## **Sugar Labs**

### **Project Information:**

**Name**: **AI-powered Debugger for Music Blocks**

**Length**: 350 Hours (Large - 22 weeks)

**Mentor**: [Walter Bender](https://github.com/walterbender/) [Sumit Srivastava](https://github.com/sum2it)

**Assisting Mentor:** [Devin Ulibarri](https://github.com/pikurasa/)

### **Student Details:**

**Full Name:** Sneha Ravi Poojary

**Email:** [snehapoojary2004@gmail.com](mailto:snehapoojary2004@gmail.com)

**GitHub:** <https://github.com/SnehaPoojary20>

**LinkedIn:** [www.linkedin.com/in/snehapoojary2020](http://www.linkedin.com/in/snehapoojary2020)

**Preferred Language:** I’m proficient in English for communication, both spoken and written

**Location:** Mumbai (Maharashtra), India

**Time Zone:** Indian Standard Time (IST) (UTC +05:30)

**Phone Number:** +91 9356781788

**Institution:** Mumbai University

**Program:** B.E in Computer Engineering

**Stage of completion:** 2nd Year (expected June 2026)

#### **About Me:**

I have always been fascinated by technology and its ability to solve real-world problems. My journey into programming started in college, where I initially explored competitive programming before discovering my true passion for web development and machine learning. I enjoy building scalable, user-centric applications and have worked on projects integrating React for frontend and FastAPI with Firebase and MERN stack for backend development.

My interest in AI led me to develop a collaborative filtering recommendation model for travel destinations, incorporating Scikit-learn, Matrix Factorization (SVD), and Google Places API. I also actively participate in coding competitions and hackathons, constantly pushing my problem-solving skills.

Beyond development, I love learning in public and sharing my experiences. I am always eager to contribute to impactful projects, collaborate with like-minded individuals, and explore innovative technologies to create meaningful solutions.

**My Skills**

Languages: Java, JavaScript, Python

Frontend: HTML, CSS, JavaScript ,Bootstrap, Tailwind CSS , React, Redux

Backend : Node.js , Express.js ,Next.js,Fastapi

Database:Mongoose, MongoDb, MySQL

Tools : Git , Github , VS Code

Other Skills : RESTful APIs, MVC, MVT, NLP

**Reason for Choosing Sugar Labs:**

I chose Sugar Labs because my background in AI and ML, combined with my passion for music, makes me excited to contribute. By enhancing Music Blocks, I aim to bridge technology and creativity. Whether I am stressed or simply looking to relax, listening to music helps me find calmness and focus. Music Blocks, a Sugar Labs project, provides an engaging way for users to explore music through coding. By enhancing it with AI-powered debugging, I want to make the experience smoother and more accessible for users of all skill levels. This project excites me because it aligns with my passion for both technology and music, allowing me to contribute meaningfully to an open-source community that promotes creative learning.

The AI-powered Debugger for Music Blocks aims to enhance the platform’s usability by helping users troubleshoot issues in their projects, understand block functionalities, and get creative suggestions in real-time. The debugger will leverage an open-source Large Language Model (LLM) trained on Music Blocks' documentation and user interactions. It will use Retrieval-Augmented Generation (RAG) to improve contextual responses, ensuring that the AI assistant provides accurate and relevant support.

Key features include:

1. Real-time debugging of Music Blocks projects.
2. Interactive chatbot for answering user queries.
3. Feature explanations and creative music composition tips.
4. Seamless integration with the Music Blocks UI.
5. API-based architecture for efficient deployment.
6. Minimizing AI hallucinations for better accuracy.

**My Projects**

### **QueueSphere :** QueueSphere is a smart queue management system that optimizes waiting times by providing real-time queue tracking, automated ticketing, and AI-driven predictions for estimated wait times.

Features:

* Real-time queue tracking and updates.
* AI-based wait time prediction.
* Digital ticketing system with notifications.
* Analytics dashboard for business insights.

Technology Used:

* Frontend: React.js, TailwindCSS
* Backend: Node.js, Express.js
* Database: MongoDB

What Makes It Stand Out? QueueSphere stands out because of its AI-powered predictive analysis, ensuring reduced wait times and better customer satisfaction compared to traditional queue systems.

Github link: <https://github.com/SnehaPoojary20/QueueSphere>

### **VoyageVista :** VoyageVista is a personalized AI-powered travel planner that suggests travel destinations based on user preferences and collaborative filtering.

Features:

* AI-based destination recommendations.
* Integration with Google Places API.
* Personalized itinerary generation.
* Collaborative filtering recommendation engine.

Technology Used:

* Frontend: React.js
* Backend: FastAPI, Firebase
* Database: MongoDB

What Makes It Stand Out? VoyageVista differentiates itself by using a collaborative filtering recommendation model, making travel planning highly personalized and dynamic compared to static itinerary generators.

Github link: <https://github.com/SnehaPoojary20/VoyageVista>

### **PharmaGuard :** PharmaGuard is a medication tracking and authentication system that helps users verify medicine authenticity and get reminders for their medication schedules.

Features:

* Medicine authenticity verification via barcode scanning.
* AI-powered medication reminders.
* Secure storage of prescription details.
* User alerts for expired medications.

Technology Used:

* Frontend: React.js
* Backend: FastAPI with Firebase
* Database: Firebase Firestore

What Makes It Stand Out? PharmaGuard stands out because of its real-time medicine authentication system, ensuring safety and preventing counterfeit drugs, unlike standard medicine reminder apps.

Github link: <https://github.com/SnehaPoojary20/PharmaGuard>

Here is my resume:

<https://docs.google.com/document/d/1PhjqWoN8QvADHZuS1DgMtUaSyhj9kEmr5DeIl9-yZqs/edit?usp=sharing>

### 

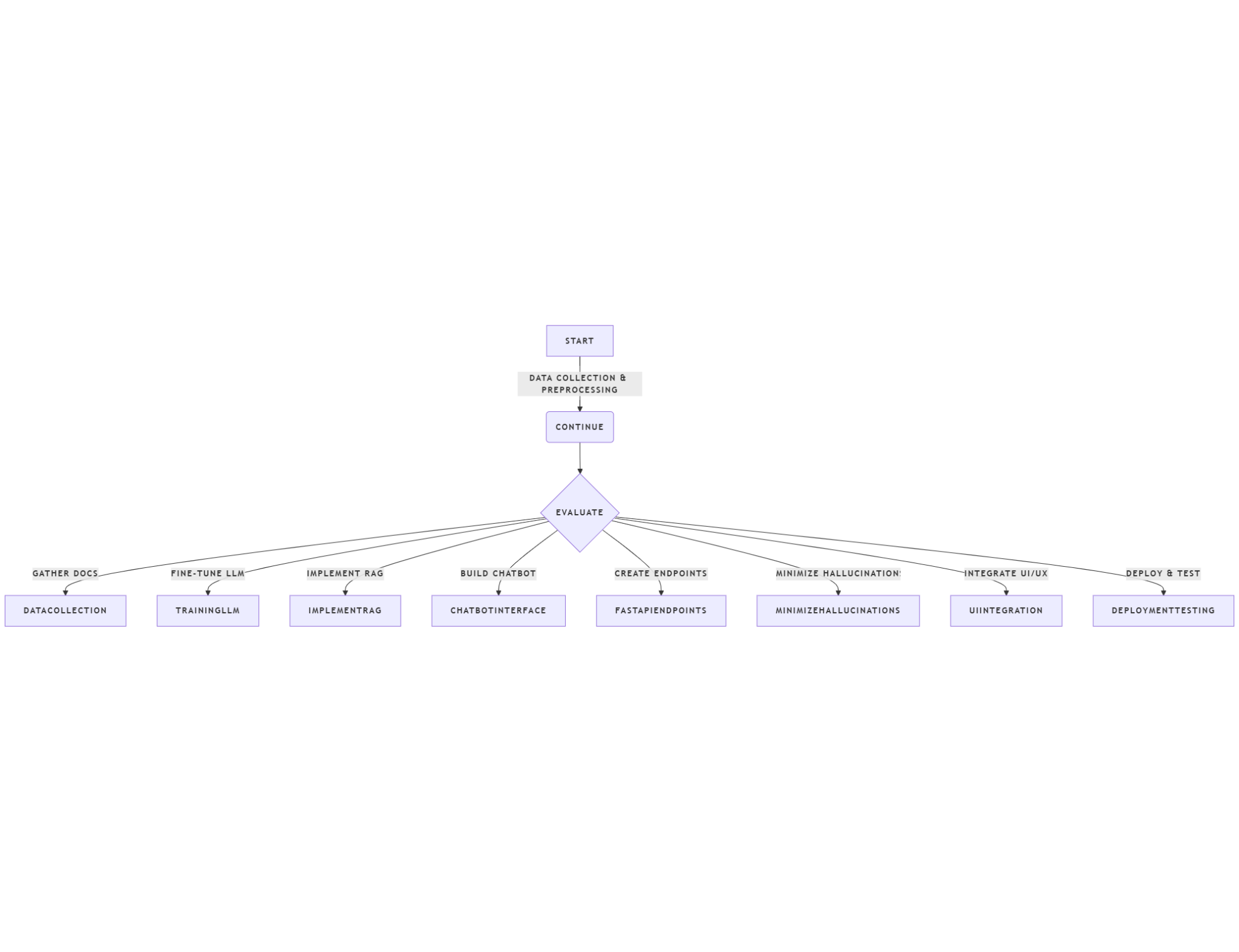
### **AI-Powered Debugger for Music Blocks**

This project aims to integrate an AI-powered debugger into Music Blocks to help users quickly identify and fix errors in their programs. The debugger will highlight incorrect block connections, suggest corrections, and provide real-time explanations.

To achieve this, I will fine-tune an existing open-source LLM and enhance it with Retrieval-Augmented Generation (RAG) for better contextual understanding. The AI assistant will be accessible as an interactive side panel or overlay, ensuring smooth integration with the platform. By implementing confidence scoring and rule-based fallback mechanisms, I will minimize AI errors and improve accuracy.

**Detailed Solution:**

1. Data Collection & Preprocessing:
   * Gathering Music Blocks documentation, FAQs, and common user issues.
   * Extracting structured information for better training.
   * Technology Used: Python, BeautifulSoup (for web scraping if needed), Pandas, OpenAI’s tiktoken (for tokenization).
2. Training an Open-Source LLM:
   * Fine-tuning an open-source LLM (e.g., Mistral-7B, Llama 3) on Music Blocks' data.
   * Implementing supervised fine-tuning with relevant debugging examples.
   * Technology Used: PyTorch, Hugging Face Transformers, PEFT (Parameter-Efficient Fine-Tuning), LoRA (Low-Rank Adaptation).
3. Retrieval-Augmented Generation (RAG) Implementation:
   * Setting up a vector database to retrieve relevant Music Blocks documentation.
   * Using embeddings to improve LLM’s contextual understanding.
   * Technology Used: FAISS (Facebook AI Similarity Search), ChromaDB, LangChain.
4. AI Chatbot and Debugging Interface:
   * Building a chatbot interface within the Music Blocks platform.
   * Providing interactive debugging assistance with suggested solutions.
   * Technology Used: React (for UI), TailwindCSS, FastAPI (for backend API handling).
5. FastAPI Endpoints for AI Model Integration:
   * Creating endpoints to communicate between Music Blocks and the AI model.
   * Ensuring efficient query handling and low-latency responses.
   * Technology Used: FastAPI, Uvicorn, Pydantic.
6. Minimizing Hallucinations & Improving Accuracy:
   * Implementing confidence scoring for AI responses.
   * Will use response verification techniques like self-consistency checks.
   * Technology Used: LlamaIndex, RLHF (Reinforcement Learning from Human Feedback).
7. Seamless UI/UX Integration:
   * Designing an intuitive chatbot interface within Music Blocks.
   * Ensuring accessibility features for a diverse user base.
   * Technology Used: React.js, Zustand (for state management), Material UI.
8. Deployment & Testing:
   * Deploying the model on AWS Lambda or Google Cloud Functions.
   * Conducting user testing to refine accuracy and user experience.
   * Technology Used: AWS S3, EC2, Cloud Run, Docker.



**User Experience (UX) & Validation**

The AI-powered debugger will be integrated into the Music Blocks UI as a **side panel or floating chatbot**, providing real-time debugging suggestions when users encounter errors in their block-based code.

* **Error Detection & Highlighting**: The debugger will identify incorrect block structures and highlight them.
* **Interactive Suggestions**: It will suggest **corrected block arrangements** or explanations of mistakes.
* **Chat-Based Assistance**: Users can ask questions, and the debugger will provide relevant solutions.

### **Defining the Debugger’s UX & Integration**

The AI-powered debugger will be seamlessly integrated into the Music Blocks UI as an interactive side panel or an overlay, ensuring it remains accessible without disrupting the workflow.

#### Debugger Features:

* Real-time error detection – Highlights incorrect block connections or missing dependencies.
* Contextual debugging suggestions – Provides recommendations to fix issues with explanations.
* Interactive chatbot assistance – Users can ask questions about blocks and receive AI-powered explanations.

This integration will enhance accessibility for beginners while allowing advanced users to debug efficiently without manually inspecting complex structures.

### **Minimizing AI Hallucinations**

To ensure accurate debugging and explanations, I will implement:

✔ Confidence scoring – The AI will only provide suggestions when it meets a certain accuracy threshold.

✔ Threshold-based filtering – If the AI is uncertain, it will default to rule-based debugging rather than making unreliable suggestions.

✔ Self-consistency checks – Multiple responses will be evaluated before presenting the most accurate suggestion.

By combining AI reasoning with predefined debugging rules, we can minimize hallucinations and improve the overall reliability of the system.

**Usability and effectiveness:**

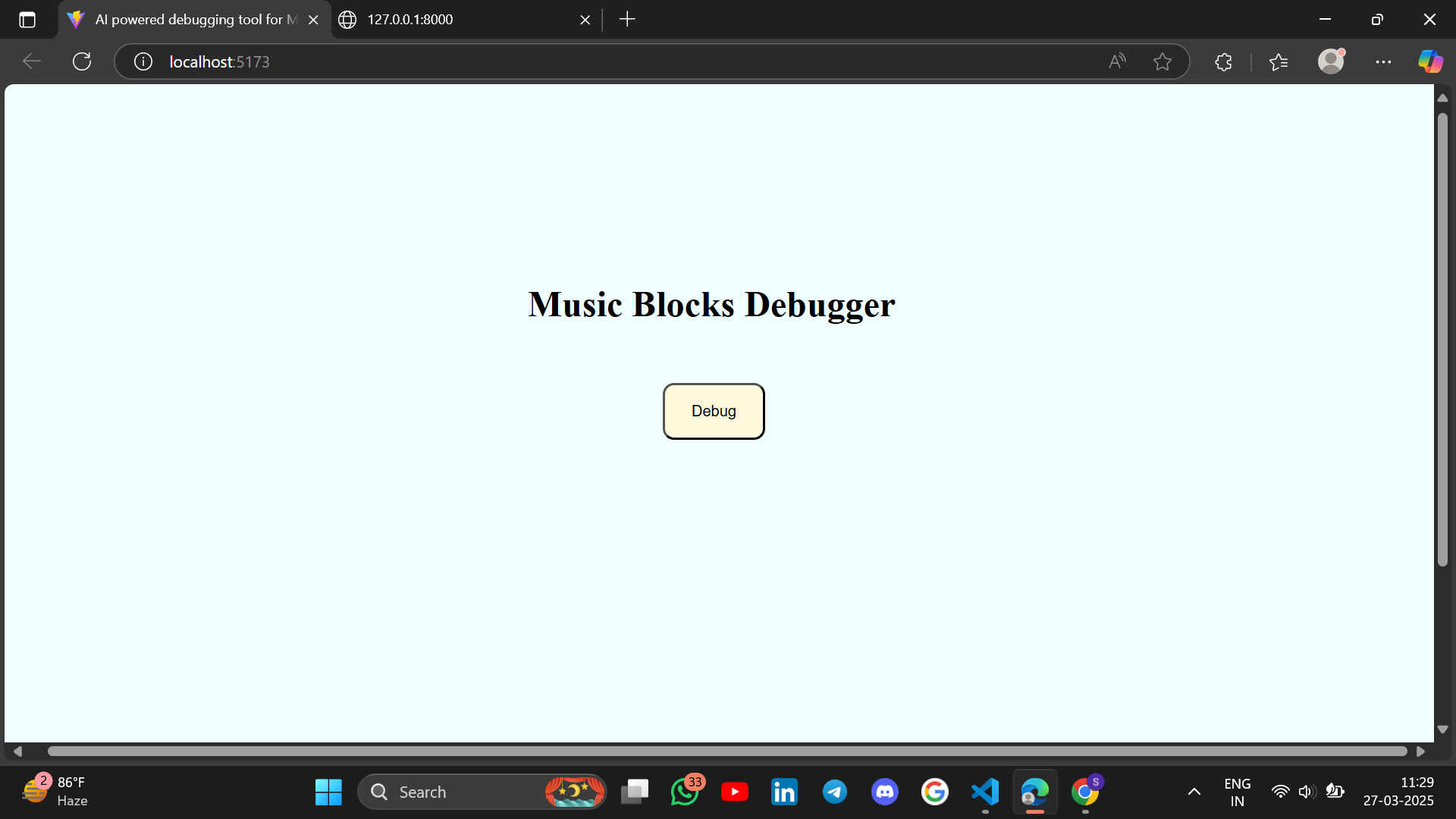
To ensure usability and effectiveness, I will validate the debugger through:

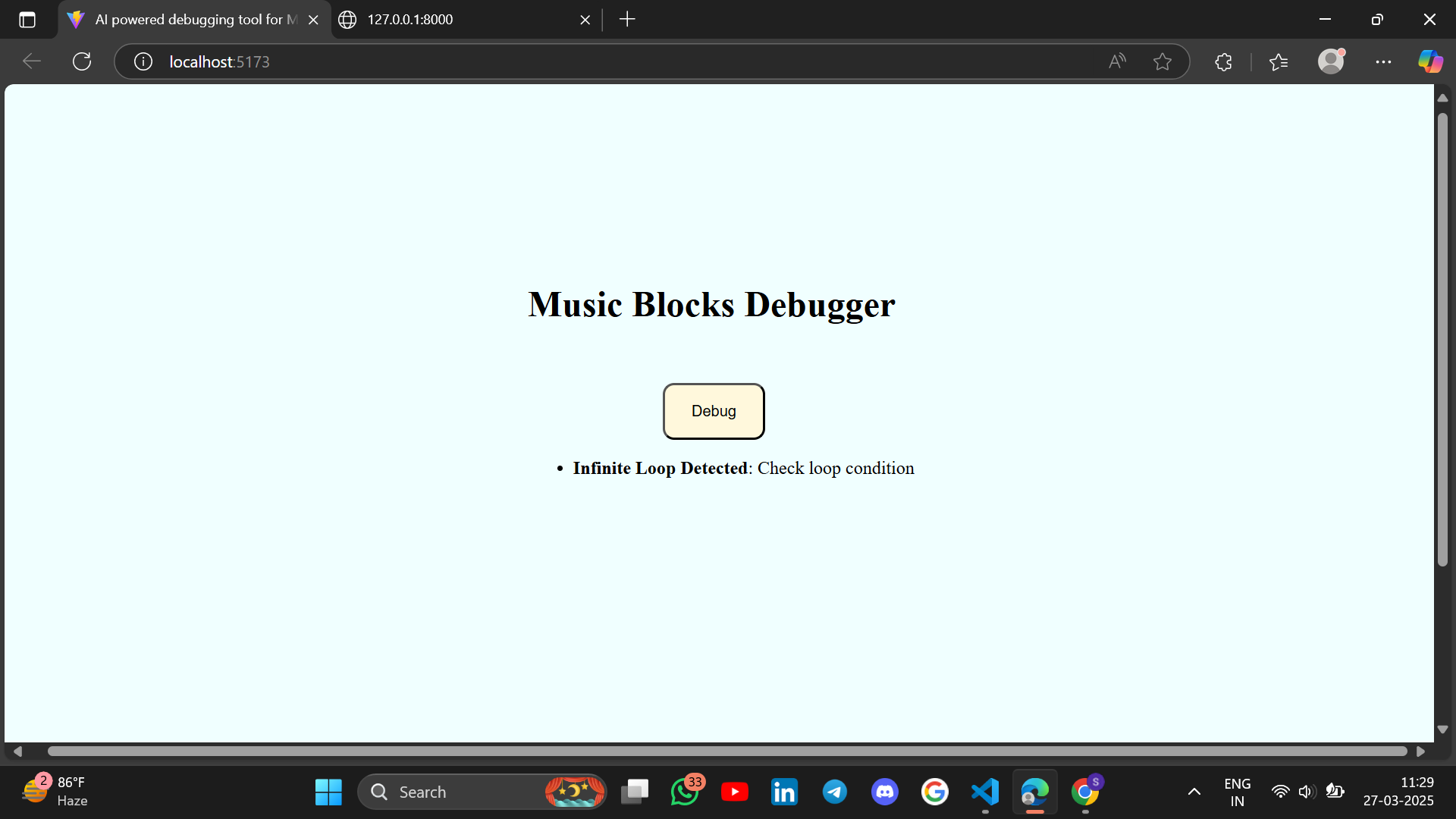
1. User Testing  
   * Internal Testing (Weeks 9-10): Testing with predefined errors.
   * Beta Testing (Weeks 11-12): The feature will be tested by Sugar Labs contributors, mentors, and Music Blocks users to gather real-world feedback and identify any usability or performance issues.
2. User Feedback Collection  
   * Thumbs-up/down rating system for AI-generated suggestions.
   * Surveys & Feedback Forms to gather user opinions.
3. Defining Success Metrics  
   * 80% accuracy in debugging suggestions.
   * 60% user engagement (users interacting with suggestions).
   * 70% positive feedback from Music Blocks users.

**Prototype AI Debugger: Proof of Concept**

This is a basic prototype demonstrating an AI-powered debugging system for Music Blocks. It identifies common errors and suggests corrections based on predefined rules.

However, this is only a proof of concept; the final implementation will have a completely different UI, fully integrated with Music Blocks for a seamless user experience





Github Link: <https://github.com/SnehaPoojary20/AI-Powered-Debugger-for-Music-Blocks-.git>

### **🔹 What the Debugger Does**

1. **Receives a List of Blocks as Input**
   * The frontend sends a request with an array of blocks (representing a Music Blocks program).
   * Example input:

| {  "blocks": [  {"type": "loop", "condition": "forever"},  {"type": "play", "note": "C"}  ]  } |
| --- |

**Analyzes the Blocks for Errors**

* Currently, it only checks for **infinite loops** (i.e., if a loop has a "forever" condition).
* It can be expanded it to detect other issues like:  
  + Unconnected blocks
  + Incorrect block types
  + Syntax errors in input values

**Returns Debugging Suggestions**

* If an infinite loop is found, it sends back a **suggestion to fix it**.
* Example output:

| {  "errors": [  {  "error": "Infinite Loop Detected",  "suggestion": "Check loop condition"  }  ]  } |
| --- |

The React frontend can display this message to the user.

**Prototype Features**:

🔹 Accepts a sample **Music Blocks** program as input.

🔹 Identifies **common errors** (e.g., missing connections, infinite loops).

🔹 Suggests corrections based on **predefined rules** or a simple AI model.

**Impact on Sugar Labs**:

1. Enhanced User Experience:
   * Users will get real-time assistance in debugging and understanding Music Blocks, leading to an improved learning curve.
2. Increased Engagement:
   * AI-powered suggestions will inspire users to explore new musical patterns and compositions.
3. Reduced Support Overhead:
   * Community maintainers will have fewer repetitive queries, allowing them to focus on core development.
4. Scalability & Future Integrations:
   * The AI-powered debugger can serve as a foundation for future AI enhancements in Sugar Labs projects.

**Timeline**

I have broken down the implementation into clear milestones to ensure practical feasibility within 12 weeks.

#### **Week 1-2: Data Collection & Preprocessing**

* Collecting and preprocessing Music Blocks documentation, FAQs, and common debugging issues.
* Converting unstructured data into structured formats for efficient retrieval.
* Use of Python, BeautifulSoup (for scraping, if needed), Pandas for data structuring, and OpenAI’s tiktoken for tokenization.

**Deliverable:** A cleaned dataset ready for training the AI model.

#### **Week 3-4: Model Training & Fine-Tuning**

* Fine-tuning an open-source LLM (e.g., Mistral-7B, Llama 3) on Music Blocks’ debugging data.
* Training the model using supervised fine-tuning with relevant debugging examples.
* Optimizing performance by adjusting hyperparameters and using Parameter-Efficient Fine-Tuning (PEFT).

**Deliverable:** A fine-tuned LLM capable of understanding debugging queries.

#### **Week 5-6: Implementing RAG (Retrieval-Augmented Generation) & Vector Database**

* Setting up FAISS/ChromaDB for efficient retrieval of relevant Music Blocks documentation.
* Implementing embedding-based search for better contextual understanding.
* Integrating retrieval with the LLM to ensure fact-based responses.

**Deliverable:** A retrieval-enhanced AI model ready for chatbot integration.

#### **Week 7-8: Developing the AI Chatbot & Debugging Interface**

* Building a chatbot interface within Music Blocks.
* Developing interactive debugging assistance for users.
* Setting up a FastAPI backend to handle model queries efficiently.
* Integrating Zustand for state management in the React UI.

**Deliverable:** A basic working chatbot interface integrated into Music Blocks.

#### **Week 9-10: Optimizing AI Accuracy & Minimizing Hallucinations**

* Implementing confidence scoring for AI responses.
* Introducing response filtering techniques like self-consistency checks.
* Conducting internal testing with edge cases to refine AI accuracy.

**Deliverable:** A more reliable debugging assistant with minimal hallucinations.

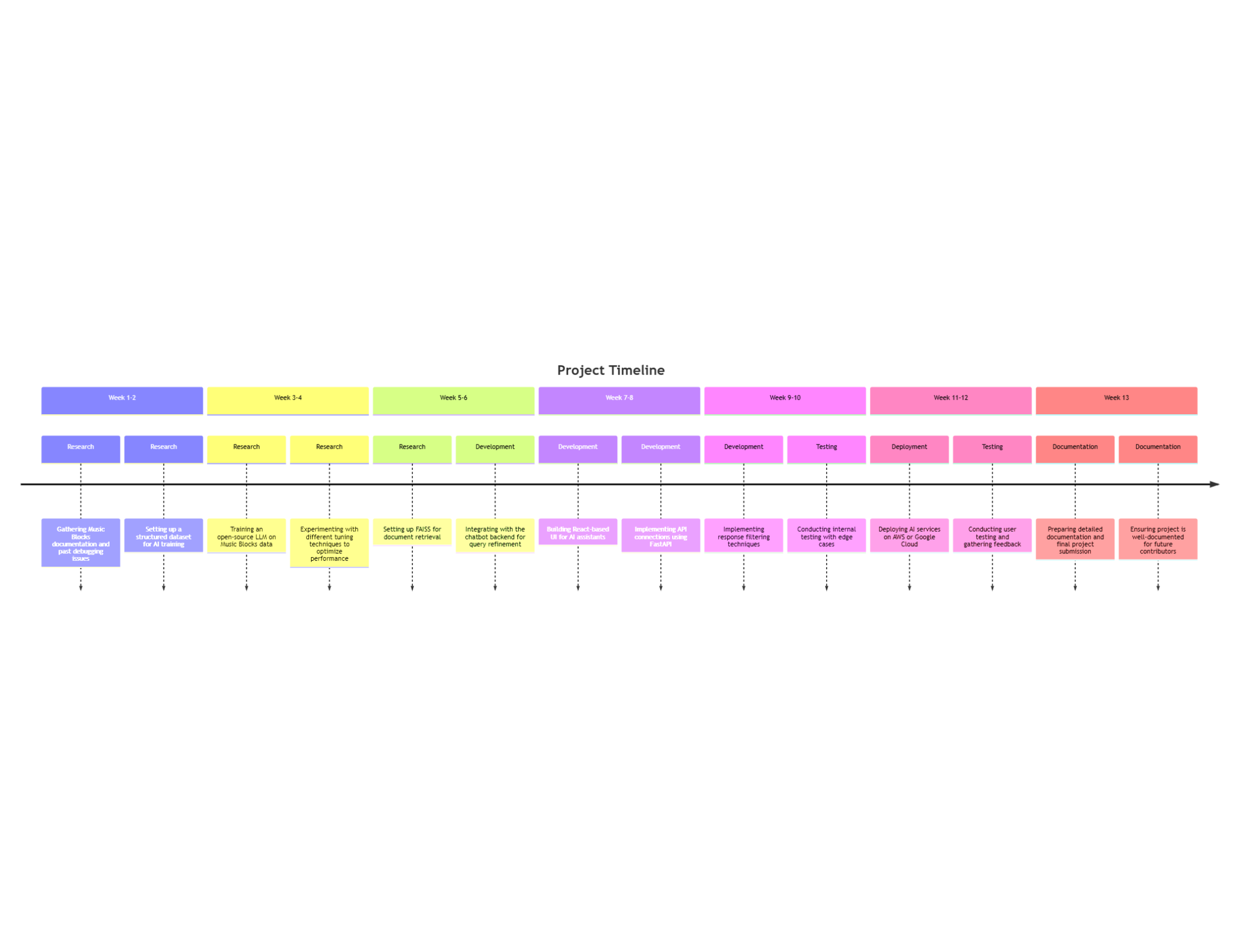
#### **Week 11-12: Final Integration, Testing & Deployment**

* Deploying AI services on AWS Lambda or Google Cloud for scalability.
* Conducting real-user testing with the Sugar Labs community and collect feedback.
* Implementing final optimizations for performance and latency reduction.

**Deliverable:** Fully functional AI-powered debugger, tested and deployed for real-world use.

#### **Week 13: Documentation & Final Submission**

* Preparing detailed documentation for future contributors.
* Submiting the final project to Sugar Labs.



**Deliverable:** Well-documented AI Debugger integrated into Music Blocks, ready for long-term use.

By following this structured approach, I will ensure that the AI-powered debugger is successfully implemented within the GSoC timeline while also making it scalable and accessible for a diverse user base.

### **Clarifying Feasibility**

To ensure this project is achievable within the **GSoC timeline**, I will focus on **fine-tuning an existing open-source LLM** rather than training one from scratch. Initially, I will experiment with models such as **Mistral-7B** to determine if fine-tuning is necessary or if prompt engineering can provide accurate results.

For efficiency, I will integrate **Retrieval-Augmented Generation (RAG)** to enhance the model's understanding of Music Blocks without requiring extensive retraining. If computational constraints arise, I will use an **API-based approach initially** while planning for a lightweight, self-hosted solution.

#### **Scalability: Expanding AI Debugger Beyond Music Blocks**

* The AI-powered debugging framework can be extended to other Sugar Labs projects (e.g., TurtleBlocks, Etoys) by fine-tuning the model on different datasets.
* The API-based design ensures that new Sugar Labs tools can integrate the AI assistant easily.
* Future contributors can enhance the debugger with multi-modal capabilities (e.g., voice-based debugging for accessibility).

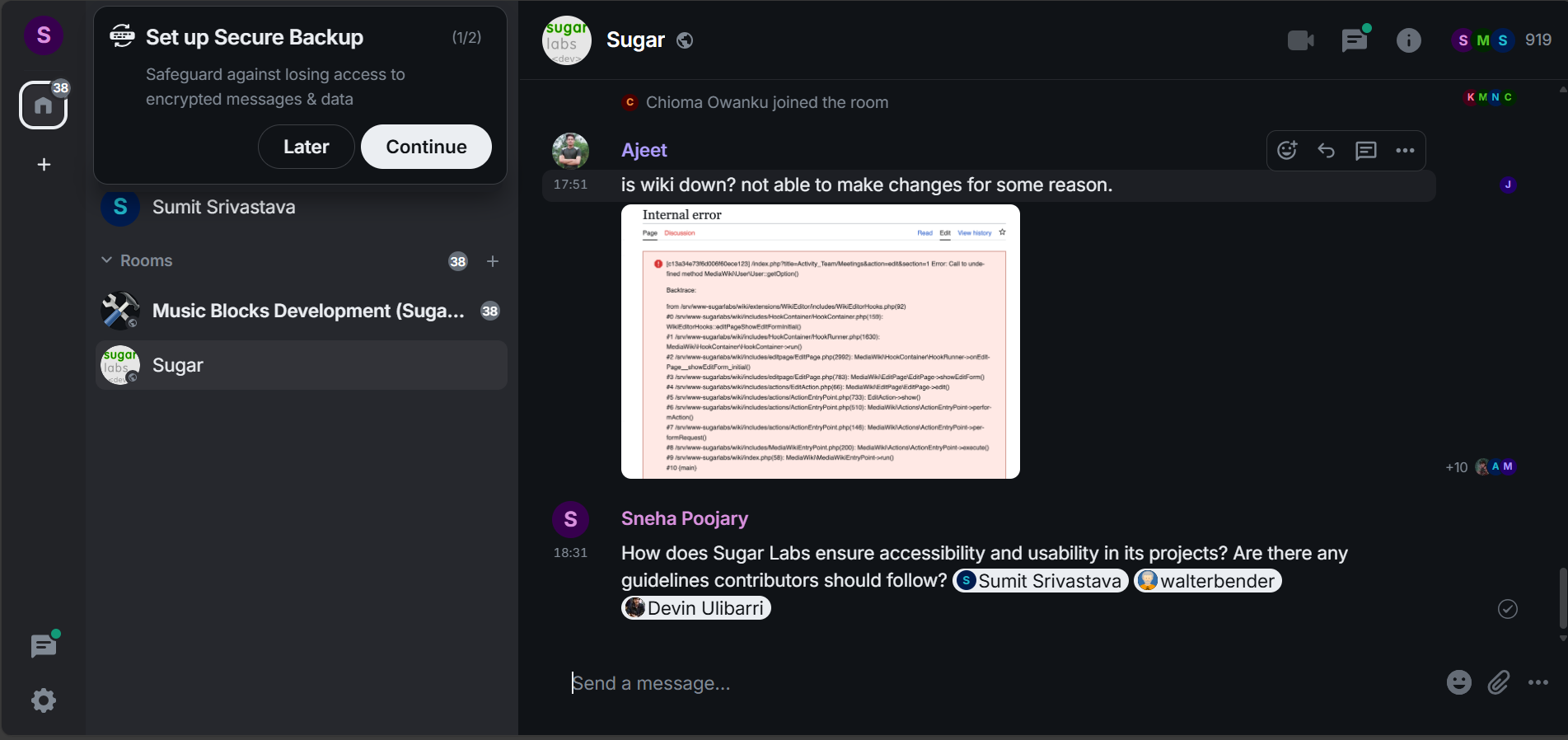
#### **Accessibility: Making Debugging Inclusive**

* **Multi-Language Support:** The model can be fine-tuned with multilingual datasets to assist non-English users.
* **Simplified Explanations:** AI can provide step-by-step debugging guidance in simpler language for beginners.
* **Keyboard Navigation & Voice Input:** Ensuring screen-reader compatibility and possible voice-input interactions for greater usability.

**My Contributions for SugarLabs:**

| I investigated and participated in debugging the Browse activity issue (**#987**). I analyzed possible causes and engaged with the Sugar Labs community to explore solutions  Link:<https://github.com/sugarlabs/sugar/issues/987> |
| --- |
| I have improved the tutorial descriptions for the Abacus activity. The changes make it easier for beginners and educators to understand (#837)  Link:<https://github.com/sugarlabs/sugarizer/pull/12> |
|  |

**My presence in community:**

****

### **Availability:**

I plan to dedicate 35-50 hours per week to the project and will be most active between Saturday and Sunday from 9 AM to 8 PM IST. I will have my End-semester exams between 13 May - 31 May i.e. the community bonding period and will be able to dedicate 2-3 hours a day during that period.

### 

### 

### **Summary**

During the GSoC 2025 period, I will develop an AI-powered Debugger for Music Blocks, enhancing the platform’s usability by providing real-time debugging assistance, interactive explanations, and creative composition suggestions.

The project will involve:

* Training an Open-Source LLM (e.g., Mistral-7B, Llama 3) on Music Blocks documentation and common debugging issues.
* Implementing Retrieval-Augmented Generation (RAG) to improve AI responses using a vector database (FAISS/ChromaDB).
* Developing an AI Chatbot Interface integrated within Music Blocks for user-friendly debugging.
* Optimizing AI Accuracy by minimizing hallucinations using response filtering techniques.
* Seamless UI/UX Integration to ensure smooth interaction with the debugging assistant.
* Deployment & Testing on cloud platforms like AWS or Google Cloud for scalability.

This solution will enhance Music Blocks by making troubleshooting easier, improving user engagement, and reducing the support overhead for maintainers.

**Conclusion:**

I am fully committed to contributing to Sugar Labs through this AI-powered Debugger for Music Blocks. This project will significantly enhance the platform’s usability by making debugging more intuitive, reducing barriers for beginners, and fostering creativity. By integrating AI effectively, I aim to empower users to explore music programming without frustration. I am dedicated to completing this project with high-quality implementation before the GSoC 2025 deadline and ensuring it delivers long-term value to the Sugar Labs community.