**Project:** AI Nurse Assistant - **Understanding Document**

**Day 1:** 19-June-2024 (Wednesday)

**Task1:** Requirement Analysis (Query Analysing and Prompt Generation)

Inside the AI Nurse Assistant, we got Query Analysing and Prompt Generation Module. Initially I struggled (which persists for almost two days) to find what is the exact usecase and core functionalities of my module. I started to understand “what a prompt can be?” in general by reading the below blog.

Reference : <https://medium.com/@martinkeywood/llm-prompt-engineering-going-beyond-the-basics-and-how-you-can-mitigate-against-confusion-586319b03ad3>

After that I got to know a blog by Goolge Cloud - “Introduction to Prompting” , Which gives me the clarity about “what a prompt is” and “what are the context that most prompts contains?” like Tasks, System Instructions, Few shot examples and contextual information. And navigated to their other blogs to know about “how to define task specific prompts?” like code generation prompts, chat prompts, and so on.

Reference : [https://cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/introduction-prompt-design#:~:text=A%20prompt%20is%20a%20natural,model%20to%20complete%20or%20continue](https://cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/introduction-prompt-design" \l ":~:text=A prompt is a natural,model to complete or continue).

After Reading the above blogs, I got some clarity about what is a prompt and how it is used in LLM to elicit the accurate responses from them. Next up I moved towards the usecase of our project which is appointment scheduling, medicine refill, and so on and understood how is these usecases are generally done by a chatbot. And To see them in action I refered to some existing solutions available online to have a better understanding. This existing solutions include Landbot – a chatbot building application, Clara – A chatbot by “redcrossblood.org” which is used to answer patient queris about blood donation eligibility, process, impacts.. and can schedule appointment for blood donation.

LandBot: https://landbot.io/chatbots

Clara: https://www.redcrossblood.org/

Next up, we gathered up for a team meeting with TL’s and fellow developers. In which we are involved in a brain stroming session with TL’s to know about what is the overall function flow of our AI Nurse Assistant. After that they connected each module along with their overall functionalities by taking a appointment scheduling usecase and explained it incase of each module’s usecase.

After the meeting, I looked for some static prompt implementations for appointment scheduling, medicine refill usecases which is good and navigate user on a straight way to their usecases, but they are lack in flexibility as they are predefined for specific usecases. Here is a code implementation for static prompting.

**Code Implementation (Static Implementation):**

def generate\_prompt(user\_query, context=None):

**# Define intent keywords**

intents = {

'schedule\_appointment': ['schedule an appointment', 'book an appointment'],

'reschedule\_appointment': ['reschedule', 'change appointment'],

'check\_availability': ['available slots', 'availability'],

'confirm\_slot': ['confirm slot', 'choose slot'],

'medical\_refill': ['refill', 'prescription'],

'urgent\_refill': ['urgent', 'ASAP'],

'confirmation\_email': ['confirmation email', 'send email']

}

**# Determine the intent**

intent = None

for key, keywords in intents.items():

for keyword in keywords:

if keyword in user\_query.lower():

intent = key

break

if intent:

break

**# Generate prompt based on the intent**

if intent == 'schedule\_appointment':

prompt = "The user wants to schedule an appointment. Ask for the doctor's name and the preferred date and time for the appointment."

elif intent == 'reschedule\_appointment':

prompt = "The user wants to reschedule their appointment to a different date. Confirm the new date and time."

elif intent == 'check\_availability':

prompt = "The user wants to know the available appointment slots for [doctor's name] on [date]. Provide the available times."

elif intent == 'confirm\_slot':

prompt = "The user has chosen the [time] slot for their appointment. Confirm the booking and ask if they want a reminder."

elif intent == 'medical\_refill':

prompt = "The user needs a refill for their prescription. Ask for the name of the medication and the dosage."

elif intent == 'urgent\_refill':

prompt = "The user needs the refill urgently, by [timeframe]. Confirm the urgent request and inform about the process."

elif intent == 'confirmation\_email':

prompt = "The user wants a confirmation email for the refill request. Ask for their email address."

else:

prompt = "I'm sorry, I didn't understand that. Can you please provide more details?"

return prompt

**# Example usage**

user\_query = "I need to schedule an appointment."

generated\_prompt = generate\_prompt(user\_query)

print(generated\_prompt)

Next up as stated above we can’t use static prompt generation because it is lagging the flexibility that we expect, so that we moved to the idea of generating prompts dynamically. Inorder to generate prompt dynamically we need to use a ML or DL model as they are the preferred approach to do the same. From here I was more engaged to finding a model that best suits for our usecase.

**Day2:** 21-June-2024 (Friday)

**Task1:** LLM Selection

This day started with the standup meeting inorder to identify the understanding of each team about their module and addressing any difficulties that each team have. And my take in this meeting is to clarify the confusions that i have in my module and got a better understanding about the functional flow of the module.

After that I had a conversation with subash who is working on LLM Module about how their LLM module is differ from ours in case of functionalities and how they differ interm of usecase also.

After that I looked for LLMs that are available for our usecase in huggingface including Text2Text Generation, Text Generation and Question Answering categories. And after spending some time i found BlenderBot, Meta – Llama, and DistilGPT-2 models for our usecase and after reading their model cards I got a idea about how use use them.

DistilGPT-2: <https://huggingface.co/distilbert/distilgpt2>

Meta Llama 3 8b Instruct: <https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct>

After that I implemented the DistilGPT-2 on vscode by using transformers and torch packages and found that they are not working properly. Here you can find a sample code here

**Code Implementation:**

from transformers import GPT2LMHeadModel, GPT2Tokenizer

**# Load the DistilGPT-2 model and tokenizer**

model\_name = 'distilgpt2'

tokenizer = GPT2Tokenizer.from\_pretrained(model\_name)

model = GPT2LMHeadModel.from\_pretrained(model\_name)

**# Example input text**

input\_text = "I want to book an apppointment.This is the user query and i want you to Generate the accurate prompt for this which is going to be feed to the LLM"

**# Tokenize input text**

input\_ids = tokenizer.encode(input\_text, return\_tensors='pt')

**# Generate text based on the input\_ids**

output = model.generate(input\_ids, max\_length=200, num\_return\_sequences=1,

pad\_token\_id=tokenizer.eos\_token\_id)

**# Decode the generated output and print it**

generated\_prompt = tokenizer.decode(output[0], skip\_special\_tokens=True)

print("Generated Prompt:", generated\_prompt)

This code generates the same input that i given as a input. So I looked for some other alternative which is Meta Llama Model here. But before implementing Meta Llama I need a access to it. So did the same and wait for some time to grant access.

After that We gathered for a meeting again and discussed about overall team progress for the day and Clarified our doubts about our module. I raised a query that i need to clarify which is “How a optimal prompt will look like” and understands the same. And got some doubts about function calling module about how it works.

After that I headed towards downloading the Meta Llama 3 8b Instruct Model’s files which includes totally 15GB of data which took some time to complete it. I used to sample inference code of llama to initiate the downloading process which can be found here.

**Code Implementation:**

import transformers

import torch

import time

model\_id = "meta-llama/Meta-Llama-3-8B-Instruct"

pipeline = transformers.pipeline(

"text-generation",

model=model\_id,

model\_kwargs={"torch\_dtype": torch.bfloat16},

device\_map="auto",

)

messages = [

{"role": "system", "content": "You are a pirate chatbot who always responds in pirate speak!"},

{"role": "user", "content": "Who are you?"},

]

terminators = [

pipeline.tokenizer.eos\_token\_id,

pipeline.tokenizer.convert\_tokens\_to\_ids("<|eot\_id|>")

]

outputs = pipeline(

messages,

max\_new\_tokens=256,

eos\_token\_id=terminators,

do\_sample=True,

temperature=0.6,

top\_p=0.9,

pad\_token\_id = pipeline.tokenizer.eos\_token\_id

)

print(outputs[0]["generated\_text"])

After running it It shows me the error that It’s private model so you need to authenticate to use this model. So I created a account in huggingface and generated a token for the same and used it to authenticate with huggingface\_cli.

**Day3:** 24-June-2024(Monday)

**Task1:** LLM Selection and Inference

As to continue the previous day progress I started to again running the Meta Llama 3 8b Instruct Model on inference phase. After downloading the model, the model took too much time to produce the response. So that I considered to move with some Small Language Models like Roberta, Albert models as they can easily loaded within the minimal hardware resources.

After that we gathered for a Morning status meeting, In where I proposed my issue with LLM as they require high amount hardware resources (GPU). And currently I’m working on the way to find SLMs and use them. After being said my query I got some feedback that if use are going to use SLMs you can try with Microsoft’s Phi-3 Model.

So After the meeting concluded i went to implement the SLMs that I previously founded which is Roberta. After Implementing Roberta It still gives some uwanted content as a response to me. Here i also attached the inference code that i used.

**Code Implementation:**

from transformers import RobertaTokenizer, RobertaForCausalLM

# Load pre-trained model and tokenizer

tokenizer = RobertaTokenizer.from\_pretrained('roberta-base')

model = RobertaForCausalLM.from\_pretrained('roberta-base', is\_decoder = True)

prompt = "Hi, How are you?"

inputs = tokenizer(prompt, return\_tensors="pt")

outputs = model.generate(\*\*inputs, max\_new\_tokens = 100)

generated\_prompt = tokenizer.decode(outputs[0], skip\_special\_tokens=True)

print("Generated Prompt:", generated\_prompt)

After getting the same inaccuracies, I decided to move with the feedback that i got during the meeting which is Phi3. In the meantime I looked for some sample conversations with user query, prompt, response in ChatGPT to understand how the prompts are changing over each usecases.

After that I moved to the huggingface website to see how to use Phi-3 model in VS code by seeing it’s model card. And I also found the sample code there and implemented it in code using torch and transformers library. Here you can find the sample code for that.

**Code Implementation:**

import torch

from transformers import AutoModelForCausalLM, AutoTokenizer, pipeline

torch.random.manual\_seed(0)

model = AutoModelForCausalLM.from\_pretrained(

"microsoft/Phi-3-mini-4k-instruct",

torch\_dtype="auto",

trust\_remote\_code=True,

)

tokenizer = AutoTokenizer.from\_pretrained("microsoft/Phi-3-mini-4k-instruct")

messages = [

{"role": "user", "content": "Query: I want to book an appointment with Dr.Smith. Generate a prompt for this query"},

]

pipe = pipeline(

"text-generation",

model=model,

tokenizer=tokenizer,

)

generation\_args = {

"max\_new\_tokens": 500,

"return\_full\_text": False,

"temperature": 0.0,

"do\_sample": False,

}

output = pipe(messages, \*\*generation\_args)

print(output[0]['generated\_text'])

This code snippet also downloaded the required models and files from the huggingface website which takes some time and also crashed the system for some time. Again due to lack of GPU’s I system can’t able to process the model for inference and even to download it. But some how I managed to download them using the inference code but i can’t able to test it’s response as during the compilation the system crashes.

After that we gathered for the meeting and told our current status in search of LLM selection with Llama and Phi-3.

**Day4:** 25-June-2024 (Tuesday)

**Task1:** LLM Finalization and Logic Developement

In this day, I can be able to see the Llama Model working by help of subash who is also working on LLM Module. After that i tried to run the Llama model in my system which again took a long time to go and tried running this on Google Colab which is a cloud platform to run models online using some decent system requirements. Suddenly I can able to see that the Llama and Phi-3 models are working on Google Colab and realised the reason of why ther aren’t working on my machine which is system resources again. In the meantime I tried a SLM which is commonly used for Text Generation namely “facebook/opt-125m” which performs well on question answering and

text completion tasks.

After that we got our demo meeting in which the Tls are reviewed and saw what we have done so far. In my part, I told the system constraint that i encountered during running the model and It is working on Google colab. And told about the SLM opt-125m to the team. From now onwards we are done with the LLM selection as we found some decent models which are Llama – 3, Phi-3 are working nicely. We next started to focusing on the Logic for how to do **function calling** inorder to fetch relevant data for user query. After the end of the meeting Santhosh sir provided a approach which is “sequence based prompt generation” used to navigate the conversation for the specified task choosen.

After that I started to worked on the logic to generate prompts dynamically and gather data required from the user and send gathered response from system to users. I did a sample json format which consists the sequence for appointment scheduling and did the logic for the same and defined the strcuture and various processes in the flow.

**Code Implementation:**

import json

sequence\_json = {

'appointment\_scheduling': {

'check\_availability':{

'status': False,

'required': ['doctor\_name', 'date', 'specialization'],

'prompts' : {

'request': 'We have a request for check availability. To process the request we need list of user datas that provided below.',

'response': 'We did the availablity check for the user request and given the data relevant to the request.'

}

},

'initiate\_booking': {

'status': False,

'required': ['name', 'phone\_number', 'address'],

'prompts' : {

'request': 'We have a request for appointment booking initialization. To process the request we need list of user datas that provided below.',

'response': 'We did the request of initial booking and the given data relevant to the request.'

}

},

'confirm\_booking': {

'status': False,

'required': ['name', 'phone\_number', 'address', 'diagonosis', 'medical\_docs'],

'prompts' : {

'request': 'We have a request for appointment booking confirmation. To process the request we need list of user datas that provided below.',

'response': 'Your appointment booked successfully and the given data relevant to the request.'

}

}

}

}

def availability\_data():

return "Dr.Strange is available on 21-25th june 2024"

def booking\_availability\_check():

return "The requested slot is available"

def confirm\_booking():

return "Your appointment booked successfully"

collected\_entities = {}

current\_task = {}

while True:

user\_intent = input("Enter Intent:").lower()

if user\_intent == 'exit':

break

entity = input("Enter Available Entities (dict values):")

**## Collecting the entities from NLP module**

try:

entities = json.loads(entity)

except Exception as e:

print("Not a JSON")

**## Iterate the entities and store them for future use**

for key, value in entities.items():

collected\_entities[key] = value

# print("Stored Entities: ", collected\_entities )

**## Assigning the current task**

current\_task = sequence\_json[user\_intent]

**## Getting the current sequence to do**

current\_sequence = [[sequence, current\_task[sequence]] for sequence in current\_task if current\_task[sequence]['status'] == False ][0]

# print("current sequence: ", current\_sequence)

current\_sequence\_json = current\_sequence[-1]

**## Getting required parameters of current sequence**

collected\_entity\_list = list(collected\_entities.keys())

missing\_params = [entity for entity in current\_sequence\_json['required'] if entity not in collected\_entity\_list]

**# print("missing\_params : ", missing\_params)**

if missing\_params:

**## Prompt used to get the missed params for current sequence**

prompt = current\_sequence\_json['prompts']['request'] + "\nHere is the list of inputs that we need from the user. {}".format(missing\_params)

else:

**## Getting the relevant data from RAG**

avail\_data = availability\_data() if current\_sequence[0] == 'check\_availability' else booking\_availability\_check() if current\_sequence[0] == 'initiate\_booking' else confirm\_booking()

**## Setting that specific sequence as completed**

current\_task[current\_sequence[0]]['status'] = True

**## Generating the Prompt for LLM about the Data Retrived**

prompt = current\_sequence\_json['prompts']['response'] + "\nHere is the data for given request of the user. {}".format(avail\_data)

print("Generated Prompt: ", prompt)

**## Use the prompt to generate the response for LLM**

**## LLM code goes here**

After that we gathered for a meeting where we discussed about the current status of the module and I told the progress of the sequence based approach.

**Day 5:** 26-June-2024 (Wednesday)

In this day I wanted to verify the sequence flow that i developed so far is going on a right track. To do that i share the code that i developed to the both Tls. And Firstly I went to santhosh sir and asked about the structure that we developed then he explained about the flow that i given and give suggestion to handle the various senarios and suggested to improve and increase the number of sequences. Even he shared some sample sequences for appointment scheduling, cancel request and so on.

And I even clarified about the use of LLM in our module because I can’t able to identify the potential need for the LLM in my module. Santhosh sir also gave some scenarios that may involve the usage of LLM like checking entities with the LLM before proceeding with the RAG for data insertion or updation.

After that before gathering for the morning meeting I had same review session with nagarajan sir about the flow we are going. He also provided some suggestions on the flexibility side of the sequence. Then we gathered for a meeting in which i told the same progress and flow that we are going in.

Then I got some chance to work with subash who is working on LLM team inorder to help him to check the responses that the LLM gives are relevant or we need any improvements on the prompt and analysed the prompt inorder to fine tune the depth of the prompt. Then I started to work on the new sequence JSON creation for ‘cancel appointment’ which is driven through the sequence suggestions given. I spend some time inorder to understand the sequences improvements and developed the sample flow.

**Code Implementation:**

sequence\_json = {

'cancel\_appointment': {

'check\_upcoming\_appointment':{

'status': False,

'required': ['appointment\_id'],

'prompts' : {

'request': 'To process the request we need list of datas from users that provided below.',

'response': 'Here is the data for given appointment check request. Verify it with the patient.'

}

},

'get\_cancellation\_reason': {

'status': False,

'required': [],

'prompts' : {

'request': 'Collect patient about the reason for cancellation. And ask if they want to reschedule?',

'response': ''

}

},

'register\_cancel\_request':{

'status': False,

'required': ['cancelation\_reason', 'mobile\_number'],

'prompts': {

'request':'To process this request we need the list of datas from users that provided below',

'response': ''

}

},

'reply\_appointment\_status': {

'status': False,

'required': ['appointment\_id', 'cancelation\_reason'],

'prompts': {

'request': '',

'response': 'Here is the data for users cancellation request'

}

}

}

}

After that I spend some time with Thangam who is working on Image to Text module inorder to solve the issues that she is encountering with python packages and tried to solve it.

After that i back to my track to explore the improvements that i need to infuse into the structure. After that we gathered for the meeting with team for status update and updated the flow that i was working on.

**Day 6:** 27-June-2024 (Thursday)

In this day, I started to work on the Prompt Generator logic which decides when to call the RAG data and when to provide output to response to LLM and how to move from one sequence to another. And Inorder to do that I need do to some changes on the existing sample strcuture sequence that i developed for ‘cancel\_appointment’ sample case. So here you can find the modified JSON sequence for ‘cancel appointment’ usecase.

**Code Implementation:** (JSON structure)

sequence\_json = {

'cancel\_appointment': {

'collect\_appointment\_details':{

'status': False,

'isDataCollection': True,

'function\_call': None,

'required': ['appointment\_id'],

'prompts' : {

'request': 'To process the request we need list of datas from users that provided below.',

'response': None

}

},

'check\_upcoming\_appointment':{

'status': False,

'isDataCollection': False,

'function\_call': cancel\_appointment\_check,

'required': ['appointment\_id'],

'prompts' : {

'request': None,

'response': 'Here is the data for given appointment check request. Verify it with the patient.'

}

},

'get\_cancellation\_reason': {

'status': False,

'isDataCollection': True,

'function\_call': None,

'required': [],

'prompts' : {

'request': 'Collect patient about the reason for cancellation. And ask if they want to reschedule?',

'response': None

}

},

'register\_cancel\_request':{

'status': False,

'isDataCollection': True,

'function\_call': None,

'required': ['cancelation\_reason', 'mobile\_number'],

'prompts': {

'request':'To process this request we need the list of datas from users that provided below',

'response': None

}

},

'reply\_appointment\_status': {

'status': False,

'isDataCollection': False,

'function\_call': cancel\_appointment\_request,

'required': ['appointment\_id', 'cancelation\_reason'],

'prompts': {

'request': None,

'response': 'Here is the data for users cancellation request'

}

}

}

}

During this work going on, we got a call for the standup meeting and gathered for the same with team and in that i told about the changes that i did on the sequence JSON and about to work on the logic implementation that required for prompt generation. In the meantime, I have some small discussions with swetha who is working on my team and subash who is in LLM team inorder to brain storm about the implementation of Prompt Generater. And provided a overall picture about how the prompt generator is gonna like like to swetha.

After that I worked with the logic to implement the prompt generator having a thought in mind that this logic needs to be supported for all sequence based tasks that we are gonna use like schedule appointment, cancel appoinment, reschedule appointment, and medicine refills. This takes some time to develope it.

So far I developed the logic which is 3/4 of the work that i need to do for the current level of sequences that we are having. At next, we got our evening standup meeting in which we have to present a demo for Prabu sir and TL about the work that we done so far. In that demo session, I showed the viewers about the current status of the work also explained them how far we are in acheiving the milestones. Here is the code implementation of prompt generator logic you can find that i developed so far.

**Code Implementation:** (Prompt Generator)

def prompt\_generation(entity, intent, conv\_id):

## Getting whether we have an existing json for the given conv\_id

try:

entity = json.loads(entity)

except Exception as e:

entity = {}

if intent in ['appointment\_scheduling', 'cancel\_appointment', 'reschedule\_appointment']:

previous\_task = [conv[conv\_id] for conv in temp\_storage if (list(conv.keys())[0] == conv\_id and conv.get(conv\_id))]

## Checking whether we have that json or it's a fresh conv

entity = {key:value for key, value in entity.items()}

if not previous\_task:

## Creating a new task json for this conversation

sequence\_list = list(sequence\_json.keys())

sequence\_status = [False for \_ in sequence\_list]

total\_json = {'task': intent,'sequence\_list': sequence\_list, 'sequence\_status': sequence\_status, 'entity': entity}

temp\_storage.append({conv\_id: total\_json})

## Getting the current task for details

current\_task = [conv[conv\_id] for conv in temp\_storage if (list(conv.keys())[0] == conv\_id and conv.get(conv\_id))][0]

## Updating the entities of the current conversation

current\_task['entity'].update(entity)

## Getting the current sequence of the task

current\_sequence = current\_task['sequence\_list'][current\_task['sequence\_status'].index(False)]

## while there is a data collection sequence that can be skipped based on missed entity run this loop

while sequence\_json[current\_task['task']][current\_sequence]['isDataCollection']:

## Check if the current sequence missing any entities

missed\_entities = [enti for enti in current\_task['entity'] if enti not in sequence\_json[current\_task['task']][current\_sequence]['required']]

print("Missed Entities: ", missed\_entities)

## Check that the sequence have all required params

if not missed\_entities:

## If yes, then set the status of current sequence as True

current\_task['sequence\_status'][current\_task['sequence\_list'].index(current\_sequence)] = True

## Updating the short term memory about the conversation

temp\_storage[conv\_id] = current\_task

## Setting the new current sequence

current\_sequence = current\_task['sequence\_list'] [current\_task['sequence\_status'].index(False)]

else:

## If yes, then set the status of current sequence as True

current\_task['sequence\_status'][current\_task['sequence\_list'].index(current\_sequence)] = True

## Updating the short term memory about the conversation

temp\_storage[conv\_id] = current\_task

## Sending the prompt to LLM

return current\_sequence['prompts']['request'] + str(missed\_entities)

## If it is a non data collection task then to the needful

else:

## Call the relevant RAG

## Sending the prompt to LLM

return current\_sequence['prompts']['response']

So that’s the work that i done on this day. From next day i have to complete the remaining part of the prompt generator and check that’s everything is working properly.

**Day 7:** 28-June-2024 (Friday)

This day started with the team meeting where to share/report our current progress and what we are gonna do on today?. As to answer a question I stated that as we completed the 3/4th of the work for prompt generator logic for normal positiv flow of the sequence which is the current status of our module and I’m gonna work on the remaining part of the prompt generation logic so that we can have something which is a first level implemetation of prompt generator for sequence based tasks including appointment scheduling, cancel appointment and so on.

During the meeting the TL gave us some possible improvements that we can make our prompt generator more flexible to every usecases. As of now we only focused on defining the sequence that each task require and a logic to how the sequences are gonna happened in real time whether it is going to call the RAG API with relevant params or it just gonna gather the information (data collection sequence) from user to complete the sequence.

But we are maintaining the prompts as a static one which is not so good one while looking for the flexi bility. So the suggestion is to introduce some LLMs to do the prompt generation dynamic. And to use Langchain to create workflows. Keeping this in mind, I started to working on the same to complete the basic logic for prompt generation which requires some of the modification from the previous sequence that we are maintaining like small changes on function call parameter and isDataCollection parameter.

After working for some hours finally i did able to complete the basic flow which should work as expected for the positive case flow. And I went for the testing of the logic that i developed so far with some sample tasks including cancel appointment. This is where the real work came in, During the testing i initailly encountered some errors which took some to resolve them after spending some time on that. Finally I can able to see the logic is work as expected atleast for the basic positive flow. You can find the code impleementation for the prompt generator function here.

**Code Implementation:**

import json

## Cancel Appointment APIs

def cancel\_appointment\_check(payload):

assert type(payload) == dict, 'Payload should be a JSON'

appointment\_id = payload['appointment\_id']

return "The patient has an appointment of the given appointment\_id {}".format(appointment\_id)

def cancel\_appointment\_request(payload):

assert type(payload) == dict, 'Payload should be a JSON'

## payload data can be optional in some cases like initial data collection

appointment\_id = payload['appointment\_id']

cancelation\_reason = payload['cancelation\_reason']

return "We have cancellled the appointment with appointment\_id {} for {} reason according to your request.".format(appointment\_id, cancelation\_reason)

## Schedule Appointment APIs

def schedule\_appointment\_check(payload):

assert type(payload) == dict, 'Payload should be a JSON'

doctor\_name = payload['doctor\_name']

preferred\_date = payload['preferred\_date']

return "We have a slot availability with {} on {}".format(doctor\_name, preferred\_date)

def register\_appointment(payload):

assert type(payload) == dict, 'Payload should be a JSON'

patient\_name, phone\_number, address = payload["name"], payload["phone\_number"], payload["address"]

doctor\_name, preferred\_date, visit\_reason = payload["doctor\_name"], payload["preferred\_date"], payload['visit\_reason']

return f"Appointment Booked for {patient\_name} with {doctor\_name} at {preferred\_date} for the {visit\_reason} reason"

## General Information APIs - RAG calling

def general\_rag\_calling(payload):

assert type(payload) == dict, 'Payload should be a JSON'

user\_query = payload['user\_query']

## Calling the RAG function here and return the data

return "Here is the data for RAG retrived data for this user query {}".format(user\_query)

sequence\_json = {

'cancel\_appointment': {

'collect\_appointment\_details':{

'status': False,

'isDataCollection': True,

'function\_call': None,

'required': ['appointment\_id'],

'prompts' : {

'request': 'To process the request we need list of datas from users that provided below.',

'response': None

}

},

'check\_upcoming\_appointment':{

'status': False,

'isDataCollection': False,

'function\_call': cancel\_appointment\_check,

'required': ['appointment\_id'],

'prompts' : {

'request': None,

'response': 'Here is the data for given appointment check request. Verify it with the patient.'

}

},

## This step is skipping check a alternative to this

'get\_cancellation\_reason': {

'status': False,

'isDataCollection': False,

'function\_call': None,

'required': [],

'prompts' : {

'request': None,

'response': 'Collect patient detail about the reason for cancellation. And ask if they want to reschedule?'

}

},

'register\_cancel\_request':{

'status': False,

'isDataCollection': True,

'function\_call': None,

'required': ['cancelation\_reason', 'mobile\_number'],

'prompts': {

'request':'To process this request we need the list of datas from users that provided below',

'response': None

}

},

"confirm\_request\_details": {

"status": False,

'isDataCollection': False,

'function\_call': None,

"required": [],

"prompts": {

"request": None,

'response':"Please confirm the appointment details for cancelling your appointment."

}

},

'reply\_appointment\_status': {

'status': False,

'isDataCollection': False,

'function\_call': cancel\_appointment\_request,

'required': ['appointment\_id', 'cancelation\_reason'],

'prompts': {

'request': None,

'response': 'Here is the data for users cancellation request'

}

}

},

"appointment\_scheduling": {

'check\_available\_details':{

'status': False,

'isDataCollection': True,

'function\_call': None,

'required': ['doctor\_name', 'preferred\_date'],

'prompts' : {

'request': 'We have a request for check availability. To process the request we need list of user datas that provided below.',

'response': None

}

},

"check\_slot\_availability": {

"status": False,

'isDataCollection': False,

'function\_call': schedule\_appointment\_check,

'required': ['doctor\_name', 'preferred\_date'],

"prompts": {

'request': None,

"response": "We did the availablity check for the user request and given the data relevant to the request."

}

},

"collect\_appointment\_details": {

"status": False,

'isDataCollection': True,

'function\_call': None,

"required": ["name", "phone\_number", "address", 'visit\_reason'],

"prompts": {

"request": "Please provide the following detail to initiate a appointment booking.",

'response':''

}

},

"confirm\_appointment\_details": {

"status": False,

'isDataCollection': False,

'function\_call': None,

"required": [],

"prompts": {

"request": None,

'response':"Please confirm the appointment details for register your appointment."

}

},

"register\_appointment\_request": {

"status": False,

'isDataCollection': False,

'function\_call': register\_appointment,

"required": ["name", "phone\_number", "address", "doctor\_name", "preferred\_date", 'visit\_reason'],

"prompts": {

"request": None,

"response": "We provided the appointment status for the patient request"

}

}

}

}

## This Storage should be imagined as a short term or long term memory from storage Team

temp\_storage = []

def prompt\_generation(intent, entity, conv\_id, user\_query = None):

try:

entity = json.loads(entity)

except Exception as e:

print(e)

entity = {}

## Checking whether we have that json or it's a fresh conv

previous\_task = [conv[conv\_id] for conv in temp\_storage if (list(conv.keys())[0] == conv\_id and conv.get(conv\_id))]

## If it's not, adding a new entry for this task

if not previous\_task:

## adding sequence based structure for sequence based tasks

if intent in ['appointment\_scheduling', 'cancel\_appointment', 'reschedule\_appointment']:

## Creating a new task json for this conversation

sequence\_list = list(sequence\_json[intent].keys())

sequence\_status = [False for \_ in sequence\_list]

total\_json = {'task': intent,'sequence\_list': sequence\_list, 'sequence\_status': sequence\_status, 'entity': entity}

temp\_storage.append({conv\_id: total\_json})

## If the task is a general task

if intent in ['general\_information']:

## Creating a new task json for this conversation

total\_json = {'task': intent, 'entity': entity}

temp\_storage.append({conv\_id: total\_json})

## Getting the current task for details

current\_task = [conv[conv\_id] for conv in temp\_storage if (list(conv.keys())[0] == conv\_id and conv.get(conv\_id))][0]

print("current\_task", current\_task)

## Getting the intent from that task to define the sequence for that task

intent = current\_task['task']

if intent in ['appointment\_scheduling', 'cancel\_appointment', 'reschedule\_appointment']:

## Updating the entities of the current conversation

current\_task['entity'].update(entity)

## Getting the current sequence of the task

## If all process are completed

try:

current\_sequence = current\_task['sequence\_list'] [current\_task['sequence\_status'].index(False)]

except ValueError as e:

return "prompt: all sequence for the " + current\_task['task'] + " task is complete. Thank you!"

## while there is a data collection sequence that can be skipped based on missed entity run this loop

while sequence\_json[current\_task['task']][current\_sequence]['isDataCollection']:

print("current sequence: ", current\_sequence)

## Check if the current sequence missing any entities

missed\_entities = [enti for enti in sequence\_json[current\_task['task']] [current\_sequence]['required'] if enti not in current\_task['entity']]

print("Missed Entities: ", missed\_entities)

## Check that the sequence have all required params

if not missed\_entities:

## If yes, then set the status of current sequence as True

current\_task['sequence\_status'] [current\_task['sequence\_list'].index(current\_sequence)] = True

## Updating the short term memory about the conversation

## Getting the current conv's index in temp storage to update the conv json

current\_conv = [temp\_storage.index(conv) for conv in temp\_storage if list(conv.keys())[0] == conv\_id][0]

temp\_storage[current\_conv][conv\_id] = current\_task

## Setting the new current sequence

try:

current\_sequence = current\_task['sequence\_list'] [current\_task['sequence\_status'].index(False)]

except ValueError as e:

return "prompt: all sequence for the " + current\_task['task'] + " task is complete. Thank you!"

else:

return "prompt: " + sequence\_json[current\_task['task']][current\_sequence] ['prompts']['request'] + str(missed\_entities) + '\n' + "data: "

## If it is a non data collection task then to the needful

else:

print("current sequence: ", current\_sequence)

## If there is a sequence which doesn't need to call RAG, directly return their prompt here

if not sequence\_json[current\_task['task']][current\_sequence]['function\_call']:

## If yes, then set the status of current sequence as True

current\_task['sequence\_status'] [current\_task['sequence\_list'].index(current\_sequence)] = True

## Updating the short term memory about the conversation

## Getting the current conv's index in temp storage to update the conv json

current\_conv = [temp\_storage.index(conv) for conv in temp\_storage if list(conv.keys())[0] == conv\_id][0]

temp\_storage[current\_conv][conv\_id] = current\_task

return "prompt: " + sequence\_json[current\_task['task']][current\_sequence] ['prompts']['response'] + '\n' + "data: "

## Call the relevant function from the RAG

## Getting the required params for the relevant sequence

required\_params = sequence\_json[current\_task['task']][current\_sequence]['required']

## Creating a payload for the sequence by getting values for the given params

payload = {key: current\_task['entity'][key] for key in required\_params}

## Calling the required RAG function to get the data

data = sequence\_json[current\_task['task']][current\_sequence]['function\_call'] (payload)

## If yes, then set the status of current sequence as True

current\_task['sequence\_status'] [current\_task['sequence\_list'].index(current\_sequence)] = True

## Updating the short term memory about the conversation

## Getting the current conv's index in temp storage to update the conv json

current\_conv = [temp\_storage.index(conv) for conv in temp\_storage if list(conv.keys())[0] == conv\_id][0]

temp\_storage[current\_conv][conv\_id] = current\_task

return "prompt: " + sequence\_json[current\_task['task']][current\_sequence]['prompts'] ['response'] + '\n' + "data: " + str(data)

elif intent in ['general\_information']:

## Updating the entities of the current conversation

current\_task['entity'].update(entity)

## Updating the short term memory about the conversation

## Getting the current conv's index in temp storage to update the conv json

current\_conv = [temp\_storage.index(conv) for conv in temp\_storage if list(conv.keys()) [0] == conv\_id][0]

temp\_storage[current\_conv][conv\_id] = current\_task

## Returning the prompt to the LLM

rag\_data = general\_rag\_calling(user\_query)

return """prompt: Here is the relevant data fetched from the RAG for the given user query. So while generating the response for the user query kindly include the data that i provided along with your response generation.

""" + '\n' + "data: {}".format(rag\_data)

##For testing purpose

while True:

user\_intent = input('Enter a user intent: ')

entity = input('Enter a user entity: ')

conv\_id = (input('Enter a conv\_id: '))

prompt = prompt\_generation(user\_intent, entity, conv\_id)

print(prompt)

if 'Thank you!' in prompt:

break

After the tested the above code for the basic positive workflow, I went straight to Langchain to know more about it and how it can actually help us to dynamically generate prompt and develope the workflow that we are calling as prompt generator.

**Day 8:** 1-July-2024(Monday)

In this day, I started to continue the analysis of how we can use langchain for better of our module so we don’t necessarily hard code everything on our own. In search of the objective, I started my analysis by reading the blog from the langchain website itself, where they taken a basic customer support bot for example. Here you can find the link for the page.

Link: <https://langchain-ai.github.io/langgraph/tutorials/customer-support/customer-support/>

In this link I find something called tools which we generally call functions that acts as a API in common scenarios, and i can able to some multi turn conversations example within it, which gives me the hope that this can be used somewhere inside our workflow to improve our module’s flexibility. I also attached the sample first level conversations for reference

**Sample Conversations:**

================================ **Human Message** =================================

Hi there, what time is my flight?

================================== **Ai Message** ==================================

Hello, to check the time of your flight, I will need to look up your ticket information first. Could you please provide me with your ticket number or booking reference? I'd be happy to retrieve the details of your flight once I have that information.

================================ **Human Message** =================================

Am i allowed to update my flight to something sooner? I want to leave later today.

================================== **Ai Message** ==================================

[{'text': 'Let me check the company policies first on changing flights:', 'type': 'text'}, {'id': 'toolu\_016BZDgoB6cLVCWYGjsHiuFE', 'input': {'query': 'changing flights same day'}, 'name': 'lookup\_policy', 'type': 'tool\_use'}]

Tool Calls:

lookup\_policy (toolu\_016BZDgoB6cLVCWYGjsHiuFE)

Call ID: toolu\_016BZDgoB6cLVCWYGjsHiuFE

Args:

query: changing flights same day

================================= **Tool Message** =================================

Name: lookup\_policy

## Booking and Cancellation

1. How can I change my booking?

\* The ticket number must start with 724 (SWISS ticket no./plate).

\* The ticket was not paid for by barter or voucher (there are exceptions to voucher payments; if the ticket was paid for in full by voucher, then it may be possible to rebook online under certain circumstances. If it is not possible to rebook online because of the payment method, then you will be informed accordingly during the rebooking process).

\* There must be an active flight booking for your ticket. It is not possible to rebook open tickets or tickets without the corresponding flight segments online at the moment.

\* It is currently only possible to rebook outbound (one-way) tickets or return tickets with single flight routes (point-to-point).

2. Which tickets/bookings cannot be rebooked online currently?

\* Bookings containing flight segments with other airlines

\* Bookings containing reservations, where a ticket has not yet been issued

\* Bookings with several valid tickets for the same person and route

\* Tickets with a status other than O (open) (A)

\* Bookings with segments with a status other than OK (e.g. containing flight segments with the status Waitlist) (HK|RR)

\* Tickets that do not display the tariff calculation (IT tickets)

\* Bookings that contain special services (e.g. transportation of animals/transportation of medica ... (truncated)

================================== **Ai Message** ==================================

Based on the policy information, it looks like you are generally allowed to change your flight on the same day for an earlier option, provided your ticket number starts with 724 and some other conditions are met. However, to check if your specific ticket can be updated today, I will need to look up your ticket number or booking reference.

Could you please provide your ticket number or booking reference? Then I can check your flight details and whether you are eligible to change to an earlier flight today per the policy.

================================ **Human Message** =================================

Update my flight to sometime next week then

================================== **Ai Message** ==================================

[{'text': 'Okay, let me check the policy on changing flights to a different date:', 'type': 'text'}, {'id': 'toolu\_012iAxutz45L1QFeTeu3TTRG', 'input': {'query': 'changing flight dates to next week'}, 'name': 'lookup\_policy', 'type': 'tool\_use'}]

Tool Calls:

lookup\_policy (toolu\_012iAxutz45L1QFeTeu3TTRG)

Call ID: toolu\_012iAxutz45L1QFeTeu3TTRG

Args:

query: changing flight dates to next week

================================= **Tool Message** =================================

Name: lookup\_policy

## Booking and Cancellation

1. How can I change my booking?

\* The ticket number must start with 724 (SWISS ticket no./plate).

\* The ticket was not paid for by barter or voucher (there are exceptions to voucher payments; if the ticket was paid for in full by voucher, then it may be possible to rebook online under certain circumstances. If it is not possible to rebook online because of the payment method, then you will be informed accordingly during the rebooking process).

\* There must be an active flight booking for your ticket. It is not possible to rebook open tickets or tickets without the corresponding flight segments online at the moment.

\* It is currently only possible to rebook outbound (one-way) tickets or return tickets with single flight routes (point-to-point).

2. Which tickets/bookings cannot be rebooked online currently?

\* Bookings containing flight segments with other airlines

\* Bookings containing reservations, where a ticket has not yet been issued

\* Bookings with several valid tickets for the same person and route

\* Tickets with a status other than O (open) (A)

\* Bookings with segments with a status other than OK (e.g. containing flight segments with the status Waitlist) (HK|RR)

\* Tickets that do not display the tariff calculation (IT tickets)

\* Bookings that contain special services (e.g. transportation of animals/transportation of medica ... (truncated)

================================== **Ai Message** ==================================

The policy states that you are generally allowed to change your flight and travel dates online, as long as your ticket number starts with 724 and meets the other conditions listed.

To proceed with changing your flight to sometime next week, I'll need your ticket number or booking reference. Once I have that, I can look up your specific reservation details and change your flight dates if permitted based on your fare type and the availability of flights.

Please provide me with your ticket number or booking reference whenever you're ready.

================================ **Human Message** =================================

The next available option is great

================================== **Ai Message** ==================================

[{'text': "Got it, you'd like to change your flight to the next available option sometime next week. Let me first verify your ticket details:", 'type': 'text'}, {'id': 'toolu\_01DCfdGkEsahzxNjBTC2gG1t', 'input': {}, 'name': 'fetch\_user\_flight\_information', 'type': 'tool\_use'}]

Tool Calls:

fetch\_user\_flight\_information (toolu\_01DCfdGkEsahzxNjBTC2gG1t)

Call ID: toolu\_01DCfdGkEsahzxNjBTC2gG1t

Args:

================================= **Tool Message** =================================

Name: fetch\_user\_flight\_information

[{"ticket\_no": "7240005432906569", "book\_ref": "C46E9F", "flight\_id": 19250, "flight\_no": "LX0112", "departure\_airport": "CDG", "arrival\_airport": "BSL", "scheduled\_departure": "2024-04-30 12:09:03.561731-04:00", "scheduled\_arrival": "2024-04-30 13:39:03.561731-04:00", "seat\_no": "18E", "fare\_conditions": "Economy"}]

================================== **Ai Message** ==================================

[{'text': 'Based on your ticket number 7240005432906569, it looks like you currently have a ticket booked for flight LX0112 from Paris (CDG) to Basel (BSL) on April 30th in Economy class.\n\nLet me search for the next available flight option from Paris to Basel after your current flight next week:', 'type': 'text'}, {'id': 'toolu\_01Wfy5PUGvQViroenhAsQpNS', 'input': {'departure\_airport': 'CDG', 'arrival\_airport': 'BSL', 'start\_time': '2024-05-06', 'end\_time': '2024-05-13'}, 'name': 'search\_flights', 'type': 'tool\_use'}]

Tool Calls:

search\_flights (toolu\_01Wfy5PUGvQViroenhAsQpNS)

Call ID: toolu\_01Wfy5PUGvQViroenhAsQpNS

Args:

departure\_airport: CDG

arrival\_airport: BSL

start\_time: 2024-05-06

end\_time: 2024-05-13

================================= **Tool Message** =================================

Name: search\_flights

[{"flight\_id": 19238, "flight\_no": "LX0112", "scheduled\_departure": "2024-05-08 12:09:03.561731-04:00", "scheduled\_arrival": "2024-05-08 13:39:03.561731-04:00", "departure\_airport": "CDG", "arrival\_airport": "BSL", "status": "Scheduled", "aircraft\_code": "SU9", "actual\_departure": null, "actual\_arrival": null}, {"flight\_id": 19242, "flight\_no": "LX0112", "scheduled\_departure": "2024-05-09 12:09:03.561731-04:00", "scheduled\_arrival": "2024-05-09 13:39:03.561731-04:00", "departure\_airport": "CDG", "arrival\_airport": "BSL", "status": "Scheduled", "aircraft\_code": "SU9", "actual\_departure": null, "actual\_arrival": null}, {"flight\_id": 19243, "flight\_no": "LX0112", "scheduled\_departure": "2024-05-11 12:09:03.561731-04:00", "scheduled\_arrival": "2024-05-11 13:39:03.561731-04:00", "departure\_airport": "CDG", "arrival\_airport": "BSL", "status": "Scheduled", "aircraft\_code": "SU9", "actual\_departure": null, "actual\_arrival": null}, {"flight\_id": 19251, "flight\_no": "LX0112", "scheduled\_departure": "2024-05-07 12:09:03.561731-04:00", "scheduled\_arrival": "2024-05-07 13:39:03.561731-04:00", "departure\_airport": "CDG", "arrival\_airport": "BSL", "status": "Scheduled", "aircraft\_code": "SU9", "actual\_departure": null, "actual\_arrival": null}, {"flight\_id": 19252, "flight\_no": "LX0112", "scheduled\_departure": "2024-05-06 12:09:03.561731-04:00", "scheduled\_arrival": "2024-05-06 13:3 ... (truncated)

================================== **Ai Message** ==================================

[{'text': "The next available flight option from Paris to Basel after your current one on April 30th is flight LX0112 on May 6th at 12:09pm. \n\nTo change your ticket to this new flight on May 6th, I'll need to invoke the update\_ticket\_to\_new\_flight tool:", 'type': 'text'}, {'id': 'toolu\_015rWZwtfrqHs94qtM3Zfq7j', 'input': {'ticket\_no': '7240005432906569', 'new\_flight\_id': 19252}, 'name': 'update\_ticket\_to\_new\_flight', 'type': 'tool\_use'}]

Tool Calls:

update\_ticket\_to\_new\_flight (toolu\_015rWZwtfrqHs94qtM3Zfq7j)

Call ID: toolu\_015rWZwtfrqHs94qtM3Zfq7j

Args:

ticket\_no: 7240005432906569

new\_flight\_id: 19252

================================= **Tool Message** =================================

Name: update\_ticket\_to\_new\_flight

Not permitted to reschedule to a flight that is less than 3 hours from the current time. Selected flight is at 2024-05-06 12:09:03.561731-04:00.

As these conversations are kept going, I ignored the complete flow and follows the next instructions given for improving the performance. They are divided the whole blog into three phases which includes basic flow, Conditional interupts, and User Condition. I follow through the end of the tutorial.

After that i spend some with exploring autogen which is also a multi-turn conversation tool. After that I gave a demo of the flow to the TL and explained each and every part of the code. After that collected his response and suggestions to improve the direction of the analysis. And also provided some improvements to make the existing sequence more flexible.

After then we gathered for the meeting to review the current progress of each team and told that i had analysed the sample customer support chatbot usecase given in the documentation and got some inputs from there to use them in our module.

After that i worked with Esakki muthu who is working on NLP team and helped him to solve the issues that he was encountering during the **Tensorflow Sequential** models inference process and make sure the inference process working fine.