

**RMIT University Vietnam**

School of Science, Engineering and Technology (SSET)

---

# Assignment 2

## Build a scalable app on cloud

---

Course: COSC2638 – Cloud Computing

Group: Group 2 - Exsper

Members:	Tran Phuong Anh	-	s3914138
	Nguyen Minh Anh	-	s3927589
	Bui Minh Nhat	-	s3878174
	Thieu Tran Tri Thuc	-	s3870730
	Lai Nghiep Tri	-	s3799602

Lecturers: Thanh Ngoc Nguyen

Submission Date: September 15<sup>th</sup> 2022

## Contents

<b>I. Contribution .....</b>	6
<b>II. Links.....</b>	6
<b>III. Summary .....</b>	7
<b>IV. Introduction.....</b>	7
<b>V. Related Work .....</b>	8
<b>VI. Software Design and Architecture .....</b>	11
<b>VII. Implementation – Developer Manual.....</b>	14
Create Cloud storage S3.....	14
Create data tables on DynamoDB .....	14
Create customized Lambda functions and connect to DynamoDB via API Gateway .....	18
Set Up AWS Personalize .....	21
Set up AWS Amplify.....	30
Set up Cognito on Amplify.....	33
Set up Amplify Analytics and Kinesis .....	37
<b>VIII. A small user manual.....</b>	46
1. Homepage .....	46
2. Log-in page.....	49
3. Sign-up page .....	50
4. Movie detail page.....	51
5. Search movie.....	53
6. User Dashboard.....	54
<b>IX. References .....</b>	55
<b>X. Video .....</b>	55
<b>XI. Appendices .....</b>	55
1. Sample query user result on Thunder Client.....	55
2. Sample query movie result on Postman .....	57
3. Sample query users result on Postman .....	63

Figure 1: Work Breakdown Structure (WBS) .....	6
Figure 2: IMDb's Homepage .....	8
Figure 3: IMDb's film details page .....	8
Figure 4: IMDB's sign-in form .....	9
Figure 5: IMDb's sign-up form.....	9
Figure 6: IMDb's User Ratings history and Recent Views .....	10
Figure 7: Overview of the Exsper-IMDB components .....	11
Figure 8: The architecture to construct API Layer consuming AWS Personalize recommendations and produce real time events .....	12
Figure 9: Data files uploaded into S3 bucket .....	14
Figure 10: One of the sample buckets in our project .....	15
Figure 11: Set location path of data file in s3.....	15
Figure 12: Set new table details with appropriate WCU and RCU (1 unit for both).....	16
Figure 13: Review all the settings before importing the data file to DynamoDB .....	17
Figure 14: Create a new Lambda function.....	18
Figure 15: A newly created lambda function "demo" .....	19
Figure 16: Trigger configurations for lambda function .....	20
Figure 17: A new API endpoint is added to invoke to lambda function .....	21
Figure 18: CSV data files uploaded into S3 .....	21
Figure 19: Policy allows AWS Personalize fully access a specific S3 resources .....	22
Figure 20: A new role (including built-in and customized) for AWS Personalize.....	22
Figure 21: Create a new dataset .....	23
Figure 22: Configurate details for dataset group.....	23
Figure 23: Create an interaction dataset in order to create recommendations .....	24
Figure 24: Create a dataset schema based on an existing schema .....	25
Figure 25: Import CSV data from S3 into the new dataset group along with create a new IAM role based on the customized role so far.....	26
Figure 26: Create a solution call "my-reranking" attached by "aws-personalized-ranking" recipe.....	27
Figure 27: Create two more solutions "aws-similar-items" and "aws-user-personalize" .....	27
Figure 28: Create a campaign .....	28
Figure 29: Create event tracker .....	29
Figure 30: Create Lambda function associated to Personalize and a API Gateway as a client API .....	30
Figure 31: Create an empty app at first .....	30
Figure 32: Use source code from GitHub repo .....	31
Figure 33: Locate to a repo branch.....	31
Figure 34: Deployment successfully with new resources: CloudFront, Lambda Edge, and S3 bucket.....	32
Figure 35: Connect Amplify to Cognito and Kinesis Data Stream.....	32
Figure 36: Install AWS Amplify CLI with npm .....	33
Figure 37: Pull the Amplify back-end to local.....	33
Figure 38: Configurate AWS Cognito using Amplify CLI.....	34
Figure 39: Create a new user pool .....	35
Figure 40: Configurate User Pools .....	35
Figure 41: Create AWS Cognito domain.....	36

Figure 42: Create lambda function to verify user email on sign-up.....	36
Figure 43: Attach Cognito to Amplify .....	37
Figure 44: Create and Configurate data stream .....	37
Figure 45: Configurate AWS Kinesis Data Stream using Amplify CLI .....	38
Figure 46: Create S3 bucket for data storage.....	38
Figure 47: Configure Kinesis Firehose .....	39
Figure 48: Select the source stream as the Kinesis.....	39
Figure 49: Create lambda function to handle events .....	40
Figure 50: Code deployment on Lambda function .....	41
Figure 51: Create environment variables.....	41
Figure 52: Create policy permissions for Lambda to access S3, Kinesis and Firehose.....	42
Figure 53: Add Kinesis as a trigger for Lambda function .....	43
Figure 54: Collect click stream events using Amplify SDK.....	44
Figure 55: Observe click stream event on AWS Cloud Watch.....	45
Figure 56: Exsper's Homepage .....	46
Figure 57: List of top-rated films .....	47
Figure 58: List of highest views movies .....	48
Figure 59: List of suggested movies .....	49
Figure 60: Login page after pressing Login button at the homepage.....	49
Figure 61: Signup page after pressing Register button from the homepage or login form... <td>50</td>	50
Figure 62: Movie details.....	52
Figure 63: A returned list of film that include the searching keyword.....	53
Figure 64: Dashboard contains list of recent viewed and recent rated movies .....	54
Figure 65: Top-picks for a certain user .....	56
Figure 66: List of same genre films for an anonymous user .....	56
Figure 67: List of same genre films for an authenticated user .....	57
Figure 68: 10 most view films returned in JSON format.....	58
Figure 69: 10 highest rated films returned in JSON format.....	59
Figure 70: A list of movie contains search keyword returned in JSON format .....	60
Figure 71: Movie details returned after sending a request.....	61
Figure 72: Increment number of view after sending a request.....	61
Figure 73: Current movie rate number before rating.....	62
Figure 74: A request send with the body to rate the movie.....	62
Figure 75: Current movie rate number after rating.....	62
Figure 76: User details returned in JSON format .....	63
Figure 77: Response indicate the rating timestamp of a certain user on a certain movie ....	64
Table 1: Contribution among members (%) .....	6

Table 1: Contribution among members (%) .....

**ABBREVIATION MEANING**

<b>AWS</b>	Amazon Web Service
<b>BAAS</b>	Back-end as a Service
<b>ENV</b>	Environment

## I. Contribution

Table 1: Contribution among members (%)

Members	Percentage (%)
Tran Phuong Anh	20
Nguyen Minh Anh	20
Bui Minh Nhat	20
Thieu Tran Tri Thuc	20
Lai Nghiep Tri	20

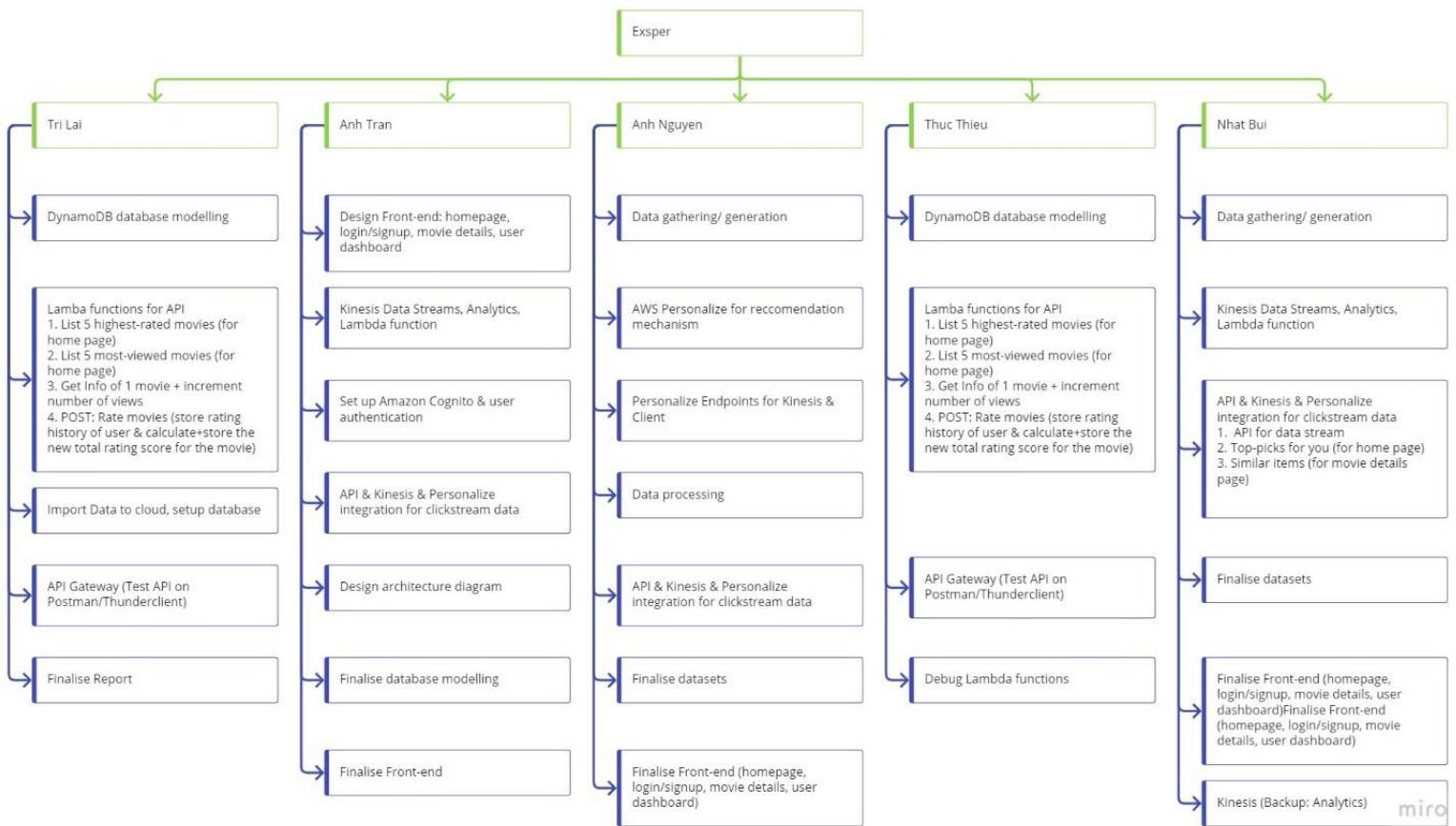


Figure 1: Work Breakdown Structure (WBS)

## II. Links

### Github Repository:

[baubau09/exsper-imdb \(github.com\)](https://github.com/baubau09/exsper-imdb)

**Demo Application Link:**

<https://www.exsper.rmitvn.io>

**Demo user login information**

User 1

Username: [s3914138@rmit.edu.vn](mailto:s3914138@rmit.edu.vn)

Password: 12345678

User 2

Username: [s3927589@rmit.edu.vn](mailto:s3927589@rmit.edu.vn)

Password: 12345678

### III. Summary

In this project, the team utilizes the fundamentals of cloud computing to develop a web application, which is deployed on Amazon Web Services. By integrating multiple cloud services such as AWS Amplify, Amazon Cognito, Amazon S3, Amazon API Gateway, AWS Lambda, Amazon DynamoDB, Amazon Kinesis Data Streams & Firehose, and Amazon Personalize, we could quickly ship a complex system where users' data could be collected and reproduced for analytics and improve the business logic.

### IV. Introduction

The team decided to build a more basic version that nevertheless does the essential tasks after being inspired by the IMDB web application, whose purpose is to offer a variety of lists of movies along with other details (IMDB, 2022). Since users must actively search for each movie, it is inefficient to pick a film that satisfies both personal preferences and cinematic trends. The problem can escalate if users' viewing habits are visualized without their data. The purpose of this program is to allow users to explore, rate certain movies, and suggest additional films in the same genre as a result of the system's ability to track their behaviour. The app's benefit

over other movie review websites is its unique and beautiful interface design, which allows users to explore movies and ratings in a seamless manner.

## V. Related Work

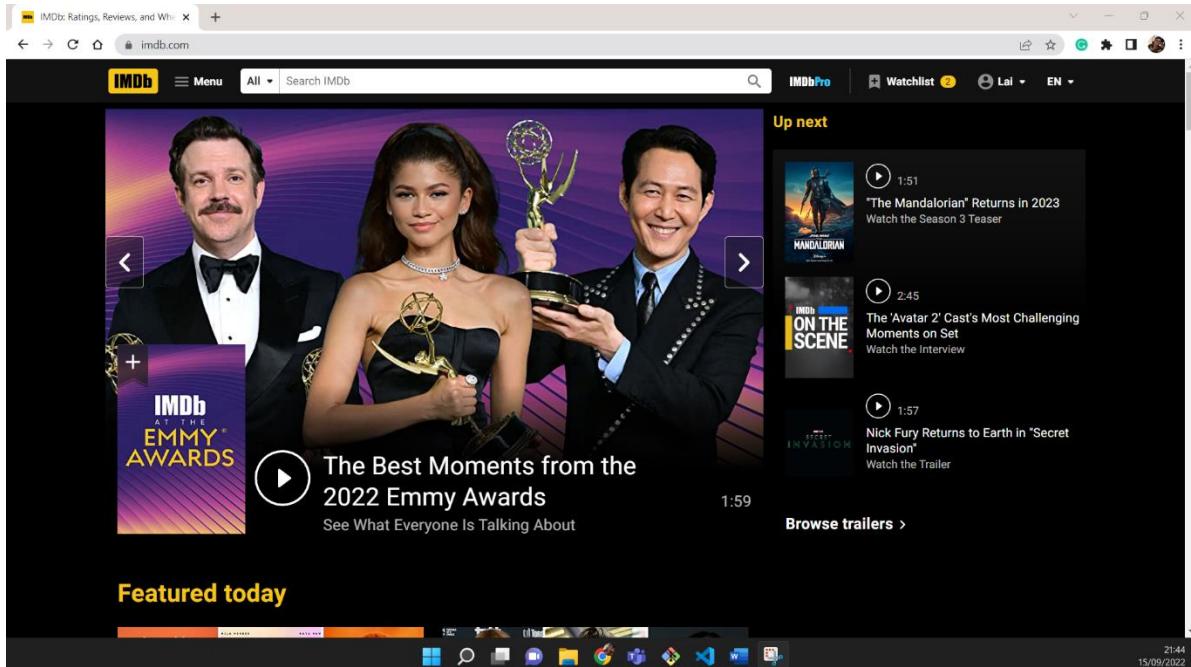


Figure 2: IMDb's Homepage

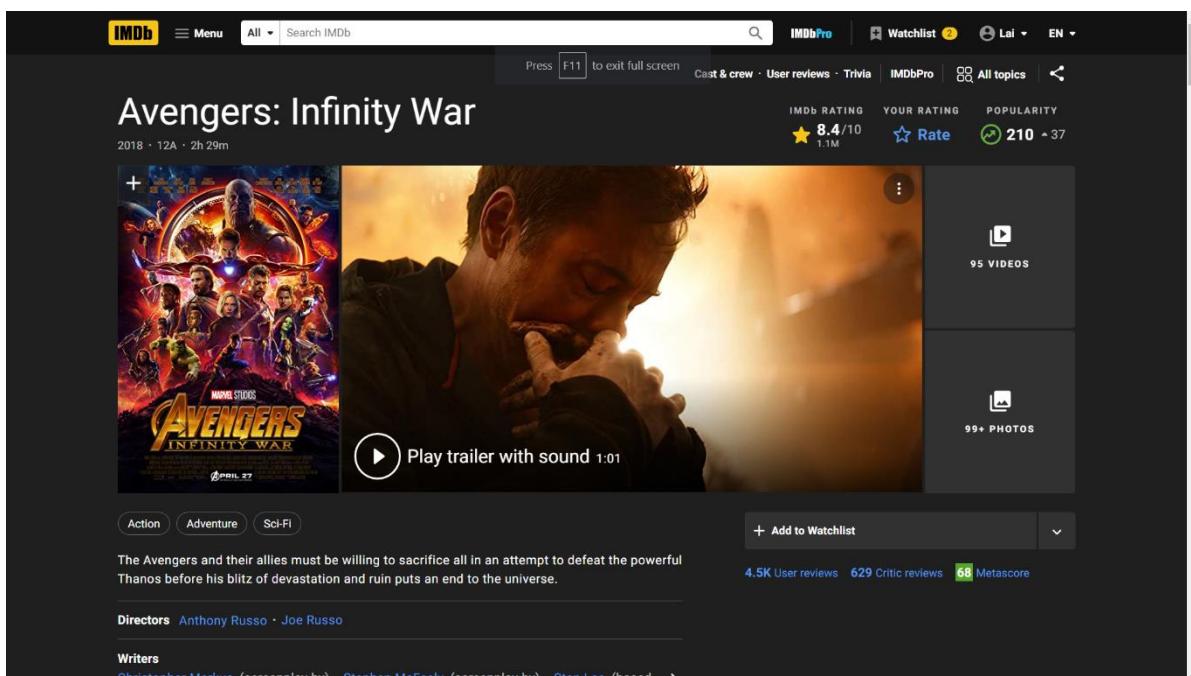
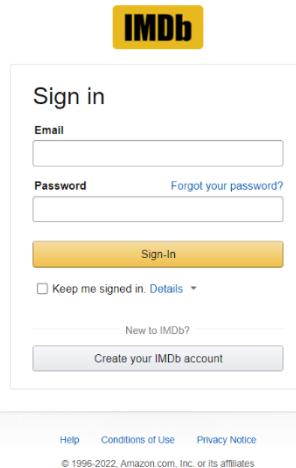


Figure 3: IMDb's film details page



The image shows the IMDb sign-in form. It features a yellow header with the IMDb logo. Below it is a white rectangular input field with a thin gray border. The field has a placeholder "Email" at the top and a standard text input box below. To the right of the input box is a blue link "Forgot your password?". Below the input field is a yellow rectangular button with the text "Sign-In" in white. Underneath the button is a small checkbox labeled "Keep me signed in." followed by a "Details" link. At the bottom of the form is a horizontal line with the text "New to IMDb?" on the left and a "Create your IMDb account" button on the right.

Help   Conditions of Use   Privacy Notice  
© 1996-2022, Amazon.com, Inc. or its affiliates

Figure 4: IMDb's sign-in form



The image shows the IMDb sign-up form. It features a yellow header with the IMDb logo. Below it is a white rectangular input field with a thin gray border. The field has a placeholder "Your name" at the top and a standard text input box below. To the right of the input box is a blue link "First and last name". Below the input field is another yellow rectangular button with the text "Create your IMDb account" in white. Underneath the button is a horizontal line with the text "Already have an account? [Sign in >](#)" on the right.

Help   Conditions of Use   Privacy Notice  
© 1996-2022, Amazon.com, Inc. or its affiliates

Figure 5: IMDb's sign-up form

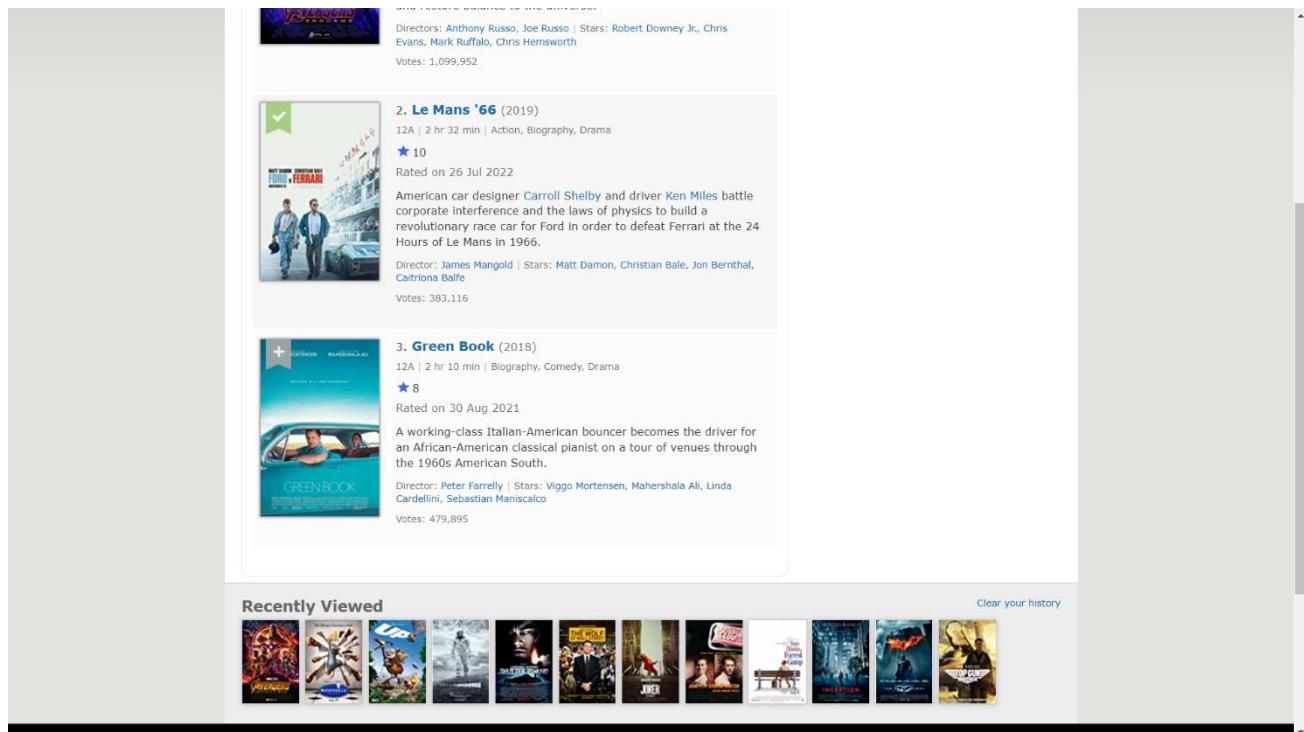


Figure 6: IMDb's User Ratings history and Recent Views

## VI. Software Design and Architecture

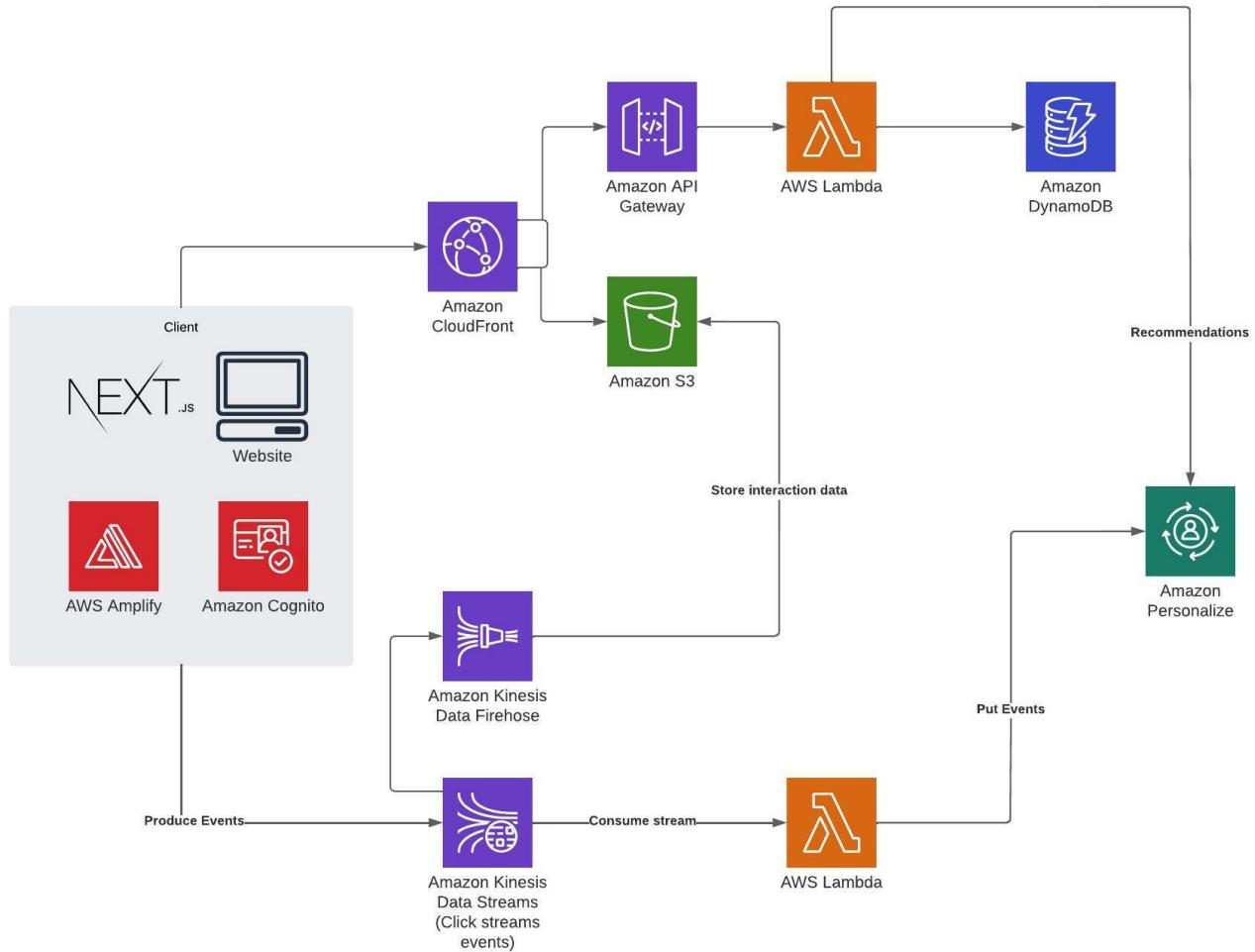


Figure 7: Overview of the Exsper-IMDB components

The app is designed using NEXTJS on the client side, which is based on React and supports hybrid static and server-side rendering, allowing the webpage to load more faster with fully rendered content at the server and the full-stack application will be hosted by AWS Amplify, which serves as a back-end as a service (BaaS). Additionally, we utilize AWS Cognito to build sign-in forms, then authenticate users through an external identity provider or credentials to access the app's resources. Then, using AWS API Gateway to make API calls in order to connect to the back-end side.

The datasets gathered from GroupLens [1] are slightly adjusted on the database side to fit the project context before being transferred to the AWS S3 bucket, which

follows a NoSQL model, will be later imported into AWS DynamoDB. Connecting the database through AWS Lambda, a serverless computing service, in order to conduct CRUD operations that are accessed through AWS API Gateway. We have integrated AWS Kinesis Data Streams to deal with the mechanism of movie suggestions or mouse click handlers. This allows us to record and store data streams (events), then feed them to AWS Kinesis Data Firehose to process them in real-time, which later store those processed data to S3. At the final stage, the team is using Amazon Personalize as machine learning to model the recommendation algorithm.

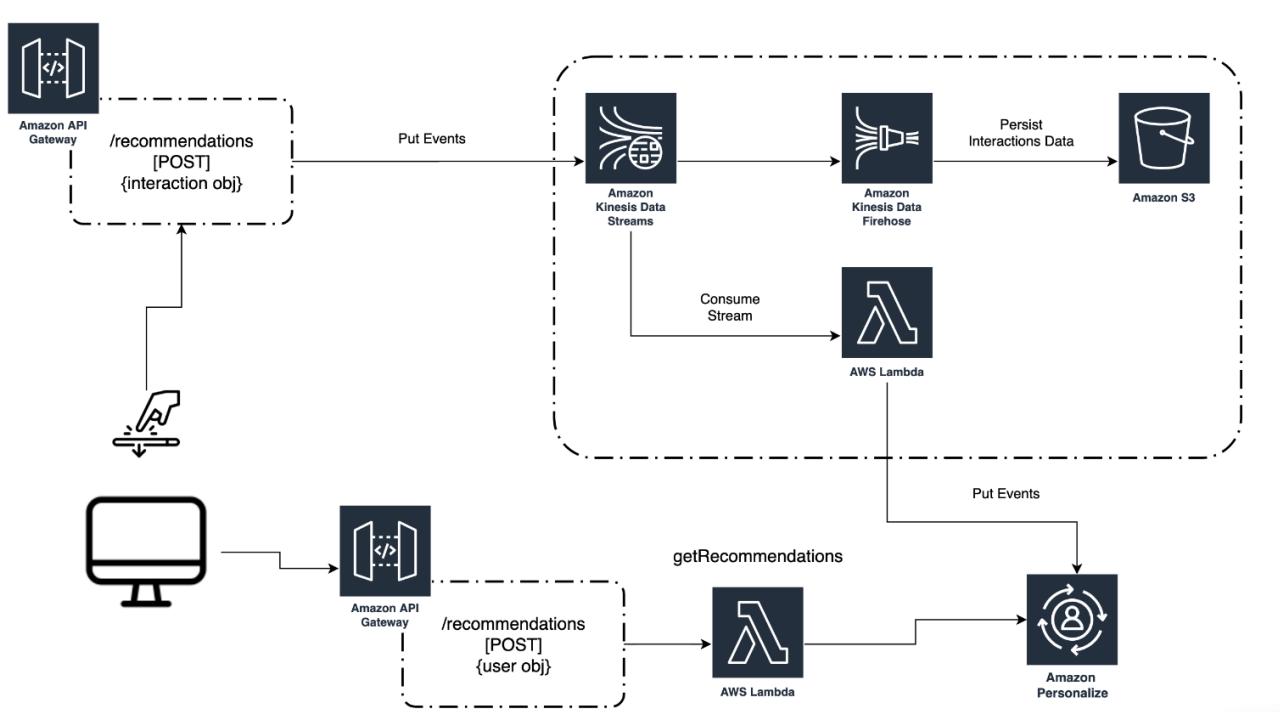


Figure 8: The architecture to construct API Layer consuming AWS Personalize recommendations and produce real time events [2]

AWS Personalize is a machine learning service which provides real-time recommendations. With the help of modern AI technology, AWS Personalize outperforms the traditional rule-based recommendation systems. In our movie system, this service is used to provide top-picked movies for a specific user and find similar movies for the target one. The frontend can get recommendations by making a request to the API Gateway which connects to an AWS Lambda function to trigger the corresponding Personalize Campaign. When users click on or watch a particular movie, the event will be sent to another API which then forwards the data to AWS Kinesis. Kinesis will record all the events until there are enough records for a batch (which is set by the developer). After that, Kinesis would trigger an AWS Lambda function which will send the events to the AWS Personalize's event tracker. After 7 days of collecting new data, Personalize will retrain the AI model to provide up to date recommendations.

## VII. Implementation – Developer Manual

*\*Note: In this project, we assume that readers have basic understandings on AWS in order to go through this tutorial sheet. All these below Amazon services are mostly based on the tutorial note so it shares the similarities in configurations and developments.*

### Create Cloud storage S3

#### Step 1: Create buckets

User has to create a bucket where data files or folders will be added later and there should be hierarchical structure of folders which highly depends on the team preferences.

#### Step 2: Upload csv data files to the buckets

The screenshot shows the AWS S3 console interface. On the left, the navigation pane includes 'Amazon S3', 'Buckets', 'Access Points', 'Object Lambda Access Points', 'Multi-Region Access Points', 'Batch Operations', 'Access analyzer for S3', 'Block Public Access settings for this account', 'Storage Lens', 'Dashboards', 'AWS Organizations settings', 'Feature spotlight', and 'AWS Marketplace for S3'. The main content area is titled 's3dbmit' and shows a list of objects. The 'Objects (6)' section lists the following files:

Name	Type	Last modified	Size	Storage class
2022/	Folder	-	-	-
links.csv	csv	August 26, 2022, 14:24:22 (UTC+07:00)	64.9 KB	Standard
movies.csv	csv	August 26, 2022, 15:06:49 (UTC+07:00)	139.8 KB	Standard
ratings.csv	csv	August 26, 2022, 14:24:05 (UTC+07:00)	20.8 MB	Standard
users_v2.csv	csv	September 10, 2022, 22:02:41 (UTC+07:00)	5.7 MB	Standard
users.csv	csv	August 26, 2022, 14:24:48 (UTC+07:00)	107.7 KB	Standard

Figure 9: Data files uploaded into S3 bucket

### Create data tables on DynamoDB

#### Step 1: Import existing sample data files from s3 buckets

Instead of creating a raw table without any data or items, we only need to import these tables, which were uploaded so far in S3 bucket, contains a list of sample data. User can refer to the **Imported from S3** keyword in the left-side menu [3],

The screenshot shows the AWS DynamoDB console with the 'users\_v2' table selected. The 'Additional settings' tab is active. Under 'Table capacity', 'Provisioned' mode is chosen. Read and write capacity are both set to 1 unit. Target utilization is set to 70% for both reads and writes. The 'Auto scaling activities' section shows no auto-scaling activities found.

Figure 10: One of the sample buckets in our project

## Step 2: Setting the import configurations [3]

The screenshot shows the 'Import options' step of the AWS Import from S3 wizard. The 'Source S3 URL' field contains 's3://1320bnml/users\_v2.csv'. The 'Destination table' is set to 'users\_v2'. The 'Import file format' is set to 'CSV'. Other options like 'Import file compression' and 'Import file header' are also visible.

Figure 11: Set location path of data file in s3

**Destination 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 5 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 settings**

Default settings  
The fastest way to create your table. You can modify these settings now or after your table has been created.

Customize settings  
Use these advanced features to make DynamoDB work better for your needs.

**Capacity calculator**

**Read/write capacity settings** Info

**Capacity mode**  
 On-demand  
Simplify billing by paying for the actual reads and writes your application performs.

Provisioned  
Manage and optimize your costs by allocating read/write capacity in advance.

**Read capacity**  
Provisioned capacity units

**Write capacity**  
Provisioned capacity units

ⓘ Auto scaling settings can't be changed during S3 import. You can configure auto scaling in table settings after the import has been completed.

**Secondary indexes** Info  
Delete Create global index

Name	Type	Partition key	Sort key	Projected attributes
No indexes Use secondary indexes to perform queries on attributes that are not part of your table's primary key.				
<small>Create global index</small>				

ⓘ Local secondary indexes cannot be created during an import from S3.

**Estimated read/write capacity cost**

Here is the estimated total cost of provisioned read and write capacity for your table and indexes, based on your current settings. This does not include the cost of the import. To learn more, see Amazon DynamoDB pricing Info for provisioned capacity.

Total read capacity units	Total write capacity units	Region	Estimated cost
1	1	ap-southeast-1	\$0.67 / month

**Encryption at rest** Info  
All user data stored in Amazon DynamoDB is fully encrypted at rest. By default, Amazon DynamoDB manages the encryption key, and you are not charged any fee for using it.

**Encryption key management**

Owned by Amazon DynamoDB Learn more Info  
The AWS KMS key is owned and managed by DynamoDB. You are not charged an additional fee for using this key.

AWS managed key Learn more Info  
Key alias: aws/dynamodb. The key is stored in your account and is managed by AWS Key Management Service (AWS KMS). AWS KMS charges apply.

Stored in your account, and owned and managed by you Learn more Info  
The key is stored in your account and is owned and managed by you. AWS KMS charges apply.

Cancel Previous Next

Figure 12: Set new table details with appropriate WCU and RCU (1 unit for both)

The screenshot shows the AWS DynamoDB Import from S3 wizard in progress. The left sidebar lists three steps: Step 1 Import options, Step 2 Destination table, and Step 3 Review and import. The current step is Step 3, titled "Review and import".

**Step 1: Import options**

Source S3 URL s3://s3dbrmit/users_v2.csv	S3 bucket owner This AWS account
Import file format CSV	CSV header Use the first line of the source file
CSV delimiter character Comma (",")	Import compression type No compression

**Step 2: Destination table**

Table name users_v2	Partition key USER_ID
Table class DynamoDB Standard	Sort key NAME

**Read/write capacity settings**

Capacity mode Provisioned	Read capacity 1 RCU
Auto scaling Off	Write capacity 1 WCU

**Secondary indexes**

Secondary indexes  
No secondary indexes have been created.

**Encryption at rest**

Encryption key  
Owned by Amazon DynamoDB

Buttons at the bottom: Cancel, Previous, Import (highlighted in orange).

Figure 13: Review all the settings before importing the data file to DynamoDB

# Create customized Lambda functions and connect to DynamoDB via API Gateway

## Step 1: Create lambda function

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The 'Author from scratch' option is selected. The 'Function name' field contains 'demo'. The 'Runtime' dropdown is set to 'Node.js 16.x'. Under 'Architecture', 'x86\_64' is selected. In the 'Permissions' section, the 'Execution role' dropdown is set to 'ApiGatewayInvokeLambdaRole'. At the bottom right, there are 'Cancel' and 'Create function' buttons.

Figure 14: Create a new Lambda function

After creating the lambda function, we have to add a trigger is API Gateway in this stage since we want to the lambda function will be invoked whenever the corresponding API was called [4].

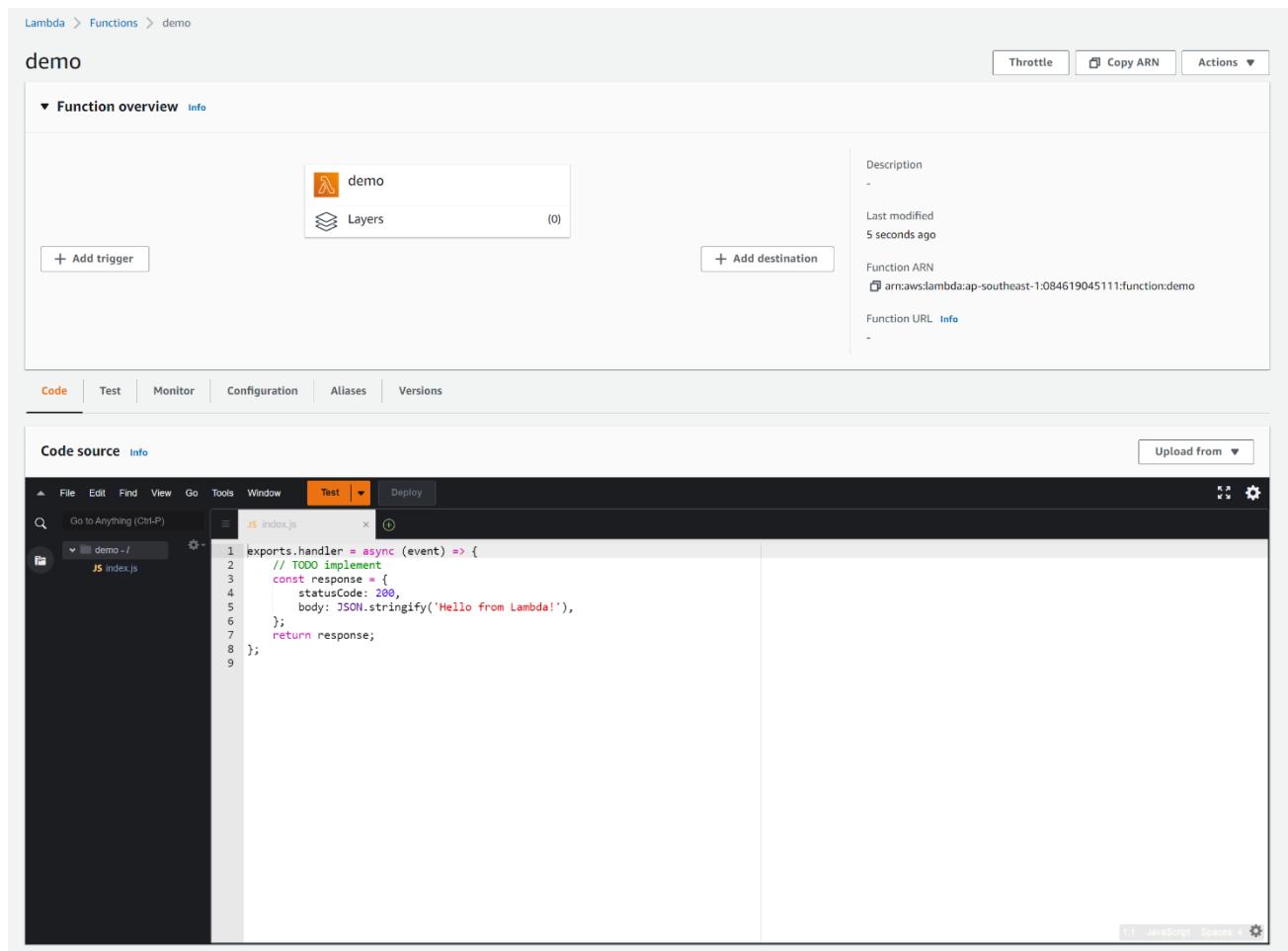


Figure 15: A newly created lambda function "demo"

## Step 2: Configure API Gateway

Searching for the AWS API Gateway service to create a new REST API with the IAM security [5].

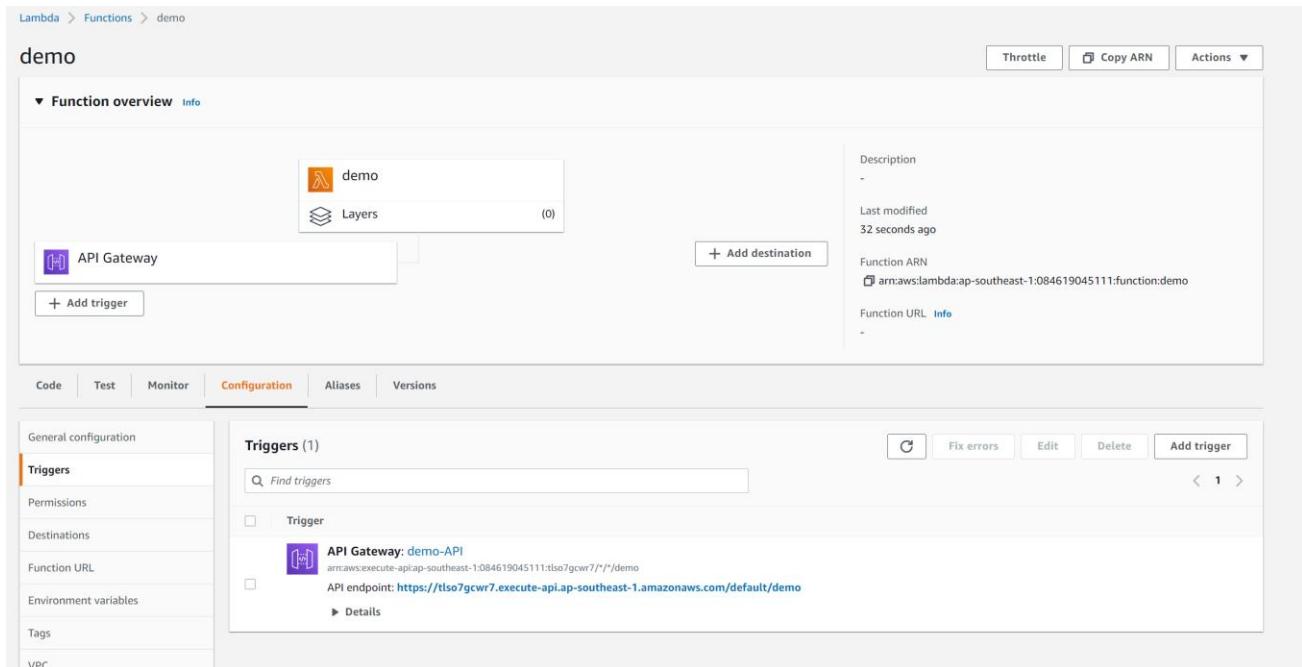
The screenshot shows the 'Add trigger' configuration page for a Lambda function. At the top, there's a breadcrumb navigation: 'Lambda > Add trigger'. Below it, the title 'Add trigger' is displayed. A 'Trigger configuration' section is open, showing the 'API Gateway' option selected from a dropdown menu. The dropdown also includes 'api', 'application-services', 'aws', and 'serverless' options. The main configuration area has the following sections:

- Intent:** A note stating "Add an API to your Lambda function to create an HTTP endpoint that invokes your function. API Gateway supports two types of RESTful APIs: HTTP APIs and REST APIs. [Learn more](#)".
- Create a new API:** This option is selected, indicated by a blue radio button.
- Use existing API:** An unselected option indicated by a grey radio button.
- API type:** Two options are shown:
  - HTTP API:** A note says "Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support." This option is unselected.
  - REST API:** A note says "Develop a REST API where you gain complete control over the request and response along with API management capabilities." This option is selected, indicated by a blue radio button.
- Security:** A note says "Configure the security mechanism for your API endpoint." A dropdown menu is open, showing 'IAM' as the selected option.
- Additional settings:** A note says "Lambda will add the necessary permissions for Amazon API Gateway to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model."

At the bottom right of the configuration panel, there are 'Cancel' and 'Add' buttons. The 'Add' button is highlighted with an orange background.

Figure 16: Trigger configurations for lambda function

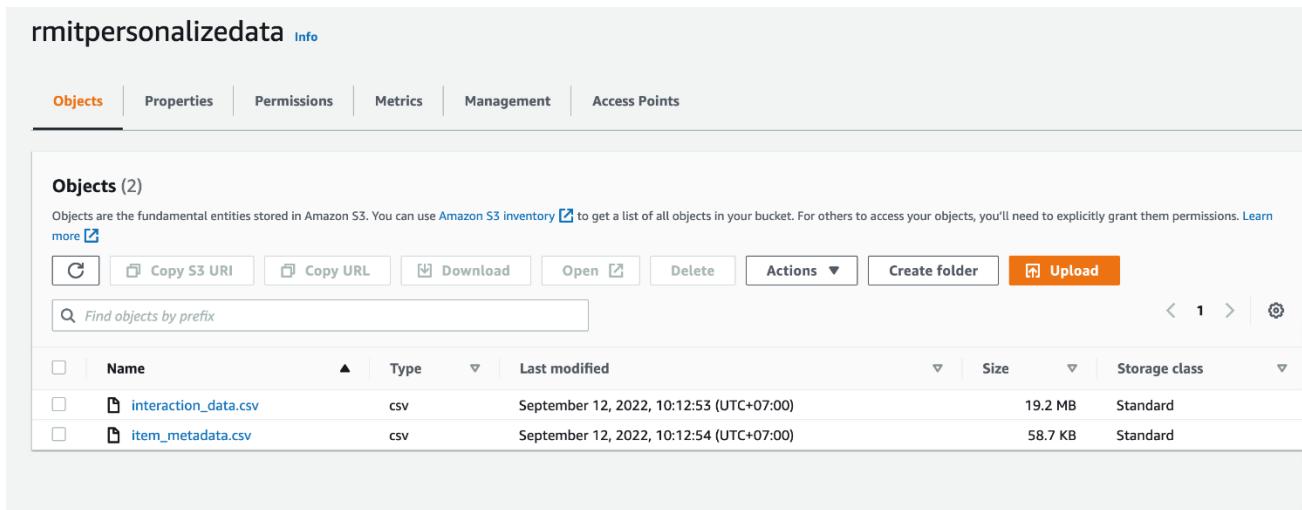
After adding the API Gateway to the Lambda function, the system automatically initiates a new endpoint in the Configuration tab in order to trigger the corresponding function whenever the client has a call to that such API [5].



*Figure 17: A new API endpoint is added to invoke to lambda function*

## Set Up AWS Personalize

## Step 1: Upload Data to S3



*Figure 18: CSV data files uploaded into S3*

## Step 2: Update S3 policy

Update S3 policy so that AWS Personalize can access the data

The screenshot shows the 'Bucket policy' section of an AWS S3 bucket. A message box at the top states: 'Public access is blocked because Block Public Access settings are turned on for this bucket'. Below this, the JSON policy document is displayed:

```
{
  "Version": "2012-10-17",
  "Id": "PersonalizeS3BucketAccessPolicy",
  "Statement": [
    {
      "Sid": "PersonalizeS3BucketAccessPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "personalize.amazonaws.com"
      },
      "Action": [
        "s3:Object",
        "s3>ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::rmitpersonalizedata",
        "arn:aws:s3:::rmitpersonalizedata/*"
      ]
    }
  ]
}
```

Buttons for 'Edit' and 'Delete' are visible in the top right corner.

Figure 19: Policy allows AWS Personalize fully access a specific S3 resources

### Step 3: Create a role for Personalize

Create a role to grant all necessary permissions for Personalize.

The screenshot shows the 'PersonalizeRole' page in the AWS IAM service. The 'Summary' tab is selected. Key details shown include:

- Creation date:** September 12, 2022, 10:48 (UTC+07:00)
- Last activity:** 2 days ago
- ARN:** arn:aws:iam::557044894694:role/PersonalizeRole
- Maximum session duration:** 1 hour

The 'Permissions' tab is active, showing two attached policies:

- PersonalizeFullAccessPolicy** (Customer managed)
- S3ForPersonalize** (Customer managed)

Figure 20: A new role (including built-in and customized) for AWS Personalize

### Step 4: Create a Personalize dataset group

Dataset groups (1)			
Dataset groups are domain-specific containers for your datasets. For example, you can create one project for an internal application, and another for an external application.			
	<a href="#">View details</a>	<a href="#">Delete</a>	<a href="#">Create dataset group</a>
<input type="text"/> Find dataset group			
Name	Domain	Status	Created
<input type="radio"/> exper-reccsys-group	Custom	<input checked="" type="checkbox"/> Active	Mon, 12 Sep 2022 04:10:40 GMT

Figure 21: Create a new dataset

## Step 5: Configure the dataset group

Amazon Personalize > Create dataset group

Step 1  
Create dataset group

Step 2  
Create interactions dataset

Step 3  
Import interactions data

### Create dataset group Info

**Dataset group details**

**Name**  
The name you enter here distinguishes this dataset group from others.

The dataset group name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and \_ - (hyphen).

**Domain**  
Choose a domain for your use cases.

**E-commerce**  
Grow your business by recommending the right products at the right time.

**Video on demand**  
Increase engagement by recommending relevant content to your users.

**Custom**  
Create and manage custom resources for your use cases.

**Tags - optional (0) Info**  
A tag is an administrative label that you assign to AWS resources to make it easier to manage them. Each tag consists of a key and an optional value. Use tags to search and filter your resources or track your AWS costs.

[Cancel](#) [Create dataset group and continue](#)

Figure 22: Configure details for dataset group

## Step 6: Create the interaction dataset

The screenshot shows the 'Create interactions dataset' step in the Amazon Personalize interface. On the left, a vertical navigation bar lists three steps: 'Step 1 Create dataset group', 'Step 2 Create interactions dataset' (which is currently selected), and 'Step 3 Import interactions data'. The main content area is titled 'Dataset details' and contains fields for 'Dataset name' (set to 'my-dataset') and 'Dataset schema' (set to 'exper-reccsys-group-event-schema'). A note indicates that the dataset name must be between 1-63 characters and can include a-z, A-Z, 0-9, and \_ - (hyphen). The 'Use an existing schema' option is selected.

Figure 23: Create an interaction dataset in order to create recommendations

## Step 7: Setup the dataset schema

## Schema definition

Ensure your dataset's schema matches the following schema.

```
1  {
2      "type": "record",
3      "name": "Interactions",
4      "namespace": "com.amazonaws.personalize.schema",
5      "fields": [
6          {
7              "name": "USER_ID",
8              "type": "string"
9          },
10         {
11             "name": "ITEM_ID",
12             "type": "string"
13         },
14         {
15             "name": "TIMESTAMP",
16             "type": "long"
17         },
18         {
19             "name": "EVENT_TYPE",
20             "type": "string"
21         },
22         {
23             "name": "SESSION_ID"
24         }
25     ]
26 }
```

### Tags - optional (0) Info

A tag is an administrative label that you assign to AWS resources to make it easier to manage them. Each tag consists of a key and an optional value. Use tags to search and filter your resources or track your AWS costs.

[Cancel](#) [Previous](#) [Create dataset and continue](#)

Figure 24: Create a dataset schema based on an existing schema

### Step 8: Import file from S3 and attach the above-created role

### Data import source

**Additional S3 bucket policy required**

In addition to the IAM service role defined above, Amazon Personalize also requires you to add a bucket policy to the S3 bucket containing your data files so that it can process them. Follow the instructions [described here](#) to add the required bucket policy to your S3 bucket.

### Data location Info

Choose the S3 location of your data.

Your file needs to be in a CSV format and reflect the schema.

### IAM Role

#### IAM service role

Amazon Personalize requires permissions to access your S3 bucket. Choose an existing role with access or create a role in the IAM console with the [AmazonPersonalizeFullAccess](#) IAM policy attached.

#### Custom IAM role ARN

Figure 25: Import CSV data from S3 into the new dataset group along with create a new IAM role based on the customized role so far

## Step 9: Repeat step 6 to 8 to create another dataset for movie items

## Step 10: Create and train solutions

The solution will retrieve data from S3 and start training a machine learning model for the chosen task. Below is the setup for the user aws-personalize-ranking solution.

## Create solution Info

You create a solution using a recipe that is tailored to a specific use case.

### Solution detail

**Solution name**  
The solution name that you enter here can help you distinguish this solution from others.

The solution name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and \_ - (hyphen).

**Solution type Info**  
Choose the type of solution you want to create. The type determines what recipes are available for solution creation.

**Item recommendation**  
 Create a solution that generates item recommendations.

**User segmentation - new**  
 Create a solution that predicts groups of users based on item input data.

**Recipe Info**  
Recipes are preconfigured algorithms tailored to specific use cases.

**aws-personalized-ranking**  
 Re-ranks an input list of items for a given user. Trains on user-item interactions dataset, item ...

### ▼ Solution configuration - optional Info

**Event type Info**

If you have varying event types in your user-item interactions dataset, you can enter which event this solution should use.

The event type is limited to the number of event types listed in your data.

Figure 26: Create a solution call "my-reranking" attached by "aws-personalized-ranking" recipe

In our system, we also created 2 more solutions: aws-similar-items and aws-user-personalize.

Solutions (3)					<a href="#">View details</a>	<a href="#">Create solution</a>
Analyze your solutions and compare their performance to each other. Solutions are created using recipes.						
<input type="text"/> <b>Find solution</b>					< 1 >	
Name	Status	Recipe used	Created			
my-personalize	<input checked="" type="radio"/> Active	--	Mon, 12 Sep 2022 04:57:12 GMT			
my-sim	<input checked="" type="radio"/> Active	--	Mon, 12 Sep 2022 04:56:32 GMT			
my-reranking	<input checked="" type="radio"/> Active	--	Mon, 12 Sep 2022 04:55:51 GMT			

Figure 27: Create two more solutions "aws-similar-items" and "aws-user-personalize"

## Step 11: Create a campaign for each solution

A campaign will serve a solution so that we can send a request to get recommendations via AWS Client API.

## Create new campaign

### Campaign details

**Campaign name**  
The text you enter here appears in the Campaign dashboard and detail page. It can help you distinguish this campaign from others.  
The campaign name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and \_ - (hyphen).

**Solution**  
The selected solution is used to generate the recommendations provided in your campaign.  
▼

**Solution version ID**  
The selected solution version is used to generate the recommendations provided in your campaign.  
▼  
Mon, 12 Sep 2022 04:56:32 GMT

**Minimum provisioned transactions per second** Info  
The minimum amount of throughput in transactions per second (TPS) that is provisioned for this campaign.  
Enter a number from 1-500.

**Tags - optional (0)** Info  
A tag is an administrative label that you assign to AWS resources to make it easier to manage them. Each tag consists of a key and an optional value. Use tags to search and filter your resources or track your AWS costs.

**Cancel** **Create campaign**

Figure 28: Create a campaign

## Step 12: Create an Event Tracker

The Event Tracker will be responsible for collecting Put Events sent from Kinesis.

The screenshot shows the 'Configure tracker' step in the AWS Personalize console. On the left, a sidebar lists 'Step 1: Configure tracker' and 'Step 2: Install the SDK'. The main area is titled 'Tracker configurations' and contains a 'Tracker name' field with the value 'my-event-tracker'. A note below the field specifies that the name must be between 1 and 63 characters, containing only lowercase letters, uppercase letters, numbers, and underscores. Below this is a section for 'Tags - optional (0)' with a note about using tags for resource management. At the bottom right are 'Cancel' and 'Next' buttons.

Figure 29: Create event tracker

### Step 13: Create an API and Lambda function

Create an API using AWS API Gateway and build a Lambda function to get recommendations from Personalize.

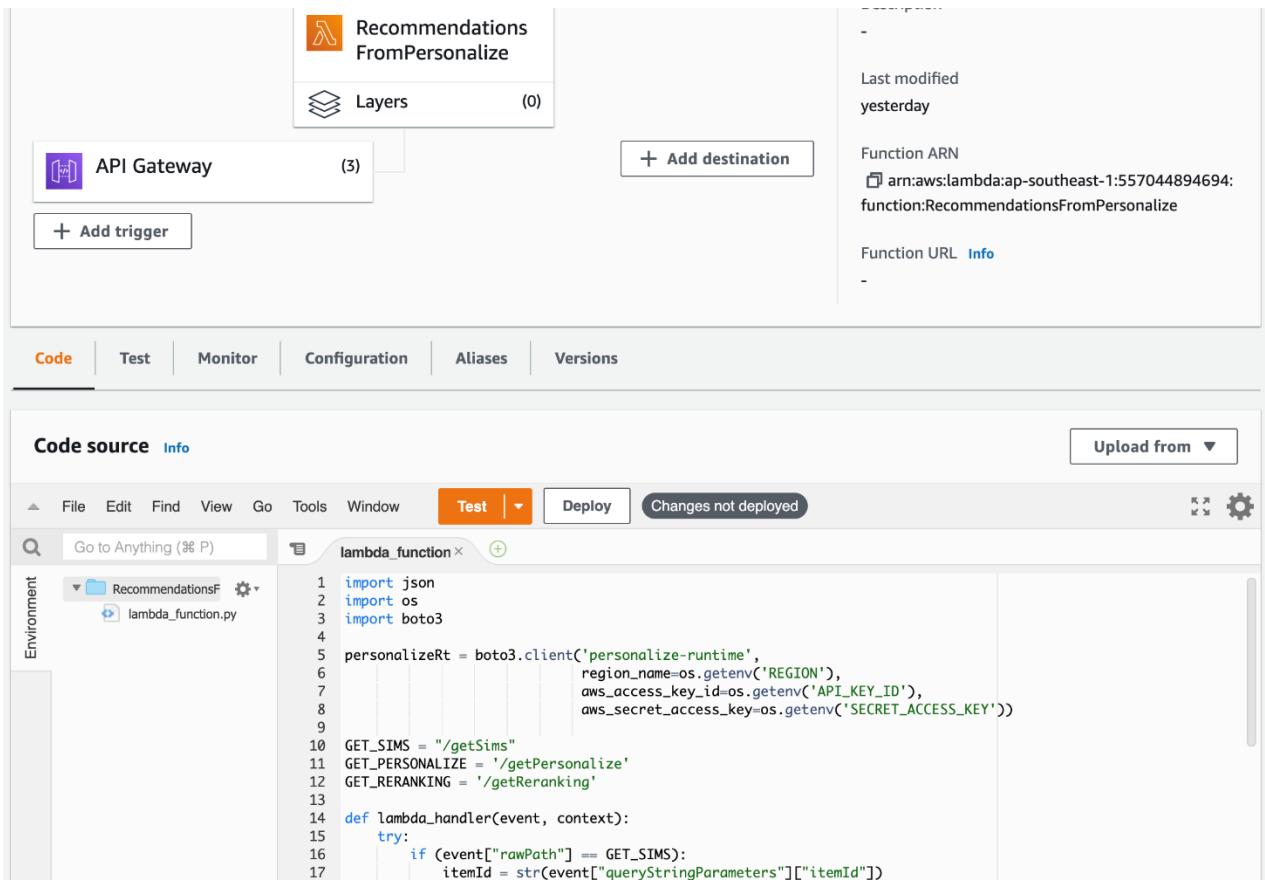


Figure 30: Create Lambda function associated to Personalize and a API Gateway as a client API

## Set up AWS Amplify

### Step 1: Create a new app by connecting with Github in Amplify console

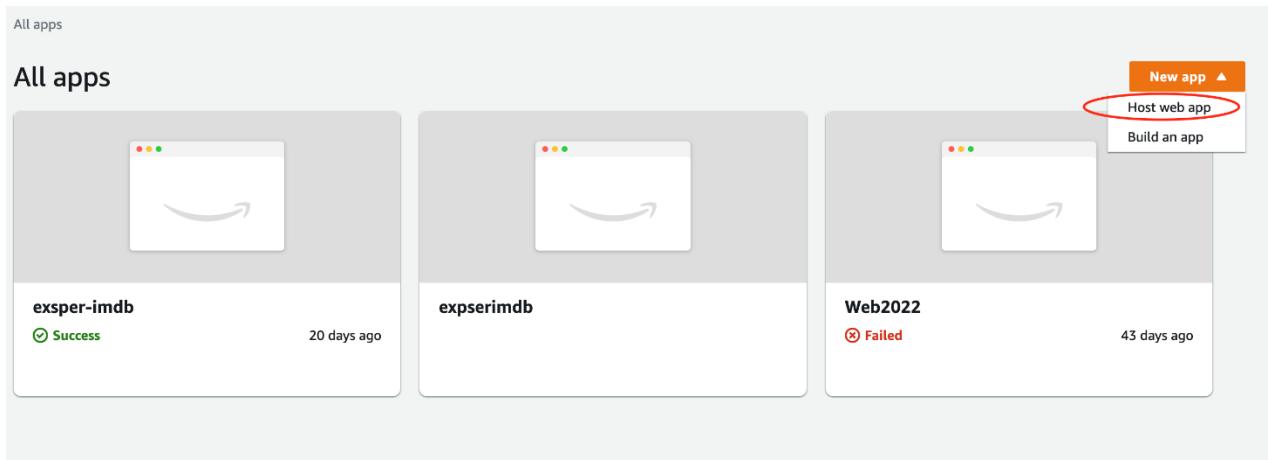


Figure 31: Create an empty app at first

All apps > Host your web app

## Get started with Amplify Hosting

Amplify Hosting is a fully managed hosting service for web apps. Connect your repository to build, deploy, and host your web app.

### From your existing code

Connect your source code from a Git repository or upload files to host a web app in minutes.

GitHub



Bitbucket



GitLab



AWS CodeCommit



Deploy without Git provider



Amplify Hosting requires read-only access to your repository.

**Continue**

Figure 32: Use source code from GitHub repo

All apps > Host your web app

Step 1

Add repository branch

Step 2

Configure build settings

Step 3

Review

## Add repository branch

### GitHub

GitHub authorization was successful.

Repository service provider



Recently updated repositories

Select a repository



If you don't see your repository in the dropdown above, ensure the Amplify GitHub App has permissions to the repository. If your repository still doesn't appear, push a commit and click the refresh button.

[View GitHub permissions](#)

Cancel

[Previous](#)

**Next**

Figure 33: Locate to a repo branch

## Step 2: Commit and push latest update to Github repository to build and deploy the frontend

```

1 2022-09-14T13:18:14 [INFO]: Beginning deployment for application d2zpym7slp1dfu, branch:beta, buildId 0000000013
2 2022-09-14T13:18:43 [INFO]: Deploying SSR Resources. Distribution ID: E2KQ1DEB8FR1EJ. This may take a few minutes
...
3 2022-09-14T13:18:44 [INFO]: Deployed the following resources to your account:
4 2022-09-14T13:18:44 [INFO]: - CloudFront Domain ID: d3cdrrns7z3wfny
5 2022-09-14T13:18:44 [INFO]: - SSR Lambda@Edge: gu7a4l-ci6i5nvv
6 2022-09-14T13:18:44 [INFO]: - API Lambda@Edge: gu7a4l-m94z8
7 2022-09-14T13:18:44 [INFO]: - Image Optimization Lambda@Edge: gu7a4l-8cik6or
8 2022-09-14T13:18:44 [INFO]: - S3 Bucket: gu7a4l-4c22ma
9 2022-09-14T13:18:44 [INFO]: Deployment complete
10 ||

```

Figure 34: Deployment successfully with new resources: CloudFront, Lambda Edge, and S3 bucket

## Step 3: Add Amplify backend environment to connect with Cognito and Kinesis Data Stream for Analytics

This tab lists all backend environments. Each backend environment is a container for all of the cloud capabilities added to your app such as API, auth, and storage.

Backend builder	UI component builder	App management
Define a data model and API, add authentication, and file storage to your app. No AWS account required.	Accelerate UI development with a pre-built React component library. Export Figma designs to code.	Invite app admins to modify app content, audit users and groups, and store files. No AWS account required.

Figure 35: Connect Amplify to Cognito and Kinesis Data Stream

## Step 4: Set up Amplify CLI on local machine using NPM

GET STARTED

## Installation

### Install the Amplify CLI

The Amplify Command Line Interface (CLI) is a unified toolchain to create AWS cloud services for your app. Let's go ahead and install the Amplify CLI.



NPM    cURL (Mac and Linux)    cURL (Windows)

```
npm install -g @aws-amplify/cli
```

copy

Figure 36: Install AWS Amplify CLI with npm

### Step 5: Pull the Amplify backend to local machine

```
> (base) katietran@current_working_directory imdb % amplify pull --appId d2zpym7slpldfu --envName staging
```

Figure 37: Pull the Amplify back-end to local

## Set up Cognito on Amplify

### Step 1: Add Cognito and config through Amplify CLI

```

(base) katietran@current_working_directory imbd % amplify add auth
Using service: Cognito, provided by: awscloudformation

The current configured provider is Amazon Cognito.

Do you want to use the default authentication and security configuration? Manual configuration
Select the authentication/authorization services that you want to use: User Sign-Up, Sign-In, connected with AWS IAM controls
Provide a friendly name for your resource that will be used to label this category in the project: exsperPoolWithAnalytics
Enter a name for your identity pool. exsperPoolWithAnalytics
Allow unauthenticated logins? (Provides scoped down permissions that you can control via AWS IAM) Yes
Do you want to enable 3rd party authentication providers in your identity pool? No
Provide a name for your user pool: exsperPoolWithAnalytics
Warning: you will not be able to edit these selections.
How do you want users to be able to sign in? Email
Do you want to add User Pool Groups? No
Do you want to add an admin queries API? No
Multifactor authentication (MFA) user login options: OFF
Email based user registration/forgot password: Disabled (Uses SMS/TOTP as an alternative)
Please specify an SMS verification message: Your verification code is {####}
Do you want to override the default password policy for this User Pool? Yes
Enter the minimum password length for this User Pool: 8
Select the password character requirements for your userpool:
Warning: you will not be able to edit these selections.
What attributes are required for signing up? Email
Specify the app's refresh token expiration period (in days): 30
Do you want to specify the user attributes this app can read and write? Yes
Specify read attributes: Address, Birthdate, Email, Family Name, Middle Name, Gender, Locale, Given Name, Name, Nickname, Phone
Specify write attributes: Address, Birthdate, Family Name, Middle Name, Gender, Locale, Given Name, Name, Nickname, Phone Number
Do you want to enable any of the following capabilities?
Do you want to use an OAuth flow? No
? Do you want to configure Lambda Triggers for Cognito? Yes
? Which triggers do you want to enable for Cognito
✓ Successfully added auth resource exsperPoolWithAnalytics locally

```

```

(base) katietran@current_working_directory imbd % amplify push
✓ Successfully pulled backend environment staging from the cloud.

```

Current Environment: staging

Category	Resource name	Operation	Provider plugin
Auth	exsperPoolWithAnalytics	Create	awscloudformation
Auth	exsperimdb	Unlink	awscloudformation
Function	ExsperAmplifyPresignupAutoconfirm	No Change	awscloudformation

? Are you sure you want to continue? Yes

Deployment completed.

```

Deployed root stack exsperimdb [ ===== ] 3/3
    amplify-amplifyd5bd157199814... AWS::CloudFormation::Stack      UPDATE_COMPLETE
    functionExsperAmplifyPresignu... AWS::CloudFormation::Stack      UPDATE_COMPLETE
    authexsperPoolWithAnalytics   AWS::CloudFormation::Stack      CREATE_COMPLETE
Deployed auth exsperPoolWithAnalytics [ ===== ] 11/11
    SNSRole                      AWS::IAM::Role                CREATE_COMPLETE
    UserPool                     AWS::Cognito::UserPool        CREATE_COMPLETE
    UserPoolClient               AWS::Cognito::UserPoolClient  CREATE_COMPLETE
    UserPoolClientWeb            AWS::Cognito::UserPoolClient  CREATE_COMPLETE
    UserPoolClientRole           AWS::IAM::Role                CREATE_COMPLETE
    UserPoolClientLambda         AWS::Lambda::Function        CREATE_COMPLETE
    UserPoolClientLambdaPolicy  AWS::IAM::Policy              CREATE_COMPLETE
    UserPoolClientLogPolicy     AWS::IAM::Policy              CREATE_COMPLETE
    UserPoolClientInputs        Custom::LambdaCallout       CREATE_COMPLETE
    IdentityPool                 AWS::Cognito::IdentityPool   CREATE_COMPLETE
    IdentityPoolRoleMap          AWS::Cognito::IdentityPoolRole  CREATE_COMPLETE

```

Figure 38: Configure AWS Cognito using Amplify CLI

## Step 2: Revise User Pool configurations on the Cognito Console in Amazon AWS Console

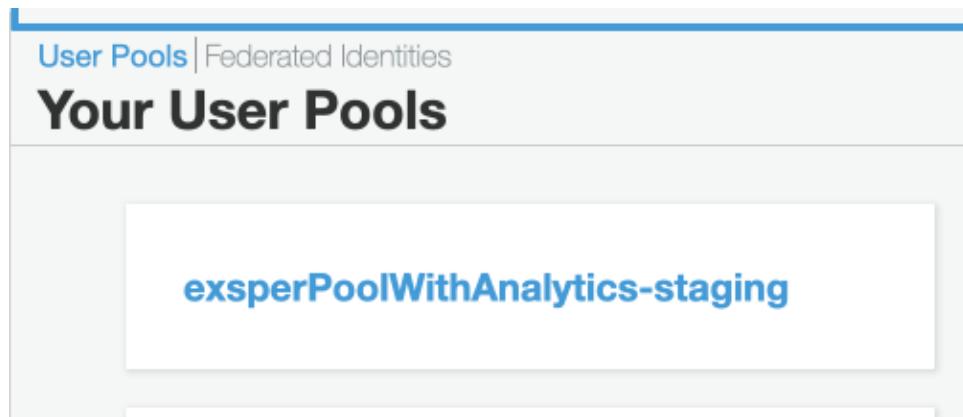


Figure 39: Create a new user pool

### Step 3: Add custom attributes and turn off all verifications and unrelated services such as Email, SNS, SES

The configuration screen for the 'exsperPoolWithAnalytics-staging' user pool. It is divided into several sections:

- Required attributes:** email
- Alias attributes:** none
- Username attributes:** email
- Enable case insensitivity?**: Yes
- Custom attributes:** custom:full\_name, custom:USER\_ID

- Minimum password length:** 8
- Password policy:** no requirements
- User sign ups allowed?**: Users can sign themselves up

- FROM email address:** Default
- Email Delivery through Amazon SES:** No
  - Note:** You have chosen to have Cognito send emails on your behalf. Best practices suggest that customers send emails through Amazon SES for production User Pools due to a daily email limit. [Learn more about email best practices.](#)

- MFA:** [Enable MFA...](#)
- Verifications:** none

Figure 40: Configure User Pools

### Step 4: Add domain name

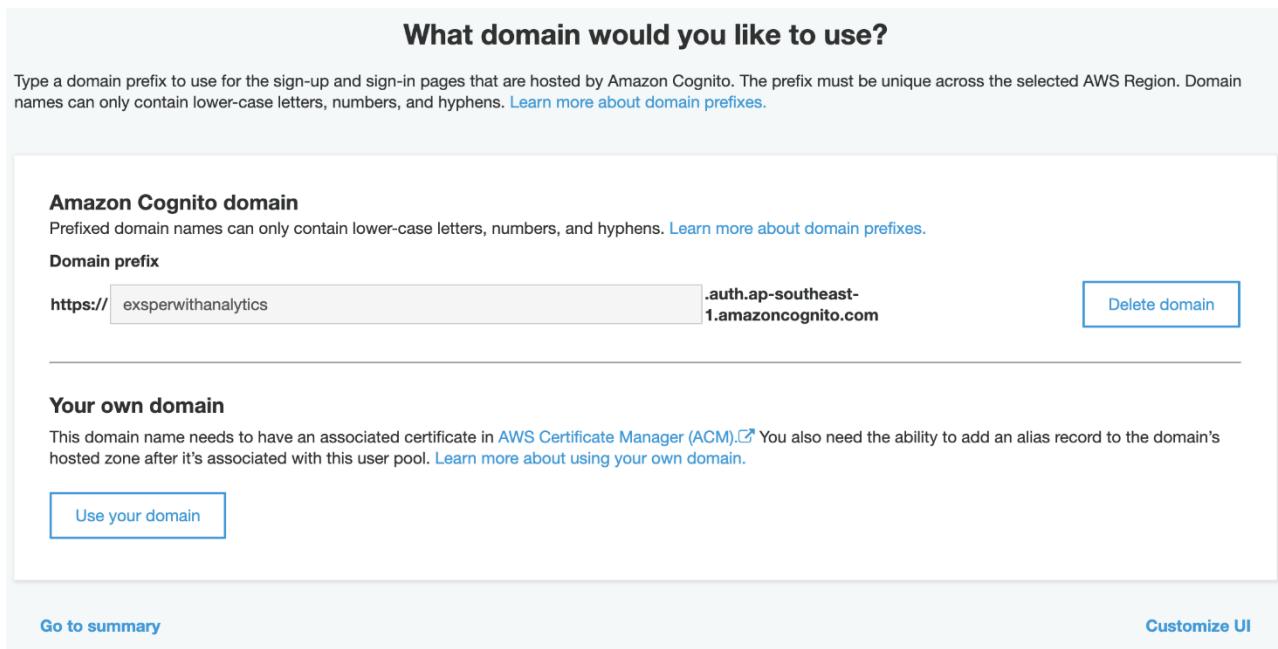


Figure 41: Create AWS Cognito domain

## Step 5: Create a Lambda Function to automatically confirm user on signing up

```

File Edit Find View Go Tools Window Test Deploy
Go to Anything (% P) index.js
ExperAmplifyPresc node_modules
event.json
index.js
package.json
yarn.lock
1 exports.handler = (event, context, callback) => {
2
3     // Confirm the user
4     event.response.autoConfirmUser = true;
5
6     // Set the email as verified if it is in the request
7     if (event.request.userAttributes.hasOwnProperty("email")) {
8         event.response.autoVerifyEmail = true;
9     }
10
11
12     // Return to Amazon Cognito
13     callback(null, event);
14 };

```

Figure 42: Create lambda function to verify user email on sign-up

## Step 6: Add Cognito credentials to Amplify Environment variables

Variable	Value	Branch
AMPLIFY_NATIVECLIENT_ID	287fjl00jdt2qk7henrkqiqv2	All branches
AMPLIFY_USERPOOL_ID	ap-southeast-1_1eyeYTptG	All branches
AMPLIFY_WEBCLIENT_ID	2grsb7tll46df8772dnrisbjbr	All branches
COGNITO_APPCLIENT_WEB	2grsb7tll46df8772dnrisbjbr	All branches
COGNITO_DOMAIN	experwithanalytics.auth.ap-southeast-1.amazoncognito.com	All branches
COGNITO_ID	287fjl00jdt2qk7henrkqiqv2	All branches
COGNITO_POOL_ID	ap-southeast-1_1eyeYTptG	All branches
COGNITO_REGION	ap-southeast-1	All branches
COGNITO_SECRET	1kvjhdl6cmqlq7tqglu5c05c1q63n3tpobeijb71p7s2mnei2t7m3	All branches

Figure 43: Attach Cognito to Amplify

## Set up Amplify Analytics and Kinesis

### Step 1: Create Kinesis Stream

To prepare for streaming data, we need to set up a

The screenshot shows the 'Create data stream' configuration page in the AWS Kinesis service. The 'Data stream configuration' section includes a 'Data stream name' input field containing 'kinesisStream'. The 'Data stream capacity' section shows 'Capacity mode' with 'Provisioned' selected (radio button is checked). It also includes fields for 'Provisioned shards' (set to 1) and 'Total data stream capacity' (Maximum shard capacity is 2 MiB/second). The 'Write capacity' and 'Read capacity' sections show maximum values of 1 MiB/second and 2 MiB/second respectively. A note at the bottom states: 'Provisioned mode has a fixed-throughput pricing model. See [Kinesis pricing for Provisioned mode](#)'.

Figure 44: Create and Configure data stream

## Step 2: Add Kinesis Stream to Amplify using the Amplify CLI

```

• (base) katietran@current_working_directory imdb % amplify add analytics
? Select an Analytics provider Amazon Kinesis Streams
? Enter a Stream name amplifyAnalyticsExper
? Enter number of shards 1
Successfully added resource amplifyAnalyticsExper locally

Some next steps:
"amplify push" builds all of your local backend resources and provisions them in the cloud
"amplify publish" builds all your local backend and front-end resources (if you have hosting category added) and provisions them in the cloud

• (base) katietran@current_working_directory imdb % amplify push
✓ Successfully pulled backend environment staging from the cloud.

  Current Environment: staging



| Category  | Resource name                    | Operation | Provider plugin   |
|-----------|----------------------------------|-----------|-------------------|
| Analytics | amplifyAnalyticsExper            | Create    | awscloudformation |
| Auth      | experPoolWithAnalytics           | No Change | awscloudformation |
| Function  | ExperAmplifyPresignupAutoconfirm | No Change | awscloudformation |



? Are you sure you want to continue? Yes

Deployment completed.
Deployed root stack experimdb [ ===== ] 4/4
amplify-amplifyd5bd157199814... AWS::CloudFormation::Stack UPDATE_COMPLETE
analyticsamplifyAnalyticsExper AWS::CloudFormation::Stack CREATE_COMPLETE
functionExperAmplifyPresignu... AWS::CloudFormation::Stack UPDATE_COMPLETE
authexperPoolWithAnalytics AWS::CloudFormation::Stack UPDATE_COMPLETE
Deployed analytics amplifyAnalyticsExper [ ===== ] 3/3
KinesisStream AWS::Kinesis::Stream CREATE_COMPLETE
CognitoAuthPolicy AWS::IAM::Policy CREATE_COMPLETE
CognitoUnauthPolicy AWS::IAM::Policy CREATE_COMPLETE
Thu Sep 15 2022 09:10:30...
Thu Sep 15 2022 09:10:27...
Thu Sep 15 2022 09:09:26...
Thu Sep 15 2022 09:09:28...
Thu Sep 15 2022 09:09:37...
Thu Sep 15 2022 09:10:14...
Thu Sep 15 2022 09:10:22...

```

Figure 45: Configure AWS Kinesis Data Stream using Amplify CLI

## Step 3: Create S3 bucket to store data

To store event data after done processing for future reference, we create an s3 bucket using the following command

```

(base) nhatbuiminh@MBP-de-Nhat ~ % aws s3 mb s3://exper-movie-data
make_bucket: exper-movie-data
(base) nhatbuiminh@MBP-de-Nhat ~ %

```

Figure 46: Create S3 bucket for data storage

## Step 4: Create Kinesis Firehose Delivery Stream to send data to S3

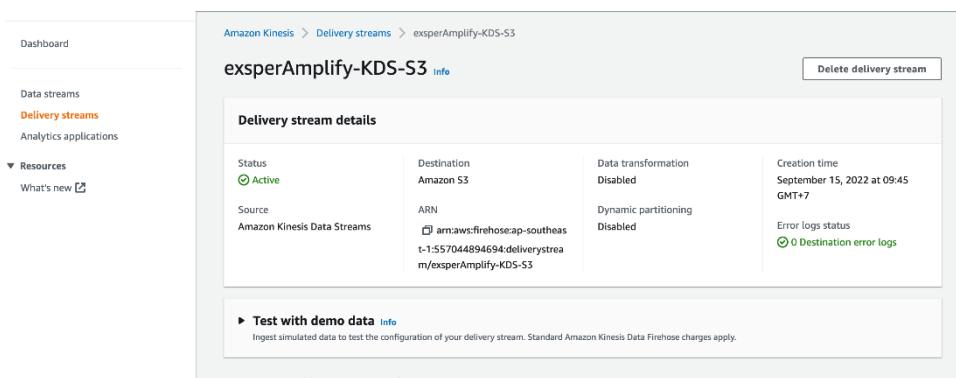


Figure 47: Configure Kinesis Firehose

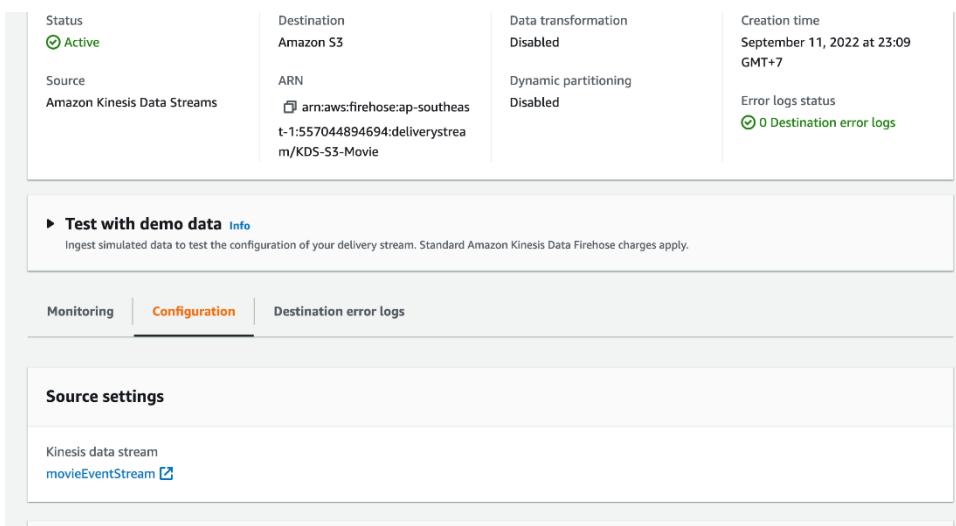


Figure 48: Select the source stream as the Kinesis

## Step 5: Create a Lambda function to handle streaming events and connect with AWS Personalize

After setup the Kinesis and Kinesis Firehose, we create a streaming function for processing click event data.

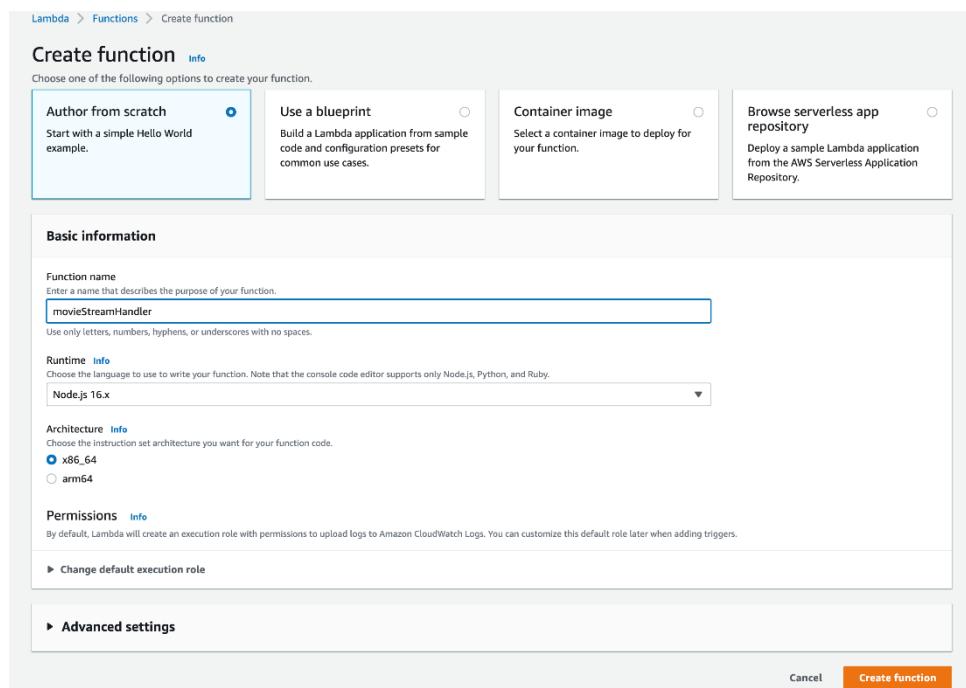


Figure 49: Create lambda function to handle events

Deploy the following code to lambda:

The code part contains necessary functions to received payload from Kinesis and process data for AWS Personalize

```

AWS.config.update({
  region:'ap-southeast-1',
  accessKeyId: process.env.ACCESS_KEY_ID,
  secretAccessKey: process.env.SECRET_ACCESS_KEY
});

var personalizeevents = new AWS.PersonalizeEvents();
// var dynamoClient = new AWS.DynamoDB.DocumentClient();

console.log('Loading function');

exports.handler = (event, context, callback) => {
  console.log(JSON.stringify(event, null, 2));

  event.Records.forEach(function(record) {
    // Kinesis data is base64 encoded so decode here
    var payload = Buffer.from(record.kinesis.data, 'base64').toString('ascii');
    console.log('Decoded payload:', payload);
    payload = JSON.parse(payload);
    var eventDate = new Date();
    var putEventsParams= {
      'sessionId': payload.SessionId, /* required */
      'trackingId': process.env.TRACKING_ID, /* required */
      'userId': payload.UserId,
      'eventList: [
        {
          'eventType': payload.EventType, /* required */
          'itemId': payload.ItemId, /* required */
          'sentAt': eventDate
        },
      ]
    }
    console.log("THIS IS THE OBJECT = " + JSON.stringify(putEventsParams,null,3))
    //Create a personalize event
    personalizeevents.putEvents(putEventsParams, function (err, data) {
      if (err) {
        console.log(err, err.stack); // an error occurred
      } else{
        console.log(data);           // successful response
        putEventsParams['eventList'][0]['sentAt']=putEventsParams['eventList'][0]['sentAt'].toTimeString();
        const putEventsErrResponse = {
          statusCode: 500,
          body: JSON.stringify(err),
        };
        callback(null, putEventsErrResponse);
        const response = {
          statusCode: 200,
          body: JSON.stringify(putEventsParams),
        };
        callback(null, response);
      }
    });
  });
};


```

Figure 50: Code deployment on Lambda function

## Step 6: Set up environment variable

The following are the necessary variable for the lambda function to process the incoming streaming data

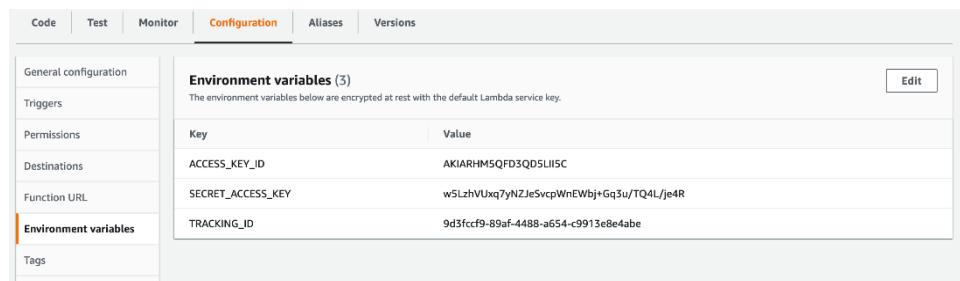


Figure 51: Create environment variables

## Step 7: Config Lambda IAM Role permission to add trigger

In order for the lambda function to work properly, we have to assign IAM Role to allow it to access to S3, Kinesis and Firehose.

The screenshot shows the AWS Lambda configuration interface. The top navigation bar includes tabs for Code, Test, Monitor, Configuration (which is selected), Aliases, and Versions. On the left, a sidebar lists General configuration, Triggers, Permissions (which is selected and highlighted in orange), and Destinations. The main content area is titled "Execution role" and shows a role name: "experAmplifyEventStream-role-oo3w54hh". Below this, there is a section titled "Permissions policies (7)" with a note: "You can attach up to 10 managed policies." A search bar is present with the placeholder "Filter policies by property or policy name and press enter". A list of seven attached policies is shown, each with a checkbox and a plus sign icon:

- AWSLambdaBasicExecutionRole-89bad845-33a2-4af8-81b7-c6c10836ac25
- KinesisFirehoseServicePolicy-experAmplify-KDS-S3-ap-southeast-1
- KinesisFirehoseServicePolicy-KDS-S3-Movie-ap-southeast-1
- KinesisFirehoseServicePolicy-movieDeliveringStream-ap-southeast-1

Figure 52: Create policy permissions for Lambda to access S3, Kinesis and Firehose

### Step 8: Add Kinesis trigger to Lambda function

The function is triggered when the click event is broadcasted to Kinesis

The image consists of two screenshots from the AWS Lambda console.

The top screenshot shows the "Triggers" page with one trigger listed:

- Trigger:** Kinesis: movieEventStream
- arn:aws:kinesis:ap-southeast-1:557044894694:stream/movieEventStream
- state: Enabled

The bottom screenshot shows the "movieEventHandling" Lambda function overview:

- Function overview:** movieEventHandling
- Description:** -
- Last modified:** 3 days ago
- Function ARN:** arn:aws:lambda:ap-southeast-1:557044894694:function:movieEventHandling
- Function URL:** -

Below the overview, there are tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. The Code tab is selected.

Figure 53: Add Kinesis as a trigger for Lambda function

## Step 9 : Record the click stream event from the front end with Amplify SDK

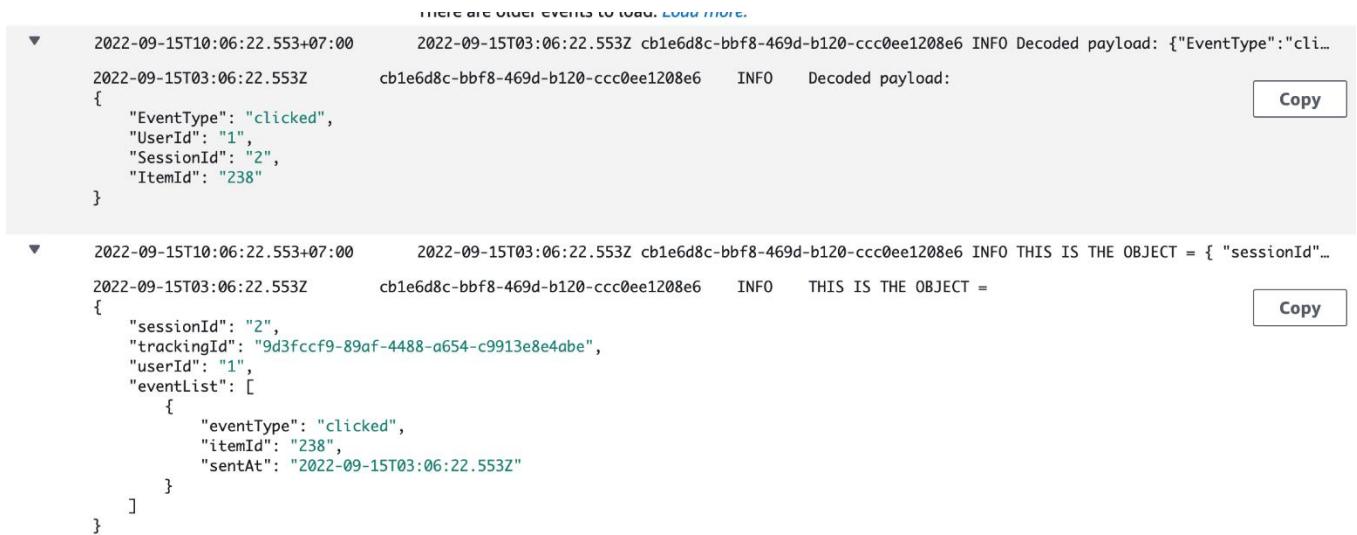
After done setting up the server side, we set up built in AWS Amplify analytics event to record the click event and transmitted it to Kinesis

```
useEffect(() => {
  const runAnalytics = async () => {
    const currentUser = await Auth.currentAuthenticatedUser();
    if (currentUser) {
      Analytics.record({
        data: {
          EventType: "clicked",
          UserId: currentUser.attributes["custom:USER_ID"],
          SessionId: "",
          ItemId: movieID,
        },
        streamName: "amplifyAnalyticsExsper-staging",
      },
      "AWSKinesis"
    );
  }
  runAnalytics()
}, [])
```

Figure 54: Collect click stream events using Amplify SDK

## Step 10: Monitor events on Cloudwatch

After setting up properly, the data payload is streamed using Kinesis and process. The payload data is written in the log for verification purposes.



The screenshot shows a log stream from AWS CloudWatch. The logs are timestamped at 2022-09-15T03:06:22.553Z. The first log entry is a decoded payload object:

```
2022-09-15T03:06:22.553Z cb1e6d8c-bbf8-469d-b120-ccc0ee1208e6 INFO Decoded payload: {"Event...  
{  
  "EventType": "clicked",  
  "UserId": "1",  
  "SessionId": "2",  
  "ItemId": "238"  
}  
2022-09-15T03:06:22.553Z cb1e6d8c-bbf8-469d-b120-ccc0ee1208e6 INFO THIS IS THE OBJECT = { "sessionId"...  
2022-09-15T03:06:22.553Z cb1e6d8c-bbf8-469d-b120-ccc0ee1208e6 INFO THIS IS THE OBJECT =  
{  
  "sessionId": "2",  
  "trackingId": "9d3fccf9-89af-4488-a654-c9913e8e4abe",  
  "userId": "1",  
  "eventList": [  
    {  
      "eventType": "clicked",  
      "itemId": "238",  
      "sentAt": "2022-09-15T03:06:22.553Z"  
    }  
  ]  
}
```

Each log entry includes a 'Copy' button to the right.

Figure 55: Observe click stream event on AWS Cloud Watch

## VIII. A small user manual

\*Note: Before experiencing the app, user must follow on the link in the section 2.

### 1. Homepage

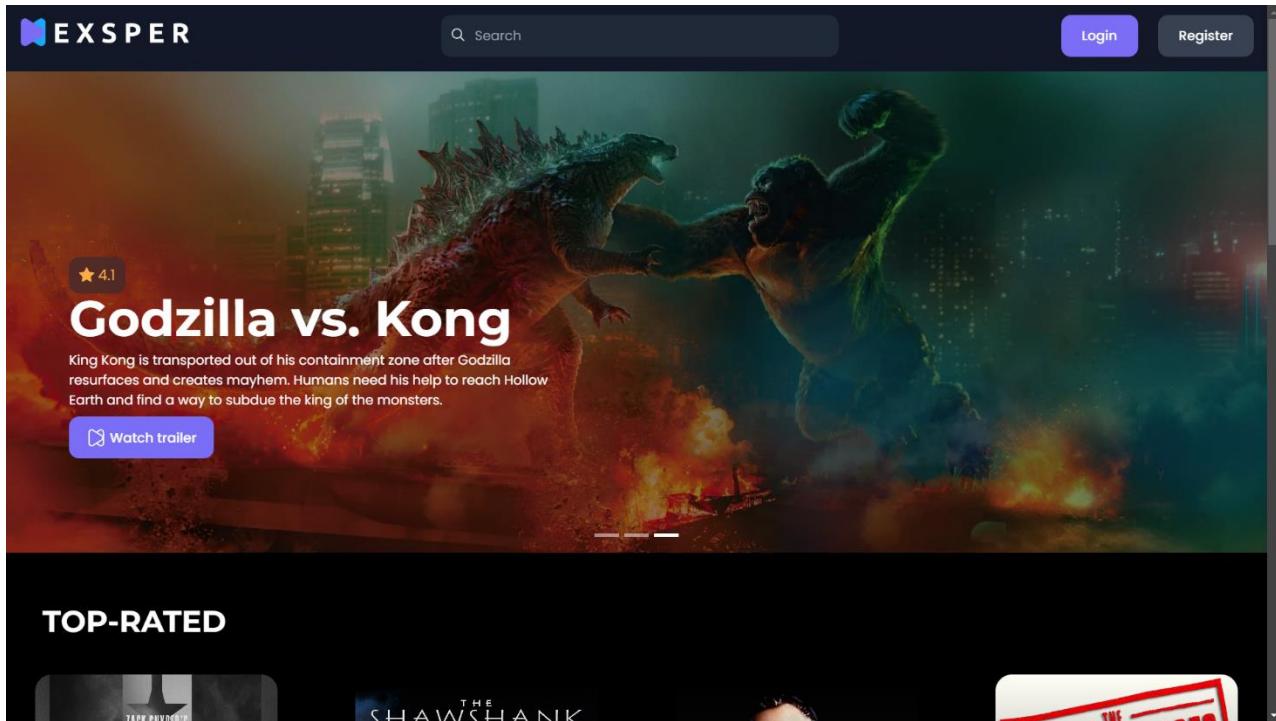


Figure 56: Exsper's Homepage

After being directed by AWS Amplify, the user will be on the homepage spotlighted by top trend movies. These such movies are briefly described by their names, rating stars, sub-headings, and a trailer, embedded with a corresponding YouTube link. The homepage also provides the basic features including the search bar, log-in option for existing users, and sign-up option for new ones.

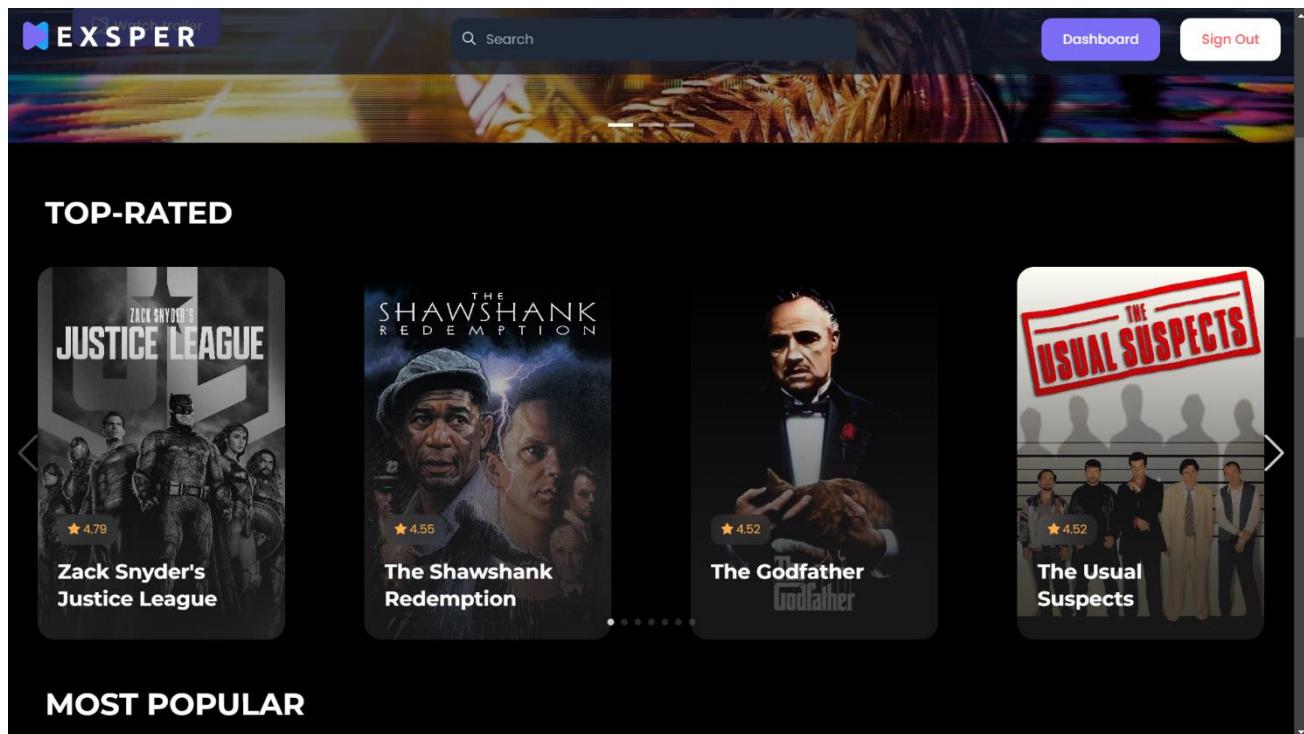


Figure 57: List of top-rated films

Scrolling down the homepage a bit, the application offers a list of movies that have a lot of favorable reviews from both experts and viewers. We design these movie items as a card list using image and text, and even icons to rapidly grab visitors' attention and make it simple for them to move between various films.

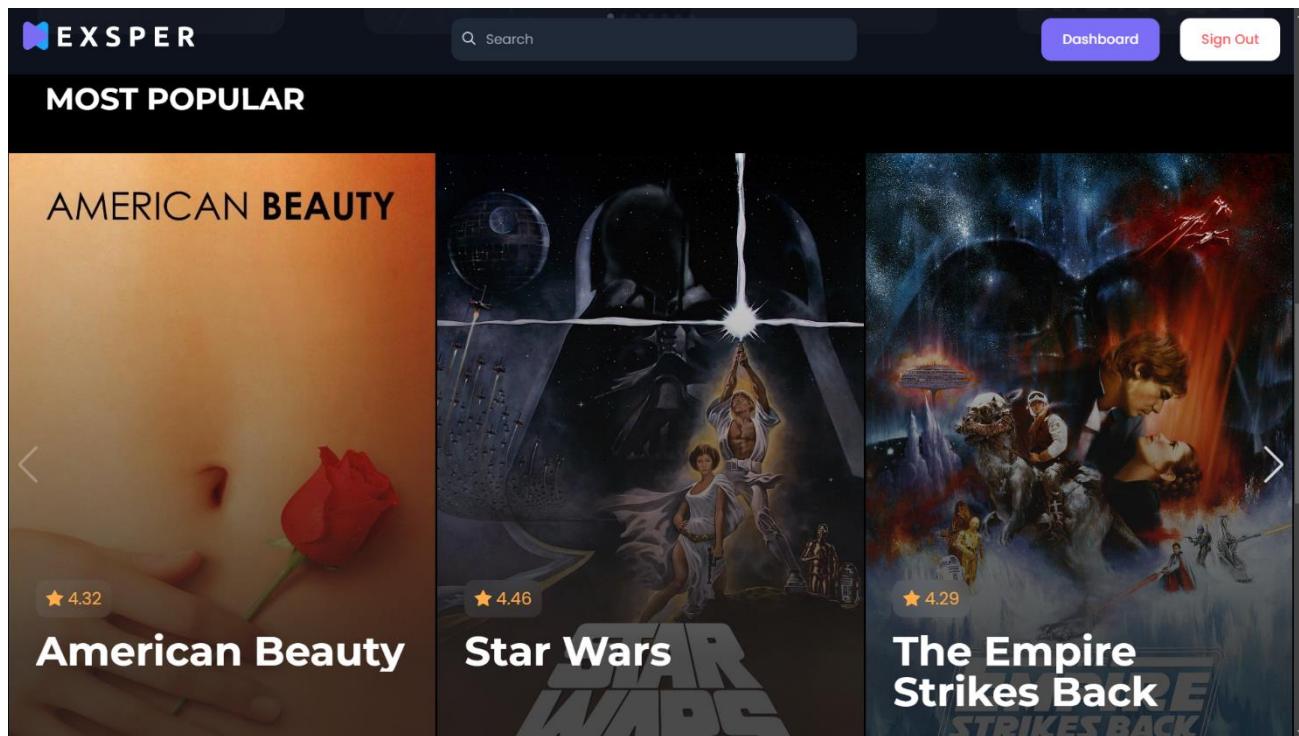


Figure 58: List of highest views movies

At the end of the homepage, there is the list of the series that attracted the most views all the time. The same design was chosen in this movie list because we expected it to be effective in terms of visual appeal.

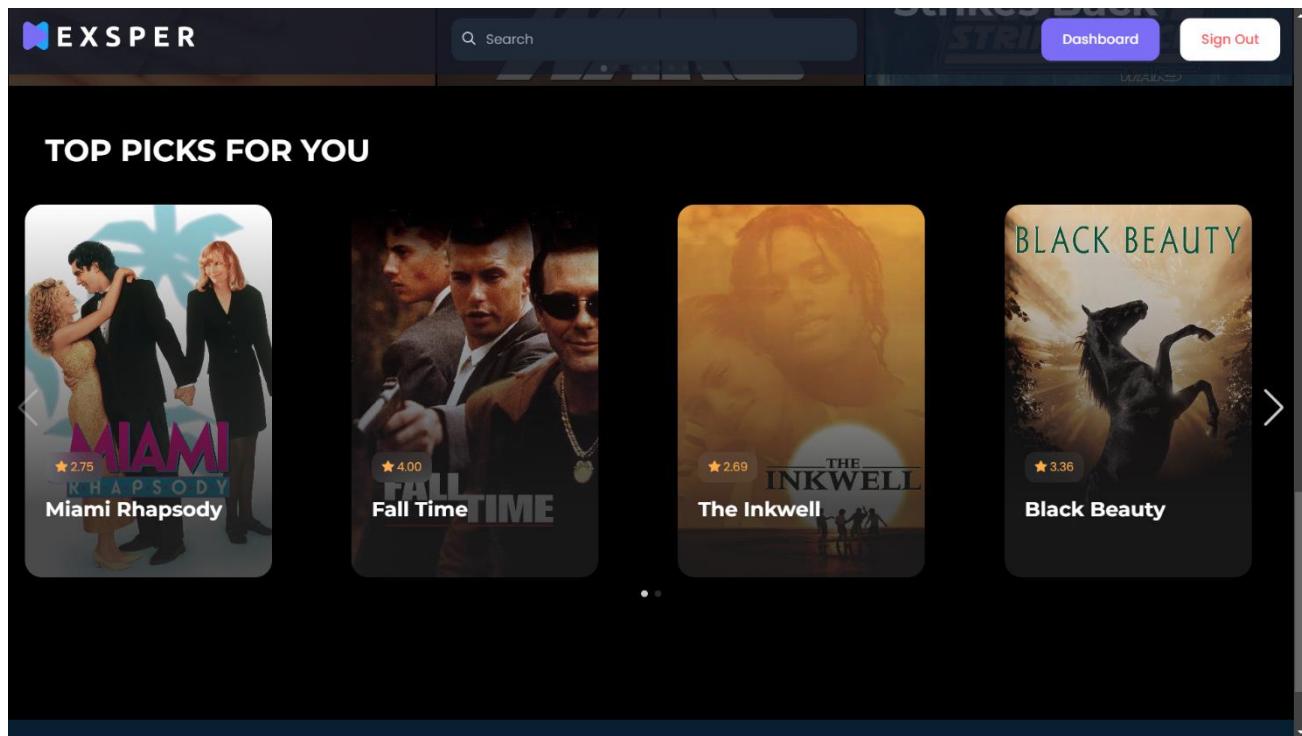


Figure 59: List of suggested movies

## 2. Log-in page

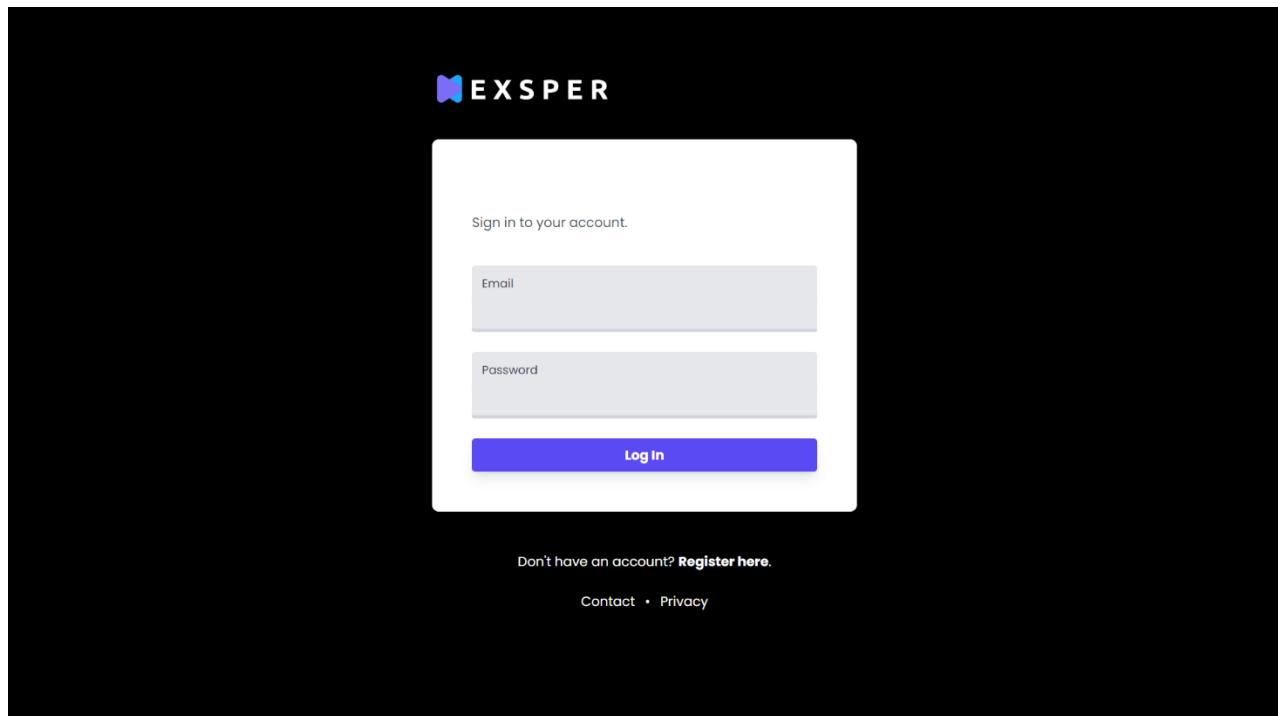


Figure 60: Login page after pressing Login button at the homepage

By clicking on the sign-in button, the system displays a login form containing two fields (Email and Password) then the user has to fill these with their account, which is provided or created beforehand, to jump into the app.

### 3. Sign-up page

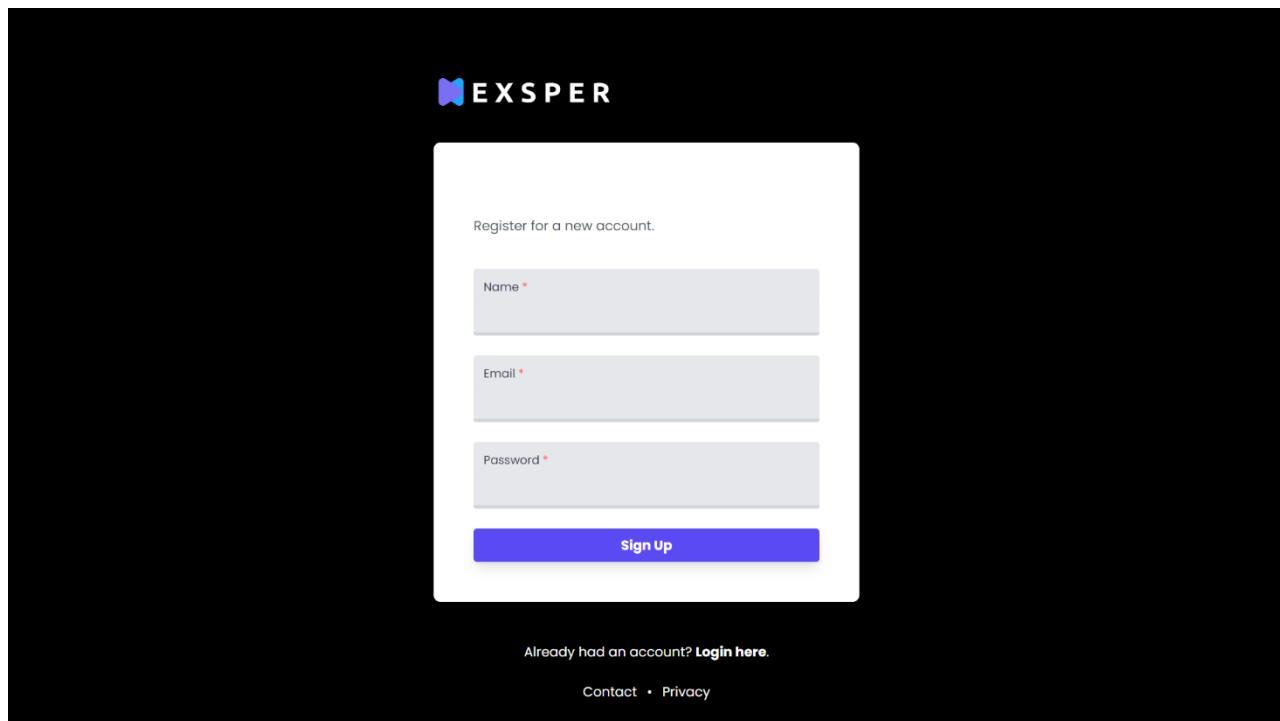


Figure 61: Signup page after pressing Register button from the homepage or login form

In case user is new to the app, there is a sign-up button in the homepage, or they can refer to the keyword **Register here**, located in log-in form, to get into the sign-up form. To pass this step, user must provide the required information (Name, Age, Email, Password, Confirmed Password) in order to get valid credentials after all.

## 4. Movie detail page

The screenshot shows the movie detail page for "Godzilla vs. Kong (2021)". At the top, there is a large image of the two titanic monsters fighting in a city. Below the image, a dark overlay displays the movie's title, rating (4.23), and genres (Action, Fantasy, Science Fiction). The main content area features a large image of the movie's poster, followed by a summary: "One Will Fall. In a time when monsters walk the Earth, humanity's fight for its future sets Godzilla and Kong on a collision course that will see the two most powerful forces of nature on the planet collide in a spectacular battle for the ages." To the right of the summary, there is a "Rate this movie" section with a 5-star rating icon.

The screenshot shows the movie detail page for "Godzilla vs. Kong (2021)". On the left, there is a large image of the movie's poster, showing the two titanic monsters, Godzilla and Kong, facing each other over a city skyline. To the right of the poster, there is a summary: "One Will Fall. In a time when monsters walk the Earth, humanity's fight for its future sets Godzilla and Kong on a collision course that will see the two most powerful forces of nature on the planet collide in a spectacular battle for the ages." Below the summary, there are sections for "Release Date" (March 24th 2021), "Duration" (114 min), "Starrings" (Alexander Skarsgård, Millie Bobby Brown, Rebecca Hall), "Directed by" (Adam Wingard), and "Written by" (Terry Rossio, Michael Dougherty, Eric Pearson, Max Borenstein, Zach Shields). At the bottom, there is a "Rate this movie" section with a 5-star rating icon.

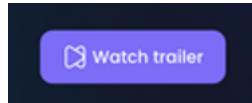
The image consists of two vertically stacked screenshots of the Exper-IMDB mobile application. Both screenshots show the same movie page for 'Zack Snyder's Justice League' (2021).  
The top screenshot shows a modal window titled 'Trailer' displaying a video thumbnail of a shirtless man (likely Henry Cavill as Superman) standing in front of a city skyline at night. The video player has a progress bar and a 'CLOSE' button. Below the trailer, there is a summary of the movie's plot: 'Determined to ensure Superman's ultimate sacrifice was not in vain, Bruce Wayne aligns forces with Diana Prince with plans to recruit a team of metahumans to protect the world from an approaching threat of catastrophic proportions.'  
The bottom screenshot shows the main movie details page. It features the movie's title 'Zack Snyder's Justice League (2021)' with a rating of 4.79, genres (Action, Adventure, Fantasy, Science Fiction), and a large poster image. On the right side of the page, there is a 'Watch trailer' button, a plot summary, release date (March 18th 2021), duration (242 min), and starring information.

Figure 62: Movie details

When users click one of the spotlighted films on the homepage, the app will drive them to the expansive page containing more details about that movie. Some of the details could be shown up such as Description, Release Date, Duration, Director,

Writers, and a “Rate this movie” module where user can rate the film based upon their preference.

User can also view the trailer of that movie by clicking on the “Watch trailer” button



## 5. Search movie

Movie Title	Release Year
Star Wars	1977
Star Trek: Insurrection	1998
Star Trek: The Motion Picture	1979
Star Trek: First Contact	1996

Figure 63: A returned list of film that include the searching keyword

The app provides a searching feature as an efficient way to find the specific film by looking up its name. It is notable that the search operation will search by word, meanings the searched result may return a list of movies whose title includes the searching keyword.

## 6. User Dashboard

In the user dashboard page, user can view their personal information such as Name, Age, Gender, and Email address. When scrolling down, users can see their own history of recently viewed items and recently rated items.

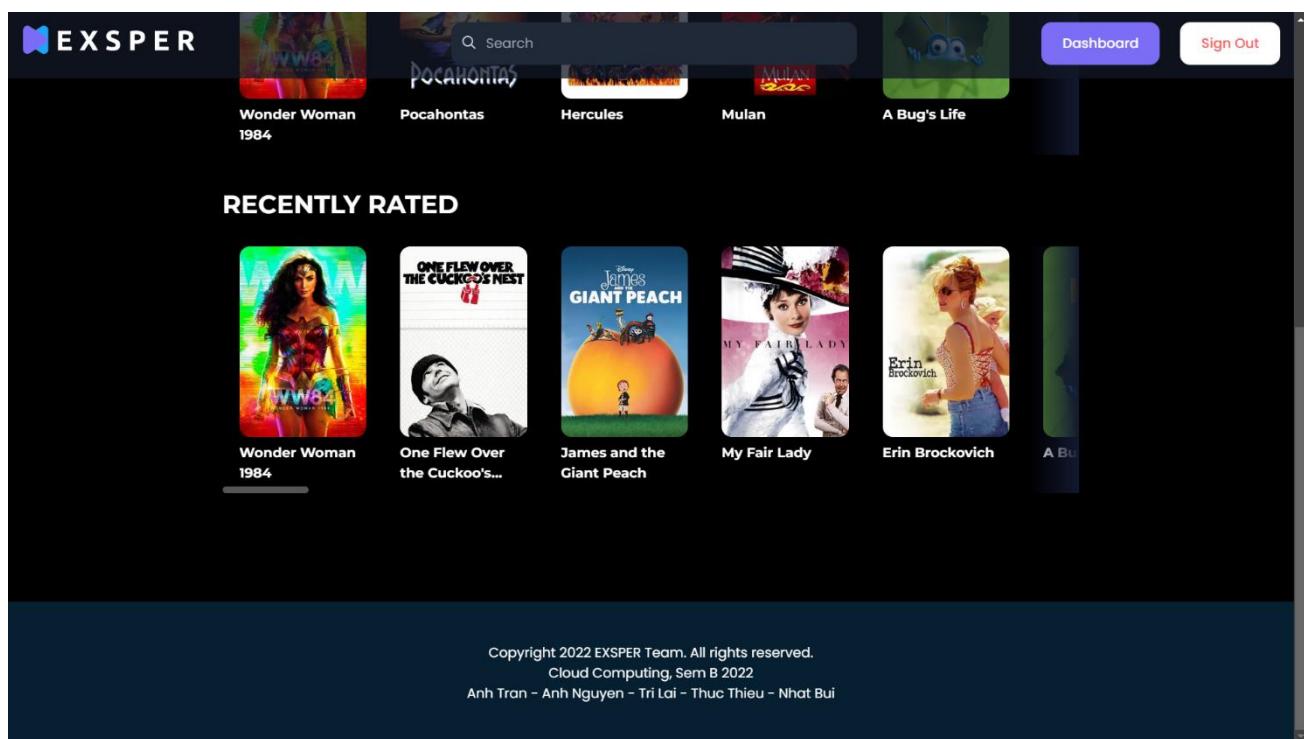
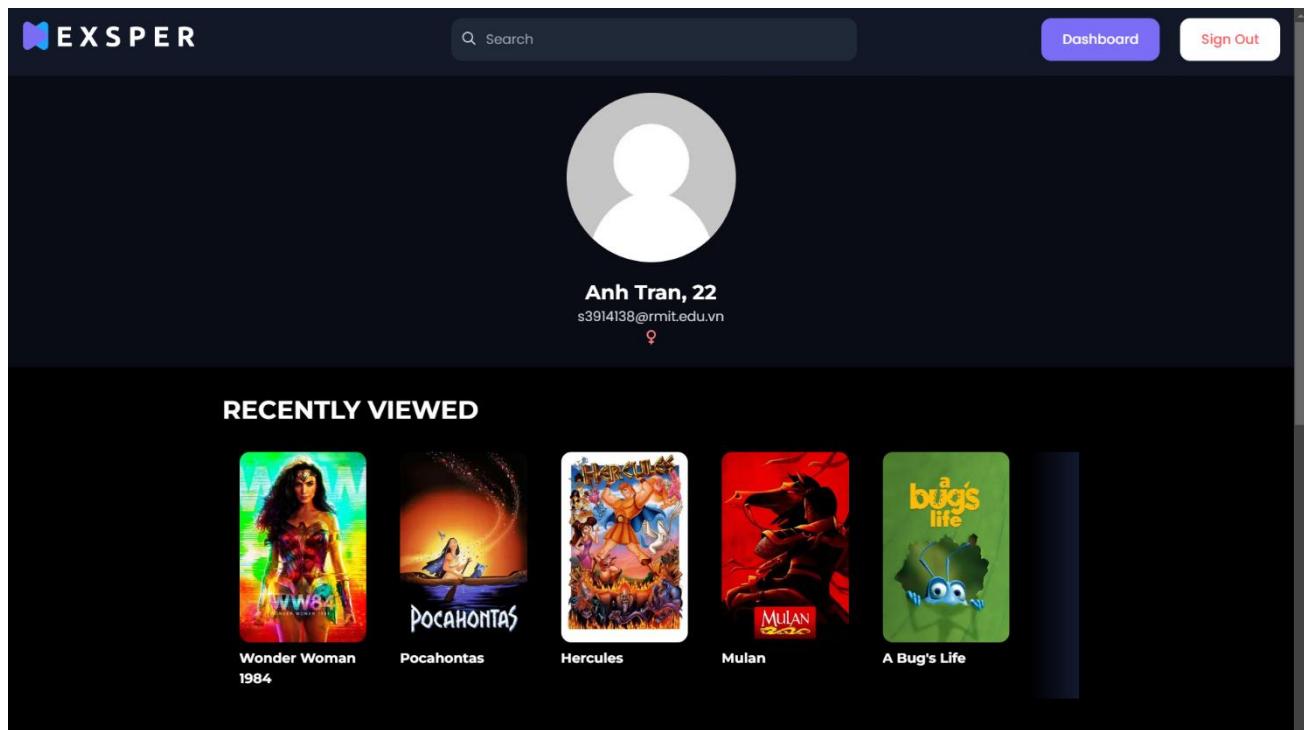


Figure 64: Dashboard contains list of recent viewed and recent rated movies

## IX. References

- [1] F. Harper and J. Konstan, "MovieLens 25M Dataset", *GroupLens*, 2015. [Online]. Available: <https://grouplens.org/datasets/movielens/25m/>. [Accessed: 28- Aug- 2022].
- [2] "amazon-personalize-samples/next\_steps/operations/streaming\_events at master · aws-samples/amazon-personalize-samples", *GitHub*, 2022. [Online]. Available: [https://github.com/aws-samples/amazon-personalize-samples/tree/master/next\\_steps/operations/streaming\\_events](https://github.com/aws-samples/amazon-personalize-samples/tree/master/next_steps/operations/streaming_events). [Accessed: 03- Sep- 2022].
- [3] N. Thanh, "AWS DynamoDB", *Google Slides*, 2022. [Online]. Available: <https://docs.google.com/presentation/d/1fDtnGRdunp1hvhLiM1FZdNy54Ved2-IH/edit?pli=1#slide=id.p1>. [Accessed: 16- Aug- 2022].
- [4] N. Thanh, "AWS Lambda", *Google Slides*, 2022. [Online]. Available: [https://docs.google.com/presentation/d/12sTN8\\_4A0DPmnA1BYooOG-CMiya4\\_51S/edit#slide=id.p1](https://docs.google.com/presentation/d/12sTN8_4A0DPmnA1BYooOG-CMiya4_51S/edit#slide=id.p1). [Accessed: 18- Aug- 2022].
- [5] N. Thanh, "REST API", *Google Slides*, 2022. [Online]. Available: <https://docs.google.com/presentation/d/1rA9-j5oCGn9GfFoE8luxvVuMWU41vRz/edit#slide=id.p1>. [Accessed: 25- Aug- 2022].

## X. Video

<https://www.youtube.com/watch?v=bddrYQZe-Mk>

## XI. Appendices

### 1. Sample query user result on Thunder Client

Domain: <https://z1hi6ssas5.execute-api.ap-southeast-1.amazonaws.com>

#### 1.1 Get recommended videos for a given user

GET: {domain}/getPersonalize?userId={userId}&numResults={number of recommendations}

Response: list of tmdbId (string) of recommended videos

#### Example

Request: <https://z1hi6ssas5.execute-api.ap-southeast-1.amazonaws.com/getPersonalize?userId=15&numResults=5>

Response: ["71754", "39953", "9614", "48787", "12158"]

The screenshot shows a Postman interface with a GET request to <https://z1hi6ssas5.execute-api.ap-southeast-1.amazonaws.com/getPersonalize?userId=15&numResults=5>. The 'Query' tab is selected, showing parameters: userId (15) and numResults (5). The 'Send' button is blue. The 'Response' tab shows a status of 200 OK, size of 44 bytes, and time of 190 ms. The response body is a JSON array of movie IDs: [71754, 39953, 9614, 48787, 12158].

Figure 65: Top-picks for a certain user

## 1.2 Get similar videos to the target video

**GET:** {domain}/getSims?itemId={itemId}&numResults={number of recommendations}

Response: list of tmdbId (string) of similar videos

### Example

Request: <https://z1hi6ssas5.execute-api.ap-southeast-1.amazonaws.com/getSims?itemId=862&numResults=10>

Response: ["16934", "9073", "34444", "9623", "9073", "78802", "18256", "688", "8987", "4584"]

The screenshot shows a Postman interface with a GET request to <https://z1hi6ssas5.execute-api.ap-southeast-1.amazonaws.com/getSims?itemId=862&numResults=10>. The 'Query' tab is selected, showing parameters: itemId (862) and numResults (10). The 'Send' button is blue. The 'Response' tab shows a status of 200 OK, size of 83 bytes, and time of 195 ms. The response body is a JSON array of movie IDs: [16934, 9073, 34444, 9623, 9073, 78802, 18256, 688, 8987, 4584].

Figure 66: List of same genre films for an anonymous user

## 1.3 Get recommended videos given the target video and current user

**GET:** {domain}/getReranking?

userId={userId}&itemId={itemId}&numResults={number of recommendations}

Response: list of tmdbId (string) of recommended videos

### Example

Request: <https://z1hi6ssas5.execute-api.ap-southeast-1.amazonaws.com/getReranking?userId=2&itemId=91&numResults=5>  
Response: ["11862", "11443", "2625", "25066", "2086"]

The screenshot shows the Postman interface with a GET request to `https://z1hi6ssas5.execute-api.ap-southeast-1.amazonaws.com/getReranking?userId=2&itemId=91&numResults=5`. The query parameters are: userId (2), itemId (91), numResults (5). The response status is 200 OK, size is 43 Bytes, and time is 276 ms. The response body is a JSON array:

```
[{"id": "11862"}, {"id": "11443"}, {"id": "2625"}, {"id": "25066"}, {"id": "2086"}]
```

Figure 67: List of same genre films for an authenticated user

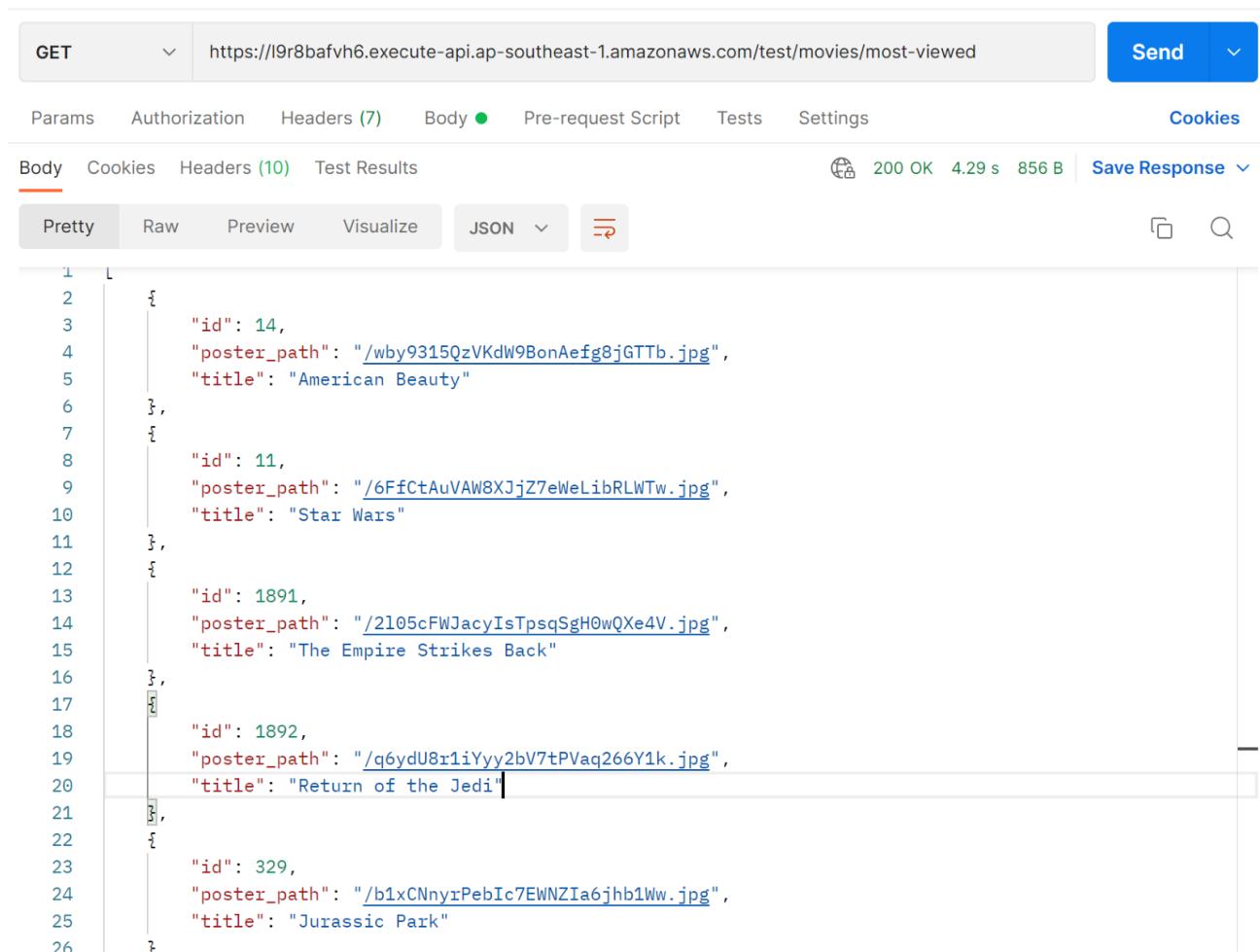
## 2. Sample query movie result on Postman

### 1. List 10 most-viewed movies

URL: Invoke URL + /movies/most-viewed

Method: GET

Sample result:



```
1 L
2 {
3     "id": 14,
4     "poster_path": "/wby9315QzVKdW9BonAefg8jGTTb.jpg",
5     "title": "American Beauty"
6 },
7 {
8     "id": 11,
9     "poster_path": "/6FfCtAuVAW8XJjZ7eWeLibRLWTw.jpg",
10    "title": "Star Wars"
11 },
12 {
13     "id": 1891,
14     "poster_path": "/2l05cFWJacyIsTpsqSgH0wQXe4V.jpg",
15     "title": "The Empire Strikes Back"
16 },
17 [
18     {
19         "id": 1892,
20         "poster_path": "/q6ydU8r1iYyy2bV7tPVaq266Y1k.jpg",
21         "title": "Return of the Jedi"
22     },
23     {
24         "id": 329,
25         "poster_path": "/b1xCNnyrPebIc7EWNZIa6jhblWw.jpg",
26         "title": "Jurassic Park"
27     }
28 ]
```

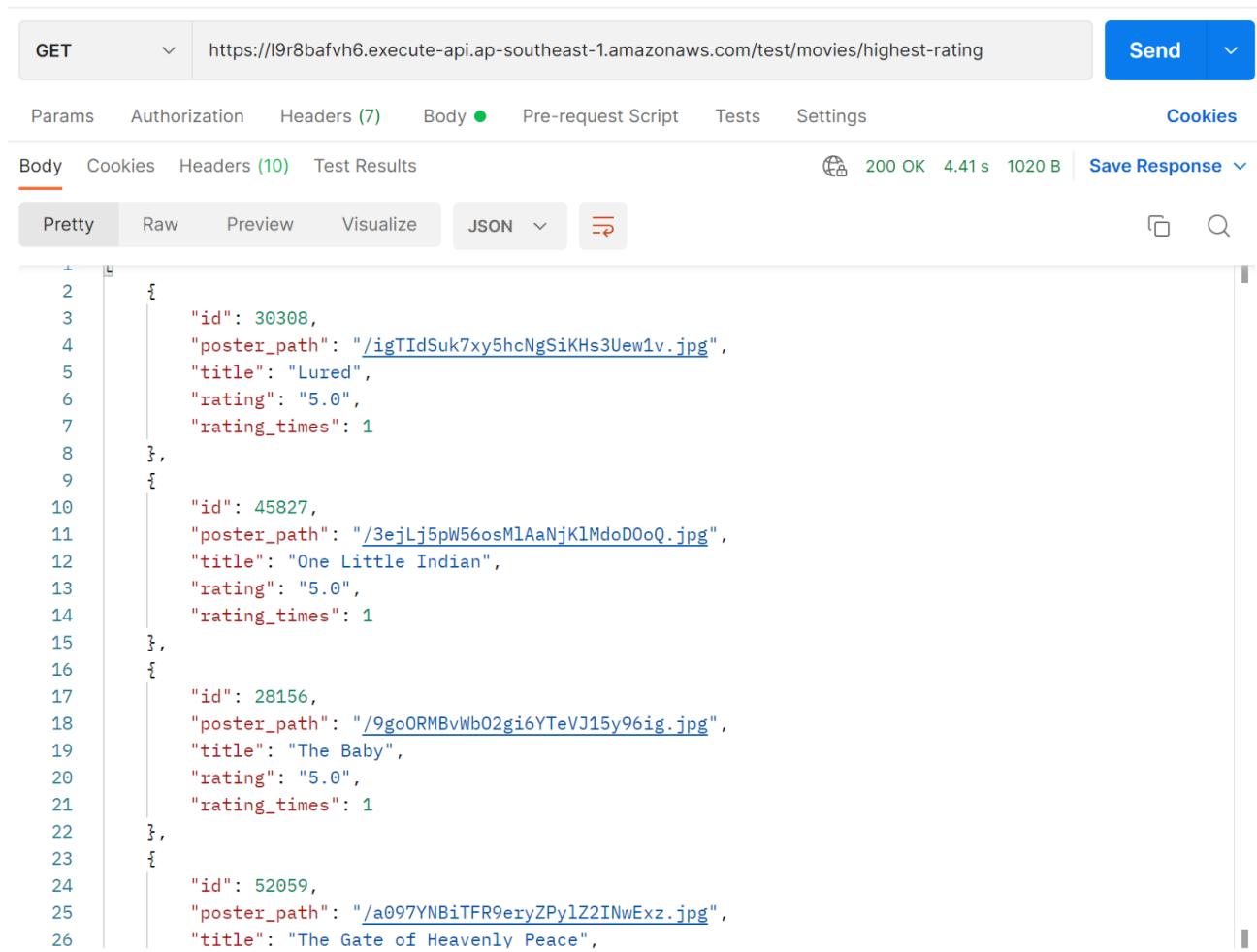
Figure 68: 10 most view films returned in JSON format

## 2. List 10 highest-rated movies

URL: Invoke URL + /movies/highest-rating

Method: GET

Sample result:



The screenshot shows the Postman application interface. At the top, there is a header bar with 'GET' selected as the method, the URL 'https://l9r8bafvh6.execute-api.ap-southeast-1.amazonaws.com/test/movies/highest-rating', a 'Send' button, and a dropdown menu. Below the header are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body' (which is currently selected and has a green dot), 'Pre-request Script', 'Tests', 'Settings', and 'Cookies'. Under the 'Body' tab, there are buttons for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'JSON' (which is selected and has a green dot). To the right of the body area, there is a status bar showing '200 OK', '4.41 s', '1020 B', and a 'Save Response' button. The main content area displays a JSON response with 10 movie entries, each containing an ID, poster path, title, rating, and rating count. The JSON is formatted with line numbers on the left.

```
2
{
  "id": 30308,
  "poster_path": "/igTIdSuk7xy5hcNgSiKhs3Uew1v.jpg",
  "title": "Lured",
  "rating": "5.0",
  "rating_times": 1
},
{
  "id": 45827,
  "poster_path": "/3ejLj5pW56osMlAaNjKlMdoD0oQ.jpg",
  "title": "One Little Indian",
  "rating": "5.0",
  "rating_times": 1
},
{
  "id": 28156,
  "poster_path": "/9go0RMBvWb02gi6YTeVJ15y96ig.jpg",
  "title": "The Baby",
  "rating": "5.0",
  "rating_times": 1
},
{
  "id": 52059,
  "poster_path": "/a097YNBiTFR9eryZPyIz2INwExz.jpg",
  "title": "The Gate of Heavenly Peace",
}
```

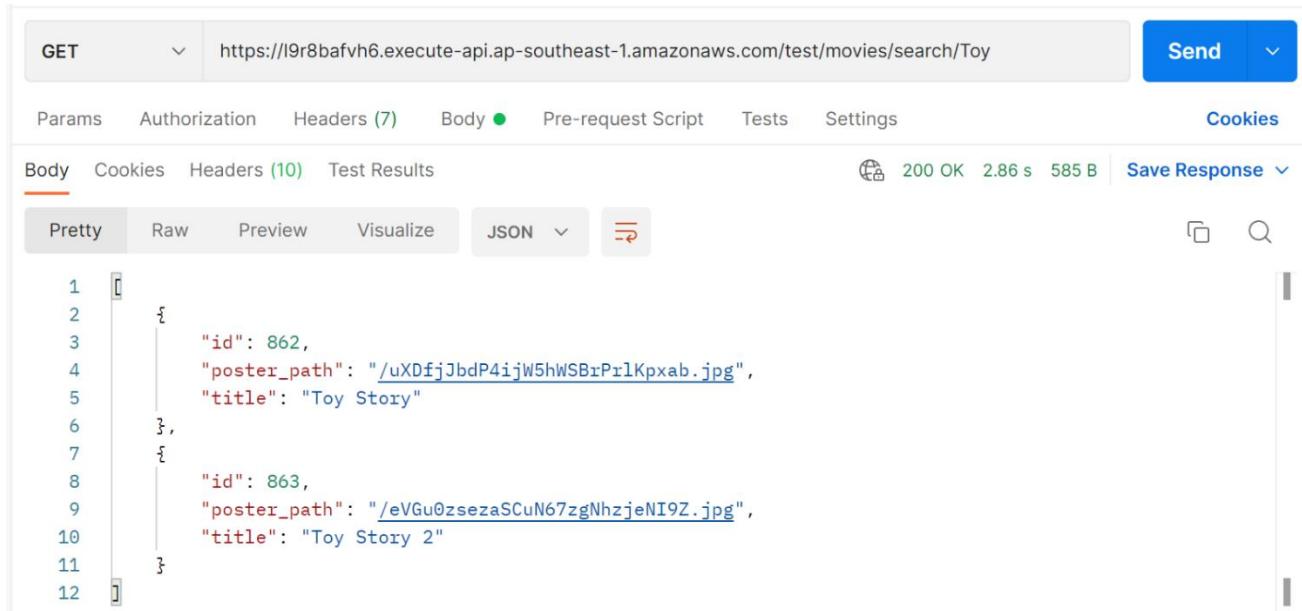
Figure 69: 10 highest rated films returned in JSON format

### 3. Search movies by title

URL: Invoke URL + /movies/search/{title}

Method: GET

Sample result:



The screenshot shows the Postman application interface. At the top, it displays a GET request to the URL <https://l9r8bafvh6.execute-api.ap-southeast-1.amazonaws.com/test/movies/search/Toy>. Below the URL, there are tabs for Params, Authorization, Headers (7), Body (green dot indicating JSON), Pre-request Script, Tests, Settings, and Cookies. The Body tab is selected, showing the following JSON response:

```
1
2
3
4
5
6
7
8
9
10
11
12
{
  "id": 862,
  "poster_path": "/uXDfjJbdP4ijW5hWSBrPr1Kpxab.jpg",
  "title": "Toy Story"
},
{
  "id": 863,
  "poster_path": "/eVGu0zsezaSCuN67zgNhzejNI9Z.jpg",
  "title": "Toy Story 2"
}
```

At the bottom right of the response area, it shows a status of 200 OK, a duration of 2.86 s, and a size of 585 B, with a 'Save Response' button.

Figure 70: A list of movie contains search keyword returned in JSON format

#### 4. Get movie information

URL: Invoke URL + /movie/{id}

Method: GET

Sample result:

```

1
2   "backdrop_path": "/mAQ0hJfI7Q1UPHhEqIRIAxwSpZM.jpg",
3   "genres": [
4     {
5       "id": 53,
6       "name": "Thriller"
7     },
8     {
9       "id": 80,
10      "name": "Crime"
11    }
12  ],
13  "id": 26174,
14  "overview": "Two U.S. Treasury (\\"T-men\\") agents go undercover in Detroit, and then Los Angeles, in an attempt to
break a U.S. currency counterfeiting ring.",
15  "poster_path": "/gmFk1BSdi38WDjJUq0mXRXe2qt.jpg",
16  "release_date": "1947-12-15",
17  "runtime": 92,
18  "tagline": "Terrific... and true!",
19  "title": "T-Men",
20  "rating": "3.733333492279053",
21  "views": 15,
22  "rating_times": 15
23

```

Figure 71: Movie details returned after sending a request

After calling:

	movie_id	rating	rating times	views
	26174	3.7333333...	15	16

Figure 72: Increment number of view after sending a request

## 5. Rate movies

URL: Invoke URL + /movies/rate

Method: POST

Request body sample:

```

1  {
2   ...
3   "movie_id": "49299",
4   ...
5   "user_id": "1",
6   ...
7   "rating": 4
8

```

Sample result:

- Before call:

	movie_id	rating	rating times	views
	49299	4.5	2	2

Figure 73: Current movie rate number before rating

- Calling:

The screenshot shows the Postman application interface. The request method is set to POST, and the URL is https://l9r8bafvh6.execute-api.ap-southeast-1.amazonaws.com/test/movies/rate. The 'Body' tab is selected, showing a JSON payload with fields: movie\_id: "49299", user\_id: "1", rating: 4. The response status is 200 OK, with a message "success".

Figure 74: A request send with the body to rate the movie

- After call:

	movie_id	rating	rating times	views
	49299	4.3333333...	3	2

	id	movie_id	rating	timestamp	user_id
	2	49299	4	166211969...	1

Figure 75: Current movie rate number after rating

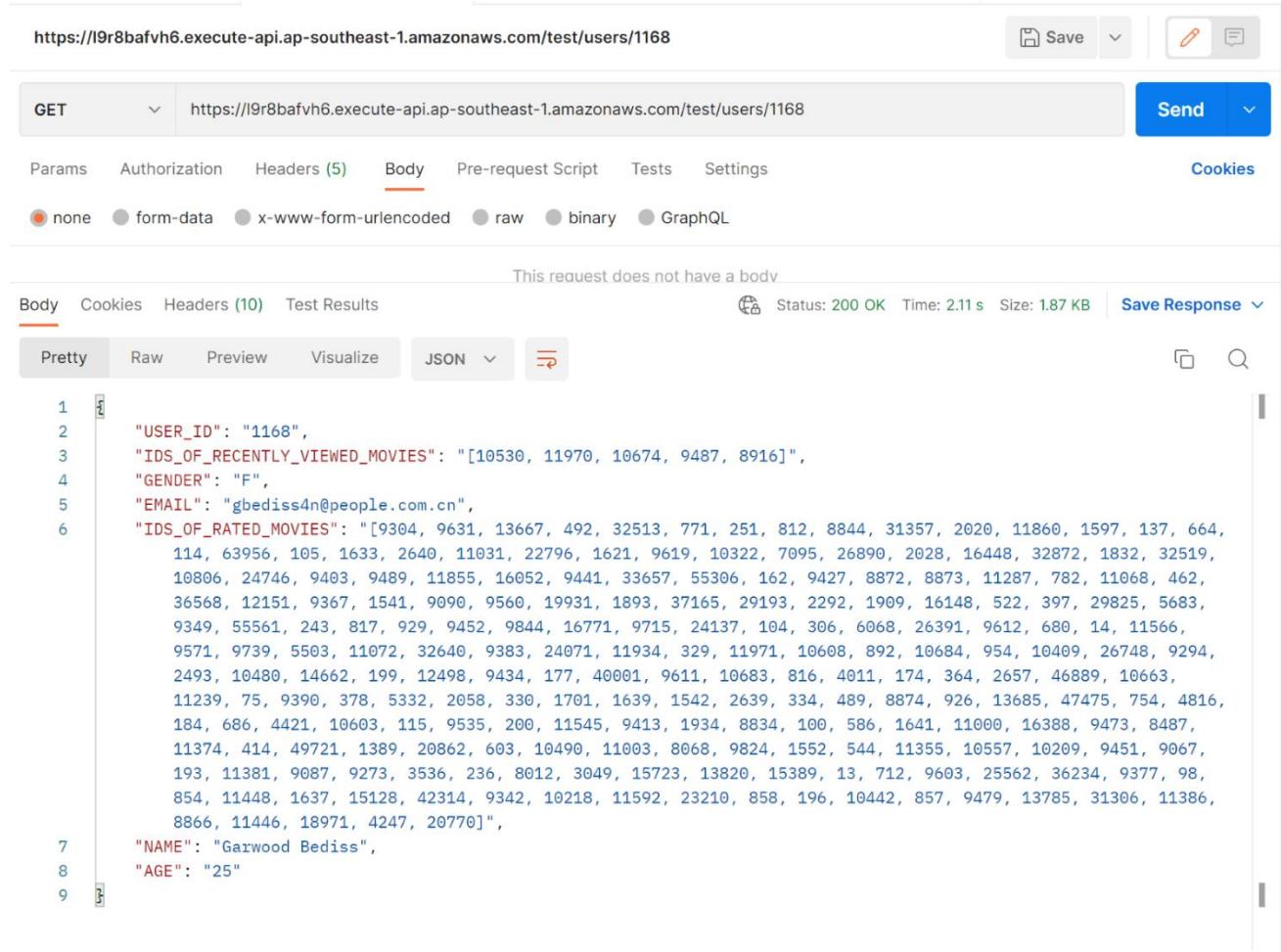
### 3. Sample query users result on Postman

#### 3.1 Get all information of an user on data base by id

URL: Invoke URL + /users/{id}

Method: GET

Sample result:



```

1 "USER_ID": "1168",
2 "IDS_OF_RECENTLY_VIEWED_MOVIES": "[10530, 11970, 10674, 9487, 8916]",
3 "GENDER": "F",
4 "EMAIL": "gbediss4n@people.com.cn",
5 "IDS_OF_RATED_MOVIES": "[9304, 9631, 13667, 492, 32513, 771, 251, 812, 8844, 31357, 2020, 11860, 1597, 137, 664,
114, 63956, 105, 1633, 2640, 11031, 22796, 1621, 9619, 10322, 7095, 26890, 2028, 16448, 32872, 1832, 32519,
10806, 24746, 9403, 9489, 11855, 16052, 9441, 33657, 55306, 162, 9427, 8872, 8873, 11287, 782, 11068, 462,
36568, 12151, 9367, 1541, 9090, 9560, 19931, 1893, 37165, 29193, 2292, 1909, 16148, 522, 397, 29825, 5683,
9349, 55561, 243, 817, 929, 9452, 9844, 16771, 9715, 24137, 104, 306, 6068, 26391, 9612, 680, 14, 11566,
9571, 9739, 5503, 11072, 32640, 9383, 24071, 11934, 329, 11971, 10608, 892, 10684, 954, 10409, 26748, 9294,
2493, 10480, 14662, 199, 12498, 9434, 177, 40001, 9611, 10683, 816, 4011, 174, 364, 2657, 46889, 10663,
11239, 75, 9390, 378, 5332, 2058, 330, 1701, 1639, 1542, 2639, 334, 489, 8874, 926, 13685, 47475, 754, 4816,
184, 686, 4421, 10603, 115, 9535, 200, 11545, 9413, 1934, 8834, 100, 586, 1641, 11000, 16388, 9473, 8487,
11374, 414, 49721, 1389, 20862, 603, 10490, 11003, 8068, 9824, 1552, 544, 11355, 10557, 10209, 9451, 9067,
193, 11381, 9087, 9273, 3536, 236, 8012, 3049, 15723, 13820, 15389, 13, 712, 9603, 25562, 36234, 9377, 98,
854, 11448, 1637, 15128, 42314, 9342, 10218, 11592, 23210, 858, 196, 10442, 857, 9479, 13785, 31306, 11386,
8866, 11446, 18971, 4247, 20770]",

7 "NAME": "Garwood Bediss",
8 "AGE": "25"
9

```

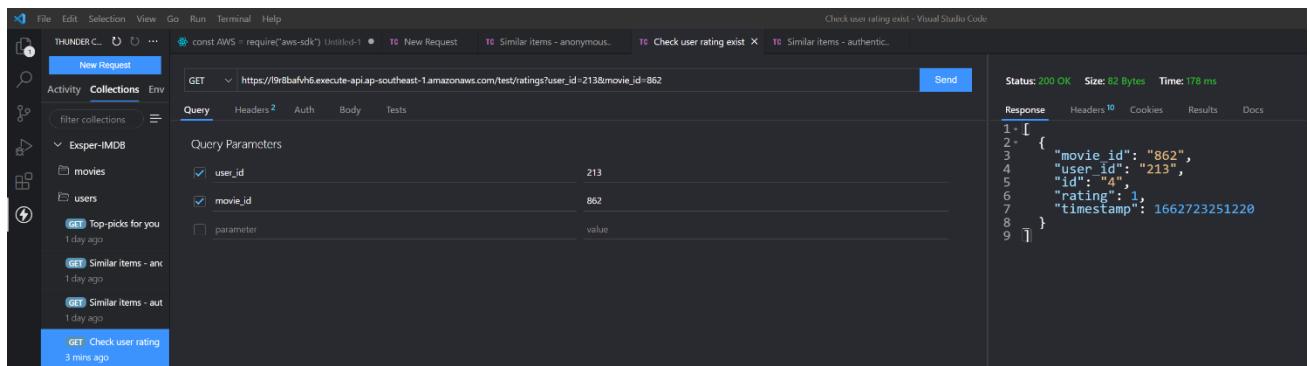
Figure 76: User details returned in JSON format

#### 3.2 Check if an user has rated a movie, then return the infomation if exist

URL: Invoke URL + /ratings?user\_id=213&movie\_id=862

Method: GET

Sample result:



The screenshot shows the THUNDER API testing tool interface. On the left, there's a sidebar with 'Activity' and 'Collections'. Under 'Collections', there's a section for 'Exper-IMDB' which contains 'movies' and 'users'. Below these are several recent requests: 'Top-picks for you' (1 day ago), 'Similar items - an' (1 day ago), 'Similar items - aut' (1 day ago), and 'Check user rating' (3 mins ago). The main area shows a request for 'Check user rating exist' with a status of 200 OK, size 62 bytes, and time 178 ms. The URL is https://0e8bafvh6.execute-api.ap-southeast-1.amazonaws.com/test/ratings?user\_id=213&movie\_id=862. The 'Query' tab is selected, showing parameters: user\_id (213) and movie\_id (862). The 'Response' tab shows the JSON output:

```
1: [
2:   {
3:     "movie_id": "862",
4:     "user_id": "213",
5:     "id": "4",
6:     "rating": 1,
7:     "timestamp": 1662723251220
8:   }
9: ]
```

Figure 77: Response indicate the rating timestamp of a certain user on a certain movie