Nana University Chatbot Deployment Report

Done by: Malleeswari D
Date:13-06-2025

1. PROJECT OVERVIEW:

This document outlines the successful deployment of a Dialogflow CX chatbot for Nana University using Python Flask and Google Cloud Shell. The chatbot responds to university-related queries like admission, courses, and facilities in real-time.

2. TOOLS AND TECHNOLOGIES USED:

- Dialogflow CX
- Google Cloud Platform (IAM, Cloud Shell)
- Python 3 (Flask framework)
- HTML/CSS/JavaScript
- Service Account with Dialogflow API Admin role

3. FOLDER STRUCTURE

```
nana-university-chatbot/
├── index.html
├── app.py
└── vertexaidemokeys.json
```

4. CODE FILES:

4.1 index.html

Contains the chatbot interface for users to interact with.

```
<!DOCTYPE html>
<html>
<head>
  <title>Nana University Chatbot</title>
  <style>
    body { font-family: Arial; text-align: center; padding: 30px; background-color: #f4f4f4; }
    #chatbox { width: 90%; height: 300px; border: 1px solid #ccc; margin: 20px auto; padding: 10px; background: #fff; overflow-y: scroll; }
    input { width: 70%; padding: 10px; }
    button { padding: 10px 20px; }
    .logo { font-size: 30px; font-weight: bold; color: #2c3e50; }
  </style>
</head>
<body>
  <div class="logo">🎓 NU</div>
  <h2>Welcome to Nana University</h2>
  <p><i>Empowering the future, one student at a time.</i></p>

  <div id="chatbox"></div>
  <input id="userInput" placeholder="Ask me anything...">
  <button onclick="sendMessage()">Send</button>

  <script>
    function sendMessage() {
      const input = document.getElementById("userInput").value;
      const chatbox = document.getElementById("chatbox");
      chatbox.innerHTML += "<div><b>You:</b> " + input + "</div>";
      document.getElementById("userInput").value = "";

      fetch('/chat', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ message: input })
      })
      .then(response => response.json())
      .then(data => {
        chatbox.innerHTML += "<div><b>Bot:</b> " + data.reply + "</div>";
        chatbox.scrollTop = chatbox.scrollHeight;
      })
      .catch(error => {
        chatbox.innerHTML += "<div><b>Bot:</b> Sorry, I couldn't reach the server.</div>";
```

```
        chatbox.innerHTML += "<div><b>Bot:</b> Sorry, I couldn't reach the server.</div>";
      });
    }
  </script>
</body>
</html>
```

4.2 app.py

The Python backend that handles communication between the chatbot UI and Dialogflow CX API.

```
from flask import Flask, request, jsonify, send_from_directory
from google.cloud import dialogflowcx_v3beta1 as dialogflowcx
from google.oauth2 import service_account
import uuid
import os

app = Flask(__name__, static_folder='.', static_url_path='')

PROJECT_ID = 'vertexaiconversationaldemo'
LOCATION = 'global'
AGENT_ID = 'ef3194a3-44de-4ca9-bc82-9c6ce59d8e49'
LANGUAGE_CODE = 'en'
SERVICE_ACCOUNT_FILE = 'vertexaidemokeys.json'

credentials = service_account.Credentials.from_service_account_file(SERVICE_ACCOUNT_FILE)
client = dialogflowcx.SessionsClient(credentials=credentials)

@app.route('/')
def serve_index():
    return send_from_directory('.', 'index.html')

@app.route('/chat', methods=['POST'])
def chat():
    user_input = request.json['message']
    session_id = str(uuid.uuid4())
    session_path = client.session_path(PROJECT_ID, LOCATION, AGENT_ID, session_id)

    text_input = dialogflowcx.TextInput(text=user_input)
    query_input = dialogflowcx.QueryInput(text=text_input, language_code=LANGUAGE_CODE)

    response = client.detect_intent(request={"session": session_path, "query_input": query_input})
    reply = response.query_result.response_messages[0].text.text[0]

    return jsonify({"reply": reply})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```

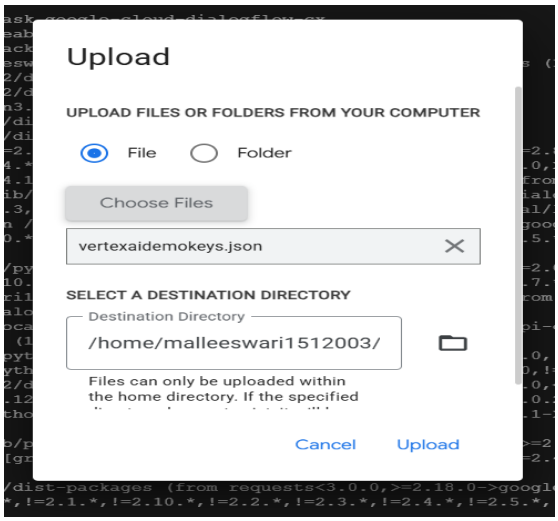
5. SERVICE ACCOUNT & IAM SETUP:

A service account named `dialogflow-cx-web@vertexaiconversationaldemo.iam.gserviceaccount.com` was created and assigned the following IAM roles:
- Dialogflow API Admin
- Dialogflow API Client

This ensured secure communication with the Dialogflow CX API.

6. DEPLOYMENT STEPS:

1. Created folder and added files in Google Cloud Shell.
2. Uploaded the service account key (vertexaidemokeys.json).



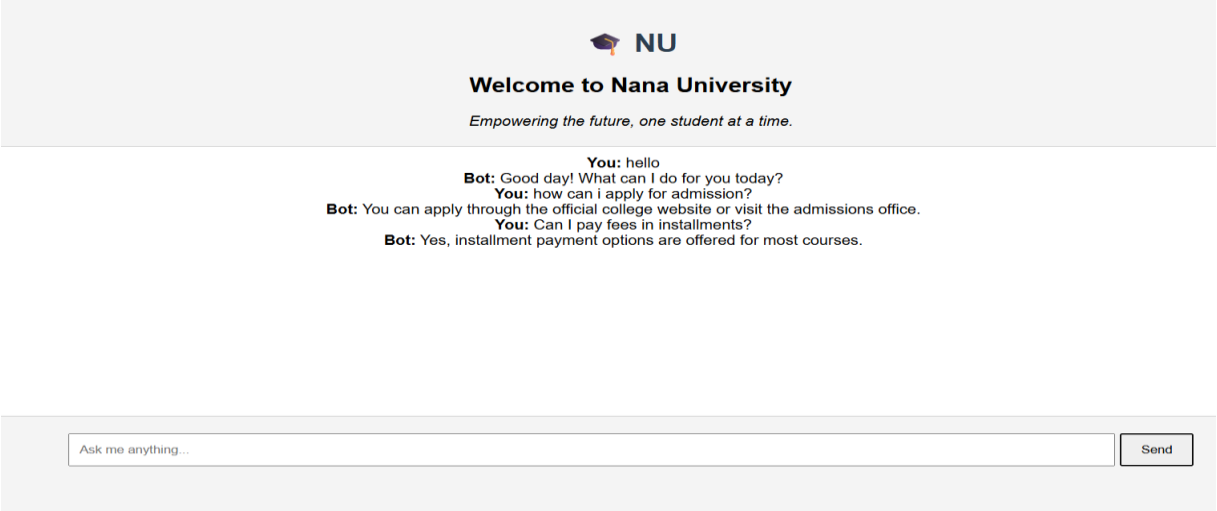
After uploading:

3. Installed required Python libraries:
   pip3 install flask google-cloud-dialogflow-cx
4. Ran the Flask app using:
   python3 app.py
5. Opened Preview → Port 8080 to access the chatbot.
6. Granted IAM roles to the service account to resolve 403 errors.
7. Confirmed real-time working responses from Dialogflow CX.

7. SAMPLE TEST RESULT:

User: Hello

Bot: Hello! How can I help you today?



8. Conclusion

The chatbot for Nana University was successfully developed and integrated with Dialogflow CX. It is now able to respond to student queries in real time with a functional frontend and backend. The system is ready for further expansion such as hosting, advanced NLP, or database integration.