

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

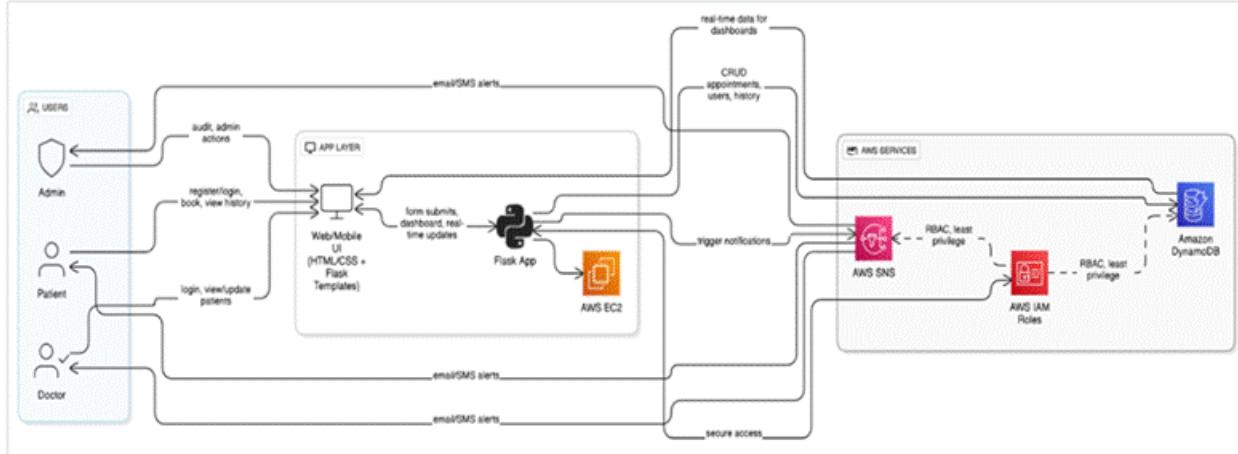
A patient logs into the MedTrack web app, which is hosted on Amazon EC2 instances. They schedule an appointment, and the booking details are saved in Amazon DynamoDB, enabling fast retrieval and updates. Immediately, AWS SNS sends a confirmation SMS and email to the patient with appointment details.

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.

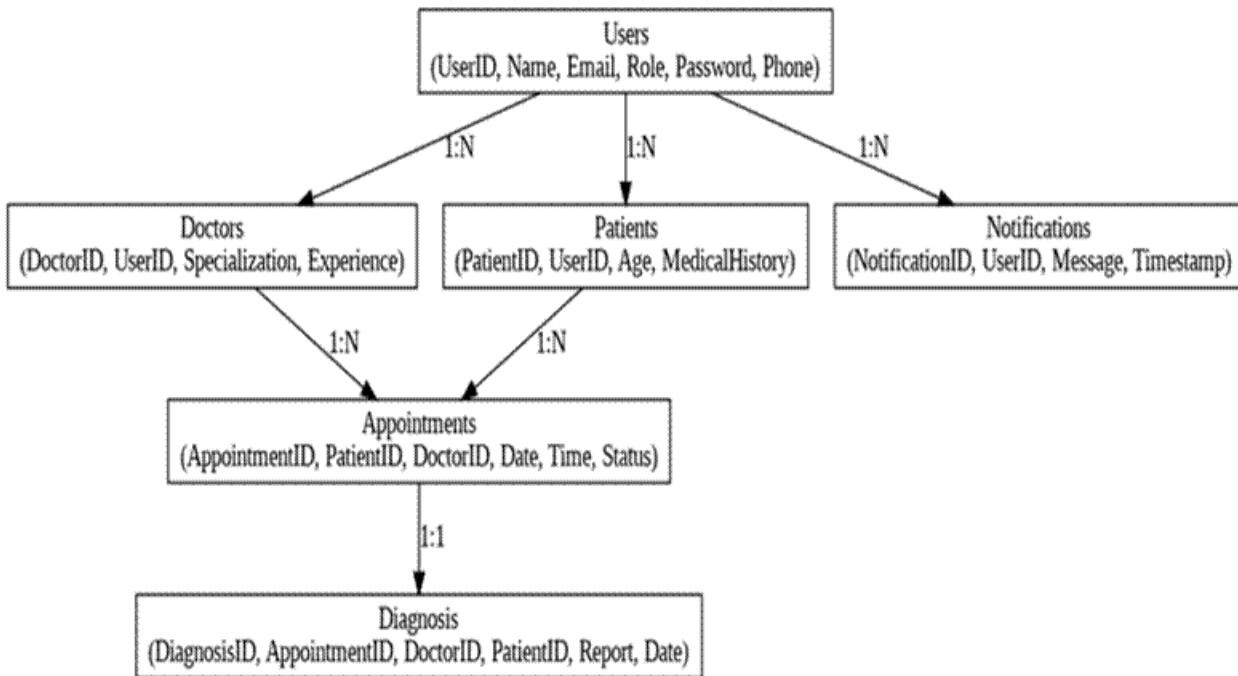
Scenario 3 – Emergency Alerts and Notifications

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.

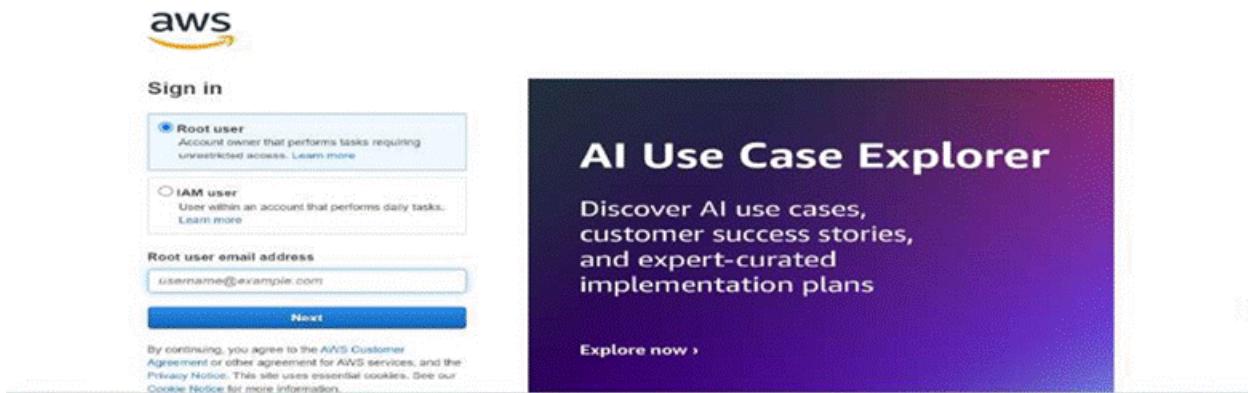
Milestone 1: AWS Account Setup and Login

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

- i. Sign up for an AWS account and configure billing settings.
- ii.

b. Activity 1.2: Log in to the AWS Management Console

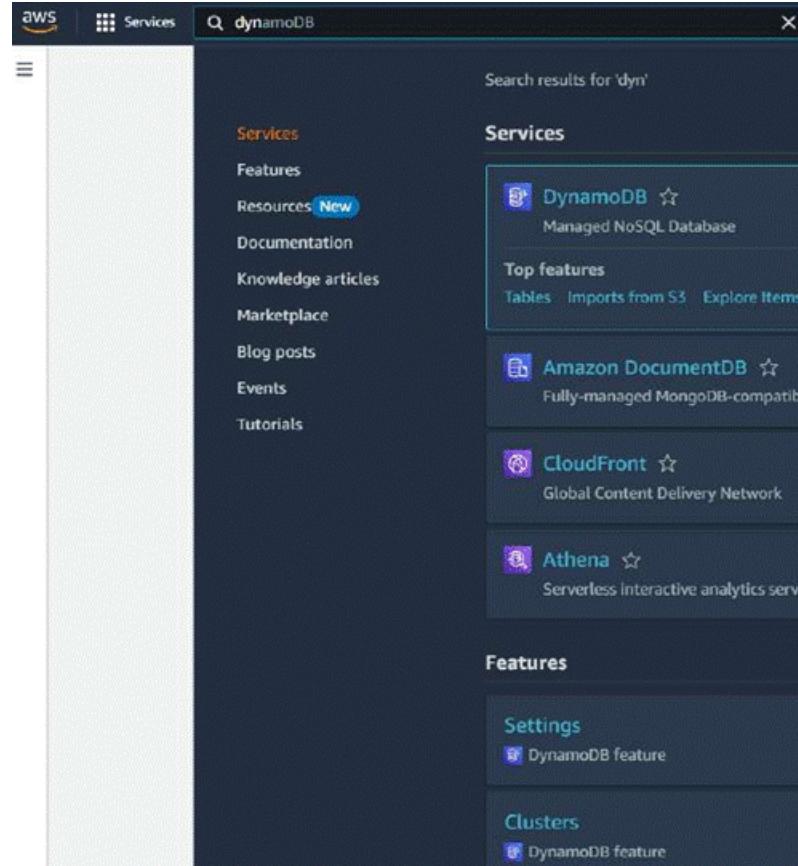
- i. After setting up your account, log in to the [AWS Management Console](#).



Milestone 2: DynamoDB Database Creation and Setup

c. Activity 2.1: Navigate to the DynamoDB

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



ii.

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', 'Reserved capacity', and 'Settings'. A collapsed section for 'DAX' contains 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main dashboard area has two sections: 'Alarms (0)' and 'DAX clusters (0)'. Both sections have search bars, status indicators, and buttons for 'Manage in CloudWatch' and 'Create cluster'. To the right, there is a 'Create resources' section for creating a new DynamoDB table, a 'Create DAX cluster' button, and a 'What's new' section with a note about AWS Cost Management providing purchase recommendations for Amazon DynamoDB.

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

DynamoDB > Tables > Create table

Create table

Table details Info

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

Table name
 This will be used to identify your table.
 Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.)

Partition key
 The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.
 String ▾
 1 to 255 characters and case sensitive.

Sort key - optional
 You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.
 String ▾
 1 to 255 characters and case sensitive.

The Users table was created successfully.

DynamoDB > Tables

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

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

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**

iii.

" DynamoDB \ Tables } Create table
 retrieve items from your table and allocated data across
 email

1 to 255 characters and case sensitive.

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
 Enter the sort key name

same partition key.

4 to 255 characters and case sensitive.

[DynamoDB \ Tables} Create table](#)

Createtable

Table details info

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

Tablename

This will be used to identify your table.

Smart Internz

Requests

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

email

1 to 255 characters and case sensitive.

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

Enter the sort key name

same partition key.

4 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](#)

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

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

The screenshot shows the AWS search interface 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 Features

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

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

Application Integration

Amazon Simple Notification Service

Pub/sub messaging for microservices and serverless applications.

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

Create topic

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

MyTopic

Next step

Pricing

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

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

Amazon SNS > Topics

Name	Type	ARN

No topics

To get started, create a topic.

Create topic

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

[Amazon SNS](#) \ [Topics](#) [Createtopic](#)

Create topic

Details

Type [trrfo](#)

Topic type cannot be modified after topic is created

Smart Internz

○ FJFO (first-in, first-out)

1. Strictly-preserved message ordering
2. Exactly-once message delivery
3. High throughput, up to 1000 publishes/second
4. Subscription protocol: SOS

@ Standard

Best-effort message ordering at-least once message delivery

Highest throughput in publishes/second Subscription protocols: SQS, Lambda, HTTP, SIS, email, mobile application endpoints

Name

BookRequestNotificationDr6

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

Display name - optional Irrfo

To use this topic with 5M5 subscriptions, enter a display name. Only the first 10 characters are displayed in an SIS message.

Maximum 100 characters.

1.

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.

Create

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

ID	Endpoint	Status	Protocol
No subscriptions found.			

No subscriptions found.
You don't have any subscriptions for this topic.

Create

V.

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

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

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

Protocol
The type of endpoint to subscribe
Email

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

Info 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.

The screenshot shows the AWS SNS console. In the left sidebar, under the 'Subscriptions' section, there is a single item: 'Subscription to BookRequestNotifications created successfully'. The main content area displays the details of this subscription:

- Subscription:** d78e0371-9235-404d-952c-85c2743607c4
- Details:**
 - Subject:** [aws.amazon sns results - 1:037100014631 BookRequestNotifications/d78e0371-9235-404d-952c-85c2743607c4](#)
 - Endpoint:** instantlibrary2@gmail.com
 - Topic:** BookRequestNotifications
 - Subscription Protocol:** email
- Subscription Filter policy:** No filter policy (match all topics)
- Subscription Filter policy (info):** No filter policy configured for this subscription. To apply a filter policy, edit this subscription.

ii. After subscription request for the mail confirmation

The screenshot shows the AWS SNS console. In the left sidebar, under the 'Topics' section, there is one topic named 'BookRequestNotifications'. The main content area displays the details of this topic:

- Details:**
 - Name:** BookRequestNotifications
 - ARN:** arn:aws:sns:ap-south-1:557690616436:BookRequestNotifications:2ae5d76-13ed-4791-91a4-c343c2213e05
 - Type:** Standard
- Subscriptions (2):**

ID	Endpoint	Status	Protocol
Pending confirmation	instantlibrary2@gmail.com	Pending confirmation	EMAIL

iii. Navigate to the subscribed Email account and Click on the confirmsubscription 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](#).

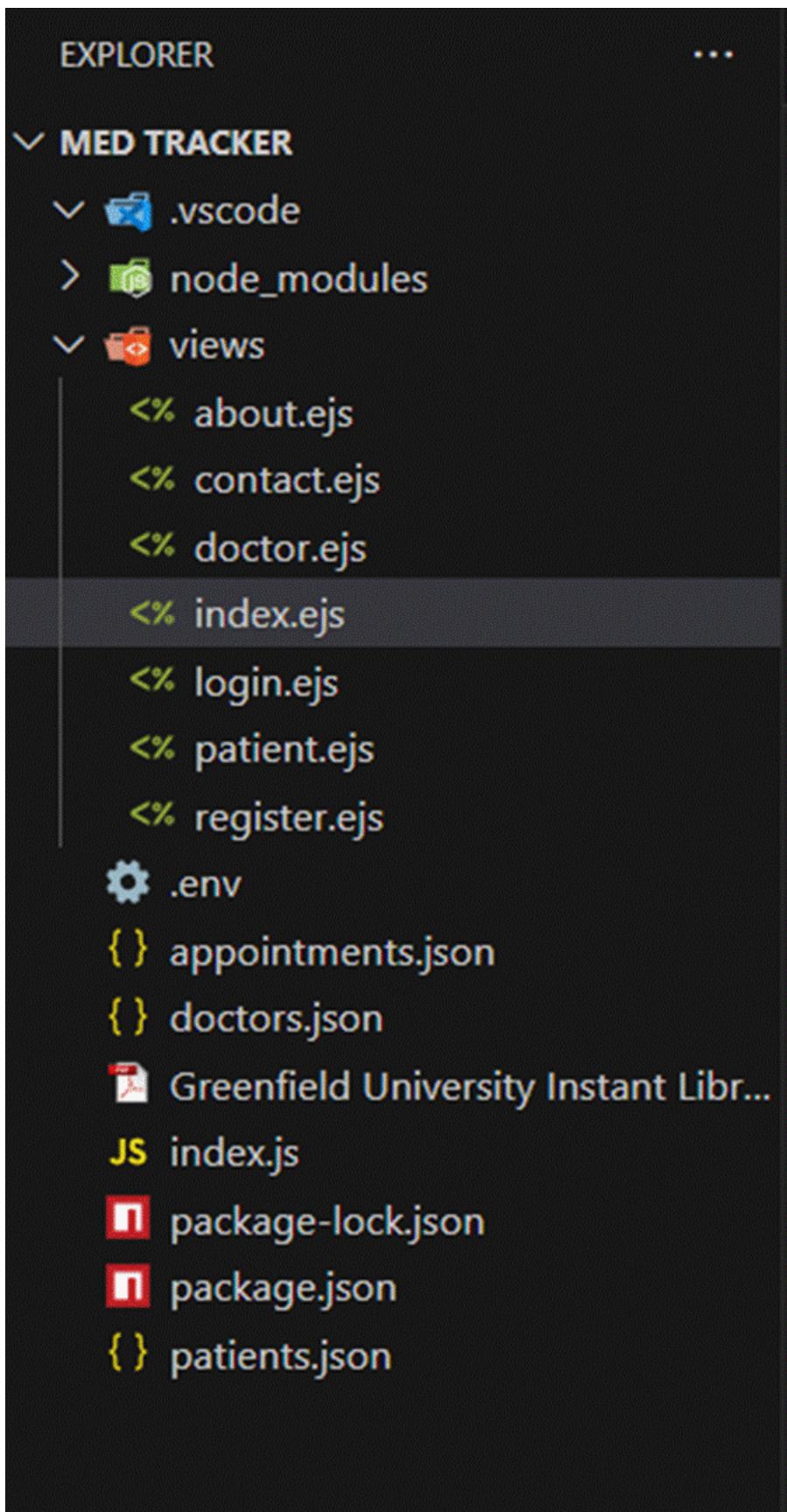
The screenshot shows the AWS SNS console with the following details:

- Topic Details:** Name: BookRequestNotifications, ARN: arn:aws:sns:ap-south-1:557600616050:BookRequestNotifications, Type: Standard.
- Subscriptions:** There are two confirmed subscriptions listed:
 - ID: 0260321, Endpoint: test@library2@gmail.com
 - ID: 0260322, Endpoint: test@library3@gmail.com
- Actions:** Top right buttons include Edit, Delete, and Publish message.

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

Milestone 4: Backend Development and Application Setup

- g. 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 :

h. Node js (Express) Initialization

```
JS index.js  X  .env  <x 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.

```
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
```

i. 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 DynamoDBresource for the ap-south-1 regionand set up access to the Users and Requests tables for storing user details and book requests.

j. 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.

k. Routes for Web Pages

1. 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 redirectusers to the register page when they access the base URL.

a. 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

2. 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 usingBcrypt, update the login countonsuccessful authentication, and redirect users to the home page

1. Home, E-book buttons andsubject 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.

a. 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 confirmsubmission with a success message.

Exit Route:

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

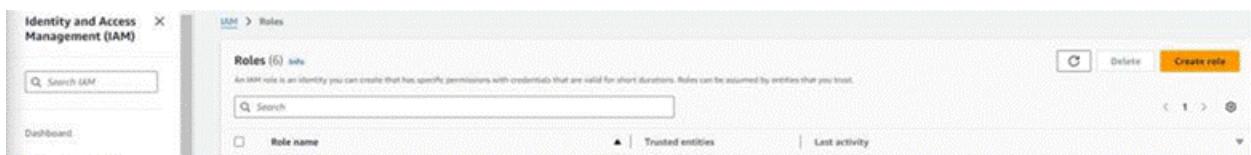
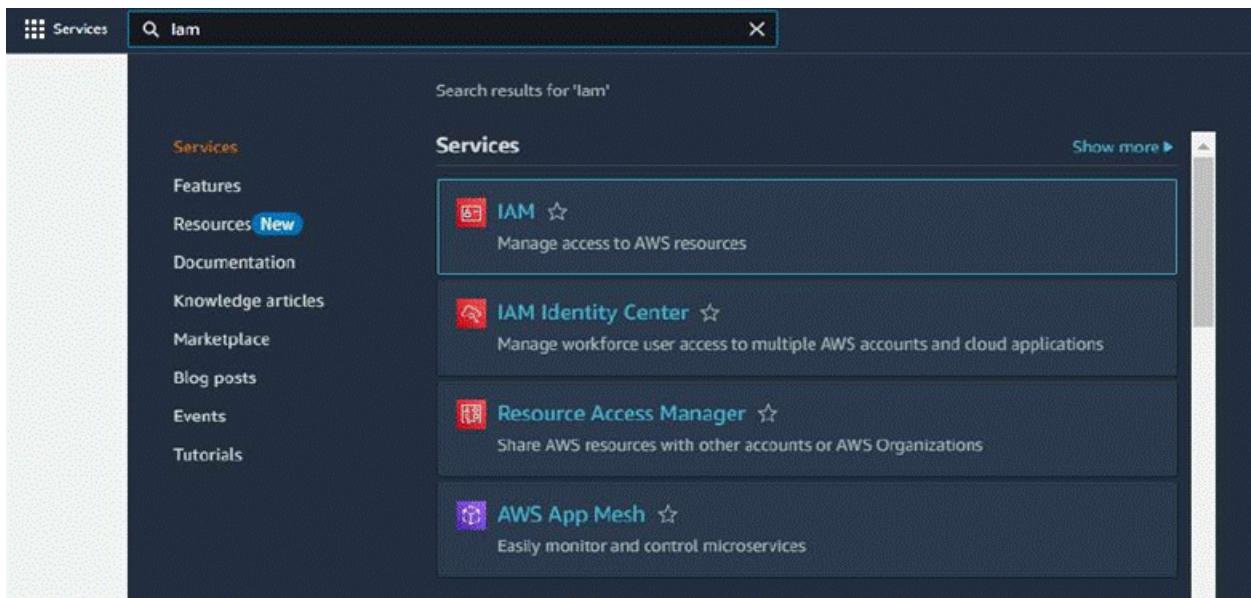
Description: define /exitroute to render the exit.html page when the user chooses to leave or close the application.

Deployment Code:

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

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



3. Activity 5.2: Attach Policies.

Attach the following policies to the role:

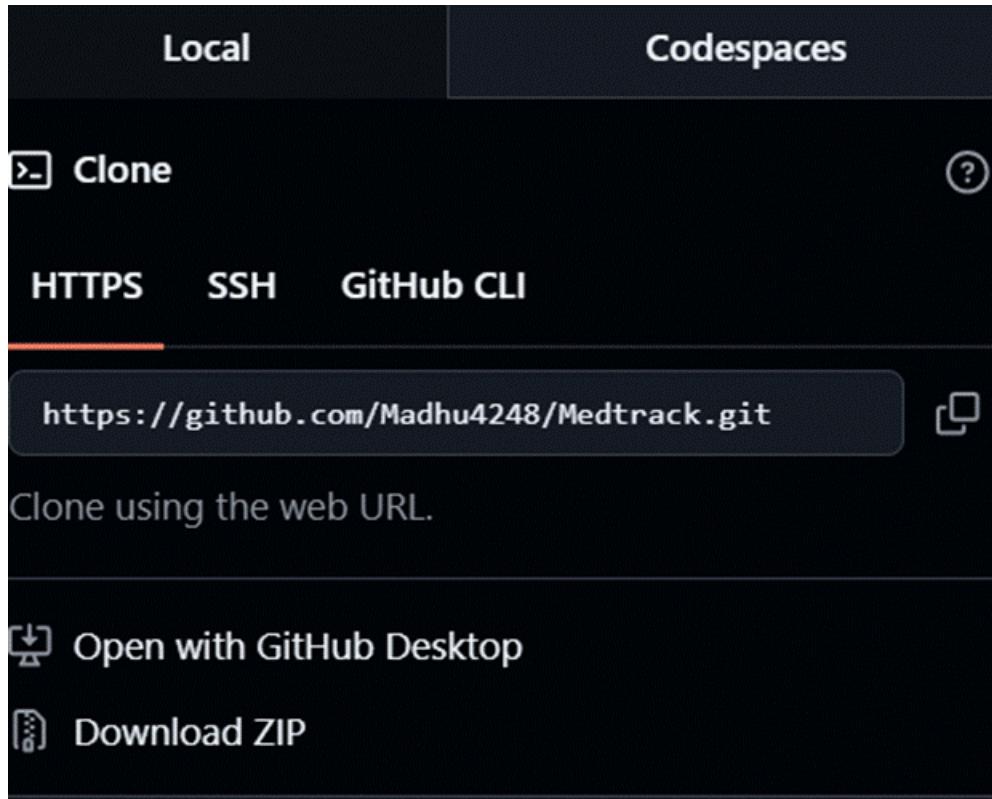
- a. AmazonDynamoDBFullAccess: Allows EC2 to perform read/write operations on DynamoDB.

- b. AmazonSNSFullAccess: Grants EC2 the ability to send notifications via SNS.

Milestone 6: EC2 Instance Setup

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

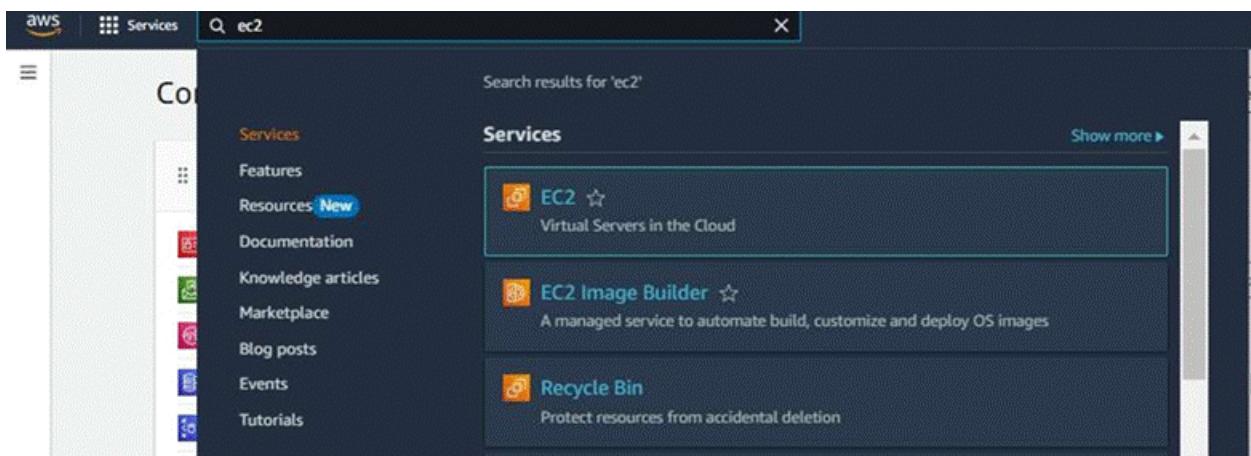
Varun-panchakarla	index.js	ca95df5 · 3 days ago	8 Commits
views	Add files via upload	4 days ago	
.env	.env	3 days ago	
README.md	Initial commit	4 days ago	
appointments.json	Add files via upload	4 days ago	
doctors.json	Add files via upload	4 days ago	
index.js	index.js	3 days ago	
package-lock.json	Add files via upload	4 days ago	
package.json	Add files via upload	4 days ago	
patients.json	Add files via upload	4 days ago	



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

a. Launch EC2 Instance

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



6. Click on Launch instance to launch EC2 instance

The screenshot shows the AWS EC2 Instances page. The left sidebar includes links for EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, and Savings Plans. The main content area has a search bar and filters for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. A message states 'No instances' and 'You do not have any instances in this region.' A 'Launch instances' button is at the bottom.

The screenshot shows the 'Launch an instance' wizard. Step 1: Name and tags. It shows a 'Name' field containing 'InstantLibraryApp' and a 'Add additional tags' link. To the right, there's a 'Summary' section with 'Number of instances: 1', 'Software Image (AMI) - Amazon Linux 2023 AMI 2025.5.2...', 'Virtual server type (instance type) - t2.micro', and a 'Firewall (security group)' link.

- 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 AMIs: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and others. A 'Browse more AMIs' link is available. Below, the 'Amazon Linux 2023 AMI' is selected, showing its details: 'ami-02b49a24cfb95941c (64-bit (x86), uefi-preferred) / ami-04ad8c7fcc828fad4 (64-bit (Arm), uefi)', 'Free tier eligible', '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 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](#)

8. Activity 6.2:Configure securitygroups for HTTP, and SSH access.

Architecture: 64-bit (x86) Boot mode: uefi-preferred AMI ID: ami-078264b8ba71b

Number of instances: 1

Software image: Amazon Linux 2022 (x86_64)

Instance type: t2.micro

Key pair (login): None

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

Key pair name - required

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.

Network settings

VPC - required: vpc-03cdc7b6f19dd7211 (default)

Subnet: No preference

Auto-assign public IP: Enable

Additional charges apply when outside of free tier allowance.

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

Create security group Select existing security group

Security group name - required: launch-wizard

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

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

Inbound Security Group Rules

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

Type : Info	Protocol : Info	Port range : Info
ssh	TCP	22
Source type : Info	Source : Info	Description - optional : Info
Anywhere	Add CIDR, prefix list or security	e.g. SSH for admin desktop
0.0.0.0/0 X		
- ▼ Security group rule 2 (TCP, 80, 0.0.0.0/0)

Type : Info	Protocol : Info	Port range : Info
HTTP	TCP	80
Source type : Info	Source : Info	Description - optional : Info
Custom	Add CIDR, prefix list or security	e.g. SSH for admin desktop
0.0.0.0/0 X		
- ▼ Security group rule 3 (TCP, 5000, 0.0.0.0/0)

Type : Info	Protocol : Info	Port range : Info
Custom TCP	TCP	5000
Source type : Info	Source : Info	Description - optional : Info
Custom	Add CIDR, prefix list or security	e.g. SSH for admin desktop
0.0.0.0/0 X		

[Add security group rule](#)

EC2 > [Launch an instance](#)

Status: Successfully initiated launch of instance i-001861022fbca2189

[Launch log](#)

Next Steps

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

C 1 2 3 4 5

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	Create a new RDS database	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 allow you to create budgets, forecast spend, and take actions on your costs and usage from a single location.	Create or update Amazon CloudWatch alarms for the instance.	Recover the instances you just launched into a different Availability Zone or a different Region using AWS Elastic Disaster Recovery (EDR).	Amazon GuardDuty enables you to continuously monitor for malicious runtime activity and unauthorized behavior, with near real-time visibility into on-hand 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

[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											Actions			
Last updated less than a minute ago											Connect	Instance state	Actions	Last modified
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	Launch config	
InstantLibraryApp	i-001861022fbcac290	Stopped	t2.micro	-	View alarms	ap-south-1b	-	-	-	-	disabled	Launch config		

EC2 > Instances > i-001861022fbcac290

Instance summary for i-001861022fbcac290 (InstantLibraryApp)

Updated less than a minute ago

Instance ID	i-001861022fbcac290	Public IPv4 address	Private IPv4 addresses						
IPv6 address	-	Instance state	172.31.3.5						
Hostname type	IP name: ip-172-31-5-5.ap-south-1.compute.internal	Private IP DNS name (IPv4 only)	Public IPv4 DNS						
Answer private resource DNS name	ip-172-31-5-5.ap-south-1.compute.internal	ip-172-31-5-5.ap-south-1.compute.internal	-						
IPv4 (4)		Instance type	Classic IP addresses						
Auto-assigned IP address	-	t2.micro	-						
VPC role	aws_dynamodb_role	VPC ID	AWS Compute Optimizer finding						
IMDv2		vpc-030370f9bf93d7211	Opt-in to AWS Compute Optimizer for recommendations Learn more						
Required		Subnet ID	Auto Scaling Group name						
		subnet-09893144400c0e9a	-						
		Instance ARN							
		arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290							
Details	Status and alarms	Monitoring	Security	Networking	Storage	Tags	Connect	Instance state	Actions

EC2 > Instances > i-001861022fbcac290

Instance summary for i-001861022fbcac290 (InstantLibraryApp)

Updated less than a minute ago

Instance ID	i-001861022fbcac290	Public IPv4 address	Private IPv4 addresses						
IPv6 address	-	Instance state	172.31.3.5						
Hostname type	IP name: ip-172-31-5-5.ap-south-1.compute.internal	Private IP DNS name (IPv4 only)	Public IPv4 DNS						
Answer private resource DNS name	ip-172-31-5-5.ap-south-1.compute.internal	ip-172-31-5-5.ap-south-1.compute.internal	-						
IPv4 (4)		Instance type	Change security groups						
Auto-assigned IP address	-	t2.micro	Get Windows password						
VPC role	aws_dynamodb_role	VPC ID	Modify IAM role						
IMDv2		vpc-030370f9bf93d7211	Networking						
Required		Subnet ID	Security						
		subnet-09893144400c0e9a	Image and templates						
		Instance ARN	Monitor and troubleshoot						
		arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290							
Details	Status and alarms	Monitoring	Security	Networking	Storage	Tags	Connect	Instance state	Actions

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

10. Now connect the EC2 with the files

```
A newer release of "Amazon Linux" is available.
Version 2023.6.20241010!
Run "/usr/bin/dnf check-release-update" for full release and version update info
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023
Last login: Tue Oct 16 04:17:59 2024 from 10.233.177.3
[ec2-user@ip-172-31-3-5 ~] 6
```

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

11. 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 Node.js 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 a CloudShell terminal window with the following content:

```

Verifying : git-core-2.47.1-1.amzn2.0.3.x86_64
Verifying : lperl-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

3/6
4/6
5/6
6/6

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)
PublicIP: 52.91.51.94 PrivateIPs: 172.31.18.239

```

The terminal shows the deployment of a Node.js application named "MedTrack-repo". It includes cloning the repository, installing dependencies, and listing files in the directory.

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

The screenshot shows a web browser window with multiple tabs open. The active tab displays terminal output from an EC2 instance. The output includes:

```
Patients Table: PatientsTable
Doctors Table: DoctorsTable
Appointments Table: Appointments
^C
[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
```

Below the terminal output, a modal window titled "i-09920a49e73472f79 (medtrack-server)" shows the public IP as 54.167.78.35 and the private IP as 172.31.20.85.

The browser interface includes standard navigation controls, a search bar, and a toolbar with various icons. At the bottom, there's a taskbar with icons for File Explorer, Edge, and other applications, along with system status indicators like battery level, signal strength, and date/time.

Access the website through:

Public IPs: <http://52.91.51.94:3000/>

Milestone 8: Testing and Deployment

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

Login Page:

Student - Skill Wallet SI-27282-1751965849 chandraname/Medtrack MedTrack - Registration

localhost:3000/register

Create Your Account

Patient Registration Doctor Registration

First Name

Last Name

Age

Gender

Email Address

Phone Number

Address

Password

Confirm Password

Light rain At night Search ENG IN 17:29 08-07-2025

Register Page:

Student - Skill Wallet SI-27282-1751965849 chandraname/Medtrack MedTrack - Registration

localhost:3000/register

Create Your Account

Patient Registration Doctor Registration

First Name

Last Name

Age

Gender

Email Address

Phone Number

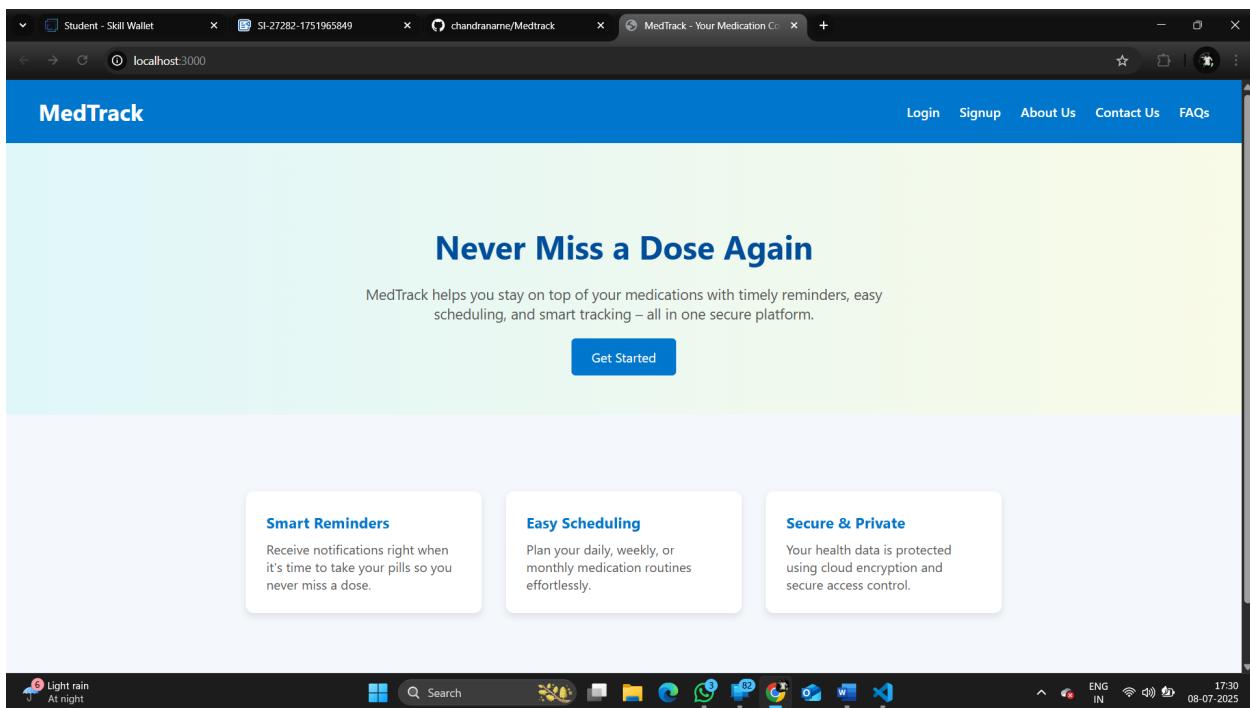
Address

Password

Confirm Password

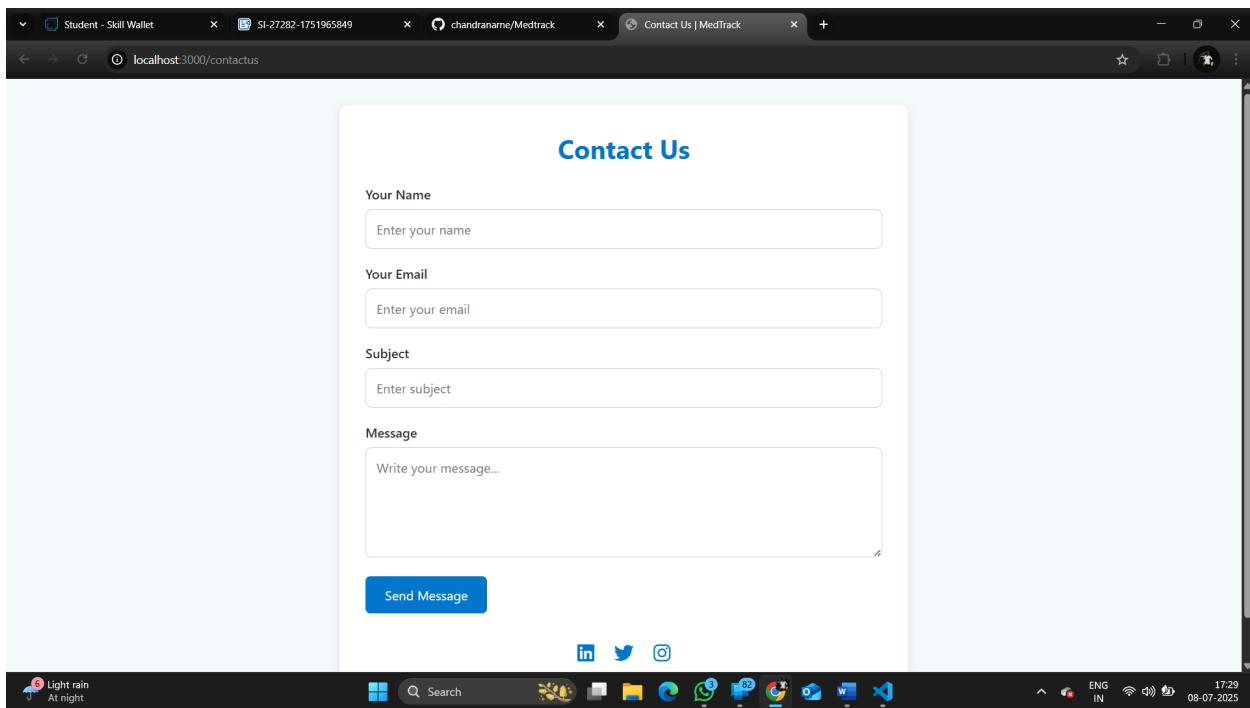
Light rain At night Search ENG IN 17:29 08-07-2025

Home page:



The screenshot shows the MedTrack application running on a Windows desktop. The browser window title is "localhost:3000". The page features a blue header bar with the "MedTrack" logo on the left and navigation links for "Login", "Signup", "About Us", "Contact Us", and "FAQs" on the right. Below the header is a large yellow section containing the text "Never Miss a Dose Again" in bold blue font. A subtext below it reads: "MedTrack helps you stay on top of your medications with timely reminders, easy scheduling, and smart tracking – all in one secure platform." A blue "Get Started" button is centered in this section. At the bottom of the page, there are three white callout boxes: "Smart Reminders" (Receive notifications right when it's time to take your pills so you never miss a dose.), "Easy Scheduling" (Plan your daily, weekly, or monthly medication routines effortlessly.), and "Secure & Private" (Your health data is protected using cloud encryption and secure access control.). The Windows taskbar at the bottom shows various pinned icons and the system tray with the date and time (08-07-2025).

ContactUs :



The screenshot shows the "Contact Us" page of the MedTrack application. The browser window title is "localhost:3000/contactus". The page has a white background with a central form titled "Contact Us" in blue. The form contains four input fields: "Your Name" (placeholder: "Enter your name"), "Your Email" (placeholder: "Enter your email"), "Subject" (placeholder: "Enter subject"), and a larger "Message" area (placeholder: "Write your message..."). Below the message area is a blue "Send Message" button. At the bottom of the page, there are social media sharing icons for LinkedIn, Twitter, and Instagram. The Windows taskbar at the bottom shows various pinned icons and the system tray with the date and time (08-07-2025).

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.