TECHNICAL HUB CHATBOT

TH – BOT

Problem Statement:

The TH-Bot website project aims to develop an interactive, web-based chatbot interface that integrates with Amazon Lex to provide intelligent, real-time conversations with users. The chatbot is accessible through a user-friendly frontend built using HTML, CSS, and JavaScript, and communicates with the Lex bot via Amazon API Gateway, optionally using AWS Lambda for backend logic. The system ensures secure access through IAM roles and policies, and supports basic user queries such as greetings, FAQs, and informational responses. The goal is to deliver a responsive, cloud-powered chatbot experience that enhances user engagement and support through natural language interaction.

Team Members:

23MH1A05K1 – P. Nithya Santhoshi - **Data Analyst**

23MH1A05N8 – N. Yuktha Sri – **Backend Developer**

23MH1A05O0 – P. Harshini - **Cloud Engineer**

23MH1A0524 – K. Bhavani Surya Prabha – **Frontend Developer**

23P31A0543 – N. Suvarna Ratnam – **Backend Developer**

23P31A0565 – Y. Sri Veni - **Frontend Developer**

Table of Contents:

Abstract

- The Technical Hub Chatbot is a smart, user-friendly virtual assistant developed specifically for the Technical Hub platform. It acts as a guide for students, visitors, and other users by providing instant answers to common questions, mentor details, available technologies, certification info, event updates, and course registration assistance.

- This chatbot is designed to improve the overall user experience by making information easily accessible without the need for human intervention. Instead of manually searching through different pages or waiting for someone to respond, users can simply ask the chatbot and get real-time, accurate answers within seconds.

- Whether someone wants to know about the certifications available in the second year, the name of a mentor, upcoming tech events, or how to register for a course—this bot has it covered.

Purpose

- The main purpose of the TH-Bot project is to create a smart chatbot that can chat with users through a website and provide quick, helpful responses.
- Many users visit websites with questions or need support, and waiting for a human to reply can be slow.
- This chatbot solves that problem by instantly responding to users in a friendly and intelligent way. It's built using Amazon Lex, which understands natural language, so users can talk to the bot just like they would talk to a person.
- This reduces the workload on human support teams and gives users a better, faster experience.

🔷 Scope

- The project involves creating a **simple and user-friendly chatbot interface** using **HTML, CSS, and JavaScript**.
- The chatbot is connected to **Amazon Lex** for processing user input and generating responses.
- **Amazon API Gateway** acts as a **bridge** between the frontend website and the backend chatbot logic.
- **AWS Lambda** can be optionally used to perform **custom backend operations** (e.g., fetching dynamic data).
- **AWS IAM** is used to **manage secure access** between all integrated AWS services.
- The chatbot is currently designed to handle **basic user queries** such as:
    - Greetings (e.g., "Hi", "Hello")
    - Help messages
    - Frequently Asked Questions (FAQs)
- The current version supports **text-based interaction only**.

🔷 Tools and Services Used

- **Amazon Lex**
  • Powers the chatbot by understanding user questions and matching them to the right responses.

- **AWS Lambda**
  • Runs backend code (like fetching data or processing responses) without needing a server.

- **Amazon API Gateway**
  • Connects the frontend chatbot with backend services securely through APIs.

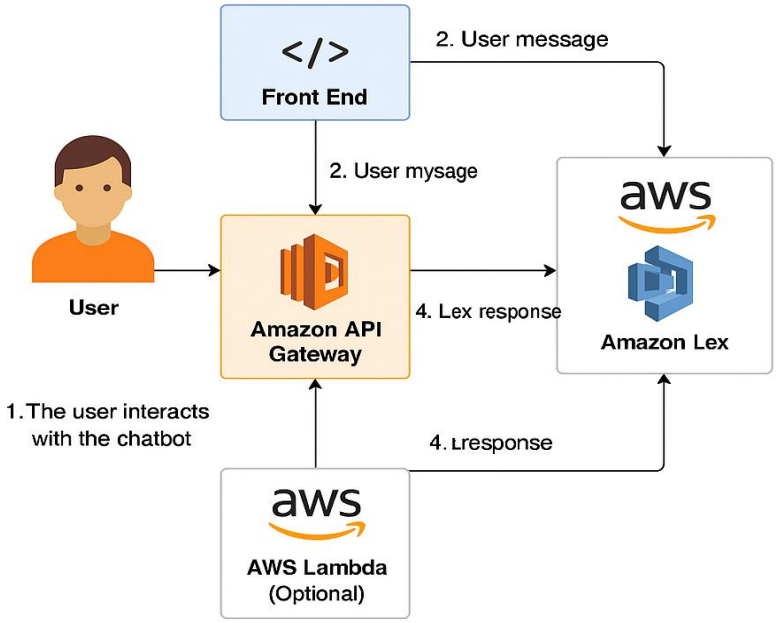
- **AWS IAM**
  • Manages roles and permissions to make sure only authorized components access the system.

- **Frontend (HTML/CSS/JavaScript or React)**
  • Creates the chatbot interface that users interact with on the website.

- **Amazon CloudWatch**
  • Monitors the system and logs user interactions to help fix issues and track performance.

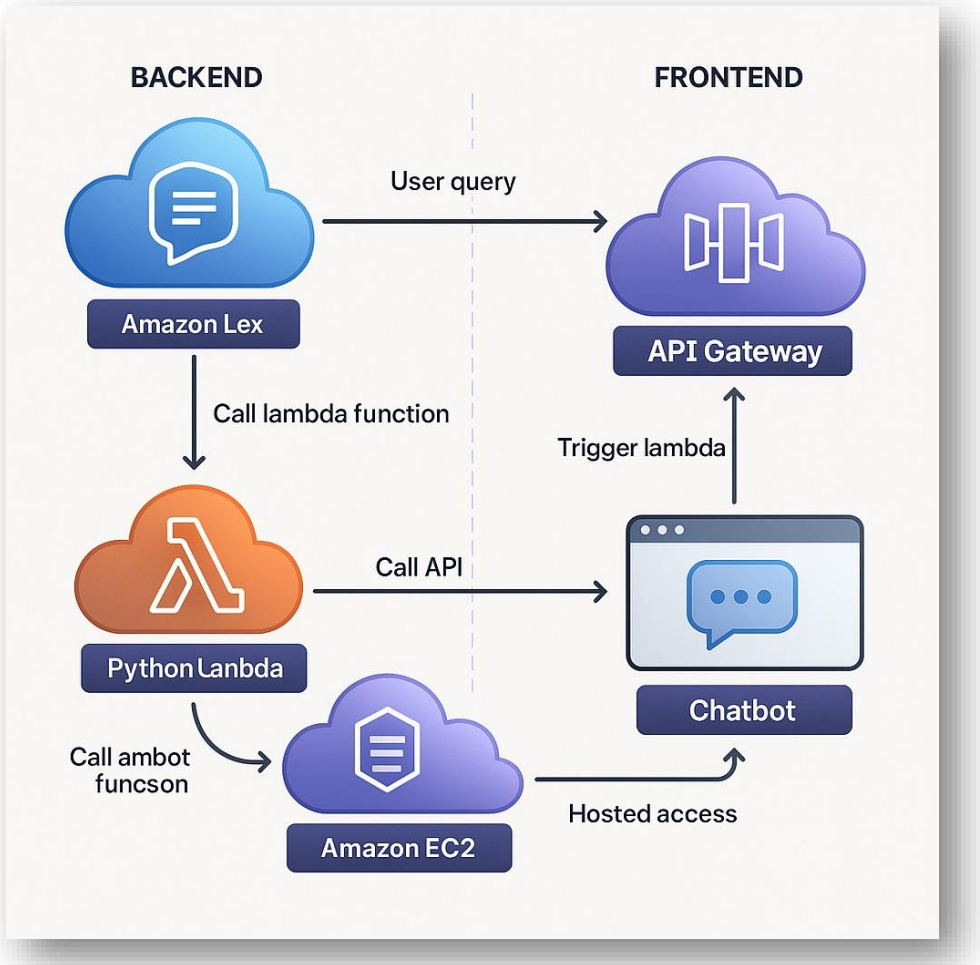
- **AWS S3 (Optional)**
  • Hosts the chatbot frontend files (like HTML and JS) for easy access and deployment.

Work Flow



- User opens the chatbot on the website.
- User types a question in the chatbox.
- The message is sent to API Gateway.
- API Gateway triggers a Lambda function.
- Lambda sends the message to Amazon Lex.
- Lex identifies the intent and prepares a reply.
- The reply goes back to Lambda.
- Lambda sends the final response to API Gateway.
- API Gateway returns the reply to the frontend.
- Chatbot displays the answer to the user

Architecture



Frontend Side (User Interaction)

1. Chatbot Interface

- This is the web-based chatbot (built with HTML/JavaScript) where users type their queries.
- It interacts with the backend through a REST API.

2. Amazon API Gateway

- Acts as a secure doorway between the chatbot frontend and backend AWS services.
- When a user submits a question, the API Gateway captures the request and sends it to the backend services securely.

🔵 **Backend Side (Processing the Query)**

3. Amazon Lex

- This is the heart of your chatbot's brain.
- It understands what the user is saying (using Natural Language Processing) and figures out the intent.
- Based on the intent, it may respond directly or call a backend service for help.

4. AWS Lambda (Python Function)

- Amazon Lex invokes a Lambda function when it needs to do some logic or fetch data.
- This Lambda function is written in Python and acts as a middle layer.
- It processes the intent and performs actions like database queries, calculations, or sending data to other AWS services.

5. Amazon EC2 (Elastic Compute Cloud)

- If heavy lifting or more custom functionality is needed, Lambda calls an API hosted on EC2.
- EC2 runs your custom application and returns the result.
- This service gives your bot more power and flexibility to handle complex queries.

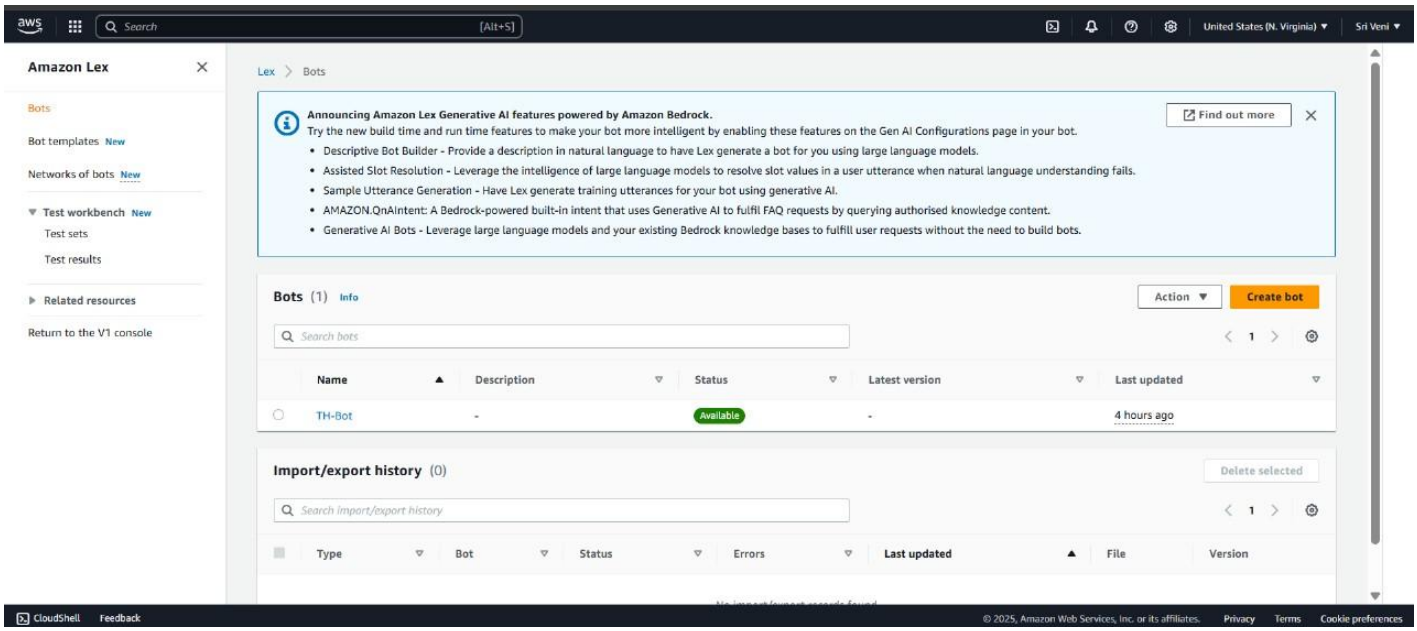
🔷 Approach

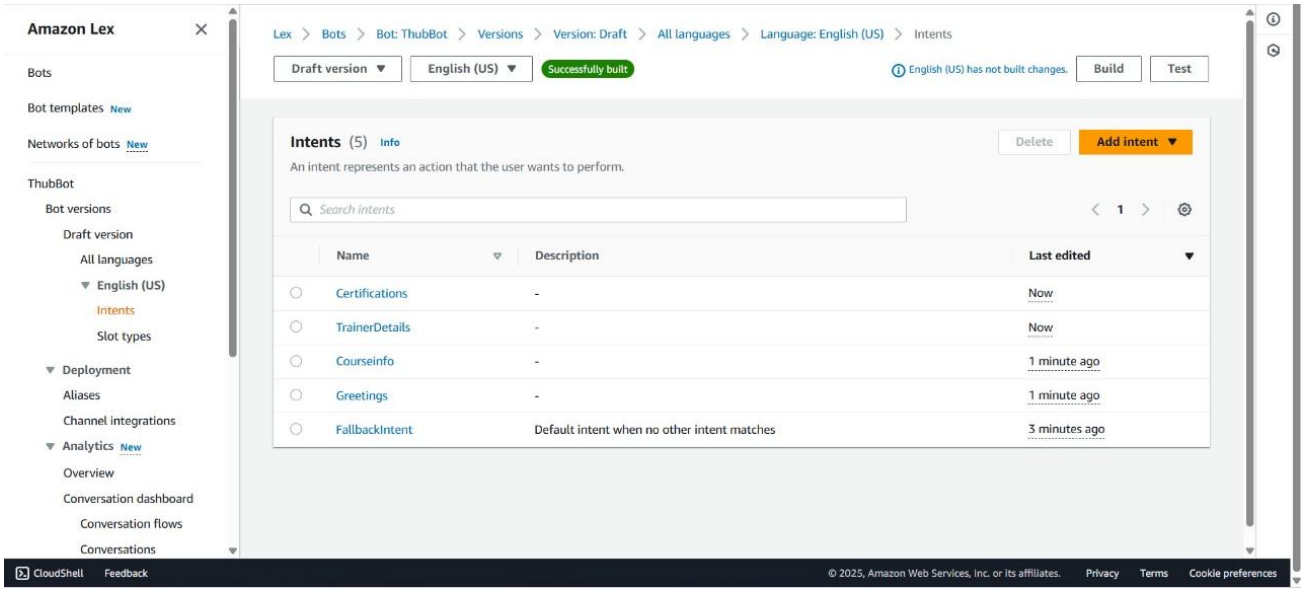
To build the TH-Bot (AI chatbot for Technical Hub), a systematic approach was followed. Each step involved the use of specific AWS services and development techniques to ensure smooth interaction between the user and backend services.

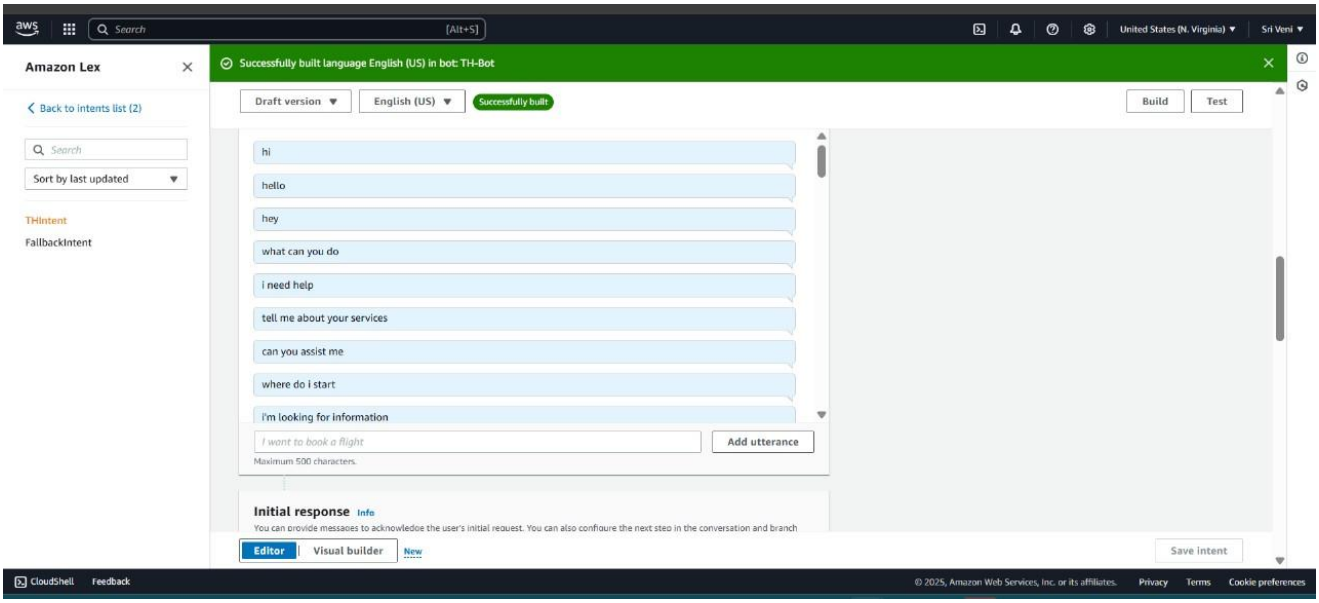
1. Defining Intents and Utterances in Amazon Lex

We started by setting up the core of the chatbot:

- Created a new Amazon Lex bot named **TH - Bot**
- Defined **multiple intents** like greetings, course info, trainer details, certifications, etc.
- Added **sample utterances** for each intent to help Lex understand user queries.
- For some intents, slot types were also defined.



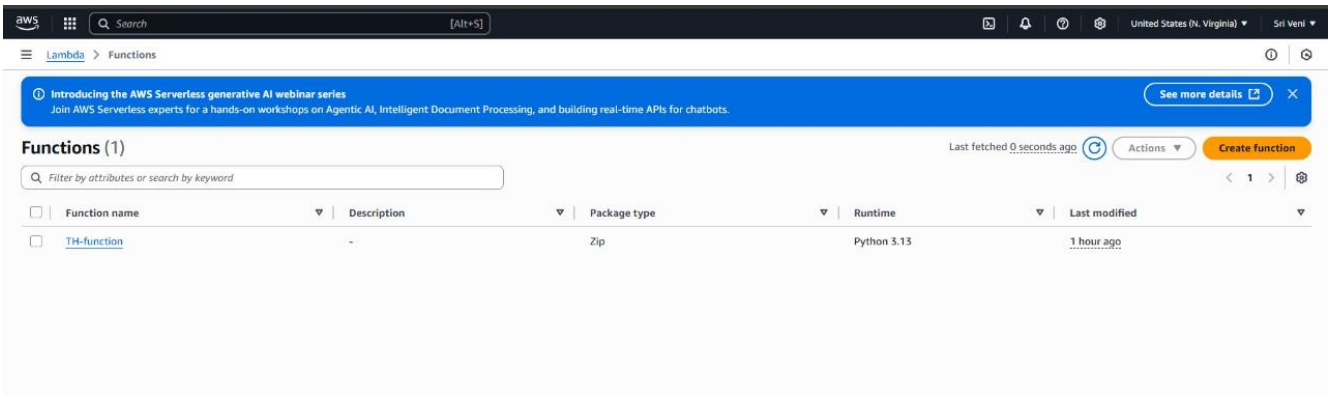


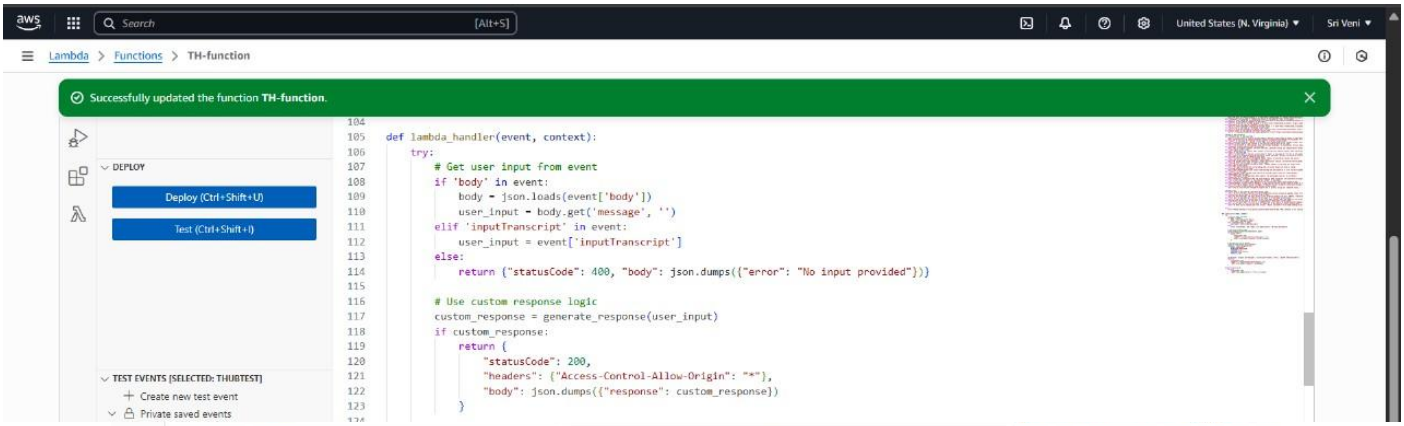


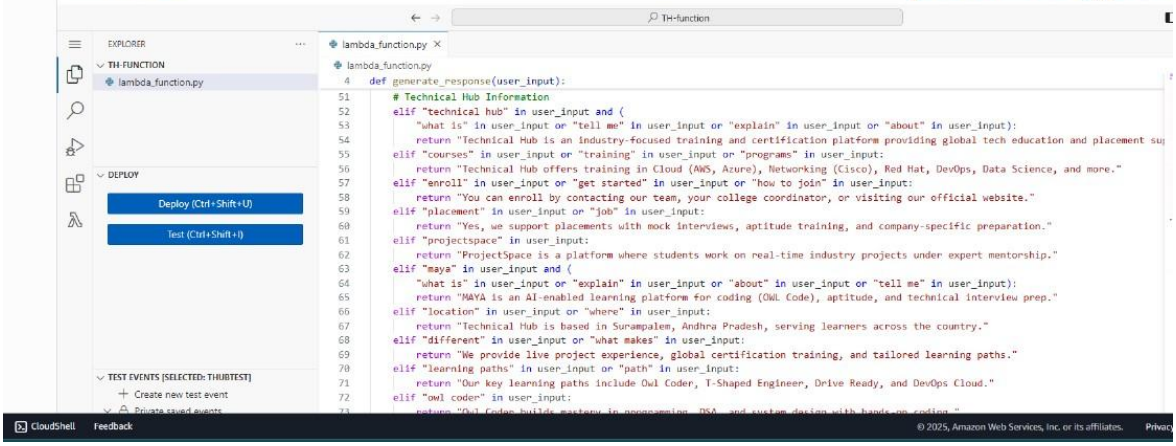
2. Creating Lambda Function for Logic Handling

A backend Lambda function was created to:

- Handle complex logic and fetch dynamic responses.
- Process different inputs like trainer names, courses, or placement-related queries.
- Return structured responses to Lex in Python format.


Search
[Alt+S]
United States (N. Virginia)
Sri Veni
Lambda > Functions
Introducing the AWS Serverless generative AI webinar series
Join AWS Serverless experts for a hands-on workshops on Agentic AI, Intelligent Document Processing, and building real-time APIs for chatbots.
See more details
Functions (1)
Last fetched 0 seconds ago
Actions
Create function
Filter by attributes or search by keyword
Function name
Description
Package type
Runtime
Last modified
TH-function
-
Zip
Python 3.13
1 hour ago

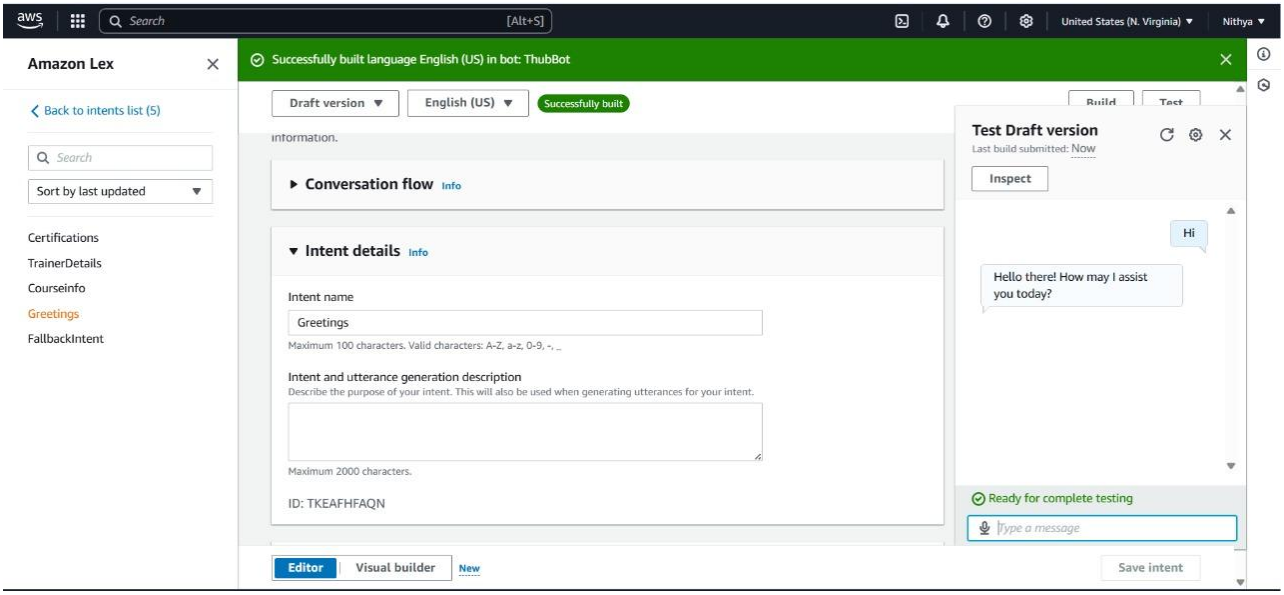

Search
[Alt+S]
United States (N. Virginia)
Sri Veni
Lambda > Functions > TH-function
Successfully updated the function TH-function.
DEPLOY
Deploy (Ctrl+Shift+U)
Test (Ctrl+Shift+I)
TEST EVENTS [SELECTED: THUBTEST]
Create new test event
Private saved events
def lambda_handler(event, context):
try:
# Get user input from event
if 'body' in event:
body = json.loads(event['body'])
user_input = body.get('message', '')
elif 'inputTranscript' in event:
user_input = event['inputTranscript']
else:
return {"statusCode": 400, "body": json.dumps({"error": "No input provided"})}
# Use custom response logic
custom_response = generate_response(user_input)
if custom_response:
return {
"statusCode": 200,
"headers": {"Access-Control-Allow-Origin": "*"},
"body": json.dumps({"response": custom_response})
}


TH-function
EXPLORER
TH-FUNCTION
lambda_function.py
DEPLOY
Deploy (Ctrl+Shift+U)
Test (Ctrl+Shift+I)
TEST EVENTS [SELECTED: THUBTEST]
Create new test event
def generate_response(user_input):
# Technical Hub Information
elif "technical hub" in user_input and (
"what is" in user_input or "tell me" in user_input or "explain" in user_input or "about" in user_input):
return "Technical Hub is an industry-focused training and certification platform providing global tech education and placement su
elif "courses" in user_input or "training" in user_input or "programs" in user_input:
return "Technical Hub offers training in Cloud (AWS, Azure), Networking (Cisco), Red Hat, DevOps, Data Science, and more."
elif "enroll" in user_input or "get started" in user_input or "how to join" in user_input:
return "You can enroll by contacting our team, your college coordinator, or visiting our official website."
elif "placement" in user_input or "job" in user_input:
return "Yes, we support placements with mock interviews, aptitude training, and company-specific preparation."
elif "projectspace" in user_input:
return "ProjectSpace is a platform where students work on real-time industry projects under expert mentorship."
elif "maya" in user_input and (
"what is" in user_input or "explain" in user_input or "about" in user_input or "tell me" in user_input):
return "MAYA is an AI-enabled learning platform for coding (OWL Code), aptitude, and technical interview prep."
elif "location" in user_input or "where" in user_input:
return "Technical Hub is based in Surampalem, Andhra Pradesh, serving learners across the country."
elif "different" in user_input or "what makes" in user_input:
return "We provide live project experience, global certification training, and tailored learning paths."
elif "learning paths" in user_input or "path" in user_input:
return "Our key learning paths include Owl Coder, T-Shaped Engineer, Drive Ready, and DevOps Cloud."
elif "owl coder" in user_input:
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.

3. Connecting Lex to Lambda

The Lambda function was integrated with the Lex bot:

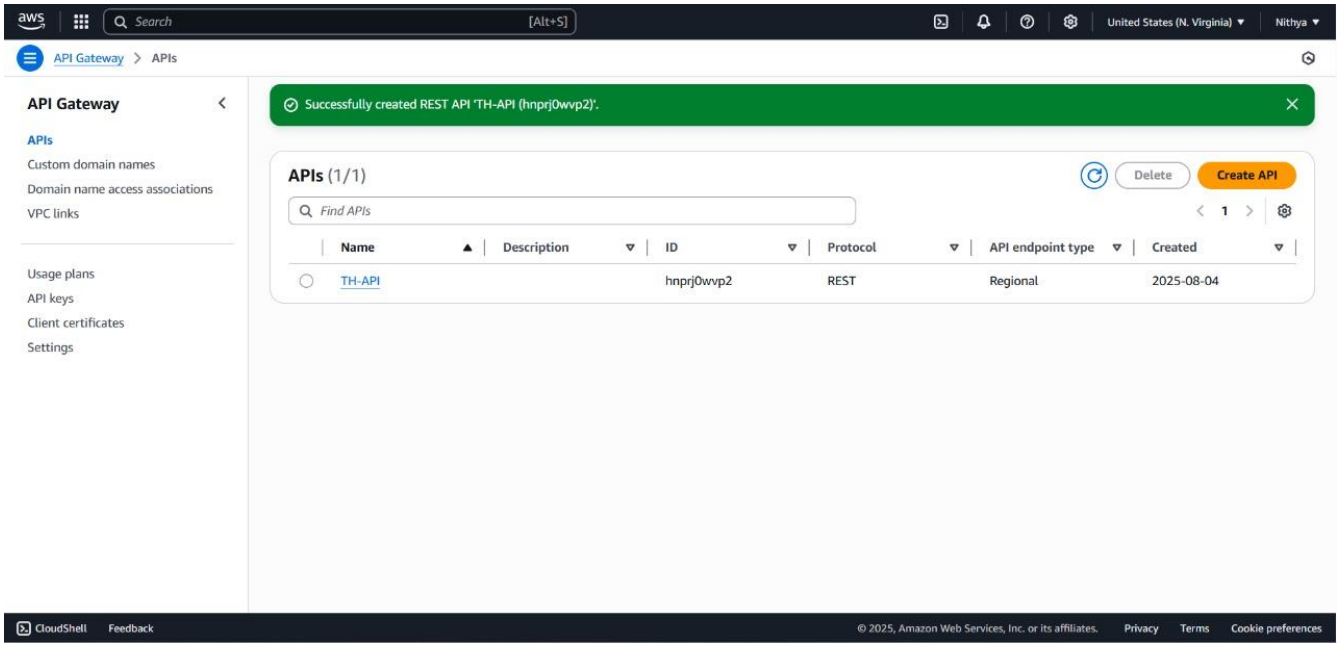
- Each intent was linked to the Lambda function using fulfillment.
- This allowed Lex to call the function after recognizing the intent.

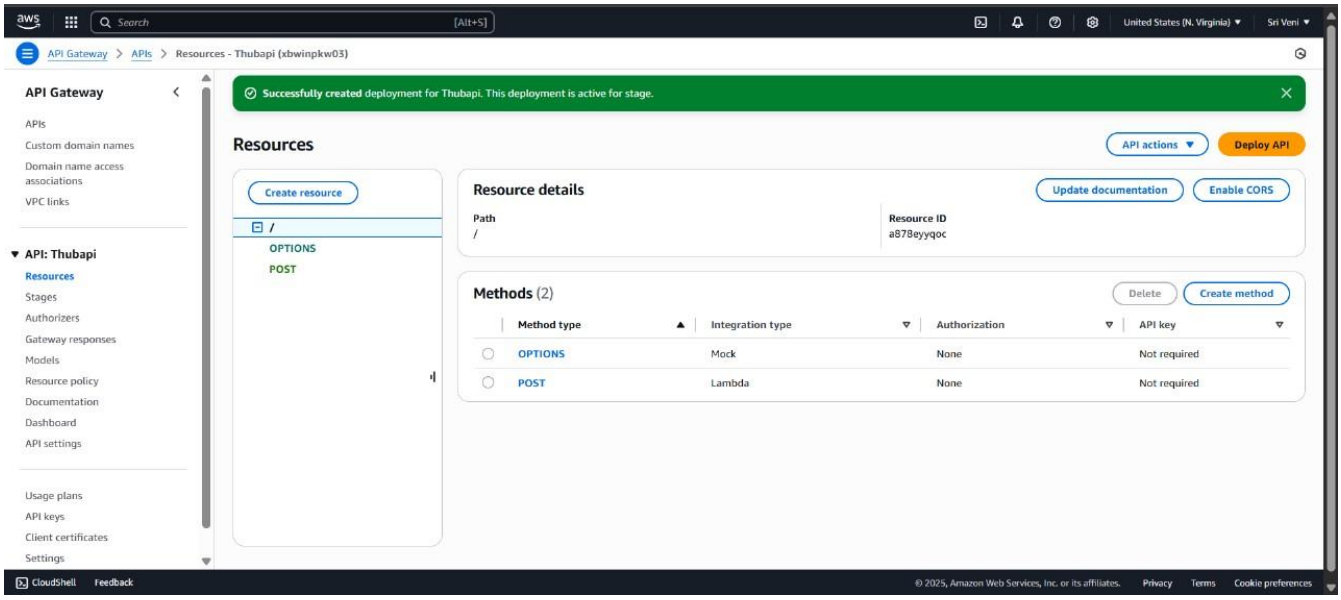


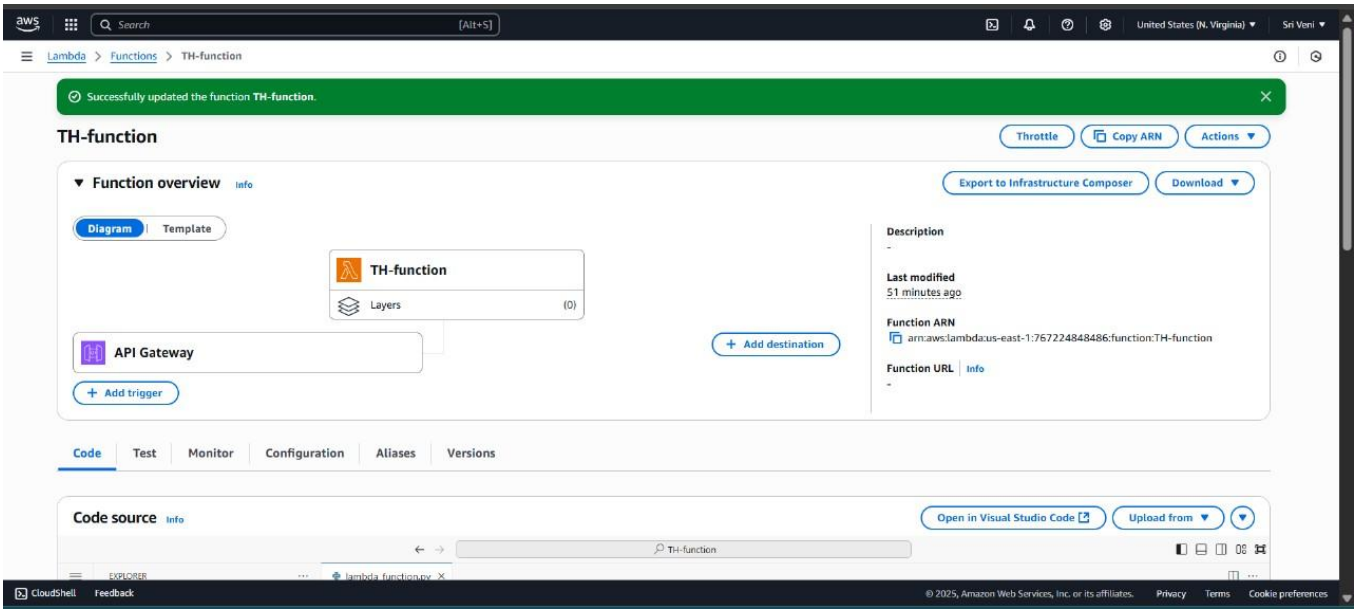
4. Creating API Gateway for Frontend Integration

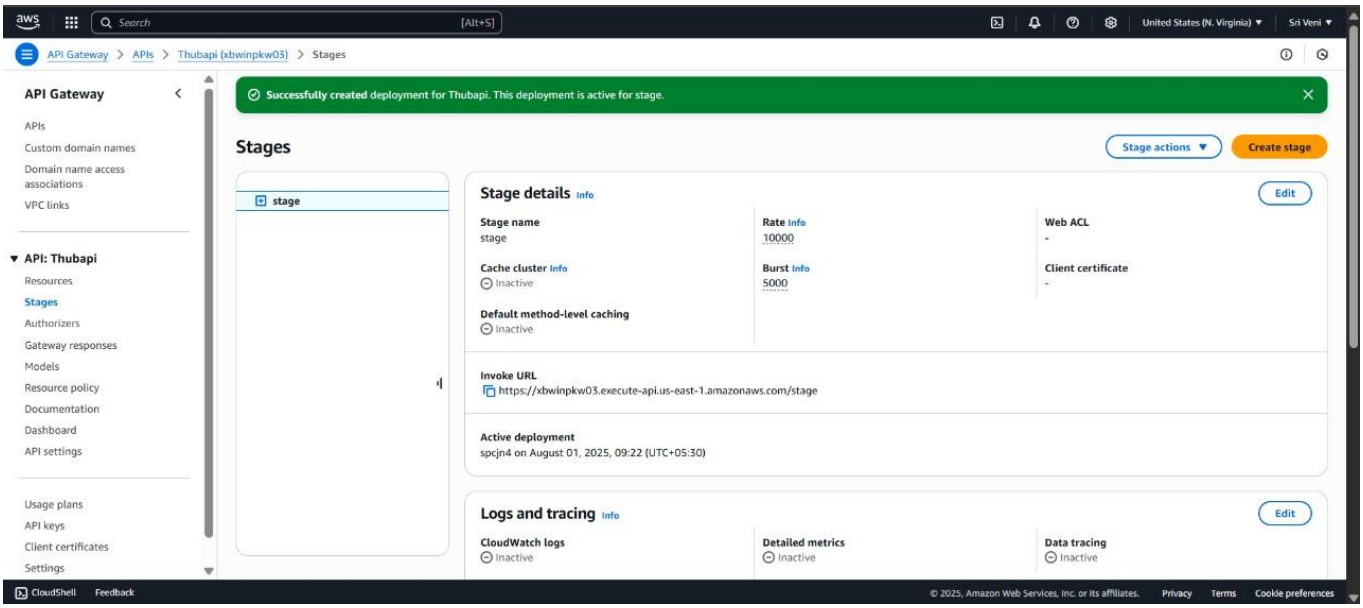
To connect the frontend to backend:

- An **Amazon API Gateway** was created.
- Configured to trigger the Lambda function when users send messages.
- CORS was enabled to allow requests from browser-based frontend.


aws
Search
[Alt+S]
United States (N. Virginia)
Nithya
API Gateway > APIs
API Gateway
APIs
Custom domain names
Domain name access associations
VPC links
Usage plans
API keys
Client certificates
Settings
Successfully created REST API 'TH-API (hnprj0wvp2)'.
APIs (1/1)
Delete
Create API
Find APIs
Name
Description
ID
Protocol
API endpoint type
Created
TH-API
hnprj0wvp2
REST
Regional
2025-08-04
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Search
[Alt+S]
United States (N. Virginia)
Sri Veni
API Gateway > APIs > Resources - Thubapi (xbwinpkw03)
API Gateway
APIs
Custom domain names
Domain name access associations
VPC links
API: Thubapi
Resources
Stages
Authorizers
Gateway responses
Models
Resource policy
Documentation
Dashboard
API settings
Usage plans
API keys
Client certificates
Settings
Successfully created deployment for Thubapi. This deployment is active for stage.
Resources
API actions
Deploy API
Create resource
/
OPTIONS
POST
Resource details
Update documentation
Enable CORS
Path
/
Resource ID
a878eyyqoc
Methods (2)
Delete
Create method
Method type
Integration type
Authorization
API key
OPTIONS
Mock
None
Not required
POST
Lambda
None
Not required
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

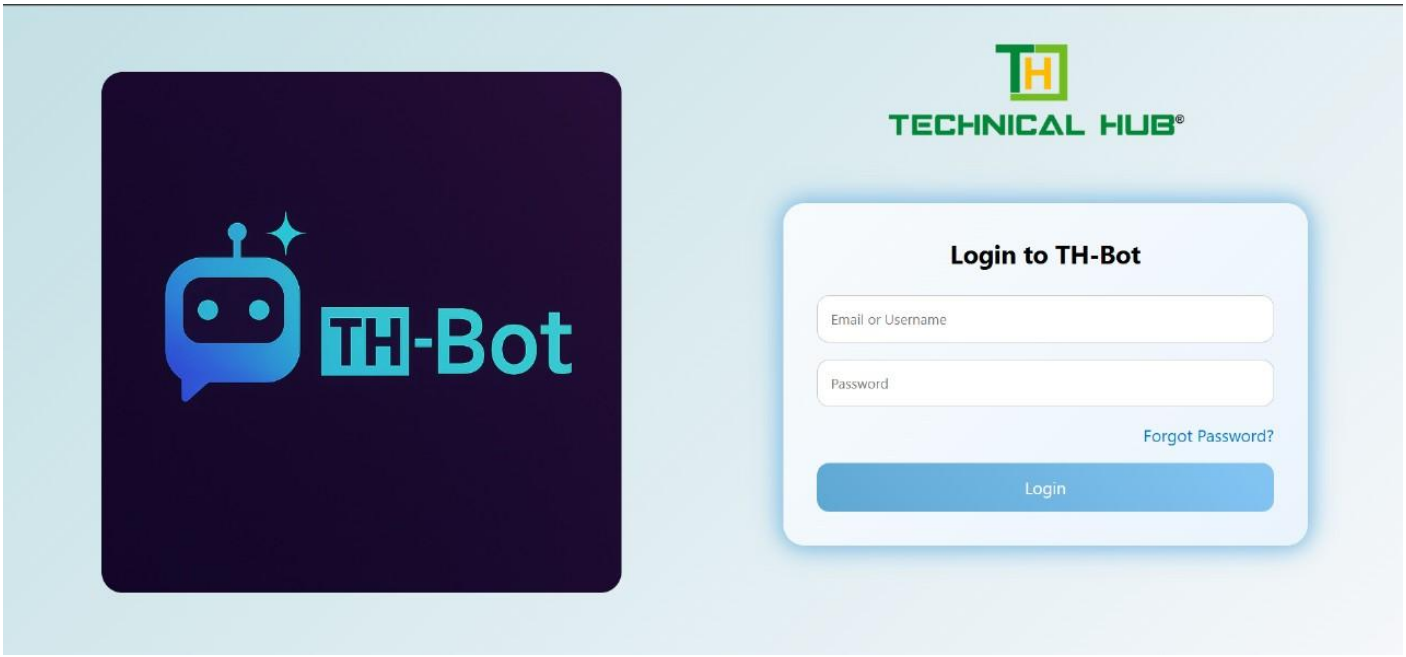


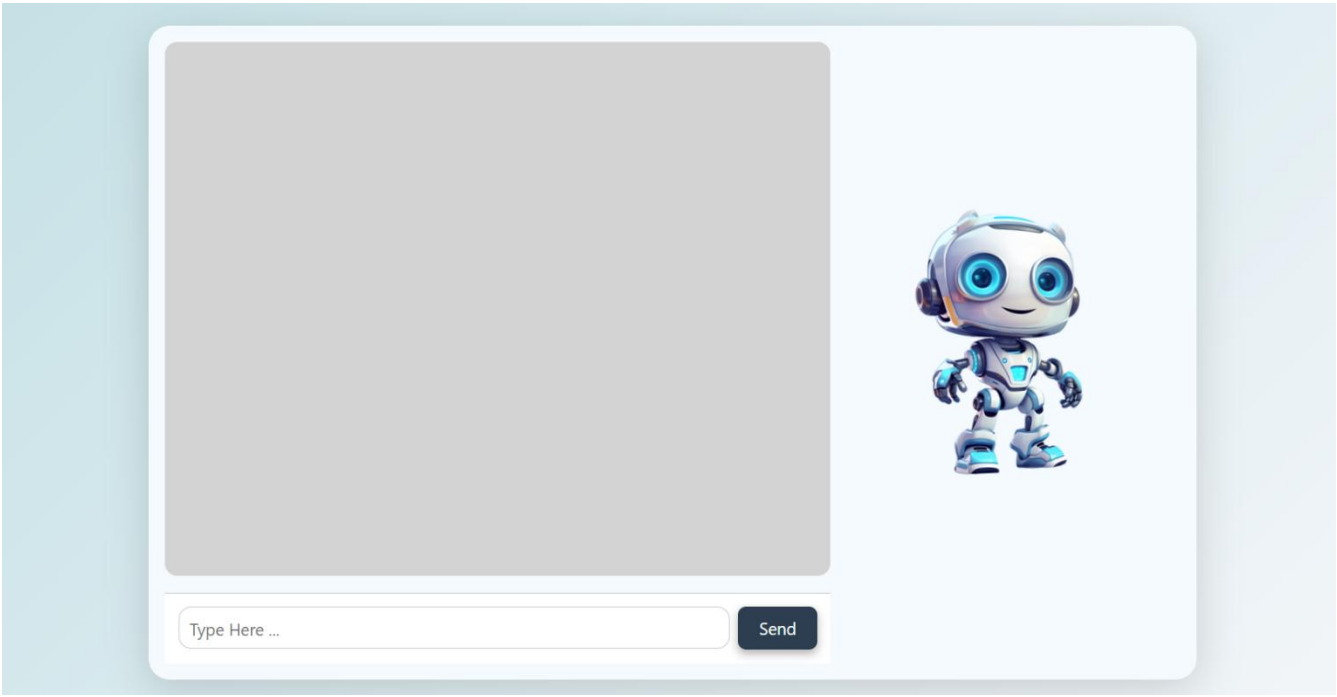


5. Developing the Frontend (Chatbot UI)

We created a user-friendly chatbot interface using:

- HTML, CSS, and JavaScript.
- Input box to capture user queries.
- Display area to show chatbot responses dynamically.

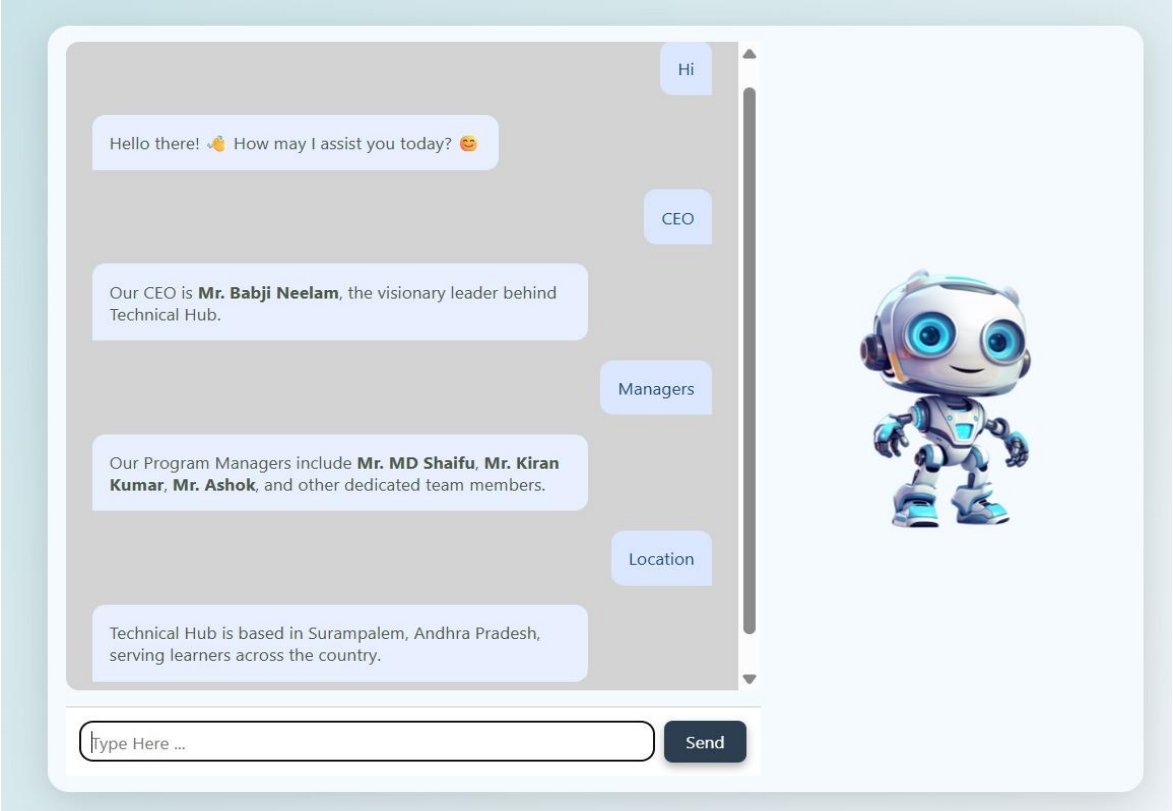

TH-Bot
TECHNICAL HUB®
Login to TH-Bot
Email or Username
Password
Forgot Password?
Login

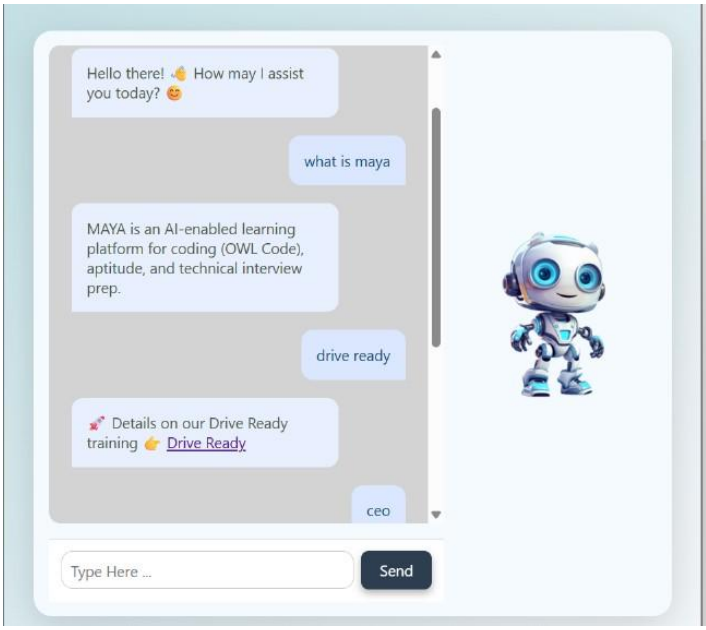

Type Here ...
Send

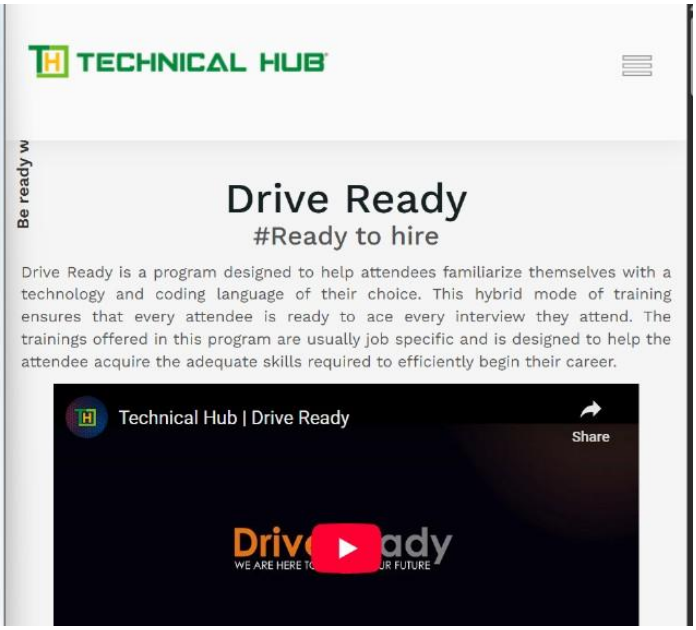
6. Hosting Frontend (Optional)

The frontend was optionally hosted:

- Either locally or on **Amazon S3** for static hosting.
- API keys and endpoints were securely linked to the hosted site.



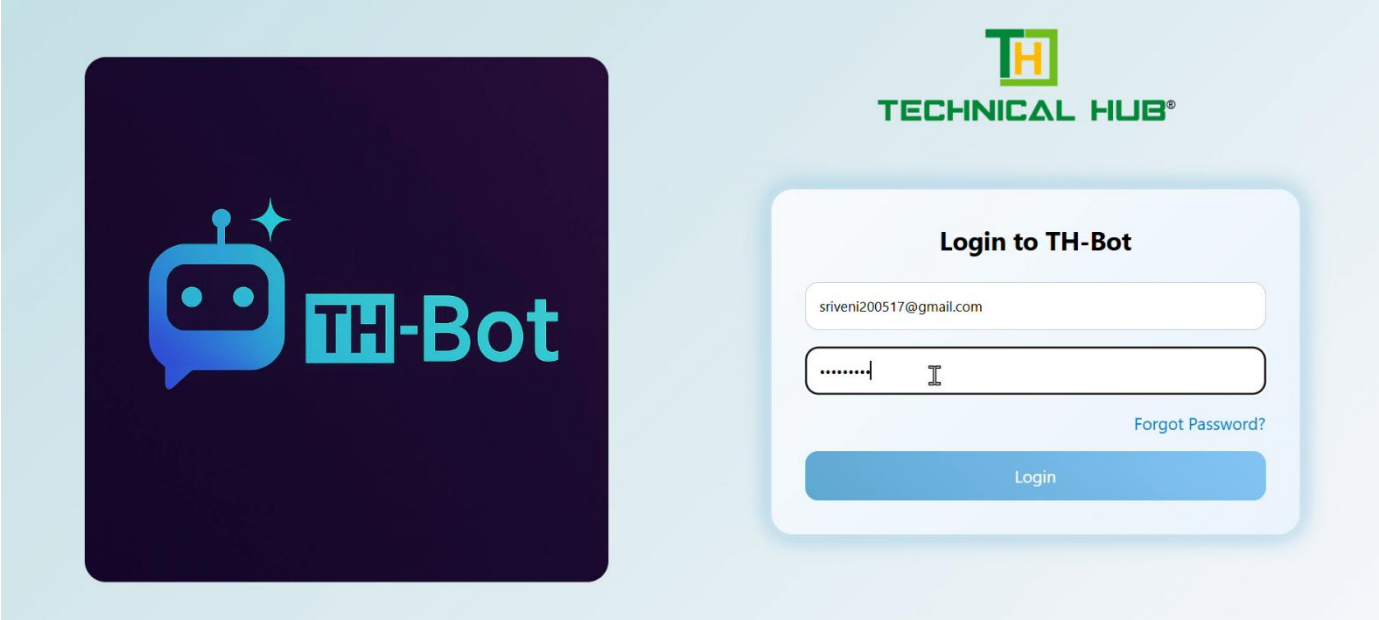


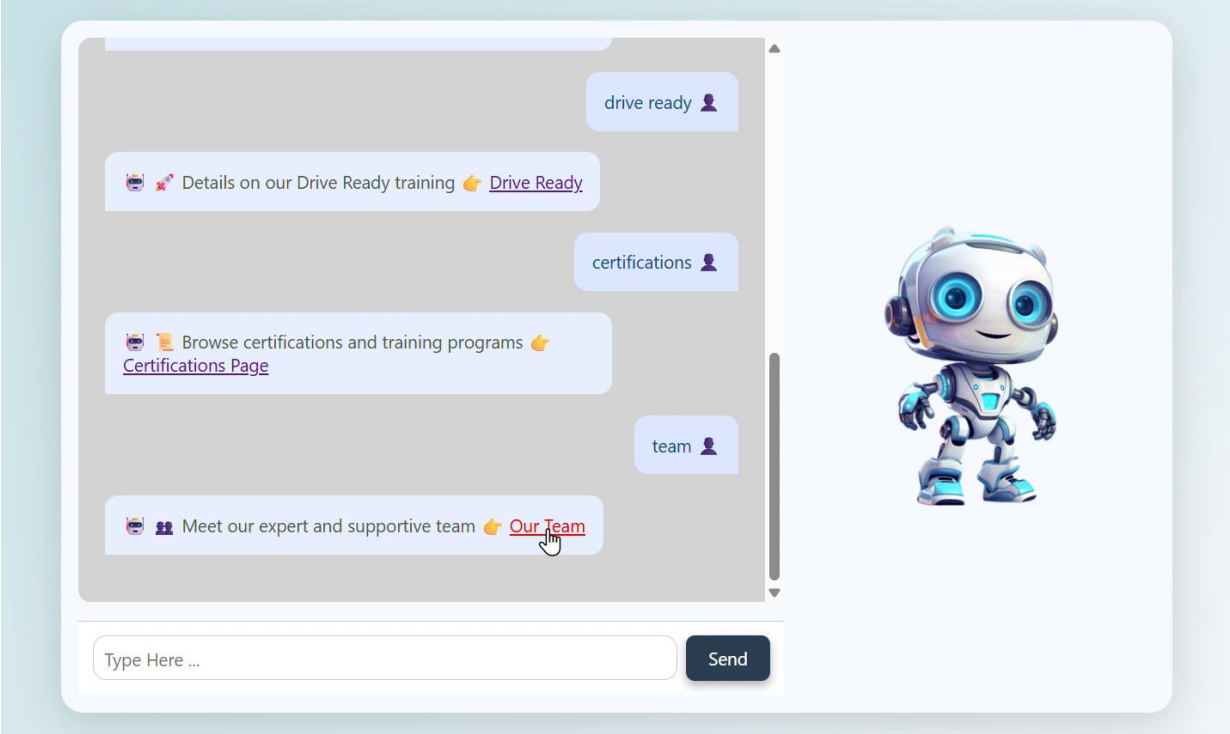


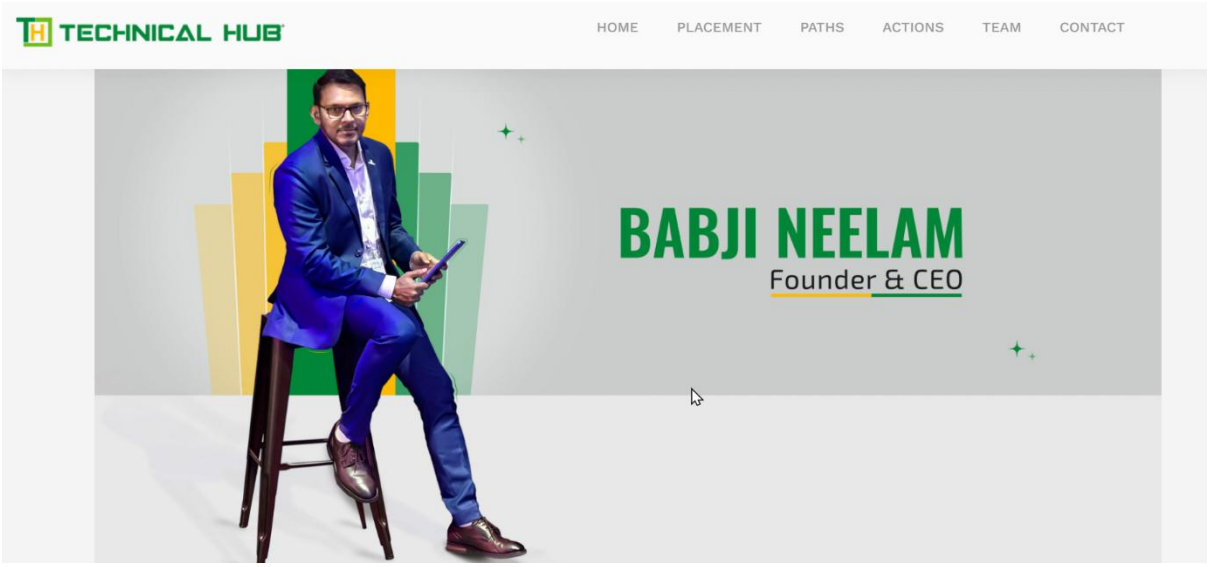
8. Final Deployment and Integration

After all configurations:

- The frontend, Lambda, Lex, and API Gateway were fully connected.
- The chatbot was now capable of handling real-time student queries regarding Technical Hub.







✅ 8. Benefits

This chatbot project offers several key benefits both technically and functionally:

- **Automated Query Support**
  - It reduces the dependency on human operators for basic queries.
  - Students and visitors can get instant responses without delays.
- **Reduces Manual Workload**
  - Faculty or support teams no longer need to answer repeated questions manually.
  - Routine queries like course details, certification info, or trainer names are handled by the bot automatically.
- **Easy Scalability with AWS Serverless**
  - With services like Lambda and API Gateway, the system is inherently scalable.
  - No need to manage physical servers or worry about load balancing.
- **Improves Student Engagement**

- Students feel more connected when they can interact with a smart assistant.
- Encourages curiosity and exploration of the Technical Hub's offerings.

🚀 9. Future Enhancements

To make the chatbot even more useful and intelligent, the following improvements can be considered in the future:

🔹 **Add Voice Support**

• Integrate Amazon Polly or Alexa voice services for voice-based interaction.

• Make the chatbot more inclusive for users with reading or typing difficulties.

🔹 **Integrate with Maya Platform**

• Expand the chatbot's knowledge to include the Maya student platform.

• Allow students to check attendance, internal marks, and schedules through the bot.

🔹 **Add Admin Panel for Dynamic Updates**

• Create a simple web-based admin interface.

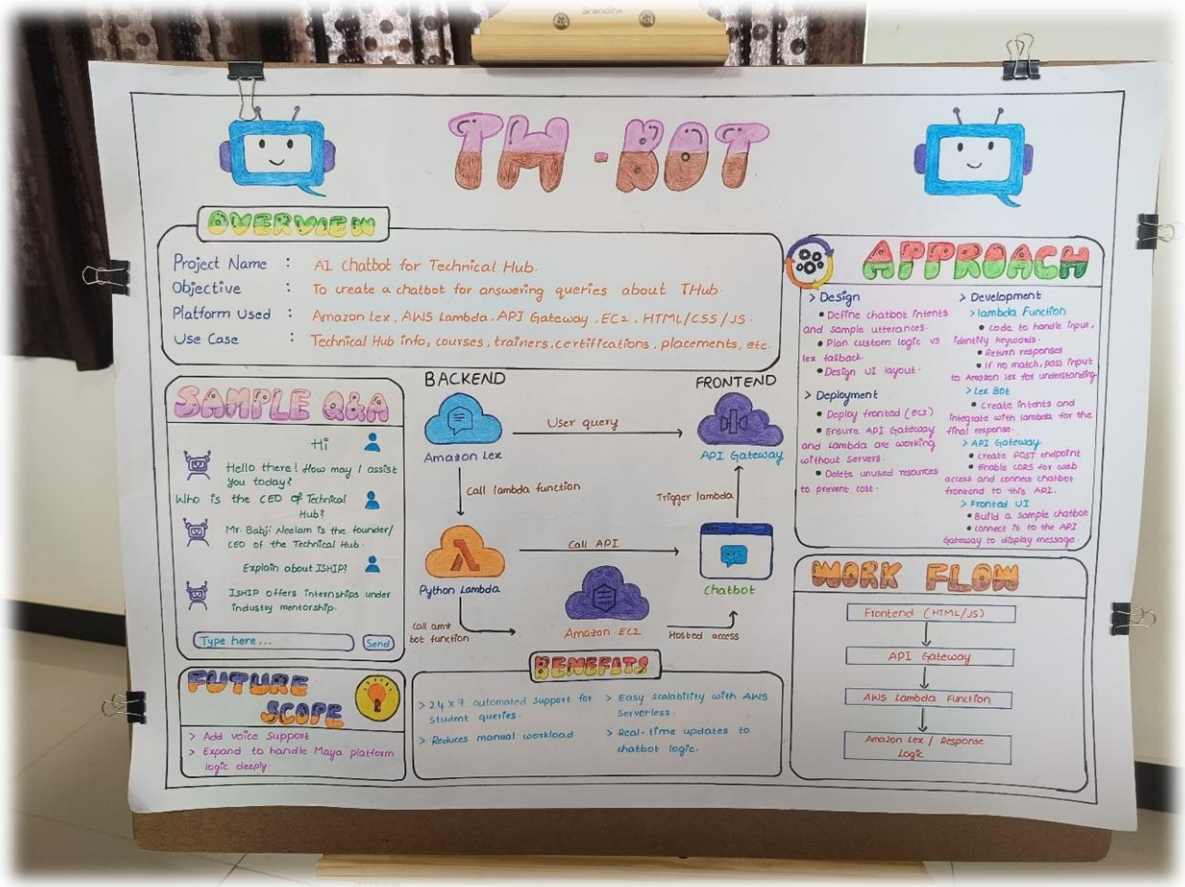
• Enable non-technical staff to update FAQs, responses, or announcements easily.

🔹 **Multi-Language Support**

• Support regional languages like Telugu or Hindi to increase accessibility.

• Use Amazon Lex's language support to expand multilingual capabilities.

Team Photos:


TH-BOT
OVERVIEW
Project Name : AI chatbot for Technical Hub.
Objective : To create a chatbot for answering queries about THub.
Platform Used : Amazon Lex, AWS Lambda, API Gateway, EC2, HTML/CSS/JS.
Use Case : Technical Hub info, courses, trainers, certifications, placements, etc.
APPROACH
> Design
• Define chatbot intents and sample utterances.
• Plan custom logic vs lex fallback.
• Design UI layout.
> Deployment
• Deploy frontend (EC2)
• Ensure API Gateway and Lambda are working without servers.
• Delete unused resources to prevent cost.
> Development
> lambda Function
• code to handle input, identify keywords.
• Return responses
• If no match, pass input to Amazon lex for understanding.
> Lex Bot
• Create intents and integrate with lambda for the final response.
> API Gateway
• Create POST endpoint
• Enable CORS for web access and connect chatbot frontend to this API.
> Frontend UI
• Build a sample chatbot
• Connect JS to the API Gateway to display message.
SAMPLE Q&A
Hi
Hello there! How may I assist you today?
Who is the CEO of Technical Hub?
Mr. Babji Neelam is the founder/CEO of the Technical Hub.
Explain about ISHIP?
ISHIP offers internships under industry mentorship.
Type here...
Send
BACKEND
FRONTEND
Amazon Lex
User query
API Gateway
call lambda function
Trigger lambda
Python Lambda
Call API
Chatbot
call amit bot function
Amazon EC2
Hosted access
BENEFITS
> 24 x 7 automated support for student queries.
> Reduces manual workload
> Easy scalability with AWS Serverless.
> Real-time updates to chatbot logic.
WORK FLOW
Frontend (HTML/JS)
API Gateway
AWS Lambda Function
Amazon Lex / Response Logic
FUTURE SCOPE
> Add voice Support
> Expand to handle Maya platform logic deeply


TH-BOT
APPROACH
WORK FLOW
FUTURE SCOPE


TM-BOT
APPROACH
GANTT CHART