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

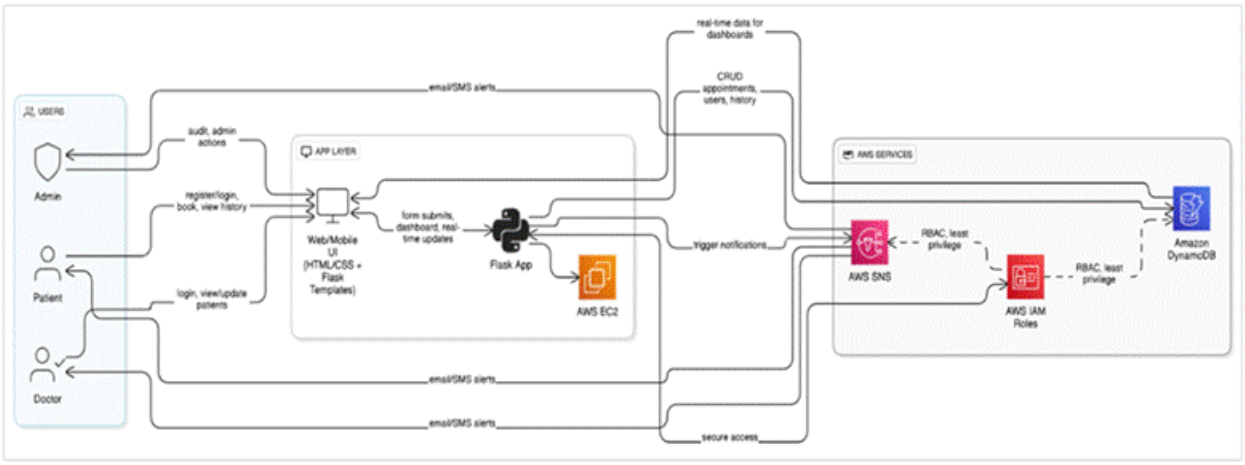
A patient logs into the MedTrack web app, which is hosted on Amazon EC2 instances. They schedule an appointment, and the booking details are saved in Amazon DynamoDB, enabling fast retrieval and updates. Immediately, AWS SNS sends a confirmation SMS and email to the patient with appointment details.

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.

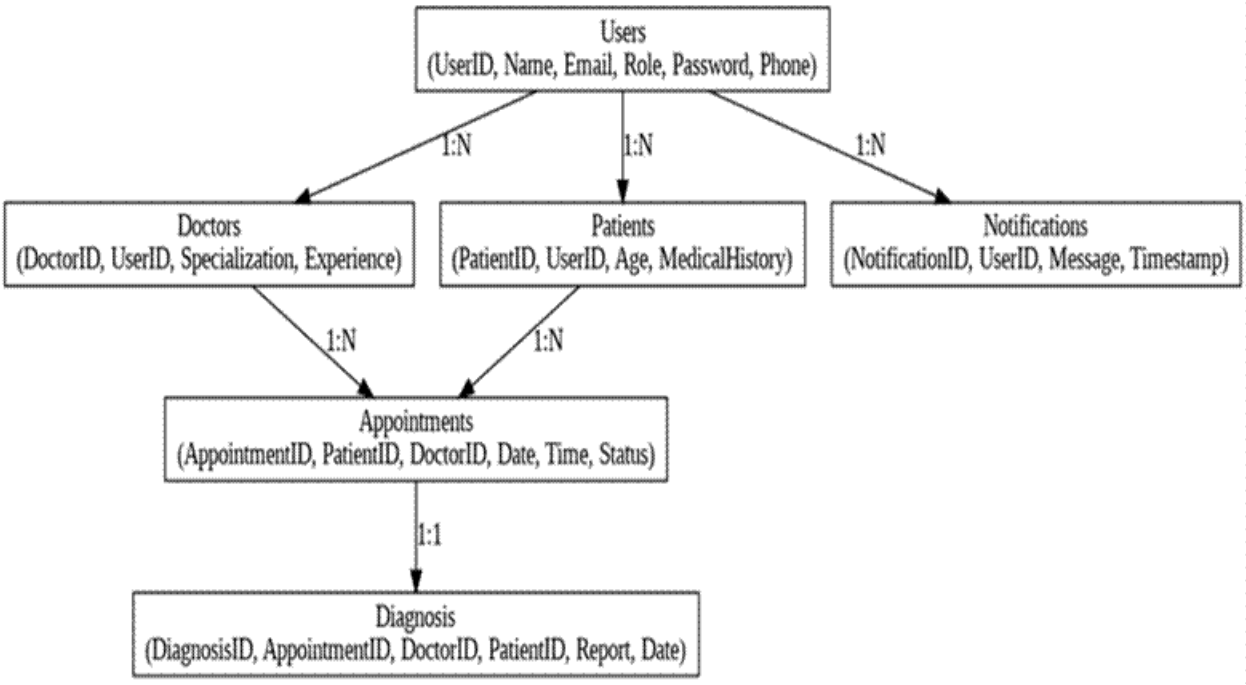
Scenario 3 – Emergency Alerts and Notifications

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 andLogin

Activity 1.1: Set up an AWS accountif not alreadydone. Activity 1.2: Log in to the AWS Management Console

2. DynamoDB Database Creation and Setup

Activity 2.1: Create a DynamoDBTable.
Activity 2.2: Configure Attributes for User Data and Book Requests.

3. SNS Notification Setup

Activity 3.1: CreateSNS topics for book requestnotifications.
Activity 3.2: Subscribeusers and librarystaff to SNS email notifications.

4. Backend Development and Application Setup

Activity 4.1:Develop the BackendUsing JavaScript.
Activity 4.2: IntegrateAWS Services Using boto3.

5. IAM Role Setup

Activity 5.1: CreateIAM Role
Activity 5.2: AttachPolicies

6. EC2 Instance Setup

Activity 6.1: Launchan EC2 instanceto host the JavaScript application.
Activity 6.2: Configuresecuritygroups for HTTP, and SSH access.

7. Deployment on EC2 Activity 7.1:Upload JavaScript Files
8. Activity 7.2: Runthe JavaScript App

9. Testing andDeployment

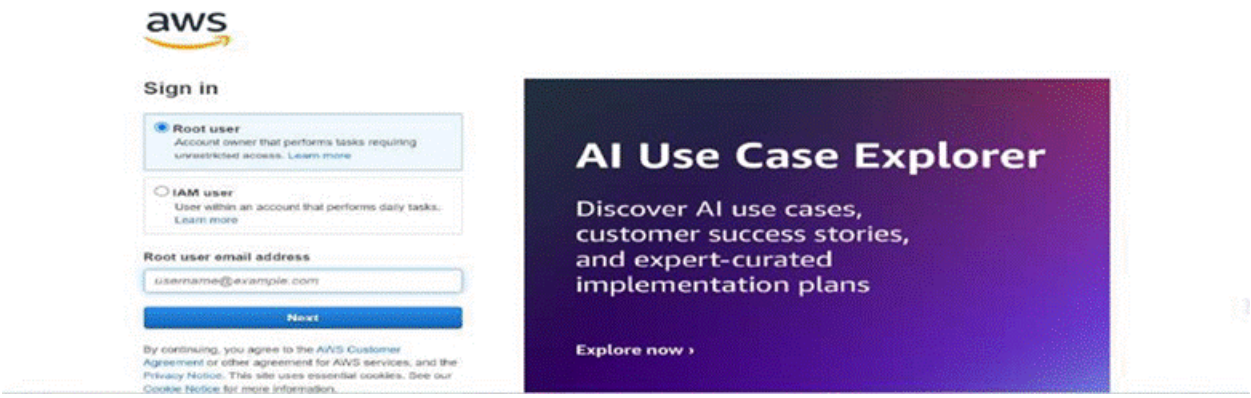
Activity 8.1: Conductfunctional testing to verify user registration, login, book requests,and notifications.

Milestone 1: AWS AccountSetup and Login

a. Activity 1.1: Set up anAWS account ifnot alreadydone.
   i. Sign upfor an AWS accountand configure billingsettings.
   ii.

b. Activity 1.2: Log in tothe AWS Management Console

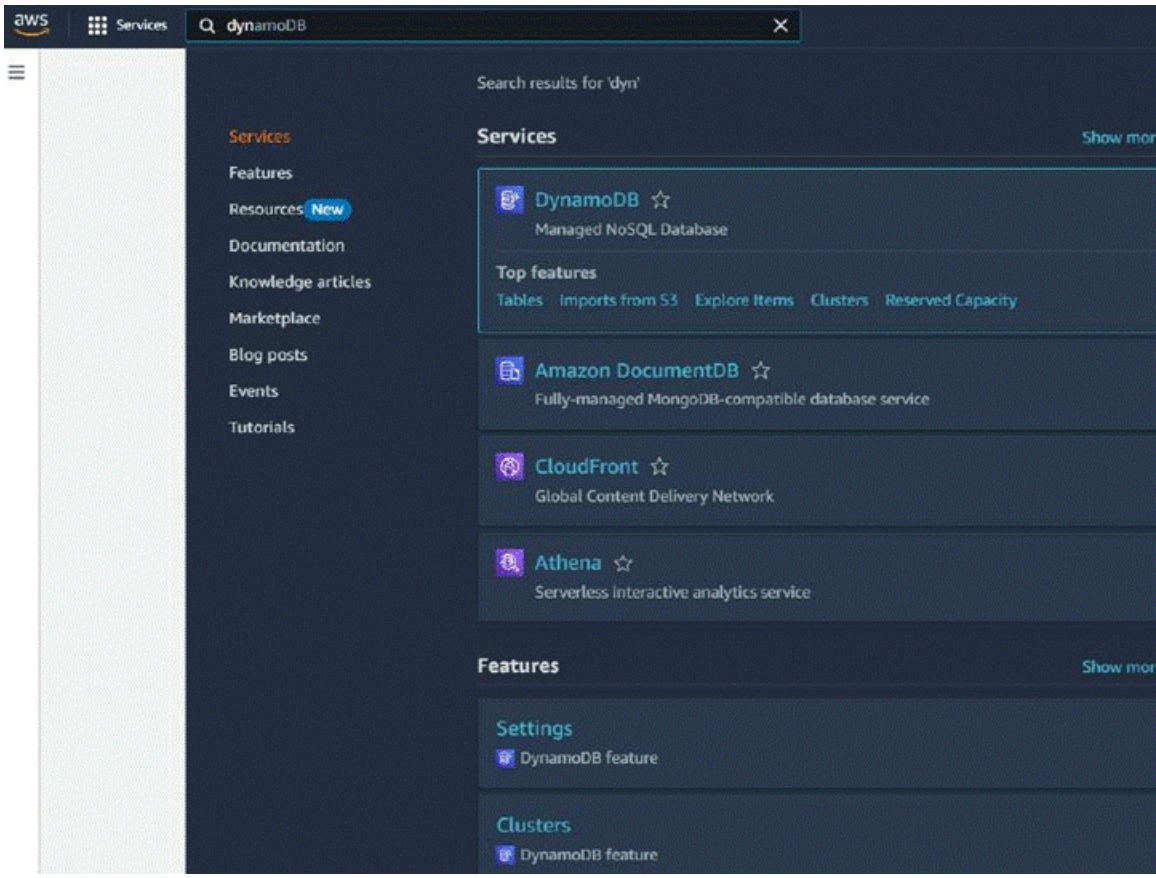
   i. After setting up your account,log in to the AWS Management Console.

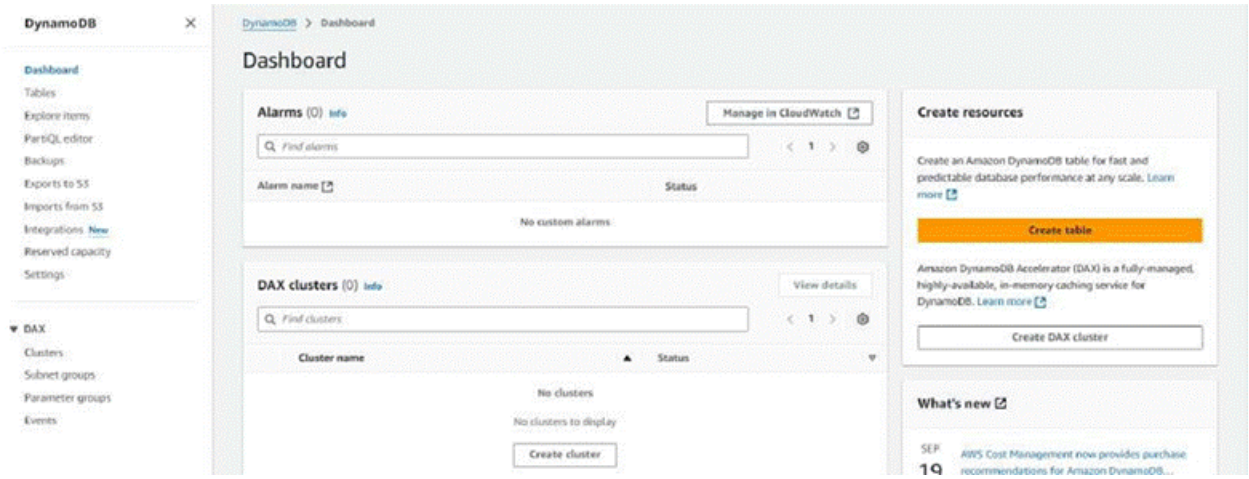


Milestone 2: DynamoDBDatabase Creation and Setup

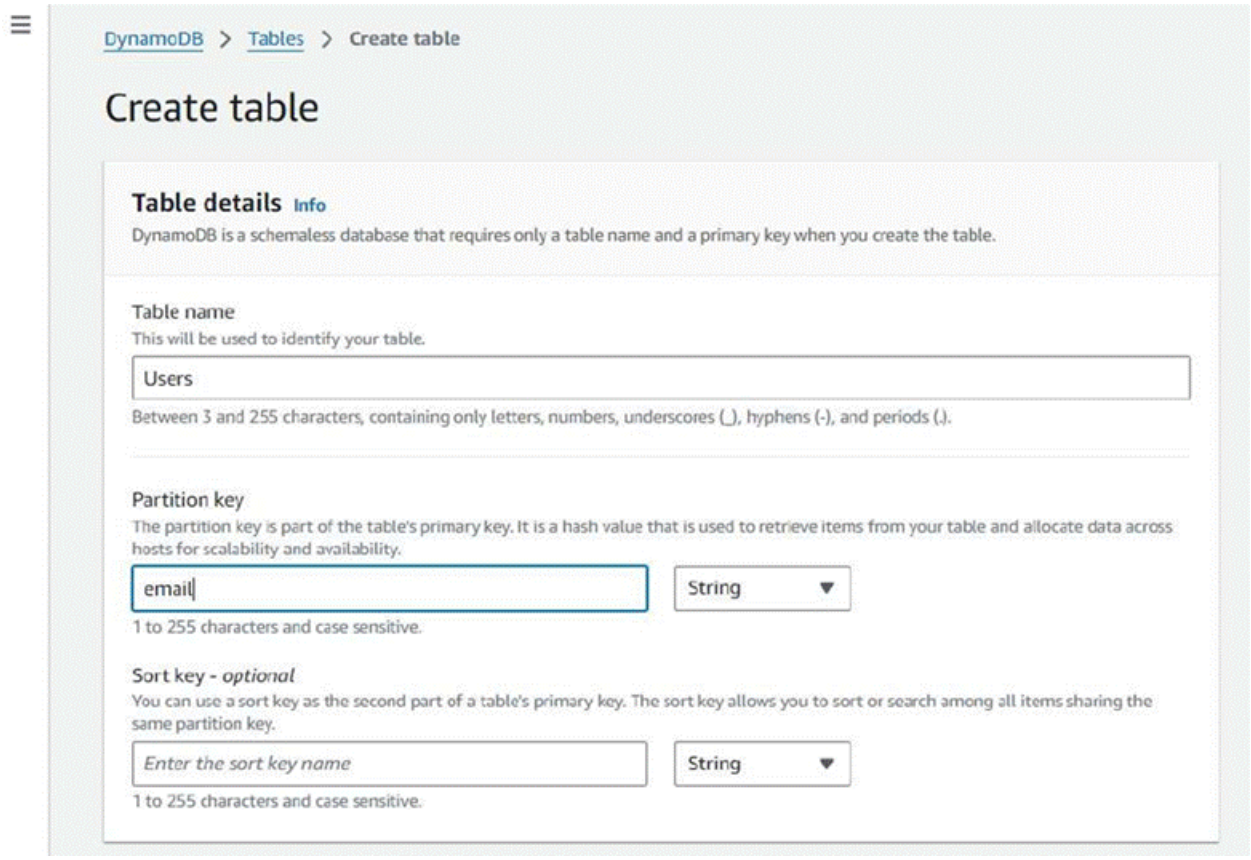
c. Activity 2.1:Navigate to the DynamoDB
   i. In the AWS Console, navigateto DynamoDB and click on create tables.

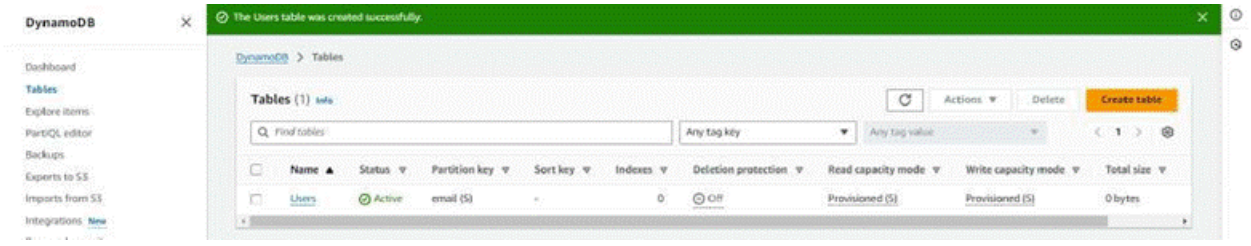
ii.





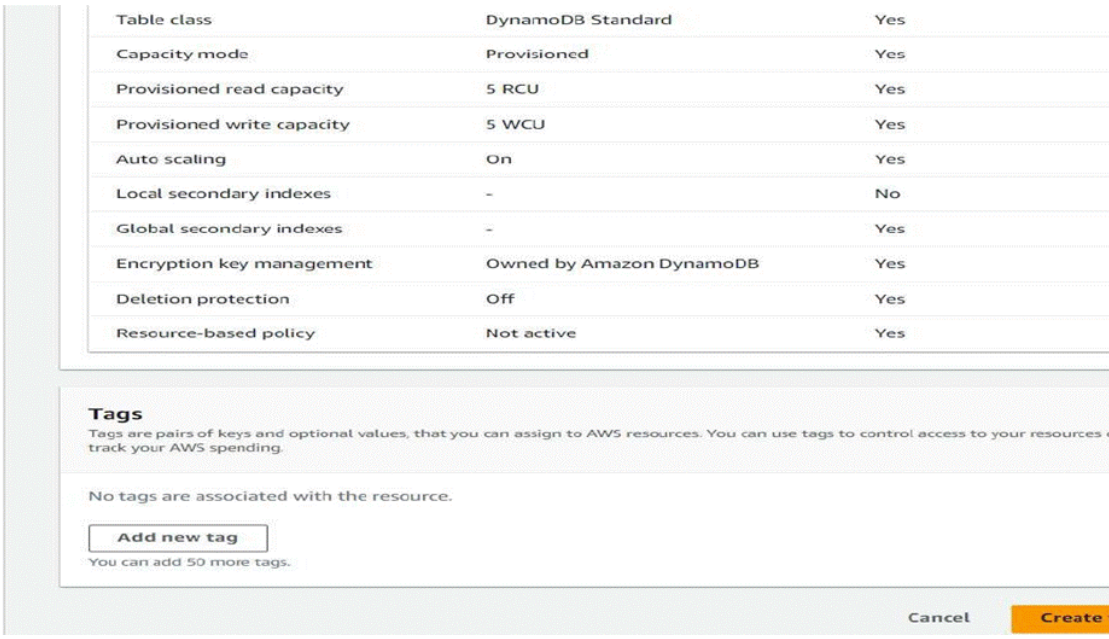
d. Activity 2.2:Create a DynamoDBtable for storingregistration details and book requests.
   i. Create Users table with partition key “Email” with type String and click on create tables.





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

iii.



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

1 ta 255 characters and case sensitive.

You can use a sort key as the second part of a tabte'sprimary key. The sort key allows you to sort orsearch among all rtems sharing the
Enter the sort key nome

same partition key.
4 to 255 characters and case sensitive.

DynamoDB \ Tables} Create table

Createtable

Table details inio

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 2S5 char acters, curyLaJning unly letters, rJumLt s, uryderscores , hmheris (-[, and p@iods (.).

Partition key

The partition key ispart of the table's primar'/ key. It isa hash value that is used to retrieve items from your table and allocate data across

email

1 ta 255 characters and case sensitive.

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

Enter the sort key nome

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

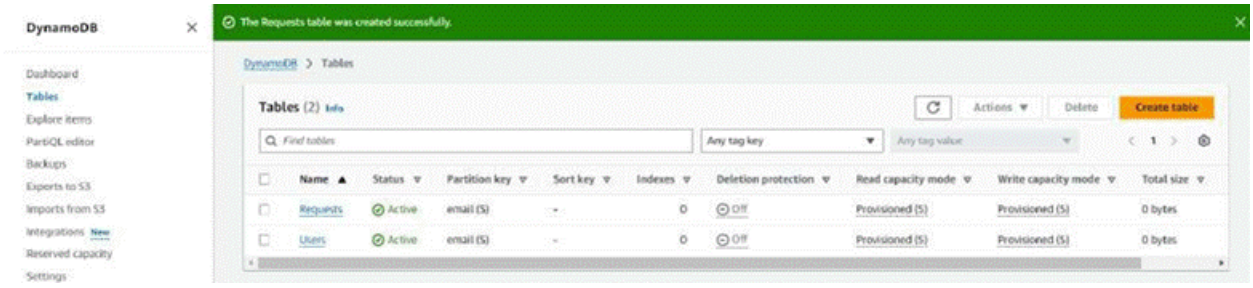
Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

Add new tag

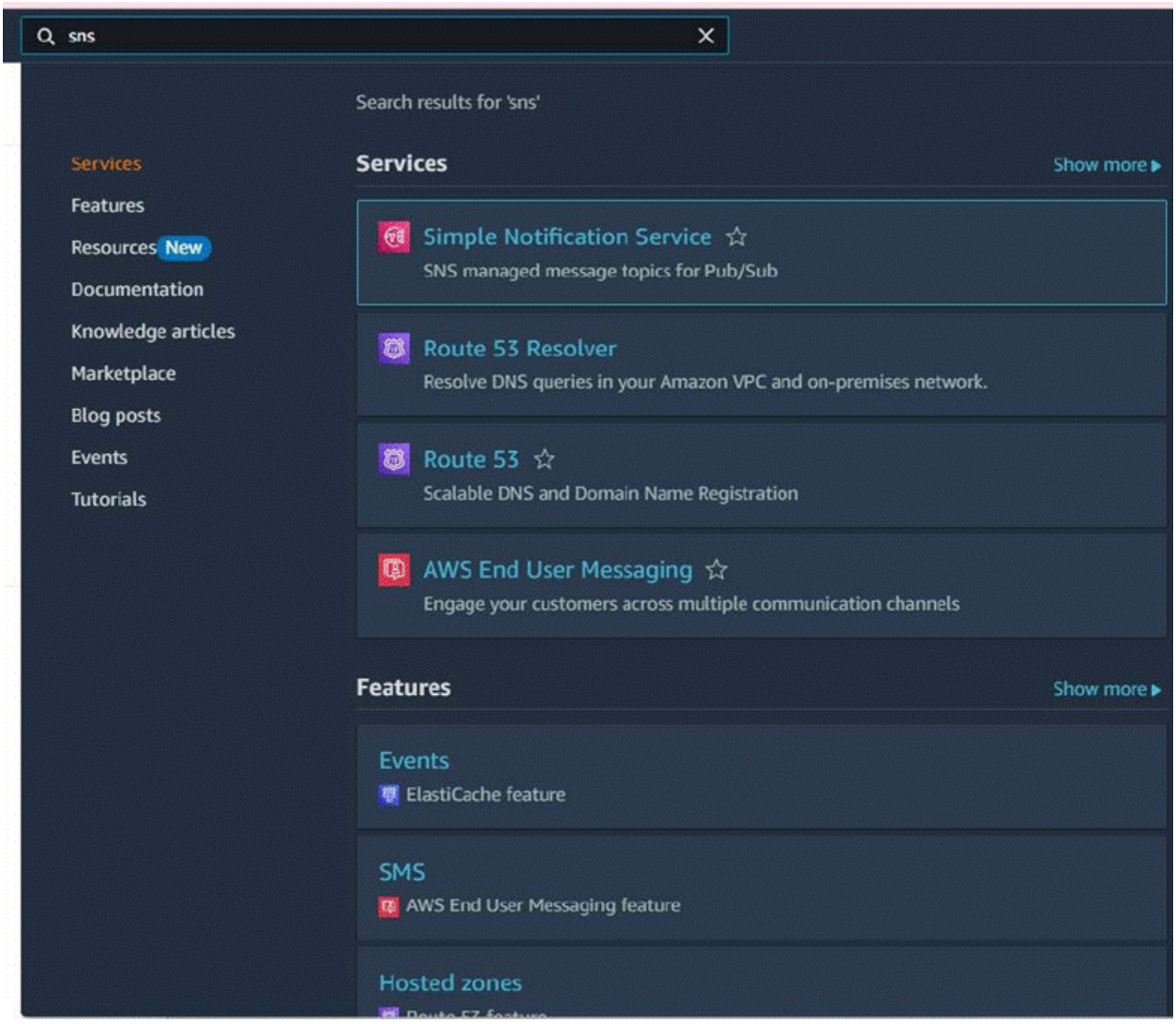
You can add 50 more tags.

Cancel Create table

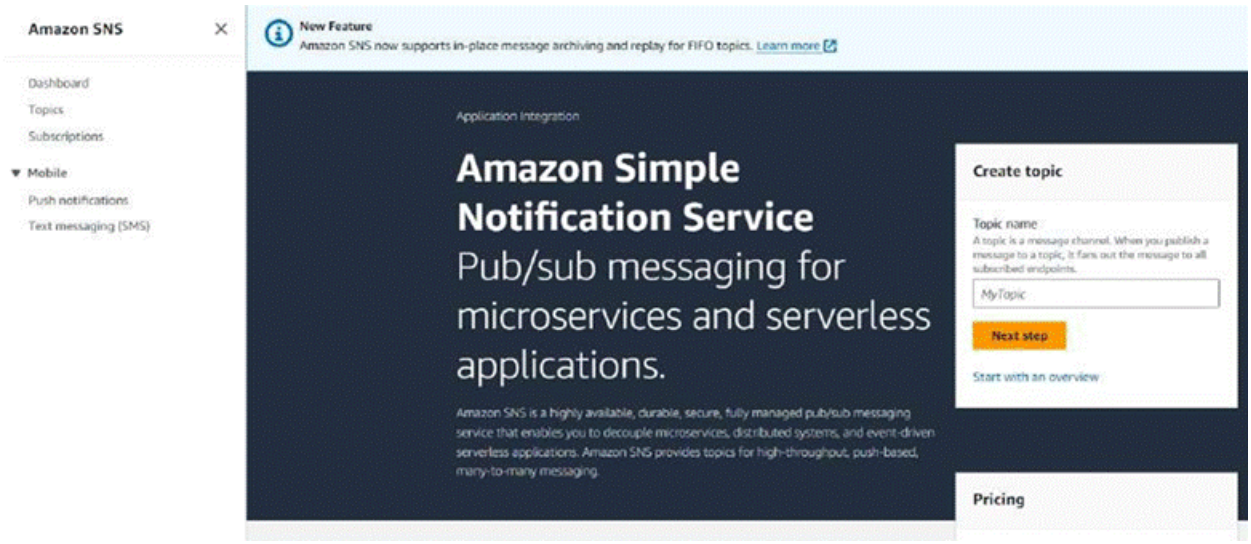


Milestone 3: SNS Notification Setup

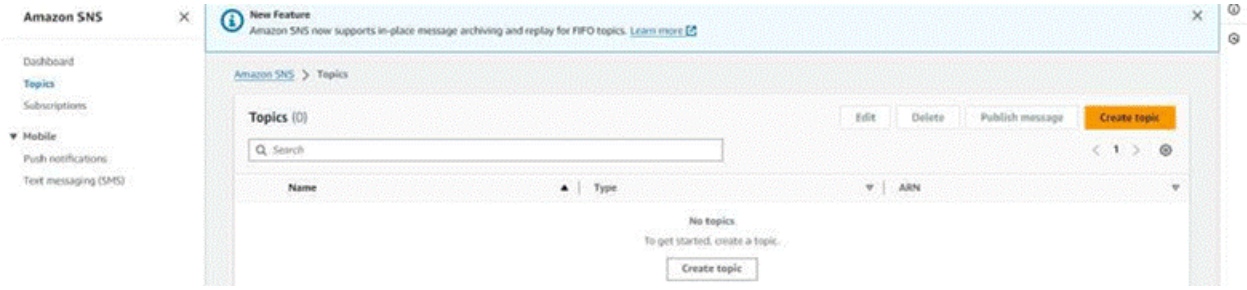
e. Activity 3.1: Create SNS topics for sending email notifications to users and library staff.



i. In the AWS Console,search for SNS and navigateto the SNS Dashboard.



ii. Click on Create Topic and choosea name for the topic.



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

Amazon 5N5 \ Topics Createtopic

Create topic

Details

Type trrfo

Topic type cannot be modified atter topic is created

Smart Internz

O FJFO (first-in, first-out)

1. Strictly—preserved message ordering
2. Mactly-onEe message delivery
3. High throughput, up to ZOO publishes/second
4. Subscription protocol: SOS

@ Standard

Best-effcrt message ordering at-least once message delivery
Highest throughput in publishes/secondSubscription protocols: SQS, Lambda, HTTP, SI IS, email, mobile application endpoints

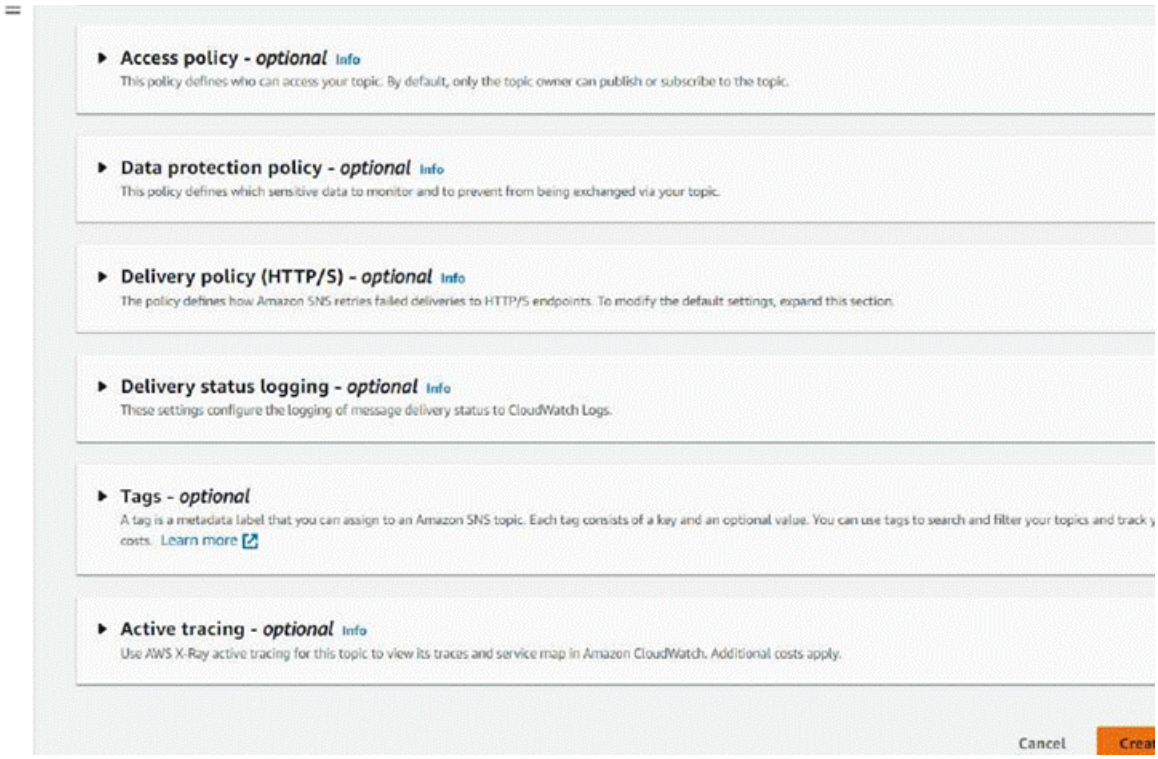
Name

BookRequeStNotificatiDr6

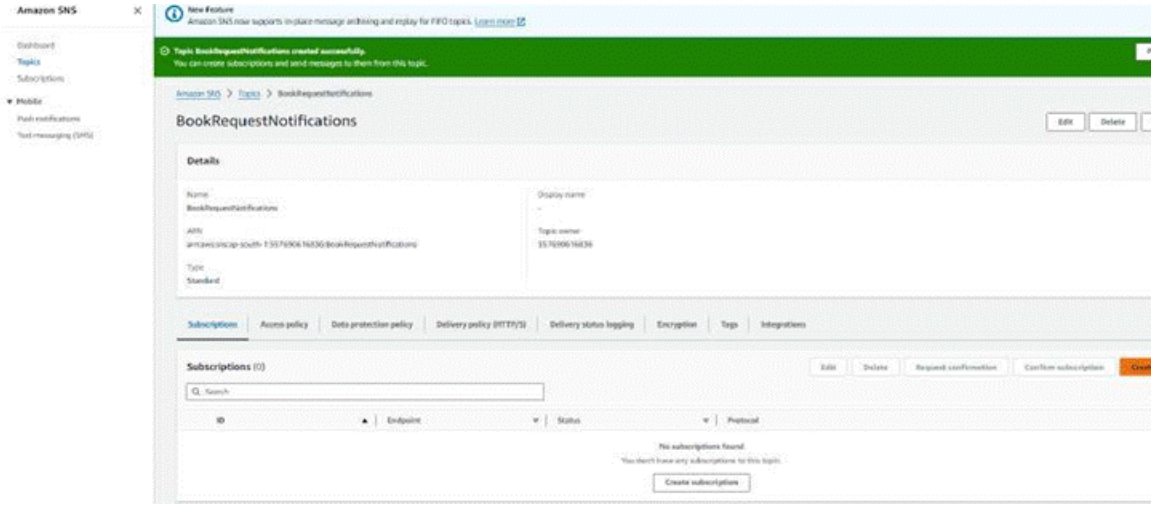
Maximum 256 characters. Can include alphanurr+eric characters, hyphens (-) and underscores M.

Display name - optional lrrfo

To use this topic with 5M5 subscriptions, enter a display name. Only the first 10 characters are displayed in an Si iS message.
Maximum 1 00 characters.

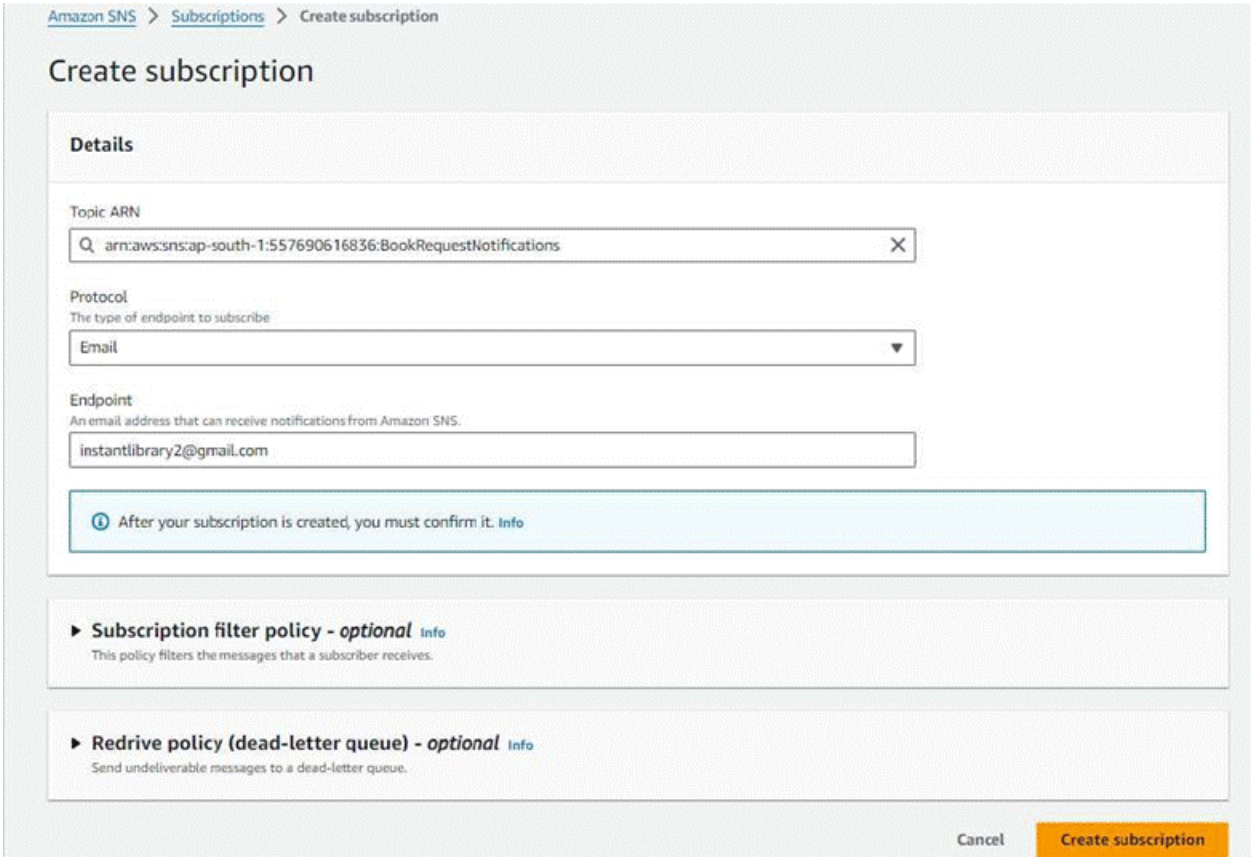
1. 

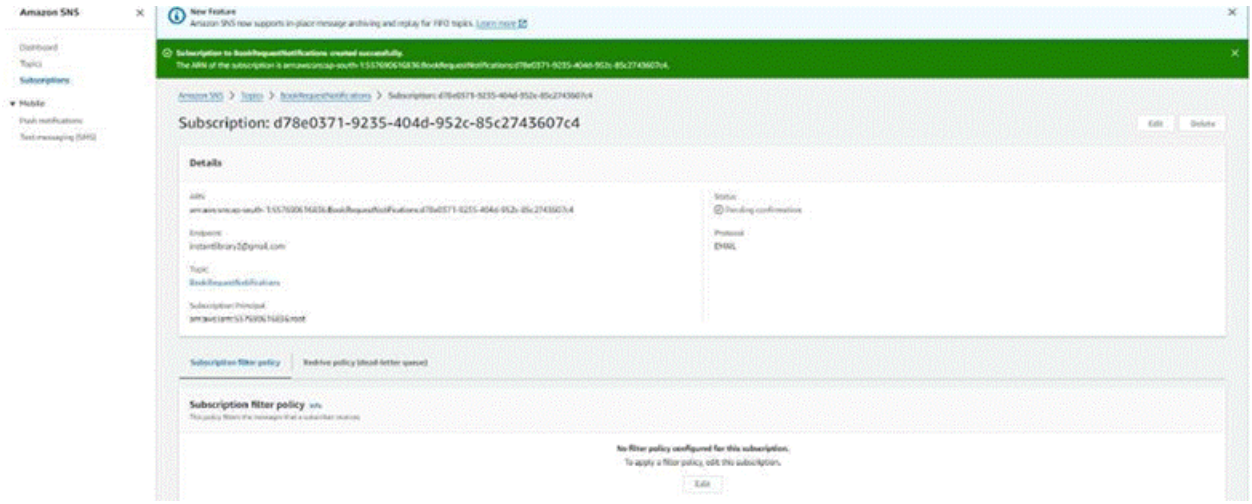
iv. Configure the SNS topic and note down the Topic ARN.

v. 

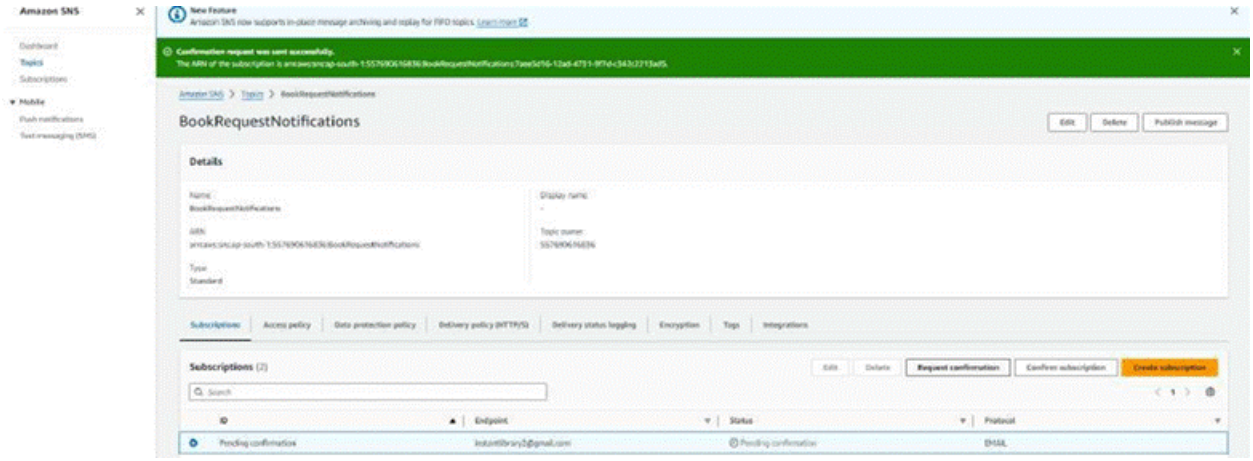
f. Activity 3.2: Subscribe users and staff to relevantSNS topics to receive real-time notifications when a book request is made.

i. Subscribe users (or admin staff) to this topic via Email.When abook request is made, notifications will be sent to the subscribed emails.





ii. After subscription requestfor the mail confirmation



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> 9
to me

You have chosen to subscribe to the topic:
arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications

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

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

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

You have chosen to subscribe to the topic:
arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications

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

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

aws

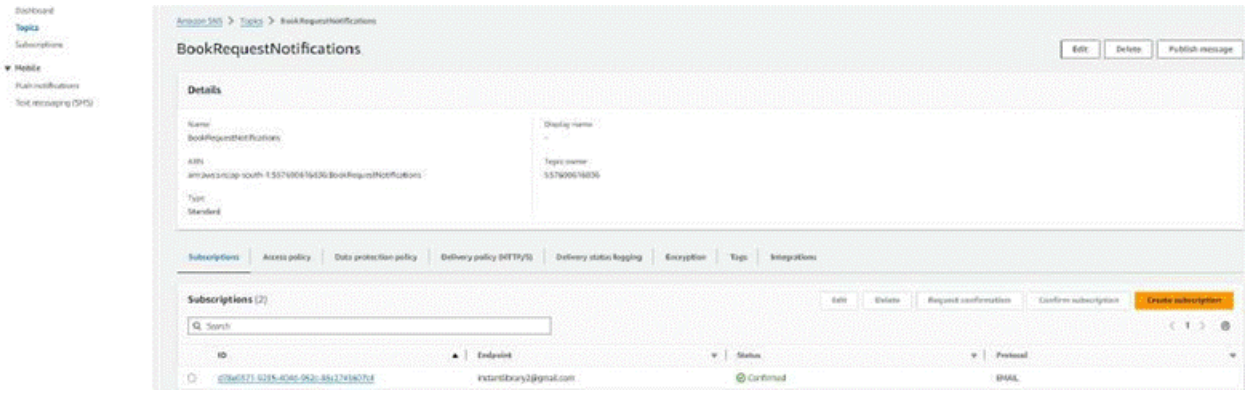
Simple Notification Service

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.

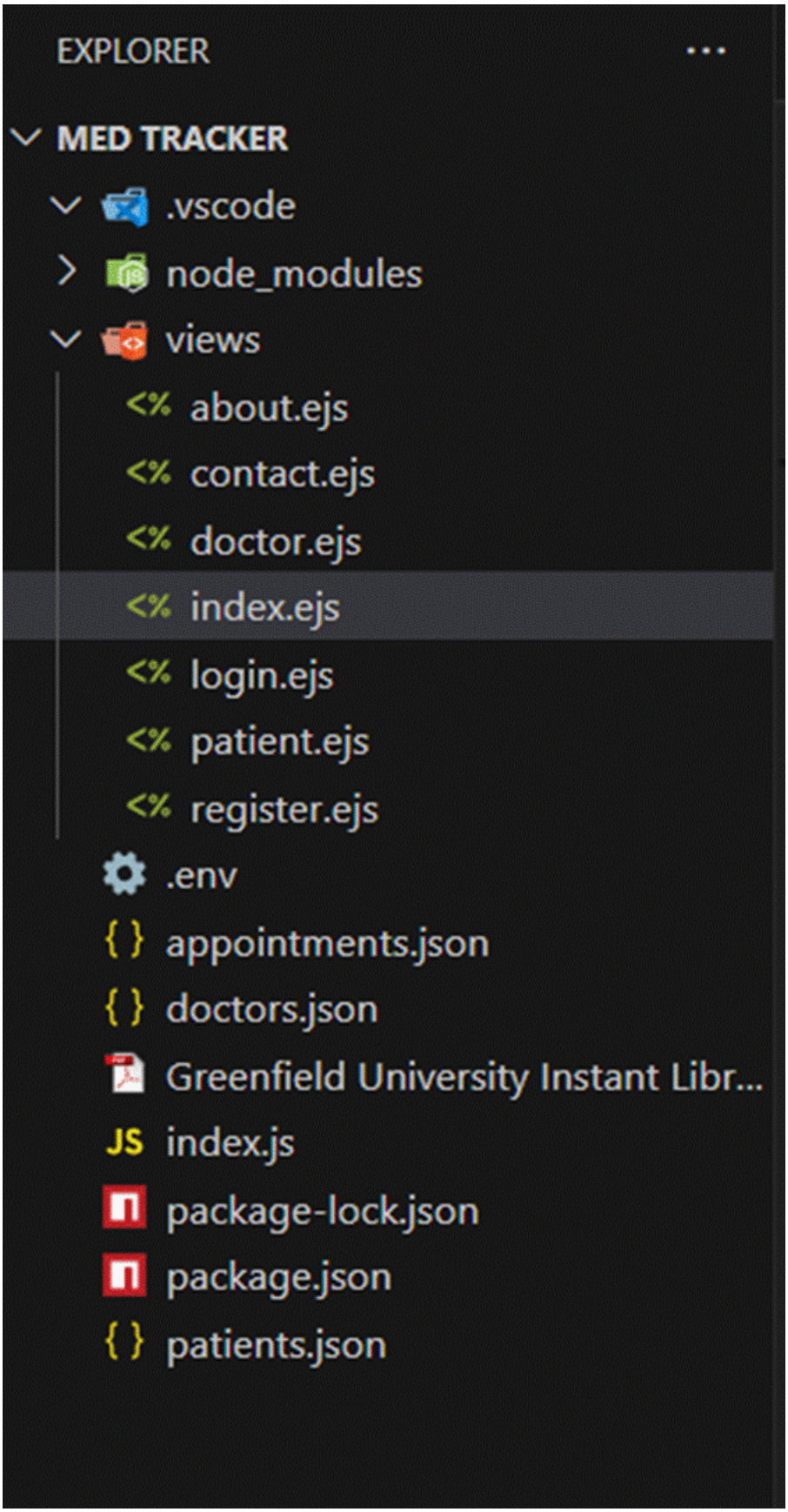


iv. Successfullydone with the SNS mail subscription and setup, now store the ARN link.

Milestone 4:Backend Development and Application Setup

g. Activity 4.1: Developthe backend usingJavaScript

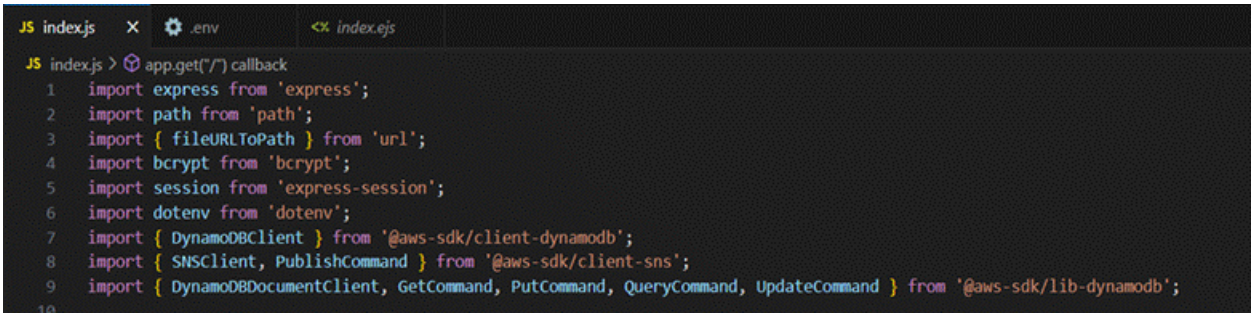
i. File Explorer Structure


EXPLORER
MED TRACKER
.vscode
node_modules
views
about.ejs
contact.ejs
doctor.ejs
index.ejs
login.ejs
patient.ejs
register.ejs
.env
appointments.json
doctors.json
Greenfield University Instant Libr...
index.js
package-lock.json
package.json
patients.json

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



Description: import essentiallibrariesincluding Express utilitiesfor routing, Boto3 for DynamoDB operations, SMTP and email modules for sending mails, and Bcrypt for password hashing and verification

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

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 });
```

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 forWeb Pages

1. Home Route:

```
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:

```
app.post('/register/patient', async (req, res) => {
  const { firstName, lastName, dob, gender, email, phone, address, password } = req.body;

  // Check if patient exists
  const existingPatient = await getUserByEmail(email, 'patient');
  if (existingPatient) return res.send('Patient exists');

  // Create new patient
  await createUser({
    id: Date.now().toString(),
    name: `${firstName} ${lastName}`,
    dob,
    gender,
    email,
    phone,
    address,
    password: await bcrypt.hash(password, 10)
  }, 'patient');

  // Send SNS notification
  if (SNS_TOPIC_ARN) {
    const snsCommand = new PublishCommand({
      TopicArn: SNS_TOPIC_ARN,
      Message: `New patient registered: ${firstName} ${lastName} (${email})`,
      Subject: 'MedTrack Registration'
    });
    await snsClient.send(snsCommand).catch(err => console.error('SNS Error:', err));
  }

  res.redirect('/login');
});
```

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 requestin DynamoDB, send a thank-youemail to the user, notify the admin, and confirmsubmission with a success message.

Exit Route:

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

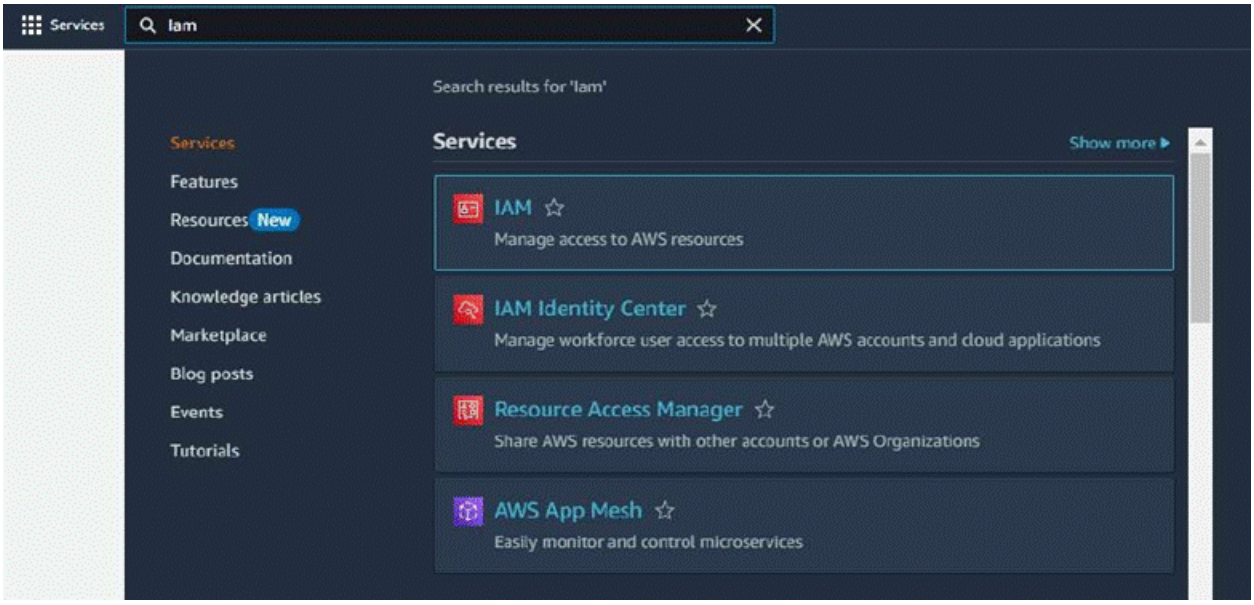
Description: define /exitrouteto render the exit.html page when the user choosesto leave or close the application.

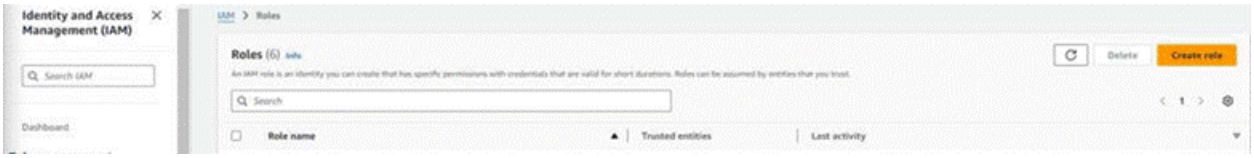
Deployment Code:

Description: start the Flask servertolisten on all networkinterfaces (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 AWSConsole,go to IAMand createa newIAM Role forEC2 to interact with DynamoDB and SNS.



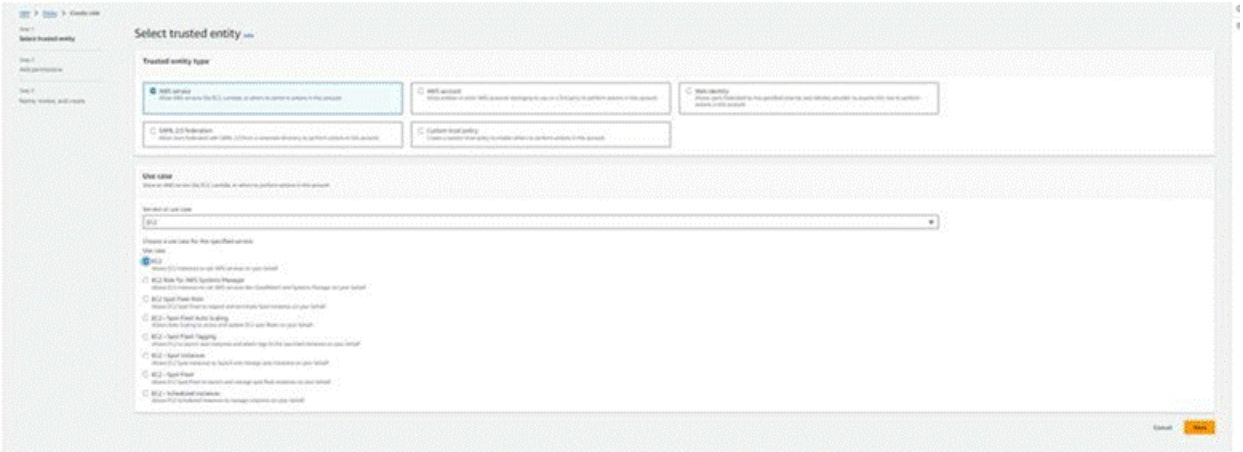


3. Activity 5.2: AttachPolicies.

Attach the followingpoliciesto the role:

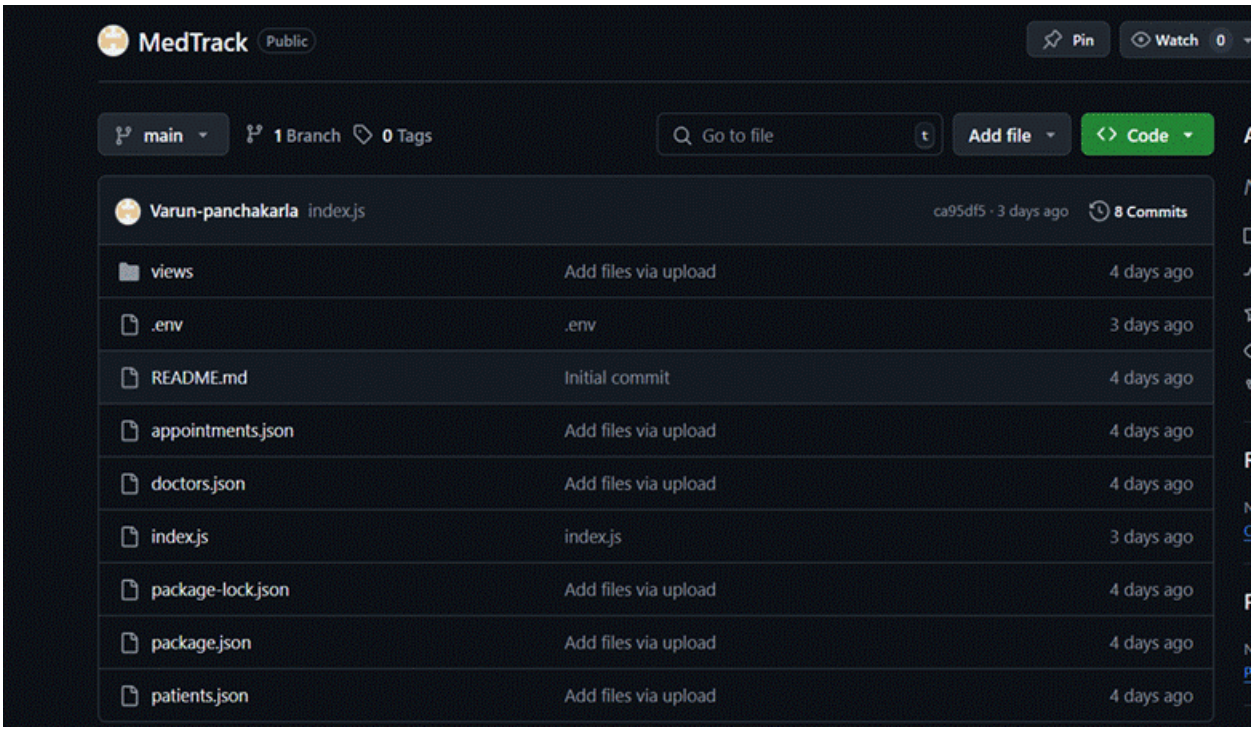
a. AmazonDynamoDBFullAccess: Allows EC2 toperform read/write operations on DynamoDB.

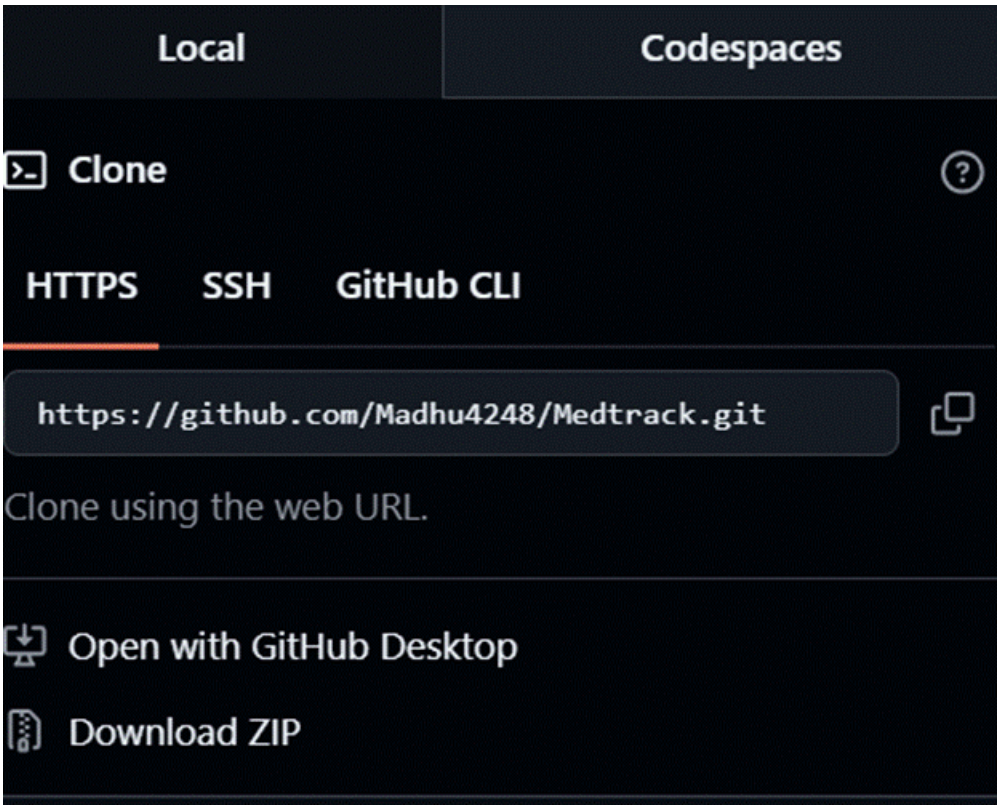
b. AmazonSNSFullAccess: Grants EC2 the abilityto send notifications via SNS.



Milestone 6: EC2 Instance Setup

4. Note: Load your index.jsand Html files into GitHubrepository.

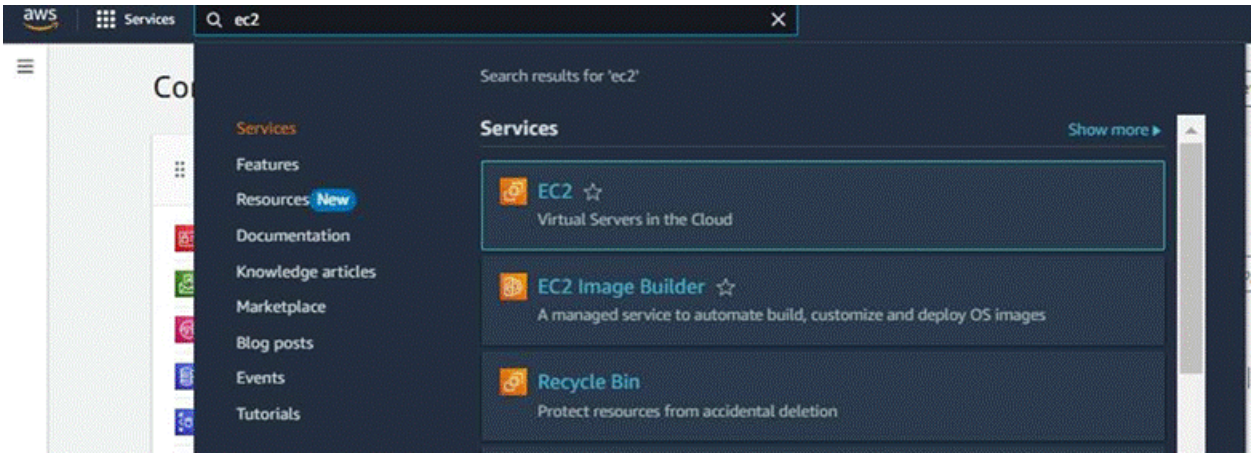




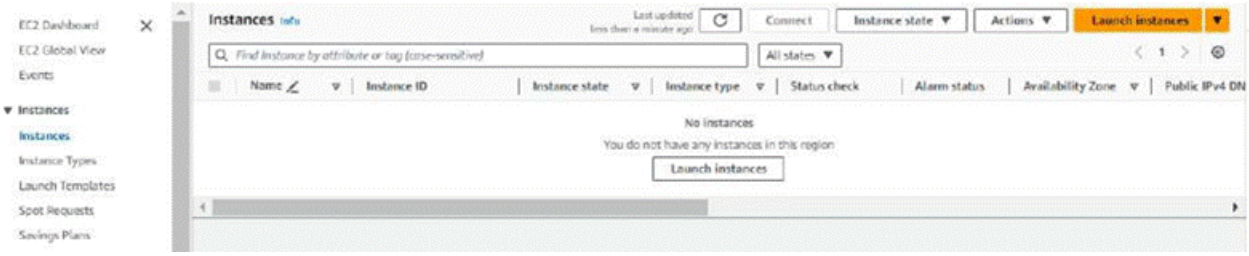
5. Activity 6.1:Launchan EC2 instance to host theFlask application.

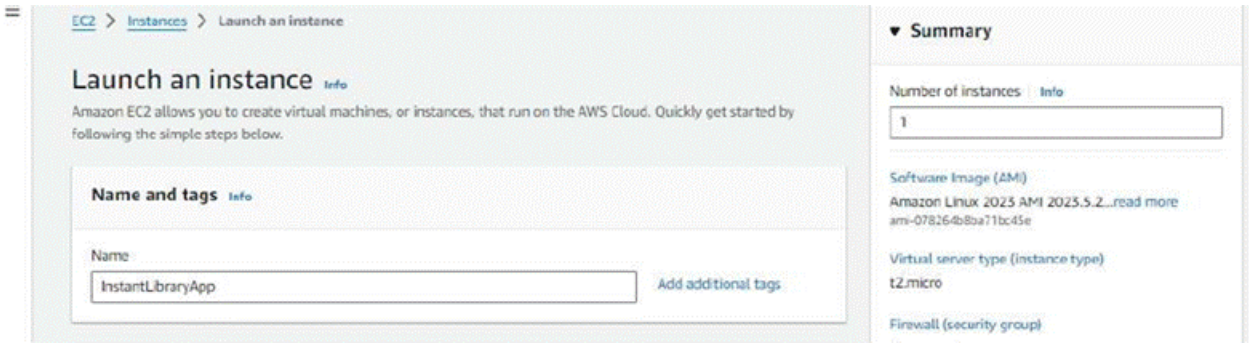
    a. Launch EC2 Instance

        i. In the AWS Console, navigateto EC2 and launch anew instance.

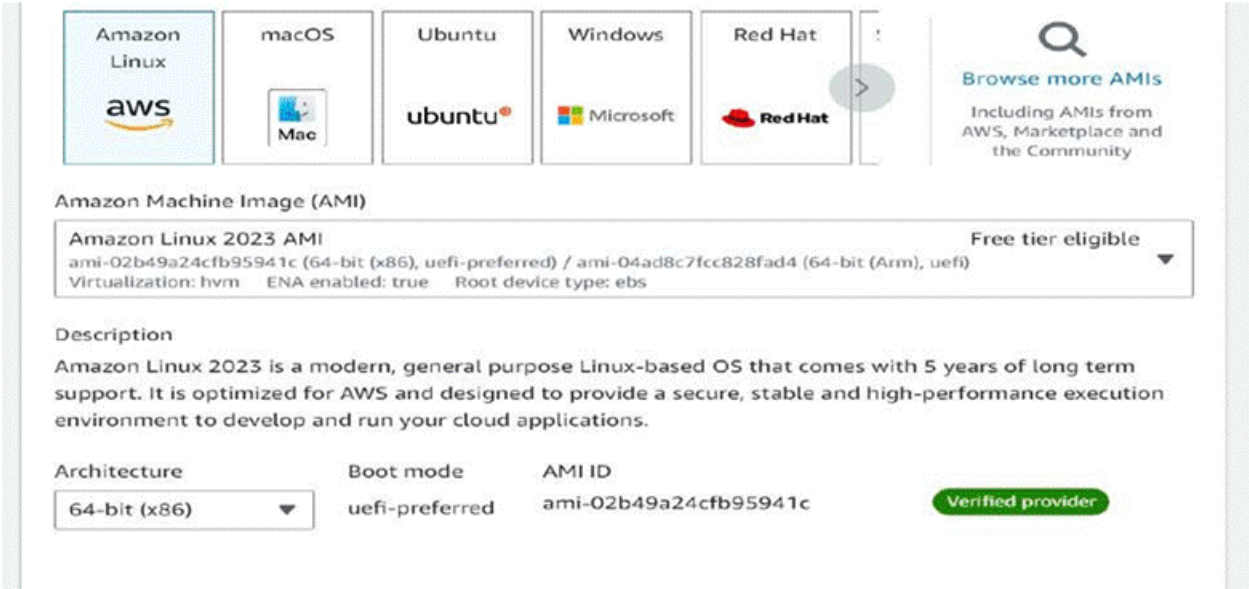


6. Click on Launchinstance tolaunchEC2 instance

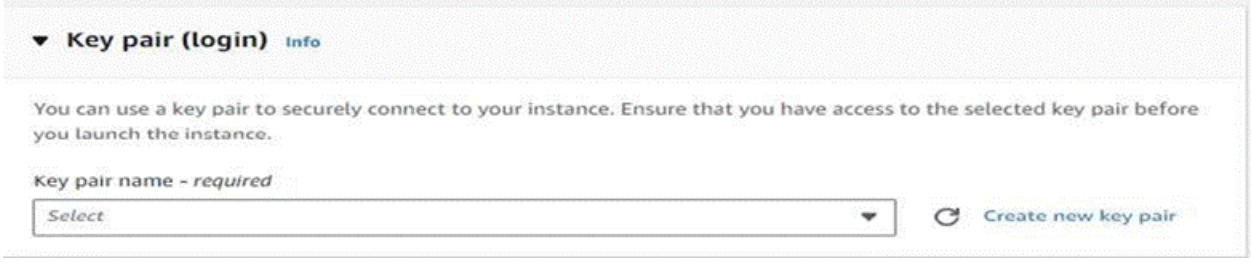


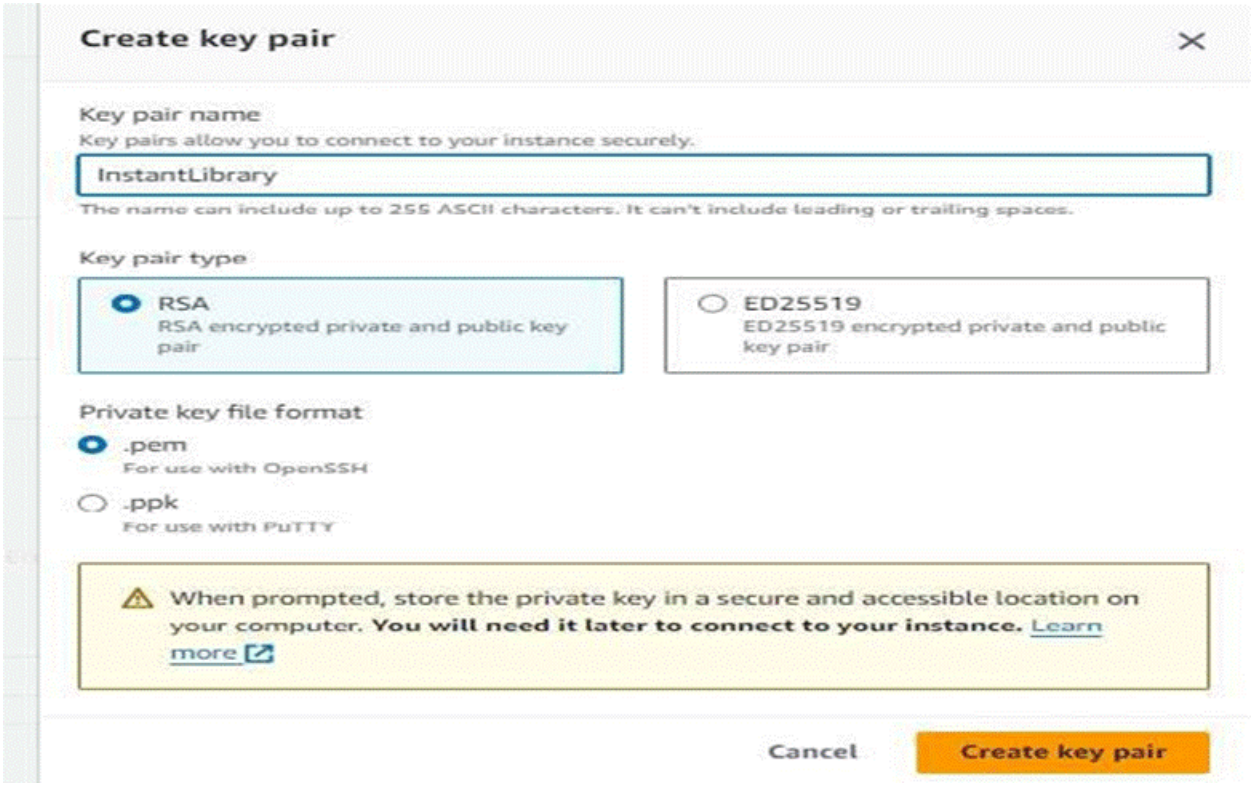


a. Choose Amazon Linux 2 or Ubuntu as the AMIand t2.micro asthe instancetype (free-tier eligible).

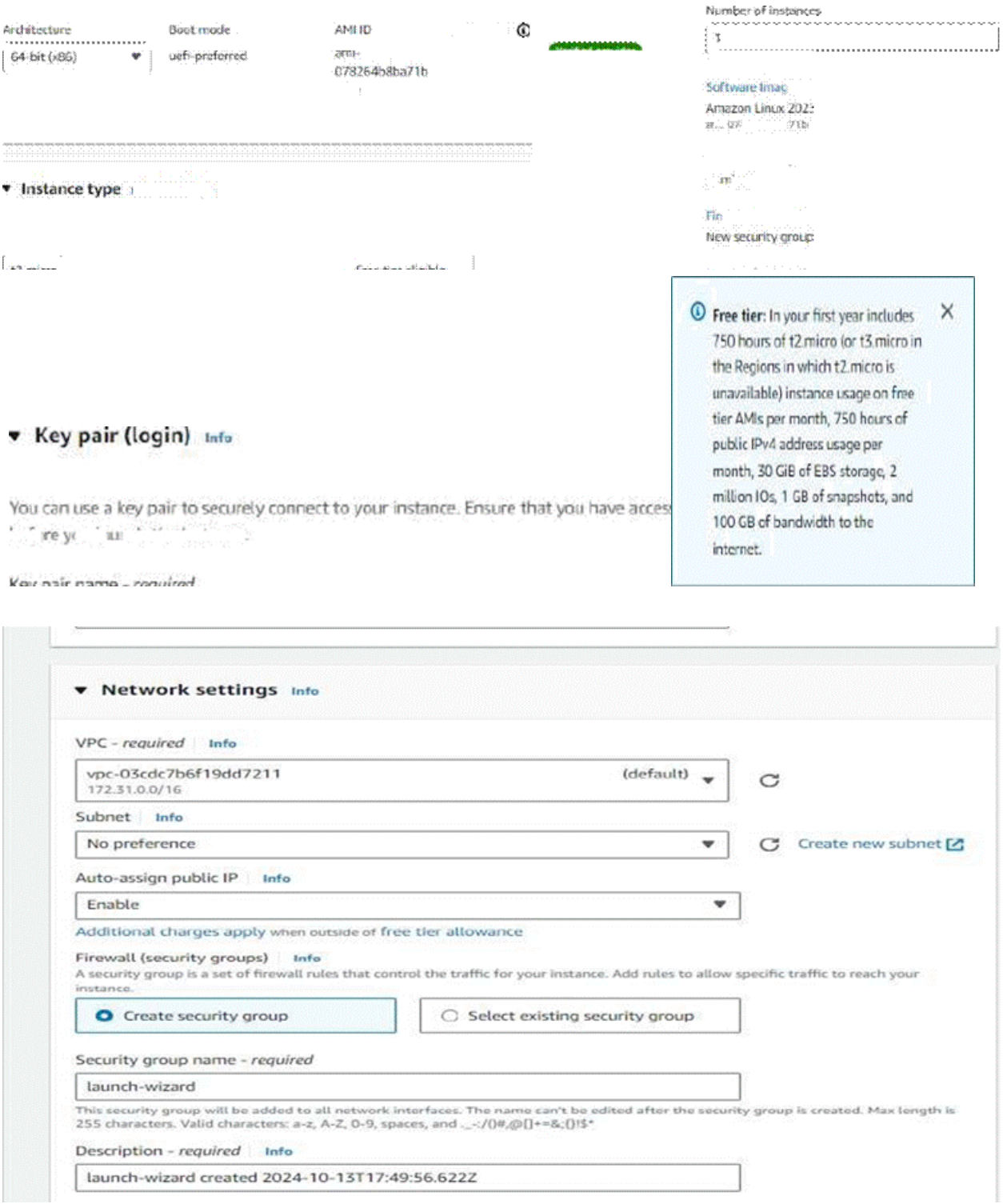


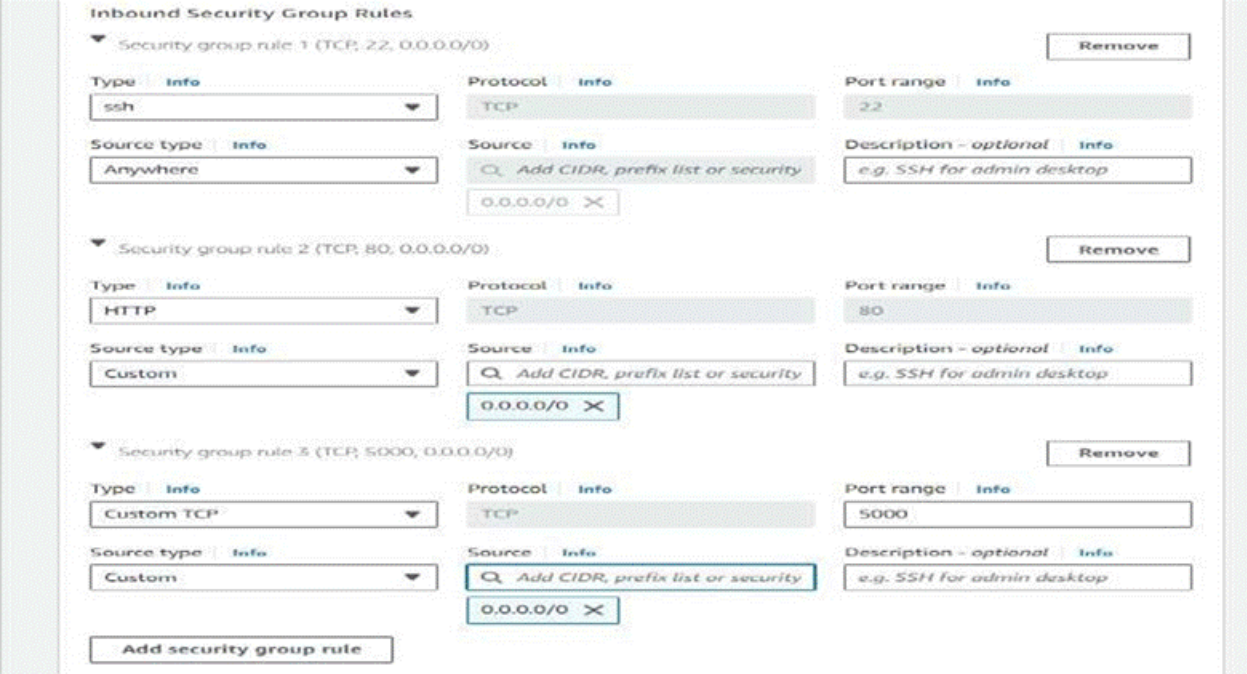
7. Create and downloadthe key pair forServeraccess.

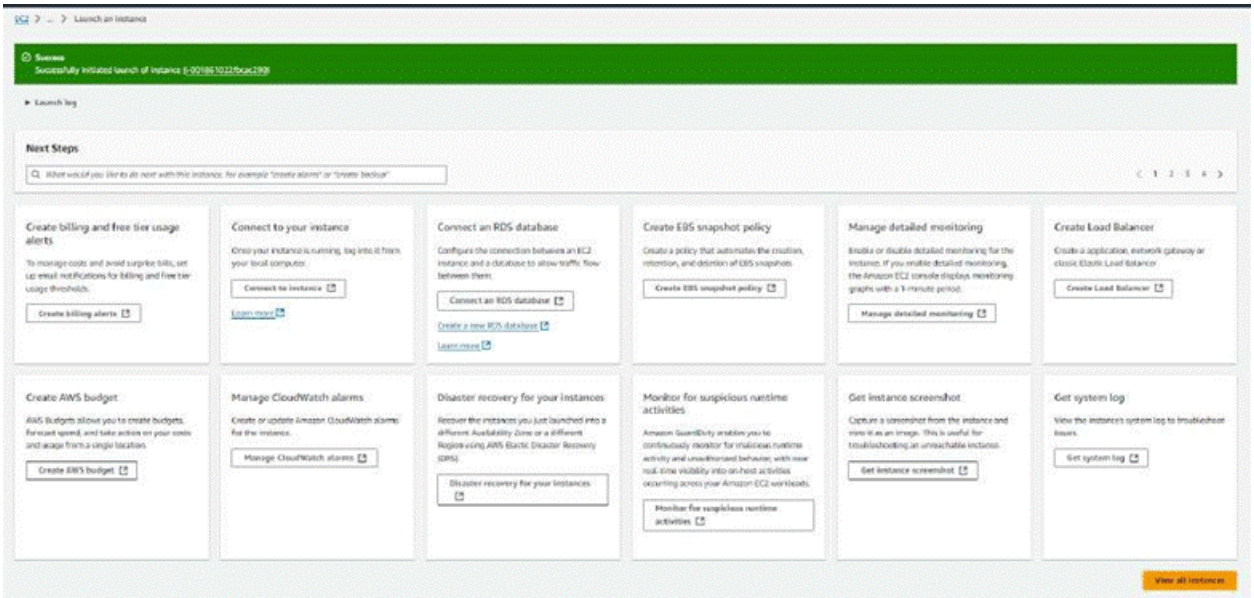




8. Activity 6.2:Configure securitygroups for HTTP,and SSH access.

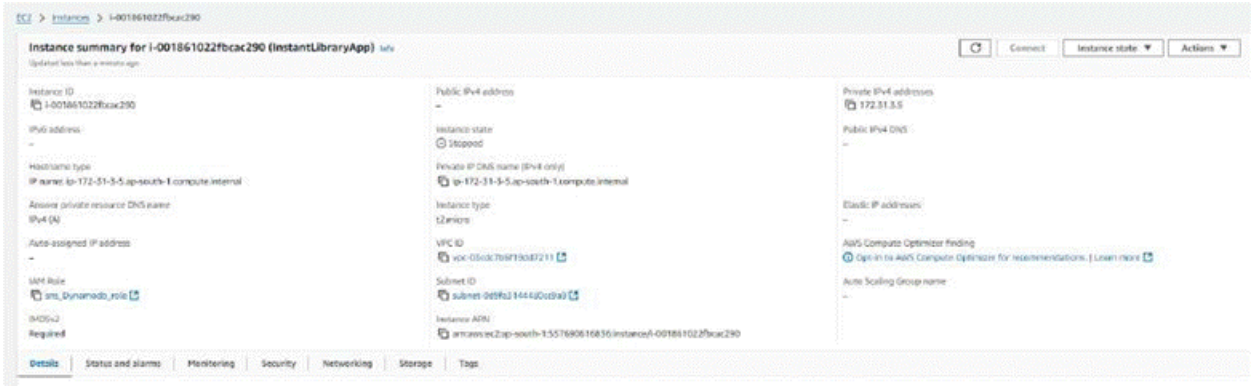

Architecture
64-bit (x86)
Boot mode
uefi-preferred
AMI ID
ami-
078264b8ba71b
Number of instances
1
Software Imag
Amazon Linux 2023
Instance type
Fir
New security group
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.
Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensure that you have acces
Key pair name - required
Network settings Info
VPC - required Info
vpc-03cdc7b6f19dd7211 (default)
172.31.0.0/16
Subnet Info
No preference
Create new subnet
Auto-assign public IP Info
Enable
Additional charges apply when outside of free tier allowance
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Create security group
Select existing security group
Security group name - required
launch-wizard
This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*
Description - required Info
launch-wizard created 2024-10-13T17:49:56.622Z



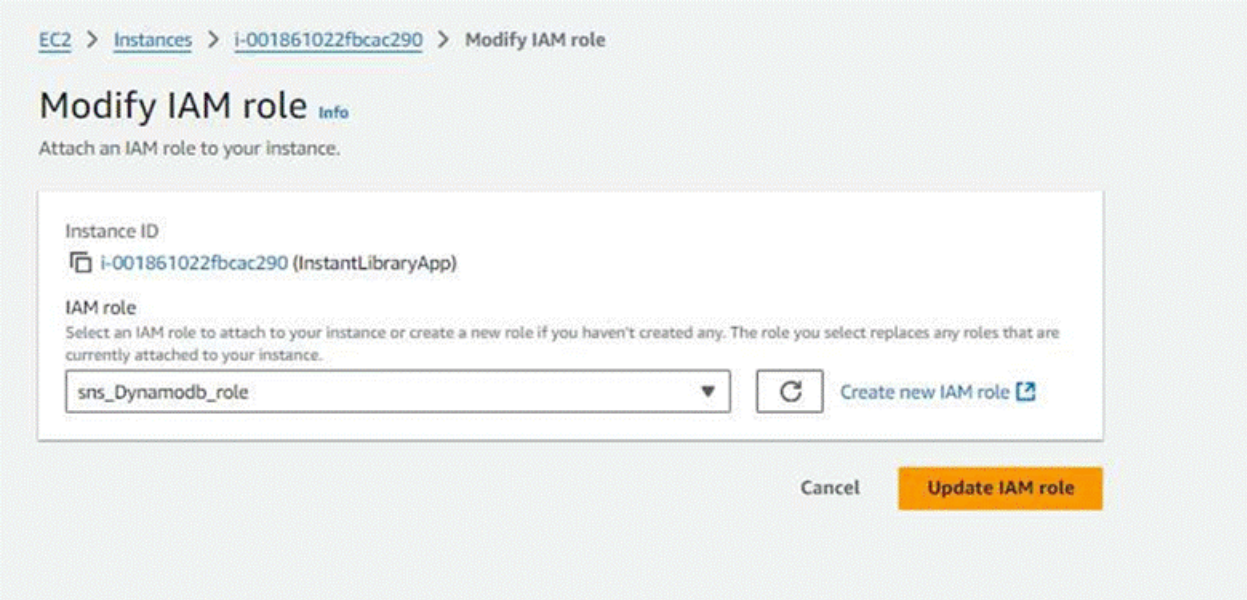


9. To connect to EC2 using EC2 Instance Connect, start by ensuring that an IAM role is attached to your EC2 instance. You can do this by selecting your instance, clicking on Actions, then navigating to Security and selecting Modify IAM Role to attach the appropriate role. After the IAM role is connected, 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 newbrowser-based terminalwill open, allowing you to access your EC2 instance directly from your browser.

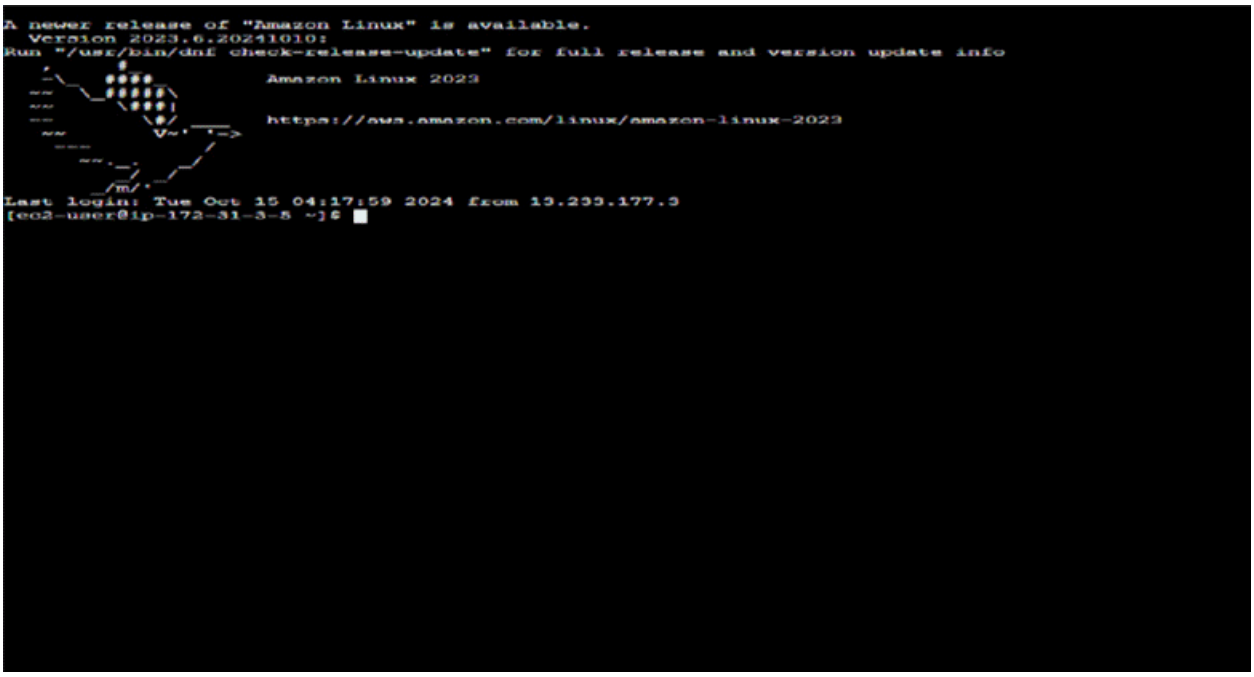

Instances (1/2) Info
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4 DNS
Public IPv4 ...
Elastic IP
IPv6 IPs
Monitoring
InstantLibrary...
i-001861022fbcac290
Stopped
t2.micro
View alarms
ap-south-1b
disabled


EC2 > Instances > i-001861022fbcac290
Instance summary for i-001861022fbcac290 (InstantLibraryApp) Info
Connect
Instance state
Actions
Instance ID
i-001861022fbcac290
Public IPv4 address
Private IPv4 addresses
172.31.3.5
Instance state
Stopped
Public IPv4 DNS
Hostname type
Private IP DNS name (IPv4 only)
Instance type
Elastic IP addresses
VPC ID
AWS Compute Optimizer finding
Subnet ID
Auto Scaling Group name
IAM Role
IMDSv2
Required
Instance ARN
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags


EC2 > Instances > i-001861022fbcac290
Instance summary for i-001861022fbcac290 (InstantLibraryApp) Info
Connect
Instance state
Actions
Instance ID
Public IPv4 address
Private IPv4 addresses
172.31.3.5
Instance state
Stopped
Public IPv4 DNS
Hostname type
Private IP DNS name (IPv4 only)
Instance type
t2.micro
Elastic IP addresses
VPC ID
AWS Compute Optimizer finding
Subnet ID
Auto Scaling Group name
IAM Role
IMDSv2
Required
Instance ARN
Connect
Manage instance state
Instance settings
Networking
Security
Image and templates
Monitor and troubleshoot
Change security groups
Get Windows password
Modify IAM role



10. Now connect theEC2 with thefiles



Milestone 7: Deployment on EC2

Activity 7.1: InstallSoftwareon 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 Projectfrom GitHub

Clone your projectrepository from GitHubinto the EC2 instanceusing Git.

Clone your Node.js project from GitHub

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

Install project dependencies

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

11. This will downloadyour project to theEC2 instance.

To navigateto the project directory, run the following command:

cd MedTrack-repo

Once inside the projectdirectory, configureand run the Nodejs application by executing the following command with elevated privileges:

Run the Flask Application

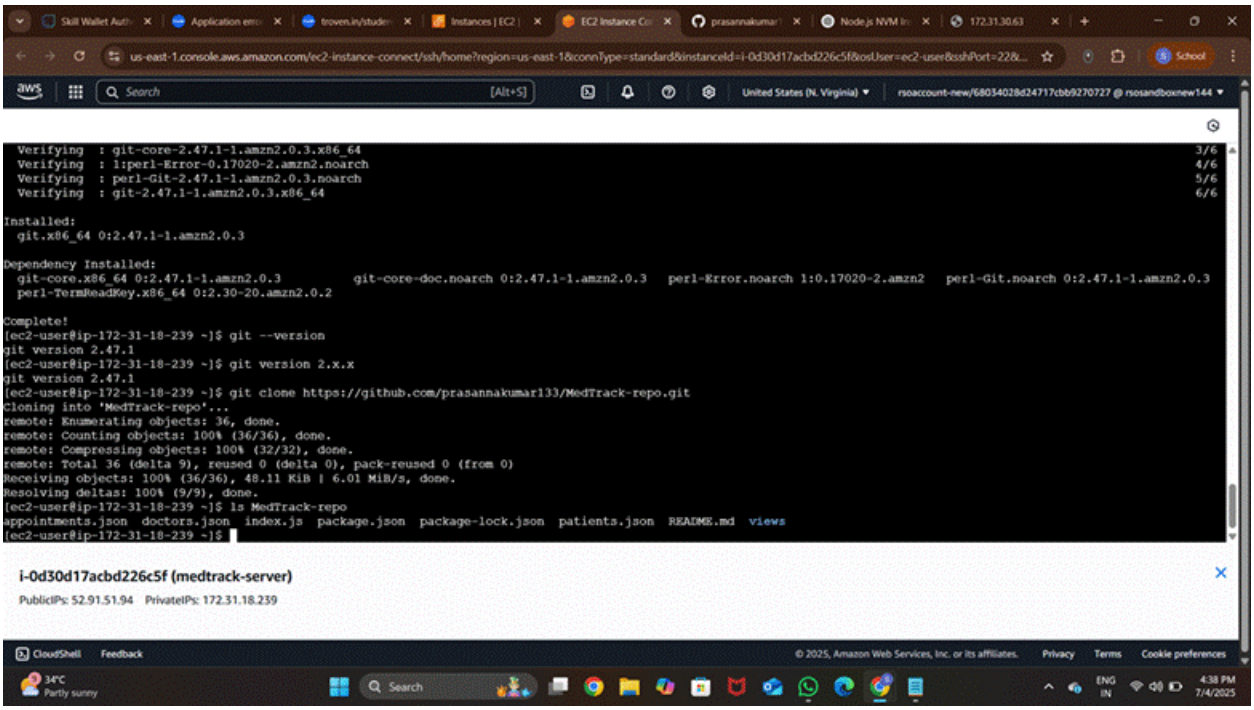
Install project dependencies

npm install

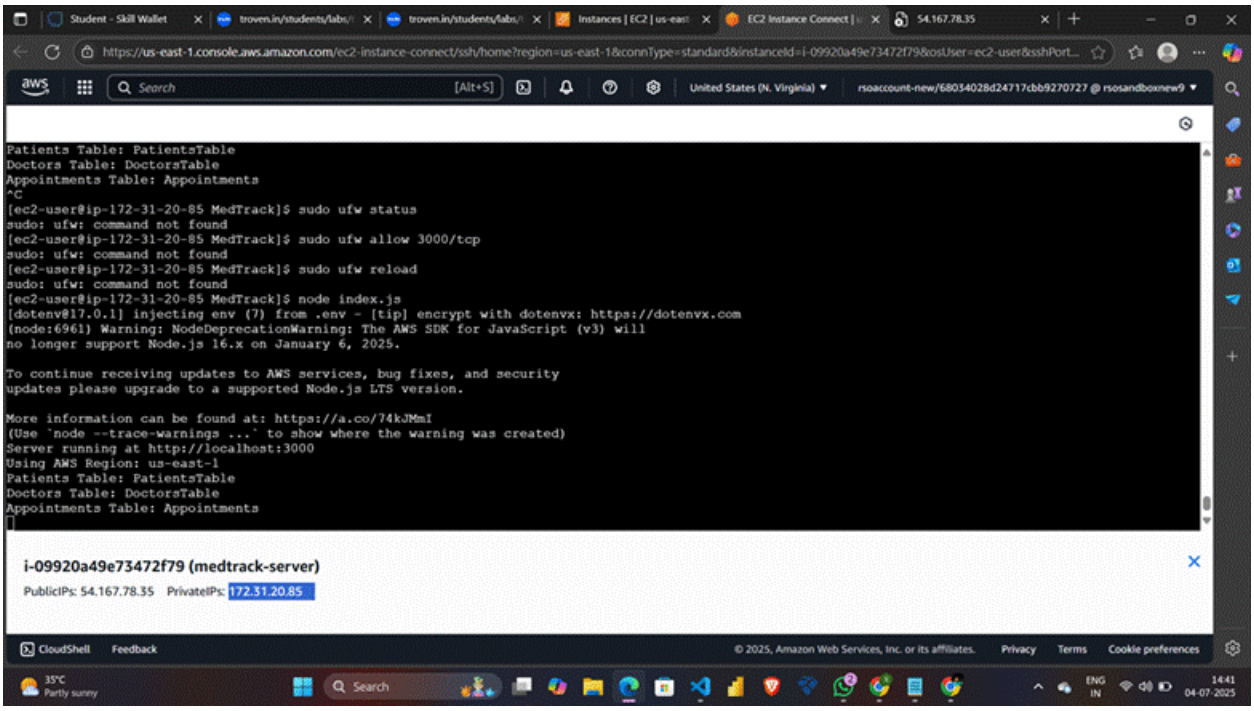
# Install PM2 globally to keep the app running
sudo npm install -g pm2

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



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

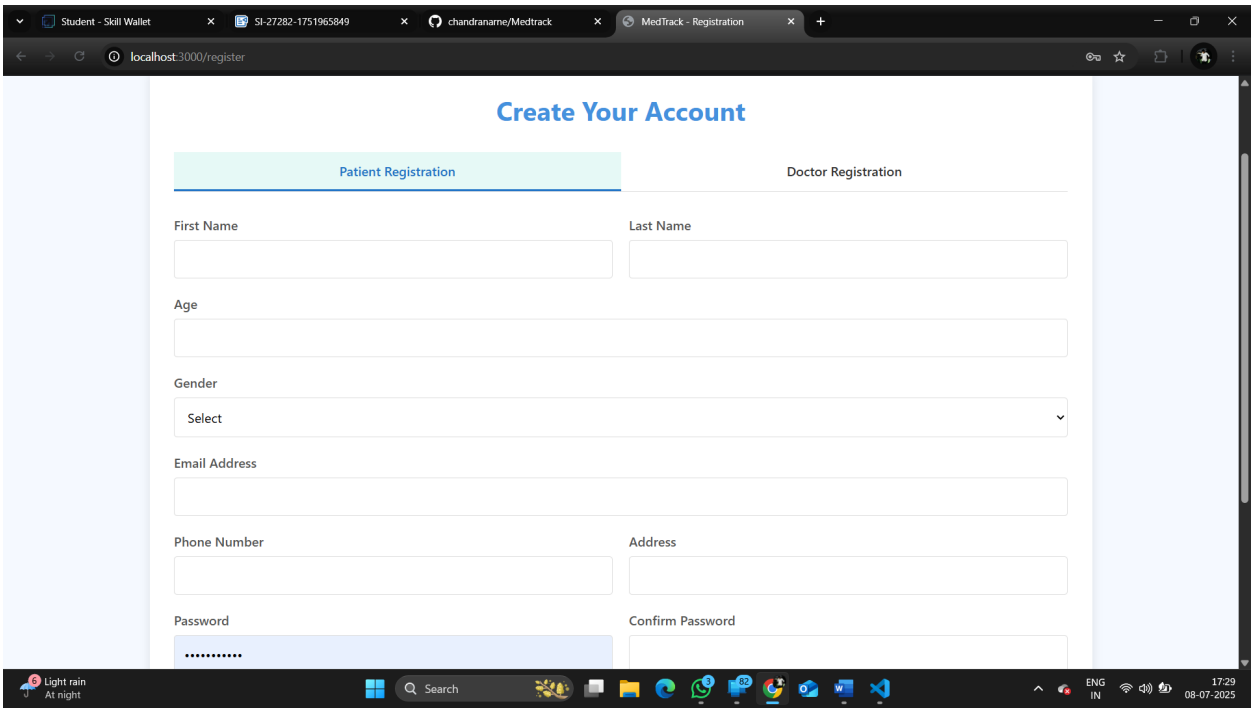


Access the websitethrough:
PublicIPs:http://52.91.51.94:3000/

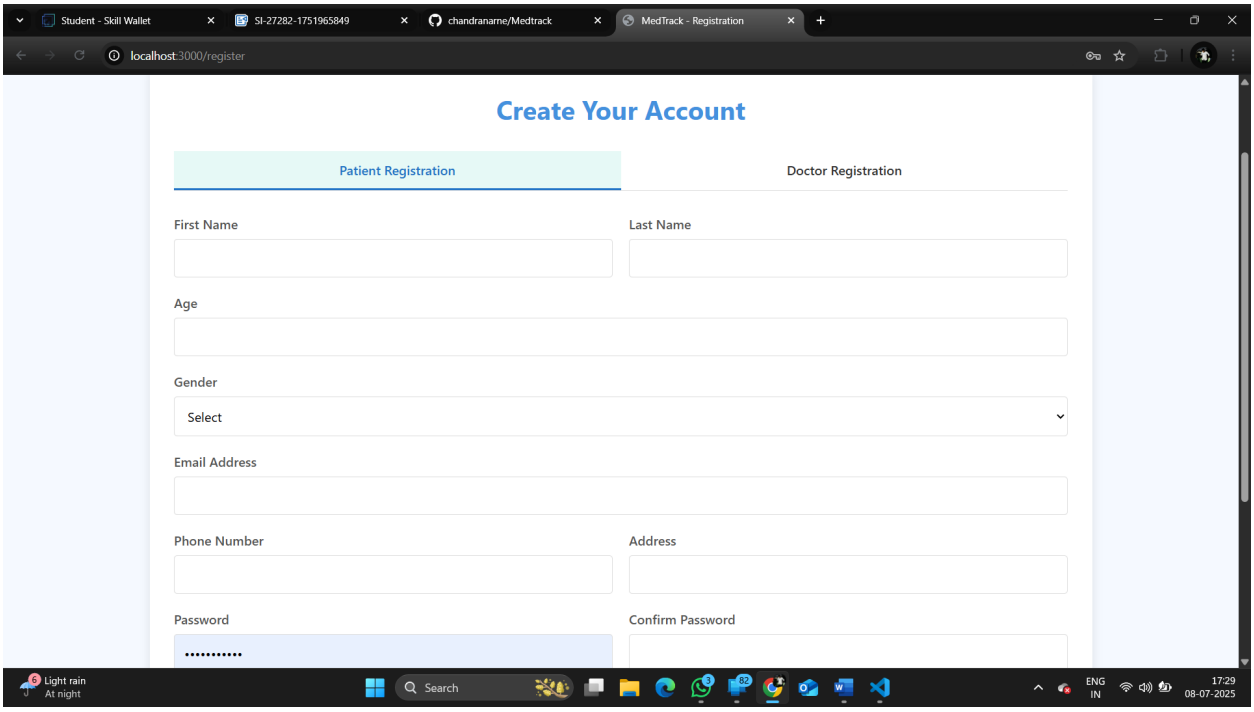
Milestone 8: Testingand Deployment

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

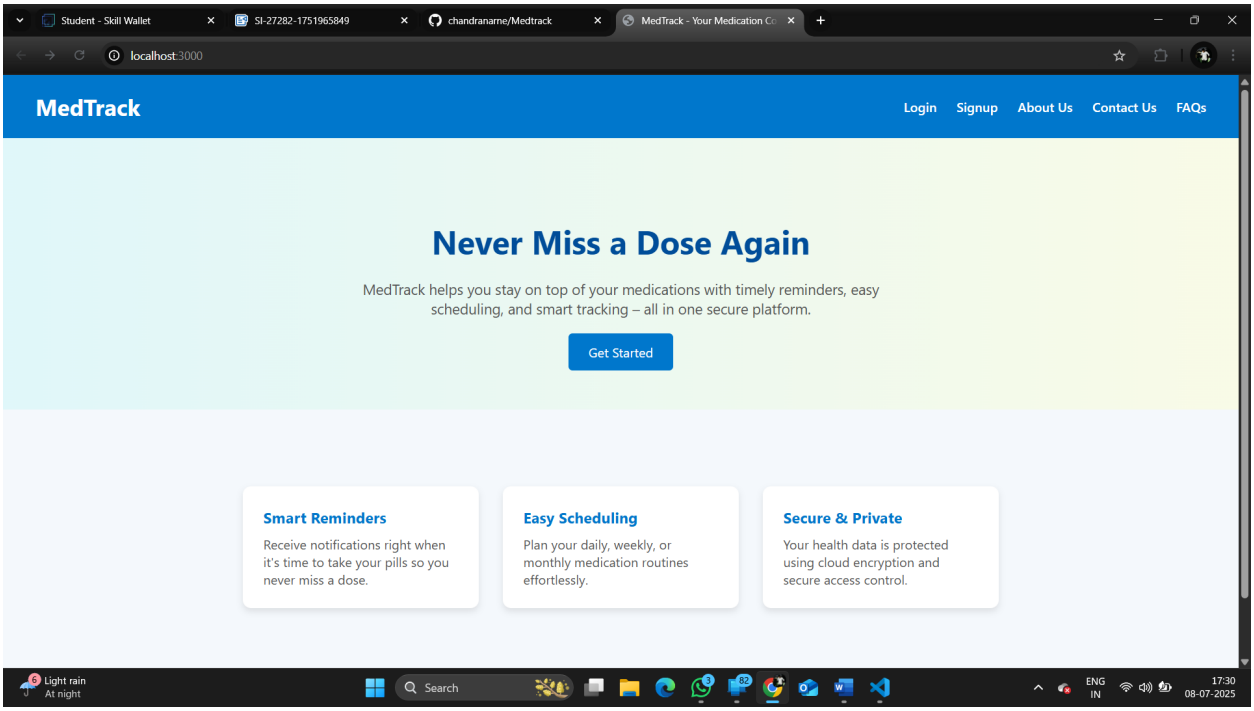
Login Page:



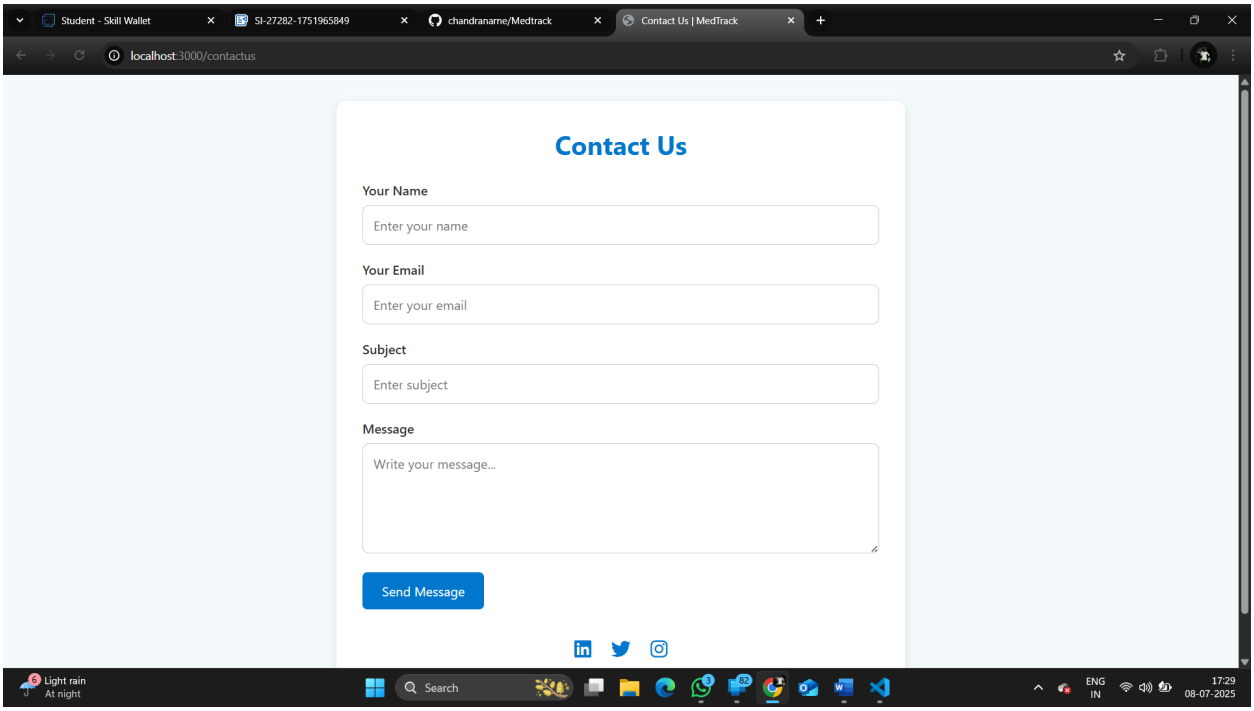
Register Page:



Home page:



ContactUs :



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.