Mini Project on AWS Core Services

Aim: Personalized Book Recommendation System Objective: Create an AWS recommendation system that makes genre suggestions for books according to user tastes. User profiles and book genres are stored in DynamoDB, and a Lambda function analyses the data to offer tailored suggestions. Book details can be optionally stored on S3, and a front-end for user interactions can be hosted on EC2, with IAM guaranteeing data security.

Project Architecture Overview:

DynamoDB: stores user profiles, book genres, and preferences.
Lambda: processes user data and makes genre-based suggestions.
S3 (optional): Contains book cover photos, descriptions, and other information.
EC2: hosts a basic front-end that allows users to interact with the recommendation system.
IAM: manages rights and safeguards each service.

Step 1:

1. Set up DynamoDB tables.
2. Create DynamoDB tables.
3. Navigate to the DynamoDB console.
4. Click Create Table to create two tables:

Users' Table:

5. Primary key: UserId (String)
6. Attributes: Add characteristics like Genre Preferences (a list of string values, such as ["Mystery", "Fantasy"].

Book Table:

7. Primary Key: Genre (String).
8. Attributes: Include properties such as title, author, description, and cover image URL (if S3 is used for storage).

Step 2:

1. Add sample data to the Users and Books tables to replicate user profiles and book categories.
2. For instance, consider the Users table:
3. User ID: "User1"
4. Genre preferences: ["Fantasy", "Science Fiction"].
5. In the Books table, add books for each category you want to propose.

Step 3:

1. Configure S3 for Book Storage (Optional).
2. Create an S3 bucket.
3. Go to the S3 console and create a new bucket (for example, book-covers-bucket).
4. Configure permissions so that the bucket is not publicly accessible.
5. Upload book images or details.
6. You may upload photos, PDF previews, and other book resources.
7. Take note of the picture URLs, which will be used in DynamoDB's Books table.

Step 4:

1. Build a Lambda Function for Recommendations
2. Create a Lambda Function:
3. Navigate to the Lambda console and select Create Function.
4. Create an Author from scratch and name it (for example, BookRecommendationFunction).
5. Choose Python or Node.js as your runtime language.

Write the Lambda code:
Use the pseudo-code below as a guideline to extract the user's chosen genres and propose books.

```
import boto3
import json
# Initialize DynamoDB client
dynamodb = boto3.client('dynamodb')
def lambda_handler(event, context):
    # Extract user ID from the input event
    user_id = event['userId']
    # Fetch user preferences from DynamoDB
    user = dynamodb.get_item(
        TableName='Users',
        Key={'UserId': {'S': user_id}}
    )
    genre_preferences = user['Item']['GenrePreferences']['L']
    recommendations = []
    # Fetch books by genre from the Books table
    for genre in genre_preferences:
        books = dynamodb.query(
            TableName='Books',
            KeyConditionExpression='Genre = :genre',
            ExpressionAttributeValues={':genre': {'S': genre}}
        )
        recommendations.extend(books['Items'])
    # Return recommendations
    return {
        'statusCode': 200,
        'body': json.dumps(recommendations)
    }
```

Set Lambda permissions:

- Ensure that the Lambda function has permission to read data from DynamoDB.
- Add an IAM role to the Lambda function with the AmazonDynamoDBReadOnlyAccess policy.

Step 5:

1. Configure the IAM roles and permissions.
2. Create IAM roles.
3. Get to the IAM console.
4. Create a new Lambda role using the AmazonDynamoDBFullAccess policy.
5. Create a new EC2 role with access to the S3 bucket (if applicable) and read-only access to DynamoDB.
6. Set permissions for S3 (if applicable):
7. Attach a bucket policy to enable the EC2 instance or Lambda function to safely access S3 items.

Step 6:

1. Launch an EC2 instance for the front-end.
2. Navigate to the EC2 console.
3. Create an instance using Amazon Linux 2 or Ubuntu.
4. Set up security groups to enable HTTP (port 80) and SSH (port 22) access.
5. Install web server:
6. SSH into your EC2 instance

```
sudo yum install -y httpd
sudo systemctl start httpd
sudo systemctl enable httpd
```

Step 7:

1. Set up a simple front-end.
2. Create a simple HTML/JavaScript frontend to interact with the Lambda function.
3. Make a JavaScript API call to the Lambda function via API Gateway (optional) to get suggestions.

Test the system:

1. Add Sample Users and Book Genres to DynamoDB: Make sure you have useful test data.
2. Invoke Lambda Function Manually (for Testing): To test, launch the Lambda console and enter an example userID event.
3. Access the web front-end. Open your browser, navigate to the EC2 instance's public IP address, and interact with the system.