

Greenfield University Instant Library: Quick Access to Learning Resources

Project Description:

At Greenfield University, the BSC Computer Science department faces a shortage of physical textbooks due to a growing student population. The limited availability of library resources has led to long wait times and challenges in accessing essential study materials.

To solve this, the university's Cloud Solutions Department developed the Instant Library—a virtual library platform. Using Flask for backend development, AWS EC2 for hosting, and DynamoDB for managing data, the system allows students to register, log in, and request books online. AWS SNS sends real-time notifications to students and library staff about requests, ensuring effective communication and resource management. This cloud-based solution enhances the availability of study materials, providing seamless access for all students.

Scenario 1: Efficient Book Request System for Students

In the Instant Library System at Greenfield University, AWS EC2 ensures a reliable infrastructure to manage multiple students accessing the platform simultaneously. For example, a student can log in, navigate to the book request page, and easily submit a request for unavailable textbooks. 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 book requests during peak periods, ensuring smooth operation without delays.

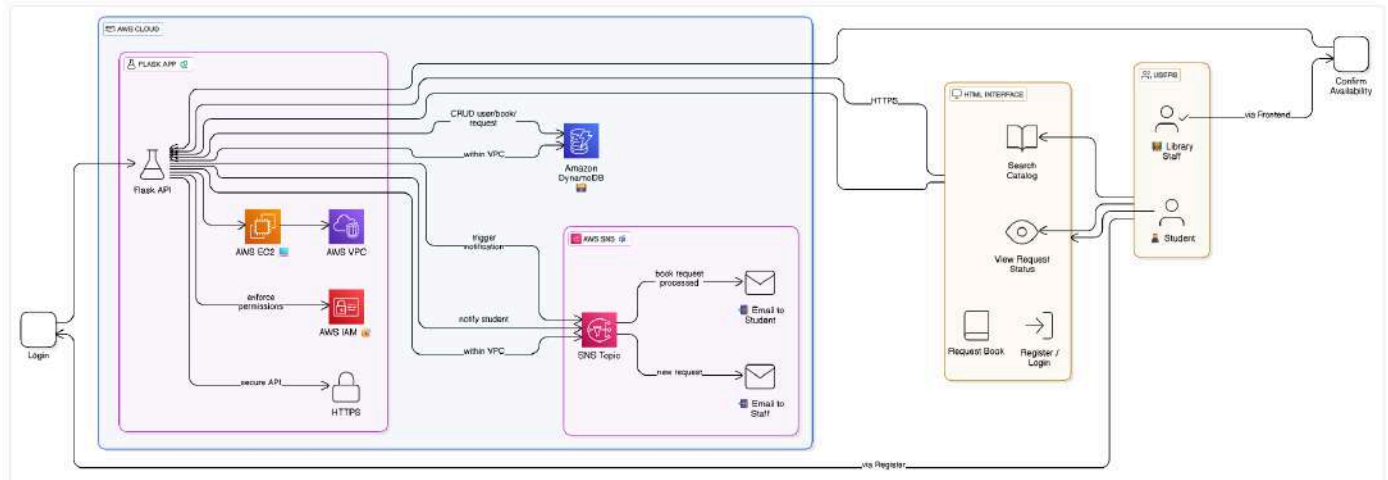
Scenario 2: Seamless Book Request Notifications for Library Staff

When students request books that are unavailable in the physical library, the Instant Library System uses AWS SNS to notify both the students and the library staff. For instance, a student requests a textbook, and Flask processes the request while SNS sends an email to both the student (confirming the request) and the library staff (informing them of the new request). The secure integration with AWS DynamoDB stores all book requests, ensuring they are tracked and resolved efficiently.

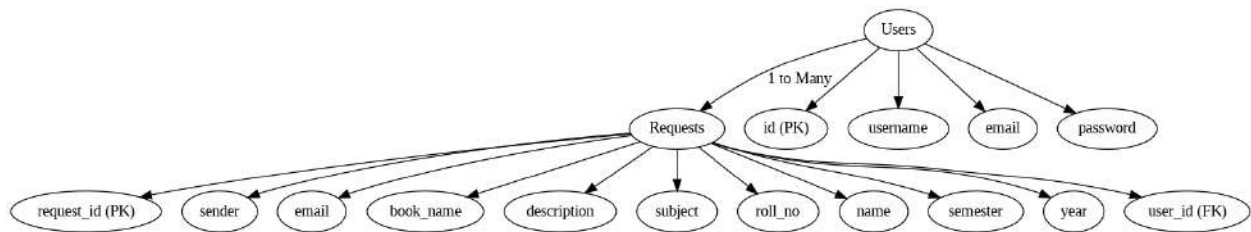
Scenario 3: Easy Access to Library Resources

The Instant Library System provides students with easy access to available resources in the library. For example, a student logs in and views the list of available books and materials on the platform. They can quickly check the availability status or place a request if the book is not in stock. Flask manages real-time data fetching from DynamoDB, while EC2 hosting ensures the platform performs seamlessly even when multiple students access it at the same time, offering a smooth and uninterrupted user experience.

AWS ARCHITECTURE



Entity Relationship (ER)Diagram:



Pre-requisites:

1. **.AWS Account Setup:** [AWS Account Setup](#)
2. **Understanding IAM:** [IAM Overview](#)
3. **Amazon EC2 Basics:** [EC2 Tutorial](#)
4. **DynamoDB Basics:** [DynamoDB Introduction](#)
5. **SNS Overview:** [SNS Documentation](#)
6. **Git Version Control:** [Git Documentation](#)

Project WorkFlow:

1. AWS Account Setup and Login

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

Activity 1.2: Log in to the AWS Management Console

2. DynamoDB Database Creation and Setup

Activity 2.1: Create a DynamoDB Table.

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

3. SNS Notification Setup

Activity 3.1: Create SNS topics for book request notifications.

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

4. Backend Development and Application Setup

Activity 4.1: Develop the Backend Using Flask.

Activity 4.2: Integrate AWS Services Using boto3.

5. IAM Role Setup

Activity 5.1: Create IAM Role

Activity 5.2: Attach Policies

6. EC2 Instance Setup

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

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

7. Deployment on EC2

Activity 7.1: Upload Flask Files

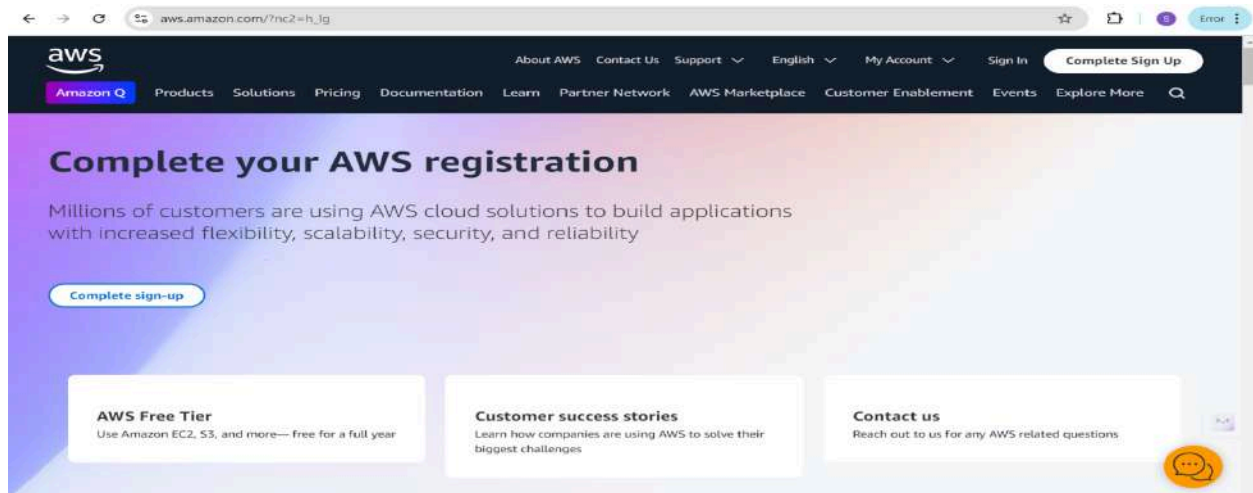
Activity 7.2: Run the Flask App

8. Testing and Deployment

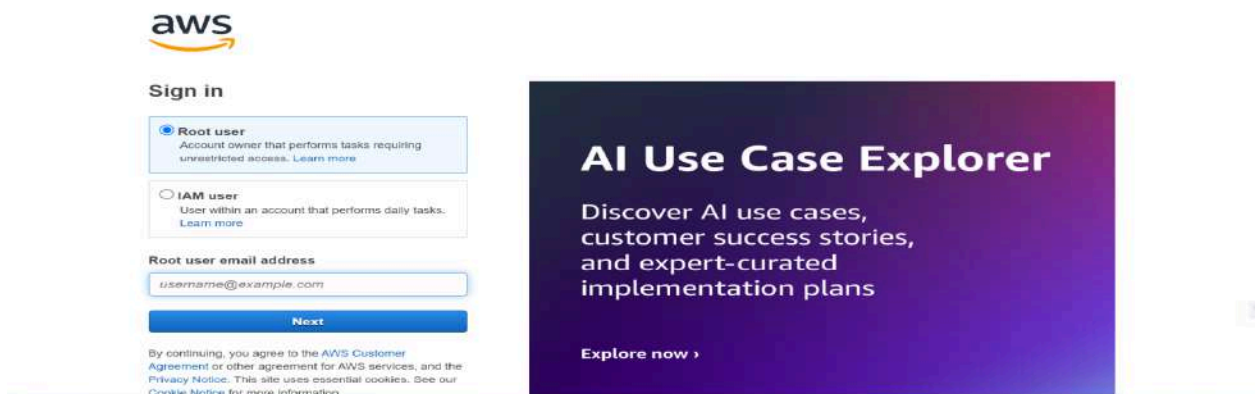
Activity 8.1: Conduct functional testing to verify user registration, login, book requests, and notifications.

Milestone 1: AWS Account Setup and Login

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

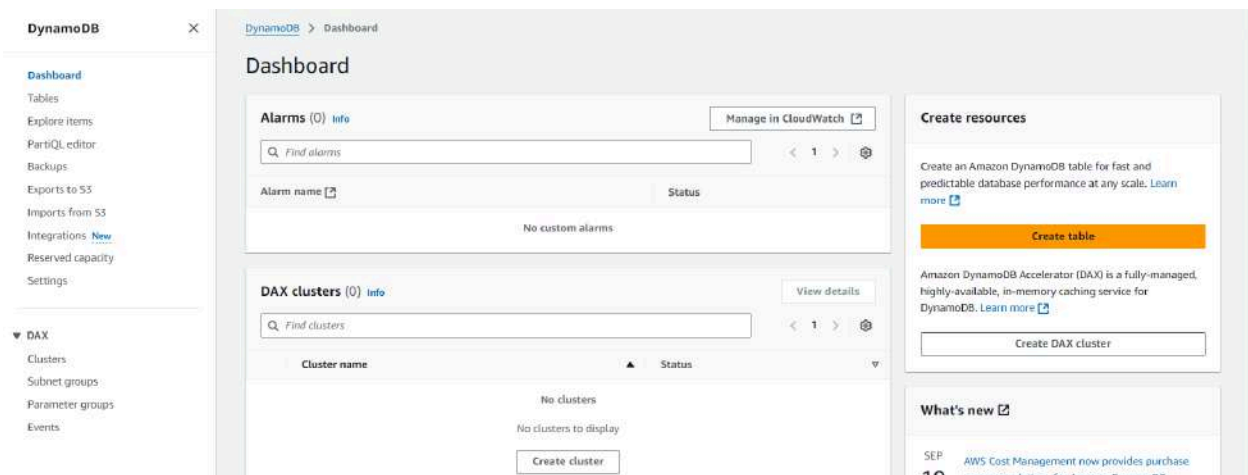
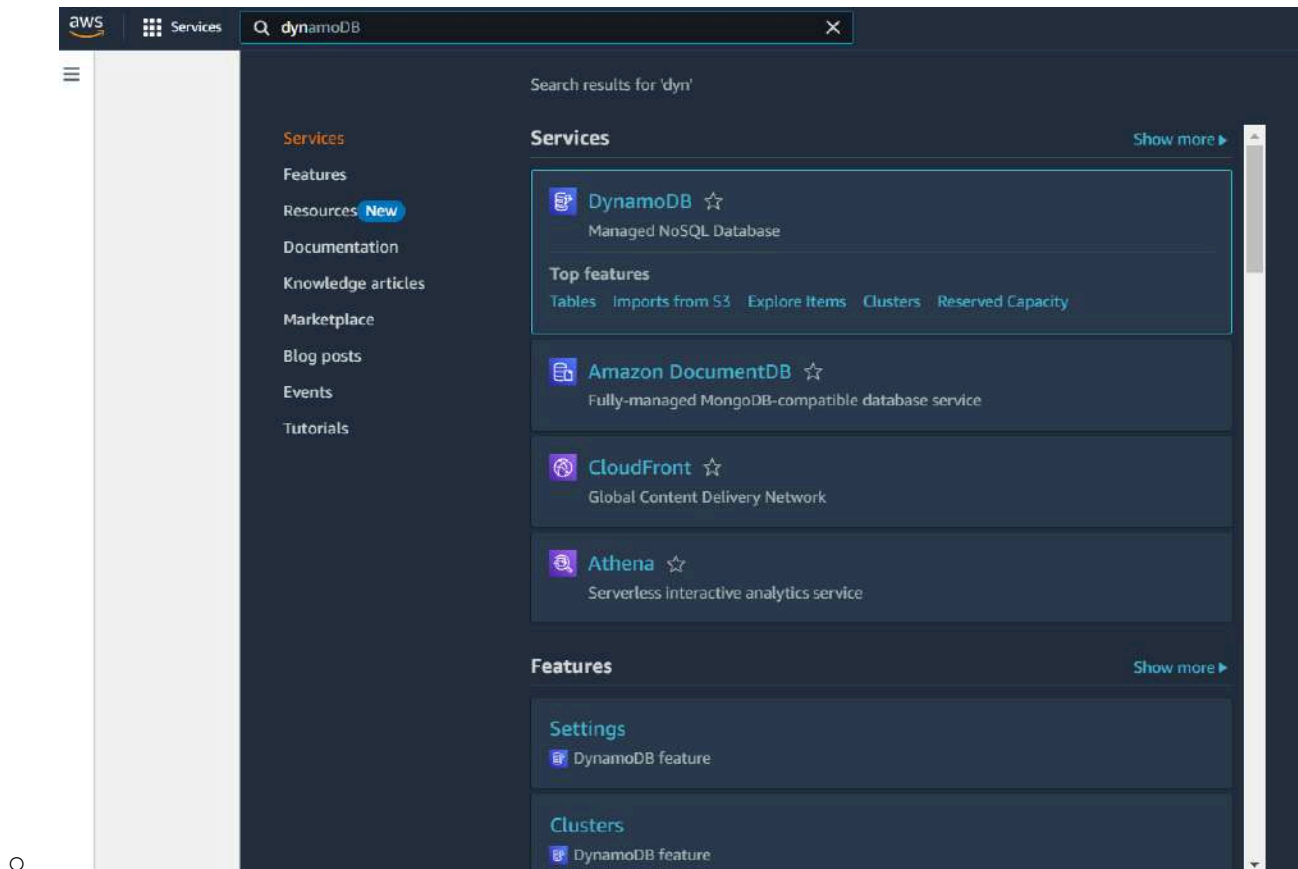


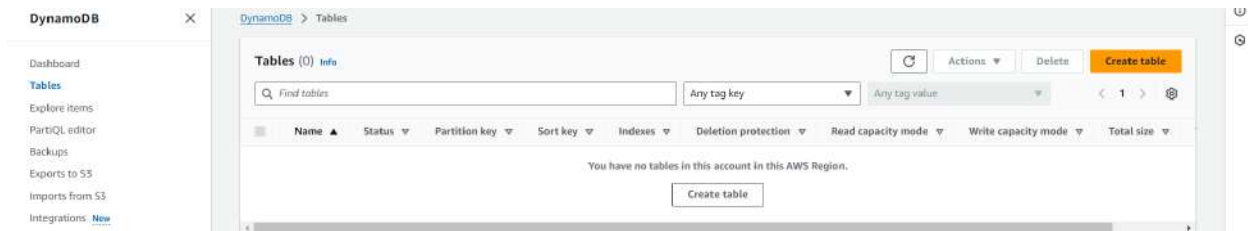
- **Activity 1.2: Log in to the AWS Management Console**
 - After setting up your account, log in to the [AWS Management Console](#).



Milestone 2: DynamoDB Database Creation and Setup

- **Activity 2.1: Navigate to the DynamoDB**
 - In the AWS Console, navigate to DynamoDB and click on create tables.





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

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

Create table

Table details
[Info](#)

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

Table name

This will be used to identify your table.

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

Partition key

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

String ▼

1 to 255 characters and case sensitive.

Sort key - optional

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

String ▼

1 to 255 characters and case sensitive.

☰

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

Tags

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

No tags are associated with the resource.

Add new tag

You can add 50 more tags.

Cancel
Create table

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

The Users table was created successfully.

DynamoDB > Tables
Refresh
Actions ▼
Delete
Create table

Any tag key ▼

Any tag value ▼

< 1 >

⚙

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

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

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

Create table

Table details [Info](#)

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

Table name

This will be used to identify your table.

Between 3 and 255 characters, 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 settings

☒ Default settings

The fastest way to configure a table. You can modify

☐ Customize settings

Use the 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

DynamoDB

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

The Requests table was created successfully.

DynamoDB > Tables

Tables (2) Info

Find tables

Any tag key

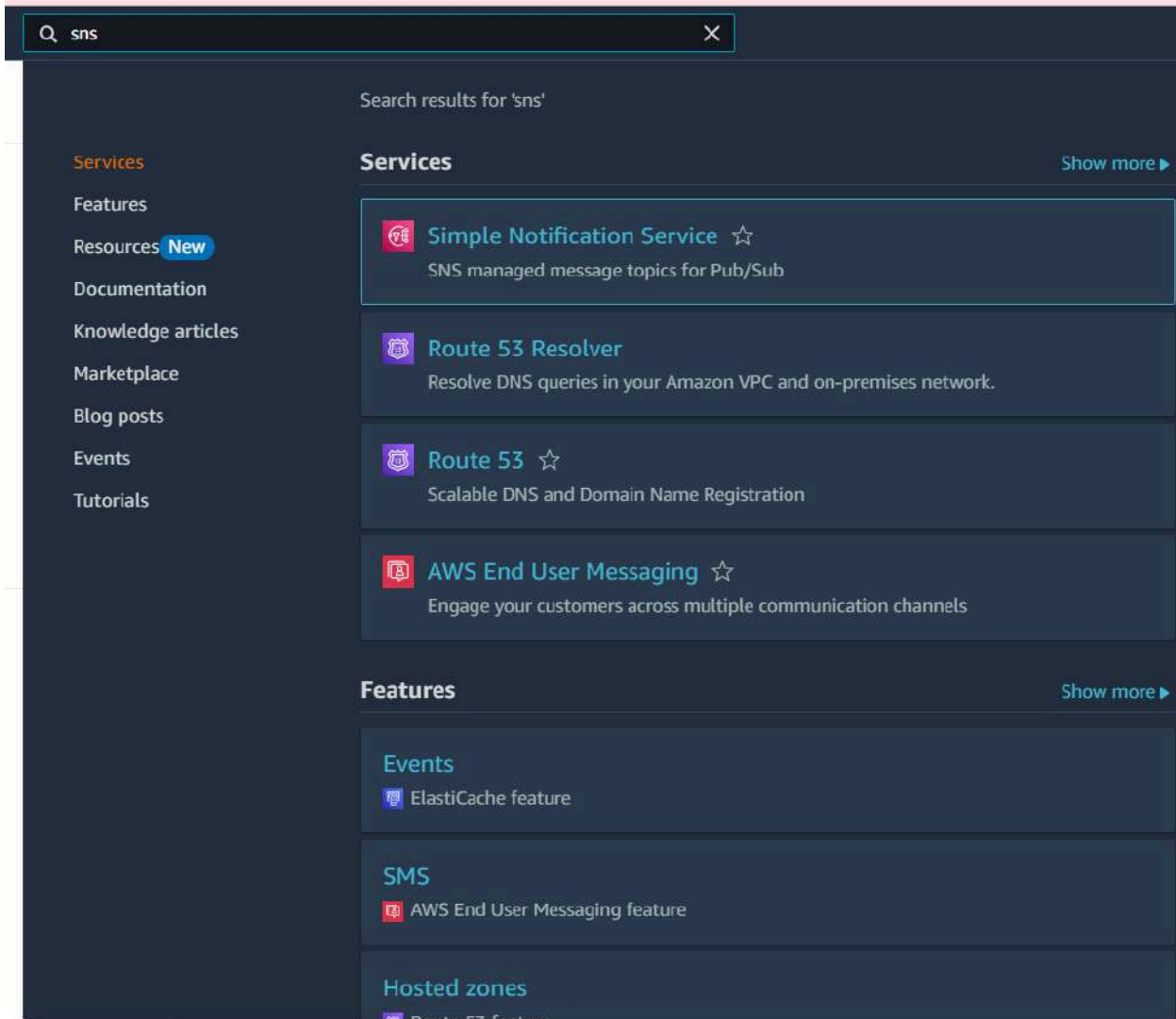
Any tag value

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

Milestone 3: SNS Notification Setup

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





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






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

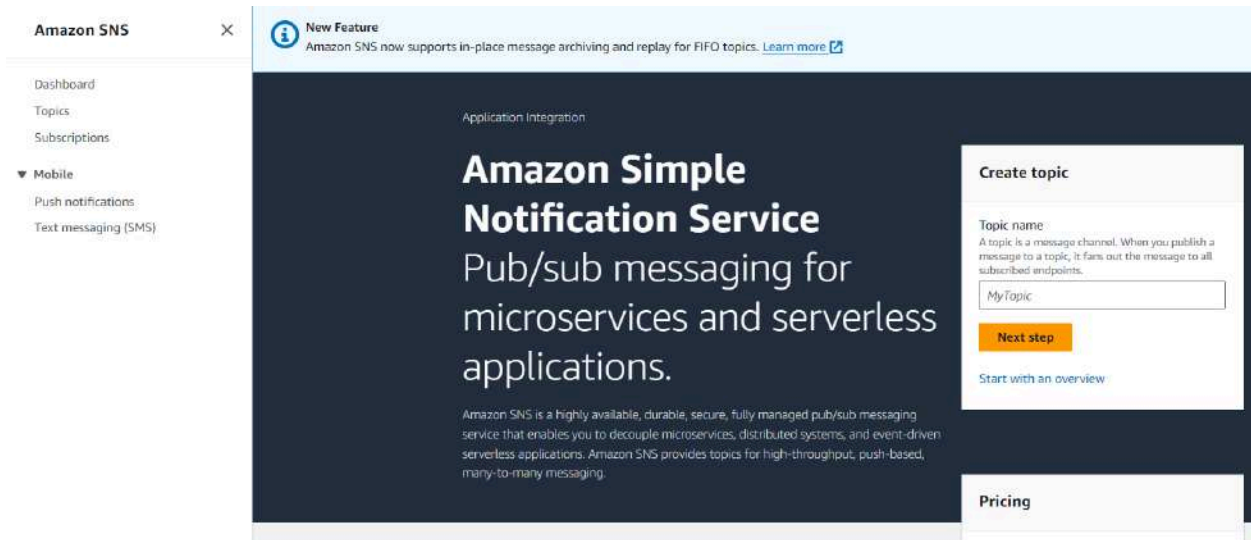
Search results for 'sns'

Services [Show more ►](#)

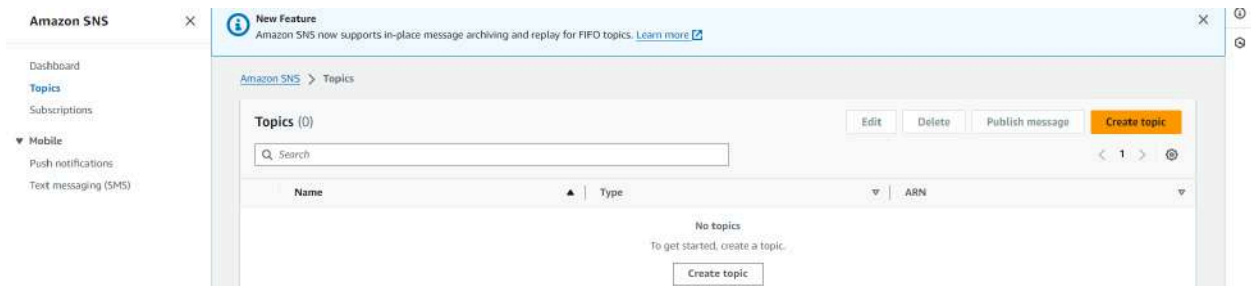
-  **Simple Notification Service** ☆
SNS managed message topics for Pub/Sub
-  **Route 53 Resolver**
Resolve DNS queries in your Amazon VPC and on-premises network.
-  **Route 53** ☆
Scalable DNS and Domain Name Registration
-  **AWS End User Messaging** ☆
Engage your customers across multiple communication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

Features [Show more ►](#)

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



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



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

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

Create topic

Details

Type | [Info](#)

Topic type cannot be modified after topic is created

☐ FIFO (first-in, first-out)

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

☒ Standard

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

Name

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

Display name - optional | [Info](#)

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

Maximum 100 characters.

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

Amazon SNS

New Feature

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

Dashboard

Topics

Subscriptions

Push notifications

Text messaging (SMS)

1

New Feature

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

Topic: BookRequestNotifications created successfully.

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

Publish message

Amazon SNS > Topics > BookRequestNotifications

BookRequestNotifications

Edit Delete Publish message

Details

Name

BookRequestNotifications

Display name

-

ARN

arn:aws:sns:us-east-1:557690616836:BookRequestNotifications

Topic owner

557690616836

Type

Standard

Subscription

Access policy

Data protection policy

Delivery policy (HTTP/S)

Delivery status logging

Encryption

Tags

Integrations

Subscriptions (0)

Q Search

Edit

Delete

Request confirmation

Confirm subscription

Create subscription

ID

Endpoint

Status

Protocol

No subscriptions found.

You don't have any subscriptions to this topic.

Create subscriptions

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

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN

Protocol

The type of endpoint to subscribe

Endpoint

An email address that can receive notifications from Amazon SNS.

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

► **Subscription filter policy - optional** [Info](#)

This policy filters the messages that a subscriber receives.

► **Redrive policy (dead-letter queue) - optional** [Info](#)

Send undeliverable messages to a dead-letter queue.

Cancel [Create subscription](#)

Amazon SNS

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

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

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

Details

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

Endpoint: `instantlibrary2@gmail.com`

Topic: [BookRequestNotifications](#)

Subscription Protocol: `arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications`

Status: Ⓢ Pending confirmation

Protocol: EMAIL

Subscription filter policy ⓘ **Redrive policy (dead-letter queue)**

Subscription filter policy ⓘ
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

Amazon SNS

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

Confirmation request was sent successfully.
The ARN of the subscription is `arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:7aee516-13ad-47f1-9f7d-c342c2713a05`.

BookRequestNotifications Edit Delete Publish message

Details

Name: `BookRequestNotifications`

Display name: `-`

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

Topic owner: `557690616836`

Type: Standard

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

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

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

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

AWS Notification - Subscription Confirmation Inbox x

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

9

You have chosen to subscribe to the topic:

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

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

[Confirm subscription](#)

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

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

...

You have chosen to subscribe to the topic:

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

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

[Confirm subscription](#)

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



Simple Notification Service

Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:

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

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

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

Amazon SNS > Topics > BookRequestNotifications

BookRequestNotifications

Edit Delete Publish message

Details

Name: BookRequestNotifications

ARN: arn:aws:sns:us-east-1:557696156836:BookRequestNotifications

Type: Standard

Display name: -

Topic owner: 557696156836

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

Subscriptions (2)

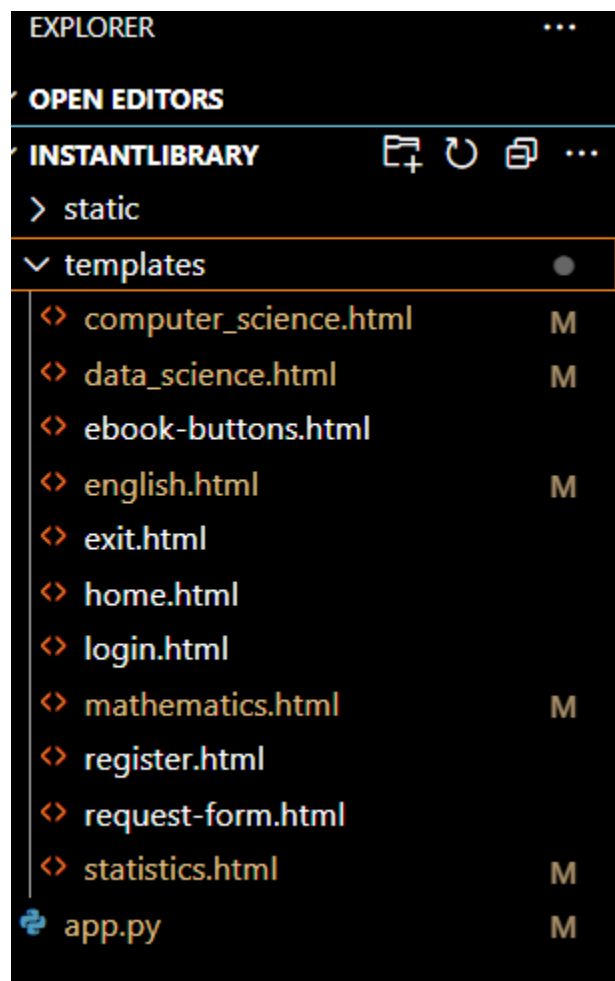
Search

ID	Endpoint	Status	Protocol
af78a5171-9235-4046-962c-89c3741807cd	instantlibrary2@gmail.com	Confirmed	EMAIL

Edit Delete Request confirmation Confirm subscription Create subscription

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 'Requests'
```

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, 'plain')
    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}\n\nsend_email('instantlibrary2@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.

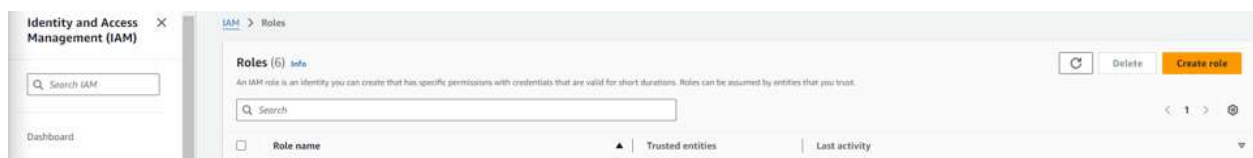
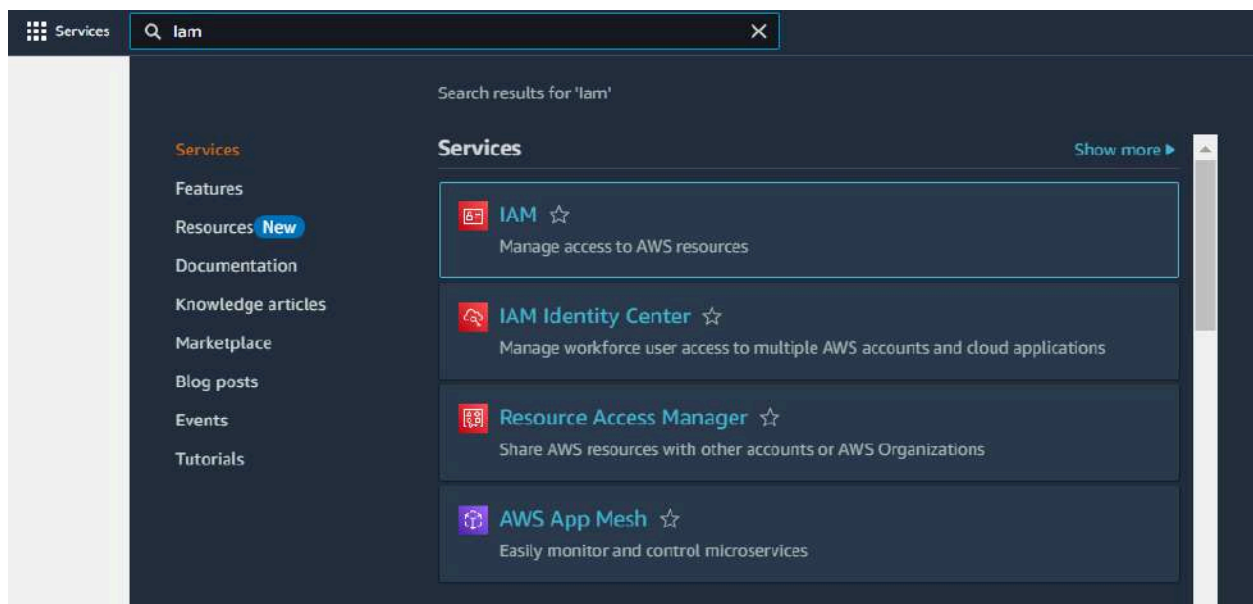
Deployment Code:

```
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.



Step 1: Create role

Step 1: Select trusted entity

Step 2: Add permissions

Step 3: Name, review, and create

Select trusted entity

Trusted entity type

- ☒ **AWS service**
Allows an AWS service (or EC2 instance, as shown in policies in this account) to perform actions on this account.
- ☐ **AWS account**
Allows an external AWS account (belonging to someone else) to perform actions on this account.
- ☐ **Web identity**
Allows your application to use a web identity provider (such as Amazon Cognito) to assume this role to perform actions on this account.
- ☐ **External ID federation**
Allows your application to use an external ID to perform actions on this account.
- ☐ **Custom trust policy**
Creates a custom trust policy to control who can perform actions on this account.

Use case
When an AWS service (or EC2 instance, as shown in policies in this account) performs actions on this account.

Service or task role

EC2

Choose a task role for the specified service.

Task roles

- ☒ **EC2**
Allows EC2 instances to call AWS services on your behalf.
- ☐ **EC2 Role for AWS Systems Manager**
Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.
- ☐ **EC2 Spot Fleet Role**
Allows EC2 Spot Fleet to request and manage Spot instances on your behalf.
- ☐ **EC2 - Spot Fleet Auto Scaling**
Allows Auto Scaling to create and update EC2 Spot Fleets on your behalf.
- ☐ **EC2 - Spot Fleet Tagging**
Allows EC2 to perform read, update, and delete tags for this specified instance on your behalf.
- ☐ **EC2 - Spot instance**
Allows EC2 to perform read, update, and delete tags for this specified instance on your behalf.
- ☐ **EC2 - Spot Fleet**
Allows EC2 Spot Fleet to request and manage Spot instances on your behalf.
- ☐ **EC2 - Scheduled instances**
Allows EC2 Scheduled instances to manage compute on your behalf.

Cancel Next

Step 1: Create role

Step 1: Select trusted entity

Step 2: Add permissions

Step 3: Name, review, and create

Add permissions

Permissions policies (1/355)

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

Filter by Type: All types 2 matches

Policy name	Type
<input checked="" type="checkbox"/> AmazonDynamoDBFullAccess	AWS managed
<input type="checkbox"/> AmazonDynamoDBReadOnlyAccess	AWS managed

Set permissions boundary - optional

Cancel Previous Next

● Activity 5.2: Attach Policies.

Attach the following policies to the role:

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

IAM > Roles > sns_Dynamodb_role

sns_Dynamodb_role [info](#) Delete

Allows EC2 instances to call AWS services on your behalf.

Summary Edit

Creation date

October 13, 2024, 23:09 (UTC+05:30)

ARN

arn:aws:iam::557690616836:role/sns_Dynamodb_role

Instance profile ARN

arn:aws:iam::557690616836:instance-profile/sns_Dynamodb_role

Last activity

6 days ago

Maximum session duration

1 hour

Permissions Trust relationships Tags Last Accessed Revoke sessions

Permissions policies (2) [info](#) Refresh Simulate Remove Add permissions

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AmazonDynamoDBFullAccess	AWS managed	1
AmazonSNSFullAccess	AWS managed	2

Milestone 6: EC2 Instance Setup


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

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


Local Codespaces

Clone ?

HTTPS SSH GitHub CLI

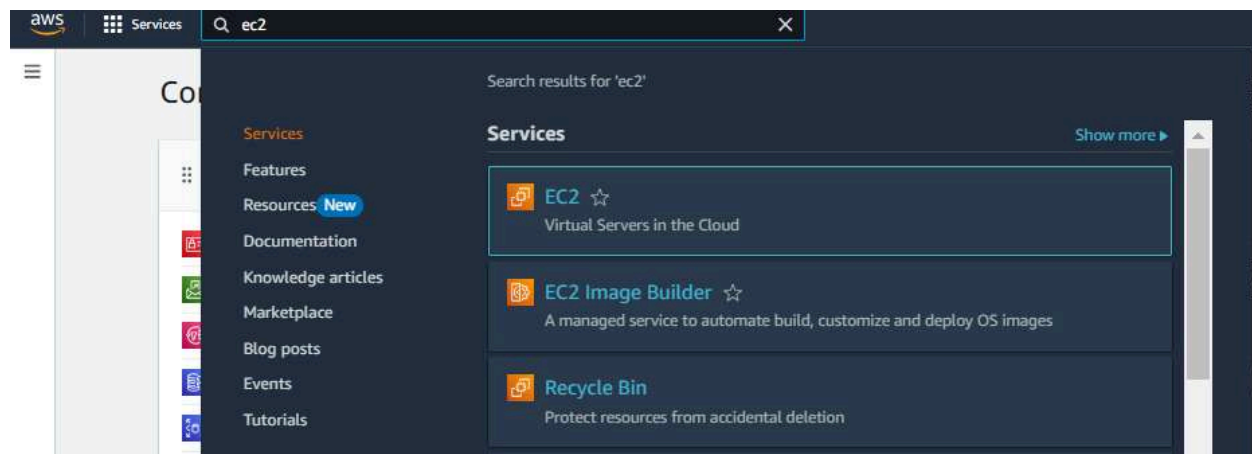
`https://github.com/AlekhyaPenubakula/InstantLil` 

Clone using the web URL.

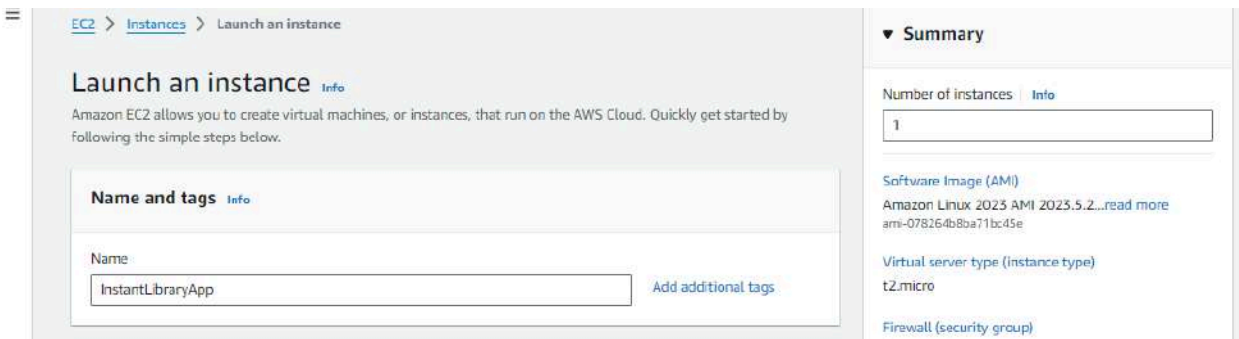
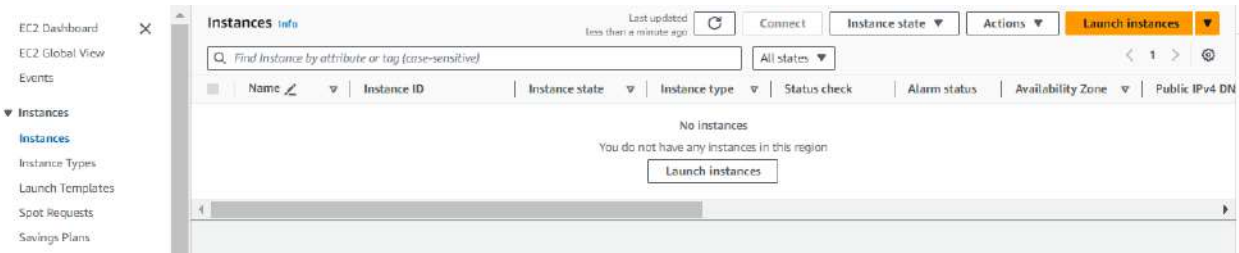
 Open with GitHub Desktop

 Download ZIP

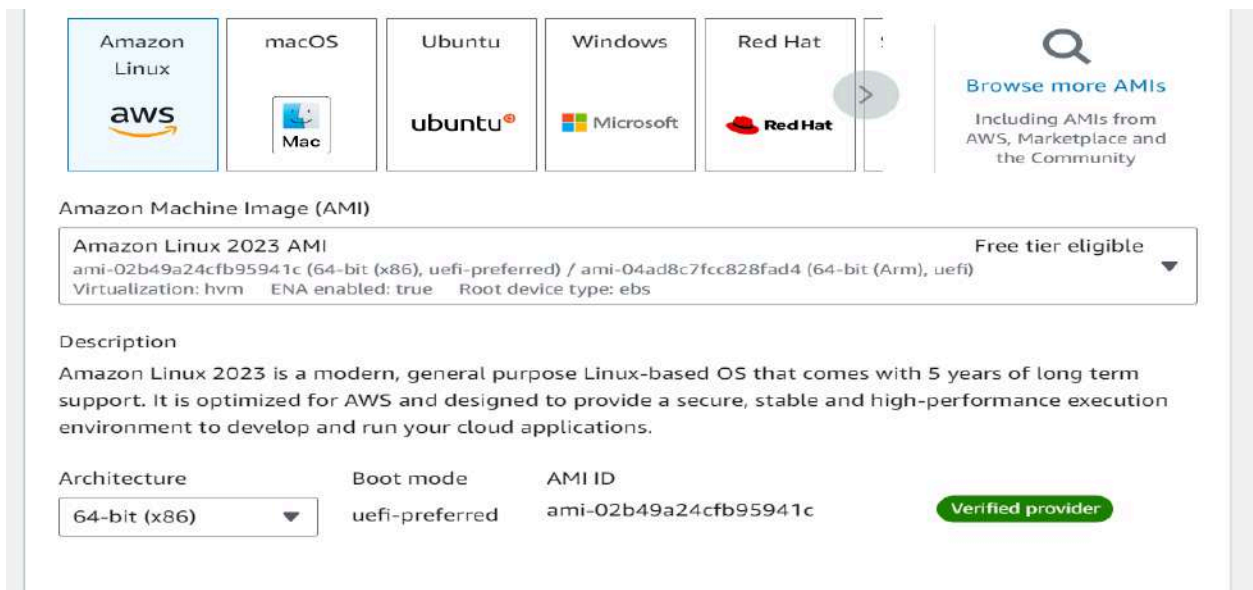
- **Activity 6.1: Launch an EC2 instance to host the Flask application.**
 - **Launch EC2 Instance**
 - In the AWS Console, navigate to EC2 and launch a new instance.



- Click on Launch instance to launch EC2 instance



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



- Create and download the key pair for Server access.

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

Instance type

t2.micro Free tier eligible

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

☐ All generations
[Compare instance types](#)


Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) [Info](#)

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

Key pair name - *required*

Select ▼

 [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

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

64-bit (x86)

Boot mode

uefi-preferred

AMI ID

ami-078264b8ba71bc45e

Username

ec2-user

Verified provider

▼ 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

InstantLibrary

Create new key pair

▼ 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)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

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.

Can cel

Preview code

Launch instance

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

Network settings
[Info](#)

VPC - required [Info](#)

vpc-03cdc7b6f19dd7211
172.31.0.0/16

(default) ↕

Subnet [Info](#)

No preference ↕

↻ Create new subnet [?](#)

Auto-assign public IP [Info](#)

Enable ↕

Additional charges apply when outside of free tier allowance

Firewall (security groups) [Info](#)

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

☒ Create security group
 ☐ Select existing security group

Security group name - required

launch-wizard

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

Description - required [Info](#)

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

Inbound Security Group Rules

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

Remove

Type [Info](#)

ssh ↕

Source type [Info](#)

Anywhere ↕

Protocol [Info](#)

TCP

Source [Info](#)

0.0.0.0/0 ✕

Port range [Info](#)

22

Description - optional [Info](#)

e.g. SSH for admin desktop

Security group rule 2 (TCP, 80, 0.0.0.0/0)

Remove

Type [Info](#)

HTTP ↕

Source type [Info](#)

Custom ↕

Protocol [Info](#)

TCP

Source [Info](#)

0.0.0.0/0 ✕

Port range [Info](#)

80

Description - optional [Info](#)

e.g. SSH for admin desktop

Security group rule 3 (TCP, 5000, 0.0.0.0/0)

Remove

Type [Info](#)

Custom TCP ↕

Source type [Info](#)

Custom ↕

Protocol [Info](#)

TCP

Source [Info](#)

0.0.0.0/0 ✕

Port range [Info](#)

5000

Description - optional [Info](#)

e.g. SSH for admin desktop

Add security group rule

EC2 > Launch an instance

Success
Successfully initiated launch of instance i-001861022fbcac290

Launch log

Next Steps

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

Create billing and free tier usage alerts

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

Create billing alerts

Connect to your instance

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

Connect to instance

Learn more

Connect an RDS database

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

Connect an RDS database

Create a new RDS database

Learn more

Create EBS snapshot policy

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

Create EBS snapshot policy

Manage detailed monitoring

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

Manage detailed monitoring

Create Load Balancer

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

Create Load Balancer

Create AWS budget

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

Create AWS budget

Manage CloudWatch alarms

Create or update Amazon CloudWatch alarms for the instance.

Manage CloudWatch alarms

Disaster recovery for your instances

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

Disaster recovery for your instances

Monitor for suspicious runtime activities

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

Monitor for suspicious runtime activities

Get instance screenshot

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

Get instance screenshot

Get system log

View the instance's system log to troubleshoot issues.

Get system log

[View all instances](#)

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

Instances (1/2) info

Last updated less than a minute ago

Find instance by attribute or tag (case-sensitive)

All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring	Security g
InstantLibrary...	i-001861022fbcac290	Stopped	t2.micro	-	View alarms +	ap-south-1b	-	-	-	-	disabled	launch-wi

EC2 > Instances > i-001861022fbcac290

Instance summary for i-001861022fbcac290 (InstantLibraryApp) info

Updated less than a minute ago

Connect Instance state Actions

Instance ID

i-001861022fbcac290

IPv6 address

-

Instance type

t2.micro

IP name: ip-172-31-3-5-ap-south-1-compute-internal

Answer private resource DNS name

IPv4 (A)

/Auto-assigned IP address

-

IAM Role

aws_dynamodb_role

IMDSv2

Required

Public IPv4 address

-

Instance state

Stopped

Private IP DNS name (IPv4 only)

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

Instance type

t2.micro

VPC ID

vpc-03cd7b8f13cd47211

Subnet ID

subnet-0d9fa3144480cc9a9

Instance ARN

arn:aws:ec2:ap-south-1:557680616836:instance/i-001861022fbcac290

Private IPv4 addresses

172.31.3.5

Public IPv4 DNS

-

Elastic IP addresses

-

AWS Compute Optimizer finding

Opt-in to AWS Compute Optimizer for recommendations. | Learn more

Auto Scaling Group name

-

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

EC2 > Instances > i-001861022fbcac290

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

Updated less than a minute ago

Instance ID i-001861022fbcac290	Public IPv4 address -	Private IPv4 addresses 172.31.3.5	Connect Manage instance state Instance settings Networking Security Image and templates Monitor and troubleshoot
IPv6 address -	Instance state Stopped	Public IPv4 DNS -	Change security groups Get Windows password Modify IAM role
Hostname type IP name: ip-172-31-3-5-ap-south-1-compute.internal	Private IP DNS name (IPv4 only) ip-172-31-3-5-ap-south-1-compute.internal	Elastic IP addresses -	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations Learn more
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	Auto Scaling Group name -	
Auto-assigned IP address -	VPC ID vpc-03c0c7b6f79d7211		
IAM Role sns_Dynamodb_role	Subnet ID subnet-0d5fa514448b0cd09		
IMDSv2 Required	Instance ARN arn:aws:ec2:ap-south-1:557690618836:instance/i-001861022fbcac290		

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

Modify IAM role [Info](#)

Attach an IAM role to your instance.

Instance ID
 i-001861022fbcac290 (InstantLibraryApp)

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

sns_Dynamodb_role

[Create new IAM role](#)

Cancel [Update IAM role](#)

- Now connect the EC2 with the files

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.

 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.

Cancel

Connect

```
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:

```
flask --version  
  
git --version
```

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-name.git>'

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

here: 'git clone <https://github.com/AlekhyaPenubakula/InstantLibrary.git>'

- This will 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

~\
#####
#####\
#####|
\#####
  \#/\
   V-'\-->
      /m/'

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 ~]$ 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-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 -
```

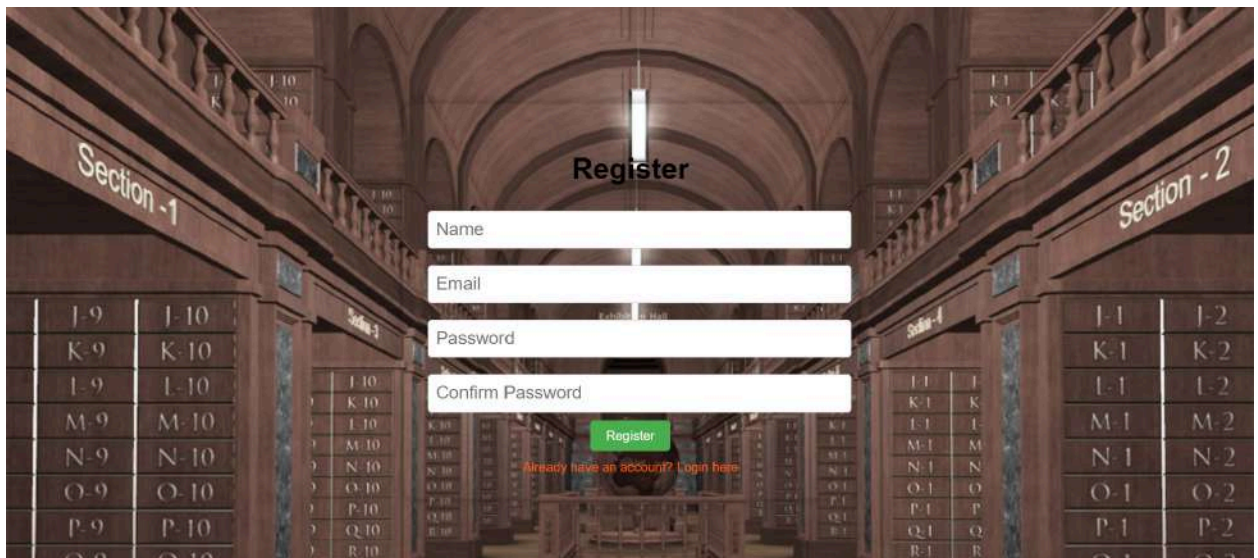

Access the website through:

PublicIPs: <https://13.201.74.42/>

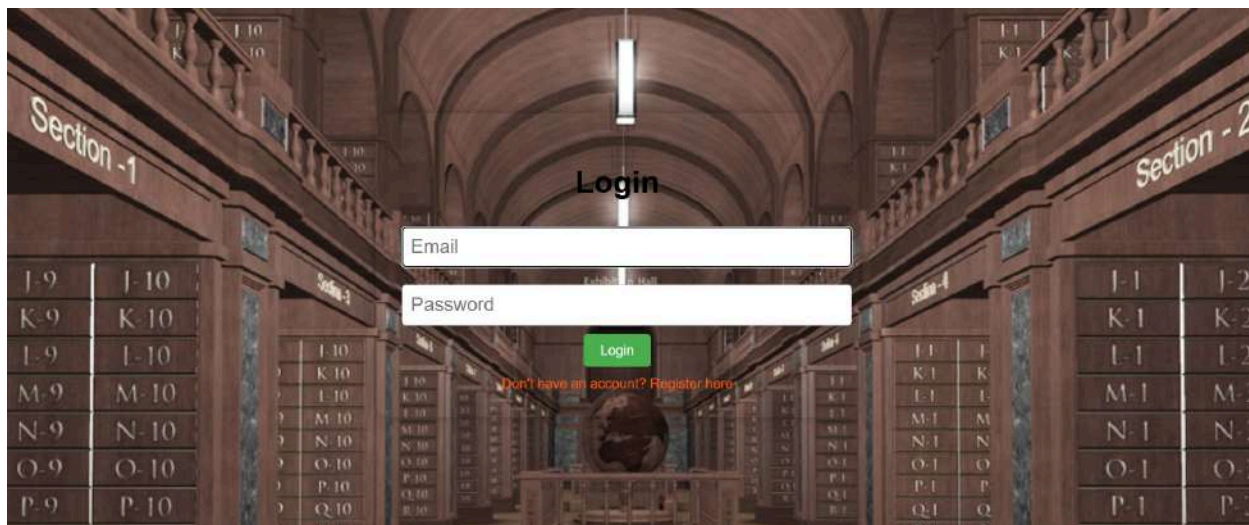
Milestone 8: Testing and Deployment

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

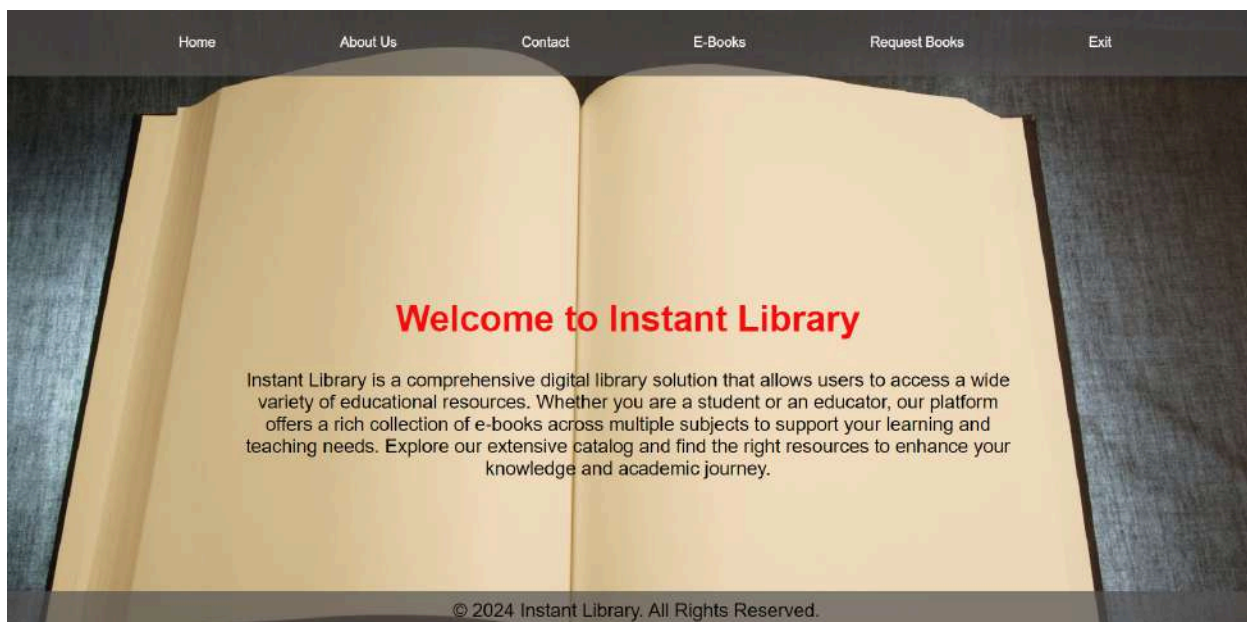
Login Page:

The image shows a registration form overlaid on a background of a library interior with wooden bookshelves. The form is titled "Register" and contains four input fields: "Name", "Email", "Password", and "Confirm Password". Below these fields is a green "Register" button. At the bottom of the form, there is a link that says "Already have an account? Login here." The background features bookshelves labeled "Section - 1" and "Section - 2" with various book spines visible.

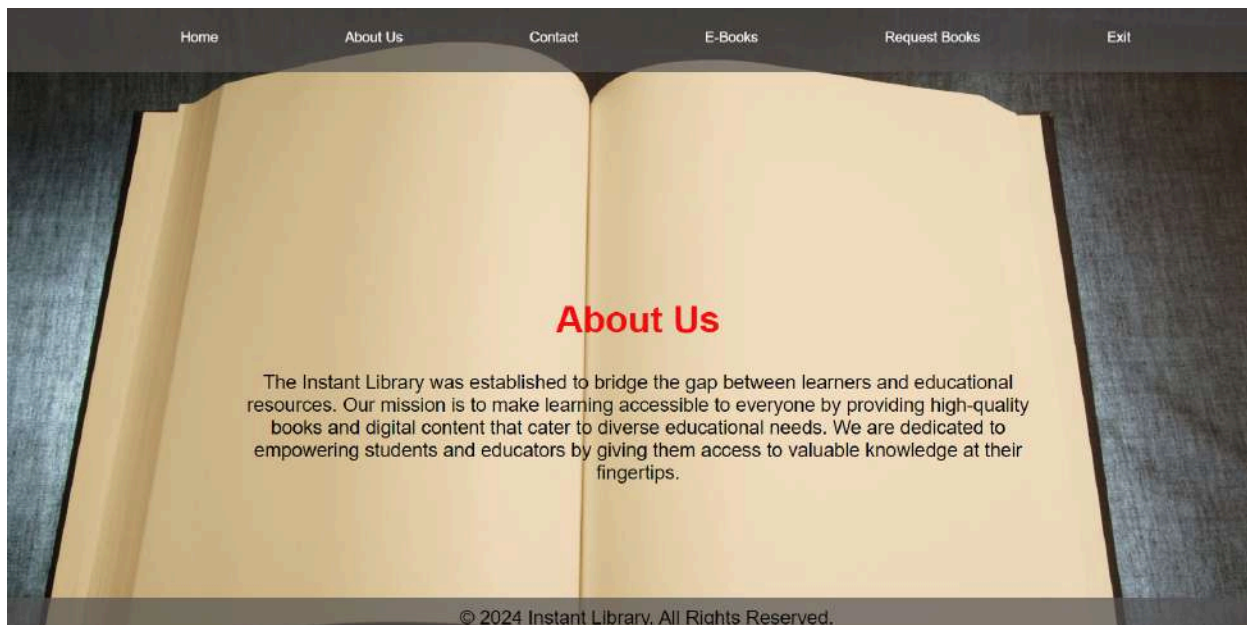
Register Page:



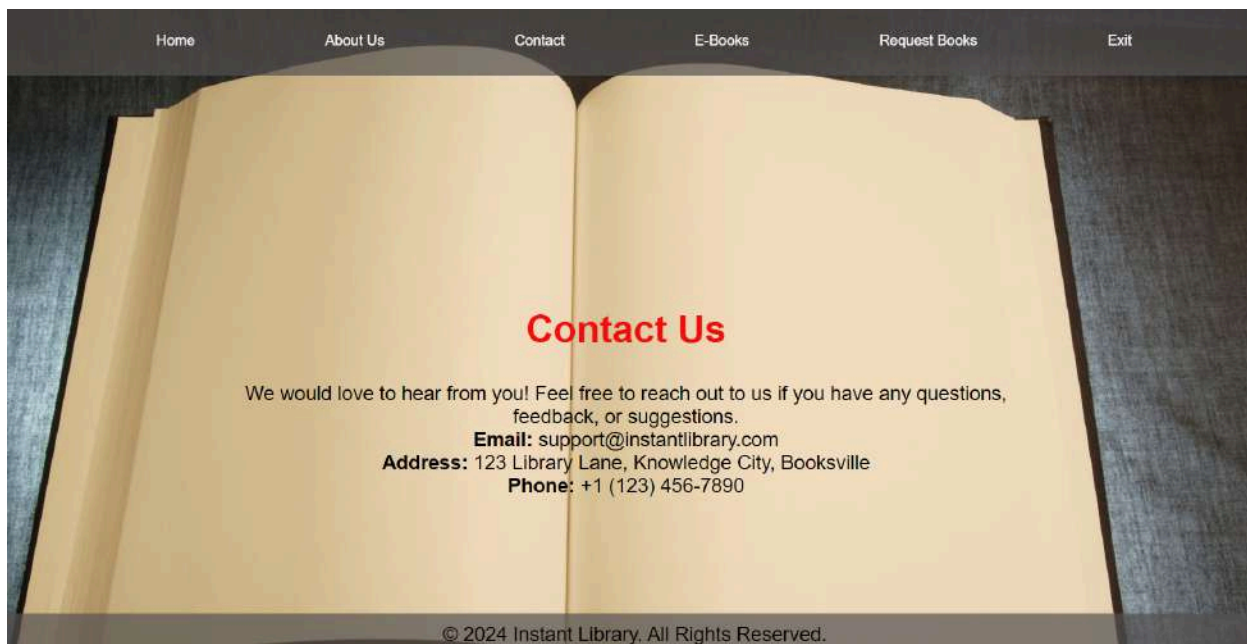
Home page:



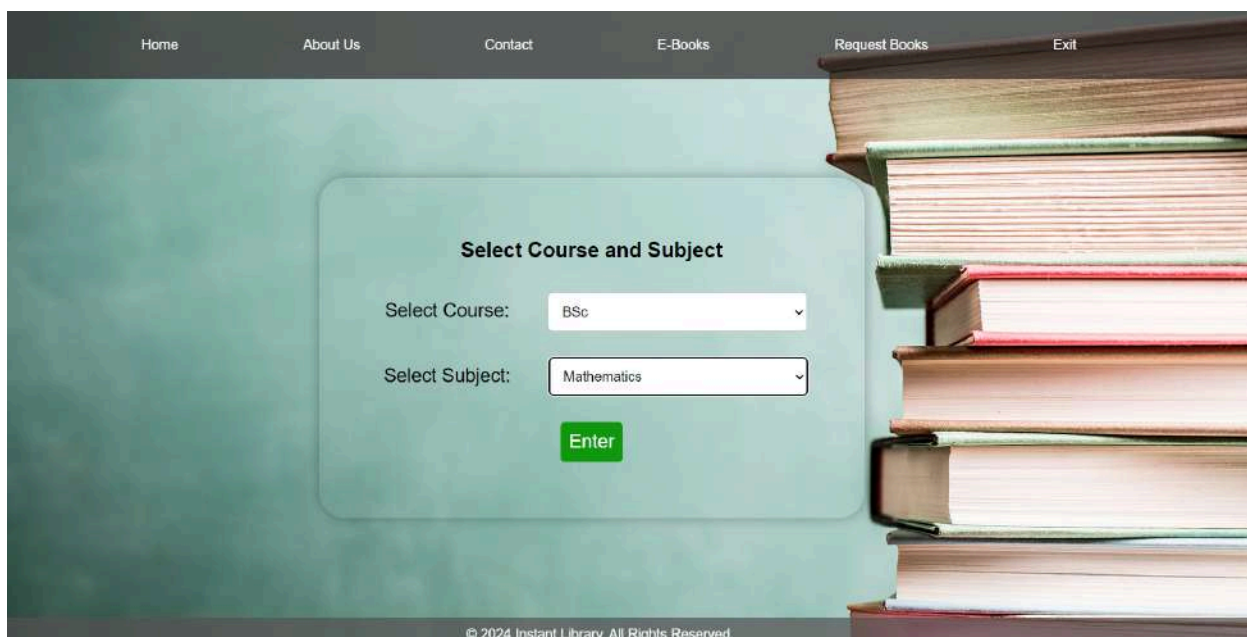
About Us page:



Contact Page:



E-Books page:



Home About Us Contact E-Books Request Books Exit

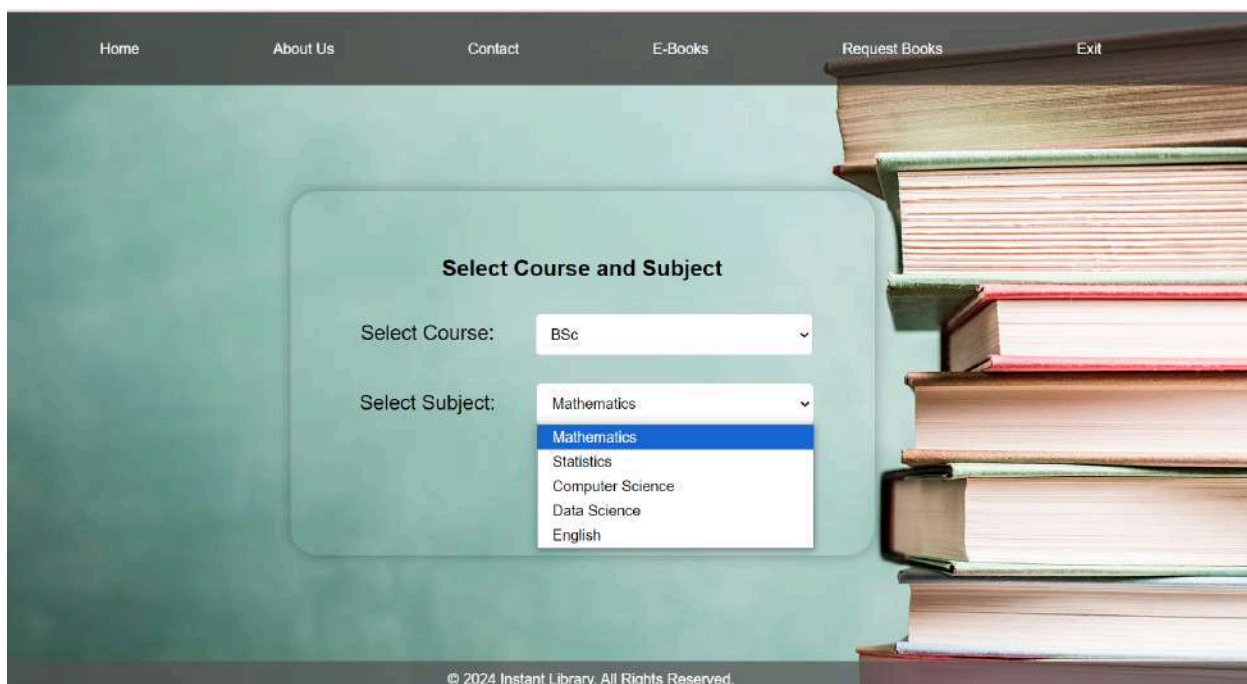
Select Course and Subject

Select Course: BSc

Select Subject: Mathematics

Enter

© 2024 Instant Library. All Rights Reserved.



Home About Us Contact E-Books Request Books Exit

Select Course and Subject

Select Course: BSc

Select Subject: Mathematics

- Mathematics
- Statistics
- Computer Science
- Data Science
- English

© 2024 Instant Library. All Rights Reserved.

Mathematica E-Books Page:

Mathematics E-Books					
S.No	Book Name	Book Author	Semester	Index Page	E-Book
1	MATHEMATICAL ANALYSIS	S C Malik, Savita Arora	sem 5	link	link
2	DIFFERENTIAL EQUATIONS AND THEIR APPLICATIONS	Zafar Ahsan	sem 1	link	link
3	NUMERICAL ANALYSIS	Richard L. Burden, J. Douglas Faires	sem 6	link	link
4	LINEAR ALGEBRA	Stephen H. Friedberg, Arnold J. Insel	sem 4	link	link
5	ALGEBRA	Chittaranjan Mallick, Susmitha Mallick	sem 3	link	link
6	INTEGRAL EQUATIONS	Shanti Swarup, Shiva Raj Singh	sem 2	link	link

© 2024 Instant Library. All Rights Reserved.

Statistics E-Books Page:

Statistics E-Books					
S.No	Book Name	Book Author	Semester	Index Page	E-Book
1	OPERATIONS RESEARCH	D.V.L.N. Jogiraju K.Ravikumar	Sem 6	link	link
2	STATISTICS AND PROBABILITY	A.Raghu Raj Bahadur	Sem 1	link	link
3	STATISTICAL INFERENCE	D.V.L.N. Jogiraju, C.Srikala, L.P.Rajkumar	Sem 2	link	link
4	STATISTICAL METHODS	D.V.L.N. Jogiraju, C.Srikala, L.P.Rajkumar	Sem 3	link	link
5	MATHEMATICAL STATISTICS	Sultan Chand & Sons	Sem 4	link	link
6	PRACTICAL STATISTICS	R.S.N. Pillai Bagavathi S.Chand	Sem 5	link	link

© 2024 Instant Library. All Rights Reserved.

Computer Science E-Books Page:

Computer Science E-Books					
S.No	Book Name	Book Author	Semester	Index Page	E-Book
1	Database Management System	Peter Rob, A. Ananda Rao, Carlos Coronel	Sem 4	link	pdf
2	PROGRAMMING IN JAVA	Sachin Malhotra, Saurabh Choudhary	Sem 5	link	pdf
3	PROGRAMMING WITH C	Sandeep Agarwalla, B. Rajani	Sem 1	link	pdf
4	SQL, PL/SQL	Ivan Bayross	Sem 4	link	pdf
5	WEB TECHNOLOGIES	S. Brinda, K. S.Rao	Sem 6	link	pdf
6	C++	Jesse Liberty	Sem 2	link	pdf
7	EXCEL FOUNDATION	Puneet Kumar, Shushil Bhardwaj	Sem 1	link	pdf
8	DATA STRUCTURES USING C & C++	Moshe J.Augstein	Sem 3	link	pdf

© 2024 Instant Library. All Rights Reserved.

DataScience E-Books Page:

Data Science E-Books					
S.No	Book Name	Book Author	Semester	Index Page	E-Book
1	EXCEL FOUNDATION	Puneet Kumar, Shushil Bhardwaj	sem 1	link	link
2	CYBER SECURITY	Puneet Kumar	sem 2	link	link

© 2024 Instant Library. All Rights Reserved.

English E-Books Page:

English E-Books					
S.No	Book Name	Book Author	Semester	Index Page	E-Book
1	The English Turf	C. Muralikrishna, Y.L. Srinivas	sem 1	link	pdf
2	ENGLISH & Soft Skills	S.P.Dhanavel	sem 2	link	pdf

© 2024 Instant Library. All Rights Reserved.

Request Books Page:

Request a Book	
Email:	<input type="text"/>
Name:	<input type="text"/>
Year:	<input type="text" value="Select Year"/>
Semester:	<input type="text" value="Select Semester"/>
Roll No.:	<input type="text"/>
Subject:	<input type="text" value="Select Subject"/>
Book Name:	<input type="text"/>
Description:	<input type="text"/>
<input type="button" value="Submit Request"/>	

© 2024 Instant Library. All Rights Reserved.

Book request submitted successfully! We will get back to you soon.

Exit:

Session Ended

Please close this tab.

Dynamodb Database updates :

1. Users table :

Users Autopreview View table details

▶ Scan or query items
Expand to query or scan items.

✔ Completed. Read capacity units consumed: 0.5

Items returned (3) Refresh Actions Create item

	email (String)	login_count	name	password
<input type="checkbox"/>	penubakulalalekhyas@...	1	Alekhyas	\$2b\$12\$cCDhD5jcnTmUf9DF5KouBvQxvGad15MrIMT3r5gmuAlplyf/6T6
<input type="checkbox"/>	alekhyas08022@gmail.com	1	Alekhyas	\$2b\$12\$mLnDh1QMh7leSKQau4ytsOCjPXlFQ6HdvYg6qU16GOMhdZ
<input type="checkbox"/>	sirichakkala@gmail.com	1	Siri Chakkala	\$2b\$12\$NvMRmgRbOUj8U3KdDHhOUH5w5/ApXg5LOt7k63cvWSWqOuDgRs6

2. Requests table :

Requests Autopreview View table details

► **Scan or query items**
 Expand to query or scan items.

✓ Completed. Read capacity units consumed: 0.5


Items returned (5) 🔄 Actions ▼ Create item

< 1 > 📄 🔍

<input type="checkbox"/>	email (String) ▼	book_na... ▼	description ▼	name ▼	roll_no ▼	semester ▼	subject ▼	year ▼
<input type="checkbox"/>	perubakulaalekhya@...	python	less stock	Alekhyia	1234	5	Data Science	3
<input type="checkbox"/>	alekhya080228@gmail...	applied stats	I need this bo...	Alekhyia	1234	3	Statistics	2
<input type="checkbox"/>	sini.chakkala@gmail.com	Python Pro...	Helle	Sini Chakkala	12	2	Data Science	1

3. Mail to the User:

Thank You for Your Book Request Inbox x

 **instantlibrary2@gmail.com**
 to me ▼

Dear Alekhya,

Thank you for submitting a book request for 'python'. We will get back to you soon.

↩ Reply
➡ Forward
😊

4. Mail to the admin:

New Book Request Inbox x



instantlibrary2@gmail.com

to me ▾

User Alekhya (penubakulaalekhya@gmail.com) has requested the book 'python'.

Details:

Year: 3

Semester: 5

Subject: Data Science

Description: less stock

↩ Reply

➦ Forward



Conclusion:

The Instant Library Website for Greenfield University has been successfully developed and deployed using a robust cloud-based architecture. By leveraging AWS services such as EC2 for hosting, DynamoDB for data management, and SNS for real-time notifications, the platform ensures reliable and scalable access to essential library services. This system addresses the challenge of limited physical resources by providing students with a convenient way to request books and receive timely updates, while library staff can efficiently manage and track these requests.

The cloud-native approach allows for seamless scalability, ensuring that as student demand increases, the platform can handle the load without compromising performance. The integration of Flask with AWS ensures that backend processes, including user authentication and book requests, run efficiently. The platform's testing phase has ensured that all functionalities, from user registration to book request notifications, work smoothly.

In conclusion, the Instant Library Website provides a modern, efficient solution for managing limited library resources, enhancing the overall user experience, and improving communication between students and library staff. This project is a testament to the potential of cloud-based systems in addressing real-world challenges in educational institutions.

