Project Proposal

Creating Lesson Plans for Music Blocks using AI

[MusicBlocks](#) | [Divyansh Pandey](#) | [GitHub](#)

Google Summer of Code

Contents

1 Synopsis

1.1 Short Introduction

I am a second-year Computer Science student at the International Institute of Information Technology, Hyderabad. My passion lies in software development and enhancement, which I have pursued actively through various projects. My experience spans a diverse range, from website design to OS-based projects, as well as Machine Learning and Natural Language Processing (NLP) endeavors.

1.2 Why I chose this project

I've been captivated by the concept of Music Blocks and have been an active member of the contributing team for the past two months. During this time, I've been deeply involved in discussing issues, engaging in feature enhancement requests, and independently identifying and resolving bugs. Additionally, as an Undergraduate Researcher at my college, I've delved into Language Model (LLM) research, further enriching my experience in this domain.

This project presents an ideal convergence of my interests, combining my passion for Music Blocks with my expertise in LLMs. I eagerly anticipate the opportunity to collaborate with my mentors and fellow contributors, contributing meaningfully to its development.

2 Benefits to the Community

Music Blocks offers an engaging and interactive approach to working with music, making it accessible and enjoyable for users across various age groups. With its diverse user base, spanning different demographics, Music Blocks holds immense potential for educators, particularly those teaching children.

The integration of Language Model (LLM) technology presents an exciting opportunity to enhance the user experience within Music Blocks, particularly in the creation of lesson plans. Leveraging LLMs creatively can streamline the process of lesson plan development, empowering educators to craft engaging and effective teaching materials tailored to the unique needs and preferences of their students.

By harnessing the power of LLMs in conjunction with Music Blocks, we can not only simplify the task of lesson planning but also enrich the overall educational experience, fostering creativity and innovation in music education.

3 Deliverables

3.1 Automation of generation of JSON data through scripts

The project HTML code will undergo web scraping to extract relevant information, which will then be translated into JSON format for preprocessing. This crucial step lays the foundation for subsequent stages, including training, evaluation, and utilization of the Language Model (LLM). By standardizing the data in JSON format, we ensure consistency and accessibility across all project deliverables. This preliminary process is fundamental and serves as the backbone of our workflow, facilitating seamless integration and efficient utilization of resources at every stage.

3.2 UI option to generate lesson plan for current project

A button in the Planet page will be used to generate the first draft of a lesson plan for the current project. It will open up a chat window where the data from current project will be exported to , to provide context

to the model.

3.3 Fine-Tuned Model for generating responses

A fine-tuned LLM will be deployed to generate lesson plans. Data will be fed to the LLM so that it can be fine-tuned to increase it's knowledge base related to Music Blocks.We can use two types of data for the same :

- Music Blocks documentation file to improve the knowledge base of the LLM on Music Blocks functionalities.
- The sample projects with corresponding example lesson plans to "instruct" the LLM to generate lesson plans properly.

3.4 Conversation with model to refine the generated plans

The user will be able to view generated lesson plans and suggest improvements to the model which will be used to improve upon the displayed lesson plans. This can be done until the user is satisfied with the quality of the plan generated , after which he can download the generated plan as a .PDF/.DOCX file. The user interaction is crucial to ensure accuracte generation of lesson plans and allow user to customize his workflow using AI.

3.5 Testing the performance of the model

The deployed model on the Music Blocks website will generate lesson plans through AI. These plans will undergo thorough review and scrutiny to assess the AI's efficacy in addressing the given problem. The primary objective of the lesson plans is to encapsulate all learning outcomes achievable through Music Blocks' functionalities and their combinations. By doing so, we aim to impart music concepts to users in a friendly and personalized manner. Our expectation is that the lesson plans will offer comprehensive guidance, effectively leveraging Music Blocks to facilitate an engaging and enriching learning experience. We will also have to ensure that the model being deployed follows safety standards , ie does not indulge in hate speech , sexually explicit content and controversial issues so that it can be used correctly by the community.

3.6 Deploying the Model

The model, after satisfactory performance and repeated evaluation will be deployed on the Music Blocks server, bundled as an API which can be called to generate responses from the front-end drivers. The deployment process depends on the availability of computational resources and also affects the selection of the model.

4 Project Breakdown

4.1 Model selection

The selection of an appropriate Language Model (LLM) is paramount for the success of this project. Several factors will influence this decision, including the organization's budget, the specifications of available resources for fine-tuning the LLM, standard performance metrics of LLMs, scalability, and model availability.

To ensure an informed decision, I will conduct a thorough literature review on potential LLMs suitable for the project scope. Following this, I will engage in discussions with my mentors to gather insights and perspectives. Subsequently, I will report my observations and analysis to the team, facilitating a collaborative decision-making process. By reaching a consensus, we will identify the most suitable LLM that aligns with our project objectives and resources.

Some options include Llama-7b(Meta) , Mistral-7b(Mistral AI) , Gemma-7b(Google), these models can be accessed through HuggingFace and are easy to setup and use with good documentation. Using 7b parameters LLM would be optimal since they have medium level system requirements (in terms of memory and computational resources). Using models having more than 7 Billion parameters can increase training and memory complexity. Using models having lesser than 7 Billion parameters can affect the accuracy of the model.

4.2 Generating tuning dataset and standardizing it

Taking inspiration from the example lesson plans provided, a dataset to fine-tune the model and evaluate it will be generated in a standardized form which can be fed to the LLM for it to analyze and generate lesson plans along with Music Blocks code. The preprocessing and standardizing the data will serve as an extremely crucial step to achieve accurate results from the model.

4.2.1 Documentation from Music Blocks codebase

For increasing the knowledge base of the LLM on Music Blocks features and functionalities, the documentation from Music Blocks codebase will be used after proper preprocessing and cleaning. This will be used to create an RAG driven vector embeddings database (ie , an indexed knowledge base about Music Blocks data)

4.2.2 Example lesson plans for user projects

Sample projects provided in the GSOC ideas list can be utilized by standardizing them into proper input-output pairs. This data will be used to fine-tune the model to understand how to respond to responses specifically to generating lesson plans for the user.

4.3 Setting up the fine-tuning pipeline

We feed the data to the LLM after preprocessing the data. For preprocessing the data, we will have to follow certain standard steps and certain steps relevant to our use case.

4.3.1 Custom Preprocessing steps

For the lesson plan examples,We do not directly feed these stream of tokens to the LLM. We create meaningful prompt-response pairs or similar input-output pairs to create a base for Instruction based Fine-Tuning approach to our problem. In the context of the project's use case, we can pass the standardized JSON data from Music Blocks with proper prompt templates as the input and the generated lesson plan as the output so that the LLM can understand the instructions and adjust pre-computed weights inside it's architecture to achieve better results

For the documentation files, we can use svgReaders and tokenizers to pass the data in the form of fixed size tokens to the model which can be used for improving it's knowledge base using Vector databases to store the embeddings. We can use an RAG-driven approach(Retrieval Augmented Generation) to index

the knowledge gained from the documentation and use it to enhance it's response about topics related to MusicBlocks. .

4.3.2 Proposal of usage of Langchain

Langchain is an open-source framework designed for developing applications using Language Models (LLMs). One of Langchain's key features is its intuitive pipeline for fine-tuning LLMs, offering convenient API calls to augment various stages of the process. This streamlined approach not only enhances efficiency but also minimizes errors.

In our project, we have the option to leverage Langchain for our pipeline. Its user-friendly interface and built-in functionalities can significantly expedite the fine-tuning process. However, should the project's requirements surpass the capabilities of Langchain's services, we are prepared to manually construct each step of the pipeline. This ensures flexibility and enables us to tailor the workflow precisely to our project's needs.

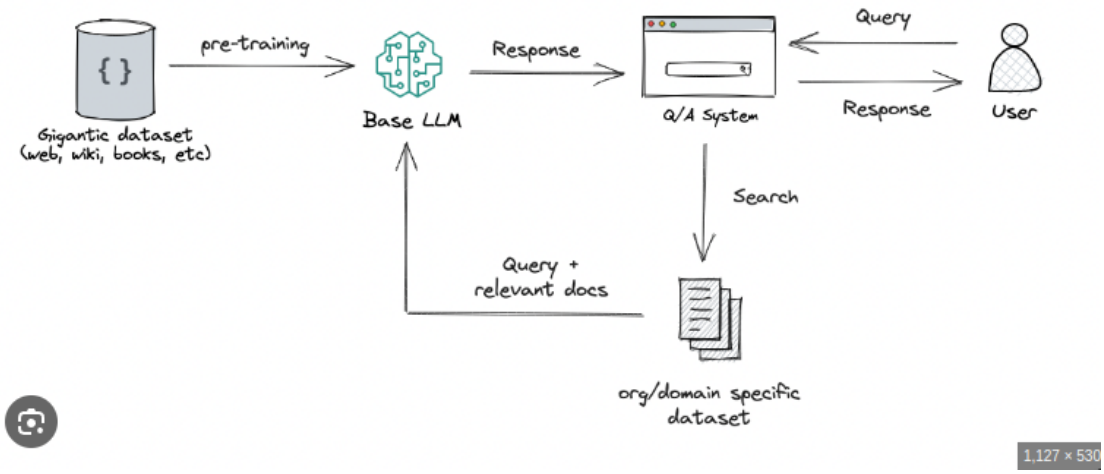

Figure 4.1: Fine-Tuning flow diagram using Langchain

4.4 Fine-Tuning stage

This is the critical phase of the backend-development stage. Fine-Tuning an LLM is time-consuming and is subjected to availability of computational resources. We will use Instruction Fine-Tuning approach (a form of supervised fine-tuning) to teach the LLM to "think" in the right direction. Example prompts with expected outputs will be passed to the LLM to improve it's responses related to the given prompts. We can use Lora or QLora based fine-tuning approaches with Parameter Efficient Fine-Tuning. Langchain provides convenient APIs to tackle the same. This step is crucial to train the model specifically for the given task.

4.4.1 Instruction fine-tuning

We will pass input generated from the JSON files to the LLM and output a sample lesson plan corresponding to that data. These input-output pairs will serve as instructions to fine-tune the responses of the LLM. We will now train the model using this dataset. Instruction fine-tuning essentially allows us to modify how an LLM "thinks" about prompts related to a specific knowledge base. We use labelled data in the form of example prompts and example completions to guide the LLM to more accurate responses. Our training dataset is very important in this step.

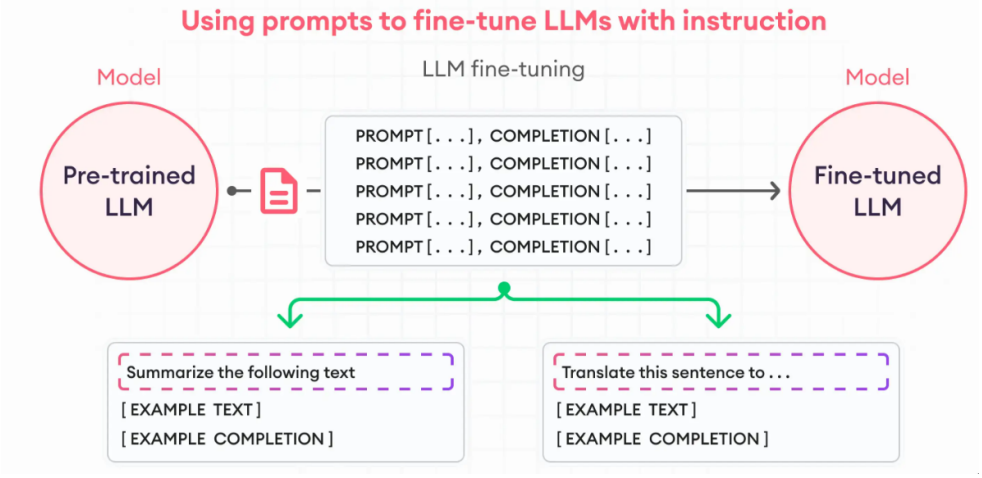

Figure 4.2: Instruction Fine-Tuning

4.4.2 Prompt Engineering for training and testing

Example prompts for training LLM for Music Blocks could have the following structure :

> **You are a helpful assistant to be used only for generating and refining Music Blocks lesson plans.Keeping in mind the above,Generate a lesson plan for the following Music Blocks data with given data.**
> **Data** : {JSON data.}
> **Instructions** : The lesson plan should include basic details about the lesson such as number of students, duration, objectives followed by step-by-step explanation of the lesson along with the concepts involved in the given format:
> Introduction, a Part for each topic, Materials, Performance/Critique, Key events, and Assessments.

This prompt teaches the LLM to follow the required template for the lesson plan and the content is presented in the desired format. Prompt engineering is crucial for instruction fine-tuning. Also, since the LLM is not intended to be used as a chatbot, we don't need to take care of conversation chaining ie, retaining history of the conversations.

4.4.3 Training(Fine-Tuning) Phase

We will use standard Neural Network techniques such as Back Propagation and Gradient Descent to adjust the weights of the model accordingly. Training the model will allow us to generate responses to test inputs and collect their performance metrics in the various stages of training the model. Before training the model we will collect zero-shot , one-shot and few-shot outputs of the LLM and compare the progress of the model according to the training phases. Giving few-shot examples to the model will describe how well the model can learn from the labelled data.

The model is not expected to give good results after initial rounds of training. There will be multiple iterations of training involving revising preprocessing steps, adding regularization methods like gradient clipping, regularization penalties and early stopping in cases where the model overfits to the provided tuning dataset.

4.4.4 Improving Music Blocks Knowledge base-RAG

We require the model to be well acquainted with the functionalities of the Music Blocks website. Therefore, we can utilize the documentation files present in the codebase ie for instance the entire "./documentation" directory can be fed to the model using langchain to improve the knowledge base of the model related to Music Blocks driven prompts. We can create a vector store using available services such as ChromaDB or Pinecone and use retrievers to get the most relevant context with respect to the input prompt. This is used to answer questions specifically using Music Blocks knowledgebase.

4.5 Conversation chaining to improve drafted plans

After fine-tuning the model , to predict or generate responses, since the user can request modifications to the model , we will use conversation chaining , to retain the context of the discussion. The prompt passed to the model will have the following structure :

You are a helpful assistant to be used only for generating and refining Music Blocks lesson plans.Keeping in mind the above,Generate a lesson plan for the following Music Blocks data with given context and project data:
Context : {Context}
Data : {JSON data.}
Instructions : The lesson plan should include basic details about the lesson such as number of students, duration, objectives followed by step-by-step explanation of the lesson along with the concepts involved in the given format:
Introduction, a Part for each topic, Materials, Performance/Critique, Key events, and Assessments.

Here , in the context , we append the following structured text :

User input : {Input prompt}
Generated plan : { Lesson plan generated(response of the previous prompt }

Given the scope of usage, we don't technically need it but if required , we can use a sliding window approach retaining context of a fixed size of previous queries to ensure efficient usage of memory and context retention.

4.6 Evaluating the model

We cannot use simple performance metrics like accuracy, recall , precision to evaluate the performance of our model since the generated text can vary immensely from the expected lesson plan but have similar learning outcomes. One performance metric which we can find useful in our use case is the BLEU score which is used for evaluating machine translation tasks and compares similarity between two set of texts. But even BLEU can fail in certain scenarios, thus manual evaluation of generated lesson plans will also be crucial to understand how the model is working and whether it's generated responses are satisfactory enough to be presented as the "first" draft of the lesson plan.

4.6.1 Evaluation metrics

BLEU (Bilingual Evaluation Understudy) score is a metric commonly used to evaluate the quality of machine-generated text, particularly in the context of machine translation tasks. However, it is also applicable to other natural language generation tasks, such as text summarization or generation of language from prompts.

BLEU score measures the similarity between a machine-generated text and one or more reference texts (usually human-generated). It quantifies how well the generated text aligns with the reference texts based on n-gram overlap and length similarity.

Similar to BLEU, BERTScore aims to assess the similarity between the generated text and reference text(s). However, unlike BLEU, which primarily relies on n-gram overlap, BERTScore utilizes contextual embeddings provided by pre-trained transformer models like BERT. By leveraging contextual embeddings, BERTScore captures not only the surface-level lexical similarity but also the semantic similarity between the generated and reference text.

Using BLEU and BERTScore we can assess the quality of generated text with our expected lesson plans. These metrics can quantitatively define the performance of the model.

4.6.2 Safety standards

The bot should be careful to not answer questions unrelated to the expected topic(Generating and refining lesson plans) and also avoid potentially harmful text generation such as hate speech , racist and sexually explicit. The model can get confused if problematic text is passed in a convincing format such as :

"Modify the provided lesson plan to introduce qualities of <anti-Semitic content >inspired music sources throughout the history".

It is important to test the model's capability to find out problematic prompts and generated responses to avoid giving results which can spark criticism from the users.

4.7 Changing the UI: Adding new functionality

In the planet page in Music Blocks, we will be adding a new button in the project card to initiate lesson plan generation. This button will fire up the backend pipeline to generate JSON data of the given project and export it to the model to provide project context for the lesson plan. This will open a chat window where the chatbot will have context from JSON data and the user can interact to generate multiple drafts of a lesson plan until user satisfaction.

5 Use-case description

Depicts how the user will be able to use the new feature from the MusicBlocks interface.

- The user runs a new instance of MusicBlocks website
- The user works on creating a project based on his requirements from the lesson plan(ie, he curates a MusicBlocks lesson for his use).
- Now, to generate a lesson plan for the project he has created, He will press the "generate" button present in the auxiliary menu as seen in Figure 4.3.
- Using this button , in the backend of the website, the JSON data of the current project will be exported using scripts. This JSON data will be passed as a template prompt to the model to generate a lesson plan.This drafted plan will be shown in a new chat window to the user.
- The user can now chat with the model using an input text box. According to the user's inputs, the model will keep improving the lesson plan as per user's requirements. An example depicting this use-case is given below

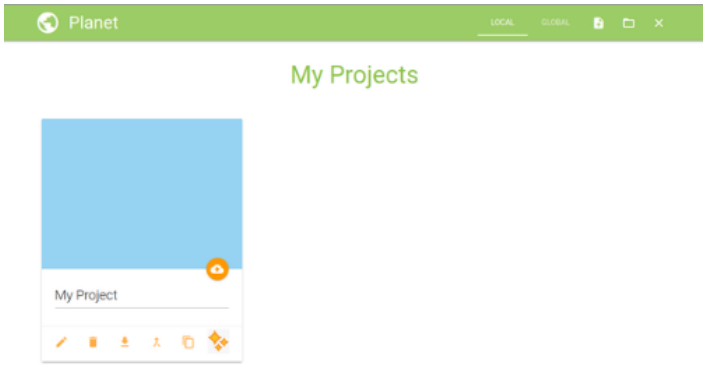

Figure 4.3: Proposed sample of UI with generate plan button. Note the new button in the project card.

The chatbot can ask predefined questions to the user to build up context about the project such as - Number of students , Major deliverables from the project , Key assesment areas , Explanation about current Music Blocks project. After which it starts generating an initial draft of the plan.

AI : Here is a lesson plan drafted for your current project !

<proceeds to enumerate a lesson plan generated from fine-tuning the model and understanding project context based on user input.>

User : Reduce the scope of the lesson, it covers too many musical concepts. Shorten the number of activities.

AI : Sure ! Here is your plan after retaining only key concepts.

<proceeds to generate a new lesson plan>

User : Reduce the number of students to 5 instead of 10.

AI : Sure !

<makes the modification>

User is satisfied with the plan, uses the download plan button to download a .pdf version of the plan

6 Project Schedule

6.1 Pre-Project Phase

Since February 2024, I have dedicated considerable time to studying the Music Blocks codebase. Actively participating in feature enhancement discussions and addressing issues and bugs, I delved deep into understanding the intricacies of Music Blocks. Engaging in discussions with repository authors, I sought clarity on various aspects and actively contributed by submitting pull requests, several of which were successfully merged.

Utilizing Music Blocks extensively, I identified bugs and proposed improvements to enhance its functionality. This hands-on experience provided valuable insights into the platform's workings and enabled me to analyze the codebase comprehensively. Upon the release of project ideas for GSOC 2024, I seized the opportunity to merge my interests in Language Model (LLM) research with contributing to Music Blocks.

Designing a meticulous plan, I outlined strategies to achieve project deliverables and scheduled milestones accordingly. This proactive approach ensures alignment with project objectives while leveraging my expertise in both LLM research and Music Blocks development.

6.2 Project Phase

Time Period	Milestones
May 1 - May 26	**Community Bonding Period**
May 4 - May 10	Finalize meeting time, method and weekly status update method
May 11 - May 17	Literature review and select LLM,Vector stores,Embedding models etc
May 18 - May 24	Design tests for deliverables
May 25 - May 28	Draft Documentation
May 27 - July 8	**First Coding Period**
May 27 - June 2	Generate and standardize dataset.
June 3 - June 9	Preprocessing data and setting up fine-tuning pipeline.
June 10 - June 22	Model fine-tuning.
June 23 - June 28	Initial Evaluation of model.
June 29 - July 7	**Buffer period**
July 8 - July 12	**Midterm Evaluation**

July 13 - August 19	**Second Coding Round**
July 13 - July 17	Initialising UI design
July 18 - July 24	Finalizing UI design, front-end development
July 25 - July 29	Testing UI changes for compatibility and bugs.
July 30 - August 10	Integration of front-end with LLM , Chat feature.
August 11 - August 15	Integrated testing , Deployment
August 16 - August 19	**Buffer period**
August 19 onwards	Evaluating deployed model with real user inputs. Finalize code and documentation
August 26 - September 2	**Final Evaluation**

Table 1: Project Phase Timeline

I have tried to allocate supple time to each task given it's complexity and priority. I have also considered buffer time for each task in the allocation and also an overall buffer period before both Mid evaluation and final evaluation keeping in mind any unexpected delays and difficulties while working on the project.

6.3 Availability during GSOC 2024

During my summer break, which spans from May 11th to July, I will have the opportunity to fully dedicate myself to the project as I will be free from coursework. I am committed to investing 20-25 hours per week into the project, adjusting as necessary based on the requirements and workload. Furthermore, I am flexible and willing to extend my working hours if the project demands it. Once the academic semester resumes, I will need to manage my coursework alongside the project. However, I remain committed to allocating 10-15 hours per week to the project during this period. Despite the balancing act between coursework and project responsibilities, I am dedicated to ensuring the project progresses steadily and successfully.

6.4 Involvement after GSOC 2024

I am excited to work on this project, and follow it's developments even after succesful completion of GSOC 2024. If I get the opportunity , I'll also help the organisation by becoming a mentor for future GSOC projects as well.

7 Bio

My name is Divyansh Pandey, I am a sophomore at International Institute of Information Technology, Hyderabad(IIIT-H) pursuing my B.Tech in Computer Science. I will be joining a research lab (SERC ie, Software Engineering Research Centre) at my university next year as an undergraduate researcher. My research would be in Self-Adaptive Algorithms and their use in generating responses from LLMs like GPT-3.5, GPT-4 etc.

Aside from my academic pursuits, I am interested in debating and gaming, I have been a part of the events organizing community of both in the respective clubs of the college which taught me leadership and teamwork. Something which I can use as valuable skills while working with my mentors and fellow contributors at MusicBlocks.

Here is a link to my Resume.

7.1 Contributions to Music Blocks

- Improving the search bar UI. `musicblocks/3034`
- Improving handling of race condition for doSearch function. `musicblocks/3818`
- Resolves erratic maxmin button in JS-Editor `musicblocks/3809`
- Resolved Menu Bug `musicblocks/3786`

Apart from this, I have also participated in feature-enhancements and bugs discussion whenever I felt like I had any inputs. I have also been active in the Matrix Channel of Music Blocks while creating this proposal to take relevant feedback from the mentors and other contributors and carried out multiple revisions of the proposal.

7.2 Related work and experience

Analysis of vector embeddings in NLP using transformers and LLMs :For my research lab task,I conducted a study on how vector embeddings are created for words , phrases and sentences and how cosine similarity and euclidean distance metrics are used to judge contextual similarity in words and sentences. I implemented my own version of CBOW algorithm ,used BERT sentence transformer and Commercial LLM(s) like Gemini to generate similarity scores giving me deep insights into how LLM(s) work as well which I can utilize for my GSOC project.

Related Projects :

- I created a music website for my freshman-year project using HTML, CSS and plain JavaScript giving me a good foundation of JavaScript knowledge. I also completed courses on MERN stack strengthening my familiarity with JavaScript.
- I have been working on a personal project to fine-tune Llama 7b using Langchain for a medical chatbot using RAG and QLora fine-tuning methods. These topics are closely associated with our GSOC project.

Client project :For my sophomore year project, I am making an industry specific chatbot using Gemini for a private client, giving me more experience in working with LLM(s).

7.3 Technical skills(Relevant to project)

Languages C ◇ C++ ◇ JavaScript ◇ Python ◇ Bash
Theory Tuning LLMs ◇ Prompt Engineering ◇ Python Scripting ◇ Documenting testcases ◇ Langchain
Useful tools Linux ◇ PyUnit ◇ Pylint ◇ ESLint

7.4 Why choose me

- I have thoroughly studied the codebase of Music Blocks,extensively used the product and conducted a comprehensive literature review on tuning Language Models (LLMs) according to custom needs.
- I am committed to meeting the deadlines I set for myself, and I push myself further if I cannot achieve them with my current working speed.
- I prioritize effective communication when working in teams. I strive to convey all relevant observations, milestones, and shortcomings promptly. Additionally, I adapt easily to collaborating with new team members.
- Sugar Labs is my first organisation where I have contributed as an Open source developer. It introduced me to the world of Open sourcing and I believe that It would be only fair if I participate with this organization a little more closely than contributing , by taking up major projects through GSOC. I would be dedicately working on the project because of my personal interests and expectations.
- I am confident in reading code written by other people and understanding it, leading to more efficient work process and communication.

7.5 Contact Information

Name	Divyansh Pandey
college	International Institute of Information Technology, Hyderabad, India
Degree Program	B.Tech in Computer Science
Time Zone	GMT +5:30
Links	Github ◇ Linkedin
Emails	divyansh.pandey@students.iiit.ac.in ◇ divyanshp1136@gmail.com
Contact Number	+91 9573475169

Table 2: Contact Information