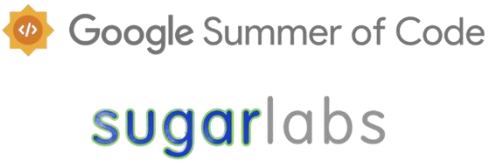
Sugar Labs

Google Summer of Code 2025



Project Information

Name: AI-powered Debugger for Music Blocks [here]
Length: 350 Hours (Large - 12 weeks)
Mentor: Walter Bender and Sumit Srivastava
Assisting Mentor: Devin Ulibarri

Student Information

Name: Om Santosh Suneri
Email: omsuneri@gmail.com
GitHub: omsuneri
LinkedIn: Om Santosh Suneri
Sugar labs Wiki: Omsuneri
Matrix: @omsuneri:matrix.org
Preferred Language: I'm proficient in English for communication, both spoken and written
Location: Chandigarh, India
Time Zone: Indian Standard Time (IST) (UTC +05:30)
Phone Number: +91 9405502004
Institution: University Institute of Engineering and Technology Panjab University, Chandigarh
Program: Bachelor of Engineering in Information Technology
Stage of Completion: 2nd Year (expected May 2027)

Introduction

A deep passion for coding and music has always driven my journey into the world of technology and open source. My adventure began at a very young age, blending seamlessly with my love for music and innovation. I have always believed that technology, when combined with creativity, can lead to extraordinary possibilities, and this belief has been the driving force behind my aspirations.

My fascination with open-source software started when I was just 11 years old. I was using VLC Media Player, a software I enjoyed not only for its functionality but also for its adaptability. When I discovered that VLC was open-source and that I could modify and personalize it to my liking, it was a game-changer for me. This revelation ignited my curiosity about Free and Open-Source Software (FOSS) and set me on a path of exploration and learning.

My interest in technology led me to participate in national science fairs, where I had the opportunity to present my projects on a larger platform. One of my proudest achievements was building a face-recognition-based system for the visually impaired using a Raspberry Pi (wearable device for visually impaired with this functionality). I showcased this prototype at a National Level Exhibition and Project Competition (Inspire Awards MANAK) while I was in Grade 9. This experience not only boosted my confidence but also introduced me to Raspbian, a Debian Linux-based operating system. My exposure to Linux further strengthened my enthusiasm for open-source technologies and encouraged me to delve deeper into this domain.

When I began my engineering journey in 2023, I attended a Software Freedom Day session at my college. Actively participating in this event opened my eyes to a broader world of open-source communities and software. This was a turning point that pushed me to seek out a FOSS project that aligned with my love for coding and music.

During my search, I discovered Music Blocks, a project by Sugar Labs that beautifully blends block-based programming with music creation. I joined the Sugar Labs community in September 2023 and became an active user of Music Blocks. The software's ability to combine coding with musical creativity resonated with me deeply, and I thoroughly enjoyed exploring its features.

In August 2024, I began contributing to Music Blocks, diving into issues, proposing solutions, and learning extensively through hands-on contributions. The welcoming community and the innovative nature of Music Blocks have provided me with invaluable learning experiences, and I am still actively contributing to this project.

My keen interest in open-source projects was further solidified when I led a team as a finalist (Ranked 2nd All India) in the Smart India Hackathon, where we worked on Improving open source software security using Fuzzing. This experience was a treasure trove of new learnings, as I delved into the intricacies of software testing and security, all while contributing to the open-source ecosystem. Additionally, I successfully completed Hacktoberfest 2024, which was another milestone in my open-source journey. These experiences have not only honed my technical skills but also deepened my love for open source, as I continue to explore, contribute, and grow within this vibrant community.

With a strong proficiency in the following technologies and a proven track record in open-source contributions, I am eager to leverage these skills while embracing new technologies to deliver impactful solutions for the Music Blocks community.

Programming Languages
Python, JavaScript, C++, Bash
Frameworks and Libraries
Hugging Face Transformers, PyTorch, TensorFlow, FastAPI, Pandas, NLTK
Web development
Html, CSS, JavaScript, Node.js
DevOps and Deployment
AWS, GitHub Actions, CI/CD pipelines
Data and ML tools
FAISS Elasticsearch Scikit-learn
Version control
Git, GitHub
Operating system
Linux (Debian, Raspbian, Fedora), MacOS, Windows

Why Sugar Labs?

My enthusiasm for music began when I was in Grade 4. The idea that just seven notes (Do, Re, Mi, Fa, Sol, La, Ti) could create a vast and diverse world of music fascinated me. I started learning piano, which introduced me to the intricacies of music theory and opened up a new realm of creativity and fun. At the same time, my interest in technology and coding grew, and I naturally gravitated towards opportunities where these two passions intersected.

When I discovered Music Blocks under the Sugar Labs, it felt like the perfect match. Music Blocks not only combines music and programming but also empowers learning through creativity—aligning perfectly with my interests and goals. The platform's use of a block-based programming environment to teach music theory and coding in an interactive way is a brilliant approach to education and creativity. It also allows me to explore and contribute to an open-source project that aligns with my passion for both technology and music.

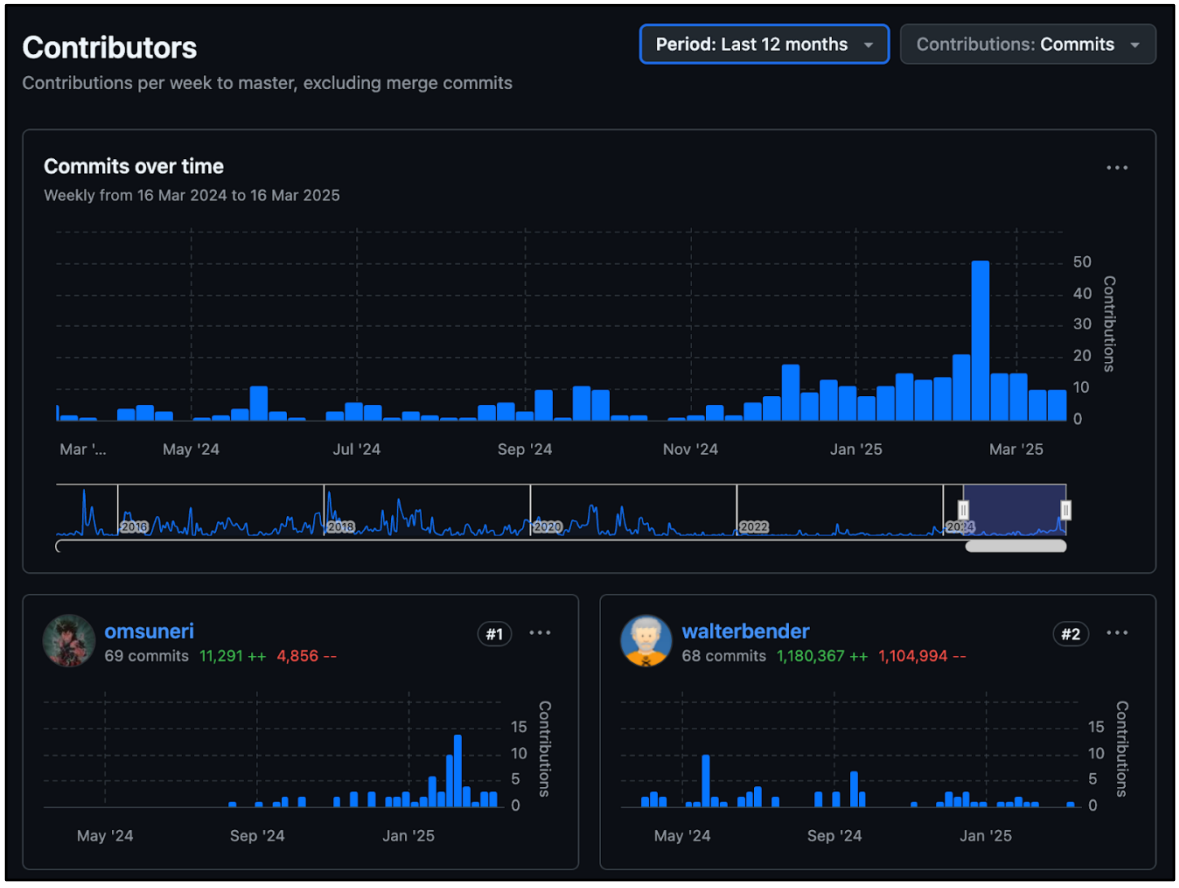
The tech stack of Music Blocks, primarily using JavaScript, matches my expertise and experience. I have been working with JavaScript extensively, and contributing to Music Blocks has helped me enhance my skills while contributing meaningful code to the project. Moreover, the organic and welcoming nature of the Sugar Labs community has motivated me to contribute regularly. I am an active contributor, consistently engaging with the community, resolving issues, and proposing enhancements on a daily basis.

Among the many open-source organizations, Sugar Labs stands out to me not just because of its innovative projects but also because of its mission to create educational tools that are free and accessible to all. I am inspired by the impact of Music Blocks on learners worldwide and want to contribute more to this mission through my work in the GSoC program.

Contributions to Sugar Labs

I have been an active contributor to Sugar Labs for the past seven months, during which **I have successfully merged 60+ pull requests (PRs) in the Music Blocks repository and 6 PRs in the www repository**. My contributions have ranged from bug fixes and feature enhancements, building test suites to documentation improvements, all aimed at strengthening the Music Blocks ecosystem. This hands-on experience has deepened my understanding of the project and its community, and I am committed to continuing my contributions in a positive and impactful manner.

I ranked 1st on the Contributors Chart over the period of last 12 months on the sugarlabs/musicblocks repository.



Issues Created by me

Below listed are the issues created by me on the sugarlabs/musicblocks repo

Issue	Title	Status
#4567	ESLint errors on master	**Closed**
#4483	Auxiliary menu Breaks !!	**Closed**
#4200	Creating tests for /js/js-exports/API	**Closed**
#4169	Github Actions need to be updated	**Closed**
#4078	piemenu in music keyboard widget responsive issue	**Closed**
#4070	The Pitch piemenu do not remember the last selected accidental value	**Closed**
#4067	Grid piemenu open along with another block's context menu	**Closed**
#4050	Optimize the closing of all the piemenus	**Closed**
#4038	Issue with Note Block collapsing	**Closed**
#4033	Temperament widget playing weird notes	**Open**
#4025	Issue with refreshing Search Bar in the palette menu	**Closed**

Pull Requests by me

Below listed are the Pull Requests created by me on the sugarlabs/musicblocks.

PR	Title	Difficulty	Status
#4568	Fixes #4567 ESLint errors on master	**Medium**	merged
#4554	Adding some more points to guide_addingtest.md	**Easy**	merged
#4552	Test suite for js/blocks/ExtrasBlocks.js	**Medium**	merged
#4548	Resolving ESLint issues on the js/ directory and.....	**Hard**	merged
#4527	Test suite for the js/Blocks/DrumBlocks.js	**Medium**	merged
#4519	Refactoring ESLint workflow for accuracy....	**Medium**	merged
#4517	Test suite for js/Blocks/BooleanBlocks.js	**Medium**	merged
#4484	Resolves conflict in index.html	**Medium**	merged

#4464	Test suites for the js/blocks/DictBlocks.js	**Medium**	merged
#4458	Test suite for js/js-export/constraints.js	**Medium**	merged
#4457	Test suite for js/js-export/ASTutils.js	**Medium**	merged
#4451	Test suite for js/js-export/generate.js	**Medium**	merged
#4450	Test suite for the js/js-export/export.js	**Medium**	merged
#4438	Workflow to run test on PR instead the upstream master	**Hard**	merged
#4427	Workflow refactored	**Hard**	merged
#4422	Resolves all the errors with VolumeActions.test.js	**Hard**	merged
#4408	Workflow for the running tests	**Hard**	merged
#4403	Test suite for js/palette.js	**Medium**	merged
#4397	Resolving all the test case errors with rubrics.js and....	**Medium**	merged
#4388	Creating a guide for adding test suite	**Easy**	merged
#4387	Test suite for js/artwork.js	**Medium**	merged
#4386	Test suite for js/turtle-singer.js	**Medium**	merged
#4383	Test suite for js/turtledefs.js	**Medium**	merged
#4382	Test suite for js/protoblocks.js	**Medium**	merged
#4378	Test suite for js/turtles.js	**Medium**	merged
#4377	Test suite for the js/themebox.js	**Medium**	merged
#4368	Test suite for turtle-painter.js	**Medium**	merged
#4363	Test suite for rubrics.js and blockfactory.js	**Medium**	merged
#4359	Jest test suite for js/macros.js	**Medium**	merged
#4334	Test for base64Utils.js	**Medium**	merged
#4329	Removing the toggle dark mode button from the planet...	**Medium**	merged
#4321	Syncing dark mode between planet page and main page	**Hard**	merged
#4318	Tests for background.js and planetInterface.js	**Medium**	merged
#4317	Test suite for notation.js	**Medium**	merged

#4314	Test suite for mxml.js	**Medium**	merged
#4310	Adding SITAR as a new instruments	**Easy**	merged
#4281	Improving Home Button Function	**Hard**	merged
#4280	Exporting Blocks artwork in PNG	**Hard**	merged
#4268	Increasing size of set pitch preview	**Easy**	merged
#4242	Resolves the module Exports undefined errors of the....	**Easy**	merged
#4237	Pie menu is remembering the last selected index...	**Medium**	merged
#4227	Adding test for the Turtleactions	**Medium**	merged
#4212	Optimising the number selector in the note block octave	**Easy**	merged
#4201	Test file for the API	**Medium**	merged
#4175	FIXES #4171 pen args mismatched inside note blocks	**Medium**	merged
#4168	Test for the platformstyle.js	**Medium**	merged
#4143	Resolving the test error	**Medium**	merged
#4141	Creating Jest config and setup file	**Easy**	merged
#4130	Adding test for mathutils	**Medium**	merged
#4084	Making grid piemenu to rotate on click	**Easy**	merged
#4083	Created a prompt notification for deleted blocks	**Easy**	merged
#4071	FIXES #4070 The Pitch piemenu do not remember....	**Hard**	merged
#4060	Increasing the size of chord pie menu	**Easy**	merged
#4059	FIXES #4051 Add note piemenu opening behind....	**Easy**	merged
#4035	FIXES #4025 Issue with refreshing search bar....	**Medium**	merged
#4032	FIXES #4012 Scalar step doesn't work for....	**Medium**	merged
#4029	FIXES #4018 More default EDO's for....	**Medium**	merged
#4022	FIXES #4018 More default EDO's for....	**Medium**	merged
#4010	FIXES #3921 Re Examine our default temperament....	**Easy**	merged
#4006	FIXES #4000 Regression set default instrument....	**Easy**	merged

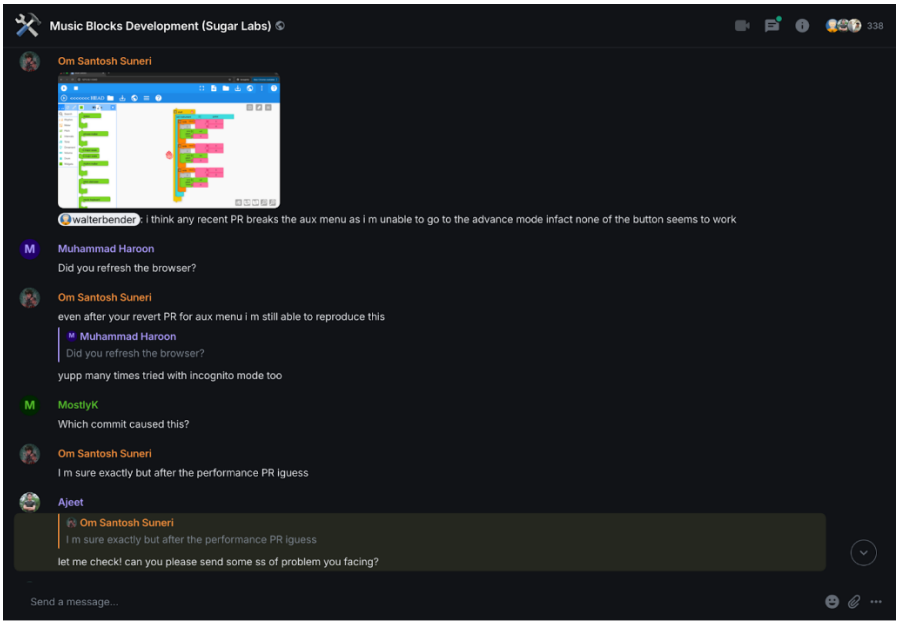
#3979	FIXES #3895 Add alphabet G as a block found in easy....	**Easy**	merged

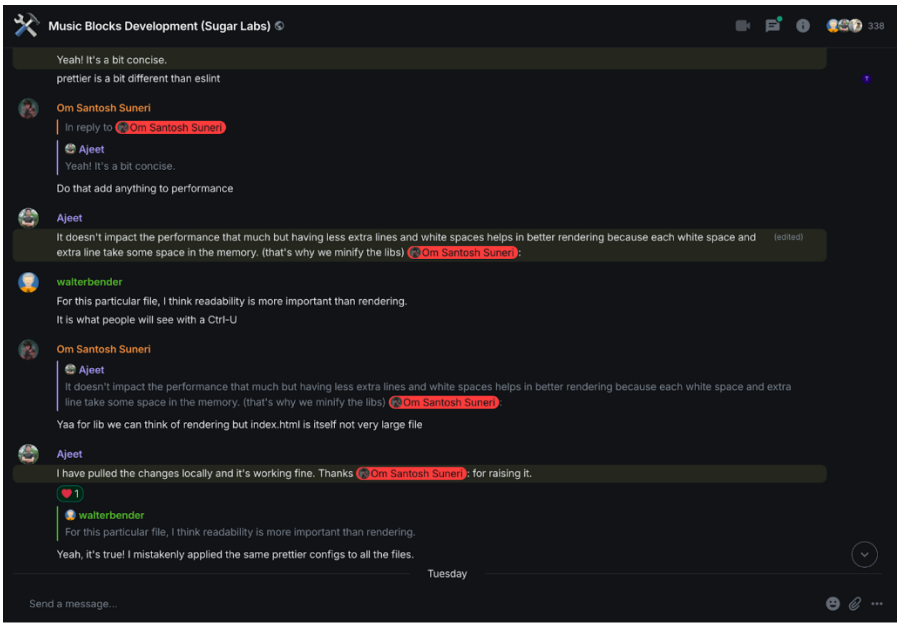
Below listed are the Pull Requests created by me on the sugarlabs/www repo

PR	**Title**	**Difficulty**	**Status**
#626	Updating number counter with commas	**Easy**	merged
#616	Adding Info page for community channels	**Medium**	open
#551	Update donate.html	**Easy**	merged
#533	Linking the custom.css for the custom styling	**Easy**	merged
#530	Adding new section named reference video in MB page	**Medium**	merged
#513	Creating a proper number counter animation....	**Medium**	merged
#467	FIXES #466 Replace Sugar stories table in html table	**Easy**	merged

Presence in the Community

As an open-source contributor, I prioritize assisting new contributors to engage with the project's issues. Active involvement within the community and regular participation in biweekly meetings underscore my commitment to fostering collaboration and supporting others in their contributions.
A few screenshots from the Element public conversation are here:





I have actively supported the community by identifying and reporting bugs, assisting with project setup, and making meaningful contributions to GitHub. Additionally, I have engaged in insightful discussions about the project with my mentors, Devin Ulibarri, Walter Bender and Sumit Srivastava which have greatly enriched my understanding and approach to contributing effectively.

Project Details

AI-powered Debugger for Music Blocks

Inspiration Behind the Idea

The idea for an AI-powered debugger for Music Blocks was born during one of the community's biweekly meetings. A newcomer to the community presented their project, which, despite having all the blocks seemingly placed correctly, failed to produce any sound. The mentors and community members, including myself, began debugging the project collaboratively. After some analysis, we discovered that the issue was caused by multiplying the hertz value by zero, which resulted in no sound being generated. While the solution was simple, the process of identifying the problem highlighted a common challenge faced by learners: troubleshooting projects without proper guidance.

This experience sparked the idea of creating a virtual assistant that could act as a debugger for Music Blocks. The debugger would help users identify and resolve issues in their projects, such as the Hertz multiplication problem, while also providing real-time answers to questions, explaining concepts, and offering creative suggestions. This combination would not only simplify the debugging process but also serve as a virtual instructor, guiding learners through their creative journey and helping them overcome obstacles independently.

I proposed this idea to the mentors, and it was met with enthusiasm. The potential to make Music Blocks more accessible and user-friendly for learners of all skill levels resonated strongly with the community's goals. I am excited to bring this vision to life during the GSoC 2025 period and contribute to making Music Blocks an even more powerful and inclusive educational tool.
The idea for this project was proposed by me in the sugarlabs/GSoC repository of Sugar Labs through these merged pull requests.

- Adding idea for gsoc-2025
- Rewriting the idea with more AI specification
- Changing the idea title suggested by @walterbender

Introduction

Music Blocks is a visual, block-based programming environment designed to teach programming and music concepts in an engaging and interactive way. However, as highlighted during my experience in the biweekly meetings, users often face challenges when troubleshooting their projects or fully utilizing the platform's features. Beginners, in particular, may struggle to understand error messages or determine how to achieve their creative goals, while advanced users may encounter difficulties in debugging complex projects. These challenges can hinder the learning process and reduce the platform's effectiveness as an educational tool.

To address these issues, this project proposes the development of an AI-powered debugger integrated into Music Blocks. The debugger will assist users in identifying and resolving issues in their projects, while also providing real-time support by answering user queries, explaining features, and offering creative suggestions. By leveraging open-source Large Language Models (LLMs), fine-tuned on a curated dataset of Music Blocks projects and debugging scenarios, the debugger will be capable of understanding and interacting with users' projects in a meaningful way. To enhance its contextual understanding, Retrieval-Augmented Generation (RAG) will be implemented, combining the generative capabilities of LLMs with a comprehensive knowledge base of Music Blocks documentation, lesson plans, and example projects. This ensures that the debugger can provide accurate, context-aware responses tailored to the user's needs.

The debugger will be seamlessly integrated into the Music Blocks platform with a user-friendly interface, while the backend will be built using FastAPI for efficient API development and deployed on AWS for scalability and accessibility. To ensure high-quality interactions, prompt engineering will be employed to minimize hallucinations and improve response accuracy. Additionally, the debugger's output will be designed in a structured and educational format, such as step-by-step explanations, to enhance the learning experience for students and make complex concepts more digestible.

By the end of the project, Music Blocks will have an intelligent assistant that simplifies troubleshooting, encourages creative exploration, and makes the platform more accessible to users of all skill levels. This enhancement aligns with Sugar Labs' mission to create engaging and accessible educational tools.

I am confident that my proposed approach, combined with my technical skills and dedication, will deliver a robust and impactful solution that benefits the Music Blocks community and supports learners worldwide.

Deliverables

Here are the major objectives on which I want to work as a part of Google Summer of Code 2025.

1. Converter for JSON-to-Text Representation

Developing a converter tool that transforms Music Blocks' JSON project code into a simplified, human-readable text format. This text representation will serve as input for the LLM and help users understand their projects better.

Implementation

- **JSON Parsing**: I will implement a robust JSON parsing mechanism using Node.js's fs module to read and parse Music Blocks JSON project code. The JSON.parse function will be used to convert the JSON string into a JavaScript object for further processing.

- **Block Mapping**: I will create a mapping system using blockMap, which associates block IDs with their corresponding data. This will enable efficient lookup and processing of connected blocks. For example, a "Pitch" block would be mapped to a textual description like "Pitch calculation.

- **Text Generation**: I will develop a recursive function, processBlock, to traverse the block hierarchy and generate a textual representation of the project. This function will handle both clamp connections (nested blocks) and sequential connections (block-to-block logic).

- **Testing and Validation**: I will implement comprehensive testing to ensure the converter is accurate, reliable, and capable of handling a wide range of Music Blocks projects.

- **Integration**: I will modularize the converter to allow seamless integration into the debugger pipeline, enabling real-time processing of Music Blocks projects.

JSON-to-Text Representation Convertor code
AI-powered Debugger for Music Blocks

Impact

Rigorous testing and validation will ensure the converter is robust and ready for real-world use, while community feedback will help tailor the tool to users' needs.

2. Data Selection and Preparation for Fine-Tuning and RAG

Curate and prepare a high-quality dataset to fine-tune the Large Language Model (LLM) and implement Retrieval-Augmented Generation (RAG). This dataset will serve as the foundation for training the model to understand and debug Music Blocks projects effectively.

Implementation

- **Data Collection**: I will gather Music Blocks projects, debugging scenarios, including community contributions, and planet projects. Additionally, I will collect real-world errors and debugging scenarios from Music Blocks teachers via a feedback form, ensuring the dataset is comprehensive and the model robust. This approach will address practical challenges and enhance the debugger's effectiveness.

- **Annotation**: Each dataset entry will be annotated with correct solutions, detailed explanations, and debugging steps. For example, if a project fails to produce sound, the annotation will include the root cause (e.g., "Hertz value set to zero") and the solution (e.g., "Set Hertz value to a non-zero number"). This will enable the model to learn not only how to identify issues but also how to resolve them.

- **Data Augmentation**: To improve the model's generalization, I will augment the dataset by generating synthetic examples, such as variations of

common errors or alternative project structures. This will help the model handle edge cases and unfamiliar scenarios effectively.

Impact

A well-curated and annotated dataset will significantly enhance the model's ability to understand and debug Music Blocks projects. By including diverse use cases and synthetic examples, the model will be better equipped to handle real-world scenarios, improving its accuracy and reliability.

3. Fine-Tuned LLM for Music Blocks

Fine-tune an open-source Large Language Model (LLM), such as LLaMA 3 or DeepSeek-R1, on the curated Music Blocks dataset to create a Music Blocks-specific model. This fine-tuned model will be capable of understanding and responding to Music Blocks-related queries, debugging scenarios, and project-specific challenges.

Implementation

- **Model Selection**: I will evaluate and select a base LLM model based on its performance, size, and compatibility with open-source tools. Models like LLaMA 3 or DeepSeek-R1 will be considered due to their balance of efficiency and capability.

- **Fine-Tuning Setup**: Using Hugging Face's transformers library, I will set up a fine-tuning pipeline with tools like trl (Transformers Reinforcement Learning) or TensorFlow. This pipeline will include data loading, model training, and evaluation steps to ensure a streamlined process.

- **Evaluation**: The fine-tuned model will be evaluated on unseen Music Blocks projects and debugging scenarios. Metrics such as accuracy, response relevance, and user satisfaction will be used to assess its performance.

- **Iteration**: Based on evaluation results and community feedback, I will refine the model through multiple iterations of fine-tuning. This iterative process will ensure continuous improvement and alignment with user needs.

Impact

A fine-tuned LLM specifically tailored for Music Blocks will significantly enhance the system's ability to understand and respond to user queries. By leveraging a model optimized for Music Blocks, the system will provide accurate, contextually relevant, and actionable solutions, improving the overall user experience.

4. Retrieval-Augmented Generation (RAG) for Music Blocks

Implement a Retrieval-Augmented Generation (RAG) system to enhance the debugger's contextual understanding and response accuracy. By combining the strengths of retrieval-based and generative models, the RAG system will provide users with precise, context-aware responses to their queries.

Implementation

- **Knowledge Base Creation**: I will build a comprehensive knowledge base using Music Blocks documentation, lesson plans, example projects, and planet projects. This data will be organized into structured formats, such as FAQs and guides, to facilitate efficient retrieval.

- **Baseline Evaluation with BM25**:
    - Creating evaluation benchmarks (evals) to assess retrieval performance.
    - Implementing BM25 as a baseline retrieval method on these evals to establish a performance benchmark.
    - Comparing the BM25 results with the RAG system to measure improvements in relevance and accuracy.

- **Retrieval Mechanism**: I will use a high-performance retrieval system, such as FAISS, to fetch relevant information from the knowledge base. The data will be indexed for fast and accurate searching, ensuring that the most pertinent information is retrieved for each query.

- **Response Generation**: The retrieved information will be combined with the generative capabilities of the fine-tuned LLM to produce context-aware responses. For example, if a user asks, "Why is my project not producing sound?", the system will retrieve relevant documentation and generate a response like, "The Hertz value is set to zero which is not valid."

- **Testing and Optimization**: I will rigorously test the RAG system on a variety of user queries to ensure accuracy and relevance. The retrieval and generation processes will be optimized for speed and precision, ensuring a seamless user experience.

Impact

The RAG system will significantly enhance the debugger's ability to provide accurate and contextually relevant responses. By leveraging a structured knowledge base and efficient retrieval mechanisms, the system will act as a reliable assistant for users, offering solutions and explanations grounded in verified information.

5. Music Blocks-Synced User-Friendly UI

Design and integrate a user-friendly UI for the debugger into the Music Blocks platform using the widget interface. This UI will seamlessly blend with the existing Music Blocks environment, ensuring a cohesive and intuitive user experience.

Implementation

- **UI Design**: Leveraging the existing Music Blocks UI framework, I will create a debugger widget that is visually consistent with the platform's aesthetics. The design will prioritize simplicity and ease of use, ensuring that users of all skill levels can interact with the debugger effortlessly.

- **Features:** The UI will include essential features such as chat history, debugging suggestions, and step-by-step explanations. For instance, guiding users through the debugging process in an interactive and educational manner.

- **Accessibility**: I will ensure the UI is accessible to users of all skill levels by incorporating clear instructions, tooltips, and user-friendly navigation. This will make the debugger approachable for beginners while remaining useful for advanced users.

- **Testing**: I will conduct usability testing with community members to gather feedback and identify areas for improvement. This iterative process will ensure the UI meets the needs of its users and aligns with the Music Blocks community's expectations.

Impact

A user-friendly and seamlessly integrated UI will enhance the overall usability of the Music Blocks platform. By providing real-time assistance, debugging support, and step-by-step guidance, the debugger will empower users to learn, create, and troubleshoot more effectively.

6. FastAPI Endpoints and AWS Deployment

I will develop FastAPI endpoints to serve the debugger and deploy the model on AWS, ensuring scalability, accessibility, and high performance. This will enable users to interact with the debugger seamlessly, regardless of their location or the scale of their usage.

Implementation

- **API Design**: I will design RESTful APIs using FastAPI to handle user queries and model inference. Key endpoints will include sending queries, receiving responses, and providing debugging suggestions. The API will be designed with clarity and simplicity in mind, ensuring ease of integration with the Music Blocks platform.

- **AWS Deployment**: I will deploy the model on AWS server (Provided by Sugar Labs) to ensure high availability and performance. Load balancing and auto-scaling will be implemented to handle and maintain system reliability.

Impact

By developing FastAPI endpoints and deploying the model on AWS, I will create a scalable and accessible solution for users worldwide. The optimized API and robust deployment infrastructure will ensure low-latency responses and high availability, making the debugger a reliable tool for the Music Blocks community.

7. Error Detection and Guided Problem-Solving

The fine-tuned LLM will analyze user projects and queries with a focus on error detection while promoting problem-solving skills. Instead of providing direct answers, the system will guide users toward identifying and resolving issues independently.

Implementation

- **Error Detection**: I will implement an error detection mechanism within the LLM to identify common issues in Music Blocks projects, such as incorrect block configurations, missing connections, or invalid parameters. When an error is detected, the system will flag it and provide a contextual explanation to help users understand why the issue is problematic. For example, if a user's project fails to produce sound, the system might detect that the Hertz value is set to zero and explain why this prevents audio output.

- **Output Formatting**: The system will present errors and suggestions in a structured format to ensure clarity and usability. For instance, the output might appear as:
    - **Error Detected**: "The Hertz value in the 'Play Note' block is set to zero, which will produce no sound."
    - **Suggestion**: "Review the Hertz value and ensure it is set to a non-zero number within the audible range."

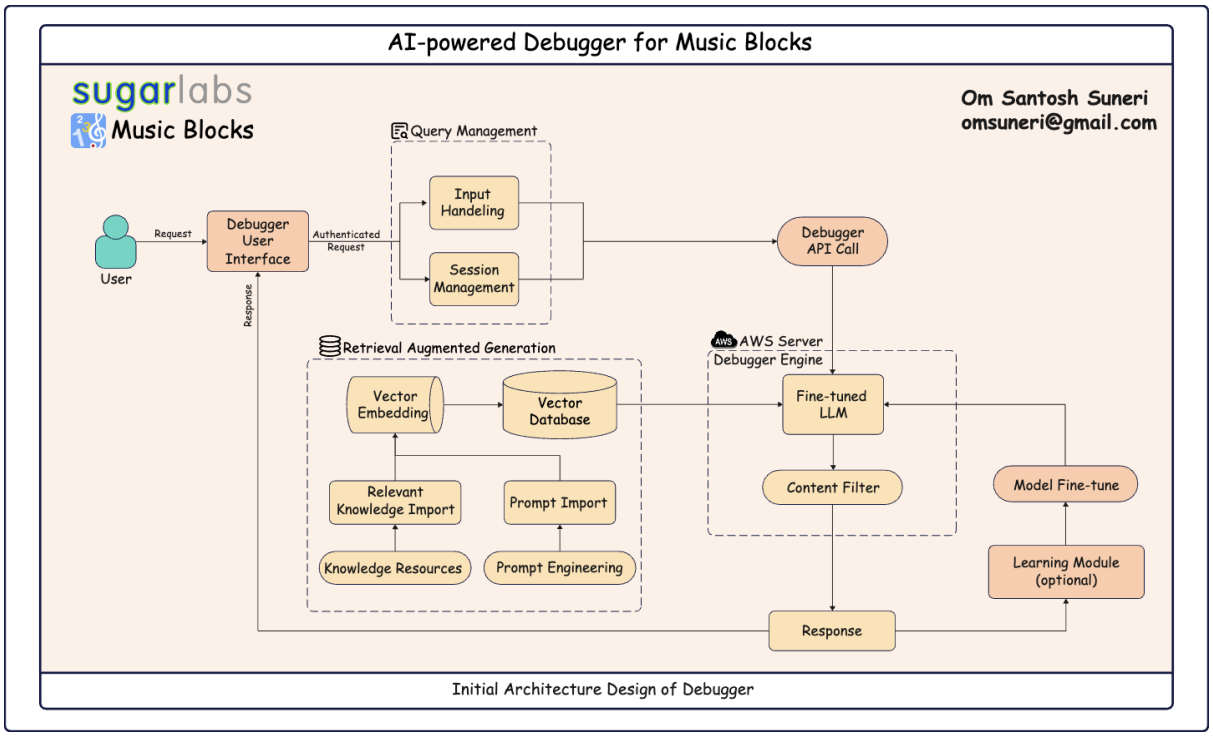
- **Guided Problem-Solving**: Instead of providing exact solutions, the LLM will offer hints and step-by-step guidance to help users troubleshoot and fix issues on their own. For example, it might suggest:
    - "Check the connections between the 'Pitch' and 'Play Note' blocks to ensure they are properly linked."
    - "Verify that the 'Repeat' block has a valid input for the number of iterations."

- **Enhanced User Support**: To further assist users, the system will include tooltips and interactive prompts that provide additional context and guidance. This approach ensures users can resolve issues independently while learning from the process.

Impact

This approach will empower users to develop critical thinking and debugging skills, making them more proficient in using Music Blocks. By focusing on guided problem-solving rather than direct answers, the system will enhance the educational value of the debugger, helping users become more confident and independent in their project creation and troubleshooting efforts.

Architecture of the Debugger for Music Blocks



The architecture for the Debugger is designed to provide a scalable, context-aware, and efficient system for assisting users. At its core, the system integrates Retrieval-Augmented Generation (RAG) to combine the strengths of retrieval-based and generative models. The frontend, a Music Blocks based debugger interface, handles user interactions. User queries are processed by the Query Management module, which ensures proper input handling and session management. The RAG system retrieves relevant knowledge from a vector database, which stores vector embeddings of Music Blocks documentation, lesson plans and project examples. This knowledge is combined with user queries using prompt engineering to generate accurate and context-aware responses. The backend, hosted on AWS, includes a Debugger Engine powered by a fine-tuned LLM (e.g., LLaMA or DeepSeek-R1) that has been specifically trained on Music Blocks data. A content filter ensures responses are accurate and appropriate, while an optional learning module allows the system to improve over time based on user feedback.

The architecture is designed for scalability and reliability, leveraging AWS infrastructure to handle high traffic and ensure high availability. Key features include real-time synchronization with the user's workspace, enabling the

debugger to provide context-aware suggestions and debugging tips dynamically. The use of a vector database ensures efficient knowledge retrieval, reducing response latency and improving accuracy. The fine-tuned LLM ensures responses are tailored to the Music Blocks context, while the content filter maintains the quality and safety of interactions.

User Interface and Experience of AI-powered Debugger

Creating a project in Music Blocks can be an exciting and creative process, but it's common to encounter errors or unexpected behavior due to issues like incorrect block connections, misplaced values, or logical flaws in the design. To address these challenges, we've introduced an AI-Powered Debugger in the widget section of Music Blocks, available for both Beginner and Advanced modes. This feature is designed to help users identify and resolve errors in their projects, ensuring a smooth and frustration-free experience.

Accessing the Debugger

When a user faces difficulty in their project, they can easily access the Debugger by searching for it in the Palette Search or by dragging it directly from the Widget Palette onto the canvas. Once the Debugger block is placed on the canvas, clicking on it opens the Debugger Widget, where users can interact with the AI-powered debugger to troubleshoot their projects.

Debugger Widget Features

1. **Real-Time Project Synchronization**: The Debugger Widget is designed for maximum convenience and efficiency. As soon as the user opens the Debugger, it automatically selects and loads the project currently present on the canvas by default. This eliminates the need for manual intervention, ensuring that the Debugger has immediate access to the project for analysis.

2. **Interactive Text Field with Stock Questions**: At the bottom of the Debugger Widget, users will find a text field where they can describe their issue or ask questions about their project or general music/programming-related queries. To assist users, a set of stock questions is also provided as examples. For instance, a user might type or select:

"I want to create a forever-playing note sequence, but it's not working. Can you please debug my project?"
This text field is accompanied by a Send button, which submits the user's query along with the project code to the Debugger Engine.

3. **AI-Powered Analysis**: Once the query and project code are sent, the Debugger Engine analyzes the project and provides a detailed response. This response includes:
    - Potential errors in the project.
    - Possible reasons for the unexpected behavior.
    - Suggestions for resolving the issue.

    For instance, the Debugger might identify a missing loop block or an incorrectly connected note block and suggest how to fix it.

4. **Guided Learning**: The Debugger not only helps users fix their projects but also promotes guided learning. By understanding the errors and the suggested solutions, users can learn how to avoid similar mistakes in the future. This feature helps them build a deeper understanding of Music Blocks' functionality.

5. **Seamless Integration**: The Debugger Widget is designed to integrate seamlessly into the Music Blocks environment, allowing users to continue working on their projects while simultaneously using the Debugger. Additionally, the Debugger not only identifies errors but also guides users in utilizing **built-in debugging tools** to enhance their troubleshooting experience. For example, it can recommend:
    - Using **Print and Comment Blocks** to track variable values or project progression for better understanding.
    - Using **run step-by-step/run slowly** to understand the error point in the project.
    - Utilizing the **Status Widget** to monitor project status in real time.

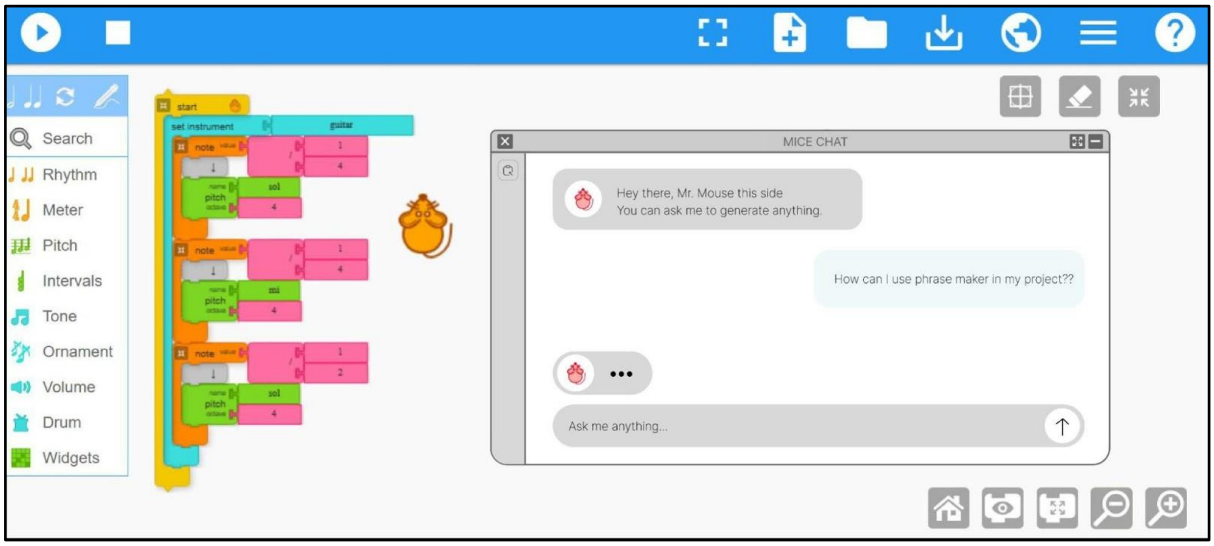
Example Workflow

1. A user creates a project but notices that the music sequence isn't playing as expected.
2. They drag the Debugger block from the Widget Palette onto the canvas and click on it to open the Debugger Widget.
3. In the text field, they type: "Why isn't my music sequence playing forever?" and click Send.
4. The Debugger analyzes the project and responds for example:

   "It looks like you're missing a 'Forever' loop block. Add a 'Forever' block around your note sequence to make it play continuously."

5. For guided learning, the Debugger may provide suggestions to use built-in debugging tools such as the Print Block, Status Widget, and more.
6. The user implements the suggestion, and their project now works as intended.

Below is the proposed design of the Debugger, and this is how the final implementation will look:



Figma design of debugger

Some Live Demo

Conversion of JSON project code to text representation for LLM

Input Project JSON code

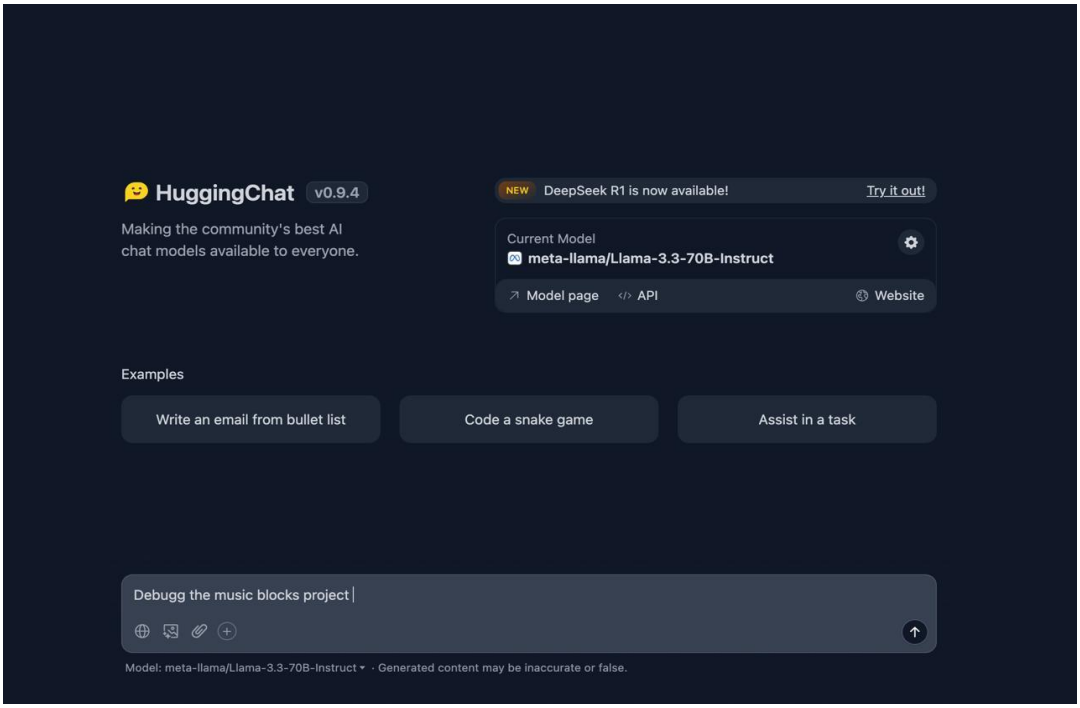

```
[[0,["start",{"id":1741159698830,"collapsed":false,"xcor":-125,"ycor":104,"heading":0,"color":-10,"shade":60,"pensize":5,
"grey":100}],193,-2,[null,1,null]],[1,"settimbre",207,39,[0,2,4,3]],[2,["voicename",{"value":"guitar"}],358,39,[1]],[3,
"hidden",207,575,[1,null]],[4,["newnote",{"collapsed":false}],221,71,[1,5,8,12]],[5,"divide",323,71,[4,6,7]],[6,["number",
{"value":1}],409,71,[5]],[7,["number",{"value":4}],409,103,[5]],[8,"vspace",235,103,[4,9]],[9,"pitch",235,135,[8,10,11,
null]],[10,["solfege",{"value":"sol"}],309,135,[9]],[11,["number",{"value":4}],309,167,[9]],[12,"hidden",221,229,[4,13]],
[13,["newnote",{"collapsed":false}],221,229,[12,14,17,21]],[14,"divide",323,229,[13,15,16]],[15,["number",{"value":1}],409,
229,[14]],[16,["number",{"value":4}],409,261,[14]],[17,"vspace",235,261,[13,18]],[18,"pitch",235,293,[17,19,20,null]],[19,
["solfege",{"value":"mi"}],309,293,[18]],[20,["number",{"value":4}],309,325,[18]],[21,"hidden",221,387,[13,22]],[22,
["newnote",{"collapsed":false}],221,387,[21,23,26,30]],[23,"divide",323,387,[22,24,25]],[24,["number",{"value":1}],409,387,
[23]],[25,["number",{"value":2}],409,419,[23]],[26,"vspace",235,419,[22,27]],[27,"pitch",235,451,[26,28,29,null]],[28,
["solfege",{"value":"sol"}],309,451,[27]],[29,["number",{"value":4}],309,483,[27]],[30,"hidden",221,545,[22,null]],[31,
"hiddennoflow",206,1072,[null,null]]]
```


Output Simplified text representation

```
Start of Project
|   ├── Start → {id: 1741159698830, xcor: -125, ycor: 104, heading: 0, color: -10, shade: 60, pensize: 5, grey: 100}
|   |   ├── Set Instrument → guitar
|   |   |   ├── Note
|   |   |   |   ├── Divider Block → 1 / 4 = 0.25
|   |   |   |   ├── Pitch Calculation
|   |   |   |   |   ├── Solfege: sol
|   |   |   ├── Note
|   |   |   |   ├── Divider Block → 1 / 4 = 0.25
|   |   |   |   ├── Pitch Calculation
|   |   |   |   |   ├── Solfege: mi
|   |   |   ├── Note
|   |   |   |   ├── Divider Block → 1 / 2 = 0.5
|   |   |   |   ├── Pitch Calculation
|   |   |   |   |   ├── Solfege: sol
```

Live video of output

Please click anywhere on the image to watch the demo



Impact on Sugar Labs

The implementation of this project will have a transformative impact on Sugar Labs and the Music Blocks community. By introducing an intelligent, context-aware debugger, the platform will become more accessible and user-friendly, empowering users of all skill levels to learn, create, and troubleshoot projects with confidence. The integration of Retrieval-Augmented Generation (RAG) and fine-tuned LLMs will ensure accurate, relevant, and educational responses, while features like error detection and guided problem-solving will foster critical thinking and independence among users. This project will not only enhance the overall user experience but also strengthen the Music Blocks ecosystem by providing a reliable, scalable, and educational tool that aligns with Sugar Labs' mission of promoting creativity, collaboration, and learning. Additionally, the real-time synchronization and AWS-based deployment will ensure the system is robust, scalable, and accessible to users worldwide, further solidifying Sugar Labs' position as a leader in educational technology.

As an experienced developer and passionate open-source contributor, I am excited to take on the responsibility of integrating the AI-powered Debugger into Music Blocks and delivering this feature during the GSoC 2025 period.

Timeline

I will adhere to a well-defined timeline, beginning with preparatory tasks in the pre-GSoC period and advancing through focused development during the GSoC coding phase, ensuring consistent progress and timely completion of the project.

Community Bonding Period

- Conduct interviews with teachers to identify common pain points in debugging Music Blocks projects.
- Analyze existing Music Blocks projects to catalog frequent error patterns.

Week 1-2: JSON-to-Text Converter Development

- Implement JSON parsing using Node.js's fs module to read and parse Music Blocks JSON project code.
- Develop the processBlock function to recursively traverse the block hierarchy and generate a textual representation.

Deliverable: Basic JSON-to-text converter with block mapping and text generation.
Tech Stack: Node.js, JavaScript, JSON parsing, FS module.

Week 3-4: Data Collection and Annotation for Fine-Tuning

- Gather Music Blocks projects, debugging scenarios, including community contributions, and planet projects.
- Additionally, collecting real-world errors and debugging scenarios from Music Blocks teachers via a feedback form.
- Annotate the dataset with correct solutions, explanations, and debugging steps.

Deliverable: Curated and annotated dataset ready for fine-tuning.
Tech Stack: Python, Pandas, NLTK, Hugging Face Datasets.

Week 5-6: Fine-Tuning the LLM

- Select a base LLM (e.g., LLaMA 3 or DeepSeek-R1) and set up the fine-tuning pipeline.
- Fine-tune the LLM on the curated Music Blocks dataset.

Deliverable: Fine-tuned LLM capable of understanding Music Blocks-specific queries.
Tech Stack: Hugging Face Transformers, trl (Transformers Reinforcement Learning), TensorFlow, Weights & Biases.

Week 7: RAG System Implementation

- Build a knowledge base using Music Blocks documentation, tutorials, lesson plans, and example projects.
- Implement a retrieval mechanism using FAISS for efficient knowledge retrieval.
- Integrate the retrieval system with the fine-tuned LLM for context-aware response generation.

Deliverable: RAG system prototype with knowledge base and retrieval mechanism.
Tech Stack: FAISS, Hugging Face Transformers, Python.

Week 8: UI Design and Integration

- Design a debugger widget that aligns with the Music Blocks UI framework.
- Implement real-time synchronization between the debugger and the user's workspace.
- Add features like chat history, debugging suggestions, and step-by-step explanations.

Deliverable: User-friendly debugger UI integrated into Music Blocks.
Tech Stack: JavaScript, HTML/CSS, Music Blocks UI framework.

Week 9: FastAPI Endpoints and AWS Deployment

- Design RESTful APIs using FastAPI to handle user queries and model inference.
- Containerize the application using Docker for consistent deployment.
- Deploy the model on AWS setting up load balancing and auto-scaling.

Deliverable: Deployed debugger with FastAPI endpoints and AWS infrastructure.
Tech Stack: FastAPI, Docker, AWS, CloudWatch.

Week 10: Error Detection and Guided Problem-Solving

- Implement error detection algorithms to identify common issues in Music Blocks projects.
- Develop structured output formatting for error messages and suggestions.
- Add guided problem-solving features (hints, step-by-step guidance) to help users troubleshoot independently.

Deliverable: Error detection and guided problem-solving module.
Tech Stack: Python, Music Blocks JSON parser.

Week 11: Testing and Optimization

- Conduct comprehensive testing of the debugger on various Music Blocks projects.
- Optimize the RAG system and LLM for low-latency responses and high accuracy.
- Use AWS CloudWatch to monitor system performance and log user interactions.

Deliverable: Fully tested and optimized debugger.
Tech Stack: Python, Pytest, AWS CloudWatch.

Week 12: Documentation, Community Feedback, and Finalization

- Document the entire system, including setup instructions, API usage, and troubleshooting guides.
- Gather feedback from the Music Blocks community through usability testing.
- Refine the system based on community feedback and finalize the project.

Deliverable: Finalized debugger with comprehensive documentation.

Availability

I plan to dedicate 40-50 hours per week to the project, with peak availability between Wednesday and Sunday from 8 AM to 7 PM IST. During the community bonding period (10 May - 30 May), I will have my End-Semester exams and will be able to contribute 2-3 hours daily. I will ensure consistent and focused efforts to meet all project milestones and deliverables.

Progress Report

I will be updating the mentors daily on Matrix chat and demonstrating my work through biweekly meetings. Additionally, I will write a blog every week on https://medium.com/ about my progress and share it on my social media profiles.

Medium: Om Santosh Suneri

Post GSoC Plans

After the successful completion of the GSoC 2025 project, I plan to continue contributing to Music Blocks by focusing on testing and quality assurance. I will work on implementing Jest and Cypress testing frameworks to ensure the stability and reliability of the platform. Additionally, I aim to guide newcomers to the community, helping them understand the project, set up their development environment, and make their first contributions. By mentoring new contributors, I hope to foster a collaborative and inclusive environment that encourages growth and innovation.

Beyond testing and community support, I am committed to enhancing Music Blocks by improving existing features and introducing new functionalities. I will work on optimizing the user experience, adding new tools for music creation, and exploring innovative features that align with the community's vision. My long-term goal is to make Music Blocks a more powerful, user-friendly, and educational platform, empowering users worldwide to explore the creative possibilities of coding and music. I am excited to continue my journey with Sugar Labs and contribute to its mission of promoting learning through technology.

Conclusion

Thank you for reading. I have provided a detailed overview of my project and how I plan to execute it. For GSoC 2025, my main goal is to further enhance my understanding of the project by building on my practical experience and research.

As for the technology stack, I am proficient in all the necessary technologies required for this project, including JavaScript, Python, Hugging Face Transformers, FastAPI, AWS, and Docker. With extensive experience in these tools and a strong foundation in open-source development, I am confident in my ability to deliver this project within the given timeline. I take full responsibility for implementing the AI-powered debugger in Music Blocks, a feature that will elevate the platform to new heights by enhancing user experience, fostering creativity, and promoting problem-solving skills. My commitment to excellence and passion for innovation will ensure the successful completion of this project, making Music Blocks a more powerful and accessible tool for users worldwide.