



## GSOC 2024 - Proposal for Neurobagel (INCF)

[A natural language interface for querying  
federated research data](#)

### **Mentors :**

Brent McPherson ([bcmcphe@gmail.com](mailto:bcmcphe@gmail.com))

Arman Jahanpour ([armanjahanpour7@gmail.com](mailto:armanjahanpour7@gmail.com))

Sebastian Urchs ([sebastian.urchs@gmail.com](mailto:sebastian.urchs@gmail.com))

Alyssa Dai ([alyssa.ydai@gmail.com](mailto:alyssa.ydai@gmail.com))

# Table of Contents -

## [1. Information about the applicant](#)

## [2. Project Overview and Description](#)

### [2.1 Project Synopsis](#)

### [2.2 Set of Deliverables](#)

## [3. Step-by-step contribution to the project](#)

### [3.1 Detailed description of the stages of the project](#)

#### [3.1.1 Convert the user query \(which is in natural language\) in JSON format.](#)

#### [3.1.2 Make several checks on the JSON object before constructing the API call](#)

#### [3.1.3 Construct the API calls by using the JSON object](#)

#### [3.1.4 Creating a user-interface for the agent](#)

### [3.2 Communication with the mentors](#)

### [3.3 Detailed description of the stages of the project \(Timeline\)](#)

## [4. Candidate Details](#)

### [4.1 Why am I motivated by this project?](#)

### [4.2 Why am I a good candidate for this project?](#)

### [4.3 Past experiences](#)

### [4.4 Availability](#)

# 1. Information about the applicant

- Name : Raya Chakravarty
- Email : [rayachakravarty5@gmail.com](mailto:rayachakravarty5@gmail.com)
- Github : <https://github.com/Raya679>
- Location : Mumbai, India
- University : Veermata Jijabai Technological Institute
- Degree : B.Tech in Computer Engineering
- Resume : [Resume](#)
- Timezone: IST (UTC +5:30)
- Neurostar ID: [@raya](#)

## 2. Project Overview and Description

### 2.1 Project Synopsis

Neurobagel is a federated data ecosystem that allows researchers and other data users to find and consume research data that has to remain at their original institute for data governance reasons. It helps researchers connect a local neuroimaging dataset with others in a decentralized framework using linked data principles.

Currently, the researcher or the data user has to answer a number of queries to get the desired results and it often requires iteration. My aim would be to make this search process more user-friendly by adding a LLM style chatbot interface. This is to be done by utilizing large language models that will be able to interpret the user prompts and initiate the API calls accurately giving the user the desired results.

### 2.2 Set of Deliverables

1. Identify and develop a model that will be able to generate accurate API results from the user prompts.
2. Create a comprehensive suite of tests to validate the functionality and accuracy of the model and API integration.
3. Prepare detailed documentation covering the usage instructions for the developed model and associated functionalities.
4. Developing a simple user interface for the agent

## 3. Step-by-step contribution to the project

### 3.1 Detailed description of the stages of the project

#### 3.1.1 Convert the user query (which is in natural language) to JSON format.

Converting the user query (which is in natural language) to JSON format would make it easier while constructing the API calls. We can use [langchain](#) for this purpose.

Langchain is an open source orchestration framework for the development of applications using large language models (LLMs).

Thus by using langchain and experimenting with several LLMs and prompts, we will be able to convert the user query into JSON object.

An example of this is shown below.

```
from langchain_community.llms import HuggingFacePipeline
from langchain import PromptTemplate, HuggingFaceHub, LLMChain
import os
import json

os.environ['HUGGINGFACEHUB_API_TOKEN'] = <HUGGINGFACE_TOKEN>

def convert_JSON(output):
    pairs = [pair.strip() for pair in output.split(",")]
    json_object = {}

    for pair in pairs:
        key, value = pair.split("=")
        value = value.strip().strip('"')
        json_object[key.strip()] = value.strip()

    fin_output = {
        "min_age": json_object.get("min_age", "null"),
        "max_age": json_object.get("max_age", "null"),
        "sex": json_object.get("sex", "null"),
        "diagnosis": json_object.get("diagnosis", "null"),
        "is_control": json_object.get("is_control", "null"),
        "min_num_imaging_sessions":
    }
    return json_object.get("min_num_imaging_sessions", "null"),
```

```

        "min_num_phenotypic_sessions":
json_object.get("min_num_phenotypic_sessions", "null"),
        "assessment": json_object.get("assessment", "null"),
        "image_modal": json_object.get("image_modal", "null")
    }
    return fin_output

def extract_data(description):
    prompt = PromptTemplate(
        input_variables=["description"],
        template=''
        Please extract the following information from the given text
only and return it in the following JSON format. Do not assume
data:

        min_age: <extracted_min_age>,
        max_age: <extracted_max_age>,
        sex: <sex>,
        diagnosis: <extracted_diagnosis>,
        is_control: <extracted_is_control>,
        min_num_imaging_sessions:
<extracted_min_num_imaging_sessions>,
        min_num_phenotypic_sessions:
<extracted_min_num_phenotypic_sessions>,
        assessment: <extracted_assessment>,
        image_modal: <extracted_image_modal>

        If a specific data is not given then put 'null'.
        Do not assume data unless specifically mentioned by the
user.

        This is the body of text to extract the information from:
{description}
        ''
    )
    chain =
LLMChain(llm=HuggingFaceHub(repo_id='google/flan-t5-xxl',
model_kwargs={'temperature':0.1}), prompt=prompt)
    output = chain.run(description)

    return convert_JSON(output)

description = str(input("Enter query : "))
JSON_output = extract_data(description)

```

```
for key, value in JSON_output.items():
    print(f"{key}: {value}")
```

## OUTPUT:

```
Enter query : Hello I need data for females between ages 2 to 7 using the assessment tool autism specturm quotient
/home/Raya/miniconda3/envs/testingAPI/lib/python3.8/site-packages/langchain_core/_api/deprecation.py:117: LangChainD
community.llms.huggingface_hub.HuggingFaceHub' was deprecated in langchain-community 0.0.21 and will be removed in 6
warn deprecated(
/home/Raya/miniconda3/envs/testingAPI/lib/python3.8/site-packages/langchain_core/_api/deprecation.py:117: LangChainD
s deprecated in LangChain 0.1.0 and will be removed in 0.2.0. Use invoke instead.
warn deprecated(
min_age: 2
max_age: 7
sex: female
diagnosis: null
is_control: null
min_num_imaging_sessions: null
min_num_phenotypic_sessions: null
assessment: autism specturm quotient
image_modal: null
```

Here's an example prompt that has yielded the most effective results thus far when utilizing the [google/Flan-T5-xxl](https://huggingface.co/google/Flan-T5-xxl) model from Hugging Face.

However, the results are not always entirely accurate. This could be due to the model not being trained on a diverse enough dataset, leading to biases in its understanding and extraction of information.

Also we need to employ a function called [convert\\_JSON](#) to transform the outputs of the large language model into JSON objects. This is necessary because the provided large language model is unable to generate complete JSON objects directly. This limitation could stem from the model's training data, which may not cover JSON data extensively. Specifically, the google/Flan-T5-xxl model is not explicitly trained on JSON data, which could contribute to its inability to generate complete JSON objects from user input.

Thus to solve the above issues and to attain maximum accuracy, I'll need to continuously experiment with new prompts and other large language models.

### 3.1.2 Make several checks on the JSON object before constructing the API call

Upon receiving the JSON object, a series of checks are initiated to ensure the accuracy of constructing the API call.

These checks encompass various aspects:

1. Checks included in model.py in the github repository <https://github.com/neurobagel/api/tree/main/app/api> will also be checked and appropriate prompts are generated for the user to rectify any discrepancies.

2. Checks are performed to ensure the data types of each parameter align with the expected format. Inconsistencies are flagged, and users are prompted to input the correct data types as required.
3. An instance for another check would be as follows  
 Eg : I want to query for only female participants."  
 The URL for such a query would be  
`https://api.neurobagel.org/query/?sex=snomed:248152002,`  
 where `snomed:248152002` is a controlled term from the SNOMED CT vocabulary corresponding to the female sex.  
 Here to facilitate a query for exclusively female participants, the model would need to internally correlate the user's request with the relevant controlled term from the SNOMED CT vocabulary and construct the API call. This internal process would ensure seamless interaction for users.
4. Apart from these other necessary checks will also have to be considered.

All validation checks will be conducted internally and abstracted from the user, resulting in an enhanced user experience.

Executing these checks will ensure that the API call construction is accurate and compliant with the specified requirements, mitigating potential errors in the process.

### 3.1.3 Construct the API calls by using the JSON object -

Following the initial validation process, the subsequent step involves transforming the JSON object into a query string format. This transformation enables the generation of the API call URL necessary for retrieving the desired results.

Here is a simple example of how we can do so.

```
import requests
def construct_query_string(JSON_output):
    query_params = {
        "min_age": JSON_output.get("min_age"),
        "max_age": JSON_output.get("max_age"),
        "sex": JSON_output.get("sex"),
        "diagnosis": JSON_output.get("diagnosis"),
        "is_control": JSON_output.get("is_control"),
        "min_num_imaging_sessions":
JSON_output.get("min_num_imaging_sessions"),
        "min_num_phenotypic_sessions":
JSON_output.get("min_num_phenotypic_sessions"),
        "assessment": JSON_output.get("assessment"),
        "image_modal": JSON_output.get("image_modal")
```

```

    }

    query_params = {key: value for key, value in
query_params.items() if value not in (None, 'null')}

    query_string = "&".join([f"{key}={value}" for key, value in
query_params.items()])

    return query_string

description = input("Enter the description: ")
JSON_output = extract_data(description)
query_string = construct_query_string(JSON_output)

url = "https://api.neurobagel.org/query/"
response = requests.get(url, params=query_string)
print("URL: ", url+query_string)

if response.status_code == 200:
    print(response.json())
else:
    print("Error:", response.text)

```

## OUTPUT:

```

(testingAPI) Raya@hp-HP-Laptop-15s-gr0xxx:~/Generative AI$ python GSOC.py
/home/Raya/miniconda3/envs/testingAPI/lib/python3.8/site-packages/langchain/_init_.py:29: UserWarning: Importing PromptTemplate from langchain root module is no longer supported. Please use langchain.prompts.PromptTemplate instead.
  warnings.warn(
/home/Raya/miniconda3/envs/testingAPI/lib/python3.8/site-packages/langchain/_init_.py:29: UserWarning: Importing HuggingFaceHub from langchain root module is no longer supported. Please use langchain_community.llms.HuggingFaceHub instead.
  warnings.warn(
/home/Raya/miniconda3/envs/testingAPI/lib/python3.8/site-packages/langchain/_init_.py:29: UserWarning: Importing LLMChain from langchain root module is no longer supported. Please use langchain.chains.LLMChain instead.
  warnings.warn(
Enter the description: Hello, I need the data between ages 2 to 7.
/home/Raya/miniconda3/envs/testingAPI/lib/python3.8/site-packages/langchain_core/_api/deprecation.py:117: LangChainDeprecationWarning: The class 'langchain_community.llms.huggingface_hub.HuggingFaceHub' was deprecated in langchain-community 0.0.21 and will be removed in 0.2.0. Use HuggingFaceEndpoint instead.
  warn_deprecated(
/home/Raya/miniconda3/envs/testingAPI/lib/python3.8/site-packages/langchain_core/_api/deprecation.py:117: LangChainDeprecationWarning: The function 'run' was deprecated in LangChain 0.1.0 and will be removed in 0.2.0. Use invoke instead.
  warn_deprecated(
URL: https://api.neurobagel.org/query/min_age=2&max_age=7
Results of API
[{'dataset_uid': 'http://neurobagel.org/vocab/0b9277fd-3d72-458f-9340-e468e67e5dc1', 'dataset_name': 'StandardRat', 'dataset_portal_uri': 'https://github.com/OpenNeuroDatasets-JSONLD/ds004116.git', 'dataset_total_subjects': 209, 'records_protected': False, 'num_matching_subjects': 127, 'subject_data': [{'sub_id': 'sub-300102', 'session_id': 'ses-nb01', 'num_sessions': '1', 'age': '2.9E0', 'sex': 'http://purl.bioontology.org/ontology/SNOMEDCT/248152002', 'diagnosis': [None], 'subject_group': None, 'assessment': [None], 'image_modal': ['http://purl.org/nidash/nidm#T2Weighted', 'http://purl.org/nidash/nidm#FlowWeighted'], 'session_file_path': '/ds004116/sub-300102'}, {'sub_id': 'sub-300103', 'session_id': 'ses-nb01', 'num_sessions': '1', 'age': '2.9E0', 'sex': 'http://purl.bioontology.org/ontology/SNOMEDCT/248152002', 'diagnosis': [None], 'subject_group': None, 'assessment': [None], 'image_modal': ['http://purl.org/nidash/nidm#T2Weighted', 'http://purl.org/nidash/nidm#FlowWeighted'], 'session_file_path': '/ds004116/sub-300103'}, {'sub_id': 'sub-300104', 'session_id': 'ses-nb01', 'num_sessions': '1', 'age': '2.9E0', 'sex': 'http://purl.bioontology.org/ontology/SNOMEDCT/248152002', 'diagnosis': [None], 'subject_group': None, 'assessment': [None], 'image_modal': ['http://purl.org/nidash/nidm#T2Weighted', 'http://purl.org/nidash/nidm#FlowWeighted'], 'session_file_path': '/ds004116/sub-300104'}, {'sub_id': 'sub-300105', 'session_id': 'ses-nb01', 'num_sessions': '1', 'age': '3.0E0', 'sex': 'http://purl.bioontology.org/ontology/SNOMEDCT/248152002', 'diagnosis': [None], 'subject_group': None, 'assessment': [None], 'image_modal': ['http://purl.org/nidash/nidm#T2Weighted', 'http://purl.org/nidash/nidm#FlowWeighted'], 'session_file_path': '/ds004116/sub-300105'}, {'sub_id': 'sub-300106', 'session_id': 'ses-nb01', 'num_sessions': '1', 'age': '4.5E0', 'sex': 'http://purl.bioontology.org/ontology/SNOMEDCT/248152002', 'diagnosis': [None], 'subject_group': None, 'assessment': [None], 'image_modal': ['http://purl.org/nidash/nidm#T2Weighted', 'http://purl.org/nidash/nidm#FlowWeighted'], 'session_file_path': '/ds004116/sub-300106'}, {'sub_id': 'sub-300107', 'session_id': 'ses-nb01', 'num_sessions': '1', 'age': '4.5E0', 'sex': 'http://purl.bioontology.org/ontology/SNOMEDCT/248152002', 'diagnosis': [None], 'subject_group': None, 'assessment': [None], 'image_modal': ['http://purl.org/nidash/nidm#T2Weighted', 'http://purl.org/nidash/nidm#FlowWeighted'], 'session_file_path': '/ds004116/sub-300107'}, {'sub_id': 'sub-300108', 'session_id': 'ses-nb01', 'num_sessions': '1', 'age': '3.0E0', 'sex': 'http://purl.bioontology.org/ontology/SNOMEDCT/248153007', 'diagnosis': [None], 'subject_group': None, 'assessment': [None], 'image_modal': ['http://purl.org/nidash/nidm#T2Weighted', 'http://purl.org/nidash/nidm#FlowWeighted'], 'session_file_path': '/ds004116/sub-300108'}],

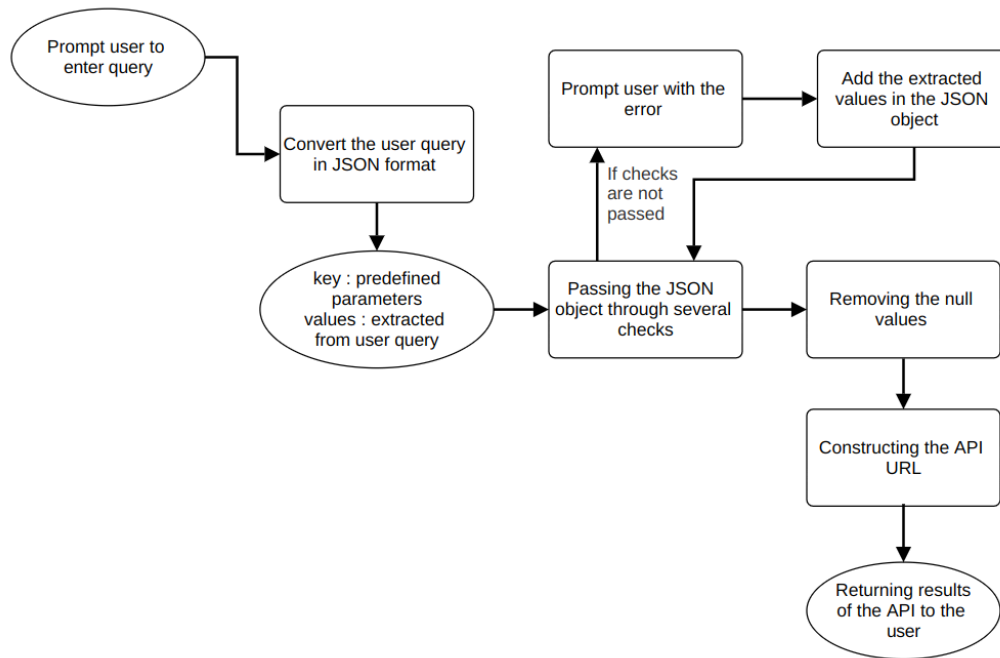
```

The given example demonstrates using 'langchain' to construct the API call and get its results. In this scenario, the results are directly utilized to generate the query string necessary for the API URL without undergoing any validation checks.



During the project, I will delve deeper into this process.

So the **basic workflow** of the project would look like:



This current workflow is prepared for modifications following the feedback provided by the mentors.

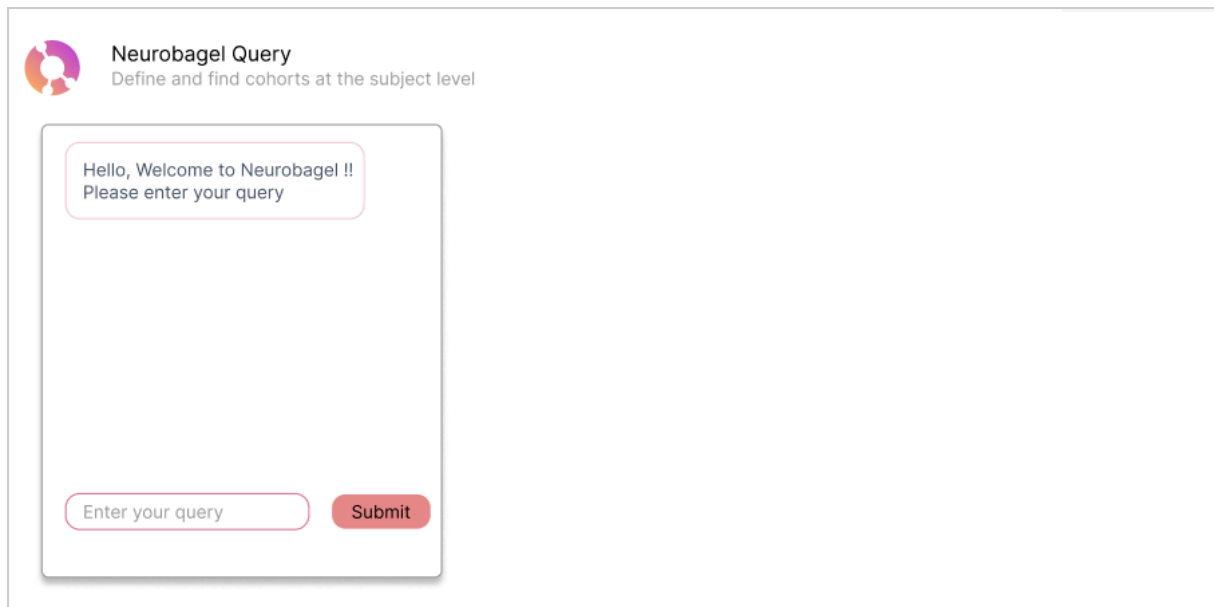
### 3.1.4 Creating a user interface for the agent (for extended timeline) -

This part will have to be first discussed with the mentors.

Currently, Neurobagel is being transitioned from Vue to ReactJS.

Depending upon what is required the user interface will be created in that particular library.

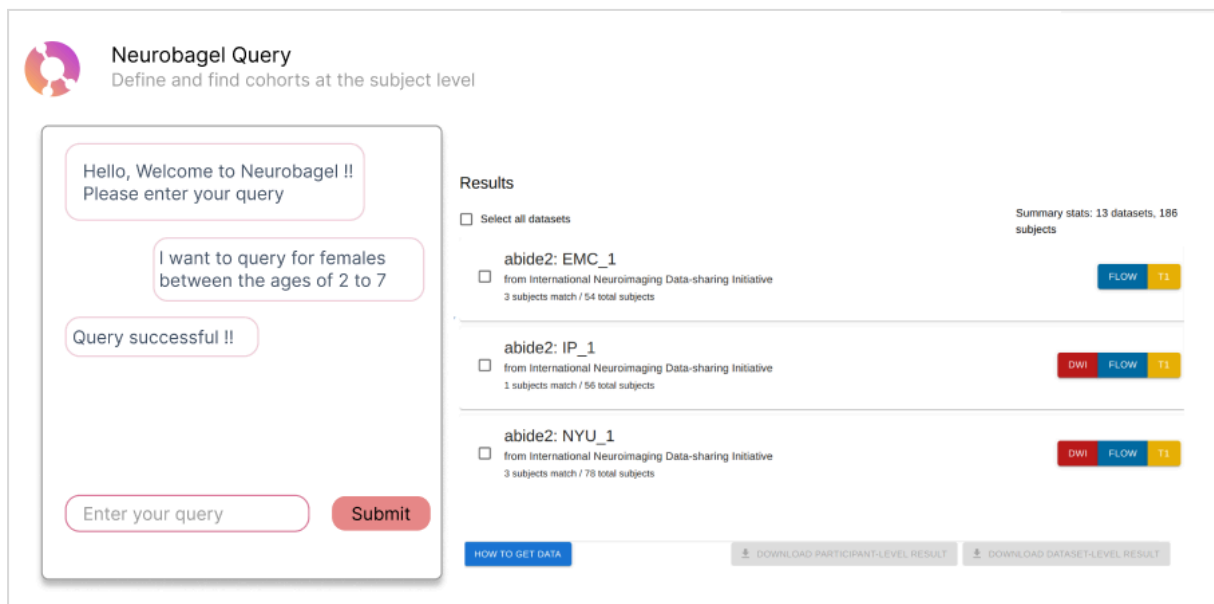
The design (using Figma) for the user interface could be something as follows -



**Neurobagel Query**  
Define and find cohorts at the subject level

Hello, Welcome to Neurobagel !!  
Please enter your query

Enter your query



**Neurobagel Query**  
Define and find cohorts at the subject level

Hello, Welcome to Neurobagel !!  
Please enter your query

I want to query for females between the ages of 2 to 7

Query successful !!

Enter your query

**Results**

☐ Select all datasets

Summary stats: 13 datasets, 186 subjects

<input type="checkbox"/> abide2: EMC_1	from International Neuroimaging Data-sharing Initiative	3 subjects match / 54 total subjects	<input type="button" value="FLOW"/> <input type="button" value="T1"/>
<input type="checkbox"/> abide2: IP_1	from International Neuroimaging Data-sharing Initiative	1 subjects match / 56 total subjects	<input type="button" value="DWI"/> <input type="button" value="FLOW"/> <input type="button" value="T1"/>
<input type="checkbox"/> abide2: NYU_1	from International Neuroimaging Data-sharing Initiative	3 subjects match / 78 total subjects	<input type="button" value="DWI"/> <input type="button" value="FLOW"/> <input type="button" value="T1"/>

[HOW TO GET DATA](#) [DOWNLOAD PARTICIPANT-LEVEL RESULT](#) [DOWNLOAD DATASET-LEVEL RESULT](#)

## 3.2 Communication with the Mentors

1. Maintaining regular communication with my mentors is a priority, and I ensure to provide them with daily updates regarding my progress.
2. I will reach out to them on the platform preferred by them (e-mail, discord or any of their convenience).
3. I would like to have a weekly meeting with the mentors in order to get a better understanding of the correctness of my approach and any improvements they suggest.

### 3.3 Detailed description of the stages of the project (Timeline)

The project timeline is set below to have a clear idea of the project and to be confident that we can keep on track, both from a personal management perspective and also to smoothly coordinate with the mentors and have a successful result.

Date and Time	Event	<ul style="list-style-type: none"> <li><b>Proposed Tasks</b></li> </ul>
	Pre Gsoc	<ul style="list-style-type: none"> <li>Researching about langchain and open source models</li> </ul>
May 1 - May 26	Accepted GSoC projects announced + Community Bonding Period	<ul style="list-style-type: none"> <li>Communicate with peers, learn from them.</li> <li>Study any additional resources, documentation and research about open source models if necessary.</li> </ul>
May 27- June 7	Coding Starts	<ul style="list-style-type: none"> <li>Get familiar with the codebase of existing tools, including the API and cohort query tool</li> </ul>
June 8 - June 14		<ul style="list-style-type: none"> <li>Experimenting with prompts and open source models which will give the maximum accuracy.</li> </ul>
June 15 - June 21	Milestone #1	<ul style="list-style-type: none"> <li>Finalizing the model and prompts which give maximum accuracy.</li> <li>Start with the documentation and write tests for the same.</li> <li>Make improvisations if any, suggested by mentors.</li> </ul>
June 22 - July 8	Milestone #2	<ul style="list-style-type: none"> <li>Applying various validation checks to the results that will further prompt the user if necessary.</li> <li>Document the progress so far and write tests for the same.</li> </ul>
July 8 - July 12	Mid term evaluation	<ul style="list-style-type: none"> <li>Create a draft for mid term evaluation along with the mentors.</li> <li>So far, the model will be finalized and the validation checks will be applied.</li> <li>The documentation and tests will be up to date for the same.</li> </ul>
July 13- July 19		<ul style="list-style-type: none"> <li>Make any changes in the code so far if suggested by the mentors.</li> <li>Solve errors or bugs if any.</li> </ul>

July 19 - July 26	Milestone #3	<ul style="list-style-type: none"> <li>Construct API calls from the results of the LLMs which returns the desirable results.</li> </ul>
July 27 - Aug 2		<ul style="list-style-type: none"> <li>See to it that the API calls return desirable results and solve any errors or bugs if present.</li> </ul>
Aug 2 - Aug 9		<ul style="list-style-type: none"> <li>Complete with the documentation of the project and writing of all the remaining tests.</li> <li>Finish with a user-friendly, easy-to-comprehend readme.</li> </ul>
Aug 10 - Aug 19	Buffer time	<ul style="list-style-type: none"> <li>Complete any previously mentioned tasks in case of a backlog</li> <li>Discuss and work on the final project report.</li> </ul>
Aug 19 - Aug 26	Project Submission	<ul style="list-style-type: none"> <li>Make improvisations if any, suggested by mentors.</li> <li>Submit the project.</li> </ul>
Aug 26 - Nov 4	For extended timelines Milestone #4	<ul style="list-style-type: none"> <li>Create a user interface for the agent.</li> <li>Discuss with the mentors what is needed and immediately start working on the same.</li> <li>Complete with all documentation and tests.</li> </ul>

## 4. Candidate Details

### 4.1 Why am I motivated by this project?

I have been interested in exploring and learning more about Large language models and how they work. I have realized that large language models are extremely powerful and have a number of use cases. This project presents an ideal opportunity for me to further explore and engage with large language models within one such specific use case.

Moreover, I'm attracted to the open-source community for its inclusive and supportive atmosphere, and I am eager to contribute to this community through my involvement in this project.

## 4.2 Why am I a good candidate for this project?

- I am fluent in python and have previous experience with Langchain. If necessary research is required to be done for Ollama I assure you that I will do that diligently.
- I am a quick learner and since I have already worked with Javascript I will be swift to learn Typescript if necessary.
- I am disciplined and always consistent and hence, will be able to start the project and finish it before its deadline.

## 4.3 Past experiences

I am a second-year (sophomore) undergraduate student at VJTI, Mumbai, pursuing a Bachelor of Technology degree in Computer Science. I have worked in the fields of machine learning and web development. I'm fluent in Python and C++.

I am an active member of the Community of Coders, a coding club in our university. We hold Machine Learning workshops and seminars.

Here are some of my previous projects:

### 1. [Healthcare Chatbot](#):

- Developed a chatbot that can offer assistance in various aspects of healthcare symptom diagnosis, and more as the already existing AI-based chat models are not trained specifically for health-related tasks.
- **Fine-tuned open-source LLMs** like Llama-2-chat-hf and Falcon-7b on Healthcare-specific dataset for higher accuracy.
- **Implemented Document-based LLM** on dolly-v2-3b using langchain which gives information to the user query in natural language by extracting information from a medical-specific book/document/pdf.

### 2. [Mental Health Wellness Journal](#):

- We developed a Website to help people improve their mental well-being and refresh their minds when they are feeling down as part of our hackathon
- **My contribution** to this project was
  - To **develop the frontend** of the website using **ReactJS**.
  - Developing a **chatbot using langchain** that helps answer user queries.
- We were the **finalists** in this hackathon.

## 4.4 Availability

- Once coding begins, I will work dedicatedly for at least 4-5 hours on weekdays (and 6-8 on weekends) until the completion of the project.
- My summer vacation will be from June 7 to Aug 5, so I will be working full-time (6-8 hours) during this time.
- After the submission of my proposal (on or before April 2), I will start researching langchain and open source Large language models.
- Initially, when the coding period begins I will be having my university examinations. But after completion of my exams, I promise I will compensate and put in more hours if necessary.
- I acknowledge that my timezone (GMT+5:30) may not perfectly align with those of the mentors assigned to this project. To address this, I am willing to adjust my schedule accordingly to ensure regular meetings with the mentors at times convenient for them. In my timezone, I will be available for communications from 8 AM to 1 AM, and I can extend my availability if necessary.
- Additionally, I will be maintaining a blog throughout my GSOC journey, documenting everything I have worked on since the very beginning.