

**MOVIE RECOMMENDATION SYSTEM**

**Final Project Report**

**CS 5610 INTRODUCTION TO CLOUD COMPUTING**

**CRN- 13526**

**Cherukuri Vandana -700759337**

**Apoorva Kyramkonda - 700742326**

**Naveen Panneti - 700757541**

**Vennela Madamshetty - 700736655**

**ABSTRACT**

The Movie Recommendation System is an innovative cloud-based application. The project integrates cutting-edge technologies and AWS services to deliver a personalized and responsive movie recommendation experience. Utilizing a robust stack, the system employs AWS services such as API Gateway, Lambda, DynamoDB, CloudWatch, EC2, and IAM, showcasing the practical application of cloud computing principles.

To develop this project, we are using some AWS services like **Api Gateway, Lambda, Dynamodb, Cloud Watch, EC2, and IAM**.

Along with AWS service, we also use other technologies like **Python, Django, HTML, CSS, BootStrap, and JavaScript.**

Application is developed over Python Django web framework. The application is hosted over an EC2 server. The application will integrate into the backend system via API. API will be created & and published using API GW.API GW will further forward requests toward lambda.

DynamoDB, a NoSQL database, serves as the system's data store. It efficiently manages user preferences and movie details, offering a scalable solution for storing and retrieving data. CloudWatch is integrated to monitor system performance, providing logging and alerting functionalities

The backend of the system is powered by the Django web framework, hosted on an EC2 server. Django's versatility and modularity streamline the development process, providing an organized structure for server-side logic. On the front end, HTML, CSS, Bootstrap, and JavaScript collectively contribute to an engaging user interface.

Overall, the Movie Recommendation System not only demonstrates the integration of cloud services into real-world applications but also emphasizes the significance of a user-centric design approach. The project underscores the principles of scalability, reliability, and efficiency in the context of cloud-based development, offering valuable insights into the intersection of cloud computing and modern web technologies.

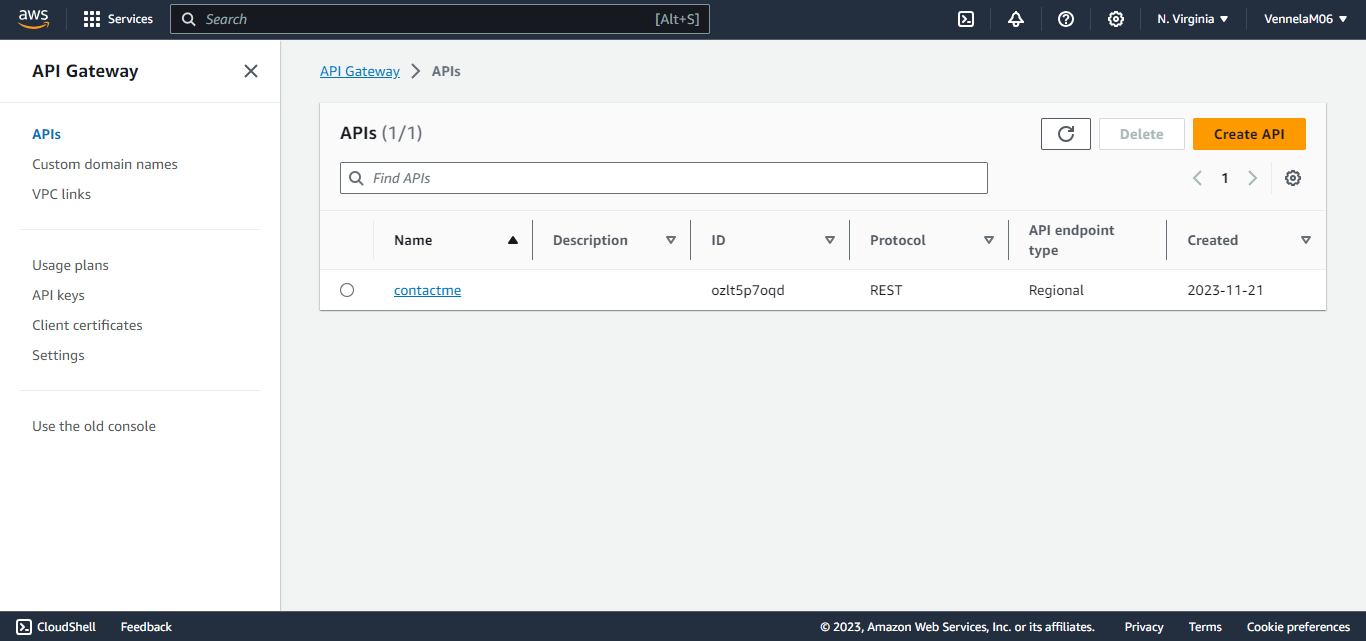
1. **System Architecture**
   1. **Overview**

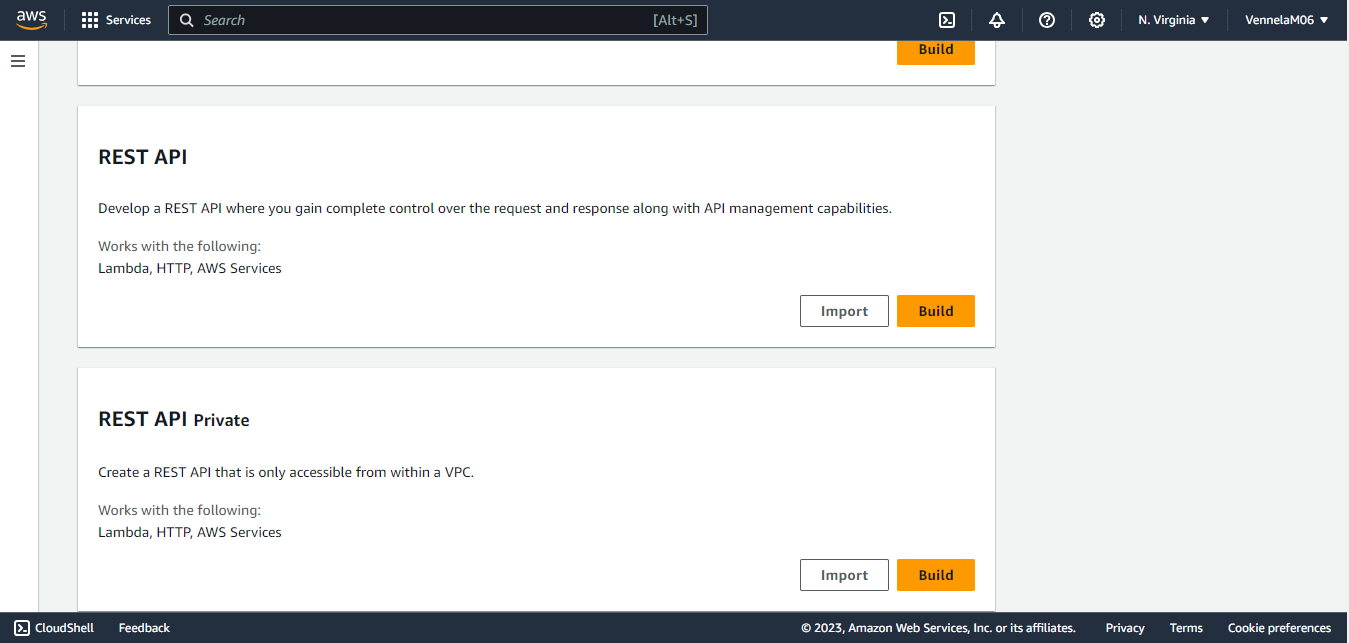
The system architecture involves the integration of various AWS services to handle different aspects of the recommendation system. The front end is developed using HTML, CSS, Bootstrap, and JavaScript, while the back end is powered by Django.

* 1. **AWS Services**

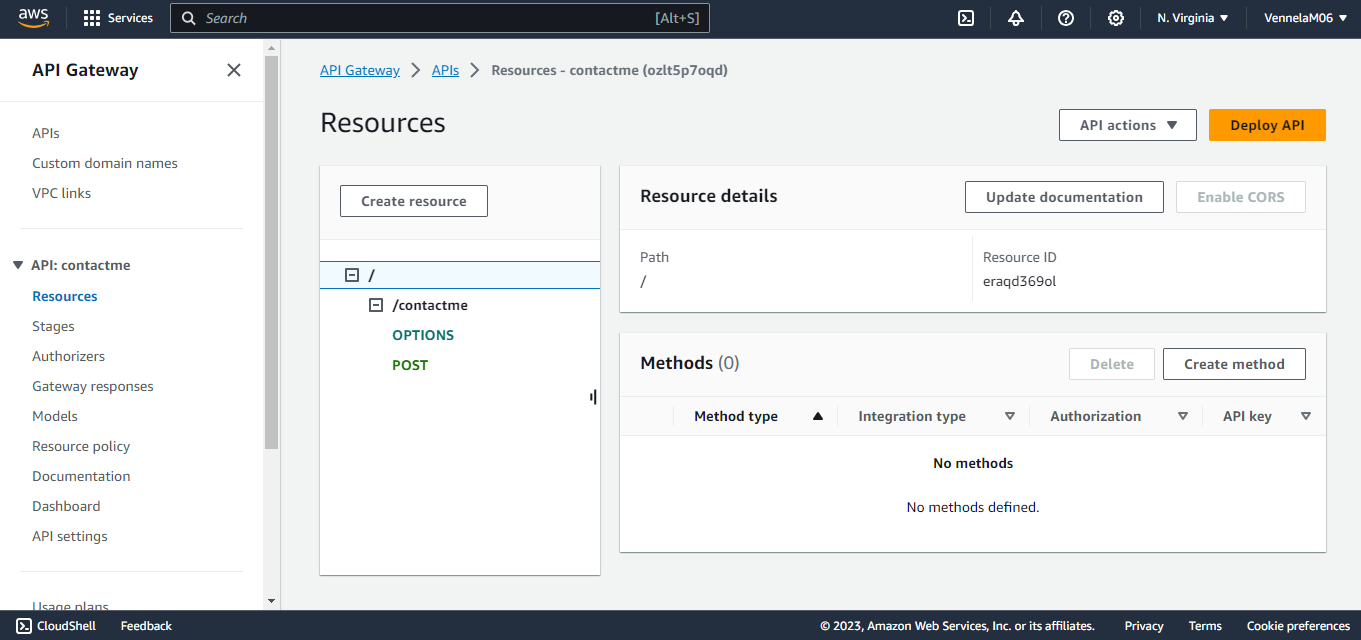
**Services Used:**

**AWS API GW** – Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. APIs act as the "front door" for applications to access data, business logic, or functionality from your backend services. Using API Gateway, you can create RESTful APIs and WebSocket APIs that enable real-time two-way communication applications. API Gateway supports containerized and serverless workloads, as well as web applications.



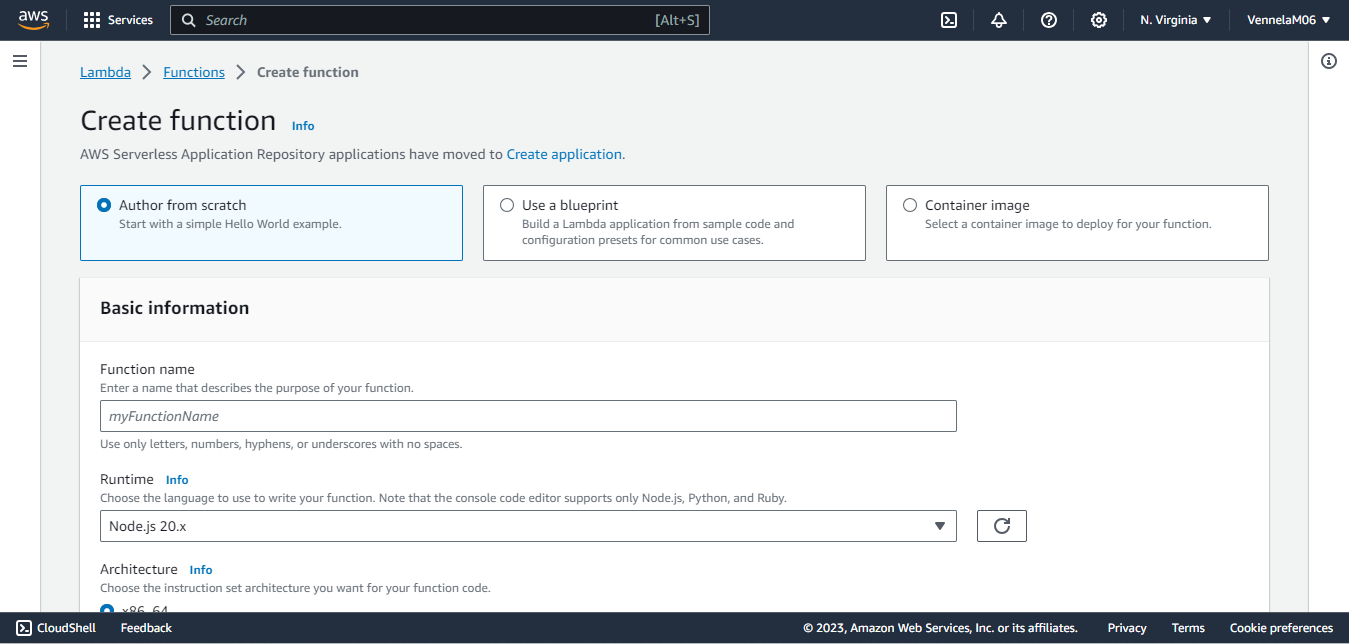


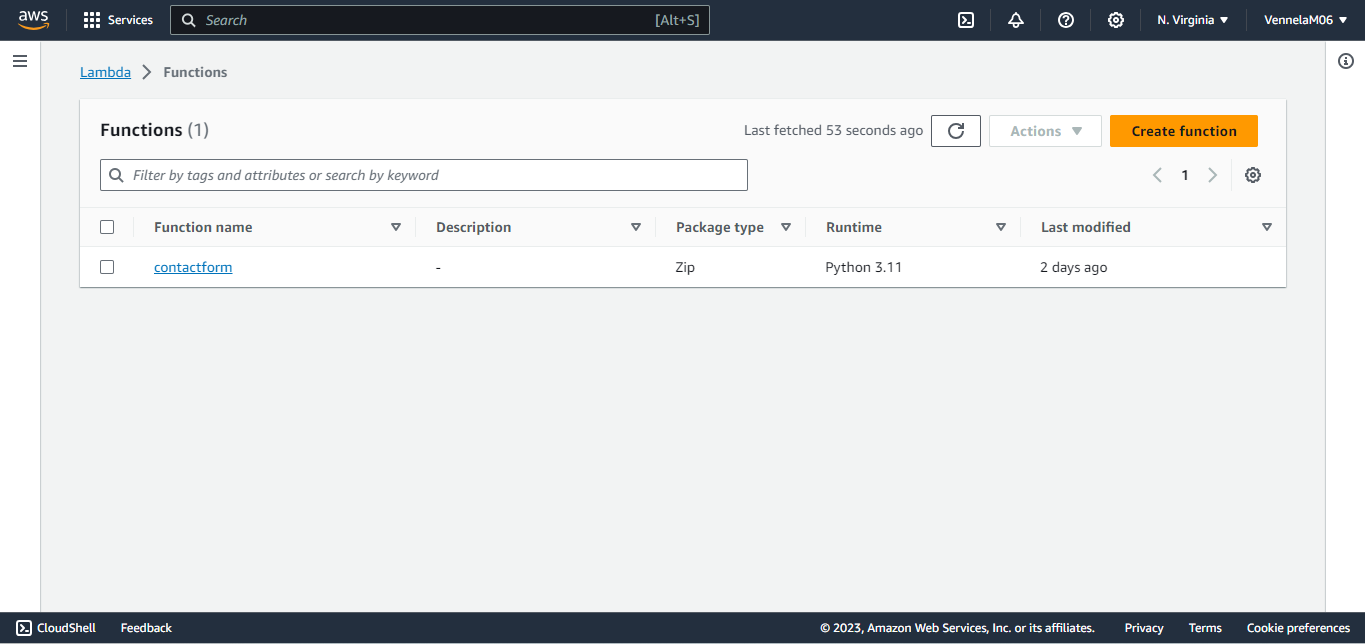


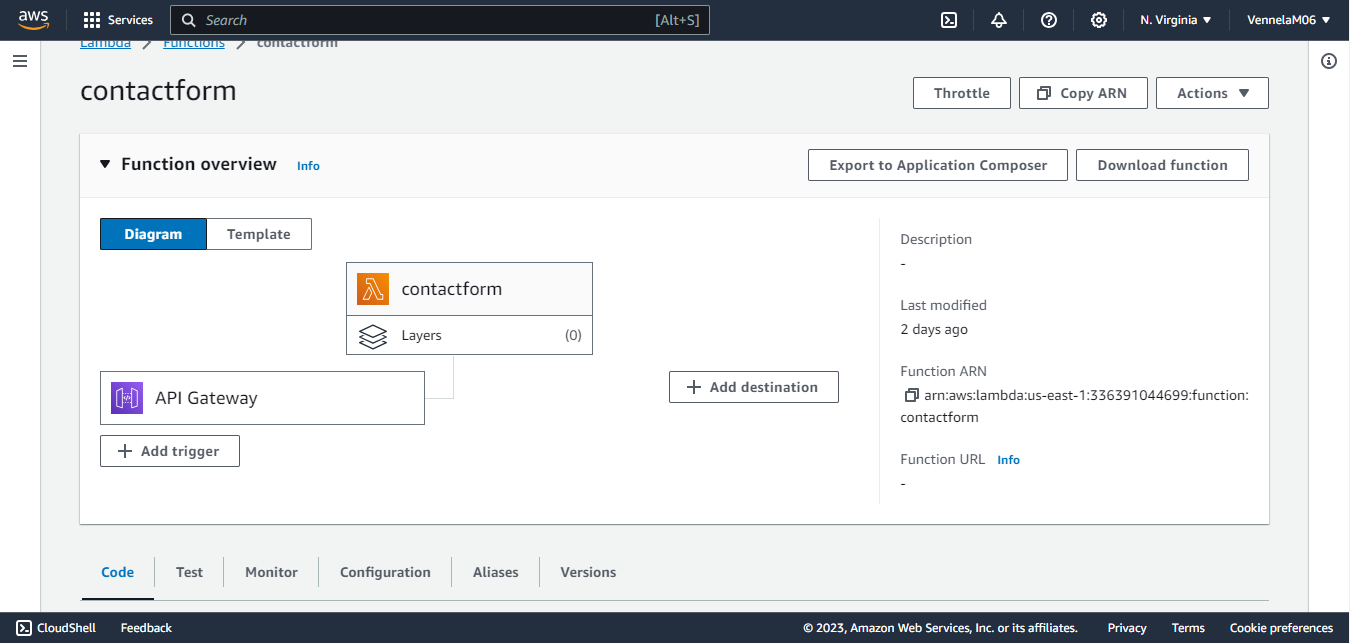


**AWS Lambda** - AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. Automatically respond to code execution requests at any scale, from a dozen events per day to hundreds of thousands per second.

* Save costs by paying only for the compute time you use—by per-millisecond—instead of provisioning infrastructure upfront for peak capacity.
* Optimize code execution time and performance with the right function memory size. Respond to high demand in double-digit milliseconds with Provisioned Concurrency.



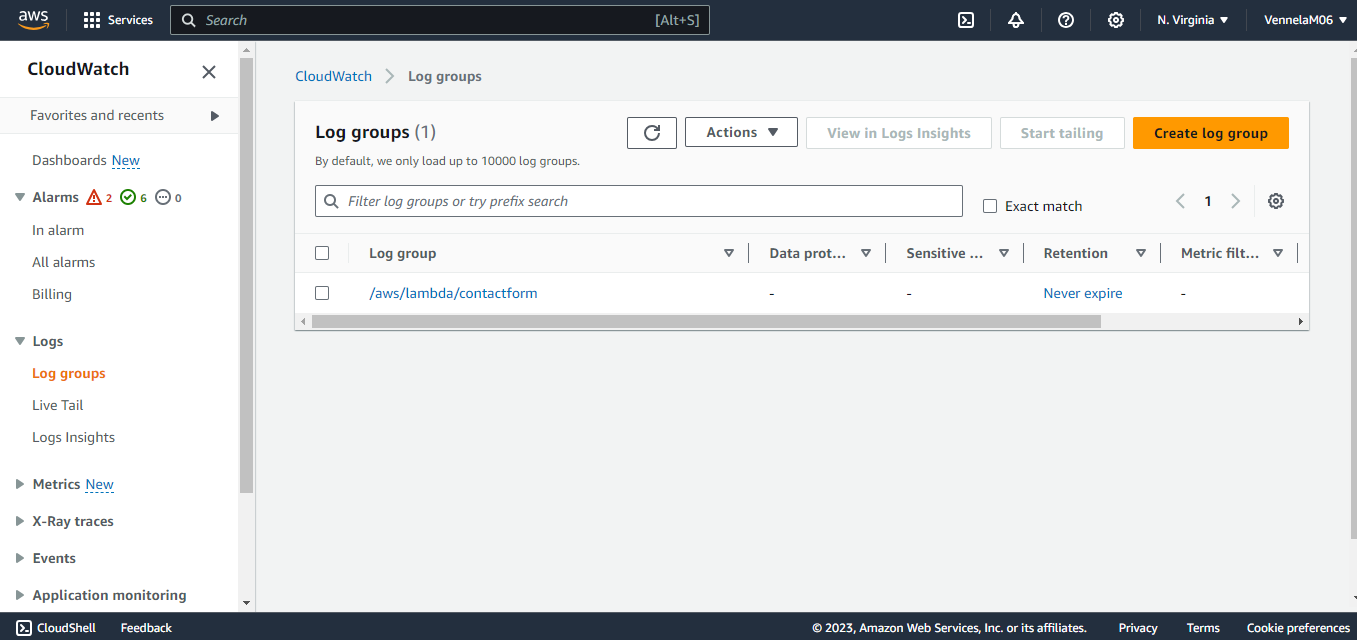


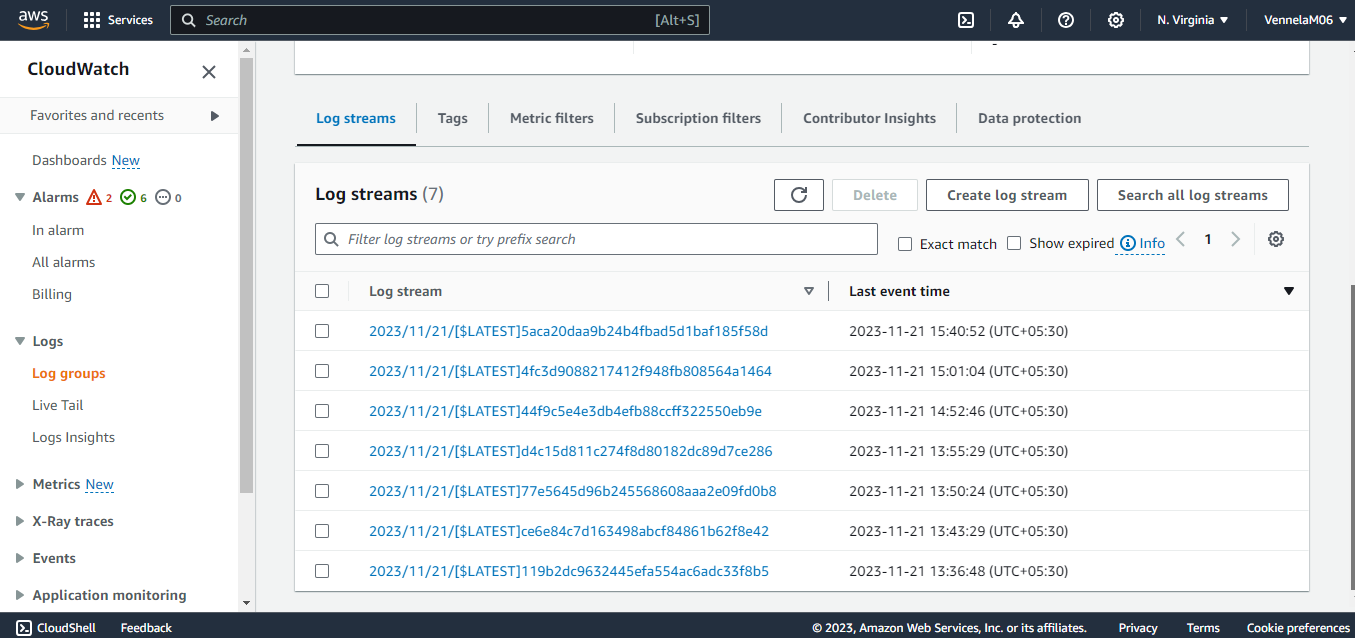




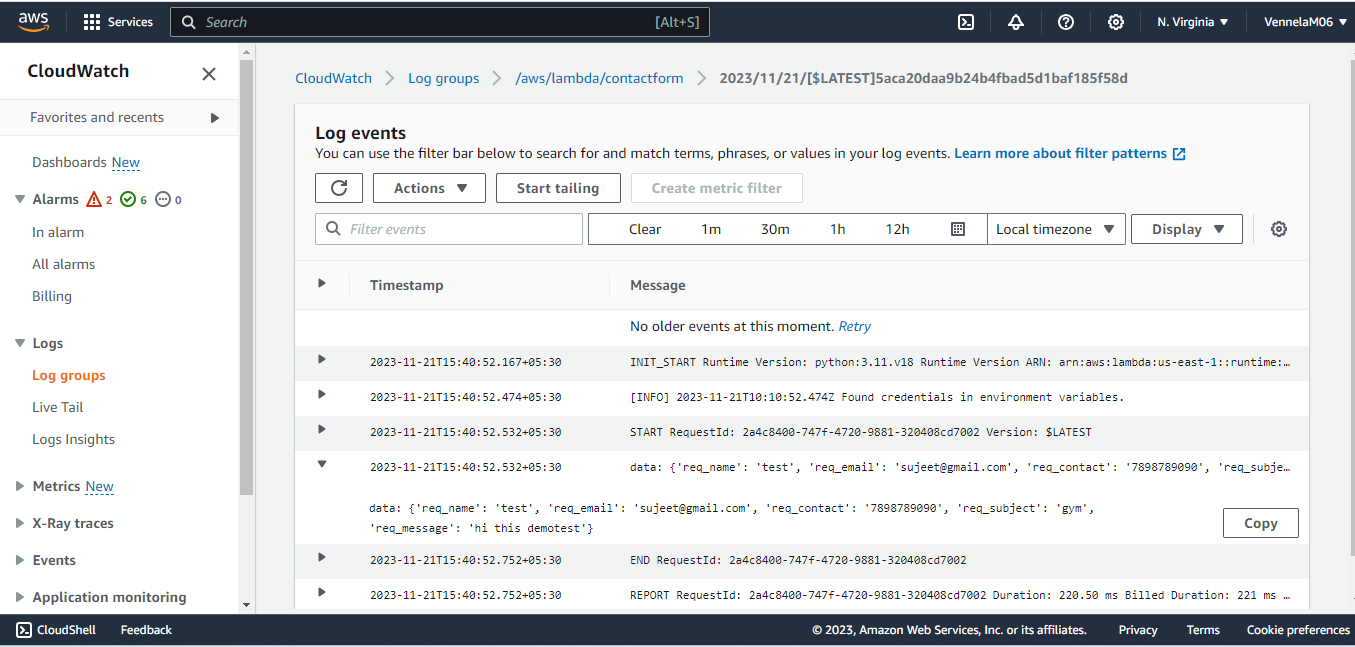
**AWS CloudWatch** - Amazon CloudWatch is a monitoring and observability service built for DevOps engineers, developers, site reliability engineers (SREs), IT managers, and product owners.

* CloudWatch provides you with data and actionable insights to monitor your applications, respond to system-wide performance changes, and optimize resource utilization.
* CloudWatch collects monitoring and operational data in the form of logs, metrics, and events.
* CloudWatch can used to detect anomalous behavior in your environments, set alarms, visualize logs and metrics side by side, take automated actions, troubleshoot issues, and discover insights to keep your applications running smoothly.





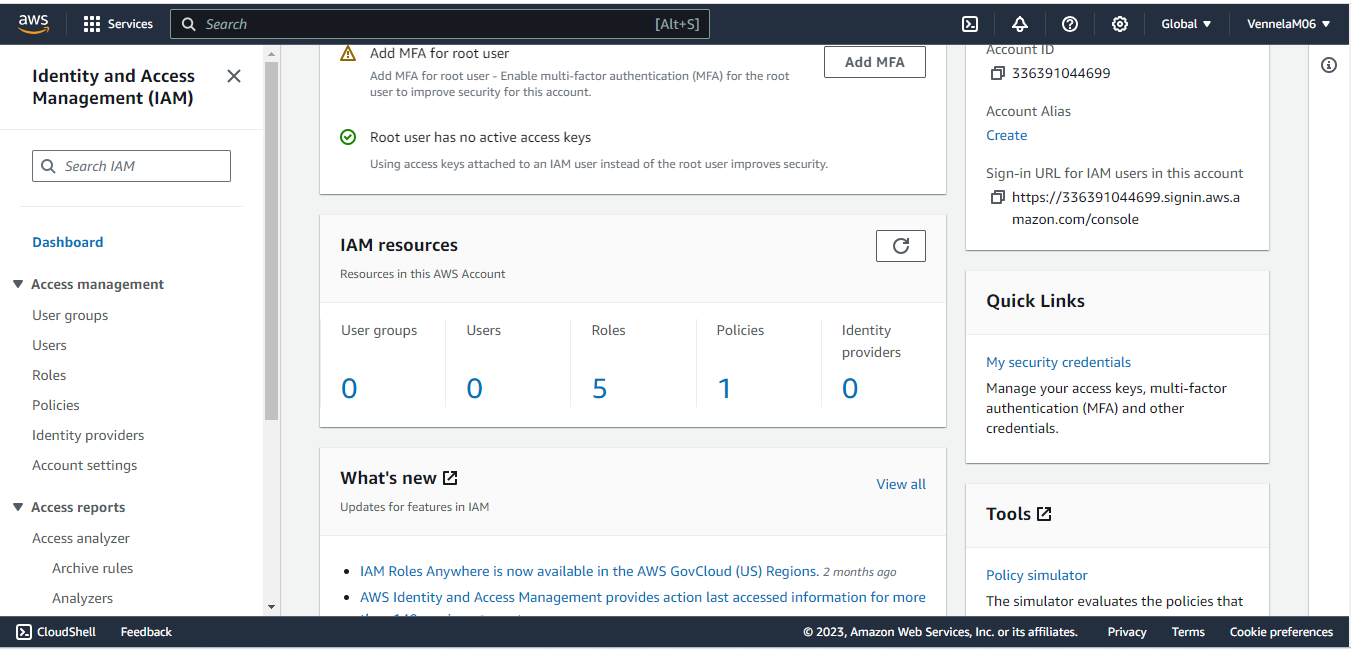


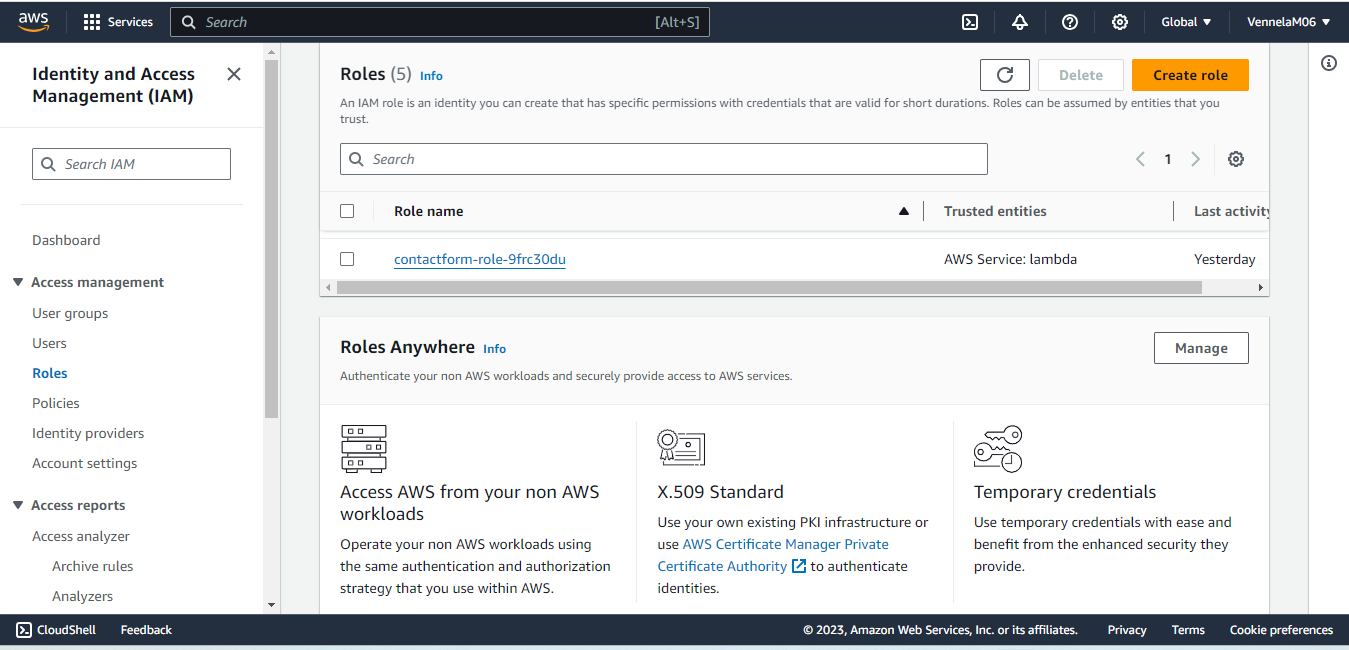


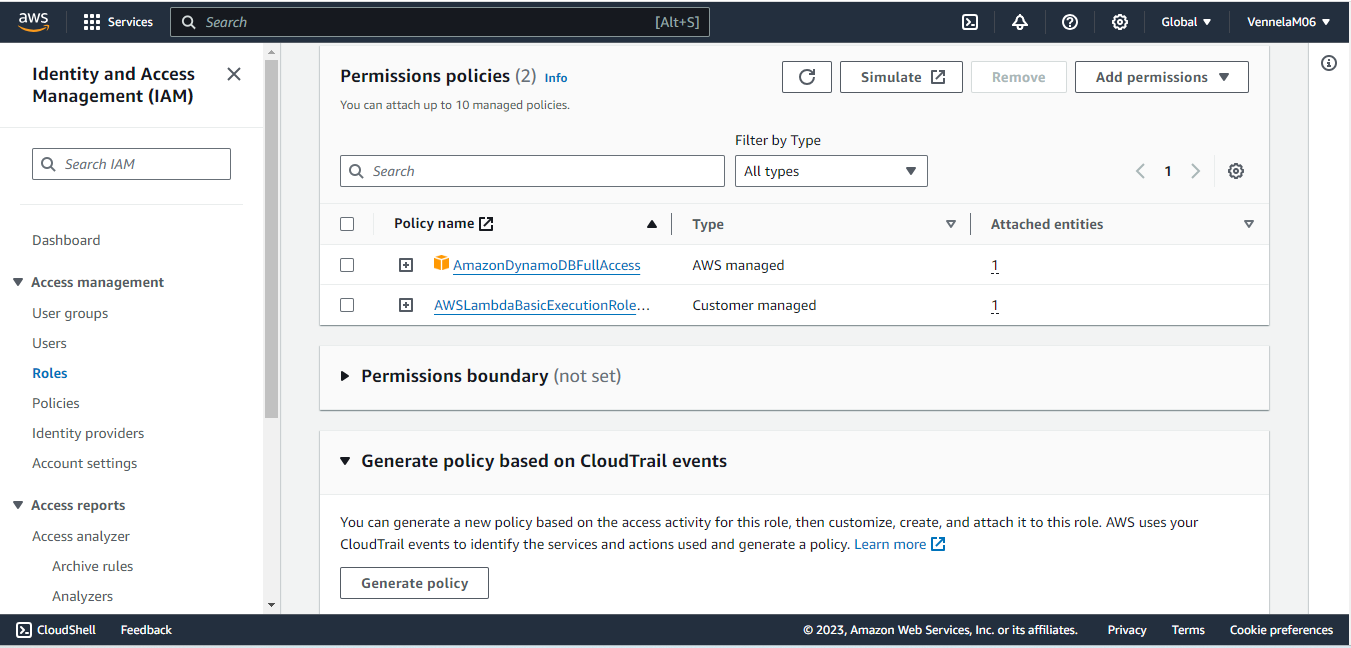
**AWS IAM** - AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS resources.

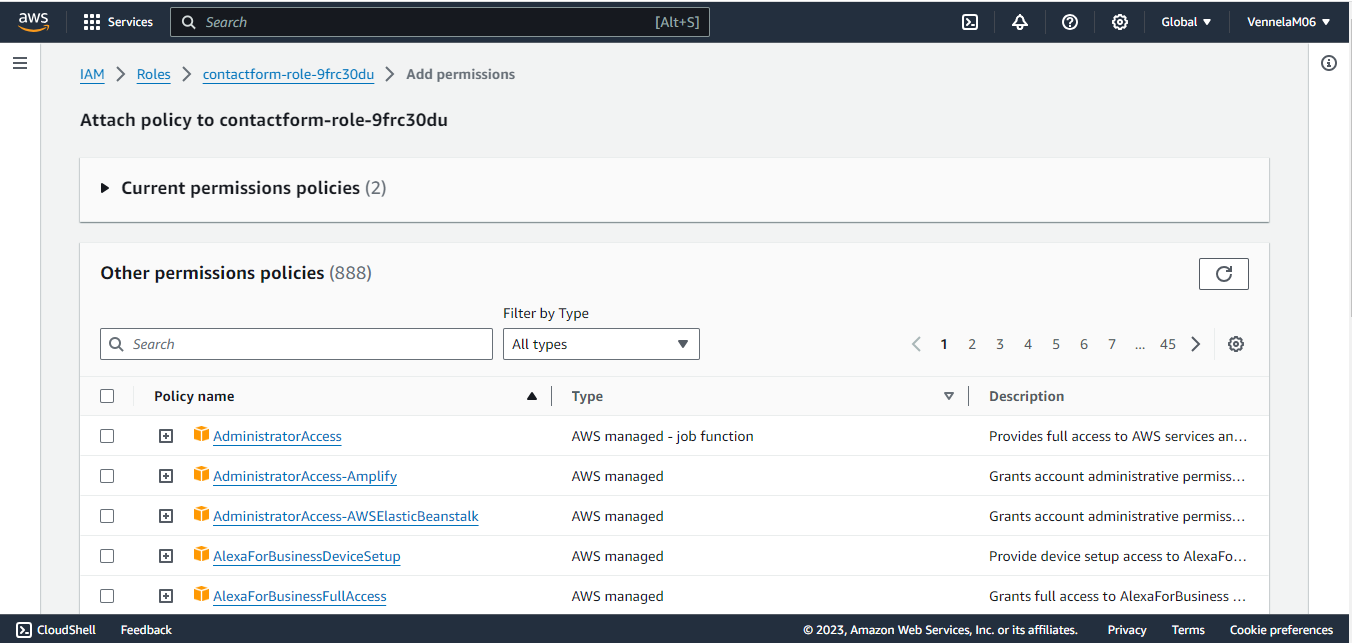
**The IAM workflow includes the following six elements:**

* A principal is an entity that can perform actions on an AWS resource. A user, a role, or an application can be a principal.
* Authentication is the process of confirming the identity of the principal trying to access an AWS product. The principal must provide its credentials or required keys for authentication.
* **Request:** A principal sends a request to AWS specifying the action and which resource should perform it.
* **Authorization:** By default, all resources are denied. IAM authorizes a request only if all parts of the request are allowed by a matching policy. After authenticating and authorizing the request, AWS approves the action.
* Actions are used to view, create, edit, or delete a resource.
* **Resources:** A set of actions can be performed on a resource related to your AWS account.

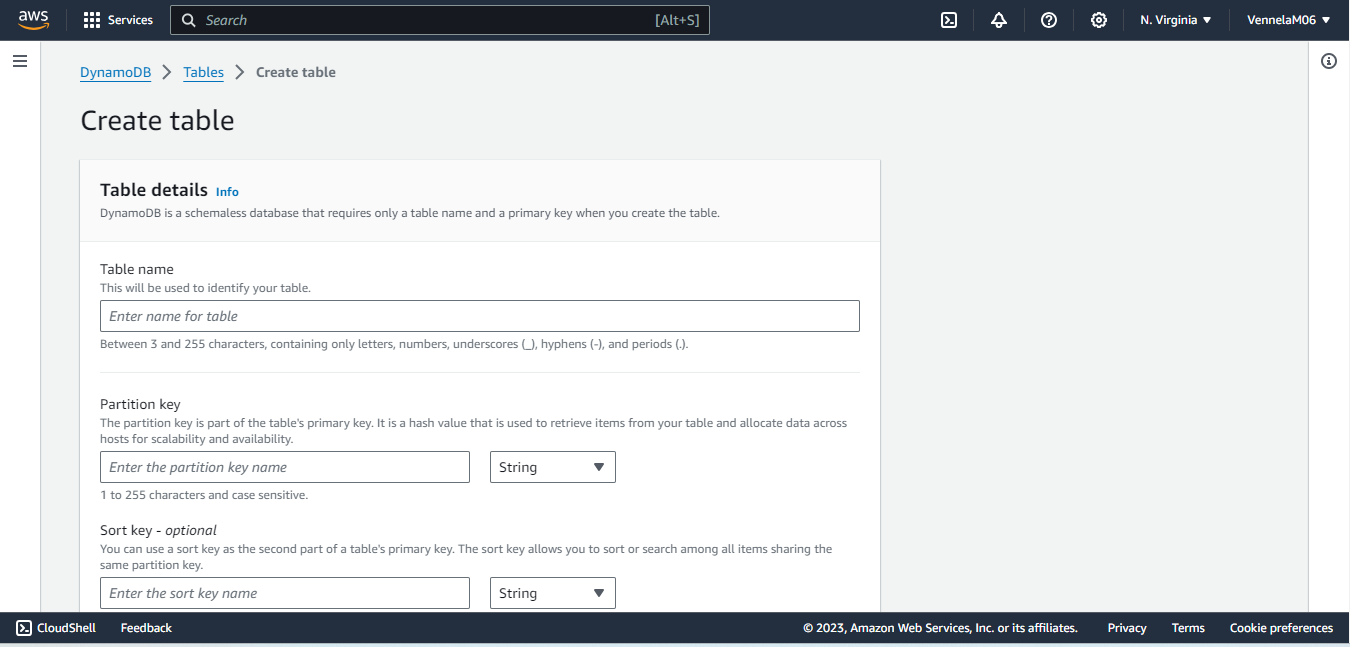


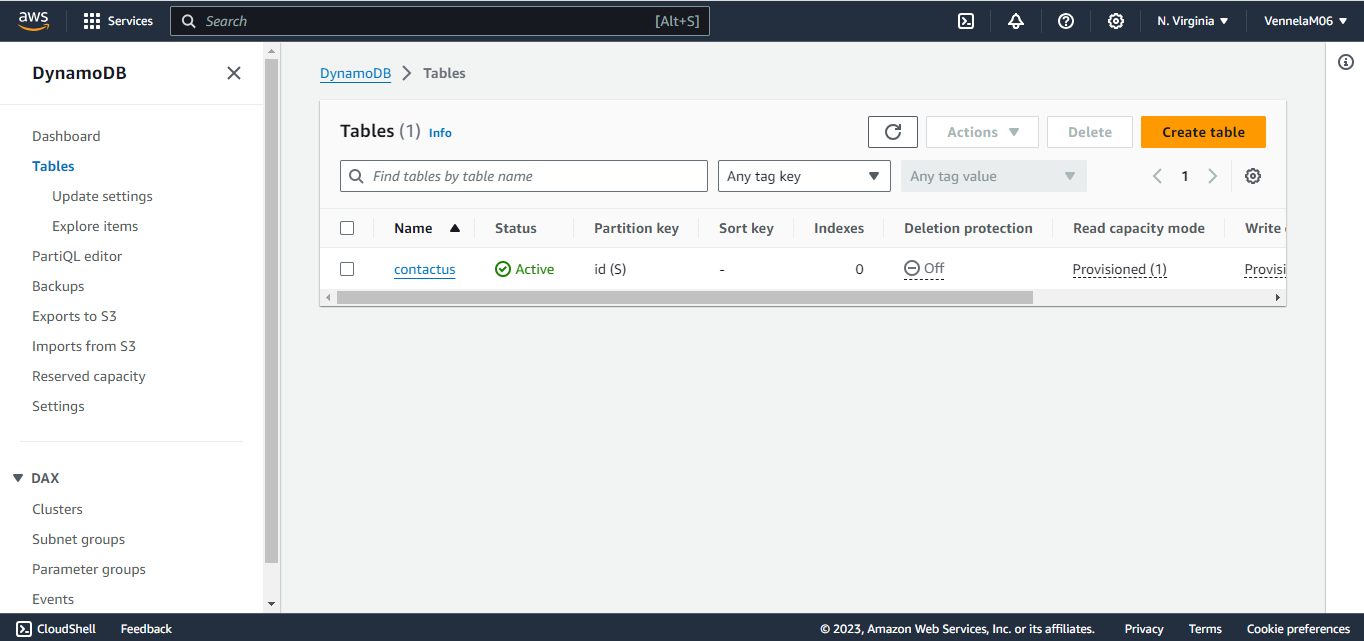


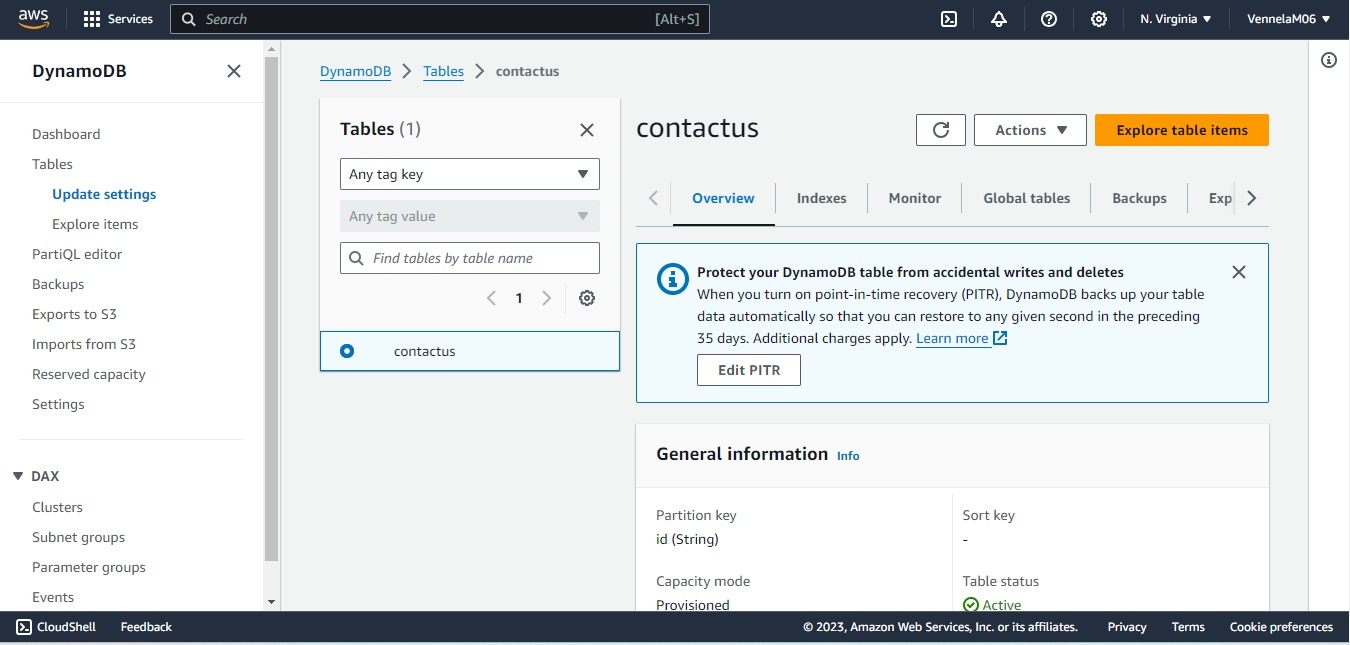


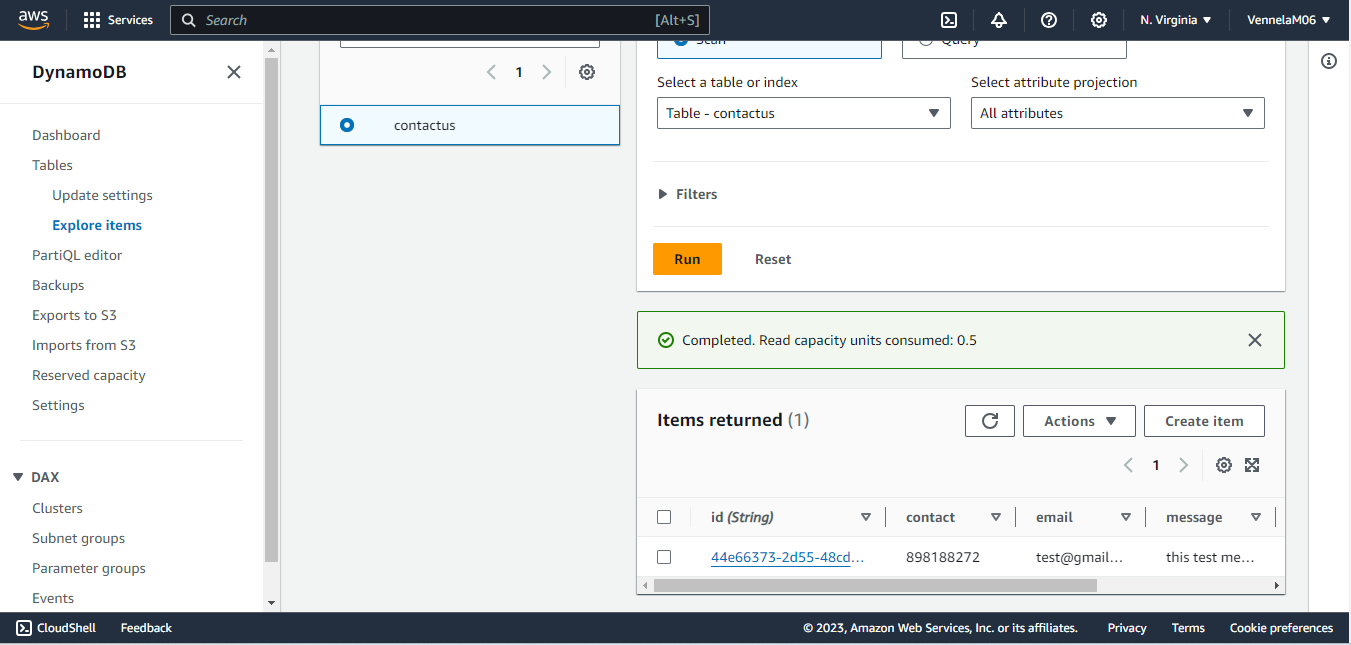


**AWS DynamoDB** - Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB lets you offload the administrative burdens of operating and scaling a distributed database so that you don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling. DynamoDB also offers encryption at rest, which eliminates the operational burden and complexity involved in protecting sensitive data. Relational databases are often used for storing transactional and analytical data.



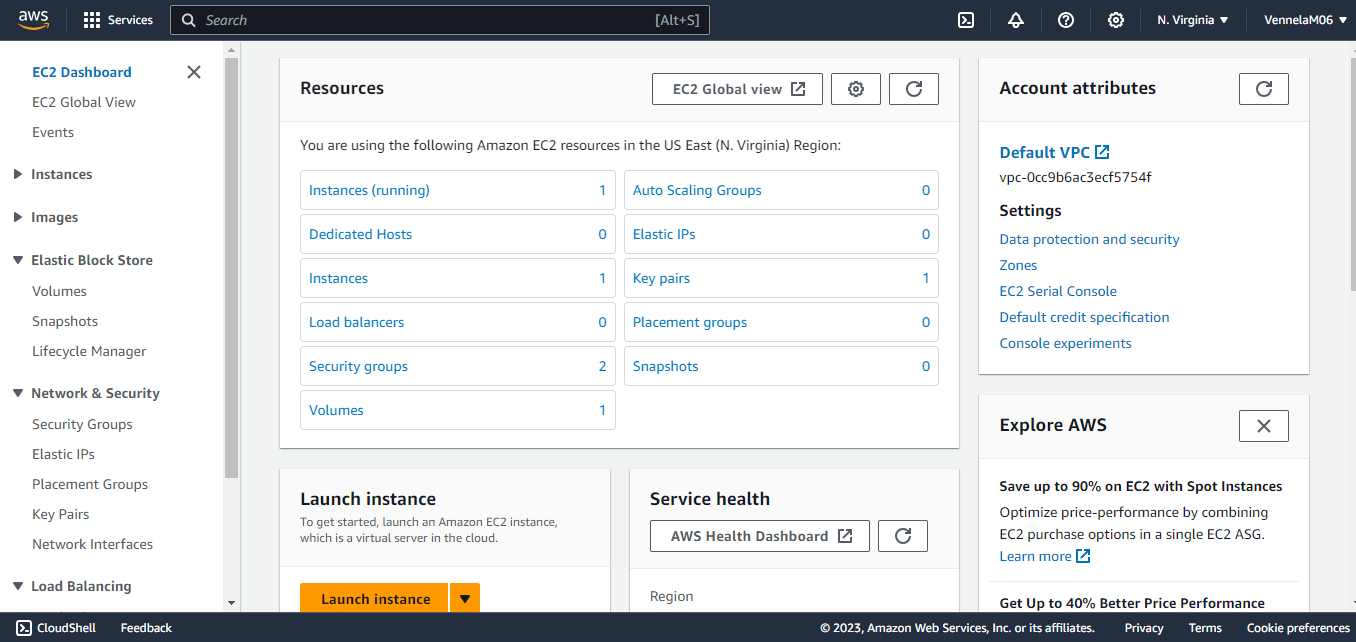


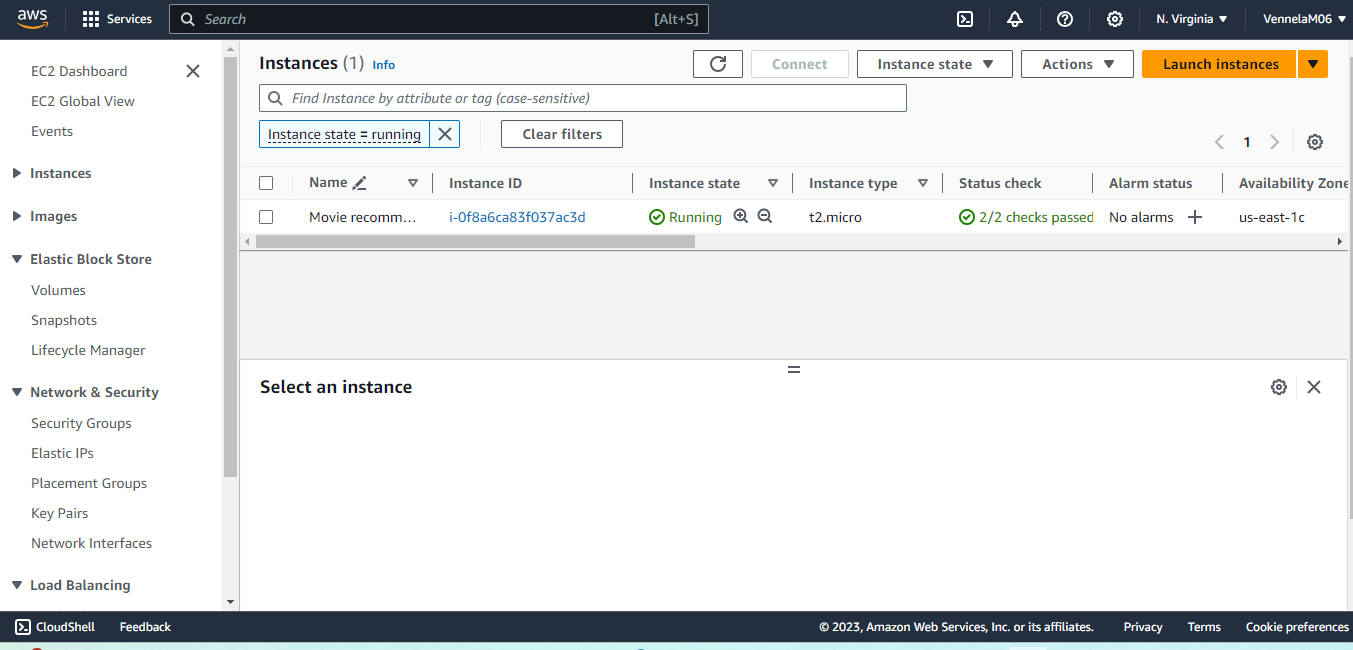


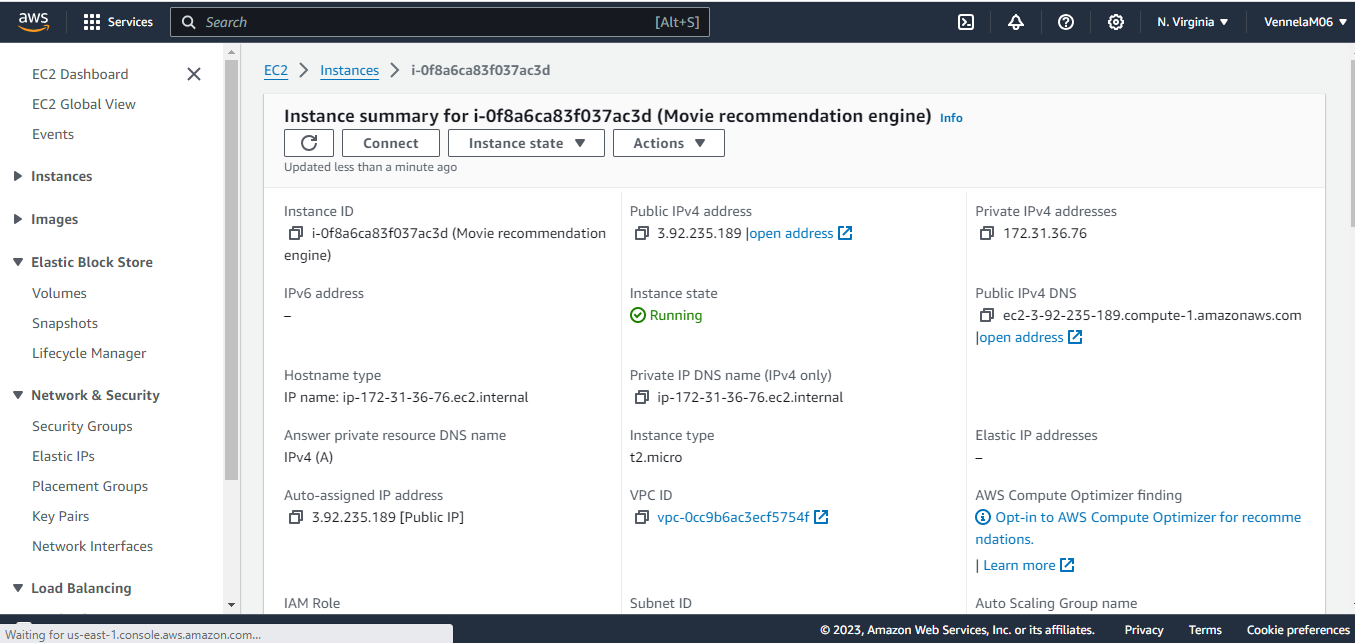


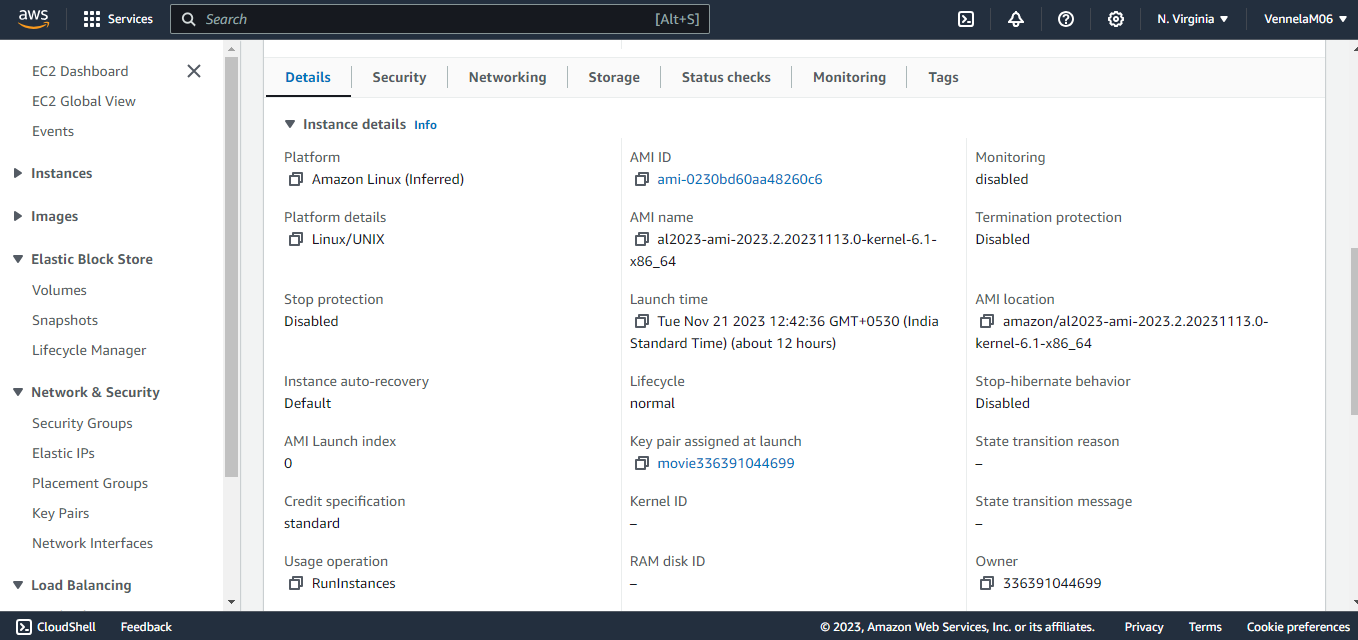
**AWS EC2** - Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2’s simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon’s proven computing environment.

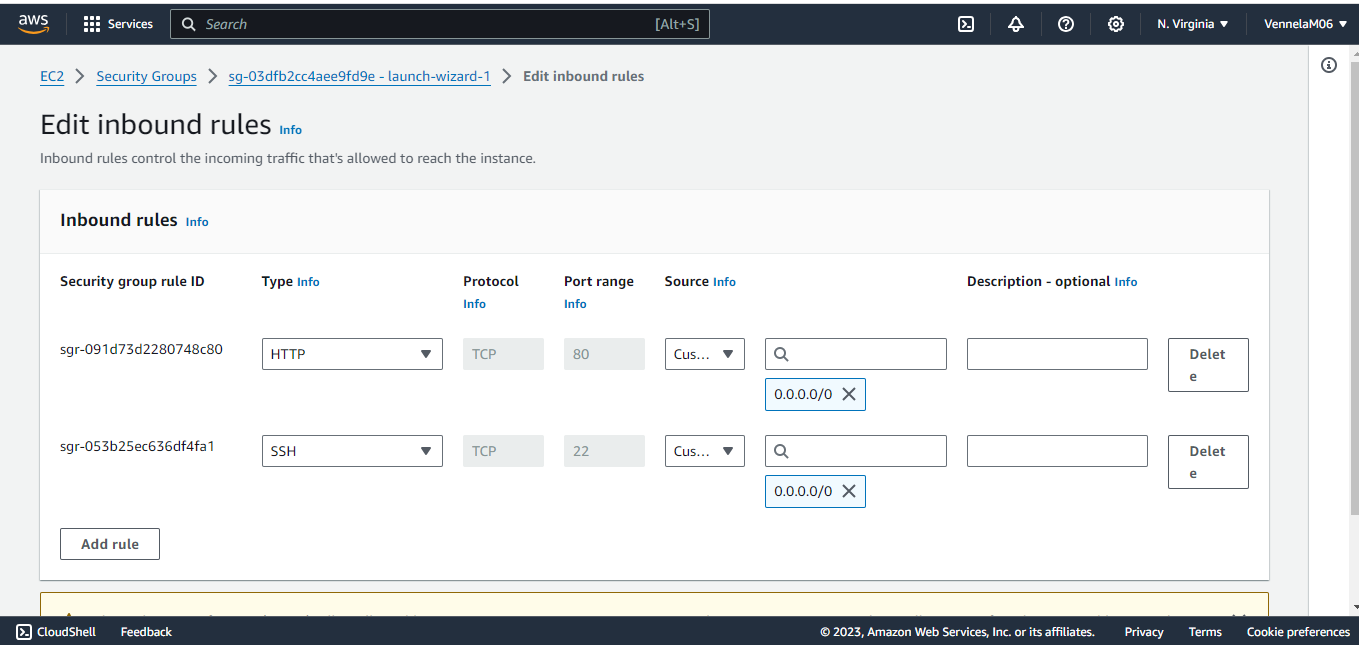
Amazon EC2 offers the broadest and deepest compute platform with choice of processor, storage, networking, operating system, and purchase model. We offer the fastest processors in the cloud and we are the only cloud with 400 Gbps ethernet networking. We have the most powerful GPU instances for machine learning training and graphics workloads, as well as the lowest cost-per-inference instances in the cloud. More SAP, HPC, Machine Learning, and Windows workloads run on AWS than any other cloud.











**Django (Python Web Framework):**

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

* **Ridiculously fast:** Django was designed to help developers take applications from concept to completion as quickly as possible.
* **Reassuringly secure:** Django takes security seriously and helps developers avoid many common security mistakes.
* **Exceedingly scalable:** Some of the busiest sites on the web leverage Django’s ability to quickly and flexibly scale.

**2. Technologies**

**2.1. Python:**

Python is the backend programming language. It is used to implement server-side logic, interact with AWS services (through SDKs), and process data. Python's readability and versatility make it well-suited for backend development.

**2.2. Django:**

Django is the web framework for backend development. It provides an ORM for database interactions, a templating engine for rendering HTML, and a clean structure for organizing code. Django simplifies the development of the backend logic, making it modular and maintainable.

**2.3. HTML, CSS, Bootstrap:**

HTML structures the content of the frontend, CSS styles the presentation, and Bootstrap is used for frontend styling and responsiveness. Together, these technologies create a visually appealing and user-friendly interface for interacting with the Movie Recommendation System.

**2.4. JavaScript:**

JavaScript enhances the user interface by adding dynamic features. It is used to create interactive elements, handle user events, and enable real-time updates without requiring a full page reload. JavaScript contributes to a more engaging and responsive user experience.

**3. IAM (Identity and Access Management)**

**3.1. EC2 Instance Hosting the Application:**

**IAM Role: "MovieRecommendationEC2Role"**

IAM role is associated with the EC2 instance hosting the Django web application. It ensures that the EC2 instance has the necessary permissions to interact with DynamoDB for data retrieval, invoke Lambda functions for recommendation generation, and log relevant activities to CloudWatch. This role follows the principle of least privilege, granting only the essential permissions required for the EC2 instance to perform its tasks.

**3.2. Lambda Functions for Recommendation Generation:**

**IAM Role: "RecommendationLambdaRole"**

**Description:** IAM role assigned to the Lambda functions responsible for generating movie recommendations. This role grants the necessary permissions to read data from DynamoDB tables to process recommendations and write logs to CloudWatch for monitoring and analysis. By having a dedicated IAM role, the Lambda functions operate with specific and limited permissions, enhancing the overall security posture.

**3.3. API Gateway:**

**IAM Role: "APIGatewayRole"**

**Description:** TheIAM role is associated with the API Gateway, which serves as the interface for external interactions with the recommendation system. The role grants permissions to invoke the Lambda functions responsible for processing recommendations. By restricting access to only the necessary actions, this IAM role ensures that the API Gateway can facilitate communication between the external user interface and the backend services securely.

**3.4. Django Web Framework Application:**

**IAM Role: "DjangoApplicationRole"**

**Description:** IAM roles are configured within the Django application to grant access to AWS services. This includes permissions to interact with DynamoDB for fetching movie data. The role is tailored to the specific needs of the Django application, following the principle of least privilege.

**4. Features**

* Users can register and log in.
* Users can search through various movies and look through their details.
* Users can give ratings to the movies.
* User can add movies to their watch list.
* Users can get movie recommendations (Recommendation algorithm (Collaborative Filtering) which suggests new movies based on the ratings given by the user.)

**5. Collaborative Filtering (Recommender Algorithm)**

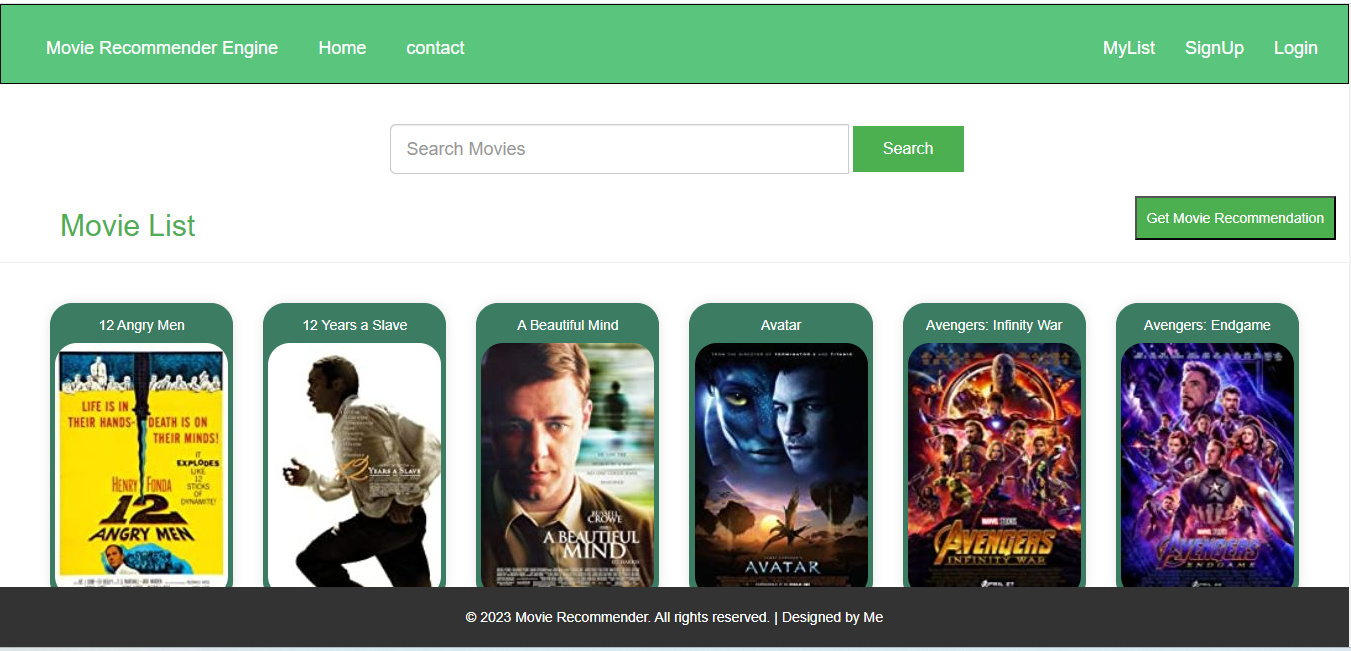
Collaborative filtering filters information by using the interactions and data collected by the system from other users. It's based on the idea that people who agree in their evaluation of certain items are likely to agree again in the future.

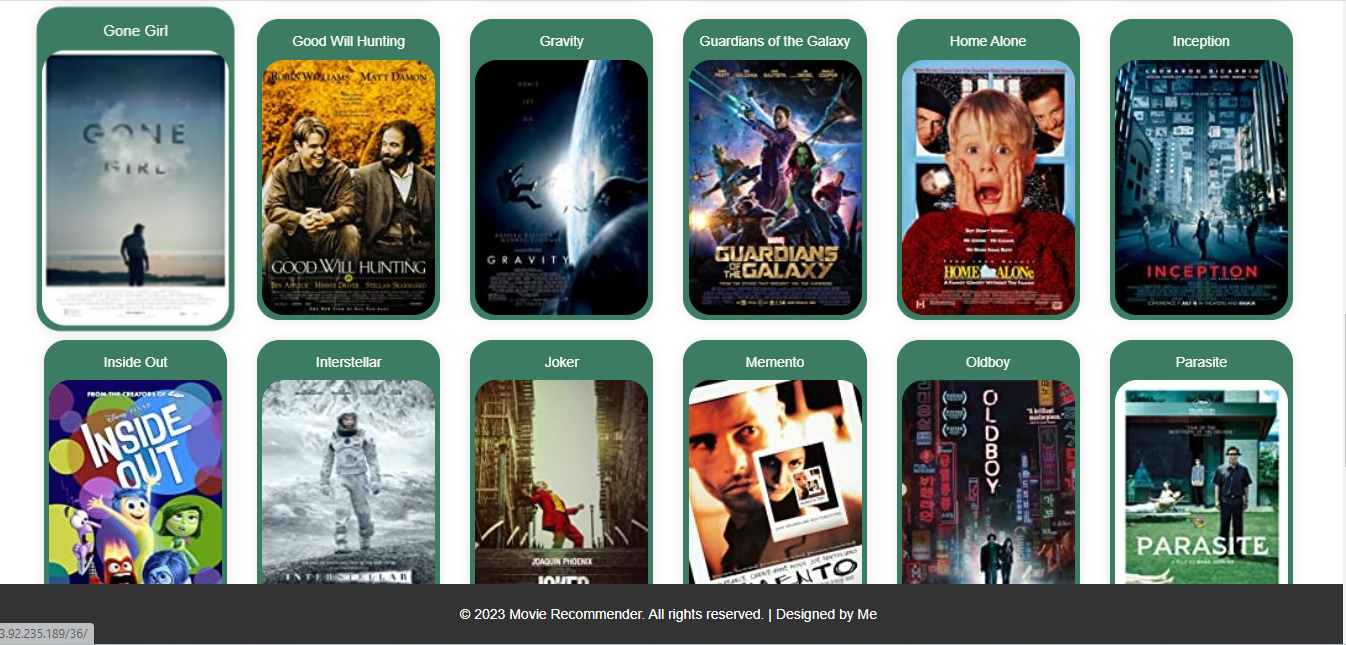
* When we want to find a new movie to watch we'll often ask our friends for recommendations. Naturally, we have greater trust in the recommendations from friends who share tastes like our own.
* Collaborative filtering systems focus on the relationship between users and items. The similarity of items is determined by the similarity of the ratings of those items by the users who have rated both items.

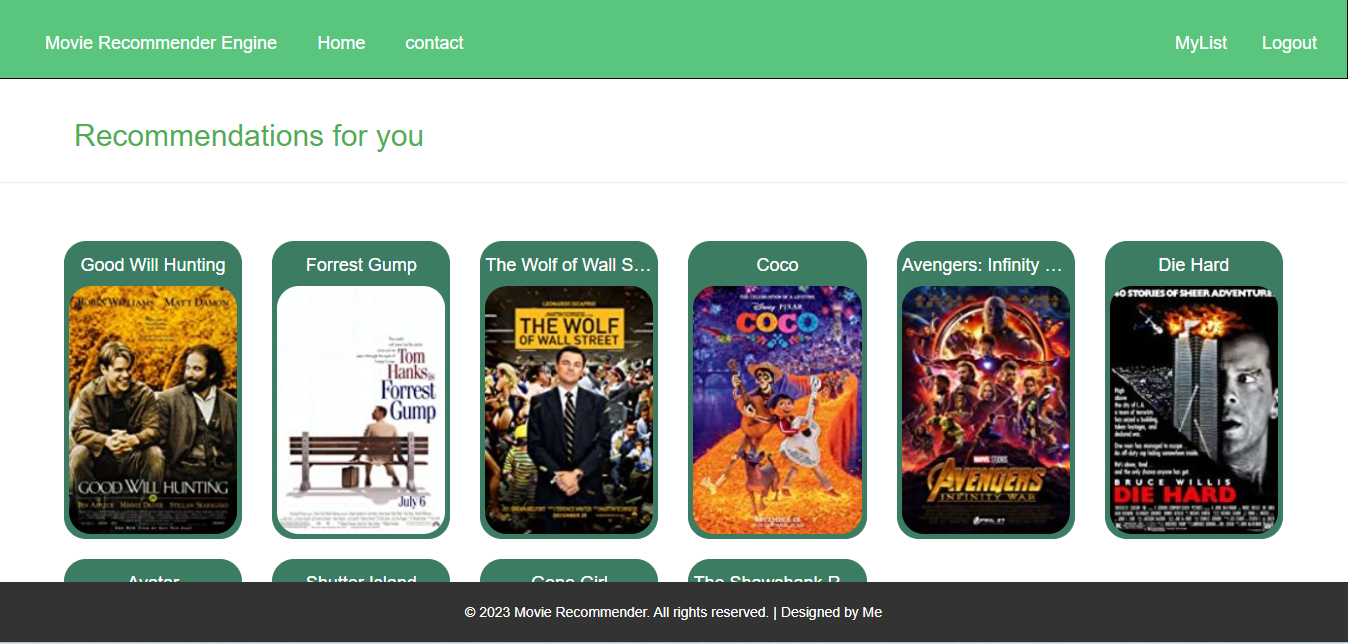
**There are two types of collaborative filtering.**

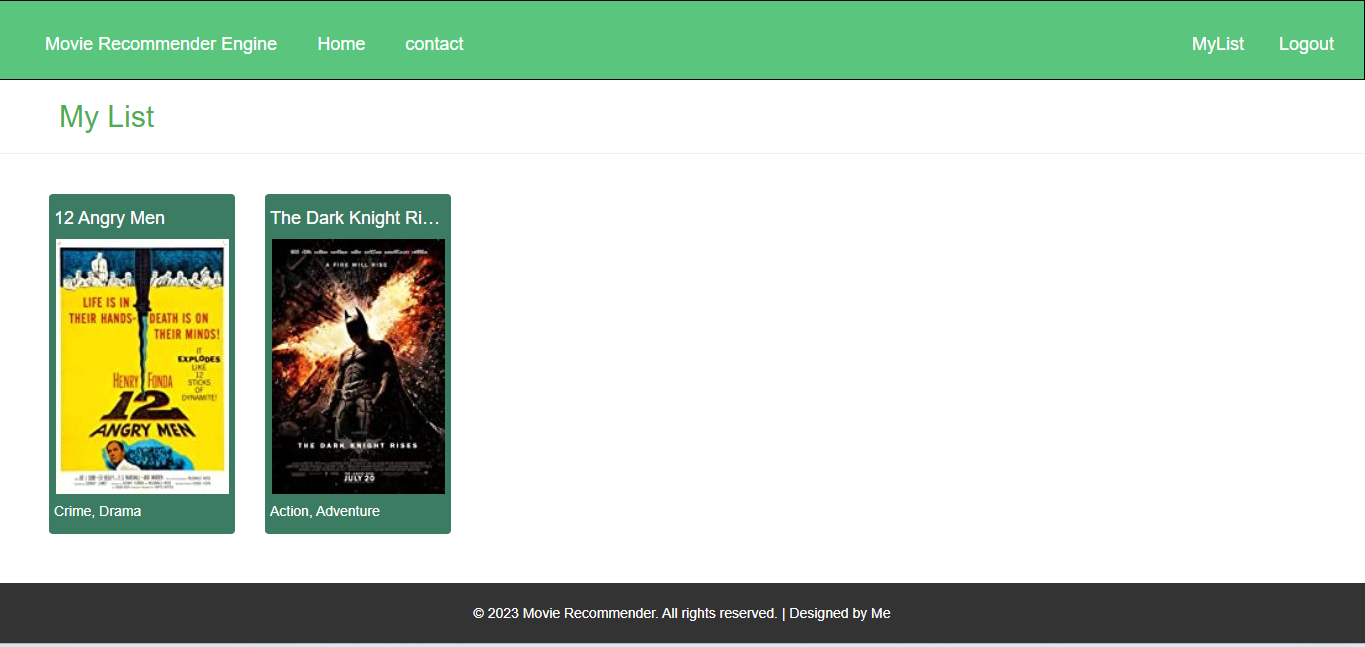
* **User-based**, which measures the similarity between target users and other users.
* **Item-based**, which measures the similarity between the items that target users rate or interact with and other items.

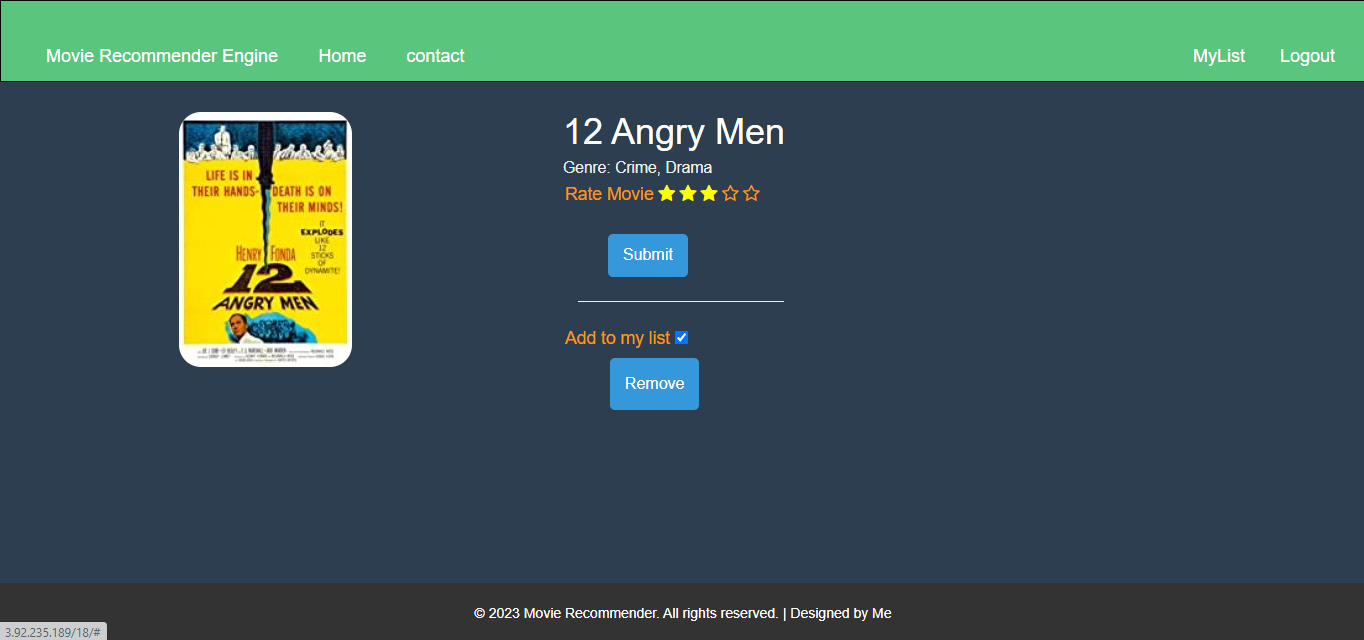
**6. User Interface**

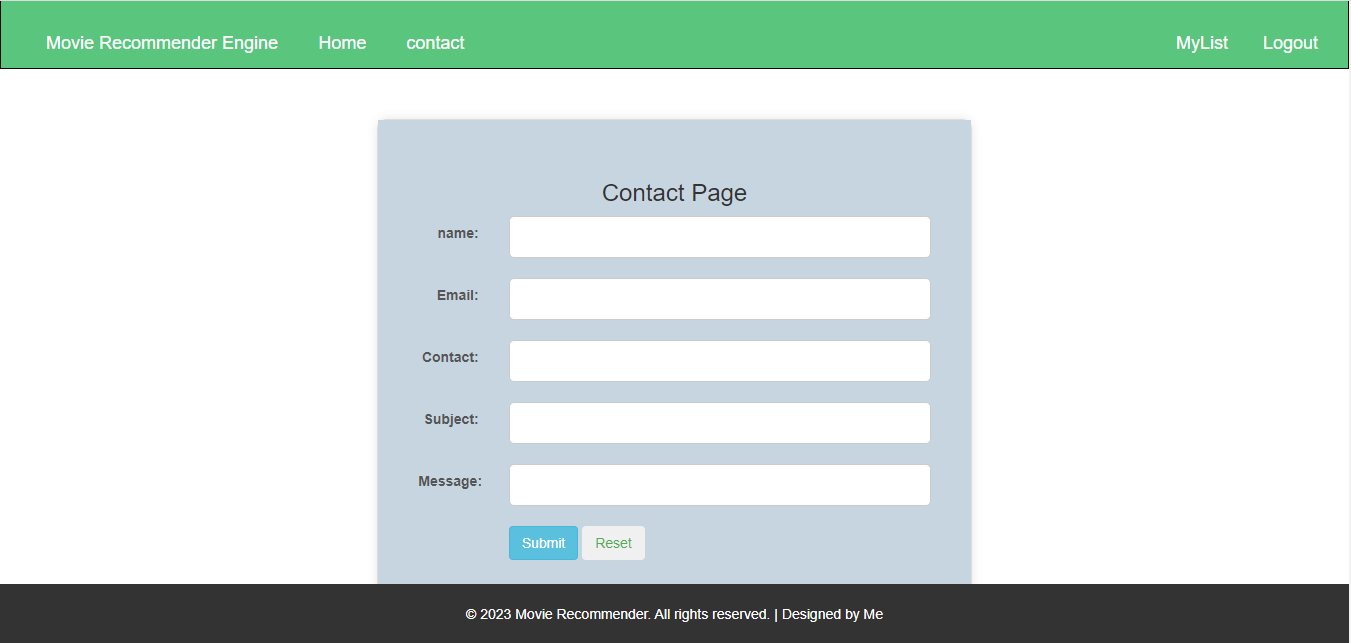


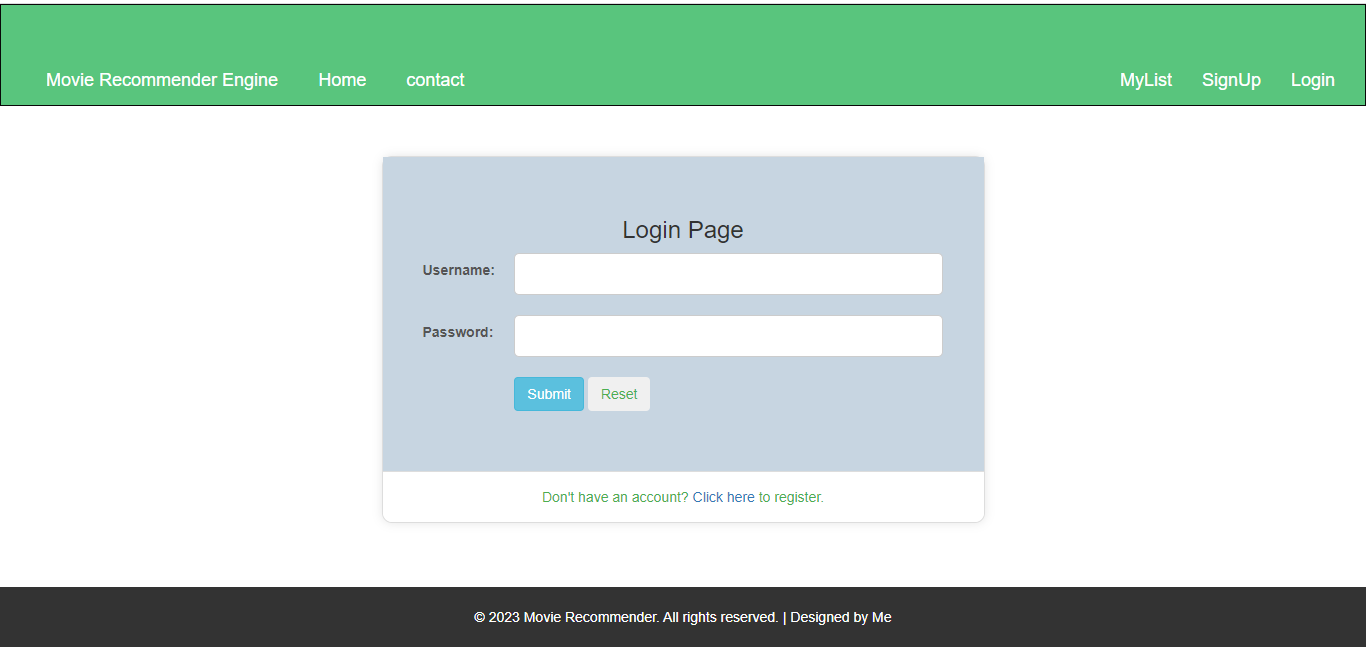


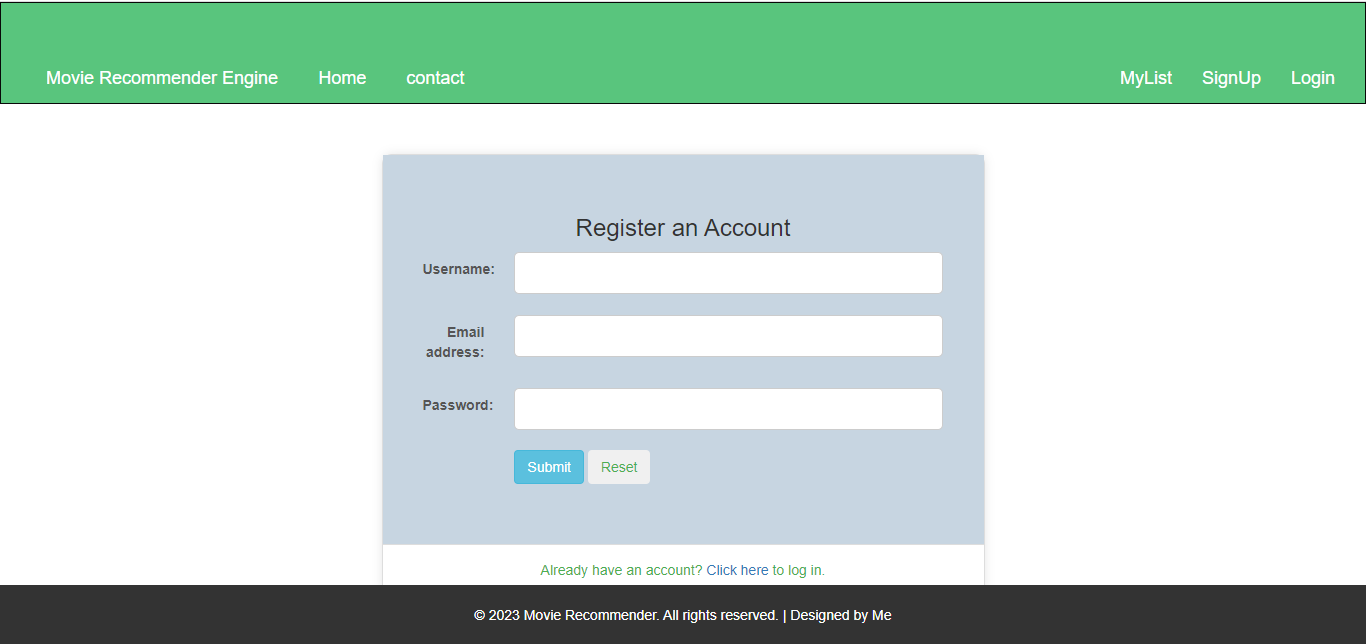












**Conclusion**

In conclusion, the Movie Recommendation System, hosted on AWS EC2, seamlessly integrates Python Django, HTML, CSS, Bootstrap, and JavaScript. Leveraging AWS services like API Gateway, Lambda, DynamoDB, and CloudWatch ensures scalability, efficiency, and secure data management. The project's success lies in its responsive user interface, meticulous IAM role implementation for security, and strategic use of serverless functions. Addressing challenges with agility, the system opens avenues for future enhancements, such as machine learning integration. This cohesive blend of cloud computing, web development, and data management culminates in a sophisticated movie recommendation platform with vast potential for ongoing innovation.