



Project Report

On

KothaGPT: The Accessible ChatGPT for Bengali Users

Submitted To:

Mimnun Sultana

Assistant Professor

School of Business & Economics,

United International University

Submitted By:

Kazi Nahian Isfar Omar

111 202 167

Spring 2023

School of Business & Economics,

United International University

Date of Submission: 18.06.2023

Letter of Transmittal

Mimnun Sultana

Assistant Professor

School of Business & Economics

United International University

Subject: Letter of Transmittal

Dear Ma'am,

With great pleasure, I am submitting you this report, **“KothaGPT: The Accessible ChatGPT for Bengali Users”** on my built project.

While conducting this project, I have learned and gathered immense technical knowledge and skills on how to build a natural language processing system presented on a user interface. I was able to challenge myself and gain expertise to facilitate my future career life.

It is to be strongly mentioned, that without your guidance and support this project would not have met its completion successfully. I hope this report will cover the intricate details, joy and challenges of implementing my project thoroughly.

Sincerely,



Kazi Nahian Isfar Omar

111 202 167

Student Declaration

I hereby declare that this project report titled, “**KothaGPT: Accessible ChatGPT for Bengali Users**”, submitted to the United International University is my authentic work. This report is being submitted to partially fulfill the requirements for the degree, Bachelor in Business Administration (BBA) with the assistance of Mimmun Sultana, an assistant professor at United International University. Although many research papers, journals and online resources have been used as reference, the subject matter and details provided in this report is part of my original work.

Kazi Nahian Isfar Omar

Kazi Nahian Isfar Omar

ID: 111 202 167

Major: Management Information System (MIS)

United International University

Acknowledgement

When it comes to such a lengthy composition of works, it cannot be done without the support of others. There have been many people to whom I wish to convey my gratitude.

First of all, I would like to express my utmost gratitude to my supervisor, Mimmun Sultana Ma'am for never-ending guidance, enlightenment and directives to finish this project report.

Secondly, I would, of course like to thank the Almighty and my family for believing in me. Without their immeasurable support and motivation, this project would not have finished.

Lastly, and most importantly, I wish to convey my gratitude to my peers and colleagues who chose to share their time, knowledge, expertise and skills with me throughout challenging times.

Despite my best efforts, this project report may consist of several inconsistencies and fallacies due to lack of skills. My sincere apologies, and I hope the purpose and efforts serve to prove my hard work.

Executive Summary

This report covers the making of a natural language processing system presented as a web application. It deals with a Bengali user who provides a Bengali input (a question) in the form of speech which is translated by the application to English, and fed into ChatGPT. In return, the response collected from ChatGPT is then translated to Bengali in text and speech to be visible to the Bengali user. In the first chapter, Introduction, it discusses the problem we are trying to solve with my application, the methodologies used and the objective of the study. In the second chapter, Literature Review, I explain and cite the used research papers, journals and articles used as reference to build my project report. In the most important chapter, Findings and Implementation, I elaborately discuss the steps, workings and code produced to create the functional application. Afterwards, I move on to provide an overall evaluation of the application in use, prove its validation, mention its constraints and provide recommendations for future self to work on. Lastly, I end the report with a conclusion and overview of this complex project work.

Table of Contents

Letter of Transmittal	1
Student Declaration	2
Acknowledgement	3
Executive Summary	4
1. Introduction	6
1.1 Background of the study	6
1.2 Objectives of the Study	11
1.3 Methodology of the study	12
2. Literature Review	12
3. Findings & Implementation	15
3.1 Language Models.....	20
3.2 ChatGPT API	22
3.3 Integration of Language Models with ChatGPT.....	24
3.4 Speech Recognition Models	26
3.5 Web Application.....	28
Main.js	30
Index.html	32
Style.css	34
App.py	38
Chat.py	39
4. Validation	42
4.1 Limitations	47
4.2 Recommendations.....	48
5. Conclusion	48
6. References	50

1. Introduction

1.1 Background of the study

When we think of the word “chatbot”, we think of the recently introduced AI chatbot developed by OpenAI, ChatGPT. However, there falls a long history behind the development of chatbots, without which the transformative ChatGPT would not have seen face.

It goes way back to 1950s, when an English Mathematician, Alan Turing conceptualized the idea of intelligent machines. Such a concept led to the development of the first ever chatbot in 1966 called ELIZA. It is said to be the landmark for further research in this field (*Caldarini, Jaf, & McGarry, 2022*). Another notable chatbot was invented in the 1980s was A.L.I.C.E (Artificial Intelligent Internet Computer Entity). The biggest difference between ELIZA and A.L.I.C.E was A.L.I.C.E used natural language processing which provided smarter conversations with the user. Gradually, new architectures and methods were emerging for chatbots.

In 2010s, we were introduced to various personal assistants. Apple Siri and Alexa were few of the firsts with particular functionalities such as looking up facts, checking weather and navigation or even initiating home automation etc. Much later in 2016, Google Assistant was established. It was evolutionary as this chatbot brought two relevant changes. It was connected to the Android operating system and the Google Nest. The application is capable of accessing information through text or voice input to perform various actions (*Černý, 2022*).

Chatbots are made with large language models which have the ability to perform abundant natural language processing activities. It is vital to discuss what is natural language processing and its evolvement over the years. According to a journal paper in 2001, natural language processing is to achieve human-like language processing. Back then, it was aimed to paraphrase an input text, translate that text into another language and answer questions about the contents etc. Aside from the first task, it wasn't able to perform the rest unlike now. The field is so wide that it is divided into 2 sub-fields, language processing and language generation. In simple terms, natural language processing is similar to the role of a reader/listener and natural language generation is similar to the role as a writer/speaker (*Liddy, 2001*). Machine translation is known to be the first computer-based application aligned to natural language processing, in the early 1950. In 2011, it was stated

that most of the above tasks that could not be performed initially can be performed easily, however, question-answering remained difficult (*Nadkarni, Ohno-Machado, & Chapman, 2011*).

On the topic of machine translation, although it was discovered during the late 1940s (early 1950s), the notable machine translation systems were made during the 1970-1980s. These include SYSTRAN, Logos and METAL which were required to be operated on large mainframes for government or military institutions only. Regardless, not until the mid-2000s did machine translation truly sparked an interest for researchers. This was happening due to improvement of Wifi networks and accessibility of Internet enabled devices. With the invention of Google Translate, everyone was able to discover its implications and usage (*Jolley & Maimone, 2022*). We can say machine translation is needed to automatically translate from one language to another, especially neural machine translation. It is a fundamental task in natural language processing. With the development of large language models like GPT-3 and ChatGPT, machine translation is making significant improvement lately with deep learning. Despite that, machine translation still faces difficulties in dealing with low-resource translations, handling uncommon words and keeping with fluency in the translation (*Lyu, Xu, & LongyueWang, 2023*). Even back in 2017, while it was showing remarkable performance on public benchmarks, it had still shown poor quality performance with low-resource languages (*Koehn & Knowles, 2017*).

Large language models assist to perform various natural language processing tasks. It has helped in the advanced for artificial intelligence over the years. A well-known large language model is ChatGPT by OpenAI, which has shown to perform impressive tasks like providing practical responses to human input, dismissing misleading or inappropriate questions and correcting its errors (*Qin et al., 2023*). Large language models are usually trained on a vast amount of text data, to show fascinating capabilities in terms of in-context learning (replication of specific input-output text generation patterns, without further updates to its parameters) which significantly improves live adaptive machine translation (*Moslem, Haque, Kelleher, & Way, 2023*). In depth, we must focus on the transformers architectures that helps in maintaining high quality performance of language models, controlling the long-range dependencies in natural-language texts. It basically allows the language model to perceive the relationship between words in a sentence despite its position (*Kasneci et al., 2023*). Models like ChatGPT and BERT are strongly relying on this

architecture, because the transformers architecture is capable of revolutionizing the natural language processing systems easily (*Kasneci et al., 2023*).

Hence, the technology advances have gradually welcomed ChatGPT (Chat Generative Pre-trained Transformer), the virtual assistant implemented with sophisticated machine learning methods and natural language processing (*King, 2023*). Various chatbots previously had integrated language models, but ChatGPT is now a better conversationalist with the user due to the incorporation of reinforcement learning. It should be mentioned that ChatGPT was built over GPT-3.5, relatively low performance language model with 175 billion parameters. According to reports, OpenAI partnered with Microsoft with a training resource of more than 350GB of memory costing \$12 million (*Shahriar & Hayawi, 2023*). It is necessary to discuss the stagnant differences, features and upgrades of GPT-3.5 and ChatGPT. As written, while ChatGPT generally is capable of performing multiple tasks, it fails to perform as well as other large language models which are often fine-tuned for specific tasks. In arithmetic reasoning, ChatGPT can perform tasks with superior performance. However, it falls behind in commonsense and logical reasoning in comparison to GPT-3.5. Despite that, ChatGPT remains superior in performing natural language inference and dialogue tasks, but underperforms when it comes to summarization tasks by providing longer summaries than GPT-3.5. While they both face issues when it comes to sequence tagging, they both perform similarly in terms of sentiment analysis.

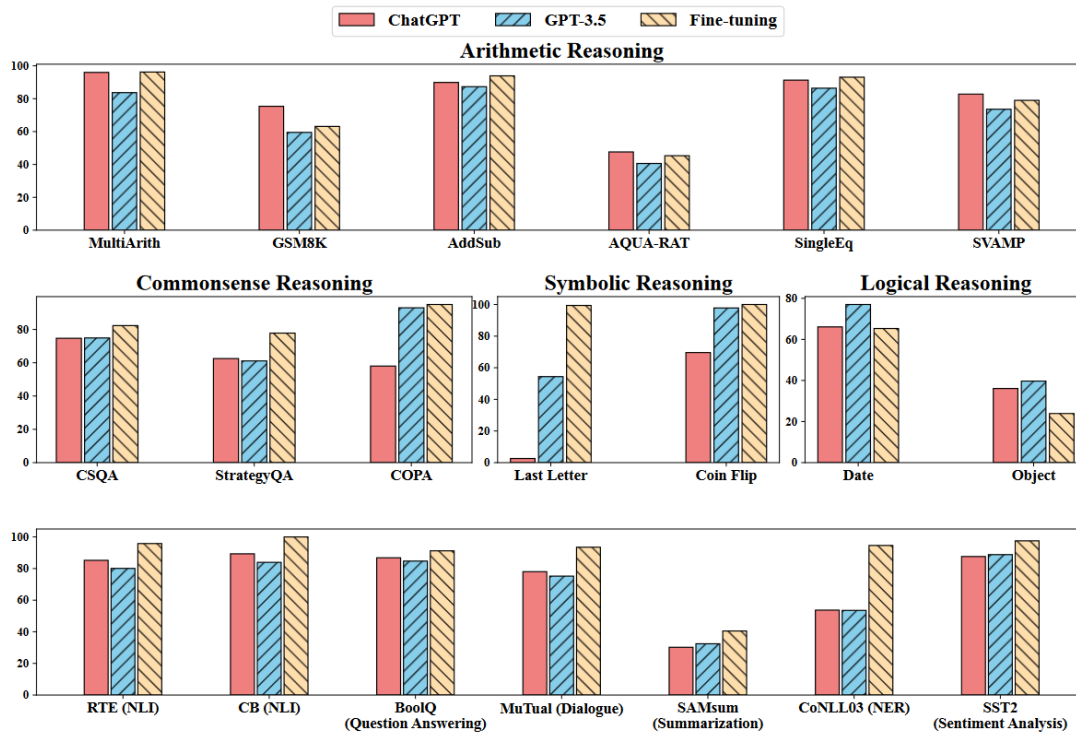


Figure 1.0: ChatGPT, GPT-3.5 and other fine-tuned models in comparison, with specific tasks given from 20 different datasets

ChatGPT

Q: Wendy was playing a video game and had 43 lives. In a hard part of the game she lost 8 lives. If she got 39 more lives in the next level, how many lives would she have?

A: The answer (arabic numerals) is

Response: 74 lives.

GPT-3.5

Q: Wendy was playing a video game and had 43 lives. In a hard part of the game she lost 8 lives. If she got 39 more lives in the next level, how many lives would she have?

A: The answer (arabic numerals) is

Response: 120 lives.

Figure 1.1: This is where we can ChatGPT corrects the mistake made by GPT-3.5.

ChatGPT

Q: f: dad , can you lend jenny and me some money ? m : for what ? f: we want to get tickets for a rock concert . we 'll pay you back after a few days . m : how much are these tickets ? f: \$ 18.00 each . m : \$ 18.00 ? that 's a lot of money . Which choice is correct ? Answer Choices: (A) f: actually , \$ 18 in total . (B) f: yes , all in all , two tickets cost \$ 44 . (C) f: yes , \$ 36 in total . but \$ 2 discount for two tickets , that is \$ 36 in total . (D) f: yes , \$ 36 in total .

A: Among A through D, the answer is

Response: D: f: yes, \$ 36 in total.

GPT-3.5

Q: f: dad , can you lend jenny and me some money ? m : for what ? f: we want to get tickets for a rock concert . we 'll pay you back after a few days . m : how much are these tickets ? f: \$ 18.00 each . m : \$ 18.00 ? that 's a lot of money . Which choice is correct ? Answer Choices: (A) f: actually , \$ 18 in total . (B) f: yes , all in all , two tickets cost \$ 44 . (C) f: yes , \$ 36 in total . but \$ 2 discount for two tickets , that is \$ 36 in total . (D) f: yes , \$ 36 in total .

A: Among A through D, the answer is

Response: C: f: yes, \$ 36 in total. but \$ 2 discount for two tickets, that is \$ 36 in total.

Figure 1.2: This is also where we can see ChatGPT answers right while GPT-3.5 answers wrong.

Label: Eric and Rob are going to watch a stand-up on youtube.

GPT-3.5: Eric and Rob are going to watch some of a comedian's stand-up on YouTube.

ChatGPT: Eric and Rob are discussing a stand-up comedy routine involving a machine, and they find it funny. They plan to watch more of the comedian's stand-up on YouTube and say "TTYL" before ending the conversation.

Label: Mark lied to Anne about his age. Mark is 40.

GPT-3.5: Anne found out that Mark lied to her about his age.

ChatGPT: Anne discovered that Mark, who she was dating, lied about his age and is actually 40, not 30 as he had previously claimed. Irene expresses sympathy for Anne's situation.

Figure 1.3: Comparison of summaries generated by ChatGPT and GPT-3.5 which shows ChatGPT gives longer summaries.

In conclusion, although ChatGPT is still facing a lot of challenges after the upgrade, it remains to be powerful generalist model (Qin et al., 2023). With this conversational AI, we are able to make applications, write complicated code, produce poetries, essays and quality stories replicating a certain author's writing style (Kung et al., 2023). Broadly speaking, ChatGPT helps in the research domain, by improving data analysis processes and interpretations and assisting in automating monotonous tasks for researchers. Further, it is restructuring the education sector, by letting users access educational resources and information and improve exam performance. Lastly, it has also been a helpful source for improving language skills, enhancing content creation, reducing digital illiteracy and providing language translations (Islam & Islam, 2023).

On the topic of language translations, the question is, 'Is ChatGPT capable of providing accurate translations of the detailed English responses generated by it?'. ChatGPT was already known to

be a good translator, with its launch of the GPT-4 engine it has been able to improve the translation performance significantly in comparison with even other translation products. However, according to studies, ChatGPT is said to perform on the same level as commercial translation products like Google Translate and DeepL on high-resource European languages but crucially lags behind on low-resource or distant languages (*Jiao, Wang, Huang, Wang, & Tu, 2023*). Another similar study tested the language understanding and language generating capability. In terms of language understanding, 2 tasks were performed: sentiment analysis (SA) and language identification (LID) which showed that for high-resource languages like English and Indonesian the SA accuracy were 84% and 80%, and the LID accuracy were both 100%. Whereas, the SA accuracy for a low-resource language like Buginese was 56% and the LID accuracy was 12%, a big difference. In terms of language understanding, ChatGPT was able to produce better quality translation for high-resource languages like French and Chinese but for low-resource languages like Japanese and Sudanese it generated mistranslated words/phrases (*Bang et al., 2023*).

Hence, we may understand that for a low-resource language such as Bengali ChatGPT underperforms. This becomes a big barrier for Bengali users who are unable to read, write or speak English. Upon testing, even with simple prompts like ‘How are you?’, it misinterprets the words and provides completely inaccurate translations. The report captures to solve this problem by making an interface for Bengali users to easily access the sophisticated virtual assistant, ChatGPT.

1.2 Objectives of the Study

There are many objectives behind conducting this project. The different objectives are:

- To investigate the language translation capabilities and incompatibility with Bengali language of ChatGPT
- To build a functional and easy to use interface to provide Bengali smartphone users a platform to take advantage of various educational resources and information via ChatGPT
- To provide access to up-to-date methods and processes, such as a doctor from a rural area in Bangladesh can find out globally used remedies to cure a patient

1.3 Methodology of the study

The first step of going about this project was to use and test ChatGPT's performance, specifically translation performance. Upon investigation, I was able to confirm that ChatGPT provided inaccurate and misleading Bengali translation. To educate myself, I read through many secondary sources such as review articles and papers via google scholar.

To be noted that, all the code snippets and backend of the application are performed on the open-source software, Jupyter Notebook, with the programming language Python. I had planned to use Google Translate's Application Programming Interface (API) for the translation data. However, the API is not accessible anymore, which led me to explore the open-source AI community, Hugging Face. It provides tools for building applications using machine learning but most notably its Transformers library is built for natural language processing applications, also allowing many users to share machine learning models and datasets. I was able to discover 2 pre-trained models named 'csebuatnlp/banglat5_nmt_bn_en' and 'csebuatnlp/banglat5_nmt_en_bn' which provided relatively better translation from Bengali to English, to English to Bengali (Bhattacharjee, Hasan, Ahmad, & Shahriyar, 2022).

Strongly aligned with deep learning and neural networks, we are able to feed Bengali speech to our first pre-trained model, convert it to English to feed into ChatGPT which provides the designated response (output), which is lastly converted into Bengali text to speech by the second pre-trained model. Amidst this, ChatGPT's API is called out on Jupyter Notebook to work and experiment with. Lastly, an interface, a web application usable both on mobile and pc is made. It is made using programming languages such as HTML, CSS and Javascript on Visual Studio Code, another form of source-code editor.

This report represents the steps, details and in-depth processes of the making of this interface with an organized outline and structure.

2. Literature Review

A literature study was conducted to analyze the advancement of chatbots over the decades and its incorporation of Artificial intelligence and machine learning (Caldarini *et al.*, 2022). Similarly,

another history explores the evolvement of chatbots based on other studies, defining the development of chatbots from 3 perspectives. On the other hand, another journal review discusses the capabilities of ChatGPT (King, 2023). Following the discussion of ChatGPT's benefits, a paper states that ChatGPT was built on another model (GPT-3) which was a low language performance model to begin with and continues to emphasize on the limitations of the current version (Shahriar & Hayawi, 2023). Further, many other journal articles have been written focusing on ChatGPT's strengths and weakness based on first impressions of using it and testing out its many possible applications in digital humanities and museology, aside from making stories and writing long snippets of code (King, 2023). Lastly, another paper continues to elaborate the various opportunities and challenges of ChatGPT in different sectors like research and personal skill development may lead to overall reducing digital literacy and providing language translations (Islam & Islam, 2023). However, several research studies argue that ChatGPT can be as good as other popular commercial translation products like Google Translate for high-level resource languages but falling behind for low-resource languages (Jiao et al., 2023). Moreover, a stronger standpoint was given in another journal article which provided tests and evidence on the performance capability of language translation for ChatGPT, comparing high-resource languages like English and Indonesian languages with Buginese language, proving its underwhelming performance with low-resource languages (Bang et al., 2023). Amidst this, very important mentions of natural language processing and machine translation have been mentioned from several papers. An old journal paper from 2001, states the initial state of natural language processing, talking in depths about its history, sub-fields and levels (Liddy, 2001). Similarly, another paper published in 2011 also discusses the historical evolution of natural language processing, highlighting its diverse efforts, approaches, problems and solutions (Nadkarni et al., 2011). When it comes to natural language processing, machine translation, a prominent part is needed to be mentioned in details. Hence, a research study has been included stating machine translation's history, carrying the relevant initial inventions under this field to notable inventions like Google Translate (Jolley & Maimone, 2022). To support more details to the relevance of machine translation, another mentionable journal covered the various new trends machine translation is evolving with, thanks to the use of large language models (Ouyang et al., 2022). Another paper discusses the challenges of neural machine translation, under which still existed the

problem of low quality performance with low-resource languages for machine translation systems in 2017 (Koehn & Knowles, 2017).

A study has been conducted on ChatGPT's translation performance of Arabic-to-English. In this paper, it was evaluated that while ChatGPT is capable of providing somewhat accurate translations focusing on grammar and general meaning, it is not able to capture the historical and cultural context via testing on translating the Qu'ran from Arabic to English. It also emphasizes on ChatGPT's limitations (Khoshafah, 2023).

Many papers discuss their own workings and machine translation systems. One of them includes “shu-torjoma”, a popular phrase-based statistical machine translation system which translates both from Bangla to English and English to Bangla. It is a revolutionary model that back then used supposedly the largest datasets (Mumin;, Seddiqui;, Iqbal;, & Islam;, 2019). Another paper explains that methods used to evaluate low resource languages is difficult to find, and FLORES evaluation for Nepali-English and Sinhala-English was introduced. The authors agree that machine translation systems have poor performance on especially low-resource language pairs (Guzman; et al., 2019). Another popular work is the “IndicBART”, a multilingual, sequence-to-sequence pre-trained model working on 11 Indic languages and English. The model is not only evaluated based on 2 natural language generation tasks like neural machine translation (NMT) and extreme summarization, but also compared to large pre-trained models like “mbart50” in the paper (Dabre; et al., 2022). Inspired from this and other works, “BanglaBERT” was introduced later. It is a Bengali pre-trained model for Natural Language Understanding (NLU). This paper covers how the model has been made, the NLU tasks based on which it has been evaluated and some comparisons to prove its quality performance (Bhattacharjee; et al., 2022). Later, the same authors came up with “BanglaT5” and “BanglaNLG” benchmark made to evaluate this model accurately. In comparison, the paper also shows it's been able to outperform other strong multilingual models (Bhattacharjee et al., 2022).

Lastly, an important study is mentioned which proves biasness of ChatGPT via making it produce Limericks for liberal and conservative politicians (McGee, 2023).

3. Findings & Implementation

I tested ChatGPT on my own to analyze its translation performance. On one occasion I ask its generated response to be translated to Bengali, on another occasion I ask a question directly in Bengali to the chatbot. I also wish to check its translation for high-resource languages like French in comparison to a low-resource language like Bengali.

<i>User:</i> Can you write a summary about AI?
<i>ChatGPT:</i> Summary: Artificial Intelligence (AI) is a transformative technology that has revolutionized various fields. It refers to the development of intelligent machines that can perform tasks that typically require human intelligence. AI systems have the ability to process vast amounts of data, learn from patterns, and make decisions or predictions. This technology finds applications in numerous domains such as data analysis, natural language processing, computer vision, robotics, and more. AI offers several benefits, including increased efficiency, improved accuracy, personalized experiences, and automation of tedious tasks. However, challenges and ethical considerations exist, such as privacy concerns, bias and fairness issues, and potential job displacement. Overall, AI has the potential to significantly impact and shape the future, making it a fascinating and rapidly evolving field.

Figure 2.0: Simple question-answering with ChatGPT (English Response)

As mentioned earlier, ChatGPT provides very long summaries, but it is to be noted it took a few seconds to provide the summary for AI in English. On the other hand, when I ask the assistant to translate the English response to Bangla, it takes relatively a long time. Moreover, the text is not fully translated remaining incomplete, with several sentence errors in between. Although the main idea of the summary can be understood somewhat, there is lack of accuracy and consistency in the summary for Bangla.

<i>User:</i> Can you please translate the text generated above to Bangla?
<i>ChatGPT:</i> সংক্ষিপ্তসার: কৃত্রিম বুদ্ধিমত্তা (আর্টিফিশিয়াল ইন্টেলিজেন্স) হলো একটি বিপ্লবী প্রযুক্তি যা বিভিন্ন ক্ষেত্রগুলিতে বিপ্লব আনে। এটি মানুষ বুদ্ধিমত্তার প্রয়োজনীয় কাজগুলি করতে সক্ষম করতে পারে বুদ্ধিমত্তামূলক যন্ত্রগুলির উন্নয়নকে বোঝায়। এই প্রযুক্তিতে সিদ্ধান্ত গ্রহণ করতে পারে, ডেটা প্রক্রিয়া করতে পারে এবং প্যাটার্ন থেকে শেখা নিতে পারে। এটির অ্যাপ্লিকেশনগুলি তথ্য বিশ্লেষণ, ন্যাচারাল ল্যাঙ্গুয়েজ প্রসেসিং, কম্পিউটার ভিশন, রোবোটিক্স এবং অন্যান্য বিভাগে পাওয়া যায়। কৃত্রিম বুদ্ধিমত্তা একাধিক সুবিধা দেয়, যেমন বৃদ্ধিমান কর্মক্ষমতা, নির্ভুলতা বৃদ্ধি, ব্যক্তিগতকৃত অভিজ্ঞতা এবং ঘনীভূত কার্যগুলির স্বয়ংক্রিয়তা। তব

Figure 2.1: English Response translated by ChatGPT to Bengali, low-resource language (English response from figure 2.0)

It is vividly different when I ask the chatbot to translate its earlier English response (summary about AI) to French. It not only takes much less time to deliver the French output, it also finishes the entire summary as given in English. Therefore, it is evident that ChatGPT provides higher quality performance for high-resource languages than for low-resource languages. Regardless, it may not provide in-depth translations. According to a research study done on translating the Qu’ran’s verses (Arabic-to-English Translation), it was found that ChatGPT is capable of providing accurate and clear answers with correct grammar. However, it does not defeat the historical context and in-depth human translation (Khoshafah, 2023).

<i>User:</i> Can you please translate the text from English to French
<i>ChatGPT:</i> Résumé: L'intelligence artificielle (IA) est une technologie transformative qui a révolutionné divers domaines. Elle fait référence au développement de machines intelligentes capables d'effectuer des tâches qui nécessitent généralement l'intelligence humaine. Les systèmes d'IA ont la capacité de traiter de vastes quantités de données, d'apprendre à partir de modèles et de prendre des décisions ou des prédictions. Cette technologie trouve des applications dans de nombreux domaines tels que l'analyse de données, le traitement du langage naturel, la vision par ordinateur, la robotique, et bien d'autres. L'IA offre plusieurs avantages, notamment une efficacité accrue, une précision améliorée, des expériences personnalisées et l'automatisation des tâches fastidieuses. Cependant, des défis et des considérations éthiques existent, tels que les préoccupations liées à la confidentialité, les problèmes de biais et d'équité, ainsi que le risque de suppression d'emplois. Dans l'ensemble, l'IA a le potentiel d'avoir un impact significatif et de façonner l'avenir, en faisant un domaine fascinant et en évolution rapide.

Figure 2.2: English Response translated to French by ChatGPT, high resource language (English response from figure 2.0)

Another notable example, is when I asked ChatGPT a question directly in Bangla. Shockingly, it answers quite accurately with a concise answer. However, I do believe this could be explained in a simpler form.

<i>User:</i> মানুষের শরীরের দীর্ঘতম হাড় কি?
<i>ChatGPT:</i> মানুষের শরীরের দীর্ঘতম হাড় হলো ফেমুর হাড়। ফেমুর হাড় হলো মানুষের পা মাংসপেশিতে অবস্থিত হাড়ের মধ্যবর্তী একটি হাড়। ফেমুর হাড়ের দীর্ঘতা প্রায় 26-27 ইঞ্চি (66-68 সেন্টিমিটার) হতে পারে। এটি শরীরের অন্যতম দীর্ঘ হাড় এবং শরীরের সামান্য উপায়ে মাপা যেতে পারে। ফেমুর হাড় মানুষের পা এবং কল এবং টানের সামরিক সংরক্ষণ এবং চলাচলের জন্য গুরুত্বপূর্ণ।

Figure 2.3: Direct Bangla Question asked by User

Thirdly, I tried to ask the same question in English, to translate it to Bangla later. This provided me an interesting insight, as can be seen a shorter response is formed when doing this as given in English.

<i>User:</i> What is the longest bone in the human body?
<i>ChatGPT:</i> The femur, or thighbone, is the longest bone in the human body. It extends from the hip joint to the knee joint and is responsible for supporting the body's weight and facilitating movement. The average length of the femur in adults is about 26-27 inches (66-68 centimeters).
<i>User:</i> Can you please translate this text to Bangla?
<i>ChatGPT:</i> মানুষের শরীরের দীর্ঘতম হাড় হলো ফেমুর বা পা-হাড়। এটি হিপ জয়েন্ট থেকে নুড়ি জয়েন্ট পর্যন্ত বিস্তৃত হয় এবং শরীরের ওজন সমর্থন এবং চলাচলের জন্য দায়ী। পূর্ণাঙ্গ ফেমুরের গড় দৈর্ঘ্য প্রায় 26-27 ইঞ্চি (66-68 সেন্টিমিটার)।

Figure 2.4: Repetition of question (English response from figure 2.1) in English, then translated to Bangla

Opposite to the last test, I tried to translate the Bengali response to English. It can clearly be perceived that ChatGPT provides much shorter responses in Bengali, with very precise and summarized answers. Once asked to translate the Bengali response to English, it successfully provides a high quality translation in English as expected.

<i>User:</i> শেক্সপিয়ার কে?
<i>ChatGPT:</i> উইলিয়াম শেক্সপিয়ার (William Shakespeare) একজন প্রমুখ ইংরেজি লেখক, নাটককার এবং কবি ছিলেন। শেক্সপিয়ার বিশ্বব্যাপী প্রশংসিত হিসাবে পরিচিত, এবং তাঁর কাব্যগ্রন্থসমূহ একজন নাটক লেখক হিসাবে বিশ্বব্যাপী মর্যাদা ধারণ করে। তাঁর জন্ম ১৫৪৬ সালে লন্ডনের স্ট্র্যাটফোর্ড-আপন-ইভন (Stratford-upon-Avon) নামক একটি ছোট শহরে হয়। শেক্সপিয়ারের রচিত নাটকগুলি, কবিতা এবং উপন্যাস মানব জীবনের গভীরতম মনোভাব এবং মানব সম্পর্কের বিভিন্ন মাত্রার চিত্রণ করে। তাঁর কাজের মধ্যে সবচেয়ে প্রশিষ্ট হলো তাঁর ৩৭টি নাটক, যেগুলোর মধ্যে থাকছেন 'হ্যামলেট', 'রোমিও ও জুলিয়েট', 'ম্যাকবেথ', 'অথেলো', 'ওথেলো', 'জুলিয়াস সিজার', 'মিডসামার নাইটের স্বপ্ন' ইত্যাদি। শেক্সপিয়ারের কাব্যগুলি একটি গভীর মানব
<i>User:</i> Can you translate the text from above to English?
<i>ChatGPT:</i> William Shakespeare was a prominent English writer, playwright, and poet. Shakespeare is widely recognized and his literary works are considered a global benchmark, with his plays establishing him as a world-renowned playwright. He was born in 1546 in Stratford-upon-Avon, a small town in London. Shakespeare's composed plays, poems, and novels depict the deepest emotions of human life and various aspects of human relationships. Among his notable works are 37 plays, including "Hamlet," "Romeo and Juliet," "Macbeth," "Othello," "Julius Caesar," "A Midsummer Night's Dream," and others. Shakespeare's poetry embodies a profound understanding of human nature and the complexities of human existence.

Figure 2.5: Direct Bengali Response translated to English

On contrast, once the AI assistant is directly asked the same question in English, the response becomes very lengthy and informative. Therefore, we understand its lack of translation and question-answering in other low-resource languages quite well now

<i>User:</i> Who is William Shakespeare?
<i>ChatGPT:</i> William Shakespeare was an English playwright, poet, and actor who lived during the late 16th and early 17th centuries. He is widely regarded as one of the greatest writers in the English language and is often called the "Bard of Avon." Shakespeare's works encompass a wide range of genres, including tragedy, comedy, and historical plays. Some of his most famous plays include "Romeo and Juliet," "Hamlet," "Macbeth," "Othello," and "A Midsummer Night's Dream." Shakespeare's plays are known for their complex characters, intricate plots, and rich language. He is credited with coining numerous phrases and expressions that have become part of the English language. Despite his enduring fame, there is limited information about Shakespeare's personal life. He was born in Stratford-upon-Avon, England, in 1564, and little is known about his education and early years. He married Anne Hathaway in 1582, and they had three children.....

Figure 2.6: English question and response (the response is much longer and detailed)

It is important to note that I built my own natural language processing system from previous related works and relevant projects. As we are using a pre-made natural language processing system (ChatGPT), our focus is on machine translation systems made over the decades.

The first ever effort to develop an English to Bangla MT system was in 1991, which was to be part of Anglabharti, part of a large English to Indian Languages system. There came many other upgrades on the way. Later in 2005, 'Banganubad', an English-to-Bangla translator was made using complex natural language engineering methods with other technical components. Much later in 2014, 'Sata-Anuvadak' was introduced, possibly one of the first systems available for this purpose. Under it, was a collection of 110 statistical machine translation systems which was able to handle multiple translations of Indian language simultaneously. Of course, came along other worldly known commercial machine translation products like Google Translate which even upgraded to a neural-based machine translation system in 2016. All this information was shared by the researchers who came up with a phrase based statistical machine translation system, shu-torjoma in 2019. It has been said to be trained on the largest dataset back then of more than 3 million token and 197,338 sentence pairs. Its purpose was to translate technical reports and financial documents. However, it was found out that translating from a richer language, English to a low-resource language like Bangla was more difficult than translating from Bangla to English (Mumin; et al., 2019). According to another research conducted, it is proven that machine

translation systems still have a hard time in learning to translate low-resource language (*Guzman; et al., 2019*).

A multilingual, pre-trained sequence-to-sequence model was developed in 2022 to be integrated with natural language generation applications for Indian languages, ‘IndicBART’. This was able to support 11 Indian languages including Bengali. This model was based on other transformer models. It was trained on about 450 million sentences and 9 billion tokens. Interestingly, the model was able to tenable translation for languages like Sinhala and Nepali even when it was not pre-trained to do so. Further, it was said that fine-tuning ‘IndicBART’ on a certain set of languages allowed translation for another overlooked set of languages (*Dabre; et al., 2022*).

In the same year, ‘BanglaBERT’ was created by pre-training on a collection of approximately 27.5 GB of Bangla pre-training data from 110 popular Bengali sites. Afterwards, a bilingual pre-trained model for zero-shot transfer learning was created called ‘BanglishBERT’. Both these pre-trained models were then tested and evaluated on 4 natural language understanding tasks such as text classification and span prediction, which were built to be part of the first ever Bangla Language Understanding Benchmark (BLUB). It was said to outperform strong models like ‘IndicBART’, previously discussed of (*Bhattacharjee; et al., 2022*). Soon after, the same authors created a sequence-to-sequence Transformer language model for Bengali called ‘BanglaT5’ which was evaluated upon the benchmark of ‘BanglaNLG’ (benchmark to evaluating natural language generation) (*Bhattacharjee et al., 2022*). This model is deeply elaborated in the following section as I have successfully incorporated this model within the project.

3.1 Language Models

The first task in hand was to search for the right API to collect dataset of Bangla to English and English to Bangla. The ideal source was the Google Translate API since it delivers relatively accurate translations. However, it is not free and was dropped as an option.

From the popular AI community, Hugging Face I was able to discover ‘BanglaT5’ model, which consisted datasets for 2 purposes, Bengali-to-English text and English-to-Bengali text.

The BanglaT5 model used the Bangla 2B+, a 2.75GB dataset made from a combination of web sources consisting 5.25 million documents for pre-training data. For pre-processing the texts,

previously used normalization pipeline was integrated. This involves a step by step process like normalizing punctuations and characters consisting of multiple unicode representations to reduce data sparsity, or preventing data leakage by removing all evaluation pairs from the training data etc. Most importantly, the model was experimented and compared to other powerful existing multilingual language models mentioned earlier, like ‘IndicBART’ and ‘mBART50’. The comparison was done in context of various ‘BanglaNLG’ tasks like machine translation, text summarization, question answering, multi-turn dialogue, news headline generation and cross-lingual summarization.

Model	Parameters	MT	TS	QA	MTD	NHG	XLS
mT5 (base)	582M	30.1/ 17.2	10.3	59.0/65.3	17.5	9.6	2.7/0.7
XLNet-ProphetNet	616M	27.5/15.4	7.8	53.0/57.3	20.0	9.5	6.2 /2.7
mBART-50	611M	29.7/15.5	10.4	53.4/58.9	18.5	11.2	5.4/ 3.7
IndicBART (unified)	244M	28.1/16.6	8.9	59.6/65.6	14.8	7.9	6.3 /2.5
IndicBART (separate)	244M	27.5/15.7	9.2	55.3/61.2	14.1	9.1	5.3/2.4
BanglaT5	247M	31.3 / 17.4	13.7	68.5 / 74.8	19.0	13.8	6.4 / 4.0

Figure 3.0: Performance comparison of the pre-trained models

The model was tested on Jupyter Notebook by hosting as Inference API.

```
#bn to en
import requests

API_URL = "https://api-inference.huggingface.co/models/csebuetnlp/banglat5_nmt_bn_en"
headers = {"Authorization": "Bearer hf_RbRXW0yHtHBWeDBuOZgZFPcsZmgZpnHirmX"}

def query(payload):
    response = requests.post(API_URL, headers=headers, json=payload)
    return response.json()

output = query({
    "inputs": "পৃথিবীর সবচেয়ে বড় প্রাণী কি?",
})

output
[{'translation_text': 'What is the largest creature on earth?'}]
```

Figure 3.1: Bengali text-to-English text

First of all, because we are calling out APIs, we will need to send HTTP requests in Python by importing requests. Secondly, I will state the API URL being used. The variable headers consist of my personalized API token made in Hugging Face, a form of password for authentication and security. Then, under the query(payload) function I contain important information about the

request, by setting a variable as response that stores the data according to the requirements, the variables, API_URL and headers. To make sure the response is stored and shown as json format, a mix of dictionary and lists I say return response.json(). Lastly, I call out the function under a new variable. The variable consists of an easy-to-understand dictionary (json), the key being the inputs (self-explanatory purpose) and the value being the text to be translated.

Similarly, the English-to-Bengali text part was tested out with almost the same required code. The difference is the API_URL is distinct as according to the requirement of English to Bengali from the ‘BanglaT5’ checkpoint at Hugging Face.

```
#en to bn
import requests

API_URL = "https://api-inference.huggingface.co/models/csebuetnlp/banglat5_nmt_en_bn"
headers = {"Authorization": "Bearer hf_RbRXW0yTHBWeDBuOZgZFPCsZmgZpnHirmX"}

def query(payload):
    response = requests.post(API_URL, headers=headers, json=payload)
    return response.json()

output = query({
    "inputs": "What is the biggest animal in the world?",
})
```

output

```
[{'translation_text': 'পৃথিবীর সবচেয়ে বড় প্রাণী কি?'}]
```

Figure 3.2: English text-to-Bengali text

3.2 ChatGPT API

Lastly, I called out the entire application programming interface of ChatGPT, the complete functional chatbot into my Jupyter Notebook to test it out with the language models mentioned above.

Firstly, I installed the required python bindings by running the following command, which I also need to import before using it as shown below:

```
pip install openai
```

Figure 4.0: Installation of the required Open AI library

```
import openai
```

Figure 4.1: OpenAI library needs to be imported every time before usage

Secondly, I made sure to open an individual account on OpenAI with my email address and necessary login information, which automatically provided me API keys for authentication. It is to be noted that an API key is a secret that should not be disclosed, and is only used to make requests for authentication.

```
Authorization: Bearer OPENAI_API_KEY
```

Figure 4.2: The Authorization HTTP header sequence

I found my API key from set keys provided by the OpenAi account, that has been inserted in the code for use of this purpose.

```
openai.api_key = 'sk-ZdG1JcNMHmMgo8JEPcT3B1bkFJwz9dZm83hNR7MYOPqrAP'
```

Figure 4.3: My retrieved API key from my OpenAI account

The chatbot, ChatGPT must be given context now and be structured to what it is supposed to do. 'You are a kind helpful assistant', context is given to the ChatGPT API so it knows how to behave, and to avoid slangs and inappropriate language. The advantage of giving context earlier is that ChatGPT holds memory while answering questions. The messages variable is equal to a json under which content is fixed. It is to be noted that there are 2 distinct roles, system role (ChatGPT) and user role (the user who is chatting).

```
messages_list = [  
    {"role": "system", "content": "You are a kind helpful assistant."},  
]
```

Figure 4.4: Setting up context for ChatGPT

After setting up context, an infinite loop is created to have chat mechanism. User message is retrieved using 'input', where we collect user input and print it out as given in the function. For this, while loop is suitable.

So, message is user message is retrieved via input function. If the message is present then the message is appended with existing messages, which is outside the while loop to prevent overwriting, stating the role and content. Role, user and content is message received from user.

To invoke OpenAI chat completion, which is the chat object with model 3.5 turbo (what is actually powers the ChatGPT interface). 'Messages=messages' implies to the message being sent.

Then the response is collected from 'chat.choices[0]' (0 meaning the first object). On the next line, message is received, with message content. The last line of the function includes once again an appended messages list within which role the role is assigned to as assistant to ChatGPT and the content as reply to provide context.

```
while True:
    message = input("User : ")
    if message:
        messages_list.append(
            {"role": "user", "content": message},
        )
        chat = openai.ChatCompletion.create(
            model="gpt-3.5-turbo", messages=messages_list
        )

        reply = chat.choices[0].message.content
        print(f"ChatGPT: {reply}")
        messages_list.append({"role": "assistant", "content": reply})
```

Figure 4.5: Fully responsive ChatGPT

3.3 Integration of Language Models with ChatGPT

Now, both the ChatGPT and the language models can be integrated and functional together by putting them all under one function in a step by step sequence. In brief, 5 steps are done here:

1. Take Bengali input as text
2. Translate Bengali to English
3. Send translated text to ChatGPT
4. Translate the English ChatGPT response into Bengali
5. Output the Bengali text

```

import requests
messages_list = [
    {"role": "system", "content": "You are a kind helpful assistant."},
]
def full_conversation():
    # step 1
    user_message = input("Enter text please: ")
    if user_message == "exit":
        return user_message
    # step 2
    bn_en_API_URL = "https://api-inference.huggingface.co/models/csebuetnlp/banglat5_nmt_bn_en"
    headers = {"Authorization": "Bearer hf_RbRXW0ytHBWeDBu0ZgZFPCsZmgZpnHirmX"}
    response = requests.post(bn_en_API_URL, headers=headers, json={"inputs": user_message})
    english_user_msg = response.json()[0]["translation_text"]
    print(english_user_msg)

    # step 3
    messages_list.append(
        {"role": "user", "content": english_user_msg},
    )
    chat = openai.ChatCompletion.create(
        model="gpt-3.5-turbo", messages=messages_list
    )

    # step 4
    reply = chat.choices[0].message.content
    print(f"ChatGPT: {reply}")
    messages_list.append({"role": "assistant", "content": reply})
    en_bn_API_URL = "https://api-inference.huggingface.co/models/csebuetnlp/banglat5_nmt_en_bn"
    headers = {"Authorization": "Bearer hf_RbRXW0ytHBWeDBu0ZgZFPCsZmgZpnHirmX"}
    response = requests.post(en_bn_API_URL, headers=headers, json={"inputs": reply})
    bn_reply = response.json()[0]["translation_text"]
    # step 5
    print(bn_reply)

```

Figure 5.0: Fully functioning ChatGPT merged with the language models

While true function, allows the function, ‘full_conversation()’ to loop forever but under it falls a ‘if’ condition which says that if ‘full_conversation()’ is equal to ‘exit’ then break indicating to terminate this loop statement. Therefore, this helps to put an end to the conversation without being an infinite loop.

```

while True:
    if full_conversation() == "exit":
        break

```

Figure 5.1: Controlling merged function with loop statement

The output is shown as below to have successfully worked, with the Bengali text as input and translated Bengali response from ChatGPT.

```
Enter text please: বিশ্বের সবচেয়ে লম্বা প্রাণী কি?  
What is the world's tallest animal?  
ChatGPT: The world's tallest animal is the giraffe, with an average height of 18 feet (5.5 meters).  
বিশ্বের সবচেয়ে লম্বা প্রাণী হল জিরাফ, যার গড় উচ্চতা ১৮ ফুট (৫.৫ মিটার)।  
Enter text please: exit
```

Figure 5.2: The output of the 'full_conversation()'

3.4 Speech Recognition Models

Next, I also had to find out speech recognition models to work along the language models for a fully functioning ChatGPT meeting the designated purpose. Similar to discovering the language models, both Bengali speech-to-text model and English text-to-Bengali speech model are needed.

For the speech-to-text model, I stumbled upon an open source offline Bengali speech-to-text package for Bengali that only involved a few simple steps for setup. I found it on GitHub, an internet hosting service that is often used by programmers and software developers. A user named shhossain shared the 'BanglaSpeech2Text' model with the setup commands that have been followed and tested in Jupyter Notebook as shown below. It is to be noted, however, that this model was taken from Silero Models, which has pre-trained STT/ TTS models and benchmarks.

```
pip install banglaspeech2text
```

Figure 6.0: Installation of the required Bangla speech-to-text model

```
pip install SpeechRecognition
```

Figure 6.1: Installation of speech recognition for processing human language

```
pip install pyaudio
```

Figure 6.2: Installation of pyaudio to allow recording and playing the audio

```

import speech_recognition as sr
from banglaspeech2text import Model, available_models

# Load a model
models = available_models()
model = models[0] # select a model
model = Model(model) #
model.load()

r = sr.Recognizer()
with sr.Microphone() as source:
    print("Say something!")
    audio = r.listen(source)
    output = model.recognize(audio)

print(output) # output will be a dict containing text
print(output['text'])

```

Figure 6.3: Testing out the banglaspeech2text model

```

Say something!
{'text': 'কিছু একটা।'}
কিছু একটা।
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

```

Figure 6.4: Output from the banglaspeech2text model

For the Bengali text-to-speech model, another model from GitHub has been used, from the group of Silero Models itself. It is the ‘BanglaTTS’ model. The set-up commands to run are provided by the same user, shhossain once again, as following:

```

pip install BanglaTTS

```

Figure 6.5: Installation of the required Bangla text-to-speech model

```

import scipy.io.wavfile

```

Figure 6.6: Importing Scipy to open the audio wav file

```
from banglatts import BanglaTTS

tts = BanglaTTS(save_location="test")
path = tts("পৃথিবীর সবচেয়ে লম্বা প্রাণী কোনটি?", voice='female', filename='1.wav') # voice can be male or female

Downloading: "https://github.com/snakers4/silero-models/zipball/master" to test\master.zip
```

Figure 6.7: Testing out the 'BanglaTTS' model

```
path

'1.wav'
```

Figure 6.8: The output from the 'BanglaTTS' model saved in the directory

3.5 Web Application

For the web application which processes all the functionalities from processing the Bengali speech to text, to converting to English for ChatGPT, to converting the English response to Bangla text and Bengali speech. All of it needs to be shown on the application for the user.

Initially, the prototype of the application was fixed according to research from inspiring designs on Dribbble and Behance (a platform for designers to promote their work). Below are the references used to make the simple layout of the application.

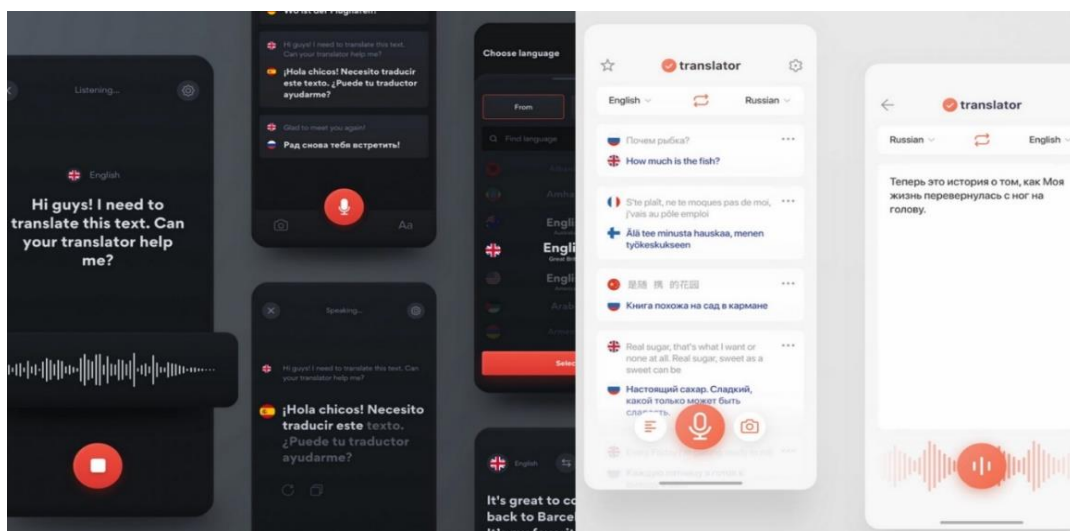


Figure 7.0: Inspiration from similar Dribbble and Behance designs

Next, once confirming how the layout shall be and its few essential features, I dived into studying HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) which are used for designing web pages easily. Instead of using CSS grid, I used CSS Flexible box layout, a newly popular web layout model. This layout model truly helps to align all items and elements, and distribute space evenly by placing them in containers, which allows easy modification and cleaner code. For the language of implementing the design, I chose to incorporate simple Javascript via Vite, a frontend tooling for better performance and reliable dev server.

After setting up the layout alone, I used Flask for hosting the server and incorporate the functions. I used Flash because it provides a convenient method to host a server for free, plus it has coherent and concise documentation for reference.

The first task then was to make sure my browser allowed permission for access of microphone. Secondly, instead of calling out my language models from VS Code, I placed all the models under the same file for quick request processing and prevention of timeout errors. This was also done for the speech recognition models.

4 separate script files were made, one main javascript file, an index.html file, a style.css file and 2 python scripts for the integration of the combination of ChatGPT, the language models and speech recognition models, and another file solely for the making of the application with Flask.

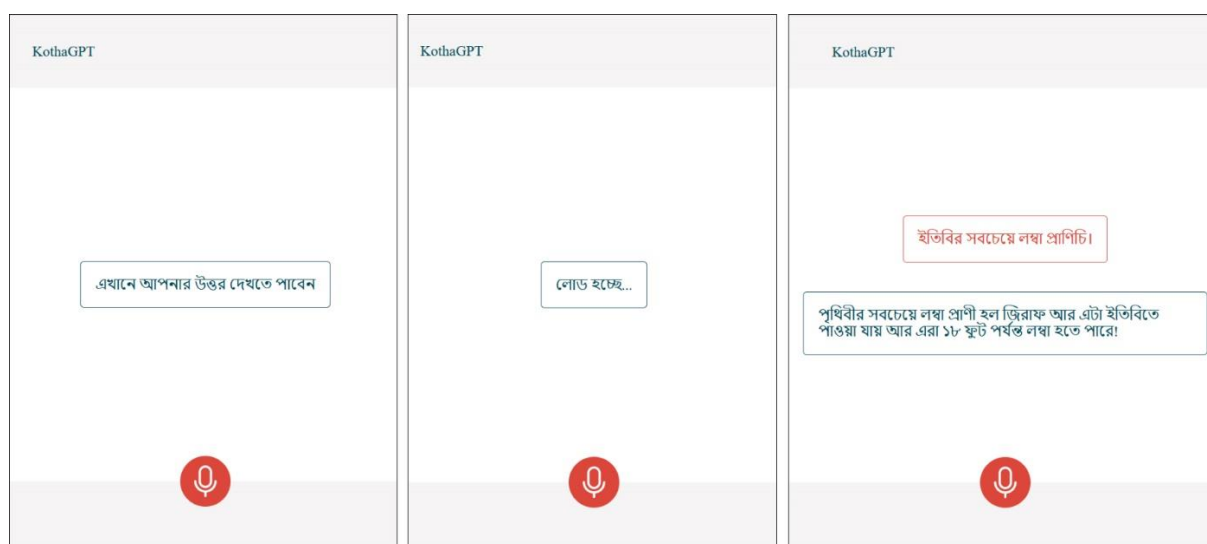


Figure 7.1: The final UI for the web application

Main.js

For execution on the web pages, the javascript code is written in a .js file. If seen below on index.html, the <script> tag consists of the link to the .js file.

All the code in this file is the integration of the mic with the models, the audio file data and the screen functionalities used by the user. One of the important properties here is the classList, that allows to add, remove, replace or toggle to check whether a specific class is present. For example, the property, classList on a button helps to either remove or add the “stop” for recording button.

How the error message appears when a network timeout occurs is also incorporated in the code, to make everything more presentable.

```
static > JS main.js > [E] handleSuccess > [E] mediaRecorder.addEventListener("stop") callback
1  const SAMPLE_RATE = 48000;
2  const defaultText = "এখানে আপনার উত্তর দেখতে পাবেন";
3
4  const query = document.getElementById("query");
5  const text_div = document.getElementById("speech_result");
6
7  // Recording icons
8  const micButton = document.getElementById("microphone");
9  const stopButton = document.getElementById("stop");
10
11  let downloadSrc = null;
12
13  const handleSuccess = function (stream) {
14    const options = { mimeType: "audio/webm" };
15    const recordedChunks = [];
16    const mediaRecorder = new MediaRecorder(stream, options);
17
18    mediaRecorder.addEventListener("dataavailable", function (e) {
19      if (e.data.size > 0) recordedChunks.push(e.data);
20    });
21
22    mediaRecorder.addEventListener("stop", async () => {
23      micButton.classList.remove("hidden");
24      stopButton.classList.add("hidden");
25
26      downloadSrc = new Blob(recordedChunks);
27
28      let formData = new FormData();
29      formData.append("voice_msg", downloadSrc);
30
31      text_div.innerText = "লোড হচ্ছে...";
```

Figure 8.0: Main.js file code (1)

```

33     try {
34         let response = await fetch("/upload", { method: "POST", body: formData });
35
36         if (!response.ok) {
37             throw new Error(
38                 `Network error ${response.status}: ${response.statusText}`
39             );
40         }
41
42         let data = await response.json();
43
44         // Display user query
45         query.classList.remove("hidden");
46         query.innerHTML = data["query"];
47
48         // Display translated text on screen
49         const translatedText = data["reply"];
50         text_div.innerHTML = translatedText;
51
52         playAudio(data["audio"]);
53     } catch (error) {
54         text_div.classList.add("error");
55         text_div.innerHTML = error.message;
56
57         console.error(error.message);
58     }
59 };
60
61 stopButton.addEventListener("click", handleStopPress);
62
63 mediaRecorder.start();
64
65 function handleStopPress() {
66     mediaRecorder.stop();
67 }
68 };

```

Figure 8.1: Main.js file code (2)

```

70 // Start recording
71 micButton.addEventListener("click", handleMicPress);
72
73 function handleMicPress() {
74     reset();
75
76     navigator.mediaDevices
77         .getUserMedia({ audio: true, video: false })
78         .then(handleSuccess);
79 }
80
81 function reset() {
82     text_div.classList.remove("error");
83     query.classList.add("hidden");
84     text_div.innerText = defaultText;
85     micButton.classList.add("hidden");
86     stopButton.classList.remove("hidden");
87 }
88
89 /**
90  * Convert list of numpy audio array to audio buffer and play the audio
91  */
92 function playAudio(ndArrayList) {
93     const context = new AudioContext();
94     const buffer = context.createBuffer(1, ndArrayList.length, SAMPLE_RATE);
95
96     // Create a single audio array buffer from the audio array list
97     for (var channel = 0; channel < buffer.numberOfChannels; channel++) {
98         var nowBuffering = buffer.getChannelData(channel);
99         for (var i = 0; i < buffer.length; i++) {
100             nowBuffering[i] = ndArrayList[i];
101         }
102     }

```

Figure 8.2: Main.js file code (3)

```
104   const source = context.createBufferSource();
105   source.buffer = buffer;
106   source.connect(context.destination);
107   source.start();
108 }
```

Figure 8.3: Main.js file code (4)

Index.html

My index.html file carries the actual content that can be seen on the screen, the basic elements of the webpage are described. Below, is all the code used in the index.html file to make the structure of my one-page application.

A html file always starts with `<!DOCTYPE html>` which basically tells the browser that the HTML codes should be rendered as HTML5. Then, it continues with the needed elements of an HTML document, html tag which is the root of the document and specifies the language being used. Under this tag, lie the `<head>` tag, `<body>` tag and other tags and elements. Below are all the tags and

`<head>` tag

Inside this tag, `<meta....>` tag consists of the standards used for the application, such as the standard encoding system and secondly, the default size for the application meaning the user's visible area on a web page.

The `<link>` tags are included to add icons, and also make relationships with other documents like as shown with style.css file.

`<body>` tag

Moving onto the `<body>` tag, which consists the content that can be seen by the users, such as headings to paragraphs or media etc.

Under the `<body>` tag falls `<main>` tag to focus on the most significant part of the web page. As we have used flexbox for CSS, we have used div classes to use as containers for style.css file. The classes consist of the svg icons that appear on the screen, and their designated linked requirements and paths.

```

templates > index.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <link rel="icon" type="image/svg+xml" href="/vite.svg" />
6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7     <title>Vite App</title>
8     <link
9       rel="stylesheet"
10      href="{{ url_for('static', filename='styles/normalize.css') }}"
11    />
12    <link
13      rel="stylesheet"
14      href="{{ url_for('static', filename='styles/style.css') }}"
15    />
16  </head>
17
18  <body>
19    <main>
20      <div class="header">
21        <span class="logo">BanglaChatGPT</span>
22      </div>
23      <div class="content">
24        <div id="speech_result">এখানে আপনার উত্তর দেখতে পাবেন</div>
25      </div>
26      <div class="recorder">
27        <button id="microphone">
28          <svg
29            xmlns="http://www.w3.org/2000/svg"
30            width="40"
31            height="40"
32            fill="red"
33            viewBox="0 0 256 256"
34          >
35            <path
36              d="M128,176a48.05,48.05,0,0,0,48-48V64a48.05,48.05,0,0,0,0-96,0v64a48.05,48.05,0,0,0,128,176Zm96,64a32,32,0,0,0,1,64,0v64a32,32,0,0,0,1-64,0Zm40,143.6V232a8,
37            </path>

```

Figure 9.0: Index.html file code (1)

```

38    </svg>
39  </button>
40
41  <button id="stop" class="hidden">
42    <svg
43      xmlns="http://www.w3.org/2000/svg"
44      width="40"
45      height="40"
46      fill="red"
47      viewBox="0 0 256 256"
48    >
49      <path
50        d="M200.73,40H55.27A15.29,15.29,0,0,0,0,40,55.27V200.73A15.29,15.29,0,0,0,55.27,216H200.73A15.29,15.29,0,0,0,216,200.73V55.27A15.29,15.29,0,0,0,200.73,40Z"
51      </path>
52    </svg>
53  </button>
54 </div>
55 <div class="nav"></div>
56 </main>
57 <script src="{{ url_for('static', filename='main.js') }}"></script>
58 </body>
59 </html>

```

Figure 9.1: Index.html file code (2)

Style.css

To format the layout of the web page, and define the structure flexbox came in handy to play around with the alignment and space distribution more freely. This also makes the code displayed more pleasing and understanding to the readers.

As shown, the size adjustments were done on the tags that can actually be visible to the users, meaning the body has been given the default view of a mobile user.

Similarly, the main tag involves the default mobile screen size.

Now, onto the flexbox usage throughout the document. ‘Display : flex’ is in every container as shown because it defines a flex container. Flex-direction is self-explanatory; it defines the direction the items of the flex container are placed. As is always viewed on mobile, it appears column-wise as given.

Another interesting addition is the flex-flow, which is the combination of flex-direction and flex-wrap which accommodate in describing both the flex container’s main and cross axes. In default, the value is always set to be ‘row nowrap’, however, in our case, the header container needed to be row wrapped.

Most importantly, justify-content is required to explain the alignment along the main axis. This really helps in dealing with extra space and management of line overflowing each other. For the header and nav container, ‘space-between’ option is preferred as items need to be evenly distributed in the row from the items on the start line to the end line. On the other hand, the content container (which consists of the recorder button) is placed on the ‘center’, meaning the items are centered along the line.

In contrast, ‘align-items’ is another kind to define the way the flex items are shown across the cross axis on the current line. All items are set to be in ‘center’ in the cross-axis.

The flex items (also referred to as children) can also be individually customized as can be seen for the recorder. The recorder button is sized and styled according to preference.

```

1  body {
2      width: 100%;
3      height: 100vh;
4  }
5
6  :root {
7      --gray: #F6F5F5;
8      --green: #004756;
9      --white: #ffffff;
10     --orange: #dc463b;
11 }
12
13
14 main {
15     max-width: 480px;
16     margin-inline: auto;
17     display: flex;
18     flex-direction: column;
19     height: 100%;
20     background-color: var(--white);
21 }
22
23 .header {
24     display: flex;
25     flex-flow: row wrap;
26     justify-content: space-between;
27     height: 80px;
28     background-color: var(--gray);
29     align-items: center;
30     padding-left: 10%;
31     padding-right: 10%;
32 }
33
34 .header,
35 span {
36     color: var(--green);
37 }

```

Figure 10.0: Style.css file code (1)

```
39  .content {
40    flex: 1;
41    display: flex;
42    justify-content: center;
43    align-items: center;
44    font-style: sans-serif;
45  }
46
47
48  .nav {
49    display: flex;
50    justify-content: space-between;
51    height: 80px;
52    background-color: var(--gray);
53    align-items: center;
54    padding-left: 10%;
55    padding-right: 10%;
56  }
57
58  .recorder {
59    position: relative;
60    display: flex;
61    flex-direction: column;
62    align-items: center;
63  }
```

Figure 10.1: Style.css file code (2)

```

65  .recorder button {
66      padding: 0.5rem 0.5rem;
67      width: 40;
68      height: 40;
69      background-color: var(--orange);
70      color: var(--white);
71      border: transparent;
72      border-radius: 100%;
73
74      display: flex;
75      align-items: center;
76      justify-content: center;
77
78      position: absolute;
79      bottom: 0;
80      transform: translateY(50%);
81      z-index: 10;
82      cursor: pointer;
83  }
84
85  .hidden {
86      display: none !important;
87  }
88
89  .error {
90      color: var(--orange);
91      font: sans-serif;
92      font-weight: bold;
93  }

```

Figure 10.2: Style.css file code (3)

```

101  .recorder,
102  microphone {
103      background-color: var(--orange);
104      color: var(--white);
105
106  }
107
108
109  select {
110      width: 200px;
111  }

```

Figure 10.3: Style.css file code (4)

App.py

An individual document is made to specify the path and route of the application, to make associations with other documents and models.

It is to be noted, that I had about 4 options to store the message history (conversation between ChatGPT and the user):

1. The messages list could be put in a file, and each file could be matched with a specific user.
2. Similar to the first option, instead of using files, a database system could be integrated.
3. Messages can be stored on request, meaning each connecting user can have history for a short duration only.
4. Lastly, no history of the messages can also be an option, meaning each new message becomes considered as the first.

I chose option 2, for its convenience. Hence, this file contains the functional code needed for the microphone to process Bengali audio and to save it, with a file path and specifications for the storage process.

```
app.py > ...
1  import json
2  from uuid import uuid4
3  from flask import Flask, render_template, request, Response
4
5  from chat import full_flow, text_to_speech, speech_to_text
6
7  app = Flask(__name__)
8
9
10 @app.route("/")
11 def homepage():
12     return render_template("index.html")
13
```

Figure 11.0: App.py file code (1)

```

22 @app.route("/upload", methods=["POST"])
23 def upload_speech():
24     user_input = request.files["voice_msg"]
25     file_path = f"temp/{uuid4()}.wav"
26     user_input.save(file_path)
27
28     # Do something like recognize text and onwards
29     # bn_text = speech_to_text(file_path)
30
31     # so on
32     # chatgpt_reply, messages_list = full_flow(bn_text)
33
34     # bn_audio_path = text_to_speech(chatgpt_reply)
35
36     # bn_audio_array = text_to_speech(chatgpt_reply)
37     bn_audio_array = text_to_speech(
38         "আমি বুঝতে পারি যে সংকটের সময় অসহায় বোধ করা কঠিন হতে পারে, কিন্তু সাহায্য করার জন্য আমরা এখনও পদক্ষেপ নিতে পারি। আমরা ক্ষতিগ্রস্তদের
39     )

```

Figure 11.1: App.py file code (2)

```

37     bn_audio_array = text_to_speech(
38         "আমি বুঝতে পারি যে সংকটের সময় অসহায় বোধ করা কঠিন হতে পারে, কিন্তু সাহায্য করার জন্য আমরা এখনও পদক্ষেপ নিতে পারি। আমরা ক্ষতিগ্রস্তদের
39     )
40
41     return Response(
42         content_type="application/json",
43         # response=json.dumps({"reply": chatgpt_reply, "audio": bn_audio_array.tolist()}),
44         response=json.dumps({"reply": "", "audio": bn_audio_array.tolist()}),
45     )
46

```

Figure 11.2: App.py file code (3)

Chat.py

All the code used to test the language and speech recognition models earlier are repeated here for quick response. Firstly, all required libraries are imported as was seen before, along with the previously used authorization and API keys.

Then under 3 different functions, 3 important set of actions and commands are performed.

```

def text_to_speech(given_text):
    (function) def text_to_speech(given_text: Any) -> Any
    Takes: given_text : Text to convert to speech
    Returns:
    audio byte array: ndarray of the generated speech audio

```

Figure 12.0: This shows how the yellow code is referring to the definition behind each function

'Def full_flow'

the function that holds the merge of the ChatGPT chatbot and the language models as exactly shown previously when tested in Jupyter Notebook. Briefly, under this function the Bengali user message is taken as input (which can be stored in the messages list), converting it to English text for ChatGPT, the response from the chatbot is then converted to Bengali text for the user. All of this returns a Bengali reply/output (bn_reply) and a message list that is newly updated or created.

'Def speech_to_text'

This function exclusively deals with the Bengali speech-to-text model, by tracking the file path of the audio file (Bangla speech input) to provide output in text.

'Def text_to_speech'

This function deals with the Bangla text-to-speech model, which converts the Bangla translated response from ChatGPT and provides output in speech by sending in the audio file which gets saved as numpy array with a file path.

```
chat.py > ...
2  from banglaspeech2text import (
3      Model as STTModel,
4      available_models as stt_available_models,
5  )
6  from banglattts import BanglaTTS
7  from translators import translation
8  from uuid import uuid4
9  import openai
10 import requests
11
12
13 openai.api_key = "sk-ZdG1JcNMHmXMgoBJEPcT3B1bkFJwz9dZm83hNR7MYOPqrAP"
14 bn_en_API_URL = (
15     "https://api-inference.huggingface.co/models/csebuetnlp/banglat5_nmt_bn_en"
16 )
17 en_bn_API_URL = (
18     "https://api-inference.huggingface.co/models/csebuetnlp/banglat5_nmt_en_bn"
19 )
20 auth_headers = {"Authorization": "Bearer hf_RbRXW0ytHBWeDBu0ZgZFPCsZmgZpnHirmX"}
```

Figure 12.1: Chat.py file code (1)

```

23 def full_flow(user_bn_msg, messages_list=None):
24     """
25     Takes:
26     'user_bn_msg': Message in bangla to send
27     'messages_list': Optional. List of message history.
28
29     Returns:
30     'bn_reply': reply from chatgpt in bangla
31     'messages_list': updated or newly created message history
32
33     ----
34
35     Translates user_bn_msg to english, sends to chatgpt, gets a reply,
36     translates to bangla, then returns the bn_reply and updated messages_list
37     """

```

Figure 12.2: Chat.py file code (2)

```

39     # response = requests.post(
40     #     bn_en_API_URL, headers=auth_headers, json={"inputs": user_bn_msg}
41     # )
42     # english_user_msg = response.json()[0]["translation_text"]
43     english_user_msg = translation.translate_to_en(user_bn_msg)
44     print(english_user_msg)
45
46     if not messages_list:
47         messages_list = [
48             {"role": "system", "content": "You are a kind helpful assistant."},
49         ]
50
51     # Adds english user msg to messages_list and sends to chatgpt
52     messages_list.append(
53         {"role": "user", "content": english_user_msg},
54     )
55     chat = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=messages_list)

```

Figure 12.3: Chat.py file code (3)

```

58     reply = chat.choices[0].message.content
59     print(f"ChatGPT: {reply}")
60     messages_list.append({"role": "assistant", "content": reply})
61
62     # Translate chatgpt english reply to bangla
63     # response = requests.post(
64     #     en_bn_API_URL, headers=auth_headers, json={"inputs": reply}
65     # )
66     # bn_reply = response.json()[0]["translation_text"]
67     bn_reply = translation.translate_to_bn(reply)
68     print(bn_reply)
69
70     # return bangla reply and messages_list
71
72     return bn_reply, messages_list

```

Figure 12.4: Chat.py file code (4)

```

75 def speech_to_text(file_path):
76     """
77     Takes:
78     'file_path': Audio file path to convert to text from
79
80     Returns:
81     text: output converted text from speech
82     """
83
84     # Load a model
85     models = stt_available_models()
86     model = models[0]
87     model = STTModel(model)
88     model.load()
89
90
91     output = model.recognize(file_path)
92
93     print(output) # output will be a dict containing text
94     return output["text"]

```

Figure 12.5: Chat.py file code (5)

```

115 def text_to_speech(given_text):
116     """
117     Takes:
118     'given_text': Text to convert to speech
119
120     Returns:
121     audio byte array: ndarray of the generated speech audio
122     """
123
124     # Define the model storage location for reuse
125     tts = BanglaTTS(save_location="tts_model")
126
127     # Get an audio file path after generating the speech from the given text
128     audio_ndarray = tts.convert_numpy(text=given_text)
129
130     return audio_ndarray

```

Figure 12.6: Chat.py file code (6)

4. Validation

It is important to prove whether the application is serving its purpose clearly. So in this section, I will share the outcomes it provides with real life usage. I will change the context and provided various prompts to get accurate answers from this application, like ChatGPT works.

One of the previously mentioned and default context that had been used was, “You are a kind and helpful assistant”. On usage, I asked several basic questions on Bangladesh that took time to process and understand. However, I changed the content context with “You are a kind and helpful

expert from Bangladesh who knows a lot about the country”. This change provided more satisfactory answers, relatively faster.

KothaGPT	KothaGPT
বাংলাদেশের চাতিয় পাখিকি।	বাংলাদেশে জাতীয় খাদ্যকী।
বাংলাদেশে বিভিন্ন ধরনের পাখি পাওয়া যায়। বাংলাদেশে সবচেয়ে বেশি দেখা যায় এমন পাখির মধ্যে রয়েছে: ১. সাদা গলা মাছরাঙা ২. প্রাচ্যের ম্যাগপাই-রবিন ৩. কালো ডোঙ্কো ৪. সোয়াস্প ফ্রাঙ্কলিন ৫. বেগুনি সানবার্ড ৬. ডাক্কি ওয়ারবলার ৭. দাগযুক্ত মুনিয়া বা নীলকণ্ঠ ৮. সাধারণ ময়না ১০. লালপেট বুলবুলি এছাড়াও ঈগল, ফ্যালকন, পঁচা, পেলিকান, সারস এবং আরও অনেক প্রজাতির পাখি রয়েছে। বাংলাদেশ তার সমৃদ্ধ জলাভূমির জন্য পরিচিত, যেখানে প্রচুর সংখ্যক পরিযায়ী পাখির বাসস্থান।	বাংলাদেশের জাতীয় খাবারকে "খিচুড়ি" (এছাড়াও "খিচুড়ি" বলা হয়) বলা হয়। এটি চাল, ডাল এবং বিভিন্ন মশলা দিয়ে তৈরি একটি খাবার। এটি সাধারণত ভাজা স্বজি বা মাছের তরকারি দিয়ে পরিবেশন করা হয়, এবং প্রায়শই উৎসব বা উদ্‌যাপনের সময় খাওয়া হয়। খিচুড়িকে একটি সান্ত্বনা খাবার হিসাবেও বিবেচনা করা হয় এবং অসুস্থ বা অসুস্থতা থেকে সুস্থ হয়ে ওঠা মানুষের জন্য পরিবেশন করা হয়, কারণ এটি সহজে হজম করা যায় এবং পুষ্টিকর।
	

Figure 13.0: As an expert from Bangladesh, the answers provided and translated

With the context given, it was able to answer quite precisely given the inaccurate spellings provided by the Bangla speech-to-text model. Regardless, it did require a few tries to train for the right response. It was able to answer all the questions related to Bangladesh more certainly because of the context provided.

Now, given this context itself I can ask it about something absolutely unrelated to the country.

Below, I asked ‘What is the national bird of China?’ which was not processed well by the speech-to-text model and was received as ‘China is a species of bird’. It was also not matching the context of Bangladesh. Hence, it responded with, ‘I’m sorry, but that statement is incorrect. China is actually a country located in East Asia. Would you like to know more about Bangladesh or any other topic related to the country?’

চীন এ জাতীয় পাখিকী।

আমি দুঃখিত, কিন্তু এই বিবৃতিটি ভুল। চীন আসলে পূর্ব এশিয়ায় অবস্থিত একটি দেশ। আপনি কি বাংলাদেশ বা দেশের সাথে সম্পর্কিত অন্য কোন বিষয় সম্পর্কে আরও জানতে চান?

Figure 13.1: “What is the national bird of China?” (Given the context that assistant is from Bangladesh)

আমার অনেক পেট বাথা কিখাব।

আমি এটা শুনে দুঃখিত। আপনি কি আপনার উপসর্গগুলো এবং কতদিন ধরে এগুলো হচ্ছে সে সম্পর্কে আমাকে আরেকটু বলতে পারবেন? এটা আমার জন্য সাহায্যকারী হবে আপনার মেডিকেল ইতিহাস এবং পাকস্থলীর সমস্যাগুলির জন্য আপনার যে কোনও অতীত রোগনির্ণয় বা চিকিৎসা সম্পর্কে জানতে। এভাবে, আমি আপনার পরিস্থিতি আরও ভালভাবে মূল্যায়ন করতে এবং আপনার জন্য সম্ভাব্য চিকিৎসার পরামর্শ দিতে পারি।

Figure 13.2: “My stomach hurts. What should I eat?” (Given the context that assistant is a doctor from America)

Next, I gave a completely different context like, “You are a kind helpful doctor from America who knows a lot about curing sickness”. I had actually put the speech input for “My stomach hurts. What can I eat?”.

While it was not able to grasp the latter part, it still processed that I have stomach problems and wished to assist like a doctor with further investigation. It responded with, “ I'm sorry to hear that. Can you tell me a little bit more about your symptoms and how long you've been having them? It would be helpful for me to understand your medical history and any past diagnoses or treatments

you may have had for your stomach issues. That way, I can better assess your situation and suggest possible treatment options for you”.

For more practical usage, I used this application in a manner most people use ChatGPT. Such as for educational purposes, like creating a summary of a story, with a given prompt of “You are a kind and helpful expert in English Literature and know a lot about historical events around the world”.

ফ্রাইকেনস্টায় নিয়ে একটা শারসংখ্যেব রিখুল।

প্রিয় ভাই ও বোনেরা, আজ আমি মেরি শেলির মাস্টারপিস - ফ্রাইকেনস্টাইন সম্পর্কে আমার চিন্তাধারা আপনাদের সাথে শেয়ার করতে চাই। এই উপন্যাসটি কেবল আতঙ্কের গল্প নয়, বরং মানুষের প্রকৃতি, বিজ্ঞানের সীমানা এবং সৃষ্টির শক্তির গভীর অনুসন্ধান। তিনি এটি জীবনে নিয়ে আসতে সফল হন, কিন্তু শীঘ্রই তার সৃষ্টির ভয়াবহ পরিণতি উপলব্ধি করেন। এই দৈত্যটি সমাজের দ্বারা প্রত্যাখ্যান করা হয় এবং আমাদের নিজেদের ভুল এবং অসিদ্ধতার প্রতিফলন। এটি আমাদের জন্য একটি সতর্কবাণী যে আমরা সবাই ভাল এবং মন্দ হতে পারি, এবং এটি আমাদের চারপাশের বিশ্বের জন্য একটি দুঃখজনক সমাপ্তি। এটি আমাদের জন্য একটি ব্যাখ্যা যে আমরা কিভাবে আমাদের থেকে আলাদা হতে পারি। এটি একটি ব্যাখ্যা যে, এই দৈত্যটি আমাদের

Figure 13.3: ‘Write me a summary on Frankenstein’ (Given the context that assistant is an expert on English Literature)

Afterwards, I also ask for the first writer in Bangladesh which it did not completely grasp, yet the lengthy response involves my needed answers quite easily. It discusses in details about the writers in Bangladesh, focusing on the prompt given ‘historical events’ which is attached to the generated response. Hence, if the language model is given very specific fine-tuning for the purpose, it responses in a more desired manner. Further, the translation both in terms of text and speech may be slightly sloppy, but the answer remains true to its essence.

বাংলাদেশে প্রথম লাখোক্ষে।

বাংলাদেশে স্বাগতম! বাংলাদেশ দক্ষিণ এশিয়ার একটি সুন্দর দেশ, যার সীমানা ভারত, মায়ানমার এবং বঙ্গোপসাগর। এর একটি সমৃদ্ধ ইতিহাস এবং সংস্কৃতি রয়েছে যা প্রাচীন কাল থেকে চলে আসছে। আপনি যদি সাহিত্যের প্রতি আগ্রহী হন, তাহলে আপনার অনেক বিখ্যাত লেখক এবং কবি রয়েছে যারা বাংলা সাহিত্যে প্রচুর অবদান রেখেছেন। কিছু বিখ্যাত নাম হল রবীন্দ্রনাথ ঠাকুর, কাজী নজরুল ইসলাম এবং জসীমউদ্দীন। বাংলাদেশের একটি আকর্ষণীয় ইতিহাস রয়েছে, যা ব্রিটিশ শাসনের অধীনে ছিল এবং ১৯৭১ সালে স্বাধীনতা অর্জনের আগে পাকিস্তানের অংশ ছিল। এছাড়াও, বাংলাদেশের একটি অনন্য রন্ধনশৈলী রয়েছে যা ভারতীয় এবং দক্ষিণ-পূর্ব এশিয়ার স্বাদের একটি সুস্বাদু মিশ্রণ। কিছু জনপ্রিয় খাবারের মধ্যে রয়েছে। আমি আশা করি আপনি বাংলাদেশে আপনার সময় উপভোগ করবেন।

Figure 13.4: ‘Who is the first writer in Bangladesh’ ((Given the context that assistant is an expert on English Literature.)

ChatGPT provided an interesting response when asked to write a poem about sadness. It referred to a writer's poem, which my used pre-trained model was able to translate fairly accurate.

আমাকে দুঃখ নিয়ে আকটা কবিতা লিখে দাও।

[illegible]

Figure 13.5: ‘Write me a poem about sadness’ (Given the context that assistant is an expert on English Literature..)

4.1 Limitations

Despite responding relatively correctly with desired answers, there are many errors and mistakes. Most importantly, as observed the speech to text model is incapable of fully adhering the Bengali speech input, unable to process the questions. This is one big constraint that disables the ChatGPT response to be desirable, and of course the Bengali translated response is then also misleading.

Moreover, the lack of features on the application may make it inflexible for the user. I did wish to integrate an option, another page which would save all the queries and could be reviewed by the user. Therefore, now, the user does not get to access the previous back and forth conversation from the message history I accumulated for the purpose. Instead, the user has to ask the previous questions again. That is also a discrepancy for the user to store information visibly on the application.

On the other hand, the user does not have an option for settings to perhaps adjust the microphone sound or flexible functionality.

One of the biggest constraints remain to be the speed of the response. It is very time consuming as the output tends to be both Bengali speech and text, while processing steps one by one in the back-end. Hence, it may not be favorable for users to wait and lose patience.

On contrast, ChatGPT's constraints should also be mentioned. I am not accountable for its discrepancies, but it greatly affects the quality of the services my application offers. A big constraint is ChatGPT is still not perfect, and will continue to improve with machine learning advancements. Its understanding remains to be that of a machine so it cannot process human thoughts and ideas like human minds. In addition, biasness is also huge limitation that comes in the way to deliver an open-ended answer (*Khoshafah, 2023*).

For further discussion on biasness from ChatGPT, another relevant research was conducted. Upon testing, for example it was discovered that the voice assistant was more lenient towards positive results in favor of democratic politicians, at times overlooking the general population. This testing was based on creating limericks (short poems) for both conservative and liberal politicians by ChatGPT. In conclusion, the AI assistant was making more positive limericks for liberal

politicians, while making negative limericks for conservative politicians which shows a significant amount of partially and unfairness (McGee, 2023).

4.2 Recommendations

After studying all the limitation, I wish to create a few recommendations for myself to continuously improve this application to actually come in handy for Bengali users in their daily lives. I wish to integrate a page for ‘saved list’ holding all the previous questions and answers for the user to refer back whenever needed. I wish to integrate a settings option to choose preferred microphones, or test and adjust sound etc. Overall, I wish to make a more stylish and presentable user interface for the user, similar to ChatGPT. Due to time limitations, my current application is simply the main idea from where I can add further practical functionalities, specifications and features etc.

Most importantly, I wish to make the process speedy and reliable for the user. In addition, I wish to integrate a better speech-to-text model for better performance and accuracy.

Lastly, it is important to state that I do not have sufficient skills and knowledge to develop a professional natural processing system via web application.

5. Conclusion

Although ChatGPT is taken over the world rapidly, it is still not accessible or known to many people in Bangladesh. To open the convenient and powerful chatbot to the people of Bangladesh is much needed task for us. Hence, without knowing how to speak and write English many Bengali smartphone/ computer users can freely use ‘KothaGPT’. This project report basically covers 5 chapters talking about the problem, the solution and purpose, and the details of making this web application. In the first chapter, I discuss why do we really need a chatbot exclusively for the Bengalis. Secondly, we state the various references and previous similar works. The latter part discusses the developing processes of the web application in extensive details, while also sharing its limitations and future fixings that can be done.

In conclusion, this web application is not perfect and it can be more consistent. Despite that, it serves to solve a big problem and help the people of our country to be more educated and updated, strongly helping in improving digital literacy.

6. References

- Bang, Y., Cahyawijaya, S., Lee, N., Dai, W., Su, D., Wilie, B., . . . Fung, P. (2023). A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity.
- Bhattacharjee, A., Hasan, T., Ahmad, W. U., & Shahriyar, R. (2022). BanglaNLG and BanglaT5: Benchmarks and Resources for Evaluating Low-Resource Natural Language Generation in Bangla.
- Bhattacharjee, A., Hasan, T., Ahmad, W. U., Samin, K., Islam, M. S., Iqbal, A., . . . Shahriyar, R. (2022). BanglaBERT: Language Model Pretraining and Benchmarks for Low-Resource Language Understanding Evaluation in Bangla.
- Caldarini, G., Jaf, S., & McGarry, K. (2022). A Literature Survey of Recent Advances in Chatbots. *Information*, 13(1), 41. doi: 10.3390/info13010041
- Černý, M. (2022). The history of chatbots: the journey from psychological experiment to educational object. *Journal of Applied Technical and Educational Sciences jATES*. doi: 10.24368/jates322
- Dabre, R., Shrotriya, H., Kunchukuttan, A., Puduppully, R., Khapra, M. M., & Kumar, P. (2022). IndicBART: A Pre-trained Model for Indic Natural Language Generation.
- Guzman, F., Chen, P.-J., Ott, M., Pino, J., Lample, G., Koehn, P., . . . Ranzato, M. A. (2019). The FLORES Evaluation Datasets for Low-Resource Machine Translation: Nepali–English and Sinhala–English.
- Islam, I., & Islam, M. N. (2023). Opportunities and Challenges of ChatGPT in Academia: A Conceptual Analysis. doi: 10.22541/au.167712329.97543109/v1
- Jiao, W., Wang, W., Huang, J.-t., Wang, X., & Tu, Z. (2023). Is ChatGPT A Good Translator? YesWith GPT-4 As The Engine.
- Jolley, J. R., & Maimone, L. (2022). Thirty Years of Machine Translation in Language Teaching and Learning: A Review of the Literature. *L2 Journal*, 14(1). doi: 10.5070/l214151760
- Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., . . . Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103, 102274. doi: 10.1016/j.lindif.2023.102274

Khoshafah, F. (2023). ChatGPT for Arabic-English Translation: Evaluating the Accuracy. doi: 10.21203/rs.3.rs-2814154/v2

King, M. R. (2023). A Conversation on Artificial Intelligence, Chatbots, and Plagiarism in Higher Education. *Cell Mol Bioeng*, 16(1), 1-2. doi: 10.1007/s12195-022-00754-8

Koehn, P., & Knowles, R. (2017). Six Challenges for Neural Machine Translation.

Kung, T. H., Cheatham, M., Medenilla, A., Sillos, C., De Leon, L., Elepano, C., . . . Tseng, V. (2023). Performance of ChatGPT on USMLE: Potential for AI-assisted medical education using large language models. *PLOS Digit Health*, 2(2), e0000198. doi: 10.1371/journal.pdig.0000198

Liddy, E. D. (2001). <Natural Language Processing.pdf>.

Lyu, C., Xu, J., & LongyueWang. (2023). New Trends in Machine Translation using Large Language Models: Case Examples with ChatGPT.

McGee, R. W. (2023). Is Chat GPT Biased against Conservatives? An Empirical Study.

Moslem, Y., Haque, R., Kelleher, J. D., & Way, A. (2023). Adaptive Machine Translation with Large Language Models.

Mumin;, M. A. A., Seddiqui;, M. H., Iqbal;, M. Z., & Islam;, M. J. (2019). shu-torjoma: An English↔Bangla Statistical Machine Translation System.

Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). Natural language processing: an introduction. *J Am Med Inform Assoc*, 18(5), 544-551. doi: 10.1136/amiajnl-2011-000464

Ouyang, L., Jeff Wu, X. J., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., . . . Lowe, R. (2022). Training language models to follow instructions with human feedback.

Qin, C., Zhang, A., Zhang, Z., Chen, J., Yasunaga, M., & Yang, D. (2023). IS CHATGPT A GENERAL-PURPOSE NATURAL LANGUAGE PROCESSING TASK SOLVER?

Shahriar, S., & Hayawi, K. (2023). Let's have a chat! A Conversation with ChatGPT: Technology, Applications, and Limitations.