

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

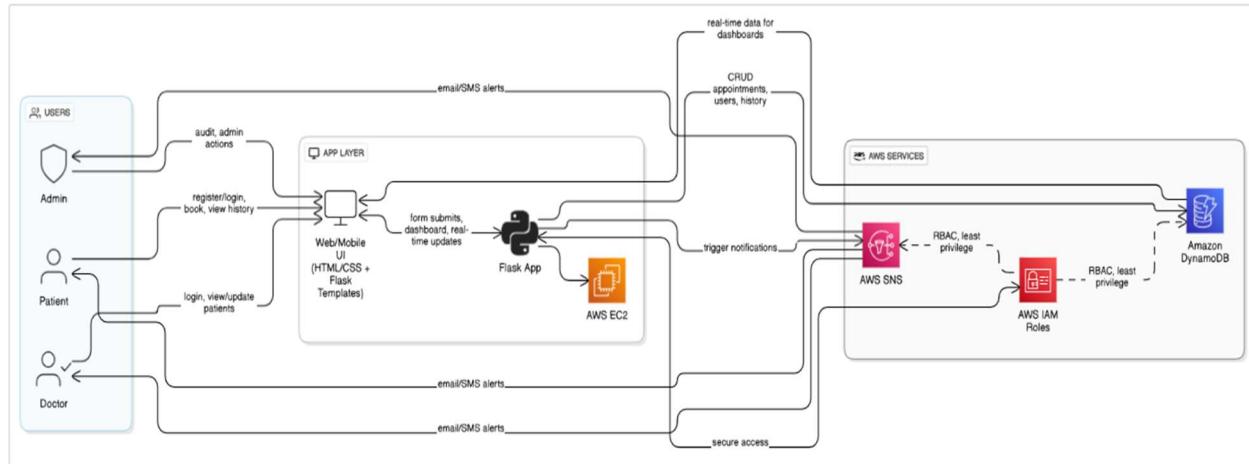
A patient logs into the MedTrack web app, which is hosted on **Amazon EC2** instances. They schedule an appointment, and the booking details are saved in **Amazon DynamoDB**, enabling fast retrieval and updates. Immediately, **AWS SNS** sends a confirmation SMS and email to the patient with appointment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

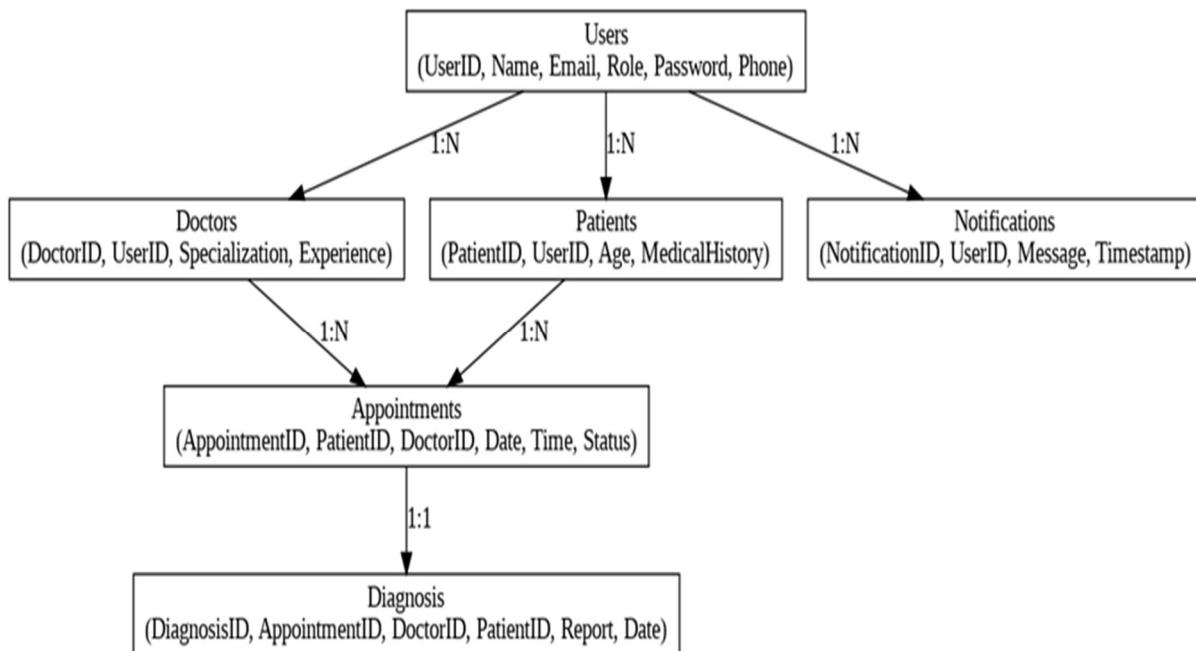
Scenario 3 – Emergency Alerts and Notifications

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:

1. **AWS Account Setup:**
<https://docs.aws.amazon.com/accounts/latest/reference/getting-started.html>
2. **AWS IAM (Identity and Access Management):**
<https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html>
3. **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.

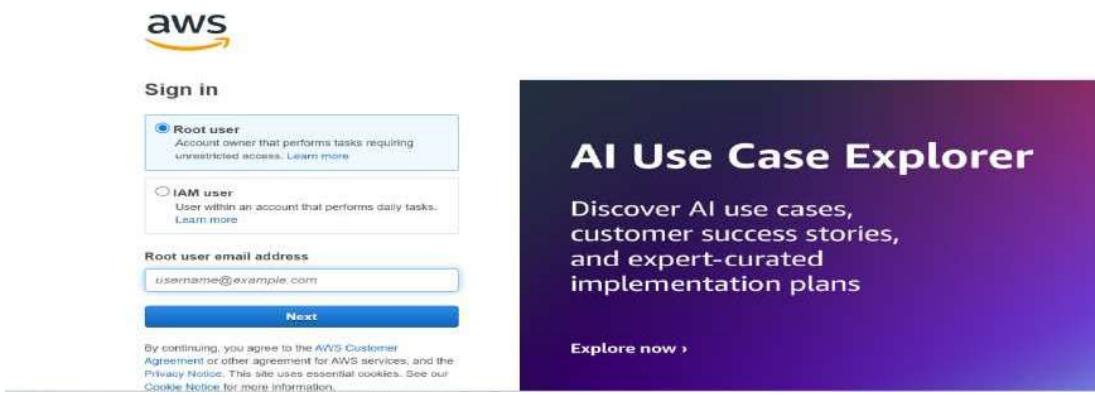
Milestone 1: AWS Account Setup and Login

- **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](#).

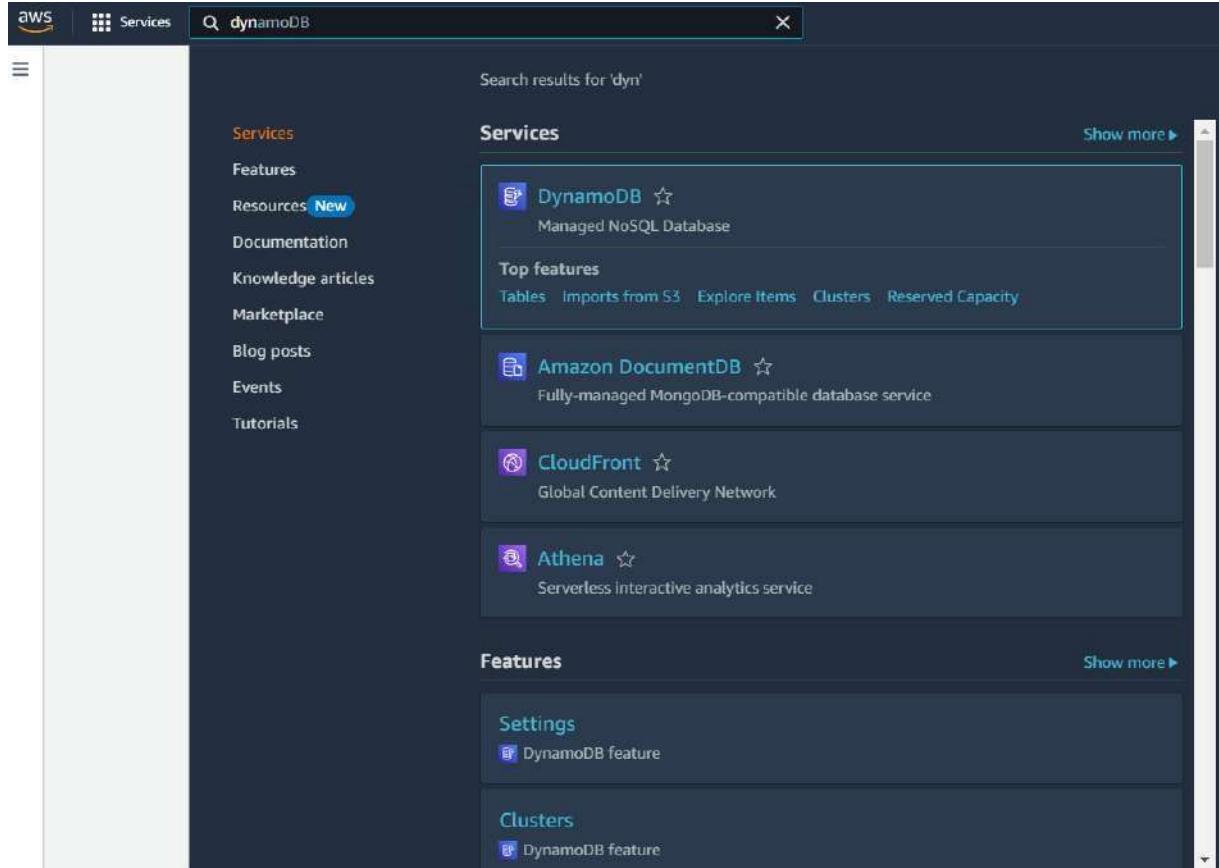


The image contains two parts. On the left is the AWS sign-in interface. It features a 'Sign in' header, a 'Root user' section (selected) with a sub-note about account owners, an 'IAM user' section (unchecked) with a sub-note about users within an account, a 'Root user email address' input field containing 'username@example.com', and a blue 'Next' button. Below the input field is a small note about agreeing to the AWS Customer Agreement and Privacy Notice. On the right is a promotional card for the 'AI Use Case Explorer'. It has a dark purple gradient background and white text. The title is 'AI Use Case Explorer' followed by 'Discover AI use cases, customer success stories, and expert-curated implementation plans'. At the bottom, there's a blue 'Explore now >' button.

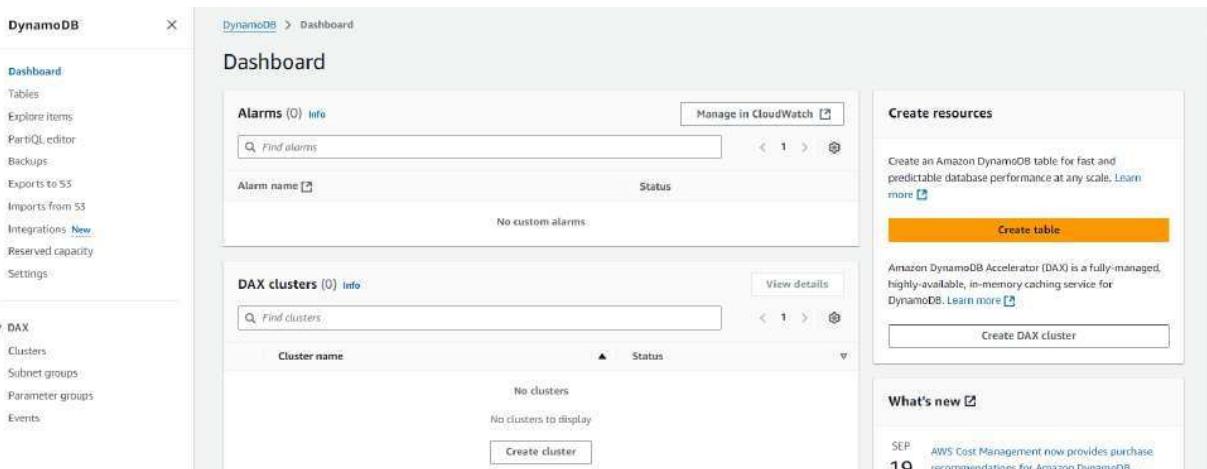
Milestone 2: DynamoDB Database Creation and Setup

- **Activity 2.1: Navigate to the DynamoDB**

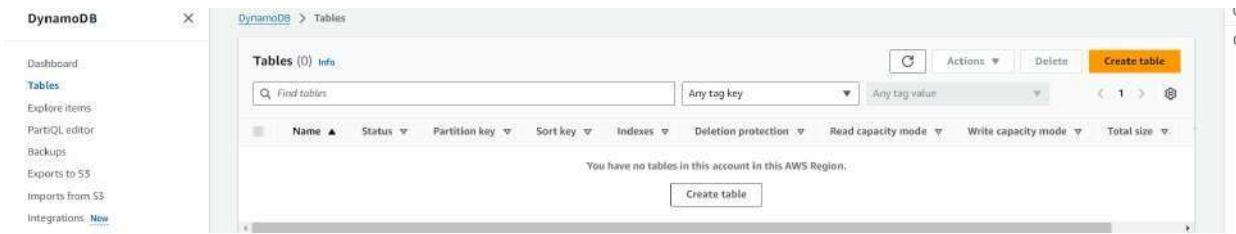
- In the AWS Console, navigate to DynamoDB and click on create tables.



The screenshot shows the AWS Services search results page. The search bar at the top contains 'dynamoDB'. On the left, there is a sidebar with links to Services, Features, Resources (New), Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main content area is titled 'Services' and lists several services: **DynamoDB** (Managed NoSQL Database), **Amazon DocumentDB** (Fully-managed MongoDB-compatible database service), **CloudFront** (Global Content Delivery Network), and **Athena** (Serverless interactive analytics service). Below this, there are sections for 'Features' (Settings, Clusters) and 'Clusters'.

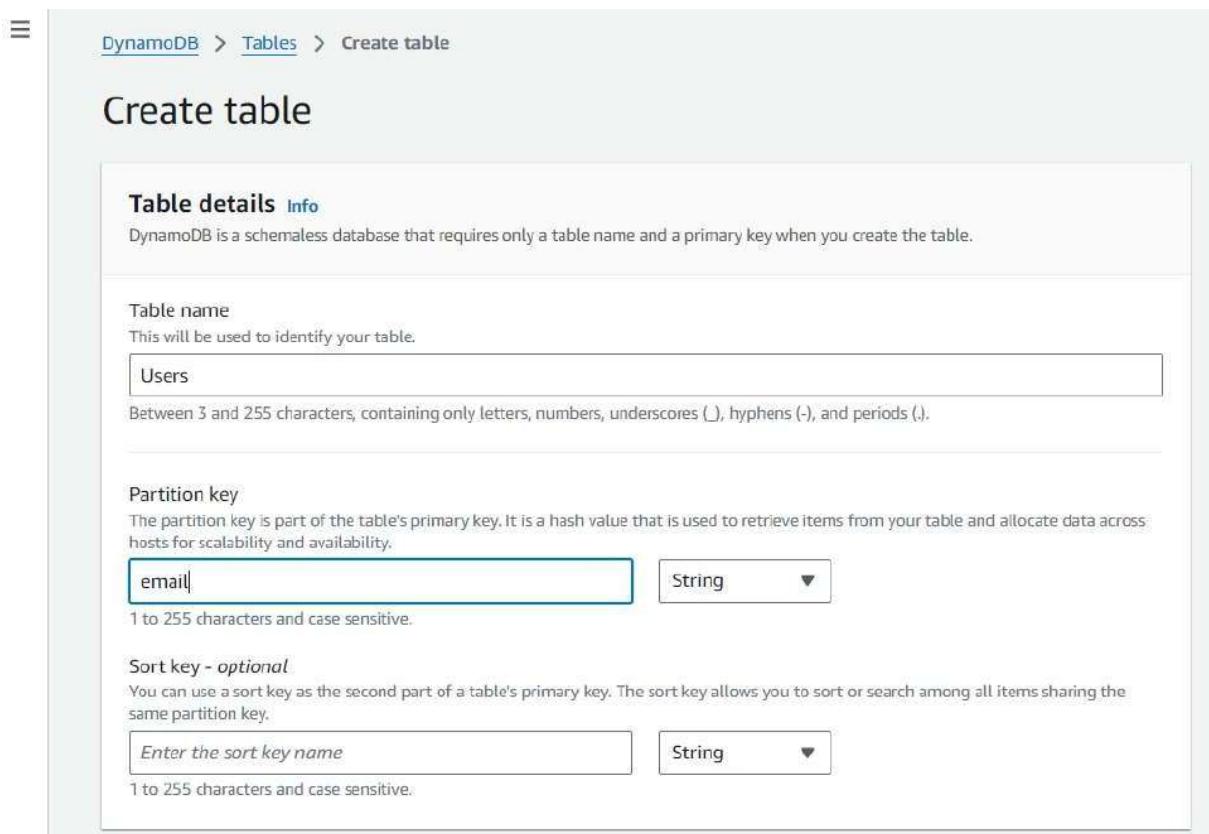


The screenshot shows the DynamoDB Dashboard. The left sidebar includes links for Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations (New), Reserved capacity, and Settings. Under the DAX section, there are links for Clusters, Subnet groups, Parameter groups, and Events. The main dashboard displays sections for **Alarms** (0), **DAX clusters** (0), and a **Create resources** section. The Create resources section includes a 'Create table' button and information about Amazon DynamoDB Accelerator (DAX). There is also a 'What's new' section at the bottom.



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

- Create Users table with partition key "Email" with type String and click on create tables.



The screenshot shows the 'Create table' wizard. At the top, there is a breadcrumb navigation: DynamoDB > Tables > Create table. The main title is 'Create table'. Under 'Table details', there is a sub-section 'Table name' with a note: 'DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.' A text input field contains 'Users'. Below it is a note: 'Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).'. Under 'Partition key', there is a note: '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.' An input field contains 'email' and a dropdown menu shows 'String'. Below it is a note: '1 to 255 characters and case sensitive.' Under 'Sort key - optional', there is a note: '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.' An input field contains 'Enter the sort key name' and a dropdown menu shows 'String'. Below it is a note: '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](#)

The Users table was created successfully.

Tables (1) Info									
	Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size
<input type="checkbox"/>	Users	Active	email (\$)	-	0	<input checked="" type="radio"/> Off	Provisioned (5)	Provisioned (5)	0 bytes

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

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

Create table

Table details inio

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.

Requests

Between 3 and 255 characters, containing only letters, numbers, underscores, and 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

email

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.

Enter the sort key name

String



4 to 255 characters and case sensitive.

Table settings

@ Default settings

Customized settings

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

Tags

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

No tags are associated with the resource.

[Add new tag](#)

You can add 50 more tags.

[Cancel](#)

[Create table](#)

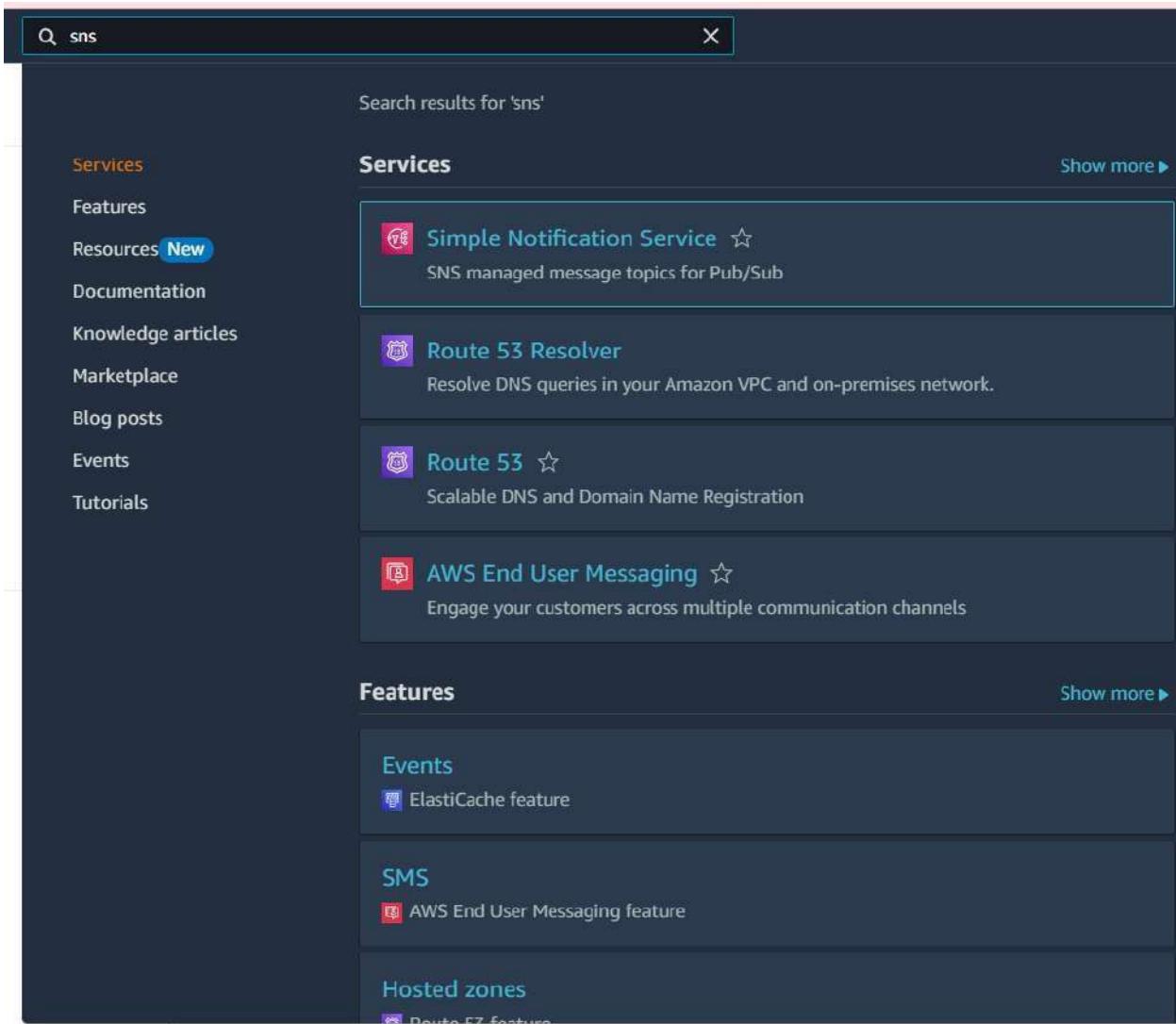
The Requests table was created successfully.

Tables (2) Info									
	Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size
<input type="checkbox"/>	Requests		email (\$)	-	0	<input checked="" type="radio"/> Off	Provisioned (S)	Provisioned (S)	0 bytes
<input type="checkbox"/>	Users		email (\$)	-	0	<input checked="" type="radio"/> Off	Provisioned (S)	Provisioned (S)	0 bytes

Milestone 3: SNS Notification Setup

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

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



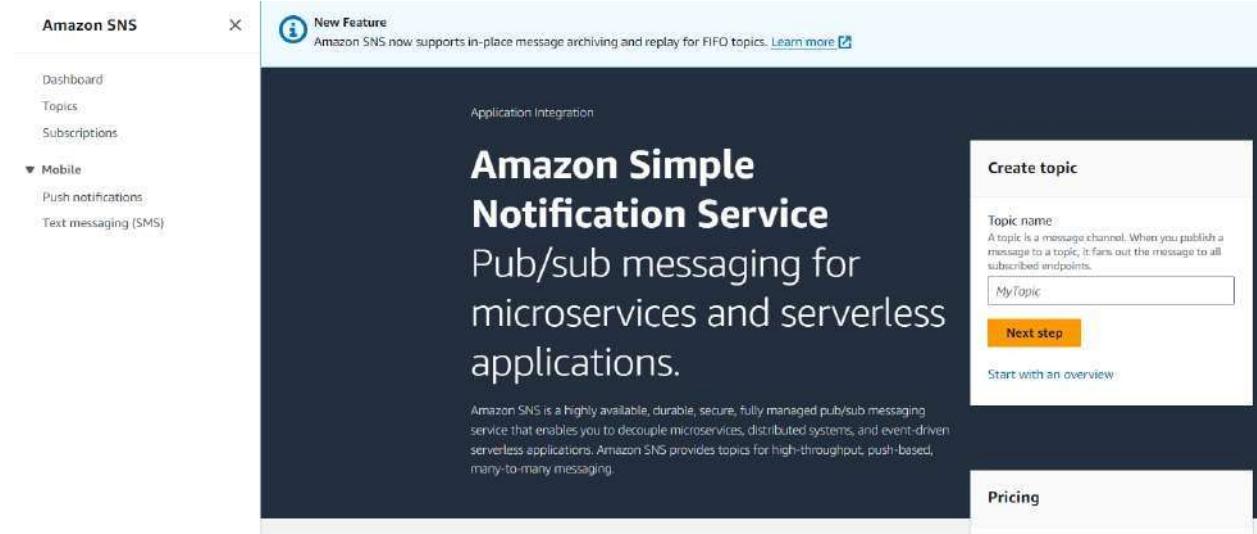
The screenshot shows the AWS search results page with the query 'sns' entered in the search bar. The results are categorized under 'Services' and 'Features'.

Services

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

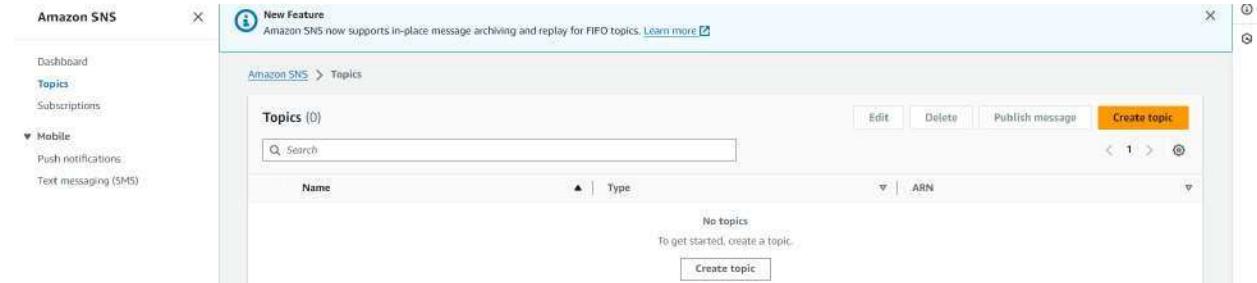
Features

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



The screenshot shows the Amazon Simple Notification Service (SNS) interface. On the left, there's a sidebar with links like Dashboard, Topics, Subscriptions, Mobile (Push notifications, Text messaging (SMS)), and Application integration. The main content area has a heading 'Amazon Simple Notification Service' and a sub-heading 'Pub/sub messaging for microservices and serverless applications.' Below this is a detailed description of the service. To the right, a large callout box titled 'Create topic' contains a 'Topic name' input field with 'MyTopic' typed into it, a 'Next step' button, and a link 'Start with an overview'. At the bottom right of the main content area, there's a 'Pricing' link.

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



The screenshot shows the 'Topics' list page in the Amazon SNS interface. The sidebar on the left is identical to the previous screenshot. The main content area shows a table with one row, indicating 'No topics'. It includes a search bar, edit, delete, publish message, and create topic buttons. A note at the bottom says 'To get started, create a topic.' and features a 'Create topic' button.

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

[Amazon 5N5](#) \ [Topics](#) Create topic

Create topic

Details

Type [trrfo](#)

Topic type cannot be modified after topic is created

FJFO (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

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

Display name - optional [lrifo](#)

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.

Access policy - optional [Info](#)
 This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic.

Data protection policy - optional [Info](#)
 This policy defines which sensitive data to monitor and to prevent from being exchanged via your topic.

Delivery policy (HTTP/S) - optional [Info](#)
 The policy defines how Amazon SNS retries failed deliveries to HTTP/S endpoints. To modify the default settings, expand this section.

Delivery status logging - optional [Info](#)
 These settings configure the logging of message delivery status to CloudWatch Logs.

Tags - optional
 A tag is a metadata label that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track your costs. [Learn more](#)

Active tracing - optional [Info](#)
 Use AWS X-Ray active tracing for this topic to view its traces and service map in Amazon CloudWatch. Additional costs apply.

[Cancel](#) [Create topic](#)

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

Amazon SNS New Feature

Topics Publish message

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

Amazon SNS > Topics > BookRequestNotifications

[Edit](#) [Delete](#) [Publish message](#)

Details

Name:	BookRequestNotifications	Display name:	-
ARN:	arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications	Topic owner:	557690616836
Type:	Standard		

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

Subscriptions (0)

ID	Endpoint	Status	Protocol
No subscriptions found. You don't have any subscriptions to this topic.			

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

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

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN
 X

Protocol
 The type of endpoint to subscribe

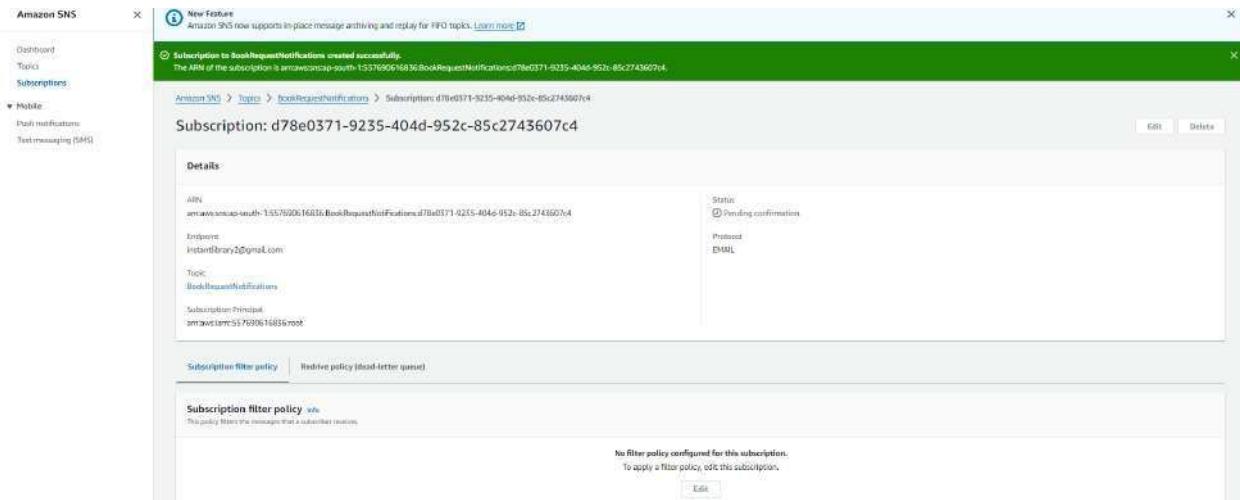
Endpoint
 An email address that can receive notifications from Amazon SNS.

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

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

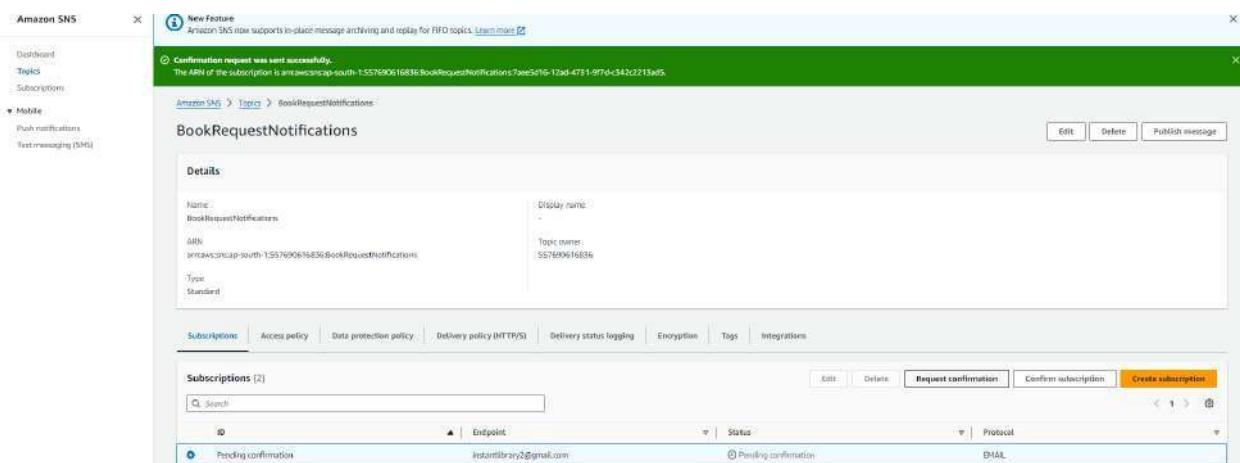
► **Redrive policy (dead-letter queue) - optional** [Info](#)
 Send undeliverable messages to a dead-letter queue.

Cancel
Create subscription



The screenshot shows the AWS SNS 'Subscription' details for a specific subscription. The subscription ARN is `d78e0371-9235-404d-952c-85c2743607c4`. The topic name is `BookRequestNotifications`. The endpoint is `instantlibrary2@gmail.com`. The status is `Pending confirmation`, and the protocol is `EMAIL`. There is no filter policy applied.

- After subscription request for the mail confirmation



The screenshot shows the AWS SNS 'Topic' details for a topic named `BookRequestNotifications`. The ARN is `arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:7ee5d16-12ad-4751-9f7d-c342c2215af5`. The topic type is `Standard`. A single subscription is listed with the ID `Pending confirmation`, endpoint `instantlibrary2@gmail.com`, status `Pending confirmation`, and protocol `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 ×

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

9

You have chosen to subscribe to the topic:

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

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

[Confirm subscription](#)

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

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

You have chosen to subscribe to the topic:

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

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

[Confirm subscription](#)

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



Simple Notification Service

Subscription confirmed!

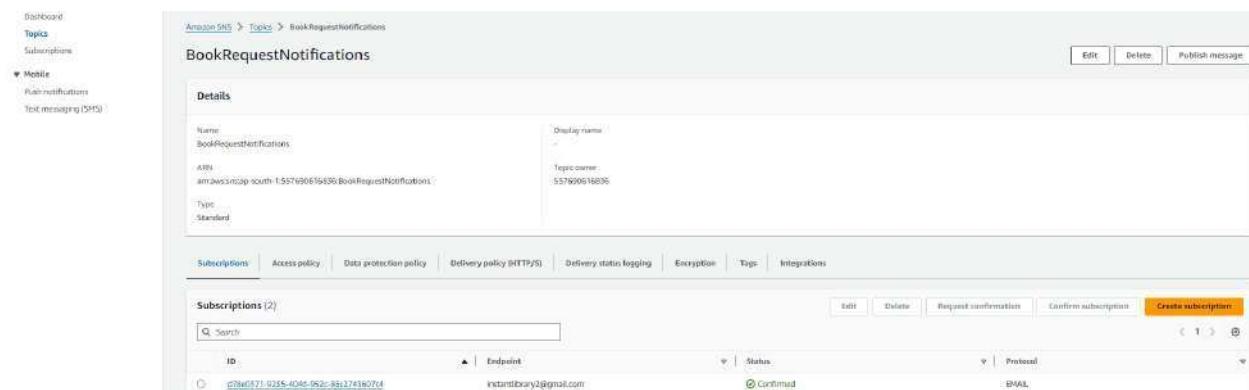
You have successfully subscribed.

Your subscription's id is:

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

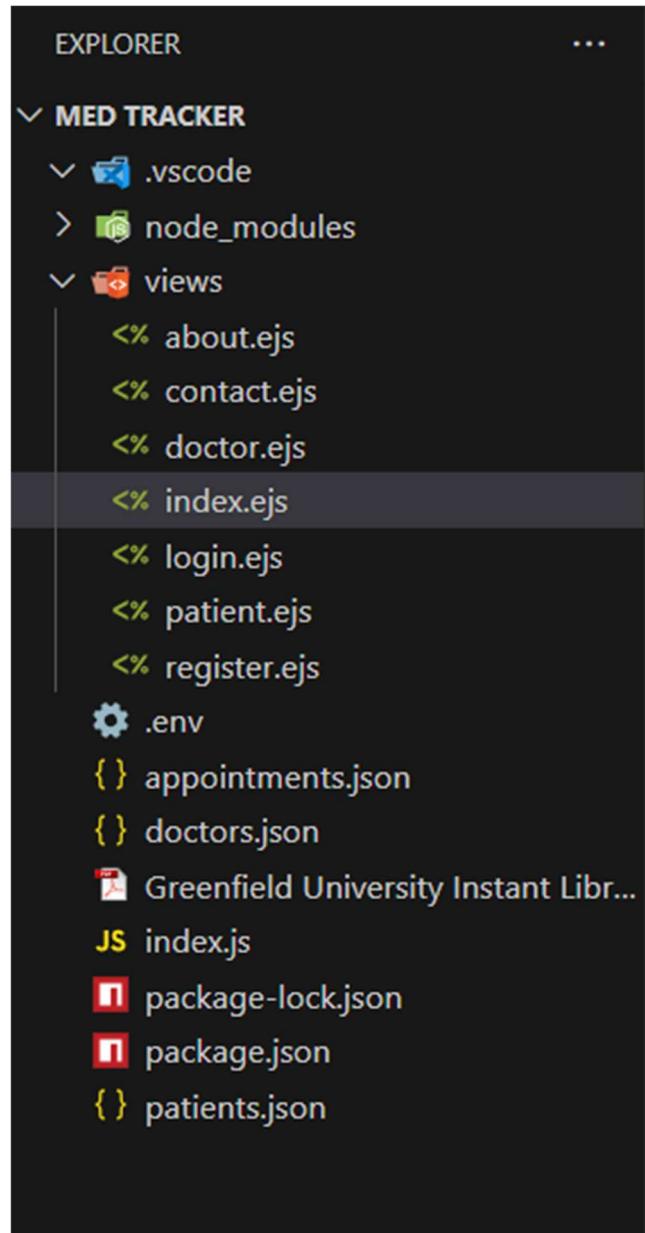
If it was not your intention to subscribe, [click here to unsubscribe](#).

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



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';
10

```

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;
18

```

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:**

```

L59
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:

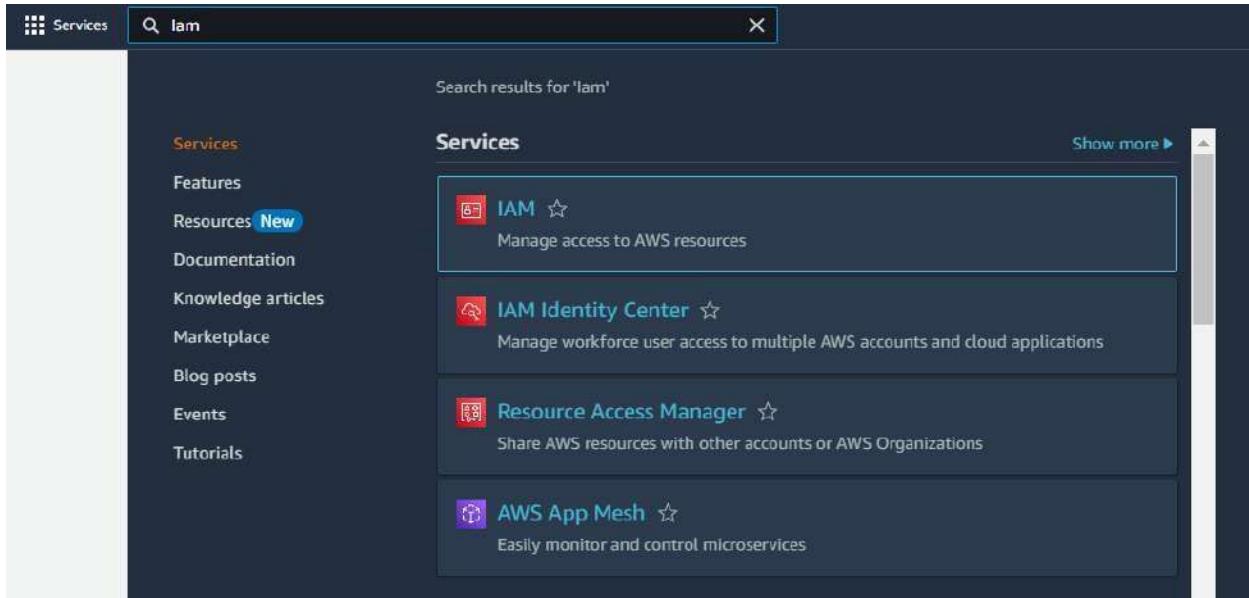
```
app.listen(3000, '0.0.0.0', () => {
  console.log(`Server running at http://localhost:3000`);
  console.log(`Using AWS Region: ${REGION}`);
  console.log(`Patients Table: ${PATIENTS_TABLE}`);
  console.log(`Doctors Table: ${DOCTORS_TABLE}`);
  console.log(`Appointments Table: ${APPOINTMENTS_TABLE}`);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

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.

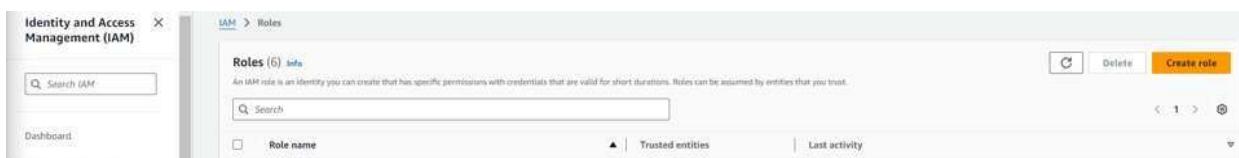
Milestone 5: IAM Role Setup

- **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.



The screenshot shows the AWS Services menu with the search bar set to 'Iam'. Below the search bar, there is a 'Search results for 'iam'' placeholder. On the left, a sidebar lists various services and features under 'Services' and 'Features'. The 'Resources' section is highlighted with a blue background. The main content area displays four service cards: 'IAM' (Manage access to AWS resources), 'IAM Identity Center' (Manage workforce user access to multiple AWS accounts and cloud applications), 'Resource Access Manager' (Share AWS resources with other accounts or AWS Organizations), and 'AWS App Mesh' (Easily monitor and control microservices). The 'IAM' card is currently selected.



The screenshot shows the 'Roles' page within the AWS IAM service. The top navigation bar includes 'Identity and Access Management (IAM)' and 'Dashboard'. The main content area is titled 'Roles (6) Info' and contains a brief description of what an IAM role is. A search bar labeled 'Search IAM:' is present. Below the search bar, there is a table with columns for 'Role name', 'Trusted entities', and 'Last activity'. The first row of the table is visible, showing a role named 'lambda-role'. At the bottom right of the table, there are buttons for 'Create role' and 'Delete'.



Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Add permissions Info

Permissions policies {1/955} Info
Choose one or more policies to attach to your new role.

Filter by Type

Policy name	Type
AmazonDynamoDBFullAccess	AWS managed
AmazonDynamoDBReadOnlyAccess	AWS managed

▶ Set permissions boundary - optional

- **Activity 5.2: Attach Policies.**

Attach the following policies to the role:

- **AmazonDynamoDBFullAccess**: Allows EC2 to perform read/write operations on DynamoDB.
 - **AmazonSNSFullAccess**: Grants EC2 the ability to send notifications via SNS.

SMARTBRIDGE



Add permissions



The screenshot shows the AWS IAM Roles page with the role 'sns_Dynamodb_role'. The 'Summary' tab is selected, displaying details like creation date (October 13, 2024), last activity (6 days ago), ARN (arn:aws:iam::557606168387:role/sns_Dynamodb_role), and maximum session duration (1 hour). The 'Instance profile ARN' is also listed. Below the summary, there are tabs for 'Permissions', 'Trust relationships', 'Tags', 'List Accessed', and 'Revoke sessions'. The 'Permissions' tab is active, showing a table of attached policies. The table includes columns for Policy name, Type, and Attached entities. Policies listed include 'AmazonDynamoDBFullAccess' and 'AmazonDynamoDBAccess'.

Milestone 6: EC2 Instance Setup

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

The screenshot shows a GitHub repository named 'MedTrack' (Public). The repository has 1 branch and 0 tags. The main branch is selected. The commit history is shown, starting with an initial commit by 'Varun-panchakarla' (index.js) at 'ca95df5 · 3 days ago'. Other commits include 'views' (Add files via upload · 4 days ago), '.env' (· 3 days ago), 'README.md' (Initial commit · 4 days ago), 'appointments.json' (Add files via upload · 4 days ago), 'doctors.json' (Add files via upload · 4 days ago), 'index.js' (index.js · 3 days ago), 'package-lock.json' (Add files via upload · 4 days ago), 'package.json' (Add files via upload · 4 days ago), and 'patients.json' (Add files via upload · 4 days ago).

Local Codespaces

Clone ?

HTTPS SSH GitHub CLI

<https://github.com/Varun-panchakarla/MedTrack.git> 

Clone using the web URL.

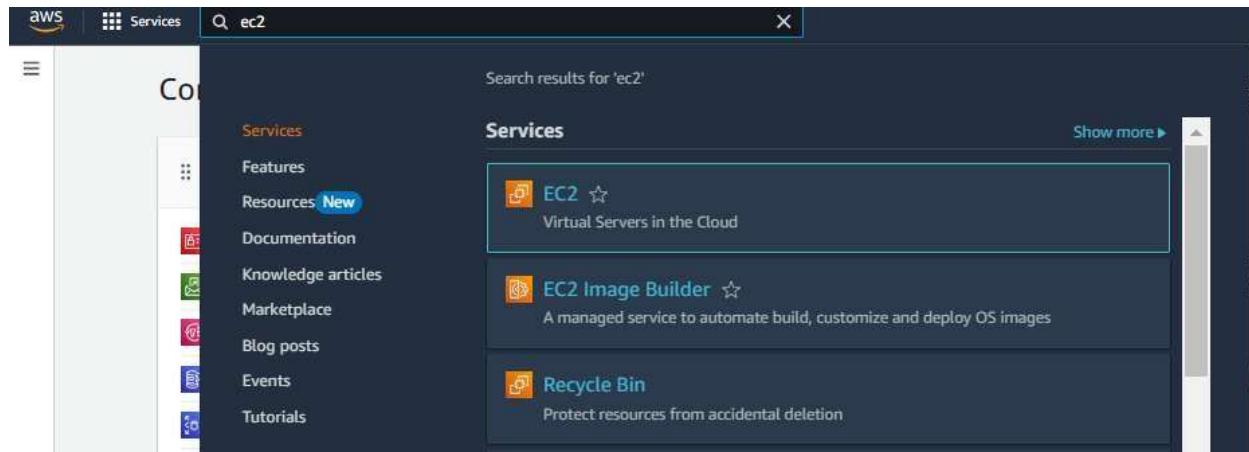
 Open with GitHub Desktop

 Download ZIP

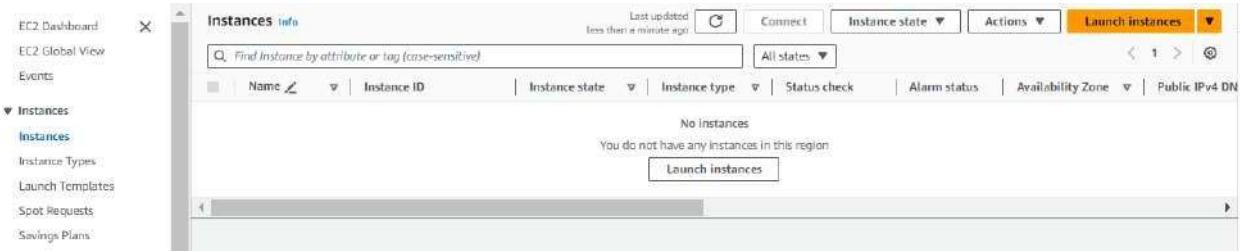
- **Activity 6.1: Launch an EC2 instance to host the Flask application.**

- **Launch EC2 Instance**

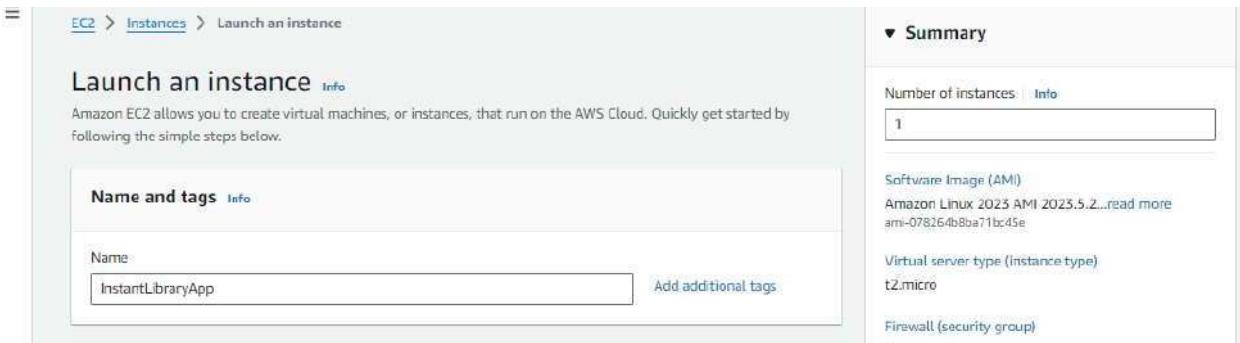
- In the AWS Console, navigate to EC2 and launch a new instance.



- Click on Launch instance to launch EC2 instance

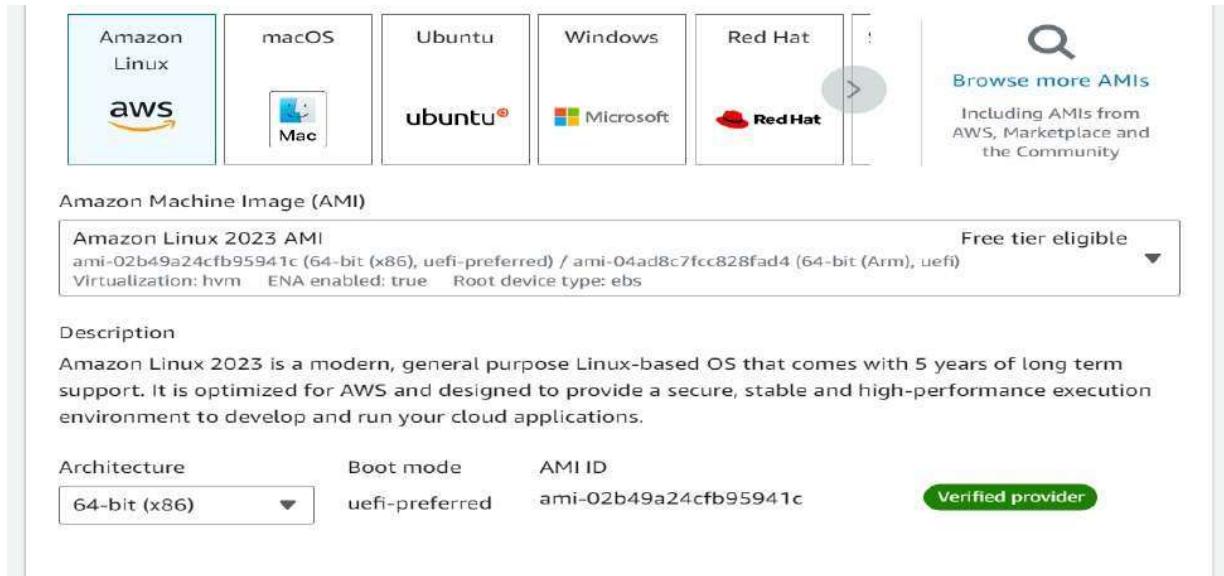


The screenshot shows the AWS EC2 Instances dashboard. On the left, there's a sidebar with options like EC2 Dashboard, EC2 Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, and Savings Plans. The main area is titled 'Instances info' and shows a search bar with 'Find Instance by attribute or tag (case-sensitive)' and a dropdown for 'All states'. Below the search bar, there are filters for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DN. A message says 'No instances' and 'You do not have any instances in this region'. At the bottom right of the main area is a large 'Launch instances' button.



The screenshot shows the 'Launch an instance' wizard. The first step, 'Launch an instance', is displayed. It has a sub-section 'Name and tags' where the name 'InstantLibraryApp' is entered. To the right, there's a 'Summary' section showing 'Number of instances: 1'. Below it are fields for 'Software Image (AMI)', 'Amazon Linux 2023 AMI 2023.5.2...', 'Virtual server type (instance type)', 't2.micro', and 'Firewall (security group)'. A 'Next Step' button is at the bottom right.

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



The screenshot shows the 'Amazon Machine Image (AMI)' selection screen. It lists several AMI options: Amazon Linux, macOS, Ubuntu, Windows, and Red Hat. A 'Browse more AMIs' link is available. Below this, the 'Amazon Linux 2023 AMI' is selected, showing its details: 'Free tier eligible', 'ami-02b49a24cfb95941c (64-bit (x86), uefi-preferred) / ami-04ad8c7fcc828fad4 (64-bit (Arm), uefi)', 'Virtualization: hvm', 'ENA enabled: true', and 'Root device type: ebs'. The 'Description' section notes that Amazon Linux 2023 is a modern, general purpose Linux-based OS with 5 years of long term support. The 'Architecture' dropdown is set to '64-bit (x86)', 'Boot mode' is 'uefi-preferred', and the 'AMI ID' is 'ami-02b49a24cfb95941c'. A 'Verified provider' badge is present.

- Create and download the key pair for Server access.

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

Instance type

t2.micro	Free tier eligible
Family: t2 · 1 vCPU · 1 GiB Memory · Current generation: true	
On-Demand Linux base pricing: 0.0124 USD per Hour	
On-Demand Windows base pricing: 0.017 USD per Hour	
On-Demand RHEL base pricing: 0.0268 USD per Hour	
On-Demand SUSE base pricing: 0.0124 USD per Hour	

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 [!\[\]\(c97a6f80a417c2fb26707e3f4823e597_img.jpg\) Create new key pair](#)

Create key pair

Key pair name
 Key pairs allow you to connect to your instance securely.
 The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type:

<input checked="" type="radio"/> RSA RSA encrypted private and public key pair	<input type="radio"/> ED25519 ED25519 encrypted private and public key pair
---	--

Private key file format:

<input checked="" type="radio"/> .pem For use with OpenSSH	<input type="radio"/> .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: t2.micro | **Number of instances**: 1

Software Image: Amazon Linux 2023 (ami-078264b8ba71b)

Key pair (login): Your key pair hasn't been created yet.

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 GB 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 (default) ▾ 

Subnet Info

No preference ▾ 

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 

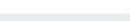
Security group name - required

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			
			
Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	
ssh	TCP	22	
Source type <small>Info</small>	Source <small>Info</small>	Description - optional <small>Info</small>	
Anywhere		e.g. SSH for admin desktop	
	0.0.0.0/0 		
			
Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	
HTTP	TCP	80	
Source type <small>Info</small>	Source <small>Info</small>	Description - optional <small>Info</small>	
Custom		e.g. SSH for admin desktop	
	0.0.0.0/0 		
			
Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	
Custom TCP	TCP	5000	
Source type <small>Info</small>	Source <small>Info</small>	Description - optional <small>Info</small>	
Custom		e.g. SSH for admin desktop	
	0.0.0.0/0 		
			

EC2 > Launch an Instance

Success
Successfully launched instance i-001861022fbac290

Launch log

Next Steps

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

1 2 3 4

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

[View all instances](#)

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

Instances (1/2) Info

Last updated less than a minute ago

Find instance by attribute or tag (case-sensitive)

All states

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

EC2 > Instances > i-001861022fbac290

Instance summary for i-001861022fbac290 (InstantLibraryApp)

Updated less than a minute ago

Instance ID i-001861022fbac290	Public IPv4 address -	Private IPv4 address 172.31.3.5
IPv6 address -	Public IPv4 DNS -	Public IPv4 IP -
Hostname type IP name: ip-172-51-5-5.ap-south-1.compute.internal	Instance state stopped	Elastic IP addresses -
Associate private resource DNS name IPv4 (A)	Private IP DNS name (IPv4 only) ip-172-31-5-5.ap-south-1.compute.internal	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto-assigned IP address -	Instance type t2.micro	Auto Scaling Group name -
IAM Role sts_Dynamodo_role	VPC ID vpc-03cc07bf13ad211	
Required	Subnet ID subnet-0d9fa1444a0cc9a9	
	Instance ARN arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbac290	

[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	172.31.3.5
IPv6 address	-	Instance state	Stopped
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
Answer private resource DNS name	-	Instance type	t2.micro
IPv4 (A)	-	VPC ID	vpc-03cd7b6f19dd7211
Auto-assigned IP address	-	Subnet ID	subnet-029fa5144480cc09
IAM Role	sns_Dynamodb_role	Instance ARN	arn:aws:ec2:ap-south-1:1557690616836:instance/i-001861022fbcac290
IMDv2	Required		

Actions ▾

- Connect
- Instance state ▾
- Actions ▾
- Connect
- Manage instance state
- Instance settings
- Networking
- Security**
- Change security groups
- Get Windows password
- Image and templates
- Modify IAM role**
- Monitor and troubleshoot

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

Modify IAM role [Info](#)

Attach an IAM role to your instance.

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

IAM role

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

[▼](#) [C](#) [Create new IAM role](#)

[Cancel](#) [Update IAM role](#)

- Now connect the EC2 with the files

Connecction Instances , =

EC2 Instance Connect Session Manager EC2 serial console

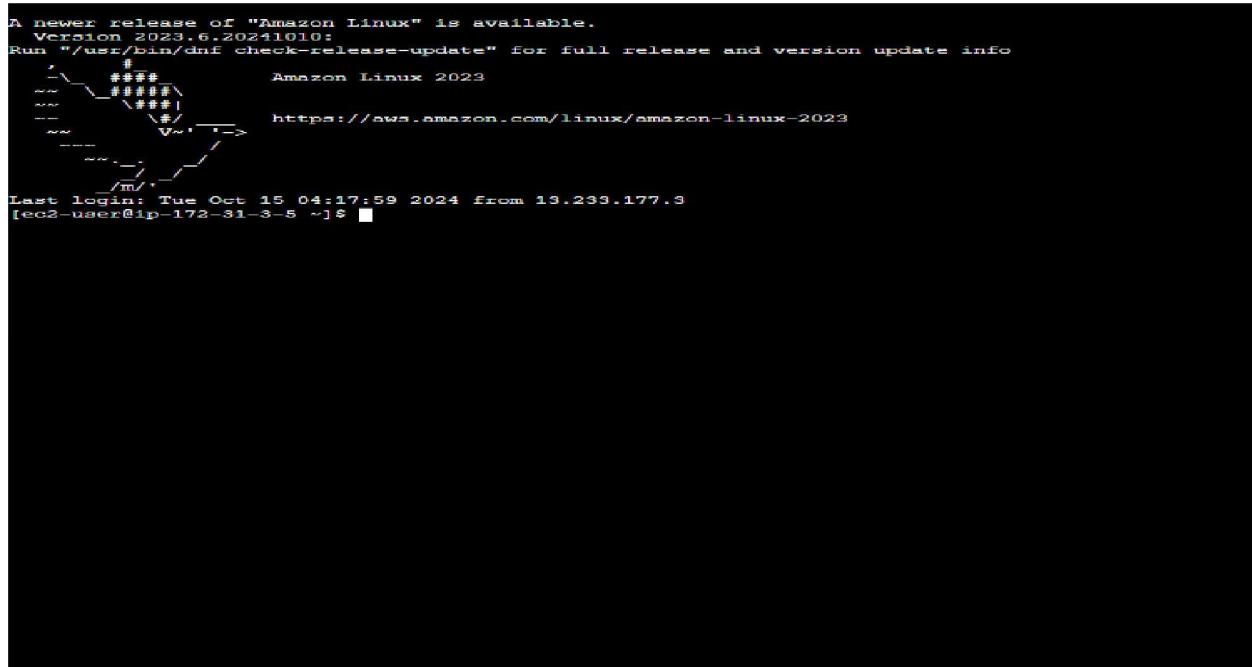
⚠ Port 22 (SSH) is open to all IPv4 addresses
Port 22 (SSH) is currently open to all IPv4 addresses, indicated by **0.0.0.0/0** in the inbound rule in your security group. For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: 13.233.177.0/29.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

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

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

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
[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 install
Note: 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:

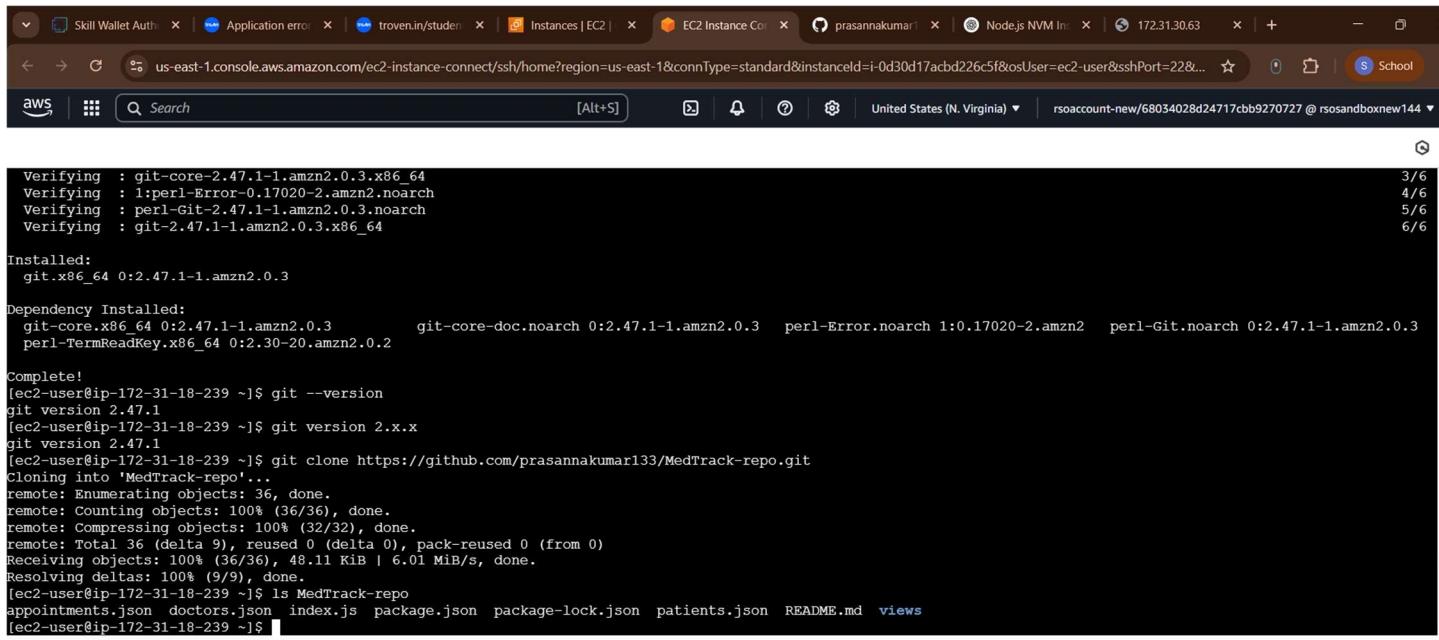
Run the Flask Application

```
# 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
```



```

Verifying : git-core-2.47.1-1.amzn2.0.3.x86_64
Verifying : 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 ~]$ 

```

i-0d30d17acbd226c5f (medtrack-server)

Public IPs: 52.91.51.94 Private IPs: 172.31.18.239



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



Patients Table: PatientsTable
Doctors Table: DoctorsTable
Appointments Table: Appointments

[ec2-user@ip-172-31-20-85 MedTrack]\$ sudo ufw status
sudo: ufw: command not found

[ec2-user@ip-172-31-20-85 MedTrack]\$ sudo ufw allow 3000/tcp
sudo: ufw: command not found

[ec2-user@ip-172-31-20-85 MedTrack]\$ sudo ufw reload
sudo: ufw: command not found

[ec2-user@ip-172-31-20-85 MedTrack]\$ node index.js
[dotenv@17.0.1] injecting env (7) from .env - [tip] encrypt with dotenvx: https://dotenvx.com
(node:6961) 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: Appointments

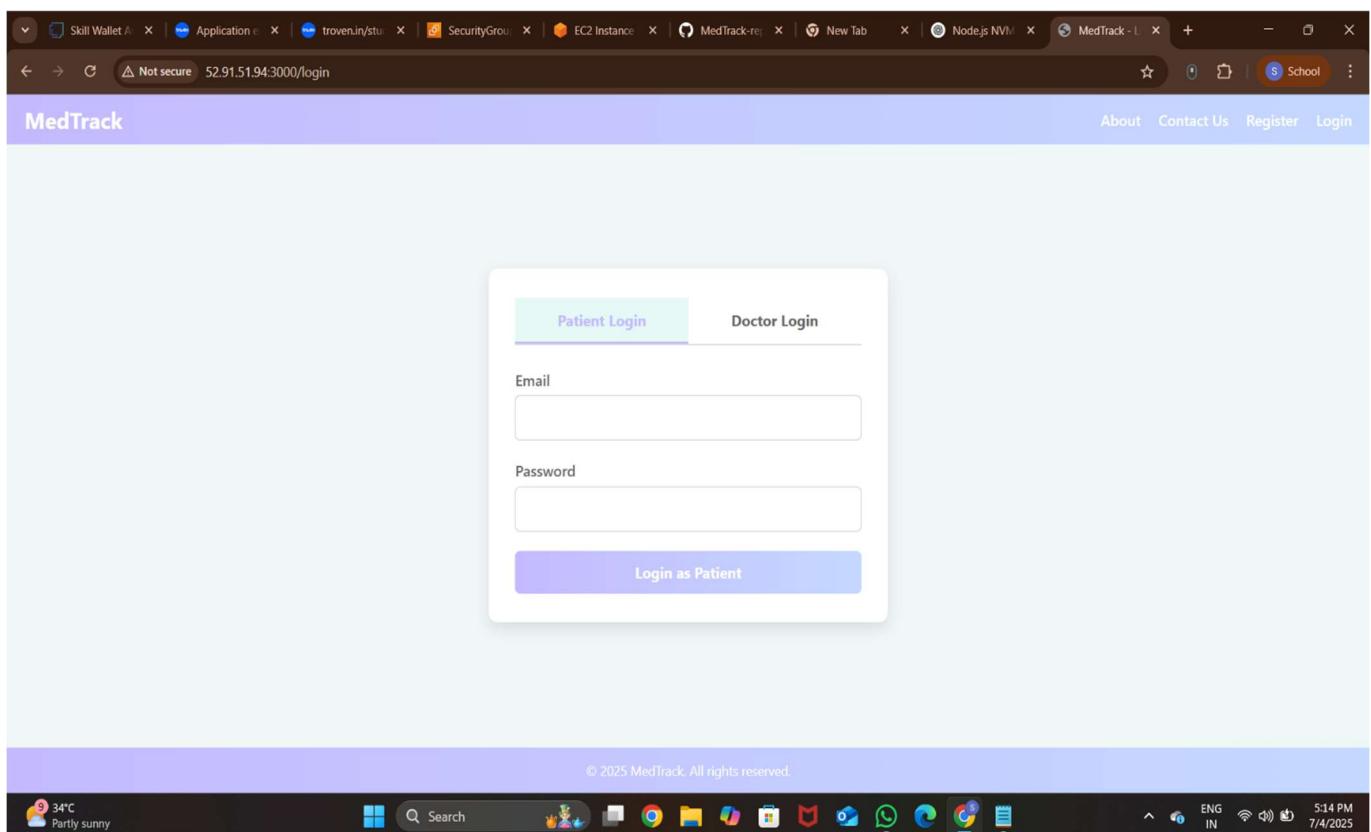
i-09920a49e73472f79 (medtrack-server)
PublicIPs: 54.167.78.35 PrivateIPs: 172.31.20.85

Access the website through:
Public IPs: <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:



Register Page:



Skill Wallet A | Application e | troven.in/stu | SecurityGrou | EC2 Instance | MedTrack-reg | New Tab | Nodejs NVM | MedTrack - R | School

Not secure 52.91.51.94:3000/register

MedTrack About Contact Us Login

Create Your Account

Patient Registration Doctor Registration

First Name

Last Name

Age

Gender Select

Email Address

Phone Number Address

34°C Partly sunny Search ENG IN 5:14 PM 7/4/2025

Home page:

MedTrack Home About Login Register

Empowering Doctor-Patient Communication

Secure, efficient healthcare management at your fingertips. Connecting patients and providers for better healthcare outcomes.

Book an Appointment

Features Designed For You

Experience seamless healthcare management with our comprehensive set of features

About Us page:

About Us

Med Tracker is dedicated to helping individuals and families manage their medications safely and efficiently. Our mission is to provide a simple, secure, and reliable platform for tracking prescriptions, reminders, and health information.

Founded by a team of healthcare and technology professionals, we believe in empowering users with tools that make medication management stress-free and accessible on any device.

ContactUs :

Contact Us

Your Name**Your Email****Subject****Message****Send Message**

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.