Project Proposal for Google Summer of Code 2025

AI Chatbot & Assistant for Support

Mentor : Dr Iraklis Giannakis

School of Geosciences, University of Aberdeen

Lakshmi Srikumar

April 8 2025

I .Basic Details

- **Full Name** : Lakshmi Srikumar
- **Location & Time Zone :** India , GMT+5:30
- **Education :** Bachelors in Technology in Computer Science Engineering (Pursuing)
- **Email :** lakshmisrikumar04@gmail.com
- **Github Username :** Lakshmi Srikumar
- **LinkedIn Username :** Lakshmi Srikumar
- **Resume :** Link

- ## Biography statement :

“The people who are crazy enough to think they can change the world are the ones who do.” This quote by Steve Jobs has always inspired me. I always want to think in a different perspective and look at things in a more creative and innovative manner. I think it was the reason behind my urge to follow computer science and pursue my dream of doing something new and unique.

Currently I am a final year undergraduate in Computer science and Engineering of Medi-Caps University . My course work along with self study plus my curiosity and enthusiasm about computer science have made me become a skilled programmer capable of using Python , C++ & JavaScript. I hade done my previous internship at SAC -ISRO, India where I worked on optimizing location based algorithm (Kalman Filter) to improve computational efficiency and stability in navigation systems. My experience in coding is about 2 years and I have done several projects which I have published in my Github. As a competitive programmer I seek opportunities to polish my skills and chase my dreams in order to reach my targets. So I suppose the experience with GSOC will be really interesting and useful for me.

II. Motivation and Experience

1. Describe your motivation for participating in Google Summer of Code?

My motivation to participate in Google Summer of Code (GSoC) stems from my deep admiration for the open-source community and the spirit of collaboration and learning that defines it. My journey into open source began with curiosity and a desire to give back. I started by contributing to beginner-friendly open-source events like Hacktoberfest, GirlScript Summer of Code (GSSoC), and other community-led programs . These programs motivated me to apply for Google Summer of Code as an opportunity to work on something impactful over a period of time. I want to challenge myself by taking on a meaningful technical project that requires both creativity and discipline. What excites me the most about GSoC is the opportunity to work under the guidance of experienced mentors on impactful projects GSoC would allow me to go beyond surface-level contributions and actually help shape the direction of a real-world open-source tool. These lessons are invaluable—not just for my career, but for my growth as a responsible and collaborative person.

2. Have you participated in Google Summer of Code in the past?

First-time applicant but done contribution in other open source such as Hacktoberfest , Girl Summer Script of Code (GSSOC).

3. Why did you choose gprMax?

During my internship, while I was optimizing the algorithm , I had a discussion with my mentor about the various disturbances and unknown variable that must be taken into account while predicting locations and for briefly we discussed about the electromagnetic radiations and how they impact the receivers . This experience sparked a deeper interest in how electromagnetic waves are used in real-world applications like imaging and detection. While researching gprMax, I discovered that it specializes in simulating electromagnetic wave propagation, This instantly caught my attention, as it connected directly with the discussions I had during my internship.

4. Why did you select this project idea?

When I came across gprMax's proposed GSoC project—which seeks to replace commercial LLMs like ChatGPT with free, open-source alternatives such as **LLaMA** or **DeepSeek**—I immediately saw it more than a simple chat bot but as a tool .

I chose this project because it aligns with both my technical interests and my passion for open-source innovation. What drew me in was gprMax's blend of scientific modeling and AI-driven accessibility. It's not every day that you come across a project at the intersection of **computational electromagnetics** and **conversational AI**, especially one that has such a strong vision for making complex tools more approachable.

The idea of fine-tuning LLMs specifically for gprMax is exciting because it's not just about answering questions—it's about **to building custom models for specialized instructions**.

5. What are your expectations from us during and after successful completion of the program?

My expectation during the program would be :

1. Technical guidance about the code base.
2. Explaining the domain knowledge of the datasets / project we will be working on.
3. Learn more about the code review and deployment best practices that will be needed.

My expectation after the program would be :

1. Know how to work using the best practices possible.
2. To have expertise in CUDA kernel optimization techniques.
3. Technical Integration Support & Code Quality Assurance.
4. Guidance on writing migration scripts for backward compatibility.

6. What are you hoping to learn?

- Effective integration of AI models with scientific simulation tools like gprMax
- Real-world deployment of open-source LLMs in engineering workflows
- Collaborative development strategies for open-source scientific AI tools

7. How much time will you be able to devote to the project? Are you doing any other internship this summer?

I am currently doing an internship that will end by the end of April . After that I can devote 35 hrs/week .

8. What kind of projects have you worked in the past? What technologies did you use?

I have worked on mostly machine learning projects most notable are the Sign Language Detection system , Gym Helper (Pose detection system) etc. The technologies that I have used are python , open-cv , RAG, Langchain , Tensorflow , Keras .

9. What is your experience with Python, C, CUDA?

- Python: 2 years (Intermediate)
- C / C++: 2 years (Intermediate)
- CUDA: Beginner (Just started learning for this project)

III. Project Information

Since this project is based on an existing implementation contributed during a previous year of Google Summer of Code (GSoC), it already has a foundational structure in place. The current setup includes:

ChromaDB for Vector Storage: The project uses ChromaDB to manage vector databases efficiently.

GPT-4.0 as the LLM and Embedding Function: It leverages ChatGPT 4.0 both for generating responses and for creating vector embeddings.

A Basic Chatbot Interface: A simple chatbot interface was developed to respond to user queries based on the data stored in the vector database.

However, with the rapid evolution in the field of Artificial Intelligence, several efficient and cost-effective open-source alternatives to GPT-4.0 have emerged—most notably DeepSeek and LLaMA. These models offer competitive performance at significantly lower infrastructure costs.

Proposed Direction for GSoC 2025:

The focus of the project moving forward will be twofold:

Integration of Open-Source LLMs:
Implement models such as DeepSeek and LLaMA in place of proprietary solutions like ChatGPT. This not only reduces dependency on paid APIs but also offers more flexibility for experimentation and fine-tuning.

Custom Model Creation Based on User Instructions:
Design and develop a pipeline that enables the creation of lightweight, fine-tuned models tailored to specific domains or tasks as defined by user instructions. This includes stages like data preparation, prompt engineering, domain adaptation, and training with PEFT (Parameter-Efficient Fine-Tuning) techniques.

Part 1 : Implementing free open source LLMs

A) Why we are choosing Deep Seek & Llama ?

Criteria	Deep Seek r1 (1.5B)	Llama 3.2 (3B)	ChatGPT (Large-scale)
Parameter Count	1.5 billion	3 billion	Tens to hundreds of billions
Inference Speed & Efficiency	Very fast due to fewer parameters, lower latency, and reduced resource consumption	Faster than very large models; provides a good balance between speed and complexity	Generally slower; requires more computational resources and has higher latency
Target Applications	Domain-specific and real-time retrieval tasks	More nuanced conversation and moderate complexity tasks	Open-domain, creative, and highly contextual conversation tasks
Customization and Fine-Tuning	Excellent for targeted fine-tuning and deploying on edge/embedded devices	Offers strong language understanding while remaining customizable	Less flexible; fine-tuning is more involved and may require proprietary access or extensive effort
Deployment Footprint	Small footprint – ideal for resource-constrained environments	Moderate footprint; balances resource demands and performance	Large footprint – typically deployed via dedicated, high-resource infrastructures
Cost Considerations	Lower cost in terms of computation and deployment	Moderate costs; reduced inference time compared to ChatGPT without sacrificing too much quality	Higher cost due to extensive hardware requirements and energy consumption for inference

B) Functionalities that can be added :

1. I've successfully downloaded the models locally onto my machine using Ollama. However, I'm now considering the scenario where we need to work with a model that contains a significantly higher number of parameters than what my current system can handle. In such cases, one potential solution is to integrate OpenRouter with LangChain.

Integrating OpenRouter with LangChain empowers you to bypass local hardware constraints and access higher-parameter models hosted remotely. This setup offers you:

- A unified interface that abstracts away the complexity of routing API calls.
- Flexibility to switch between models based on the task at hand.
- Scalability in applications where both lightweight and heavy-duty models are necessary.

2. While building the database , I found out that existing project uses the pure vector search retrieval and not a hybrid method

Basis	**Pure Vector Search (Current Setup)**	**Hybrid Search (Vector + Keyword)**
Pros	a. Semantic Understanding b. Speed: Optimized for large-scale similarity searches. c. Simplicity: Easier to implement and maintain.	a. Comprehensive: Combines semantic understanding (vector) + exact keyword matching (BM25). b.Higher Recall: Better at handling diverse queries c.Robustness: Works well for rare terms or domain-specific jargon.

Cons	a.Keyword Blindness: Misses exact keyword matches b.Cold Start: Requires high-quality embeddings and sufficient training data.	a.Complexity: Requires maintaining two retrieval systems. b. Latency: Slightly slower due to dual retrieval and reranking
Best for	Semantic queries (e.g., chatbots, recommendation systems). Projects prioritizing simplicity and speed.	Enterprise search, legal/document lookup, or queries mixing keywords + context. Projects needing high precision/recall for varied user inputs.

3 . I've been evaluating the embedding functions of both DeepSeek and Llama to determine their effectiveness in handling reasoning tasks. Based on my observations, DeepSeek's embedding function delivers more robust reasoning capabilities compared to Llama's. As a result, I'm currently concentrating on further developing and integrating DeepSeek's embedding function into our system.

4 . Is chat history needed for chat bot ? Chat history can be stored in a database like MongoDB, PostgreSQL etc.It fetches only necessary history (like the last N messages) and it scales better and avoids high RAM usage.

5. I followed the installation guide closely when setting up the project using Docker. Since I'm currently running the setup on Windows, I would like to know which operating system is ideally supported by this project. Should I switch to Linux or continue in Windows?

Part 2 : Developing a custom model

1. **Packages that needs to be installed :**

   a. **AutoTokenizer**: Automatically selects the appropriate tokenizer for a given pretrained model

   b. **AutoModelForCausalLM**: Handles causal language models (text generation)

   c. **BitsAndBytesConfig**: Configuration for 4-bit quantization parameters

   d. **peft** : Parameter-Efficient Fine-Tuning (LoRA implementation) and more.

2. Data collection :

We will be using the dataset mentioned. We will split the dataset into 80% train and 20 % validation .

3. Prompt Design:

Since the dataset contains three fields – Input , Instruction & Output – we will format the prompt in the following :

```
Prompt  = (
   f"User Instruction: {example['instruction']}  \n"
   f"Input Data: {example['input']}   \n"
   f"Assistant Response: {example['output']}"
)
```

We will apply the above format for training & validation dataset.

4. Model & Tokenizer Setup :

Quantization Configuration – We need to do quantization for

a .**Memory Efficiency (The Core Motivation)**

Problem: A 1.5B parameter model in FP32 needs ~6GB memory (2 bytes/param * 3B params)

4-bit Solution: Reduces memory usage to ~0.75GB (4 bits/param * 1.5B params / 8 bits/byte)

```
load_in_4bit=True              # Enables 4-bit storage
bnb_4bit_quant_type="nf4"      # Special 4-bit format that maintains information density
```

b. **Hardware Accessibility**

- **Before Quantization**: Requires high-end GPUs (e.g., A100 40GB)
- **After Quantization**: Runs on consumer GPUs (RTX 3060 12GB) or even CPUs

after this we will set up the tokenizer and load the model . The code snippet for this will be as follows :

```
bnb_config = BitsAndBytesConfig(
load_in_4bit=True,
bnb_4bit_quant_type="nf4",
bnb_4bit_compute_dtype=torch.bfloat16
)

# Load tokenizer
tokenizer = AutoTokenizer.from_pretrained("deepseek-ai/deepseek-r1-1.5b-base")

# Load quantized model
model = AutoModelForCausalLM.from_pretrained   (
            "deepseek-ai/deepseek-r1-1.5b-base",
            quantization_config=bnb_config,
            device_map="auto"  )
```

5. Domain Adaptation (Technical terms)

We will be preparing a list of technical terms that will be searched from the input .

```
technical_terms = ["PML", "Hertzian_Dipole", "soil_peplinski", "gaussianprime",…….]
tokenizer.add_tokens(technical_terms, special_tokens=True)
model.resize_token_embeddings(len(tokenizer))
```

6. Parameter-Efficient Fine-Tuning (LoRA configuration)

LoRA is a parameter-efficient fine-tuning (PEFT) technique that modifies only **0.1-2% of model parameters** instead of retraining the entire model. It works by:

1. **Freezing** the original model weights
2. **Injecting** trainable low-rank matrices into specific layers
3. **Updating** only these small matrices during training

7. Training Configuration :

A code snippet for it -

```
training_args = TrainingArguments  (
output_dir= "./deepseek-gprmax",
train_epochs=5,
train_batch_size=4,
gradient_steps=2,
learning_rate=3e-5,
fp16=True,  # Mixed precision training
evaluation_strategy="steps",
eval_steps=200,
save_strategy="epoch"  )
```

8 . Training Setup with SFT Trainer

Supervised Fine-Tuning Trainer (SFTTrainer) is a specialized class from Hugging Face's trl library designed for efficient LLM fine-tuning. Key features:

1. **Memory Optimization**
    - Implements gradient checkpointing
    - Supports parameter-efficient methods (LoRA, quantization)
2. **Integration**
    - Built-in support for PEFT configurations
    - Automatic dataset formatting.

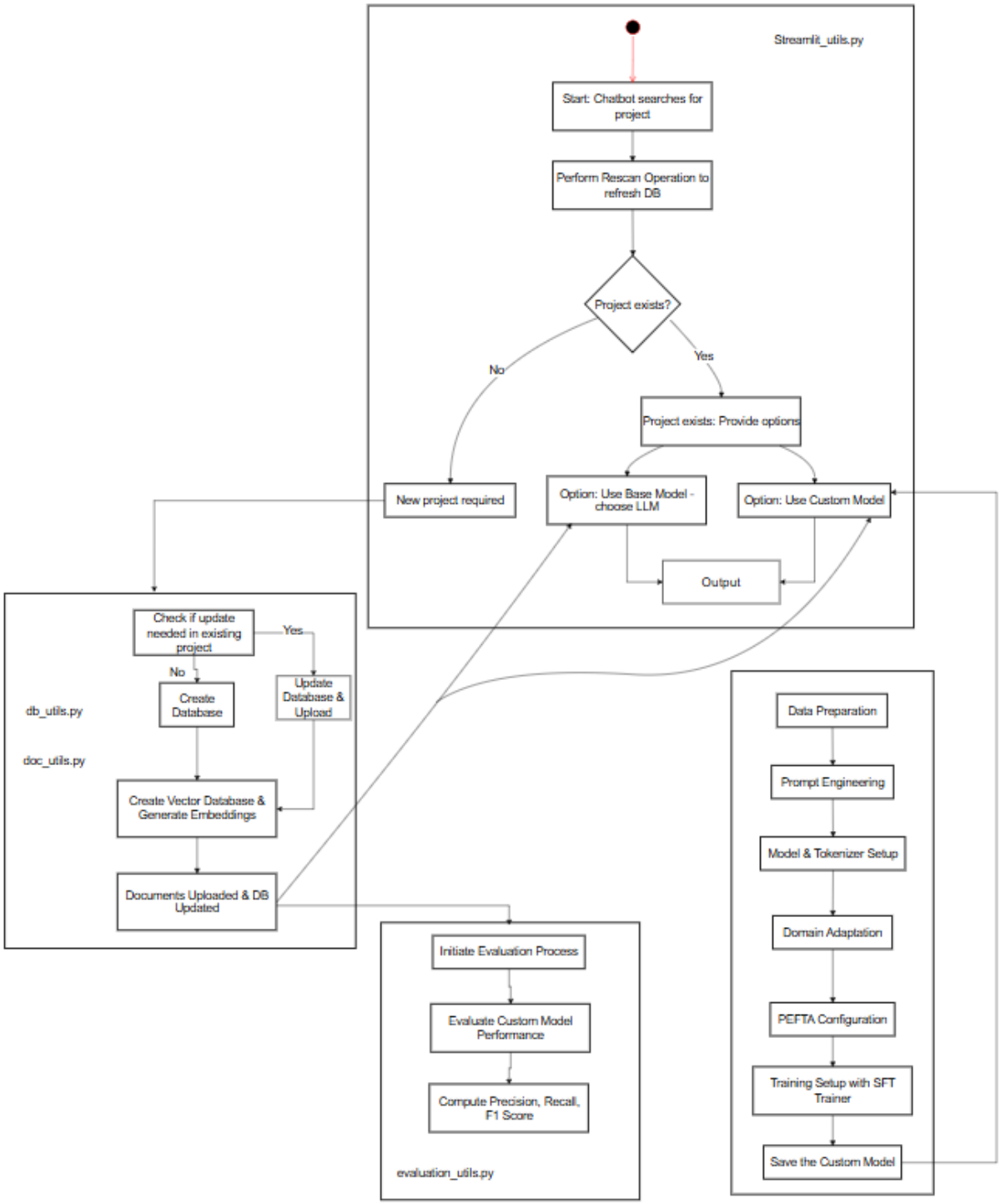
Below will be the code snippet :

```
trainer = SFTTrainer(
    model,
    args=training_args,
    train_dataset=dataset,
    formatting_func=formatting_prompts_func,
)
trainer.train()
```

9. Testing & Saving the model

We will save the model and validate on different dataset sizes to check for it's accuracy , precision & recall.

Work flow :



[Flow chart for the program]

Timeline :

Date	Phase	Key Activities
May 8 - June 1	**Community Bonding**	- Understand the existing project code base . - Learn the pre requisites needed for the code base (Like CUDA , Transformers , pipelines etc.) - Implementing an LLM model
June 2 - July 14	**Core Development**	- Initial Custom Model for small dataset -Testing validation & increasing the dataset size for the model -Finalizing the documentation for midterm evaluation
July 14 - August 25	**Optimization & Integration**	- Submission of Midterm Evaluation - Testing & Validation of Custom Model for larger dataset. - Optimizing the entire project - Dockerization - Documentation of Final Evaluation
August 25 - September 1	**Finalization**	- Finalize documentation - Submission of final project

Closing Remarks :

Q . How often and by what means will you communicate with the mentors?

I will be able to communicate with my mentors for a minimum of 3-4 days per week and am open to communicate via Zoom , Zulip , GMeet or any other medium.

Q . What might be the next steps after completion of this project ?

- Once the project is fully developed and tested, the next step will be to deploy it to a web server. This might involve choosing a cloud hosting provider such as AWS, Azure, or Google Cloud. Ensure that the hosting environment supports necessary dependencies and can scale according to user traffic.
- Implement a CI/CD pipeline so that any new changes can be automatically tested and deployed. This pipeline minimizes downtime and ensures reliable, iterative updates to the website.