

MedTrack: AWS Cloud-Enabled Healthcare Management System

Project Description:

In today's fast-evolving healthcare landscape, efficient communication and coordination between doctors and patients are crucial. MedTrack is a cloud-based healthcare management system that streamlines patient doctor interactions by providing a centralized platform for booking appointments, managing medical histories, and enabling diagnosis submissions. To address these challenges, the project utilizes Flask for backend development, AWS EC2 for hosting, and DynamoDB for managing data. MedTrack allows patients to register, log in, book appointments, and submit diagnosis reports online. The system ensures real-time notifications, enhancing communication between doctors and patients regarding appointments and medical submissions. Additionally, AWS Identity and Access Management (IAM) is employed to ensure secure access control to AWS resources, allowing only authorized users to access sensitive data. This cloud-based solution improves accessibility and efficiency in healthcare services for all users.

Scenario 1: Efficient Appointment Booking System for Patients

In the MedTrack system, AWS EC2 provides a reliable infrastructure to manage multiple patients accessing the platform simultaneously. For example, a patient can log in, navigate to the appointment booking page, and easily submit a request for an appointment. Flask handles backend operations, efficiently retrieving and processing user data in real-time. The cloud-based architecture allows the platform to handle a high volume of appointment requests during peak periods, ensuring smooth operation without delays.

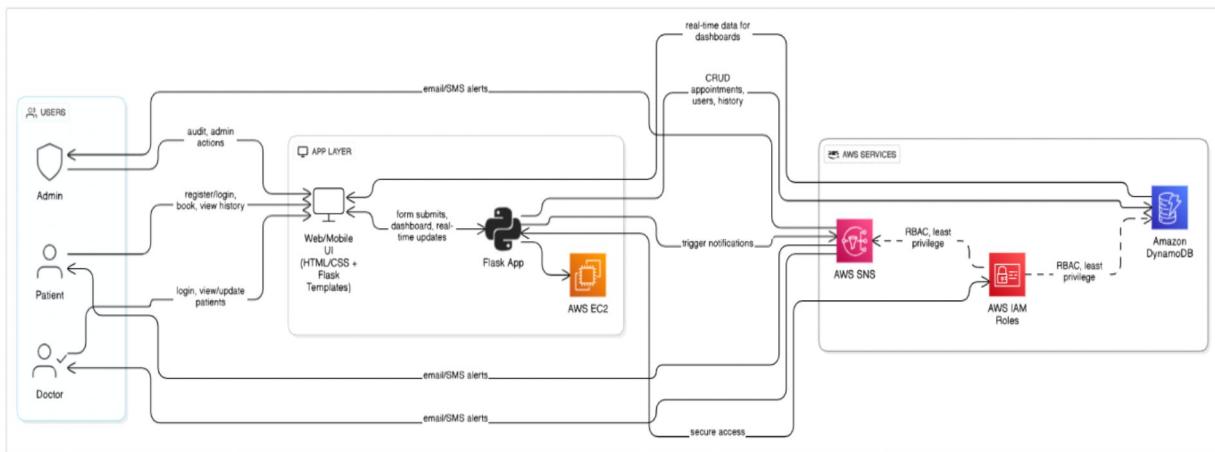
Scenario 2: Secure User Management with IAM

MedTrack utilizes AWS IAM to manage user permissions and ensure secure access to the system. For instance, when a new patient registers, an IAM user is created with specific roles and permissions to access only the features relevant to them. Doctors have their own IAM configurations, allowing them access to patient records and appointment details while maintaining strict security protocols. This setup ensures that sensitive data is accessible only to authorized users.

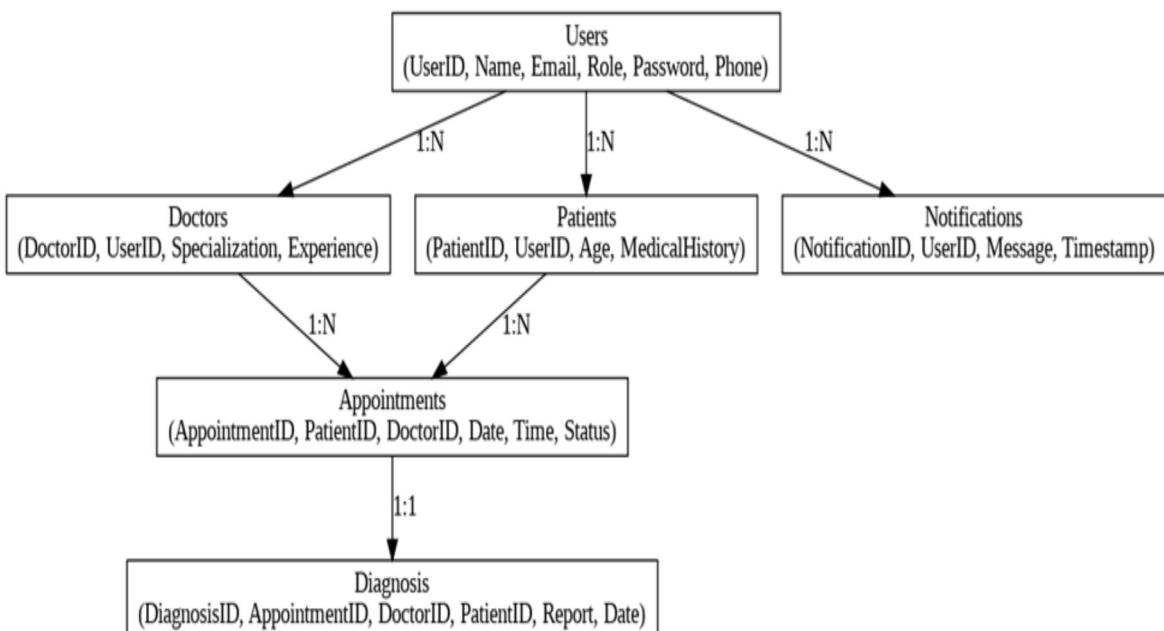
Scenario 3: Easy Access to Medical History and Resources

The MedTrack system provides doctors and patients with easy access to medical histories and relevant resources. For example, a doctor logs in to view a patient's medical history and upcoming appointments. They can quickly access, and update records as needed. Flask manages real-time data fetching from DynamoDB, while EC2 hosting ensures the platform performs seamlessly even when multiple users access it simultaneously, offering a smooth and uninterrupted user experience.

AWS ARCHITECTURE



Entity Relationship (ER) Diagram:



Pre-requisites:

- [bULgkD4LXNvR](#) HYPERLINK "https://youtu.be/gsgdAyGhV0o?si=3qgbULgkD4LXNvR" HYPERLINK "https://youtu.be/gsgdAyGhV0o?si=3qgbULgkD4LXNvR" HYPERLINK "https://youtu.be/gsgdAyGhV0o?si=3qgbULgkD4LXNvR" HYPERLINK "https://youtu.be/gsgdAyGhV0o?si=3qgbULgkD4LXNvR" Overview
- **Amazon EC2 Basics:** [EC2](#) HYPERLINK "https://youtu.be/8TlukLu11Yo?si=MUj0nEAOESRhHUIz" HYPERLINK "https://youtu.be/8TlukLu11Yo?si=MUj0nEAOESRhHUIz" HYPERLINK "https://youtu.be/8TlukLu11Yo?si=MUj0nEAOESRhHUIz" HYPERLINK "https://youtu.be/8TlukLu11Yo?si=MUj0nEAOESRhHUIz" HYPERLINK "https://youtu.be/8TlukLu11Yo?si=MUj0nEAOESRhHUIz" HYPERLINK "https://youtu.be/8TlukLu11Yo?si=MUj0nEAOESRhHUIz" Tutorial
- **DynamoDB Basics:** [DynamoDB](#) HYPERLINK "https://docs.aws.amazon.com/dynamodb" Introduction
- **SNS Overview:** [SNS](#) HYPERLINK "https://docs.aws.amazon.com/sns" HYPERLINK "https://docs.aws.amazon.com/sns" HYPERLINK "https://docs.aws.amazon.com/sns" HYPERLINK "https://docs.aws.amazon.com/sns" HYPERLINK "https://docs.aws.amazon.com/sns" HYPERLINK "https://docs.aws.amazon.com/sns" Documentation
- **Git Version Control:** [Git](#) HYPERLINK "https://git-scm.com/doc" Documentation

Project WorkFlow:

- **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

- **DynamoDB Database Creation and Setup**

Activity 2.1: Create a DynamoDB Table.

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

- **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.

- **Backend Development and Application Setup**

Activity 4.1: Develop the Backend Using Flask.

Activity 4.2: Integrate AWS Services Using boto3.

- **IAM Role Setup**

Activity 5.1: Create IAM Role

Activity 5.2: Attach Policies

- **EC2 Instance Setup**

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

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

- **Deployment on EC2**

Activity 7.1: Upload Flask Files

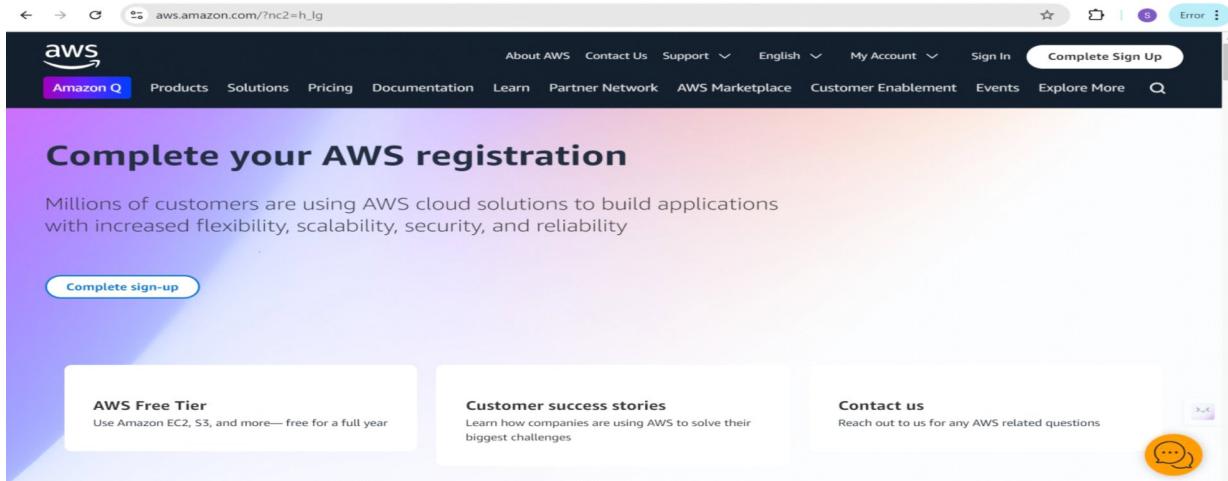
Activity 7.2: Run the Flask App

- **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 HYPERLINK](https://aws.amazon.com/console/) [" HYPERLINK "](https://aws.amazon.com/console/) [https://aws.amazon.com/console/" HYPERLINK](https://aws.amazon.com/console/) [" https://aws.amazon.com/console/" HYPERLINK](https://aws.amazon.com/console/).

Milestone 2: DynamoDB Database Creation and Setup

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

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

The screenshot shows the AWS Services search results for 'dynamoDB'. The search bar at the top contains 'dynamoDB'. Below it, a sidebar on the left lists various AWS services and features. The main content area displays a list of services under the heading 'Services'.

- DynamoDB** (Managed NoSQL Database)
- Amazon DocumentDB** (Fully-managed MongoDB-compatible database service)
- CloudFront** (Global Content Delivery Network)
- Athena** (Serverless interactive analytics service)

Below this, there's a section titled 'Features' with 'Settings' and 'Clusters' options.

The screenshot shows the DynamoDB Dashboard. The left sidebar includes links for 'Tables', 'Explore items', 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', 'Integrations', 'Reserved capacity', and 'Settings'. Under 'DAX', there are links for 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main dashboard area shows sections for 'Alarms' (0), 'DAX clusters' (0), and a 'Create resources' section for creating a new table or DAX cluster.

The screenshot shows the 'Tables' page in the DynamoDB console. The left sidebar has links for 'Dashboard' and 'Tables'. The main table view shows a message: 'You have no tables in this account in this AWS Region.' There is a 'Create table' button 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.



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.



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.



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	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.

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.

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 hosts for scalability and availability.

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



1 to 255 characters and case sensitive.

Table settings



Default settings

This is the easiest way to create your table. You can modify it later.



Customize settings

For those advanced features to make DynamoDB work.

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

Search results for 'sns'

Services

Features

Resources **New**

Documentation

Knowledge articles

Marketplace

Blog posts

Events

Tutorials

Services

Show more ►

Simple Notification Service ☆
SNS managed message topics for Pub/Sub

Route 53 Resolver
Resolve DNS queries in your Amazon VPC and on-premises network.

Route 53 ☆
Scalable DNS and Domain Name Registration

AWS End User Messaging ☆
Engage your customers across multiple communication channels

Features

Show more ►

Events
ElastiCache feature

SMS
AWS End User Messaging feature

Hosted zones
Route 53 feature

Amazon SNS

Dashboard

Topics

Subscriptions

Mobile

Push notifications

Text messaging (SMS)

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

Application Integration

Amazon Simple Notification Service

Pub/sub messaging for microservices and serverless applications.

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

Create topic

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

Next step

Start with an overview

Pricing

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

The screenshot shows the 'Topics' section of the Amazon SNS console. A banner at the top indicates a new feature: 'Amazon SNS now supports in-place message archiving and replay for FIFO topics. Learn more'. Below the banner, the 'Topics (0)' list is displayed with a search bar and buttons for 'Edit', 'Delete', 'Publish message', and 'Create topic'. A note says 'No topics' and 'To get started, create a topic.' with a 'Create topic' button.

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

The screenshot shows the 'Create topic' wizard. The title is 'Create topic' under 'Amazon SNS > Topics > Create topic'. The 'Details' tab is selected. Under 'Type', 'Standard' is chosen (indicated by a blue circle). The 'FIFO (first-in, first-out)' option is also shown with its characteristics:

- Strictly-preserved message ordering
- Exactly-once message delivery
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS

Below the type selection, the 'Name' field contains 'BookRequestNotifications'. A note says 'Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).' The 'Display name - optional' field contains 'My Topic'. A note says 'To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.' A note also says '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**.

The screenshot shows the 'Topics' section of the Amazon SNS console. A green banner at the top indicates that 'BookRequestNotifications' was created successfully. Below the banner, the topic details are shown: Name is 'BookRequestNotifications', ARN is 'arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications', and Type is 'Standard'. There are tabs for 'Subscriptions', 'Access policy', 'Data protection policy', 'Delivery policy (HTTP/S)', 'Delivery status logging', 'Encryption', 'Tags', and 'Integrations'. The 'Subscriptions' tab shows a table with one row: 'No subscriptions found'. A 'Create subscription' button is visible at the bottom of the table.

- **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.

Create subscription

Details

Topic ARN
 X

Protocol
 The type of endpoint to subscribe
 ▼

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

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

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

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

Cancel Create subscription

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

Dashboard Topics Subscriptions ▼ Mobile Push notifications Text messaging (SMS)

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

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

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

Details

ARN	Status
arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4	Pending confirmation
Endpoint	Protocol
instantlibrary2@gmail.com	EMAIL
Topic	
BookRequestNotifications	
Subscription Principal	
arn:aws:iam::557690616836:root	

[Subscription Filter policy](#) [Redrive policy \(dead-letter queue\)](#)

Subscription filter policy Info
 This policy filters the messages that a subscriber receives.

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

- After subscription request for the mail confirmation

The screenshot shows the AWS SNS Topics page. A green banner at the top indicates a 'New Feature' where SNS now supports in-place message archiving and replay for FIFO topics. Below this, a message says 'Confirmation request was sent successfully.' with the ARN of the subscription. The main area shows a topic named 'BookRequestNotifications' with details like Name, ARN, and Type. A table below lists 'Subscriptions (2)', showing one entry for 'instantlibrary2@gmail.com' in 'Pending confirmation' status. Buttons for 'Edit', 'Delete', 'Request confirmation', 'Confirm subscription', and 'Create subscription' are visible.

- 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>

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](#)



Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:

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

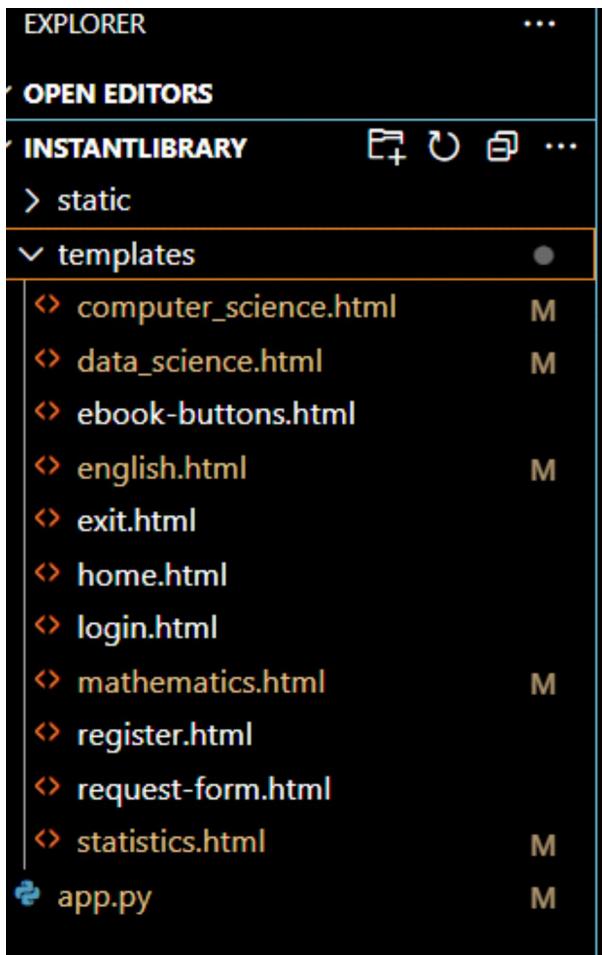
If it was not your intention to subscribe, [click here to unsubscribe](#).

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

The screenshot shows the AWS SNS console. On the left, there is a navigation sidebar with links like Dashboard, Topics, Subscriptions, Mobile, Push notifications, and Text messaging (SMS). The main area displays the 'BookRequestNotifications' topic details. The 'Details' section includes fields for Name (BookRequestNotifications), Display name (-), ARN (arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications), Topic owner (557690616836), and Type (Standard). Below this, there are tabs for Subscriptions, Access policy, Data protection policy, Delivery policy (HTTP/S), Delivery status logging, Encryption, Tags, and Integrations. The 'Subscriptions' tab is selected, showing a table with two entries. The first entry is for an endpoint instantlibrary2@gmail.com, which is confirmed and uses the EMAIL protocol. There are buttons for Edit, Delete, Request confirmation, Confirm subscription, and Create subscription at the top of the subscriptions table.

Milestone 4: Backend Development and Application Setup

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



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

Description of the code :

- **Flask App Initialization**

```
from flask import Flask, render_template, request, redirect, url_for
import boto3
from boto3.dynamodb.conditions import Key
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from bcrypt import hashpw, gensalt, checkpw
```

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

```
app = Flask(__name__)
```

Description: initialize the Flask application instance using Flask(__name__) to start building the web app.

- Dynamodb Setup:

```
# Initialize DynamoDB resource
dynamodb = boto3.resource('dynamodb', region_name='ap-south-1')

# DynamoDB Tables
users_table = dynamodb.Table('Users') # Ensure the 'Users' table
requests_table = dynamodb.Table('Requests') # Ensure the 'Reques
```

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

- SNS Connection

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

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

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

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

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

- Routes for Web Pages

- Home Route:

```

# Home route redirects to Registration page
@app.route('/')
def home():
    return redirect(url_for('register'))

```

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

- **Register Route:**

```

# Registration Page
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        password = request.form['password']
        confirm_password = request.form['confirm_password']

        # Basic Validation: Ensure all fields are filled
        if not name or not email or not password or not confirm_password:
            return "All fields are mandatory! Please fill out the entire form."
        if password != confirm_password:
            return "Passwords do not match! Please try again."

        # Check if user already exists
        response = users_table.get_item(Key={'email': email})
        if 'Item' in response:
            return "User already exists! Please log in."

        # Hash the password
        hashed_password = hashpw(password.encode('utf-8'), gensalt()).decode('utf-8')

        # Store user in DynamoDB with login_count initialized to 0
        users_table.put_item(
            Item={
                'email': email,
                'name': name,
                'password': hashed_password,
                'login_count': 0
            }
        )

        # Send SNS notification for new registration
        sns.publish(
            TopicArn=sns_topic_arn,
            Message=f'New user registered: {name} ({email})',
            Subject='New User Registration'
        )

    return redirect(url_for('login'))
    return render_template('register.html')

```

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):**

```
# Login Page
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']

        # Basic Validation: Ensure both fields are filled
        if not email or not password:
            return "Please enter both email and password."

        # Fetch user data from DynamoDB
        response = users_table.get_item(Key={'email': email})
        user = response.get('Item')

        if not user or not checkpw(password.encode('utf-8'), user['password'].encode('utf-8')):
            return "Incorrect email or password! Please try again."

        # Update login count
        users_table.update_item(
            Key={'email': email},
            UpdateExpression='SET login_count = login_count + :inc',
            ExpressionAttributeValues={':inc': 1}
        )

        # Successful login
        return redirect(url_for('home_page'))
    return render_template('login.html')
```

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:**

```
# Home Page with E-Books, Request Books, and Exit
@app.route('/home-page')
def home_page():
    return render_template('home.html')

# E-Books Page (Dropdown Selection for Course and Subject)
@app.route('/ebook-buttons', methods=['GET', 'POST'])
def ebook_buttons():
    if request.method == 'POST':
        subject = request.form['subject']
        return redirect(url_for('subject_page', subject=subject))
    return render_template('ebook-buttons.html')

# Subject Page (Example with Mathematics)
@app.route('/<subject>.html')
def subject_page(subject):
    return render_template(f'{subject}.html')
```

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 like Mathematics or English.

- **Request Routes:**

```

# Book Request Form Page
@app.route('/request-form', methods=['GET', 'POST'])
def request_form():
    if request.method == 'POST':
        # Retrieve form data from the POST request
        email = request.form['email'] # Capture email to send thank-you note
        name = request.form['name']
        year = request.form['year']
        semester = request.form['semester']
        roll_no = request.form['roll-no']
        subject = request.form['subject']
        book_name = request.form['book-name']
        description = request.form['description']

        # Store book request in DynamoDB along with the user email
        requests_table.put_item(
            Item={
                'email': email,
                'roll_no': roll_no,
                'name': name,
                'year': year,
                'semester': semester,
                'subject': subject,
                'book_name': book_name,
                'description': description
            }
        )

        # Send a thank-you email to the requesting user
        thank_you_message = f"Dear {name},\n\nThank you for submitting a book request for '{book_name}'. We will send_email(email, "Thank You for Your Book Request", thank_you_message)

        # Send an email to the Instant Library admin with the book request details
        admin_message = f"User {name} ({email}) has requested the book '{book_name}'.\n\nDetails:\nYear: {year}\nsend_email("instantlibrary@gmail.com", "New Book Request", admin_message)

        return "<h3>Book request submitted successfully! We will get back to you soon.</h3>"

    # Render the request form for GET requests
    return render_template('request-form.html')

```

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:

```

# Exit Page
@app.route('/exit')
def exit_page():
    return render_template('exit.html')

```

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

Deployment Code:

```

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=True)

```

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 search results for 'iam'. The top navigation bar has 'Services' selected. Below it, the search bar shows 'iam'. The results list includes:

- 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.
- AWS App Mesh**: Easily monitor and control microservices.

Below this, the 'Identity and Access Management (IAM)' dashboard is shown, with a link to 'Roles'.

Create Role - Step 1: Select trusted entity

This step allows selecting a trusted entity type:

- AWS service: Allows AWS services like S3, Lambda, or others to perform actions in this account.
- AWS account: Allows actions in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity: Allows users federated by the specified external web identity provider to assume the role to perform actions in this account.

Create Role - Step 2: Add permissions

This step shows the 'Add permissions' section:

Add permissions info

Permissions policies (1/955) info

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

Filter by Type: All types

Policy name: AmazonDynamoDB

Selected policies:

- AmazonDynamoDBFullAccess
- AmazonDynamoDBReadOnlyAccess

Set permissions boundary - optional

Cancel Previous Next

• Activity 5.2: Attach Policies.

Attach the following policies to the role:

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

The screenshot shows the 'Add permissions' step of creating a new IAM role. On the left, a sidebar lists 'Step 1: Select trusted entity', 'Step 2: Add permissions' (which is currently selected), and 'Step 3: Name, review, and create'. The main area is titled 'Permissions policies (2/955)' and contains a search bar and a filter dropdown set to 'All types'. A table lists five AWS managed policies:

Policy name	Type
AmazonSNSFullAccess	AWS managed
AmazonSNSReadOnlyAccess	AWS managed
AmazonSNSRole	AWS managed
AWSLambdaBasicExecutionRole	AWS managed
AWSLambdaDevice Defender Publish Findings To SNS Integration Action	AWS managed

 Below the table is a link to 'Set permissions boundary - optional'. At the bottom right are 'Cancel', 'Previous', and 'Next' buttons, with 'Next' highlighted in orange.

The screenshot shows the 'Name, review, and create' step. It includes sections for 'Role details' (Role name: sns_Dynamodb_role, Description: Allows EC2 instances to call AWS services on your behalf.), 'Step 1: Select trusted entities' (Trust policy: JSON code for AWS Lambda), 'Step 2: Add permissions' (Permissions policy summary: AmazonSNSFullAccess, AmazonDynamoDBFullAccess), and 'Step 3: Add tags' (No tags associated with the resource). At the bottom right are 'Cancel', 'Previous', and 'Create role' buttons, with 'Create role' highlighted in orange.

The screenshot shows the 'sns_Dynamodb_role' summary page. It includes a 'Summary' section with details like Creation date (October 13, 2024, 23:06 (UTC+05:30)), Last activity (6 days ago), ARN (arn:aws:iam::557690616836:role/sns_Dynamodb_role), and Maximum session duration (1 hour). Below this are tabs for 'Permissions', 'Trust relationships', 'Tags', 'Last Accessed', and 'Revoke sessions'. The 'Permissions' tab shows 'Permissions policies (2)' with links to 'AmazonDynamoDBFullAccess' and 'AmazonSNSFullAccess'. At the bottom right are 'Edit', 'Delete', 'Simulate', 'Remove', and 'Add permissions' buttons.

Milestone 6: EC2 Instance Setup

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

The screenshot shows a GitHub repository interface. At the top, there's a list of files with their commit history:

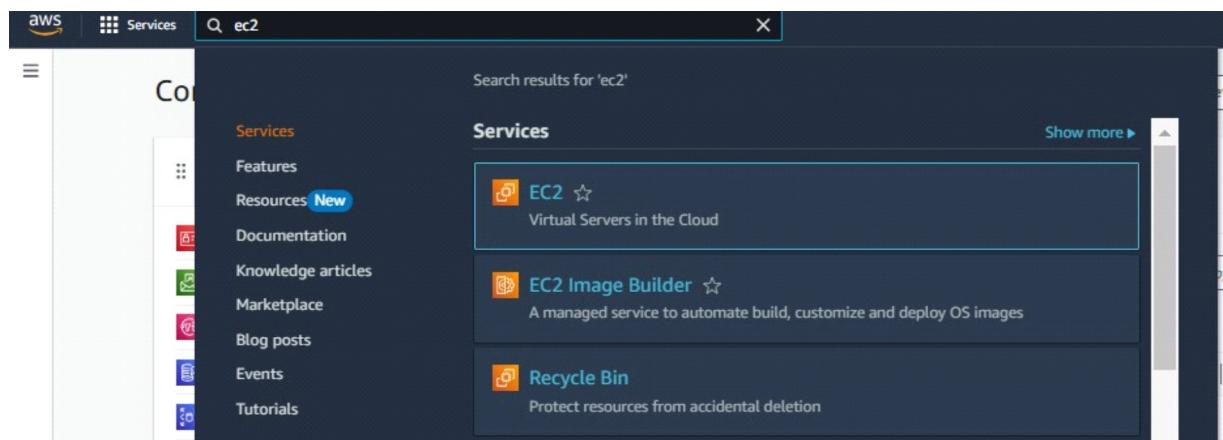
File	Commit
static	Initial commit
templates	Update statistics.html
app.py	Update app.py

Below this, there are two tabs: "Local" and "Codespaces". Under "Local", there are three cloning methods: "Clone" (with HTTPS, SSH, and GitHub CLI options), a "Clone using the web URL" button with a clipboard icon, and links for "Open with GitHub Desktop" and "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 page. On the left, there's a sidebar with options like EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, and Savings Plans. The main area has tabs for Instances (Info) and a search bar. At the top right are buttons for Last updated, Connect, Instance state, Actions, and Launch instances. Below that is a table header with columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. A message says 'No instances' and 'You do not have any instances in this region'. At the bottom right is a 'Launch instances' button.

EC2 > Instances > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: InstantLibraryApp [Add additional tags](#)

Summary

Number of instances Info: 1

Software Image (AMI): Amazon Linux 2023 AMI 2023.5.2... [read more](#)
ami-078264b8ba71bc45e

Virtual server type (instance type): t2.micro

Firewall (security group):

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

The screenshot shows the AWS AMI selection interface. It displays several AMI icons: Amazon Linux, macOS, Ubuntu, Windows, and Red Hat. To the right is a search icon and a link to 'Browse more AMIs'. Below the icons, it says 'Including AMIs from AWS, Marketplace and the Community'.

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI Free tier eligible

ami-02b49a24cfb95941c (64-bit (x86), uefi-preferred) / ami-04ad8c7fcc828fad4 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

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

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

Verified provider

- Create and download the key pair for Server access.

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

Instance type

t2.micro

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

Free tier eligible



All generations

[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) [Info](#)

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

Key pair name - required

Select



[Create new key pair](#)

Create key pair



Key pair name

Key pairs allow you to connect to your instance securely.

InstantLibrary

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type



RSA

RSA encrypted private and public key pair



ED25519

ED25519 encrypted private and public key pair

Private key file format



.pem

For use with OpenSSH



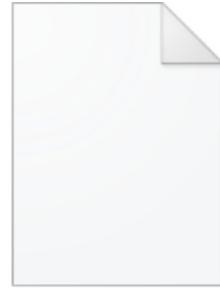
.ppk

For use with PuTTY

⚠ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#) ↗

[Cancel](#)

[Create key pair](#)



InstantLibrary.pem

The screenshot shows the AWS Launch Wizard interface for creating a new Amazon Linux 2023 instance. The 'Instance type' section is selected, showing a dropdown for 't2.micro'. Other options like 't3.micro' and 'All generations' are also available. A note about additional costs for pre-installed software is present. The 'Summary' section on the right provides an overview of the configuration, including the number of instances (1), AMI (Amazon Linux 2023 AMI 2023.5.2...), instance type (t2.micro), and storage (1 volume(s) - 8 GiB). A tooltip for 'Free tier' explains the included benefits for the first year. Finally, there are 'Cancel', 'Preview code', and 'Launch instance' buttons.

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

The screenshot shows the 'Network settings' configuration for the new instance. It includes fields for VPC (set to 'vpc-03cdc7b6f19dd7211'), Subnet ('No preference'), and Auto-assign public IP ('Enable'). A note about additional charges for public IP usage is shown. Under 'Firewall (security groups)', there are two options: 'Create security group' (selected) and 'Select existing security group'. A security group name 'launch-wizard' is specified. The 'Description' field notes the creation date and time: 'launched 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		

Remove

▼ 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		

Remove

▼ 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		

Remove

[Add security group rule](#)

EC2 > Launch an instance

Success Successfully initiated launch of instance i-00186102fbcc290

Launch log

Next Steps

Q. What would you like to do next with this instance, for example "create alarm" or "create backup"

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 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-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 unreachable 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

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring	Security
<input checked="" type="checkbox"/> InstantLibrary...	i-001861022fbcac290	<input type="radio"/> Stopped	t2.micro	-	View alarms +	ap-south-1b	-	-	-	-	disabled	launch-wi...

[EC2](#) > [Instances](#) > i-001861022fbcac290

Instance summary for i-001861022fbcac290 (InstantLibraryApp) [Info](#)

Updated less than a minute ago

Instance ID <input type="checkbox"/> i-001861022fbcac290	Public IPv4 address -	Private IPv4 addresses <input type="checkbox"/> 172.31.3.5
IPv6 address -	Instance state <input type="radio"/> Stopped	Public IPv4 DNS -
Hostname type IP name: ip-172-31-5.ap-south-1.compute.internal	Private IP DNS name (IPv4 only) <input type="checkbox"/> ip-172-31-5.ap-south-1.compute.internal	Elastic IP addresses -
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more [?]
Auto-assigned IP address -	VPC ID <input type="checkbox"/> vpc-03cdc7b6f19dd7211	Auto Scaling Group name -
IAM Role <input type="checkbox"/> sns_Dynamodb_role	Subnet ID <input type="checkbox"/> subnet-0d9fa3144480cc9a9	
IMDSv2 Required	Instance ARN <input type="checkbox"/> arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290	

[Details](#) | [Status and alarms](#) | [Monitoring](#) | [Security](#) | [Networking](#) | [Storage](#) | [Tags](#)

[EC2](#) > [Instances](#) > i-001861022fbcac290

Instance summary for i-001861022fbcac290 (InstantLibraryApp) [Info](#)

Updated less than a minute ago

Instance ID <input type="checkbox"/> i-001861022fbcac290	Public IPv4 address -	Private IPv4 addresses <input type="checkbox"/> 172.31.3.5
IPv6 address -	Instance state <input type="radio"/> Stopped	Public IPv4 DNS -
Hostname type IP name: ip-172-31-5.ap-south-1.compute.internal	Private IP DNS name (IPv4 only) <input type="checkbox"/> ip-172-31-5.ap-south-1.compute.internal	Change security groups Get Windows password
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	Modify IAM role
Auto-assigned IP address -	VPC ID <input type="checkbox"/> vpc-03cdc7b6f19dd7211	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more [?]
IAM Role <input type="checkbox"/> sns_Dynamodb_role	Subnet ID <input type="checkbox"/> subnet-0d9fa3144480cc9a9	Auto Scaling Group name -
IMDSv2 Required	Instance ARN <input type="checkbox"/> arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290	

[EC2](#) > [Instances](#) > i-001861022fbcac290 > [Modify IAM role](#)

Modify IAM role [Info](#)

Attach an IAM role to your instance.

Instance ID
 i-001861022fbcac290 (InstantLibraryApp)

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

[Create new IAM role](#) [\[?\]](#)

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

- Now connect the EC2 with the files

Connect to instance Info

Connect to your instance i-001861022fbcac290 (InstantLibraryApp) using any of these options

EC2 Instance Connect Session Manager SSH client 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](#).

Instance ID
 i-001861022fbcac290 (InstantLibraryApp)

Connection Type

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

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

Public IPv4 address
 13.200.229.59

IPv6 address

Username
 Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.

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
#
# Amazon Linux 2023
#
# https://aws.amazon.com/linux/amazon-linux-2023
#
Last login: Tue Oct 15 04:17:59 2024 from 13.233.177.3
[ec2-user@ip-172-31-3-5 ~]$ █
```

i-001861022fbcac290 (InstantLibraryApp)
 PublicIPs: 13.201.74.42 PrivateIPs: 172.31.3.5

Milestone 7: Deployment on EC2

Activity 7.1: Install Software on the EC2 Instance

Install Python3,

Flask,

and Git:

On

Amazon

Linux 2:

```
sudo yum update -y  
sudo yum install  
python3 git sudo  
pip3 install flask  
boto3
```

Verify Installations:

f

l

a

s

k

-

-

v

e

r

s

i

o

n

g

i

t

-
-
v
e
r
s
i
o
n

Activity 7.2:Clone Your Flask Project from GitHub

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

Run: 'git clone <https://github.com/your-github-username/your-repository>- HYPERLINK "<https://github.com/your-github-username/your-repository-name.git>" HYPERLINK "<https://github.com/your-github-username/your-repository-name.git>" name.git'

Note: change your-github-username and your-repository-name with

your credentials here: 'git clone

<https://github.com/Mahalakshmi13-creater/Medtrack.git>

This is download your project to the EC2 instance

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

cd InstantLibrary

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

Run the Flask Application

sudo flask run --host=0.0.0.0 --port=80

```

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
   V- '-->
   / \
  /_/
 /m/
Last login: Tue Oct 15 04:17:59 2024 from 13.233.177.3
[ec2-user@ip-172-31-3-5 ~]$ git clone https://github.com/AlekhyaPenubakula/InstantLibrary.git
fatal: destination path 'InstantLibrary' already exists and is not an empty directory.
[ec2-user@ip-172-31-3-5 ~]$ cd InstantLibrary
[ec2-user@ip-172-31-3-5 InstantLibrary]$ cd InstantLibrary
[ec2-user@ip-172-31-3-5 InstantLibrary]$ flask run --host=0.0.0.0 --port=80
 * Debug mode: off
Permission denied
[ec2-user@ip-172-31-3-5 InstantLibrary]$ ^C
[ec2-user@ip-172-31-3-5 InstantLibrary]$ ^C
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://172.31.3.5:80
Press CTRL+C to quit
^C[ec2-user@ip-172-31-3-5 InstantLibrary]$
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://172.31.3.5:80
Press CTRL+C to quit
183.82.125.56 - - [22/Oct/2024 07:42:00] "GET / HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /register HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /static/images/library3.jpg HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /favicon.ico HTTP/1.1" 404 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /static/images/library3.jpg HTTP/1.1" 304 -
183.82.125.56 - - [22/Oct/2024 07:42:21] "POST /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:24] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:27] "POST /login HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:28] "GET /home-page HTTP/1.1" 200 -

```

i-001861022fbcac290 (InstantLibraryApp)
 PublicIPs: 13.201.74.42 PrivateIPs: 172.31.3.5

Verify the Flask

app is

running:

<http://your>

[-ec2-public-](http://ec2-public-)

ip

- Run the Flask app on the EC2 instance

```

[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://172.31.3.5:80
Press CTRL+C to quit
183.82.125.56 - - [22/Oct/2024 07:42:00] "GET / HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /register HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /static/images/library3.jpg HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /favicon.ico HTTP/1.1" 404 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /static/images/library3.jpg HTTP/1.1" 304 -
183.82.125.56 - - [22/Oct/2024 07:42:21] "POST /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:24] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:27] "POST /login HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:28] "GET /home-page HTTP/1.1" 200 -

```

Milestone 8: Testing and Deployment

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

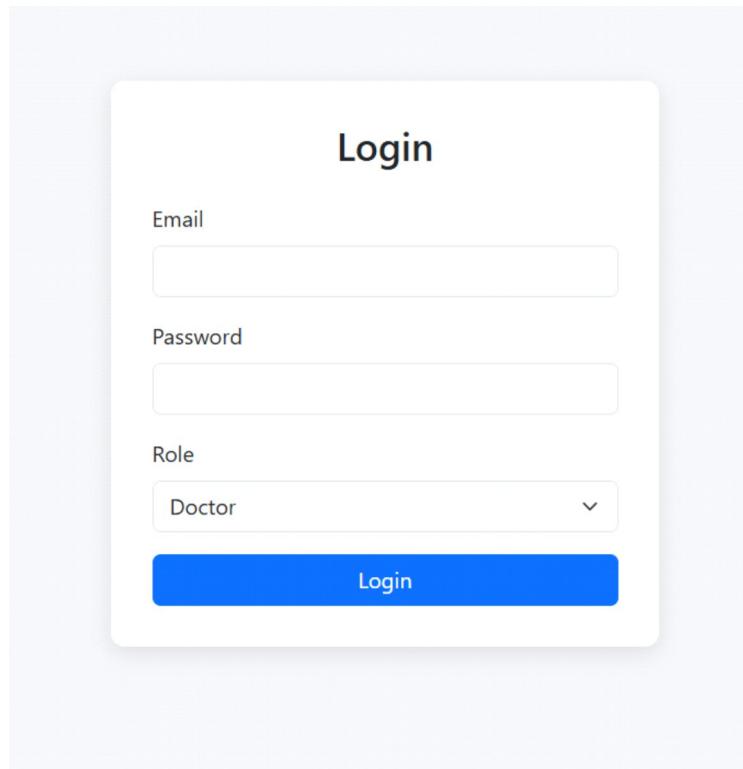
Register Page:

The form is titled "Register". It contains the following fields:

- Name: An input field.
- Email: An input field.
- Password: An input field.
- Role: A dropdown menu set to "Patient".
- Age: An input field.
- Gender: A dropdown menu set to "Male".

A large blue button at the bottom is labeled "Register".

Login Page:



Home page:

The image shows the homepage of MedTrack. At the top is a blue header bar with the 'MedTrack' logo. The main body is teal-colored. In the center, it says 'Welcome to MedTrack' and 'Your cloud-based solution for medical record management and diagnosis.' Below this are three buttons: 'Register' (white), 'Login' (white), and 'About Us' (dark grey). At the bottom, there are three sections: 'Secure Data' (with text about AWS DynamoDB), 'AI Diagnosis' (with text about AI tools), and 'Cloud Hosted' (with text about AWS EC2 hosting).

Secure Data

Store and access patient records safely using AWS DynamoDB.

AI Diagnosis

Leverage AI tools for accurate, fast medical diagnosis.

Cloud Hosted

Access MedTrack anytime, anywhere with AWS EC2 hosting.

About Us

MedTrack is a cloud-powered healthcare platform designed to streamline communication between doctors and patients. Our system allows secure storage of health records, AI-assisted diagnosis, appointment scheduling, and real-time updates using AWS services. We aim to modernize healthcare delivery with

About Us page:

Welcome to MedTrack

Your cloud-based solution for medical record management and diagnosis.

[Register](#)[Login](#)[About Us](#)

Secure Data

Store and access patient records safely using AWS DynamoDB.

AI Diagnosis

Leverage AI tools for accurate, fast medical diagnosis.

Cloud Hosted

Access MedTrack anytime, anywhere with AWS EC2 hosting.

About Us

MedTrack is a cloud-powered healthcare platform designed to streamline communication between doctors and patients. Our system allows secure storage of health records, AI-assisted diagnosis, appointment scheduling, and real-time updates using AWS services. We aim to modernize healthcare delivery with scalable, accessible technology.

Doctor DashBoard:

MedTrack

Logout

Welcome, Dr.

This is your doctor dashboard. You can access patient data and review diagnoses.

Patient List

Access and manage your assigned patients and their medical history.

[View Patients](#)

Diagnosis Reviews

Review recent AI diagnoses and provide expert feedback.

[Review Diagnoses](#)

© 2025 MedTrack. All rights reserved.

Patient DashBoard:

Welcome,

This is your patient dashboard.

View Medical Records

Access your stored medical history securely.

[View Records](#)**Run Diagnosis**

Use AI tools to run a new diagnosis.

[Start Diagnosis](#)

© 2025 MedTrack. All rights reserved.

Exit:

Session Ended

Please close this tab.

Conclusion:

MedTrack is a web-based healthcare management system built using Flask that facilitates interaction between doctors and patients. It allows users to register and log in as either a doctor or a patient. Patients can view available doctors, book appointments, and view their scheduled or past visits. Doctors can access their assigned appointments, update them with a diagnosis, treatment plan, and prescription, and mark them as completed. The system uses in-memory storage for users and appointments, making it suitable for demonstration or development purposes without requiring a database. Although email and AWS SNS notification features are present in the code, they are disabled by default. MedTrack is designed with a clear role-based workflow and is easily extendable for real-world deployment with additional features like persistent databases, email alerts, password encryption, and a modern UI.