Journal of Research High School

Vol. 2025 (3) 137 – 144
ISSN 2688-3651 [online]

JRHS Outstanding Research Paper Award

Quick Development of Chatbot Apps with OpenAI and Public Cloud Services

Yunho Kim[1] *

[1] North Gwinnett High School, Suwanee, GA, USA
*Corresponding Author: xyryan1974@gmail.com

Advisor: Dr. Bongkyoung Kwon, bkwon1@ggc.edu

Received June 30, 2024; Revised November 18, 2024; Accepted December 11, 2024

Abstract

Small business events, non-profit organization events, and many small conference events are still struggling to organize their events due to their limited budget and the lack of IT skills. The chatbot could be one of the potential solutions because it can guide event participants without hiring human resources to interact with potential and actual customers. The chatbot powered by the generative AI are used in a variety of industries like the customer service, e-commerce, healthcare, education, and so on. To use the chatbot with these events, different categories of artificial intelligence were reviewed to determine the chatbot's classification. Research was conducted to identify efficient and stable methods for implementing the chatbot. To facilitate easy and rapid development, the Python programming language was selected, along with the Flask, LangChain, and Zappa Python libraries, which were chosen for web interface development, integration with the existing large language model (LLM), and AWS deployment, respectively. The significance of LLM and LangChain to the chatbot application was analyzed. The chatbot was tested with two websites and deployed in the public cloud using AWS. Additionally, a price range for the chatbot application was provided based on the gpt-3.5-turbo model in OpenAI, along with estimates of potential usage volume. This chatbot application aims to help non-profit organizations reduce their workload and budget for customer communication.

Keywords: Chatbot, Conversational AI, OpenAI, AWS, LangChain

1. Introduction

Ever since A. M. Turing introduced the concept of machine learning that can simulate human intelligence by the famous concept, the “Turing Test”, artificial intelligence (AI) has shown its potential, but many people have been suspicious because of the lack of understanding, technical limitation, negative influence by media, ethical and security concerns, and so on (Turing, 1950). AI refers to the ability of machines to perform tasks that typically require human intelligence, such as learning, problem-solving, and understanding language. AI systems use data and algorithms to make decisions or predictions based on patterns they recognize in the data. So, AI is quickly becoming a game changer in many industries in recent years by the generative AI, and OpenAI’s ChatGPT is a great example (Luca Petriconi, 2022). A chatbot is an application of AI that interacts with users through text or speech. It is designed to simulate human conversation and can answer questions, provide recommendations, or complete tasks, such as booking appointments. Chatbots are commonly used in customer service, websites, and messaging apps to assist users quickly and efficiently.

1.1 AI Categories

This section provides an overview of the different types and classifications of AI from the perspectives of

Vol. 2025 (3) 137 – 144
ISSN 2688-3651 [online]

capability, functionality, technology, and business, and identifies the type and classification to which the chatbot application belongs, as shown in Figure 1. Simplilearn classified AI into several types based on capabilities, functionalities, and technologies (Simplilearn, 2024). Based on Capabilities, AI can be classified into Narrow AI, General AI, and Super-intelligent AI. Narrow AI known as Weak AI performs any intellectual task that a human can do but focuses on a particular function. Self-driving, image recognition, and the chatbot are examples of narrow AI. General AI known as Strong AI is like human cognitive abilities and aims to possess the ability to understand, learn, and apply knowledge across a wide range of tasks and domains. Super-intelligent AI represents a future form of AI where machines could surpass human intelligence across all fields, including creativity, general wisdom, and problem-solving. It is for now a concept and not yet realized.

AI can also be classified as Reactive Machines, Limited Memory, Theory of Mind, and Self-aware AI based on memory capabilities. Reactive machines can only react to current scenarios and cannot use past experiences to influence present decisions, so it behaves without any memory. However, limited memory AIs can store and use historical data for a limited time to make better decisions, and many machine learning models like recommendation systems, self-driving cars and the chatbot fall into this category. Theory of mind can understand human emotions, beliefs, and thoughts, but it is not yet fully realized. Self-aware AI is like the robot in SF movies. It has their own consciousness, self-awareness, and understanding of their own existence, but it does not yet exist. AI can also be classified Machine Learning, Deep Learning, Natural Language Processing (NLP), Robotics, Computer Vision, Expert System by technologies. Machine learning makes computers learn from and make predictions or decisions based on data instead of being explicitly programmed to perform tasks. Deep learning is a subset of machine learning and use neural networks to model complex patterns in large datasets. NLP will be explained in detail since the chatbot application belongs to this category. Pecan team explains generative AI, conversational AI, and predictive AI from the business perspective (Pecan Team, 2023). Generative AI creates new content and is useful for marketing, design, and entertainment, and Conversational AI enhances business interactions through chatbots and virtual assistants. Predictive AI anticipates future outcomes based on historical data, helping businesses make data-driven decisions. In the conversational AI, Cem Dilmegani showed two types of solutions: Intelligent virtual assistant and chatbot (Cem Dilmegani, 2024). Both solutions are the same in terms of answering FAQs, but chatbot is a rule-based solution, so it has a lack of ability to learn with experience, and a higher failure rate for complicated tasks as well. Merav presented a comprehensive overview of the technology and vision, future potential, and ongoing challenges in the field of conversational agents (Merav, et al., 2021). Even though chatbot has limitations, it was chosen as the solution because many open-source projects can be referenced, and it is easy to implement for the purpose of answering FAQs without having a help desk. Figure 1 shows which category the chatbot application belongs to in AI. The next section explores Natural Language Processing (NLP), Large Language Model (LLM), and GPT-3.5-Turbo from ChatGPT.

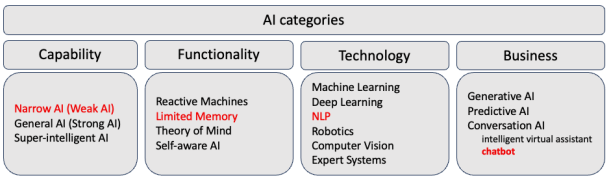

Figure 1. Which category the chatbot application belongs to.

1.2 NLP, LLM, and GPT-3.5-Turbo

In this section, NLP, LLMs, and GPT-3.5-Turbo, developed by OpenAI, are explained in simple terms to clarify their relationships, as these terminologies can often cause confusion. NLP focuses on the interaction between computers and humans through natural language in the AI field, and its goal is to enable computers to understand, interpret, and generate human language. It uses either rule-based or machine learning approaches to understand the structure and meaning of text and plays an important role in the chatbot application, voice assistants, text-based scanning programs, and so on. LLM is built upon fundamental NLP concepts and techniques. Text analysis, language translation, text summarization, speech recognition, and question answering are examples of LLM and NLP. Tom B. Brown introduced GPT-3, a powerful LLM, and demonstrated its ability to perform various tasks with few-shot learning (Tom, et al., 2020). GPT-3.5-Turbo is an example of LLM released by OpenAI in November 2022. It is a more efficient and cost-effective version of the GPT-3.5 series. After GTP-3.5 series, GPT-4 were released in March

Vol. 2025 (3) 137 – 144
ISSN 2688-3651 [online]

2023, and OpenAI released GPT-4 recently in May 2024. In summary, the chatbot application uses GPT-3.5-Turbo due to its cost effectiveness. GPT-3.5-Turbo is an example of LLM built upon NLP technology.

2. Research and Design

The goal is to implement the stable chatbot application quickly based on the information the local event host provides, so the strategies are 1) using open-source libraries for front-end and backend, 2) using the Python LangChain framework and Open-AI API for LLM, and 3) deploying the chatbot into a public cloud for stable running and cost-efficiency by serverless platform. Python was selected as the programming language due to its ease of learning and use, along with its extensive collection of open-source libraries. Flask, LangChain, and Zappa were chosen for the backend server, large language model integration, and AWS deployment, respectively.

2.1 Python libraries

LangChain

LangChain is a framework designed for developing applications using LLMs (Python LangChain, 2023). It provides tools and methods to simplify the integration of LLMs. Since LangChain acts as the glue that connects different parts of a smart application, it can help application to manage conversations, look up information, and combine responses easily. Due to its easy integration feature with LLMs, LangChain is the most important open-source library in the proposed chatbot. Figure 2 shows the high-level features of LangChain and LangChain-community libraries. The features in red are used in the proposed chatbot application. The RetrievalQA component from LangChain, along with the WebBaseLoader, CharacterTextSplitter, and FAISS components from the LangChain-community library, were utilized.

RetrievalQA combines information retrieval with question answering. The main idea of RetrievalQA is to first retrieve relevant information from a large collection provided and use this retrieved information to generate an accurate and contextually relevant answer so that it matches the purpose of the defined scenario. The WebBaseLoader component of the document loader feature from the LangChain-community was used because it can load contents from single or multiple web pages, it is the easiest interface between us and event organizers. Once the content is loaded into the web-based loader, it is converted into a format that is compatible with the rest of the LangChain processing pipelines. This allows for seamless integration with other modules for further processing, such as text analysis and summarization. It could also feed into the chosen language model.

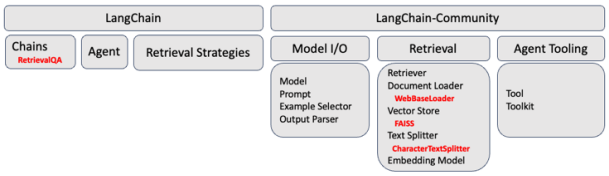

Figure 2. LangChain library features

CharacterTextSplitter is a utility in LangChain-community, and it helps in splitting text into smaller chunks based on character count. It is designed to break down a large body of text into smaller and manageable chunks, ensuring that each chunk does not exceed a specified maximum character length. This helps in managing memory and computational resources more effectively. In this case, A chunk size of 1,000 was set to split the website contents loaded by the WebBaseLoader component. FAISS (Facebook AI Similarity Search) is a library developed by Facebook AI Research. It is a robust and versatile tool that offers powerful similarity search capabilities for a wide range of data types and applications. This library from the LangChain-community was utilized.

Flask

There are three popular python web frameworks: Flask, FastAPI, and Django. First, Django was excluded due to its heaviness. Flask was chosen primarily because it is a simple and light micro-framework that helps build web apps quickly and are familiar with it (Python Flask, 2024). It fits well into small to medium-sized projects and has a large and supportive community. It also works with the Zappa library that will be explained in the next section. FastAPI could be the solution because it is simple to use and offers modern features like API document generation and

Vol. 2025 (3) 137 – 144
ISSN 2688-3651 [online]

asynchronous communication; however, the decision was made to minimize the number of required Python packages. If the chatbot needs to support a high volume of requests in the future, the FastAPI library may be considered.

Zappa

Zappa is a python library for deploying applications into a serverless platform called AWS Lambda (Python Zappa, 2024). AWS Lambda is a cloud service from Amazon Web Services (AWS) that run your code without having computer servers. With AWS Lambda, you don't need to set up or manage any computer servers to run that code. Instead, it takes care of it without servers. Zappa also creates AWS API Gateway to trigger the Lambda by Representational State Transfer (REST) API endpoints or web browsers. A REST API is like a messenger that lets different apps and websites talk to each other and share information. So, Zappa allows us to deploy, update, or roll back the chatbot application on AWS Lambda using Zappa's command line interface. In addition to its easiness, it can reduce costs because payment is required only when the chatbot application is executed. In other words, there are no charges if the chatbot application is not used. Zappa was selected because AWS Lambda offers additional advantages, such as eliminating extra maintenance and allowing integration with other AWS resources like Simple Storage Service (S3), AWS Relational Database Service (RDS), and so on.

2.2 OpenAI API

The OpenAI API is the other core component in the chatbot architecture design because it provides us with the gate to access OpenAI's powerful LLM model. The API supports various models, including as GPT-3, GPT-4, and GPT-4o that was released in 2024. This GPT series is a powerful tool that can read, understand, and generate text, almost like a human. Because of various series of LLM models, there is flexibility in choosing the optimal balance between performance and cost. Since access to the API is via RESTful endpoints, it supports easy integration to various programming environments and enables a wide range of applications, from chatbots, content generation to data analysis and more. To use the OpenAI's existing models, an account was created in OpenAI, an API key was generated, and it was configured in the chatbot application. For the used LLM model, GPT-3.5-turbo (specifically, gpt-3.5-turbo-instruct) was selected primarily due to its cost-effectiveness compared to GPT-4. The price model of GTP-3.5 is shown as the following.

Table 1. GPT-3.5-turbo price model

Model	Input price	Output price
gpt-3.5-turbo-0125	$0.50 / 1M tokens	$1.50 / 1M tokens
gpt-3.5-turbo-instruct	$1.50 / 1M tokens	$2.00 / 1M tokens
gpt-4.0o	$5.00 / 1M tokens	$15.00 / 1M tokens

As you see the table, API usage cost is calculated with tokens. A "token" is a fundamental unit of text used for processing and generating language. Input tokens are for processing, and output tokens are for generating. A token can be as short as one character or as long as one word. For example, considering the sentence: "we built a chatbot application", it could be tokenized into: "we", "built", "a", "chatbot", "application". These tokens are processed by LLM models. In this case, website content is provided as input, and gpt-3.5-turbo-instruct generates output tokens.

3. Design and Implementation

3.1 Design

Based on the defined strategy and research, the implementation process was outlined, and the chatbot application architecture was designed. As explained in the previous section, the decision was made to use the Flask framework, which can run on AWS Lambda with minimal maintenance and low cost. For LLM model, the python LangChain framework was used to integrate with the powerful LLM model through OpenAI APIs. For easy deployment into AWS Lambda, The Zappa library was selected with pre-configured AWS credentials. Figure 2 illustrates the overall implementation model. After implementing the chatbot, initial testing was conducted locally against two existing websites, and the chatbot was repeatedly fixed until all tests passed. Once the local test conditions were satisfied, the chatbot was deployed into AWS Lambda and tested via the AWS API Gateway endpoint.

Vol. 2025 (3) 137 – 144
ISSN 2688-3651 [online]

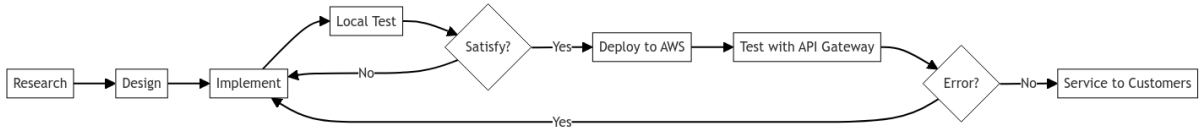

Fig 3. Implementation process

Figure 3 shows the designed chatbot application architecture, where the final chatbot application is deployed to AWS Lambda using Zappa commands. Then, the chatbot is ready with API gateway within 10 seconds. From this moment, customers use the chatbot through the AWS API Gateway endpoint. When customers type the endpoint in their browser, the chatbot returns the defined html template to the browser, where they can type a question. Once the question is submitted, the chatbot uses OpenAI's built-in model by the registered OpenAI API key and returns the answer to the browser. Customers can keep asking their questions and all the answers will accumulate in the browser until they refresh or close their browser.

3.2 Implementation and Local Test

For implementation, Python version 3.9.6 was first installed, and a virtual environment was configured on the development laptop, then used pip command to install all the necessary python packages like LangChain, OpenAI, Flask, Zappa, and so on. As the final step before deploying the chatbot into AWS Lambda, a system environment variable with the OpenAI API Key was configured. After running the chatbot application locally, it ran with the port number, 5000 as shown in (a) of Figure 4. In this case, it was observed that the previous question and answer disappeared when a new question was typed, so the HTML template was modified based on references from paramgit's GitHub repository, which stored all the questions before customer's browser was closed. Testing was conducted a small event website, "https://stepup.ksea.org/ check-list" to check the accuracy of the chatbot application. Additional testing was performed with larger content from https://python.langchain.com/v0.1/docs/get_started/quickstart. During this testing, the chatbot showed the error of the requested token is larger than the maximum context length that was set by default in the library as shown in (c) of Figure 4. To resolve this issue, two new variables, chunk_size and chunk_overlap, were added to the CharacterTextSplitter method. Various questions were used to evaluate the chatbot's performance on this complex website. Figure 4 shows the progress and fixation of the local test.

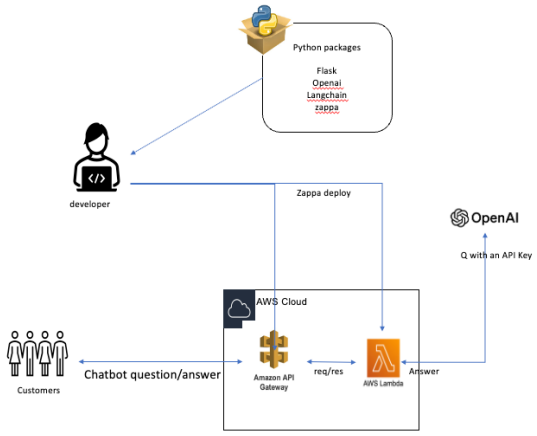

Figure 4. The Chatbot Architecture

3.3 Deployment

As previously mentioned, Zappa was chosen for deployment. Initially, a new AWS account was created, and credentials were configured to allow Zappa to deploy the code to AWS Lambda. The scope of the credential is only for AWS Lambda and API Gateway. Only two Zappa commands, zappa init and zappa deploy, were required to configure the AWS target region (us-east-2) and deploy the chatbot to AWS Lambda, respectively. When it was initially deployed, an error was encountered indicating that the package size exceeded the maximum limit for AWS Lambda. When the finalized chatbot codes were deployed, it was realized that AWS Lambda has a deployment package size limit of 50 MB for zipped files or 250 MB for unzipped files. Since Zappa compresses the required python library packages, the final deployment file was big, but Lambda provides the layer feature, by which the dependent library packages can be moved into the Lambda layer, so that only application can go to Lambda code section. Another solution was to use the slim handler variable to true provided by Zappa as shown in Figure 6(1). The slim handler option was used in the chatbot application because of its easiness. AWS Lambda has another limitation

Vol. 2025 (3) 137 – 144
ISSN 2688-3651 [online]

of running only 15 mins when it was invoked, but it did not impact the chatbot application because its response time was within a couple of second. Finally. Figure 6(b) shows the chatbot application running on AWS Lambda by the Zappa command and it was verified by the AWS API gateway and a web browser.

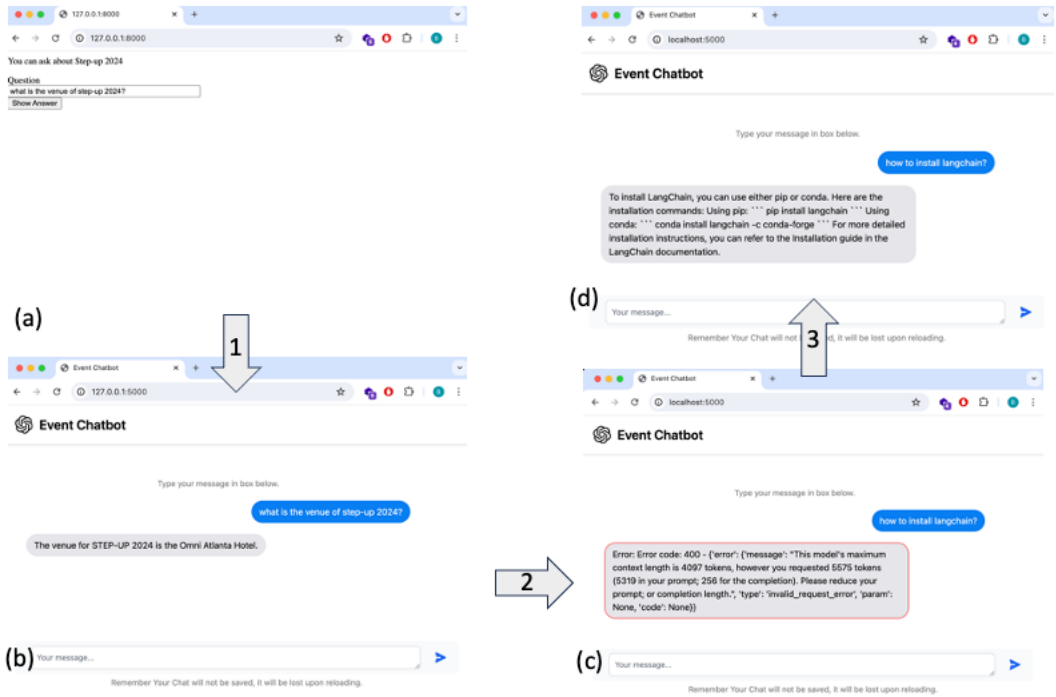

Figure 5. Test against the locally run chatbot application.

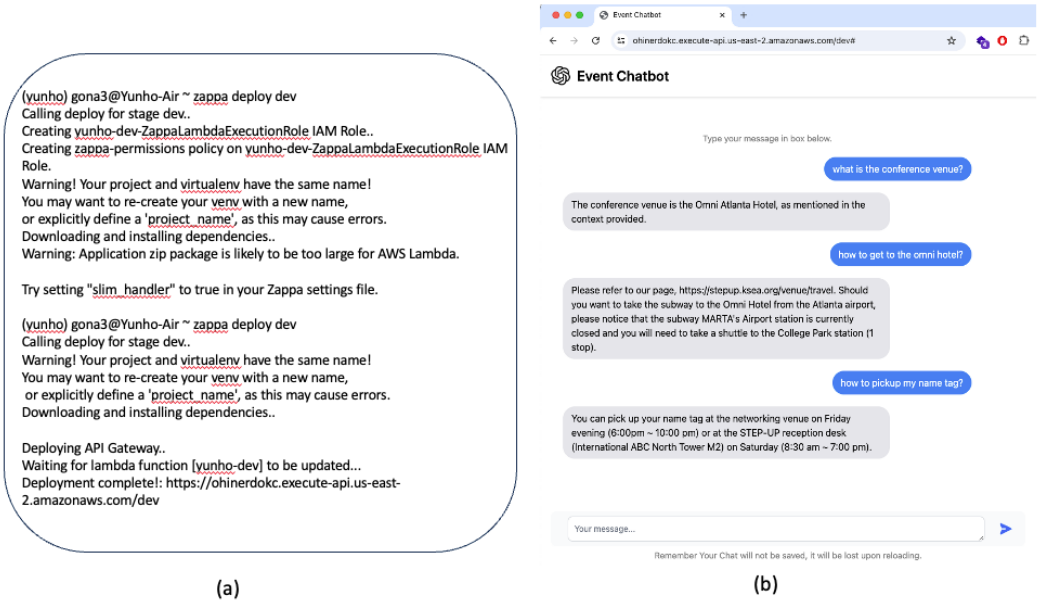

Figure 6. Deployment to AWS and Verification

4 Discussion

4.1 Cost

The cost of using OpenAI's LLM model and AWS resources for the chatbot needs to be considered, as non-profit

Vol. 2025 (3) 137 – 144
ISSN 2688-3651 [online]

organizations typically operate with limited budgets. The cost of the chatbot in AWS can be disregarded because AWS Lambda offers a free tier: 1 million free requests per month and 400,000 GB-Seconds of compute time per month. For example, if the chatbot uses 128 MB of memory and runs for 2 seconds, it consumes 256MB-Seconds of compute time, and eventually can be used 1.6 million times per month. The cost of AWS API Gateway is a few dollars per million requests. Table 2 presents the cost estimation of OpenAI API usage based on testing (50 requests) and the API pricing described in Chapter 2. This estimate is solely based on the test data. This estimate is just based on the test; the real cost will vary based on the real website content, questions, and answers.

Table 2. Cost Estimation of OpenAI API Consumption.

Case	Input token	Input cost	Output token	Output cost
A testing (50 requests)	36,315	36,315/1,000,1000 * $1.5 = $0.054	2,095	2,095/1,000,1000 * $2.0 = $0.00419
Estimation (10,000 requests)	7,263,000	7,263,000/1,000,000 * $1.5 = $10.89	419,000	419,000/1,000,000* $2.0 = $0.838

According to the table, the estimated monthly cost for a small business, with customer queries ranging from 100 to 500 per day, is roughly $3 to $20. For a non-profit organization with visitor queries between 1,000 and 3,000 per day, the monthly cost could be approximately $30 to $120. These estimates demonstrate how the chatbot system can serve as a cost-effective alternative to hiring additional staff, benefiting both small businesses and non-profits. However, this analysis is theoretical, and actual costs may be higher depending on the complexity and variety of questions. Potential cost increases will be addressed in future work.

4.2 Future enhancement

Three primary areas have been identified for improving the chatbot application: implementing user authentication and authorization, exploring free large language models (LLMs), and enhancing the user interface with Streamlit python framework. Since managing operational costs is a key priority, unrestricted public access to the chatbot could lead to high usage costs when many users engage with the application in some non-profit organizations. Introducing authentication would allow only authorized users to access the chatbot after logging in, helping to control usage and manage expenses effectively. For additional cost saving, free LLM options, such as BLOOM, GPT-Neo (via Hugging Face), and Meta AI's LLaMA can be implemented. For easier interface UI change, Steamlit python framework can be tried, but the framework cannot be deployed into AWS Lambda service due to its session management, so a different AWS service will be researched.

5 Conclusion

AI is quickly becoming a game changer in many industries in recent years by the generative AI. A chatbot is an example of generative AI. This research categorizes the chatbot application within the different types of AI and shows the importance of large language models (LLMs) and LangChain as foundational components. The integration of the OpenAI API, the python LangChain library, and AWS serverless architecture allowed for rapid, scalable deployment with minimal ongoing maintenance. The testing phase demonstrated the chatbot's capability to handle various queries effectively and showed that the chatbot application can reduce the operational expenses of small businesses and non-profit organization. The importance of this research lies in its demonstration of how AI can do customer engagement with the chatbot application for resource-constrained organizations. Future enhancements include exploring alternative Python frameworks and free LLM options to further optimize accessibility and cost efficiency.

References

Allouch, M., et al. (2021). *Conversational Agents: Goals, Technologies, Vision, and Challenges*. Sensors. MDPI

AWS API Gateway Developer Guide. (2024). *Amazon Web Services*. https://docs.aws.amazon.com/pdfs/apigateway/latest/developerguide/apigateway-dg.pdf

Vol. 2025 (3) 137 – 144
ISSN 2688-3651 [online]

AWS Lambda Developer Guide. (2024). *Amazon Web Services*. https://docs.aws.amazon.com/pdfs/lambda/latest/dg/lambda-dg.pdf

Brown, T. B., et al. (2020). *Language Models are Few-Shot Learners*. Advances in Neural Information Processing Systems (NeurIPS)

Dilmegani, C., (2024), *Chatbot vs Intelligent Virtual Assistant: Comparison in 2024*. AI Multiple Research. https://research.aimultiple.com/chatbot-vs-intelligent-virtual-assistant/

IBM topic. *What is artificial intelligence (AI)?*. IBM. https://www.ibm.com/topics/artificial-intelligence

Pecan Team (2023). *The Right AI: Generative, Conversational, and Predictive AI for Business*. Pecan. https://www.pecan.ai/blog/generative-conversational-predictive-ai-compared/

Petriconi, L., (2022). *Google is done. Here's why OpenAI's ChatGPT Will Be a Game Changer.* Medium. https://medium.com/@lucapetriconi/google-is-done-heres-why-openai-s-chatgpt-will-be-a-game-changer-98ae591ad747

Python Flask. (2024). *Flask community*. https://flask.palletsprojects.com/en/3.0.x/

Python LangChain. (2023). *LangChain Introduction*. https://python.langchain.com/v0.2/docs/introduction

Python Streamlit (2024). *Snowflake*. https://docs.streamlit.io

Python Zappa (2024). *Pypi.org*. https://pypi.org/project/zappa/0.59.0/

Simplilearn (2024), *Types of Artificial Intelligence That You Should Know in 2024*. Simplilearn. https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/types-of-artificial-intelligence

Turing, A. M., (1950). Computing Machinery and Intelligence, *Journal of Mind*, 49: 433-460.