**GROUP NAME:** CLOUD CREW

**GROUP MEMBERS:**

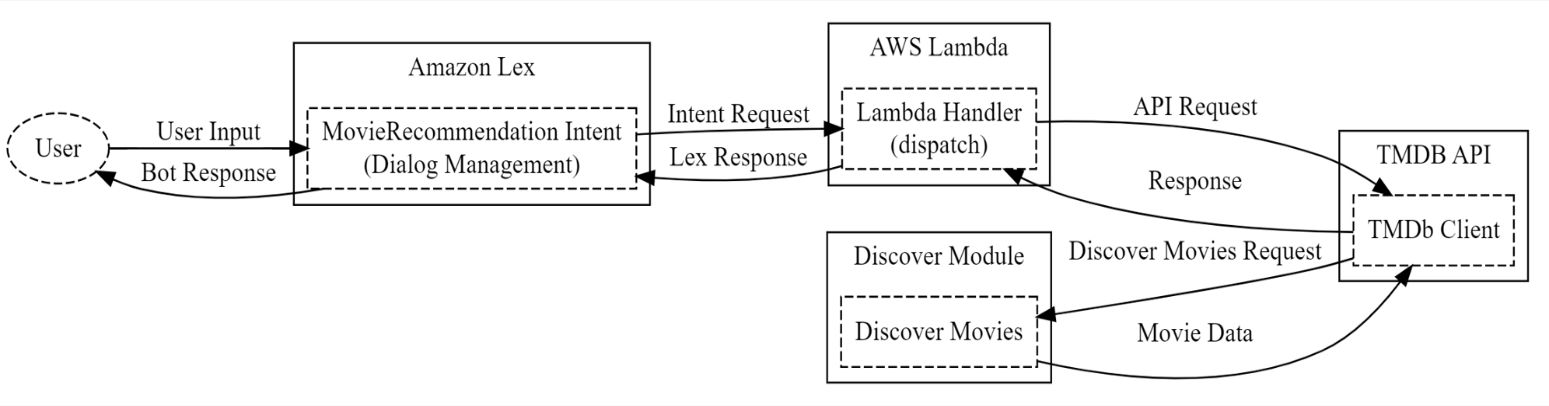
Adeshvar Kishor Mullath

Aryan Verma

Jagdish Sai Mulley

Vineeth Barrenkala

**Abstract:**

The project is an innovative chatbot designed to provide movie recommendations to users. This chatbot leverages Amazon Lex, a powerful conversational AI platform, to interact with users via text input. By analyzing the text provided by users, the chatbot employs an API key from TMDB to fetch relevant movie suggestions based on the user's preferences. Users can engage with the chatbot by entering text and expressing their movie preferences, such as genre or actors. The chatbot processes this information and queries the TMDB database to generate personalized movie recommendations.  
  
**Architecture:**  
  


**Movie Chatbot Architecture**

"Our Movie Chatbot, showcasing how Amazon Lex and AWS Lambda work together to deliver a seamless and interactive movie recommendation experience."

**Component Overview:**

"Let's start by looking at the key components of our architecture."

1. **Amazon Lex:**

A cell phone with text

Description automatically generated

Here we have Amazon Lex, serving as our dialog management system. It handles intents, utterances, slots, and fulfillment. Think of it as the brain behind understanding user inputs and managing the conversation flow.

**2. AWS Lambda:**

Next up is AWS Lambda, our powerhouse for business logic. The Lambda function, equipped with a handler and intent processing functions, takes the information from Lex and performs the heavy lifting. It's responsible for handling intent requests, processing responses, and communicating with external APIs.  
  
  
**3.External API (TMDB • The Movie Database):**  
  
We've integrated with MDB, The Movie Database, as our external API. This allows us to fetch up-to-date movie data to provide users with accurate and relevant recommendations.

Flow of Interaction:

Let's dive into how these components interact.

**- Lex to Lambda (Intent Request):**

When a user interacts with the chatbot, Lex captures the intent and forwards it as an intent request to Lambda. This marks the beginning of our conversational journey.

**- Lambda to Lex (Response):**

Once Lambda processes the intent and performs any necessary actions, it sends back a response to Lex. This response is what the user sees as the chatbot’s reply.

**- Lambda to TMDB (API Request):**

For movie recommendations, Lambda might need fresh data. It communicates with TMDB through API requests to ensure our recommendations are up-to-date and accurate.

**- TMDB to Lambda (Movie Data):**

TMDB responds with movie data, which Lambda uses to generate recommendations for the user.

**Conclusion:**

In summary, our Movie Chatbot leverages the synergies between Amazon Lex, AWS Lambda, and an external API to create an engaging and personalized movie recommendation experience.

**Benefits and Flexibility:**

"This architecture provides scalability, flexibility, and the ability to incorporate real-time data for dynamic and personalized user experiences.

A diagram of a computer program

Description automatically generated  
**1. User Input:**

The process starts when the user provides input, expressing a desire for movie recommendations. This input could be a text message indicating a movie genre, actor, or other preferences.

**2. Lex Processes Input:**

Amazon Lex, the natural language understanding service, processes user input. In this context, Lex is configured with an intent called

"MovieRecommendation," designed to capture the user's request for movie suggestions.

**3. Interaction between Lex and Lambda (Intent Request):**

Lex forwards the processed input to AWS Lambda as an intent request. The intent request contains details about the user's request, including the identified intent ("MovieRecommendation" and any associated slots (parameters like movie genre).

A screenshot of a computer

Description automatically generated

**4. Lambda's Interaction with TMDB API (API Request):**

Upon receiving the intent request, Lambda, which acts as the chatbot's business logic, interacts with the TMDB API. This interaction involves sending a request to the TMDB API to fetch relevant movie data based on the user's preferences.

**5. Flow within the "Discover Module" (Discover Movies Request):**

The Lambda function likely includes a specific module or function called the "Discover Module." This module is responsible for formulating a request to the TMDB API specifically to discover movies based on the user's preferences. The request is sent to the MDB API.

**6. Response Flow from TMDB API to Lambda (Movie Data):**

The TMDB API responds with movie data relevant to the user's request. This could include a list of recommended movies, each with details such as title, release date, and genre.

**7. Lambda Processes Response and Prepares a Reply (Response):**

Lambda processes the response from the MDB API. Depending on the received movie data, Lambda prepares a suitable reply. This reply may include a formatted list of recommended movies.

**8. Final Response Sent Back to User through Lex (Lex Response):**

The prepared reply from Lambda is sent back to Amazon Lex as a response. This response is then relayed to the user.

**9. User Receives Bot Response (Bot Response):**

Finally, the user receives the bot's response through the chat interface. The response contains the recommended movies based on the user's initial input.

This step-by-step process above illustrates how user input triggers a series of interactions between Amazon Lex, AWS Lambda, and the MDB API, resulting in the chatbot providing personalized movie recommendations back to the user.

**Features:**  
Our Movie Chatbot:

* Accepts user inputs for movie preferences (genre, actors, etc.).
* Processes inputs using Lex's "MovieRecommendation" intent.
* Interacts between Lex and Lambda for intent processing.
* Engages with TMDB API to fetch relevant movie data.
* Delivers personalized movie recommendations to users.

Example Session:

**User:** "Recommend action movies."

**Chatbot**: Displays a list of recommended action films.

**How it Works:**

**1. User Input**

The interaction begins when the user inputs preferences, such as movie genres, actors, or themes, seeking tailored movie recommendations.

**2. Lex Processing**

Amazon Lex, the natural language understanding service, interprets the user's input using a configured intent called "MovieRecommendation." This intent captures the essence of the user's request for movie suggestions.

**3. Lex to Lambda Interaction**

Lex forwards the processed input, including the identified intent and any specific parameters (slots), to AWS Lambda as an intent request. This marks the initiation of the chatbot's analytical journey.

**4. Lambda's Interaction with TMDB API**

Upon receiving the intent request, AWS Lambda, functioning as the brain behind the chatbot, interacts with TMDB API. It sends a tailored request to fetch up-to-date movie data aligned with the user's preferences.

**5. Discover Module Flow**

Within Lambda, a specialized module or function called the "Discover Module" takes charge. This module crafts a precise request to TMDB API, targeting movies aligned with the user's expressed preferences.

**6. TMDB API Response to Lambda**

TMDB API promptly responds with a wealth of movie data, including titles, release dates, genres, and other relevant details based on the user's request.

**7. Lambda Data Processing**

Lambda's analytical engine processes the received movie data, formulating a coherent and personalized response tailored to the user's initial preferences.

**8. Final Response via Lex**

The refined movie recommendations, crafted by Lambda based on the retrieved TMDB data, are seamlessly relayed back to Amazon Lex as a comprehensible response.

**9. User Interaction Conclusion**

Finally, the user receives a meticulously curated set of movie recommendations through the chat interface, fulfilling their initial input and providing an enriching movie exploration experience.

This intricate yet streamlined process illustrates the chatbot's capability to interpret user preferences, interact intelligently with external APIs like TMDB, and deliver tailored movie suggestions, resulting in a user-centric and engaging interaction.

**Try it Yourself:**  
These steps will guide you through the process of creating a Movie Recommendations Chatbot using AWS Lex and integrating it with Slack. Follow these step-by-step instructions to seamlessly execute your project.

**Step 1: Register for a TMDB API Key**

Begin by registering for a TMDB API key on TMDB Developer (https://developer.themoviedb.org/reference/intro/getting-started). Follow the instructions provided on the webpage to acquire your unique API key.

**Step 2: Update API Key in install.sh**

Locate the install.sh file in the provided zip folder. Update the APIKEY variable within the file with your TMDB API key. Afterward, zip the folder back to ensure the changes are applied.

**Step 3: Create a Bot on AWS Lex**

- Log in to the AWS Lex console and navigate to AWS Lex.

- Click on "Create Bot."

**Step 4: Bot Configuration**

- Provide a name for your bot and import the zip package created in Step 2.

**Step 5: Role Configuration**

- Create a role with basic Amazon Lex permissions.

- Accept defaults for other options and click on "Import."

- Confirm successful import by checking the intents and slots created.

A screenshot of a computer

Description automatically generated

**Step 6: Test Bot Alias**

- Navigate to Deployment/Aliases for your bot.

- Click on the TestBotAlias and select the English language.

**Step 7: Source and Lambda Configuration**

- Select "movierecommendations" for Source.

- Choose "$LATEST" for Lambda function version or alias and click Save.

A screenshot of a computer

Description automatically generated

**Step 8: Compile the Chatbot**

- In Bot versions/Draft version, under Languages, click on the English language link.

- Click the "Build" button in the bottom menu bar to compile the MovieRecommendations Chatbot.

**Step 9: Test the Chatbot**

- Confirm successful build by clicking on the "Test" button.

A screenshot of a computer

Description automatically generated

**Step 10: Integrate with Slack**

- Open the Slack application and click on "Create App."

**Step 11: Slack App Configuration**

- Give your app a name and select your Slack workspace.

- Note down the Client Secret, Client ID, Signing Secret, and Verification Token.

**Step 12: AWS Console Configuration**

- Navigate to the AWS console and access the created bot.

**Step 13: Channel Integration**

- In the left panel, click on "Channel Integration" and create a new integration.

**Step 14: Slack Integration**

- Select Slack as your integration channel.

- Enter the Client Secret, Client ID, Signing Secret, and Verification Token.

- Click on "Integrate."

Your Movie Recommendations Chatbot is now integrated with Slack, allowing you to enjoy personalized movie suggestions directly from your Slack workspace. Follow these steps carefully to ensure a smooth and successful implementation.

