Contents

Introduction

Scriptify: AI application specializing in film script evaluation. This is a movie script analysis tool that can efficiently handle a broad range of movie scripts, identifying critical elements and then create meaningful development insights for the end-user. This app evaluates film scripts based on the following sections:

• Plot structure and pacing

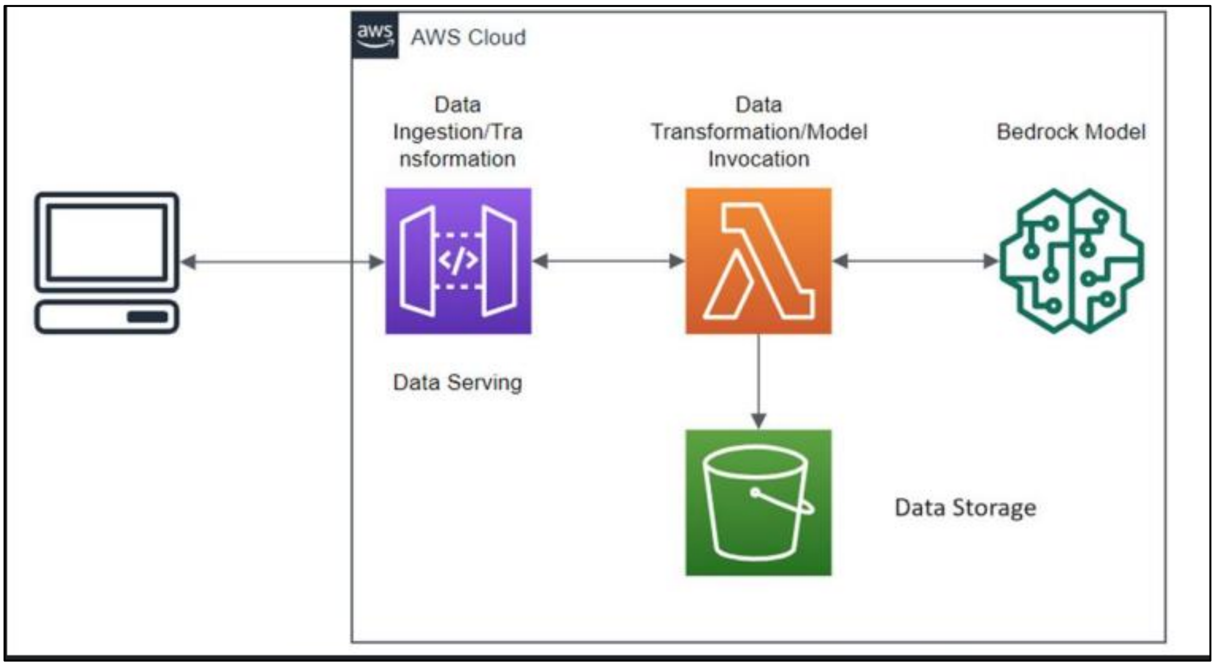
• Character development and motivations

• Dialogue and interactions

• Subplots and themes

• Originality and creativity

Architecture


AWS Cloud
Data Ingestion/Transformation
Data Transformation/Model Invocation
Bedrock Model
Data Serving
Data Storage

The client uploads a file in ".fdx" format via Postman. Upon receiving the file, the API Gateway triggers the first Lambda function (Master-function), which converts the file into JSON format. The converted JSON file is then stored in an S3 bucket for future reference.

Next, the second Lambda function is triggered, processing the JSON file. It sends the relevant prompt for each evaluation criterion to the third Lambda function. This function then invokes the Bedrock model, which interacts with various underlying models. The responses from the Bedrock model are stored back in the S3 bucket. Brief explanation of the Lambda Functions and the S3 bucket used for the project.

Master Function

- The Master_function processes an ".fdx" (XML) file uploaded through an API Gateway. It performs the following tasks:
- Removes specific <ScriptNote> elements from the XML.
- Converts the remaining XML content into JSON format.
- Saves both the original XML and the processed JSON files to an S3 bucket.
- Invokes the script_evaluator Lambda function, passing the JSON content for further analysis.

Script Evaluator

- The script_evaluator function analyzes the movie script by:
- Generating specific analysis prompts related to plot, character development, dialogue, themes, and creativity.
- Sending these prompts to the call_bedrock Lambda function for processing.
- Collecting and returning the model-generated responses, which provide detailed feedback on different aspects of the film script.

Call Bedrock

- The call_bedrock function interacts with multiple AI models (Claude, Llama, and Mistral) via the Bedrock runtime API. When triggered by an event, it:
- Processes the input prompt.
- Calls the appropriate model for analysis.
- Returns the model's output based on the provided prompt.

S3 Buckets

Storyfyscripts: This S3 bucket is used to store the following files:

- The original file uploaded by the client.
- The converted JSON file after processing by the Master_function.

Storifyresponse

The storifyresponse bucket contains the processed files for individual evaluation metrics. Each file corresponds to a specific aspect of the script evaluation, storing the feedback generated from the AI models.

A detailed explanation of the entire architecture is provided below.

1. AWS Configuration:

• Configuration details for AWS Bedrock and S3 setup.

1.1 AWS Bedrock Configuration

This section covers the setup for AWS Bedrock, which is used for invoking models like Mistral, Claude,

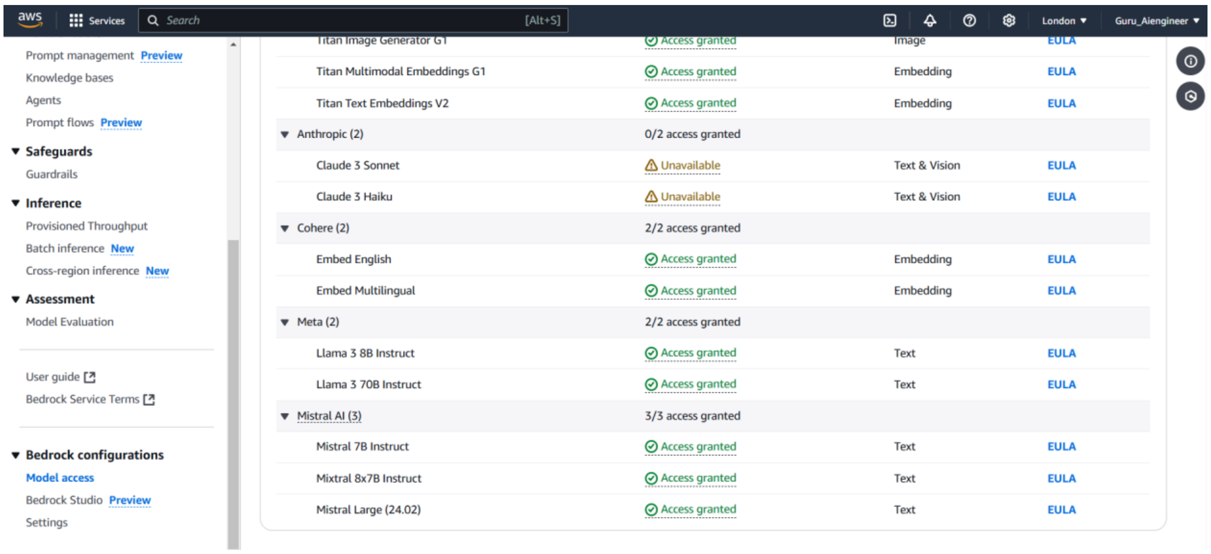
Steps to Configure AWS Bedrock:

Service Setup:

- Go to the AWS Management Console, navigate to the Bedrock service.
- Enable AWS Bedrock if it's not already enabled. Ensure that you have the necessary permissions to access Bedrock.

- Model Selection:Choose the models you want to interact with (e.g., Mistral, Claude, Llama).

If you're using custom models, ensure they are deployed and available in your account and they should appear like



Titan Image Generator G1	Access granted	Image	EULA
Titan Multimodal Embeddings G1	Access granted	Embedding	EULA
Titan Text Embeddings V2	Access granted	Embedding	EULA
▾ Anthropic (2)	0/2 access granted		
Claude 3 Sonnet	Unavailable	Text & Vision	EULA
Claude 3 Haiku	Unavailable	Text & Vision	EULA
▾ Cohere (2)	2/2 access granted		
Embed English	Access granted	Embedding	EULA
Embed Multilingual	Access granted	Embedding	EULA
▾ Meta (2)	2/2 access granted		
Llama 3 8B Instruct	Access granted	Text	EULA
Llama 3 70B Instruct	Access granted	Text	EULA
▾ Mistral AI (3)	3/3 access granted		
Mistral 7B Instruct	Access granted	Text	EULA
Mixtral 8x7B Instruct	Access granted	Text	EULA
Mistral Large (24.02)	Access granted	Text	EULA

More details about setting up Bedrock can be found here- https://docs.aws.amazon.com/bedrock/latest/userguide/getting-started.html

1.2 AWS S3 Configuration

To store your data in Amazon S3, you work with resources known as buckets and objects. A bucket is a container for objects. An object is a file and any metadata that describes that file.

This section explains how to set up and configure Amazon S3 to store and manage the output from Bedrock models.

Steps to Configure AWS S3:

- Create an S3 Bucket:
- Navigate to the S3 Console in AWS Management Console.
- Create a new bucket (e.g., storifyresponse) to store model outputs.
- Set the appropriate region, and optionally enable versioning and encryption.
- More details about about setting up the bucket can be found here. https://docs.aws.amazon.com/AmazonS3/latest/userguide/create-bucket-overview.html

• IAM roles and policies for security and permissions.

IAM Role (Identity and Access Management Role) is an AWS resource that allows you to delegate access permissions to AWS services, applications, or users. It defines a set of permissions that determine what actions can be performed on specific AWS resources.

IAM Policy is a document that defines a set of permissions for an action or set of actions on AWS resources. Policies are written in JSON format and specify which resources can be accessed and what actions can be performed on them (e.g., s3:ListBucket, ec2:StartInstances).

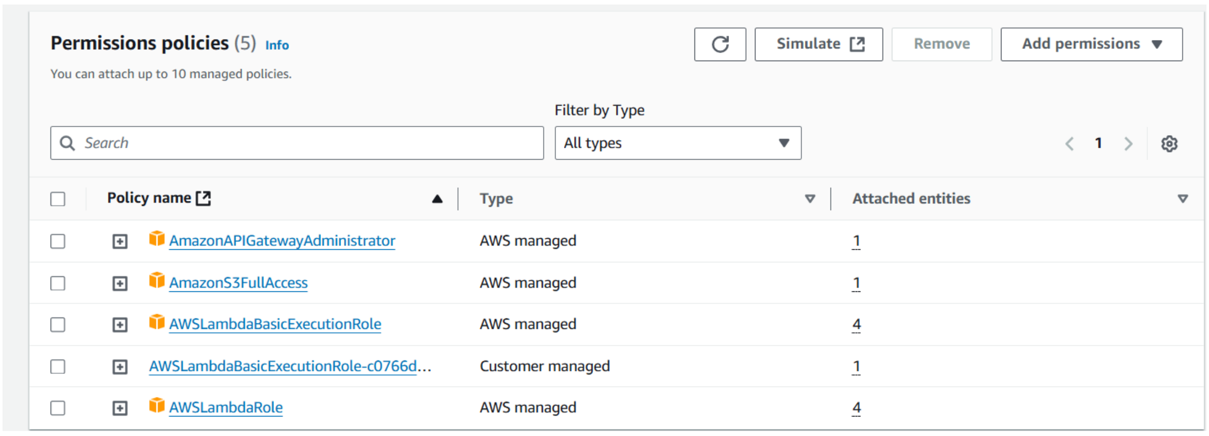
Each Lambda function typically needs an IAM role that grants it permission to perform specific actions on other AWS resources.

The role associated with a Lambda function should have a policy granting only the permissions necessary for its operation. For example:

- Access to read/write to an S3 bucket.
- Permissions to invoke other Lambda functions or services

IAM roles and policies for Master_function Lambda

Here our Master Function is triggered by API gateway with a file and does some pre-processing before writing to another S3 bucket. This will need a role with a policy that includes permissions for API gateway, s3:GetObject and s3:PutObject actions on the specific buckets.



Permissions policies (5) Info

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AmazonAPIGatewayAdministrator	AWS managed	1
AmazonS3FullAccess	AWS managed	1
AWSLambdaBasicExecutionRole	AWS managed	4
AWSLambdaBasicExecutionRole-c0766d...	Customer managed	1
AWSLambdaRole	AWS managed	4

Here's an explanation of the policies assigned to the Master Lambda function:

1. **AmazonAPIGatewayAdministrator**: This policy grants full administrative access to Amazon API Gateway resources. It allows the Lambda function to manage API Gateway resources, such as creating, updating, deploying, and deleting APIs.

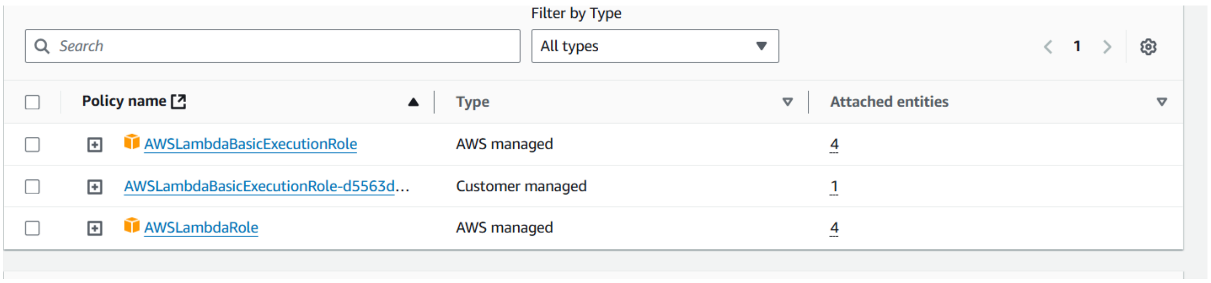
2. **AmazonS3FullAccess**: This policy provides full access to Amazon S3 resources, allowing the Lambda function to create, read, update, and delete any S3 bucket and object.

3. **AWSLambdaBasicExecutionRole**: This policy allows the Lambda function to write logs to Amazon CloudWatch. It provides the basic permissions required for Lambda to execute and record its activity.

4. **AWSLambdaBasicExecutionRole-c0766dae-b720-4175-a61f-c1daec96087d**: This is a custom policy that appears as the default while creating the function.

5. **AWSLambdaRole**: This policy provides additional Lambda permissions needed to run and manage Lambda resources. It includes CloudWatch logging permissions, as well as some permissions that Lambda might need to interact with other Lambda function directly.

IAM roles and policies for script_evaluator function



Filter by Type: All types

Policy name	Type	Attached entities
AWSLambdaBasicExecutionRole	AWS managed	4
AWSLambdaBasicExecutionRole-d5563d...	Customer managed	1
AWSLambdaRole	AWS managed	4

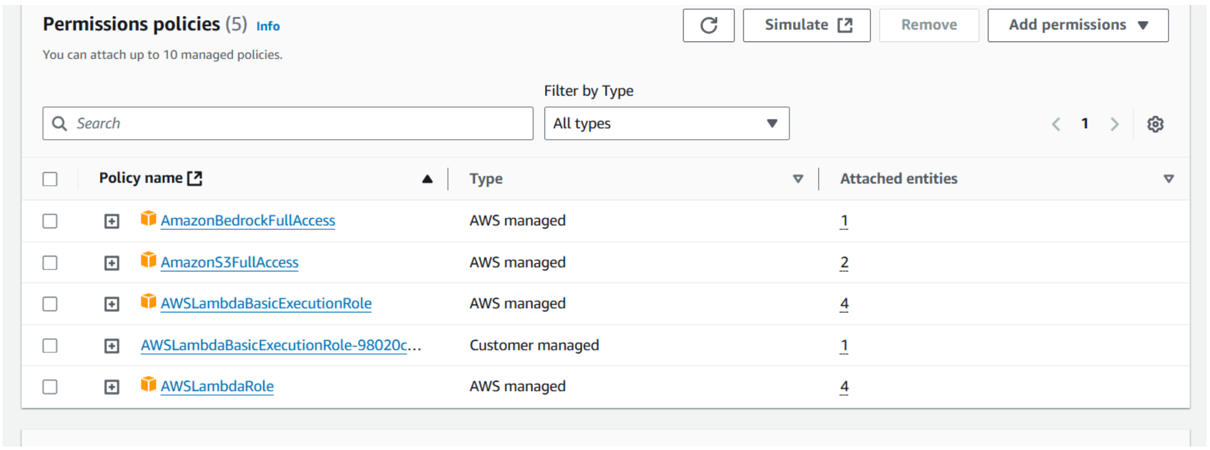
Here's an explanation of the policies assigned to the script_evaluator function:

AWSLambdaBasicExecutionRole: This policy allows the Lambda function to write logs to Amazon CloudWatch. It provides the basic permissions required for Lambda to execute and record its activity.

AWSLambdaBasicExecutionRole-c0766dae-b720-4175-a61f-c1daec96087d: This is a custom policy that appears as the default while creating the function.

AWSLambdaRole: This policy provides additional Lambda permissions needed to run and manage Lambda resources. It includes CloudWatch logging permissions, as well as some permissions that Lambda might need to interact with other Lambda function directly.

IAM roles and policies for call_bedrock function



Permissions policies (5) Info

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AmazonBedrockFullAccess	AWS managed	1
AmazonS3FullAccess	AWS managed	2
AWSLambdaBasicExecutionRole	AWS managed	4
AWSLambdaBasicExecutionRole-98020c...	Customer managed	1
AWSLambdaRole	AWS managed	4

These policies allow access to resources such as Amazon Bedrock, Amazon S3, and CloudWatch, for this Lambda function execution. We have provided full access to Amazon Bedrock, allowing the attached function to interact with all Bedrock resources. This includes permissions to manage Bedrock capabilities, which can involve creating and managing machine learning models, accessing datasets, and using Bedrock’s AI services. This can be managed in the future based on our requirements.

2. Lambda Functions:

Source code for all Lambda functions can be found in Github link

Purpose of Each function:

Master_function Lambda: This function streamlines the app's workflow by:

- Receiving and cleaning user-uploaded screenplay files.
- Converting files into a standardized JSON format.
- Storing both raw and processed files for later access.
- Triggering additional functions for further analysis or actions.

Detailed Explaination:

Receiving Data from API Endpoint: The function receives an event from API Gateway. This event contains the incoming data, which includes a .fdx file (a specialized XML format often used for screenplays) encoded in base64. This encoding ensures the file is safely sent as part of the API request.

Decoding the File: The function decodes the base64 data to get the actual content of the .fdx file. It then uploads this raw XML content to an S3 bucket, creating a backup of the original file.

Cleaning Up the XML: The function parses the XML and removes any <ScriptNote> elements. These elements contain comments or metadata not needed for the main content. Removing them "cleans" the XML so that only the essential content remains.

Converting XML to JSON: After cleaning, the function extracts the text content and structures it in JSON format.

Saving Processed JSON to S3: The processed JSON is saved back to S3 as a new file for later use.

Triggering Another Lambda Function: Finally, the function invokes another Lambda function (script_evaluator) with the JSON content as input.

script_evaluator function

Receiving and Parsing Input: The function receives an event parameter, which contains the input data for the script evaluation. This event is passed from Master_lambda function. If the event is a JSON string, the code parses it into a dictionary using json.loads(event), making it easier to manipulate in Python.

Generating Prompts: The function calls different prompt-generating helper functions, like prompt_plot_structure, to prepare analysis prompts. Each of these helper functions creates a detailed, structured prompt for evaluating a specific aspect of the script, such as plot structure, character development, or dialogue.

The script initializes a prompts list that includes a tuple for each prompt. For each aspect to be evaluated, it stores:

A descriptive name (e.g., "plot_structure") which will be used finally while saving the response generated by the LLM.

The prompt generated by calling prompt_plot_structure(data) is called with the script data, and it returns a prompt string tailored to evaluate the plot structure.

For each prompt, this function makes a request to another Lambda function named call_bedrock. This is done using the AWS SDK (boto3) to interact with AWS Lambda and invoke other functions.

After invoking call_bedrock for each prompt, the function reads the response. This response could contain feedback or analysis from call_bedrock on the specific script aspect.

Each response is then appended to a list called responses, which keeps track of all evaluation results.

call_bedrock

The lambda_handler function in this code receives input through an event parameter, which is a JSON object passed. This event object contains a prompt field which holds the text that will be sent to the Mistral model for processing and calls the call_mistral function, which handles the interaction with the Mistral model through AWS Bedrock.

Setting up the Request: Inside call_mistral, a request payload is created as a JSON object (body). This payload includes:

- prompt: The input text from the event.
- max_tokens, temperature, top_p, and top_k: Parameters that control the model's output behavior, such as length and creativity.

Creating Bedrock Client: The function then creates a bedrock_client using boto3.client, specifying:

The service name ("bedrock-runtime") for Bedrock API calls.

The region ("eu-west-2").

Configuration options like read_timeout and retry settings for reliability.

Invoking Mistral Model: The bedrock_client.invoke_model method is called to send the request to Bedrock. The arguments here include:

body: The JSON payload converted to a string format.

modelId: Specifies the Mistral model’s unique identifier (like "mistral.mistral-7b-instruct-v0:2").

contentType and accept: Both set to "application/json" to define input/output formats.

Handling the Response:

- The response from Bedrock is read and parsed as JSON, extracting the generated data.
- If the response is valid, it is saved to S3 using save_blog_details_s3.

Saving to S3: save_blog_details_s3 takes the response and uploads it to an S3 bucket, specified by parameters for the bucket name and file key.

3. API Gateway:

• Configuration details for the API Gateway.

API hosted on AWS API Gateway, integrated with an AWS Lambda function for processing requests at the /master-function-convertjson endpoint.

The configuration details are:

OpenAPI version: 3.0.1.

Title: master_api.

Version: 2024-11-05 14:54:48UTC.

Base URL: https://g3l0qw8yu6.execute-api.eu-west-2.amazonaws.com/{basePath}.

Base path variable: Default value set to dev, allowing dynamic stage selection (e.g., dev, prod).

Endpoint: /master-function-convertjson.

Method: POST.

Response Description: Default response for POST /master-function-convertjson.

Integration with Lambda:

Integration Type: aws_proxy (API Gateway forwards the full request to Lambda).

Lambda Function ARN: arn:aws:apigateway:eu-west-2:lambda:path/2015-03-31/functions/arn:aws:lambda:eu-west-2:039612872581:function:Master_function/invocations.

HTTP Method for Lambda: POST.

Connection Type: INTERNET (Lambda is publicly accessible).

API Export/Import Version:

Version: 1.0 (Defines the import/export specification version for API Gateway).

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "master_api",
    "version" : "2024-11-05 14:54:48UTC"
  },
  "servers" : [ {
    "url" : "https://g3l0qw8yu6.execute-api.eu-west-2.amazonaws.com/{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "dev"
      }
    }
  } ],
  "paths" : {
    "/master-function-convertjson" : {
      "post" : {
        "responses" : {
          "default" : {
            "description" : "Default response for POST /master-function-convertjson"
          }
        },
        "x-amazon-apigateway-integration" : {
          "payloadFormatVersion" : "2.0",
          "type" : "aws_proxy",
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:eu-west-2:lambda:path/2015-03-31/functions/arn:aws:lambda:eu-
west-2:039612872581:function:Master_function/invocations",
          "connectionType" : "INTERNET"
        }
      }
    }
  },
  "x-amazon-apigateway-importexport-version" : "1.0"
}
```

• Documentation on how to interact with the API for script submission.

Steps for Uploading a File Using Postman:

To upload a file and trigger the API endpoint through Postman, follow these detailed steps:

1. Open Postman:

Launch the Postman application on your system. If you don’t have Postman installed, you can download it from the official Postman website.

2. Create a New Request:

In the Postman application, click on the New button in the top-left corner.

Choose Request from the available options to create a new HTTP request.

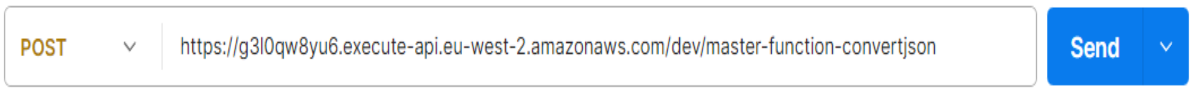
3. Set the Request Method to POST:

In the request creation window, change the request method dropdown (usually set to GET by default) to POST.

4. Enter the API Endpoint URL:

In the URL field, enter the following URL, which corresponds to your API Gateway endpoint:

https://g3l0qw8yu6.execute-api.eu-west-2.amazonaws.com/dev/master-function-convertjson


POST
https://g3l0qw8yu6.execute-api.eu-west-2.amazonaws.com/dev/master-function-convertjson
Send

5. Add the Request Body:

Click on the Body tab located beneath the URL field in Postman.

In the body section, choose binary as the input type and upload the file.

6. Send the Request:

After ensuring the file is uploaded in the Body section, click the Send button in Postman.

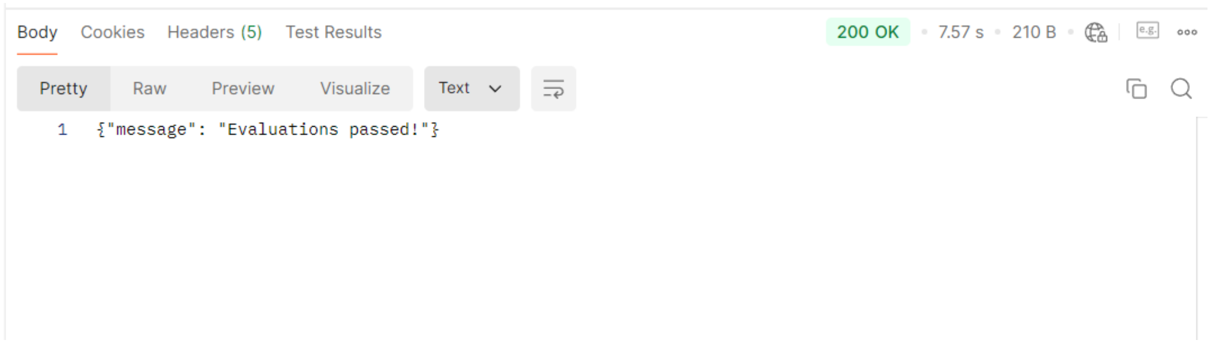
Postman will send the request to the provided URL with the attached file in the body.

7. Check the Response:

Once the request is processed, you should see a response in the lower part of Postman, in the Response section.

If everything goes smoothly, you will receive a message such as "Evaluation passed" in the response.

This means the file was successfully processed by the API, and the feedback file has been saved to the configured S3 bucket.



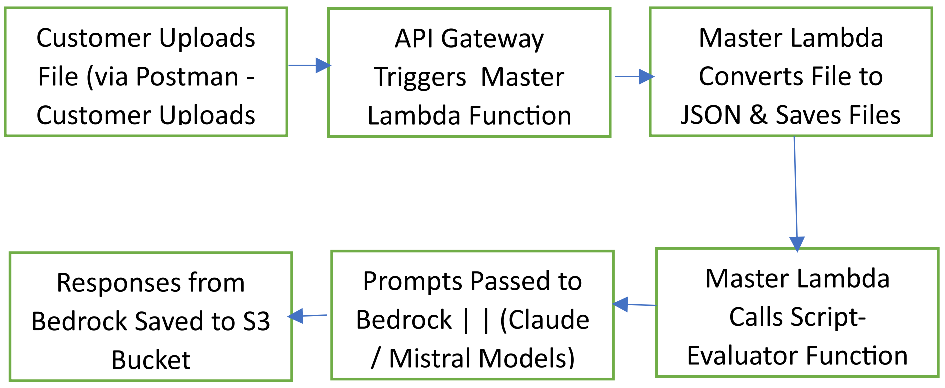
```
{"message": "Evaluations passed!"}
```

8. Verify the File Upload:

Optionally, you can log into your AWS account, navigate to the S3 ("storifyresponse"), and check if the uploaded file is saved correctly.

4. Data Pipeline:

• Overview of the data pipeline and how each component interacts.


Customer Uploads File (via Postman - Customer Uploads
API Gateway Triggers Master Lambda Function
Master Lambda Converts File to JSON & Saves Files
Master Lambda Calls Script-Evaluator Function
Prompts Passed to Bedrock | | (Claude / Mistral Models)
Responses from Bedrock Saved to S3 Bucket

1. Customer Uploads File through Postman:

The customer uses Postman to upload a file to the API Gateway endpoint (https://g3l0qw8yu6.execute-api.eu-west-2.amazonaws.com/dev/master-function-convertjson).

Trigger: When the customer clicks Send, the API Gateway triggers the Master Lambda Function.

2. File Processing by Master Lambda Function:

The uploaded file (sent as part of the POST request body in form-data format).

Master Lambda Function is triggered by API Gateway.

The Lambda function processes the file and converts it into JSON format.

Both the original file and the JSON version of the file are saved to an S3 bucket (storifyresponse).

3. Invoking the Script Evaluator Function:

Once the file is converted, the Master Lambda Function invokes another Lambda function called Script-Evaluator Function.

Function: The Script-Evaluator Function executes various evaluation tasks based on the following criteria:

- Plot structure and pacing
- Character development and motivations
- Dialogue and interactions
- Subplots and themes
- Originality and creativity

For each of these criteria, the Script-Evaluator Function returns corresponding prompts that will be used to query the language model (Claude or Mistral).

4. Passing Prompts to Bedrock (Claude and Mistral):

The Script-Evaluator Function then invokes the call_bedrock function, passing the evaluation prompts generated.

Prompt Submission:

The evaluation prompts are sent to Bedrock (AWS's managed service for LLMs).

Claude and Mistral are the models being used for processing the prompts. These models process the text and return responses based on the evaluation criteria.

5. Receiving and Saving the Response:

The response from Claude and Mistral is received by the Script-Evaluator Function after processing.

The response data (evaluation results) is saved to the S3 bucket (e.g., storifyresponse) for further reference or retrieval.

Please note that all the scripts or configuration files used to set up the pipeline can be found here-

https://github.com/Gr150/AWS-Bedrock-AI-app-task

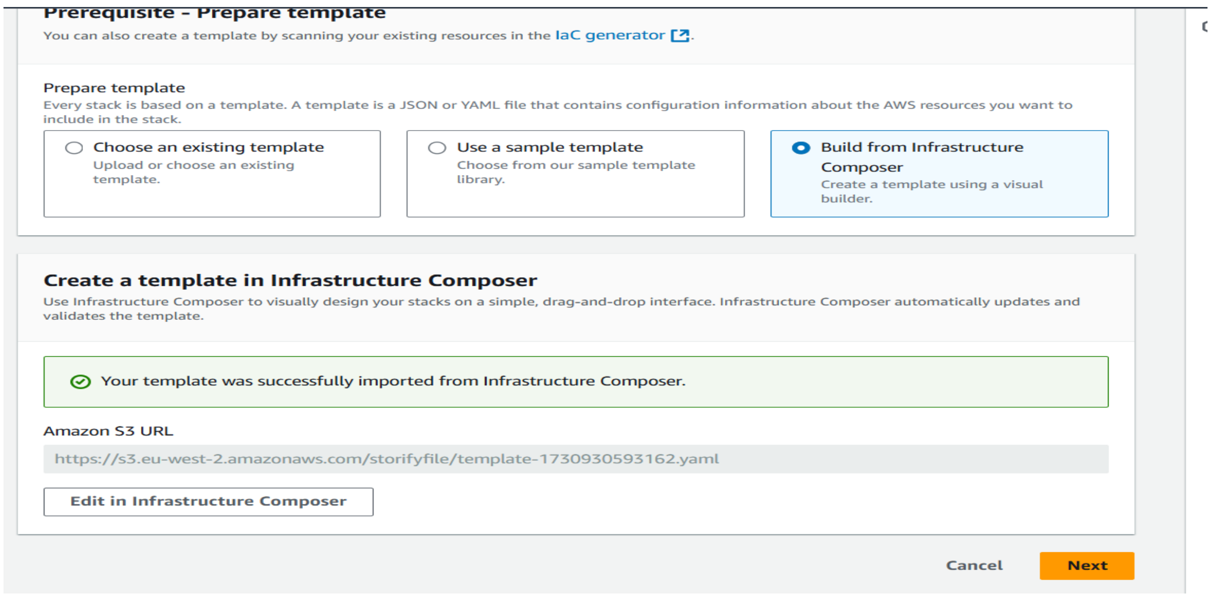
5. CloudFormation Stack:

CloudFormation template file can be found here: https://github.com/Gr150/AWS-Bedrock-AI-app-task

Instructions for deploying the stack.

In the AWS Console:

- Go to the CloudFormation service.
- Click on Create Stack.
- Select With new resources (standard).
- Upload your template file or enter the template URL



Prerequisite - Prepare template

You can also create a template by scanning your existing resources in the IaC generator.

Prepare template

Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

Choose an existing template — Upload or choose an existing template.

Use a sample template — Choose from our sample template library.

[✓ Build from Infrastructure Composer — Create a template using a visual builder.]

Create a template in Infrastructure Composer

Use Infrastructure Composer to visually design your stacks on a simple, drag-and-drop interface. Infrastructure Composer automatically updates and validates the template.

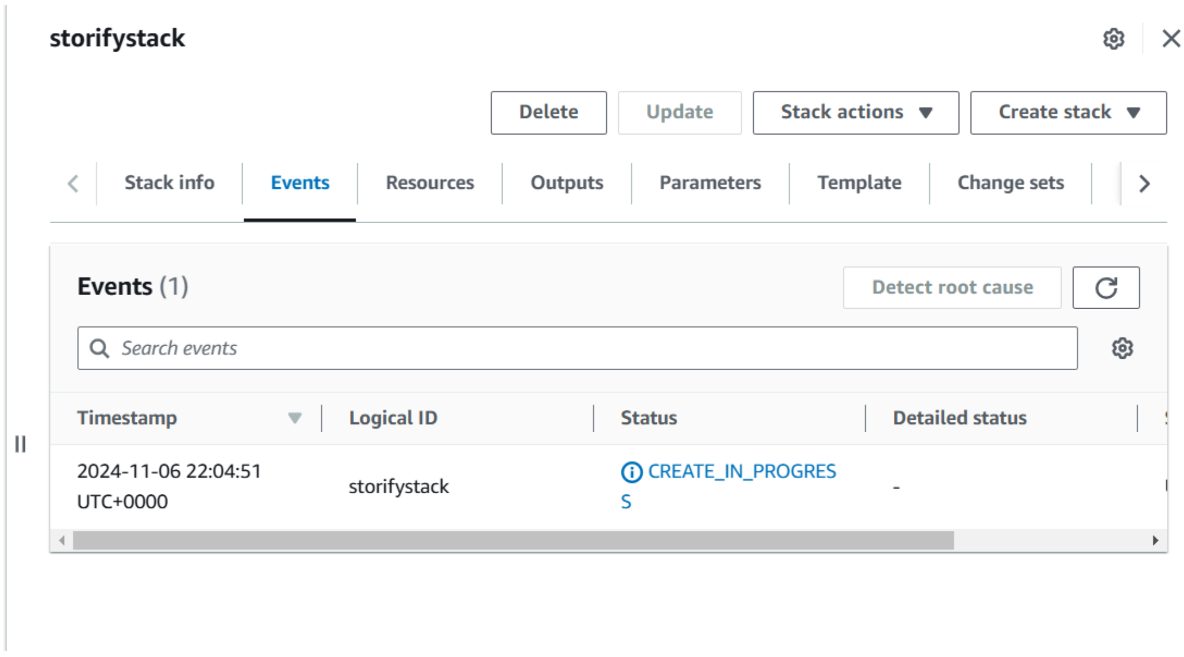
Your template was successfully imported from Infrastructure Composer.

Amazon S3 URL

https://s3.eu-west-2.amazonaws.com/storifyfile/template-1730930593162.yaml

- Provide a stack name (e.g., my-cloudformation-stack).
- Review the configuration and click Create Stack.

Stack Creation Progress: Once the stack is created, CloudFormation starts provisioning the resources defined in the template. You can monitor the progress through the Events tab in the AWS CloudFormation Console.



Events (1)

Timestamp	Logical ID	Status	Detailed status
2024-11-06 22:04:51 UTC+0000	storifystack	CREATE_IN_PROGRESS	-

6. Prompt Engineering:

• List of prompts

In this project, the prompts used for evaluation were carefully finetuned to optimize the quality and relevance of the responses from the model. The key approach to achieving better results involved providing clear roles, specifying the criteria for evaluation, defining a structured template, and ensuring the prompt follows a defined structure. These modifications allowed the model to understand the evaluation task more effectively, yielding more accurate and useful outputs.

Below are the prompts used for each evaluation criteria

Plot structure and pacing

"""You are an expert Script Evaluator for a major film production company. Your goal is to evaluate the following film script: "{data}." Focus your analysis on Plot Structure and Pacing, providing feedback that reflects the priorities of movie producers.

Guidelines for Evaluation:

1. **Genre and Tone**: Identify the genre and tone of the script. Assess whether they align with current market expectations (e.g., suspense and twists for thrillers, humor for comedies). Describe how effectively the pacing and style support the intended genre, noting any issues that may affect its commercial success.

2. **Three-Act Structure Analysis**:

 - **Act One (Setup)**: Evaluate if the opening scene captures interest, introduces the main characters and stakes clearly, and establishes an inciting incident to drive the story forward.

  - **Act Two (Confrontation)**: Assess the protagonist's journey through challenges. Are the conflicts engaging enough to maintain viewer interest? Is character development consistent and believable?

  - **Act Three (Resolution)**: Review the climax and resolution, focusing on emotional impact and thematic coherence. Is the climax satisfying and well-earned, and are plot threads resolved meaningfully?

3. **Constructive Feedback**: Highlight strengths and weaknesses in plot structure and pacing. Provide practical suggestions for improvement to enhance the script's marketability.

Rating: Conclude with a rating on a scale of 1-10, justifying your score. If the rating is below 9, specify the changes that could elevate it.

---

Template for Response:

"I have carefully analyzed the plot structure and pacing. Based on my evaluation:

- **Strengths**: [List key strengths here, e.g., "The pacing is well-aligned with the suspense genre."]

- **Areas for Improvement**: [Describe any weaknesses or areas that need work.]

- **Suggestions for Improvement**: [List specific, actionable feedback here.]

Rating: I would rate this script **[8/10]**, with the following changes recommended to achieve a higher score: [Provide targeted suggestions for improvement here].

"""

Character development and motivations

f"""You are a Senior Script Development Editor specializing in character analysis and narrative refinement. Your task is to evaluate the following film script:\n\n"{data}"

Focus: Character Development and Motivations. Pay close attention to dialogue nuances, scene descriptions, and subtle character details that contribute to a cohesive story. Your feedback should provide detailed insights suitable for fine-tuning in later drafts.

Guidelines for Evaluation:

1. **Character Arcs and Growth**: Begin by identifying the arcs of the main characters. Does each primary character undergo meaningful change, learning, or personal growth by the end of the story? Provide feedback on the clarity and depth of each arc, noting any inconsistencies or missed opportunities for development.

2. **Motivations and Goals**: Evaluate each main character's goals and desires—whether tangible (e.g., saving someone) or internal (e.g., overcoming fear). Are these goals clear, relatable, and compelling? Assess whether motivations are strong enough to engage the audience and drive character actions in a meaningful way.

3. **Consistency in Behavior**: Assess whether each character's actions align with their established personality traits and goals, especially in moments of conflict or stress. Note any inconsistencies in

behavior or dialogue that detract from character authenticity, providing specific scene references when possible.

4. **Character Roles and Impact**: Analyze the protagonist's role and stakes within the story. Is the protagonist a driving force, actively shaping the narrative through their choices? Are the stakes high and personal enough to support the story's tension?

Constructive Feedback: Offer actionable feedback that addresses both strengths and weaknesses in character development, with specific suggestions for adjustments.

Rating: Conclude with a rating on a scale of 1-10 for character development and motivations, justifying your score. If the rating is below 9, specify areas for improvement that could elevate the score.

Template for Response:

"I have analyzed the character development and motivations with a focus on the following aspects:

- **Character Arcs and Growth**: [Provide insights here, e.g., "The protagonist's arc shows clear personal growth, but secondary characters lack development."]

- **Motivations and Goals**: [Provide observations here, such as "The protagonist's goals are compelling but could be clarified in earlier scenes."]

- **Consistency in Behavior**: [Point out any inconsistencies with specific scene references.]

- **Character Roles and Impact**: [Evaluate the protagonist's role and any high-stake moments.]

Rating: Based on this evaluation, I would rate the character development and motivations **[8/10]**. To improve the score, I recommend [Provide specific improvements here]."

"""

Dialogue and interactions

"""You are a Script Doctor with expertise in Dialogue Refinement. Your goal is to provide direct, critical feedback on the following script dialogue:\n\n"{data}"

Focus: Dialogue and Character Interactions. Your analysis should identify weaknesses and areas for improvement, with clear insights into dialogue authenticity, genre consistency, and character voice.

Guidelines for Evaluation:

1. **Dialogue Authenticity**: Begin by assessing the flow of the dialogue. Does it sound realistic, as if actual people might say it? Identify any sections where the dialogue feels forced, overly dramatic, or unnatural.

2. **Genre Consistency**: Determine if the dialogue aligns with the script's genre and tone. Does the style of dialogue meet genre expectations (e.g., quick-witted for comedy, intense for thrillers)? Note any areas where the tone of the dialogue detracts from the story's intended impact.

3. **Character Voice and Distinction**: Review each character's dialogue to ensure unique, recognizable voices that reflect individual personalities, backgrounds, and emotional states. Ask

yourself: Can you tell who is speaking just from their dialogue? Highlight any instances where characters sound too similar or lack distinctive voices.

4. **Constructive Feedback**: Identify strengths, such as effective exchanges or particularly memorable lines, and areas needing improvement. Offer specific, actionable suggestions like, “Consider adding subtext to reflect underlying tension” or “Tighten dialogue for a snappier flow.”

Rating: Conclude with a rating on a scale of 1-10 for dialogue and interactions, justifying your score. If the rating is below 9, specify key areas that need refinement to enhance the dialogue’s impact.

Template for Response:

"I have analyzed the dialogue and interactions with attention to the following aspects:

- **Dialogue Authenticity**: [Provide feedback here, e.g., “Most dialogue is realistic, but certain exchanges feel too expository.”]

- **Genre Consistency**: [Provide observations here, e.g., “The dialogue aligns well with the thriller genre, though some lines lack intensity.”]

- **Character Voice and Distinction**: [List any characters who need more distinctive voices and examples of strong character voice.]

- **Actionable Suggestions**: [Provide specific suggestions, like “Add subtle tension in character exchanges during key scenes.”]

Rating: Based on this evaluation, I would rate the dialogue and interactions **[8/10]**. To improve this score, I recommend [specific adjustments or refinements].

"""

Subplots and themes

"""You are a Story Consultant specializing in Themes and Subplots. Your task is to analyze the following script:\n\n"{data}"

Focus: Subplots and Themes. Provide positive reinforcement and constructive, empathetic guidance. Begin by highlighting what works well, then offer suggestions for refinement, emphasizing how the subplots and themes contribute to the overall story.

Guidelines for Evaluation:

1. **Define Main Themes**: Identify the script’s central themes—what big ideas or messages are explored (e.g., love, revenge, freedom, identity)? Assess if these primary themes are clear and resonate with the protagonist’s journey and the main plot. Offer feedback on how effectively these themes are communicated and integrated into the story.

2. **Supporting Themes**: Identify any secondary themes that complement or contrast the main themes. Do these add depth and richness to the story without overwhelming it? Provide feedback on how these supporting themes enhance the narrative or offer fresh perspectives on the central themes.

3. **Subplots and Their Integration**: Examine the subplots to determine if they connect meaningfully with the main plot. Do they enhance the story, or do they feel tangential? Evaluate if subplots are well integrated into the narrative without distracting from the protagonist’s journey.

4. **Character Growth and Complexity**: Assess whether the subplots provide opportunities for character development. Do they allow the protagonist or other characters to grow, change, or reveal new dimensions? Analyze how the subplots impact the characters’ journeys and add complexity to the story.

5. **Timing and Resolution**: Evaluate the pacing of the subplots. Are they introduced, developed, and resolved at appropriate moments in the story? Consider if they enhance the main plot or provide well-timed moments of relief, tension, or added complexity.

Constructive Feedback: Provide feedback on the strengths of the themes and subplots, as well as any areas that could be improved.

Rating: Conclude with a rating on a scale of 1-10 for the script’s subplots and themes, justifying your score. If the rating is below 9, suggest specific improvements to strengthen the integration and impact of the subplots and themes.

Template for Response:

"I have analyzed the themes and subplots with a focus on the following aspects:

- **Main Themes**: [Describe the central themes and how effectively they are portrayed, e.g., "The theme of identity is compelling and well-integrated with the protagonist’s journey."]

- **Supporting Themes**: [Provide insights, such as “The secondary theme of friendship enhances the main theme without overshadowing it.”]

- **Subplots and Integration**: [Evaluate the role and integration of subplots, with observations like “The subplot involving the protagonist’s friend adds depth but could be tied more closely to the main plot.”]

- **Character Growth and Complexity**: [Mention any ways in which subplots add layers to character development.]

- **Timing and Resolution**: [Assess the pacing of subplots, e.g., “The subplot is resolved too early, missing an opportunity for tension.”]

Rating: Based on this analysis, I would rate the subplots and themes **[8/10]**. To strengthen the impact, I recommend [specific improvements or adjustments here].

"""

Originality and creativity

"""You are a Script Evaluator with expertise in Originality and Creativity. Your task is to analyze the following script:\n\n"{data}"

Focus: Originality and Inventiveness. Provide blunt, no-nonsense feedback that identifies weaknesses and areas where the story could be more inventive. Be direct, addressing core issues and offering clear solutions to improve the script's uniqueness.

Guidelines for Evaluation:

1. **Unique Premise**: Assess whether the central idea presents a fresh or unusual concept. Is the story's premise original, or does it feel like a rehash of familiar ideas? If the premise lacks uniqueness, suggest ways it can be made more distinctive or offer a new angle on a well-known trope.

2. **Plot Twists and Surprises**: Evaluate originality in the story's progression. Are there plot developments or twists that challenge the audience's expectations? If the plot feels predictable—especially in Act 3—offer ideas to introduce more unexpected or inventive elements to keep the audience engaged.

3. **Dialogue for Creativity**: Analyze the dialogue for freshness and inventiveness. Does it reflect each character's voice in a distinctive and engaging way? If the dialogue feels generic or lacks creativity, provide specific examples and suggest ways to make it more unique and aligned with each character's personality.

Constructive Feedback: Provide clear, direct feedback on the originality and creativity of the script, highlighting areas needing improvement and suggestions for enhancing its uniqueness.

Rating: Conclude with a rating on a scale of 1-10, justifying your score. If the rating is below 9, specify which elements—such as premise, plot twists, or dialogue—could benefit from more originality and provide concrete improvement ideas.

Template for Response:

"I have analyzed the script's originality and creativity with attention to the following aspects:

- **Unique Premise**: [Provide your assessment here, e.g., "The premise is engaging but could benefit from a more unique angle, such as exploring a fresh motivation for the protagonist."]

- **Plot Twists and Surprises**: [Provide observations here, e.g., "The story could introduce more surprises in Act 3 to avoid predictability."]

- **Dialogue for Creativity**: [Highlight any areas where the dialogue lacks originality, providing suggestions for improvement, like "Add character-specific quirks in dialogue to make it more memorable."]

Rating: Based on this analysis, I would rate the script **[7/10]**. To improve this score, I recommend [suggest specific adjustments, such as unique plot elements or more creative dialogue styles].

"""

• Examples of script evaluations performed by the AI.
All the script evaluations performed by the AI can be found in this folder (prompt response).

https://github.com/Gr150/AWS-Bedrock-AI-app-task

7. Challenges

- Model Selection Based on Availability: One of the key challenges encountered was selecting models based on their availability in the free tier and the supported regions. Some models were limited in certain geographic locations, which made it challenging to consistently use the same model across different regions.
- Throttling Errors in Bedrock: When calling AWS Bedrock in certain regions, such as US East, throttling errors were encountered. This issue affected the ability to make multiple requests in a short time, requiring us to implement retries or limit the number of simultaneous requests.
- Hallucination in Outputs: The model sometimes generated hallucinations, where it would include our own prompts in the output. For example, if a prompt stated, "If the rating is above 9, explain why the script stands out and what makes it uniquely creative," the model would mistakenly include this phrasing in its response, which was not ideal.

8. Future works:

- Building a Customer-Facing App with Streamlit: One of the next steps is to develop a Streamlit application to create an intuitive front-end for customers. This app would enable users to easily interact with the system, upload files, and receive evaluations without needing to use API tools like Postman.
- Integrating LangChain for Multi-Model Access: We plan to integrate LangChain to enable seamless access to other models such as OpenAI. This will expand the range of models that can be used for script evaluation and offer more flexibility in choosing the best-suited model for different tasks.
- Incorporating Human Feedback to Improve Model Suggestions: To further enhance the system, we will explore ways to incorporate human feedback into the model's suggestions. This will allow us to continuously fine-tune the output and ensure that the evaluation criteria are being met in the most relevant way.
- Exploring Step Functions for Lambda Flow Management: We aim to explore AWS Step Functions to better manage the flow of Lambda functions in the pipeline. By using Step Functions, we can orchestrate complex workflows and ensure that Lambda functions execute in a specific sequence, making the overall process more efficient and reliable.

- Model Comparison and Performance Evaluation: Based on initial testing, Claude has been observed to perform better than Mistral in terms of accuracy and relevance of feedback. In the future, we will continue to compare different models, including exploring other options beyond Claude and Mistral, to find the best model for script evaluation.