLE}`);  
  console.log(`Appointments Table: ${APPOINTMENTS_TABLE}`);  
});
```

Description: start the Flask server to listen on all network interfaces (0.0.0.0) at port 80 with debug mode enabled for development and testing.

Milestone 5: IAM Role Setup

- **Activity 5.1: Create IAM Role.**
 - In the AWS Console, go to IAM and create a new IAM Role for EC2 to interact with DynamoDB and SNS.



Select trusted entity

Trusted entity type

☒ **API Lambda**
Allows an AWS Lambda function to perform actions in this account.

☐ **AWS account**
Allows an AWS account to perform actions in this account.

☐ **Web identity**
Allows an external provider to perform actions in this account.

☐ **RAM, ETO Federation**
Allows an external provider to perform actions in this account.

☐ **Custom trust policy**
Creates a custom trust policy to perform actions in this account.

Use case

What is the use case for this role?

Use case or role name

EC2

Choose a use case for the specified role.

Use case

☒ **EC2**
Allows an EC2 instance to perform actions in your account.

☐ **EC2 Role for AWS Systems Manager**
Allows an EC2 instance to use AWS Systems Manager to manage its configuration.

☐ **EC2 Spot Fleet Role**
Allows an EC2 Spot Fleet to request and manage Spot instances in your account.

☐ **EC2 - Spot Fleet Auto Scaling**
Allows an EC2 Spot Fleet to request and manage Spot instances in your account.

☐ **EC2 - Spot Fleet Tagging**
Allows an EC2 Spot Fleet to request and manage Spot instances in your account.

☐ **EC2 - Spot Instance**
Allows an EC2 Spot Instance to request and manage Spot instances in your account.

☐ **EC2 - Spot Fleet**
Allows an EC2 Spot Fleet to request and manage Spot instances in your account.

☐ **EC2 - Scheduled Instance**
Allows an EC2 Scheduled Instance to request and manage Spot instances in your account.

Cancel Next

Add permissions

Permissions policies (1/955)

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

Filter by Type: All types 2 matches

Policy name	Type
AmazonDynamoDBFullAccess	AWS managed
AmazonSNSFullAccess	AWS managed

Set permissions boundary - optional

Cancel Previous Next

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.

Add permissions

IAM > Roles > sns_Dynamodb_role

sns_Dynamodb_role Info

Allows EC2 instances to call AWS services on your behalf.

Summary Edit

Creation date October 13, 2024, 23:06 (UTC+05:30)	ARN <code>arn:aws:iam::55-7690616826:role/sns_Dynamodb_role</code>	Instance profile ARN <code>arn:aws:iam::55-7690616826:instance-profile/sns_Dynamodb_role</code>
Last activity 6 days ago	Maximum session duration 1 hour	

Permissions Trust relationships Tags Last Accessed Revoke sessions

Permissions policies (2) Info


You can attach up to 10 managed policies.

Filter by Type: All types

<input type="checkbox"/>	Policy name	Type	Attached entities
<input type="checkbox"/>	AmazonDynamoDBFullAccess	AWS managed	4
<input type="checkbox"/>	AmazonSNSFullAccess	AWS managed	2


Milestone 6: EC2 Instance Setup

- Note:** Load your `index.js` and `Html` files into GitHub repository.


Medtrack Public
Pin Watch 0

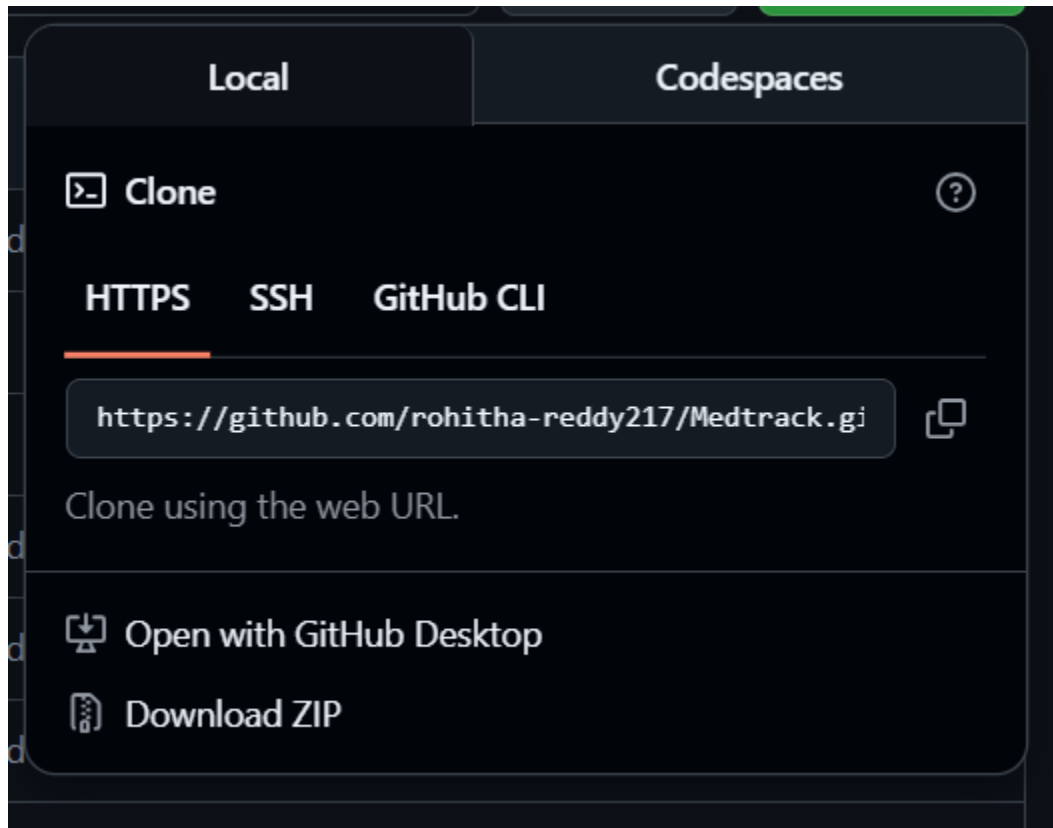
main
1 Branch
0 Tags

Add file
Code

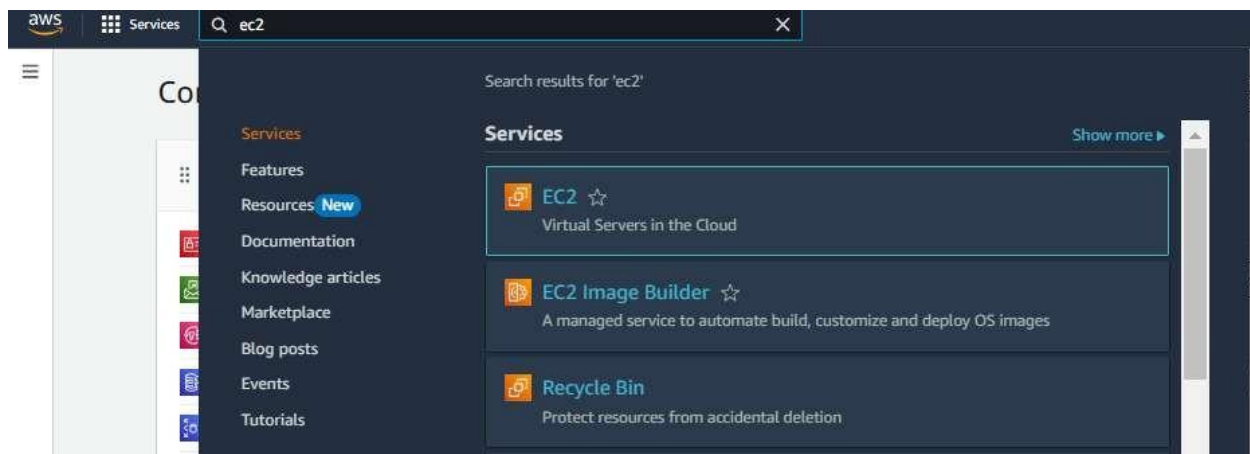

rohitha-reddy217
Update .env

1309f34 · 3 days ago
18 Commits

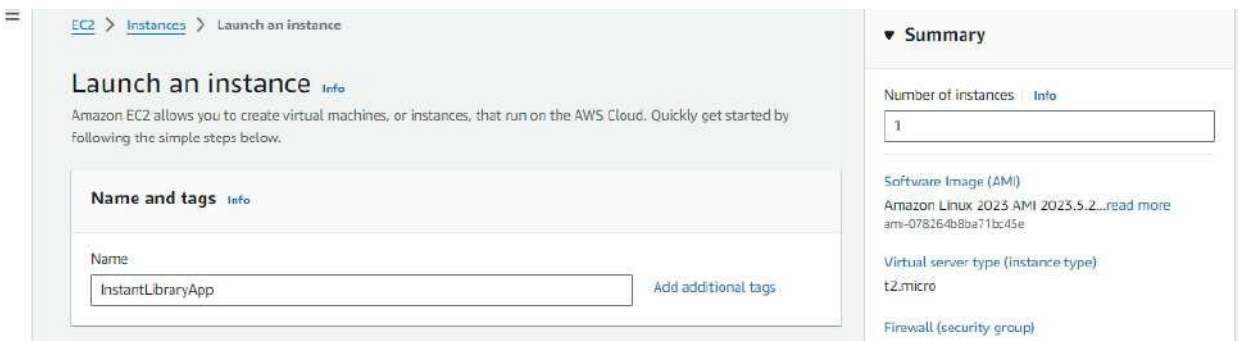
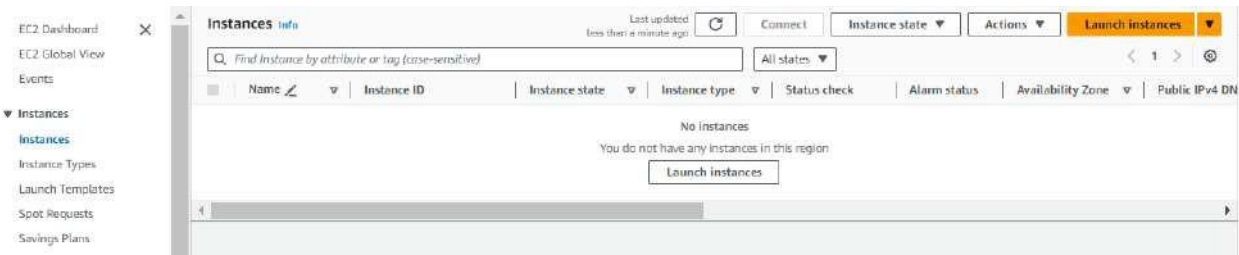
File	Action	Time
Views	Add files via upload	4 days ago
.env	Update .env	3 days ago
README.md	Initial commit	4 days ago
doctors.json	Add files via upload	4 days ago
index.js	Add files via upload	4 days ago
package-lock.json	Add files via upload	4 days ago
package.json	Add files via upload	4 days ago
patients.json	Add files via upload	4 days ago
styles.css	Add files via upload	4 days ago



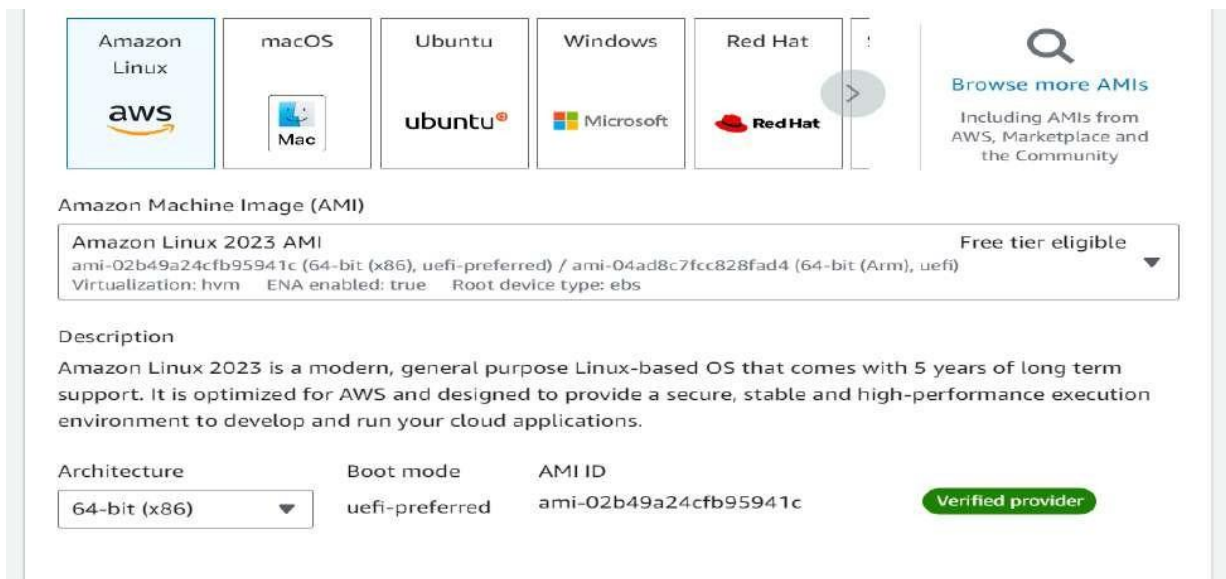
- **Activity 6.1: Launch an EC2 instance to host the Flask application.**
 - **Launch EC2 Instance**
 - In the AWS Console, navigate to EC2 and launch a new instance.



- Click on Launch instance to launch EC2 instance



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



- 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

InstantLibrary.pem

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

Architecture

64-bit (x86)

Boot mode

uefi-preferred

AMI ID

ami-078264b8ba71b

Instance type

t3.micro

Key pair (login)

Info

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

Key pair name

required

▼ Summary

Number of instances

1

Software Image

Amazon Linux 2023

ami-078264b8ba71b

Fin

New security group

Free tier

In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

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

Network settings
Info

VPC - required
Info

vpc-03cdc7b6f19dd7211
172.31.0.0/16

(default)

Subnet
Info

No preference

Create new subnet

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

Add CIDR, prefix list or security group

0.0.0.0/0

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

Add CIDR, prefix list or security group

0.0.0.0/0

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

Add CIDR, prefix list or security group

0.0.0.0/0

Description - optional
Info

e.g. SSH for admin desktop

Add security group rule

EC2 > ... > Launch an instance

Success
Successfully initiated launch of instance i-001861022fbcac290

Launch log

Next Steps

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

To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.

Create billing alerts

Connect to your instance

Once your instance is running, log into it from your local computer.

Connect to instance

Learn more

Connect an RDS database

Configure the connection between an EC2 instance and a database to allow traffic flow between them.

Connect an RDS database

Create a new RDS database

Learn more

Create EBS snapshot policy

Create a policy that automates the creation, retention, and deletion of EBS snapshots.

Create EBS snapshot policy

Manage detailed monitoring

Enable or disable detailed monitoring for the instance. If you enable detailed monitoring, the Amazon EC2 console displays monitoring graphs with a 1-minute period.

Manage detailed monitoring

Create Load Balancer

Create a application, network gateway or classic Elastic Load Balancer.

Create Load Balancer

Create AWS budget

AWS budgets allow you to create budgets, forecast spend, and take action on your costs and usage from a single location.

Create AWS budget

Manage CloudWatch alarms

Create or update Amazon CloudWatch alarms for the instance.

Manage CloudWatch alarms

Disaster recovery for your instances

Recover the instances you just launched into a different Availability Zone or a different Region using AWS Elastic Disaster Recovery (EDR).

Disaster recovery for your instances

Monitor for suspicious runtime activities

Amazon GuardDuty enables you to continuously monitor for malicious runtime activity and unauthorized behavior, with near real-time visibility into on-host activities occurring across your Amazon EC2 workloads.

Monitor for suspicious runtime activities

Get instance screenshot

Capture a screenshot from the instance and view it as an image. This is useful for troubleshooting an unreachable instance.

Get instance screenshot

Get system log

View the instance's system log to troubleshoot issues.

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

Find instance by attribute or tag (case-sensitive) All status

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 g
InstantLibrary...	i-001861022fbcac290	Stopped	t2.micro	-	View alarms	ap-south-1b	-	-	-	-	disabled	launch-wi

EC2 > Instances > i-001861022fbcac290

Instance summary for i-001861022fbcac290 (InstantLibraryApp) info

Updated less than a minute ago

Connect Instance state Actions

Instance ID

i-001861022fbcac290

Public IPv4 address

Instance state

Stopped

Private IPv4 addresses

172.31.3.5

IPv6 address

-

Private IP DNS name (IPv4 only)

ip-172-31-3-5-ap-south-1-compute.internal

Public IPv4 DNS

-

Hardware 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-03c07b5f13a0f7211

Auto Scaling Group name

-

IAM Role

aws_dynamodb_role

Subnet ID

subnet-0d9fa2164480ce9a9

Instance ARN

arn:aws:ec2:ap-south-1:557680616836: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	Connect Manage instance state Instance settings Networking Security Image and templates Monitor and troubleshoot
IPv6 address -	Instance state Stopped	Public IPv4 DNS -	Change security groups Get Windows password Modify IAM role
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 -	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
Answer private resource DNS name IPv4: -	Instance type t2.micro	Auto Scaling Group name -	
Auto-assigned IP address -	VPC ID vpc-03c02b6f19dd7211		
IAM Role sns_Dynamodb_role	Subnet ID subnet-0d9fa144486cd9d9		
IMDSv2 Required	Instance ARN arn:aws:ec2:ap-south-1:537090618036:instance/i-001861022fbcac290		

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

Modify IAM role [Info](#)

Attach an IAM role to your instance.

Instance ID
 i-001861022fbcac290 (InstantLibraryApp)

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

sns_Dynamodb_role

[Create new IAM role](#)

Cancel [Update IAM role](#)

- Now connect the EC2 with the files

EC2 Instance Connect Session Manager EC2 serial console

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](#).

Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.

Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

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

```
A newer release of "Amazon Linux" is available.  
Version 2023.6.20241010:  
Run "/usr/bin/dnf check-release-update" for full release and version update info
```

```
#####  
###      #####  
##       ##    ###  
####     ####   #  
V~>         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 ~]$ █
```

Milestone 7: Deployment on EC2

Activity 7.1: Install Software on the EC2 Instance

```
# Update system and install Node.js, npm, git
sudo yum update -y
curl -fsSL https://rpm.nodesource.com/setup_18.x | sudo bash -
sudo yum install -y nodejs git
```

Activity 7.2: Clone Your Flask Project from GitHub

Clone your project repository from GitHub into the EC2 instance using Git.

```
# Clone your Node.js project from GitHub

git clone https://github.com/prasannakumar133/MedTrack-repo.git

# Install project dependencies

npm installNote: change your-github-username and your-repository-name with your
```

- This will download your project to the EC2 instance.

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

```
cd MedTrack-repo
```

Once inside the project directory, configure and run the Nodejs application by executing the following command with elevated privileges:

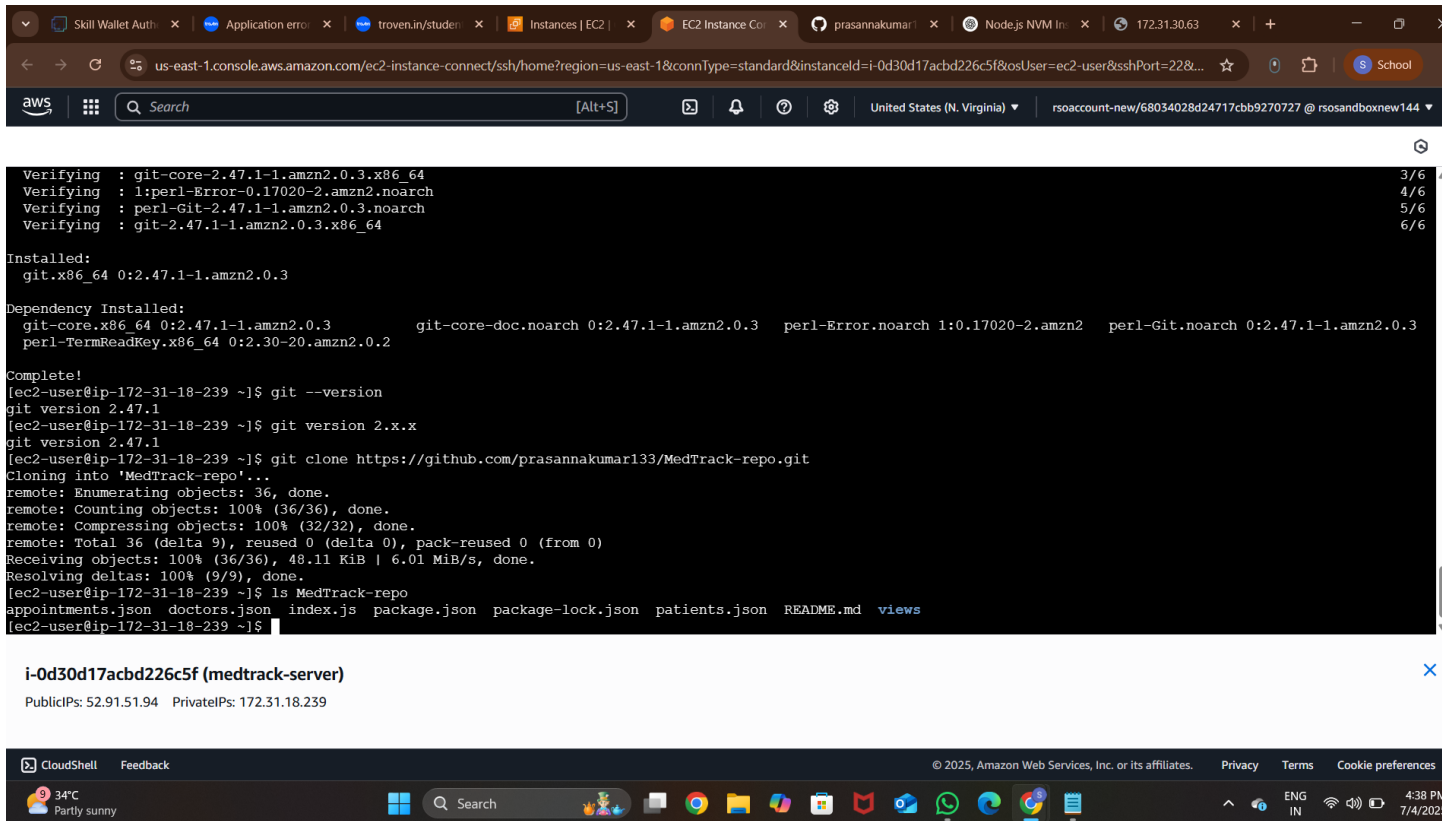
Run the Flask Application

```
# Install project dependencies
npm install

# Install PM2 globally to keep the app running
sudo npm install -g pm2

# Start the app using PM2 (replace 'app.js' with your entry point if different)
pm2 start app.js
pm2 save
pm2 startup
# Copy and run the command shown by the previous line (for startup on reboot)

# (Optional) Allow the app port (3000) through firewall
sudo firewall-cmd --zone=public --permanent --add-port=3000/tcp
sudo firewall-cmd --reload
```



The screenshot shows the AWS Management Console interface for an EC2 instance named 'i-0d30d17acbd226c5f (medtrack-server)'. The terminal output displays the following commands and results:

```

Verifying : git-core-2.47.1-1.amzn2.0.3.x86_64
Verifying : 1:perl-Error-0.17020-2.amzn2.noarch
Verifying : perl-Git-2.47.1-1.amzn2.0.3.noarch
Verifying : git-2.47.1-1.amzn2.0.3.x86_64

Installed:
git.x86_64 0:2.47.1-1.amzn2.0.3

Dependency Installed:
git-core.x86_64 0:2.47.1-1.amzn2.0.3      git-core-doc.noarch 0:2.47.1-1.amzn2.0.3      perl-Error.noarch 1:0.17020-2.amzn2      perl-Git.noarch 0:2.47.1-1.amzn2.0.3
perl-TermReadKey.x86_64 0:2.30-20.amzn2.0.2

Complete!
[ec2-user@ip-172-31-18-239 ~]$ git --version
git version 2.47.1
[ec2-user@ip-172-31-18-239 ~]$ git version 2.x.x
git version 2.47.1
[ec2-user@ip-172-31-18-239 ~]$ git clone https://github.com/prasannakumar133/MedTrack-repo.git
Cloning into 'MedTrack-repo'...
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 36 (delta 9), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (36/36), 48.11 KiB | 6.01 MiB/s, done.
Resolving deltas: 100% (9/9), done.
[ec2-user@ip-172-31-18-239 ~]$ ls MedTrack-repo
appointments.json  doctors.json  index.js  package.json  package-lock.json  patients.json  README.md  views
[ec2-user@ip-172-31-18-239 ~]$
  
```

Below the terminal output, the instance details are shown:

i-0d30d17acbd226c5f (medtrack-server)
 PublicIPs: 52.91.51.94 PrivateIPs: 172.31.18.239

The bottom of the screenshot shows the AWS CloudShell interface with a Windows taskbar overlay at the bottom, indicating the session is running on a Windows-based EC2 instance.

Verify the Nodejs app is

running: Done! Your app is

running at `http://<your-`

`ec2-public ip>:3000`

Run the Nodejs app on the EC2

instance

Skill Wallet
Application
troven.in/stu
Instances | EC2 Instance
MedTrack-repo
Node.js NVM
New Tab
New Tab

us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?region=us-east-1&connType=standard&instanceId=i-0d30d17acbd226c5f&osUser=ec2-user&sshPort=22&...

aws
Search
[Alt+S]
United States (N. Virginia)
rsoaccount-new/68034028d24717cbb9270727 @ rso.sandboxnew144

```

aws-sdk      content-disposition  express      has-symbols      merge-descriptors  parseurl      semver        tslib
balanced-match  content-type        express-session  helmet           mime-db           path          send          type-is
bcrypt        cookie              fast-xml-parser  http-errors      mime-types        path-to-regexp  serve-static  @types
binary-extensions  cookie-signature    filelist        iconv-lite       minimatch         picomatch      setprototypeof  uid-safe
body-parser     debug              fill-range      ignore-by-default  mnemonist        process        side-channel    undefsafe
bower          depd               finalhandler    inherits         ms                proxy-addr     side-channel-list  unpipe
brace-expansion  dotenv             forwarded       ipaddr.js        negotiator         pstree.remy    side-channel-map  url
braces         dunder-proto       fresh           is-binary-path   node-addon-api    punycode       side-channel-weakmap  util
bytes          ee-first           fs              is-extglob       node-gyp-build    qs             simple-update-notifier  uuid
call-bind-apply-helpers  ejs              function-bind   is-glob          nodeemon          random-bytes   @smithy        vary
call-bound     encodeurl          get-intrinsic   is-number        normalize-path     range-parser   statuses       wrappy

[ec2-user@ip-172-31-18-239 MedTrack-repo]$ node index.js
[dotenv@17.0.1] injecting env (7) from .env - [tip] encrypt with dotenvx: https://dotenvx.com
(node:5875) Warning: NodeDeprecationWarning: The AWS SDK for JavaScript (v3) will
no longer support Node.js 16.x on January 6, 2025.

To continue receiving updates to AWS services, bug fixes, and security
updates please upgrade to a supported Node.js LTS version.

More information can be found at: https://a.co/74kJmMI
(Use `node --trace-warnings ...` to show where the warning was created)
Server running at http://localhost:3000
Using AWS Region: us-east-1
Patients Table: PatientsTable
Doctors Table: DoctorsTable
Appointments Table: AppointmentsTable
^C
[ec2-user@ip-172-31-18-239 MedTrack-repo]$

```

i-0d30d17acbd226c5f (medtrack-server)

PublicIPs: [52.91.51.94](#) PrivateIPs: 172.31.18.239

CloudShell
Feedback

© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

34°C
Partly sunny

Search

ENG
IN

5:03 PM
7/4/2025

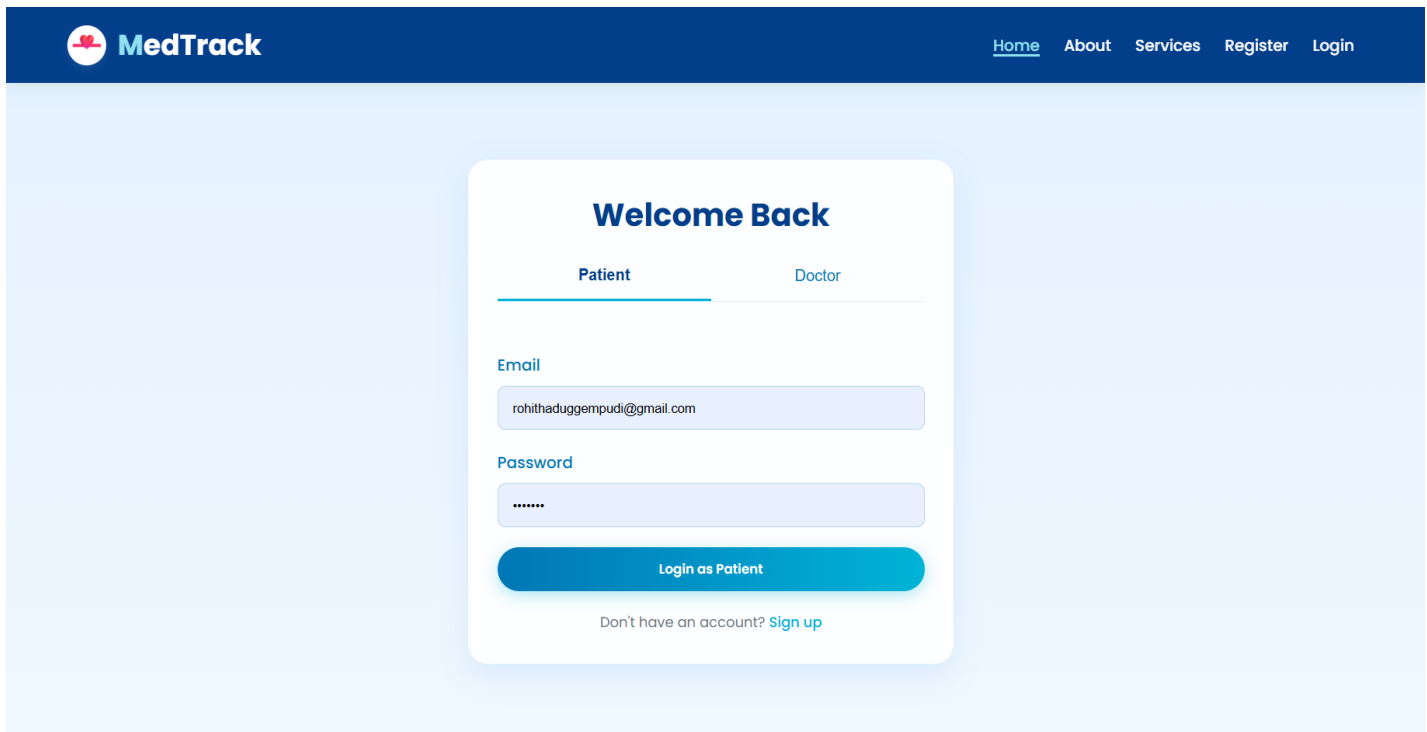
Access the website through:

PublicIPs:<http://52.91.51.94:3000/>

Milestone 8: Testing and Deployment

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

Login Page:



The screenshot shows the MedTrack login page. The header is dark blue with the MedTrack logo on the left and navigation links (Home, About, Services, Register, Login) on the right. The main content area is light blue. In the center, there is a white login card with a blue shadow. The card has a 'Welcome Back' heading. Below it are two tabs: 'Patient' (active) and 'Doctor'. There are two input fields: 'Email' with the value 'rohithaduggempudi@gmail.com' and 'Password' with masked characters. A blue 'Login as Patient' button is at the bottom of the card. Below the button is a link: 'Don't have an account? Sign up'.

MedTrack Home About Services Register Login

Welcome Back

Patient Doctor

Email


rohithaduggempudi@gmail.com

Password

Login as Patient

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

Register Page:

[Click to go back , hold to see history](#)


[About](#)
[Services](#)
[Login](#)

Create Your Account

Patient

Doctor

First Name

Last Name

Age

Gender

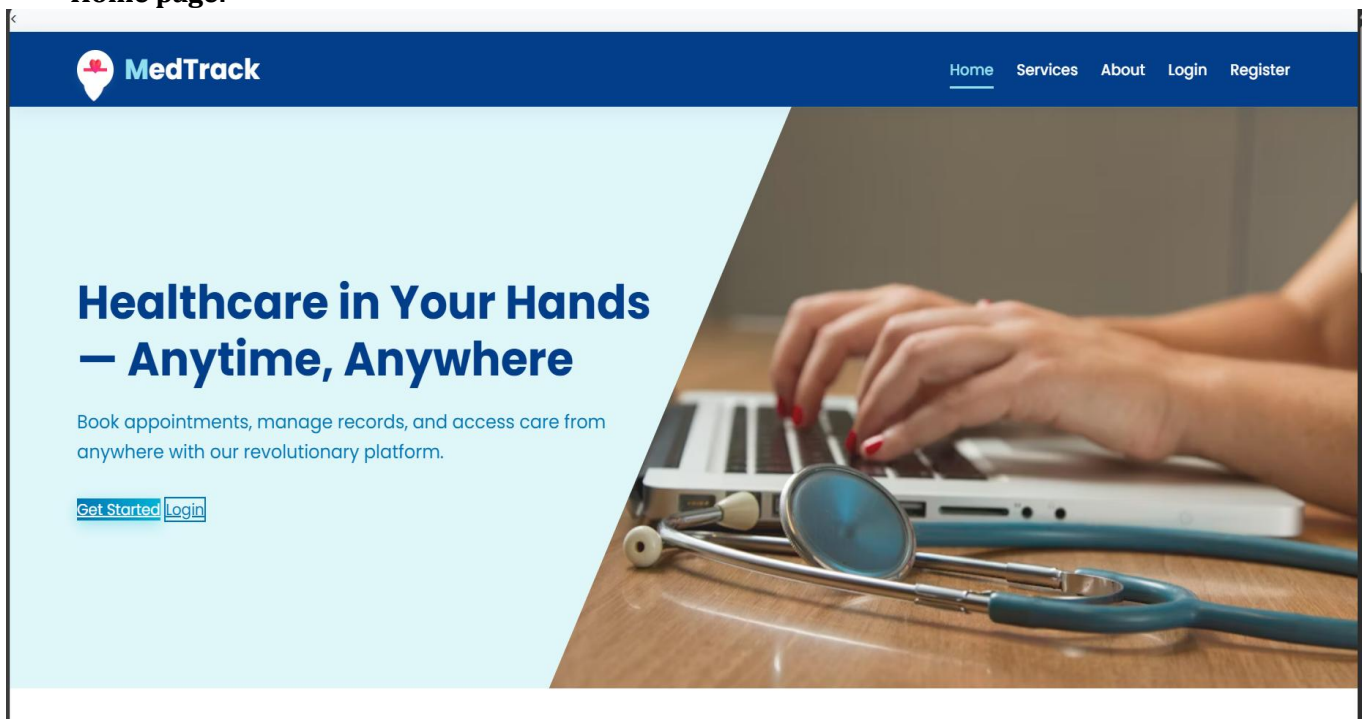
Select

Email

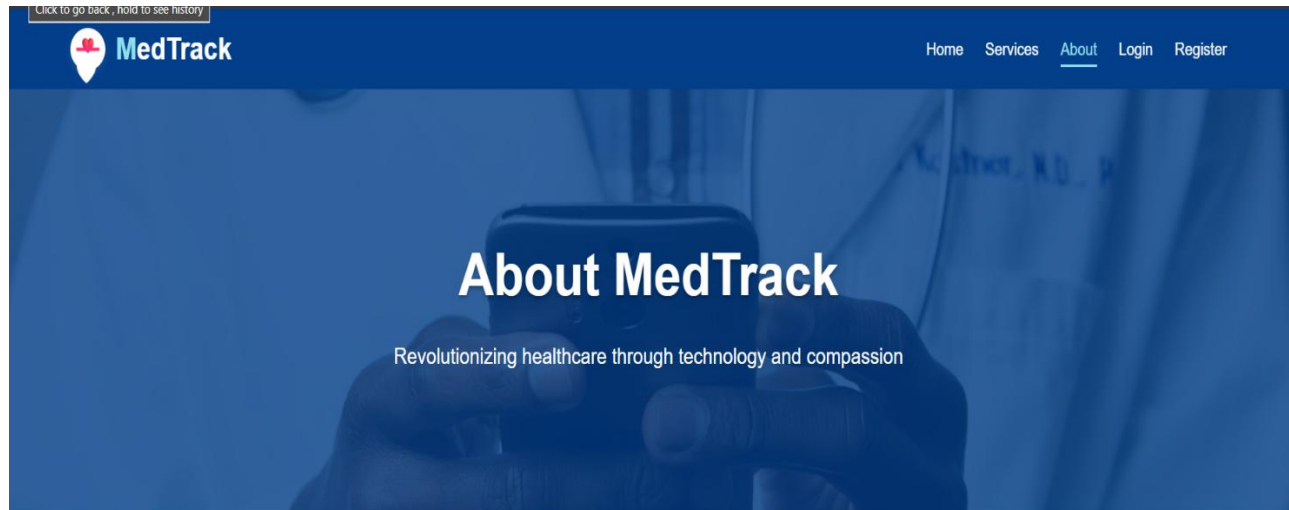
Phone

Address

Home page:



About Us page:



Founded in 2020, MedTrack emerged from a simple idea: healthcare should be accessible, efficient, and patient-centered. Our team of doctors, engineers, and healthcare professionals came together to bridge the gap between medical providers and patients through innovative technology.

Services Page



Top Services



Telemedicine



Smart Appointment



Health Records Cloud

Conclusion:

MedTrack illustrates the power of leveraging AWS cloud services to build a modern, reliable healthcare management system. By combining **Amazon EC2** for scalable application hosting, **DynamoDB** for fast and flexible data storage, **SNS** for instant notifications, and **IAM** for fine-grained access control, MedTrack ensures secure, efficient, and responsive healthcare operations.

The system streamlines the entire patient care process—from appointment scheduling and telemedicine consultations to emergency access to records and automated alerts—reducing administrative overhead and improving the patient experience. With high availability, data encryption, and compliance-ready infrastructure, MedTrack empowers healthcare providers to focus on delivering quality care while maintaining trust and data security.

Overall, this project demonstrates how cloud-native solutions can transform traditional healthcare environments into agile, scalable, and patient-centric ecosystems.













