## 

### **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.

**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: Feature Description and Feasibility Plan**

### The goal of this project is to enhance Music Blocks with an AI-powered debugger that provides real-time assistance to users. This feature aims to bridge the gap between a user's creative intent and their ability to troubleshoot, understand, and optimize their block-based musical compositions. The AI debugger will be accessible directly from the Music Blocks interface and will assist by answering questions, explaining block behavior, identifying issues, and offering creative suggestions.

### To ensure feasibility within the GSoC timeframe of 350 hours, I will focus on the following carefully scoped features:

### **✅ 1. Pre-trained LLM Integration (Avoiding Training from Scratch)**

### Rather than training a large language model from scratch, which would require substantial compute resources, data, and time.I will utilize a lightweight, open-source pre-trained LLM such as TinyLlama, GPT-Neo, Mistral, Phi-2, or OpenChat.

### These models will be adapted to understand Music Blocks projects through:

### **Prompt engineering**

### **Fine-tuning on a small, curated dataset** of Music Block errors and corrections

### **Retrieval-Augmented Generation (RAG)** to enhance contextual accuracy by fetching relevant documentation or examples dynamically

### This approach will significantly reduce development time while ensuring effective and contextually rich AI responses.

### **✅ 2. JSON-Based Debugging Instead of Full Graphical or Music Theory Analysis**

### Rather than attempting complex graphical analysis or full music theory interpretation—which are time-consuming and beyond the scope of GSoC—I will:

### Focus on parsing the **JSON representation of Music Blocks projects**

### Develop a mapping of block types and their expected behavior

### Implement logic to detect common structural issues, such as:

### Blocks not connected

### Invalid parameters (e.g., tempo values)

### Redundant or repetitive patterns

### The AI will generate explanations and suggestions based on these observations, supporting beginner users and assisting advanced users in optimizing their compositions.

### **✅ 3. Contextual Block Explanation and Suggestion System**

### Users will be able to interact with the debugger via natural language prompts, such as:

### *"What does this block do?"*

### *"Why isn't my melody repeating?"*

### *"Suggest improvements to this rhythm pattern."*

### The AI will respond by:

### Explaining block functions and their placement in the sequence

### Providing suggestions like:

### "Consider changing the tempo to match the rhythm"

### "Add a repeat block here to simplify looping"

### Offering **basic project summarization**, i.e., "This program plays a four-note melody with a fast rhythm and looped structure."

### **✅ 4. Visual Cues and Error Highlights (Time-Permitting Bonus)**

### If time permits, I will implement a basic system to highlight problematic blocks directly within the Music Blocks UI using:

### Block color changes (e.g., red for potential errors)

### Tooltips with AI-generated suggestions

### This will enhance the debugging experience without requiring full graphical analysis.

### **🎯 Core Features of the AI-Powered Debugger**

#### **1. Pre-trained LLM Integration**

#### **2. JSON-Based Project Analysis**

#### **3. Natural Language Interaction**

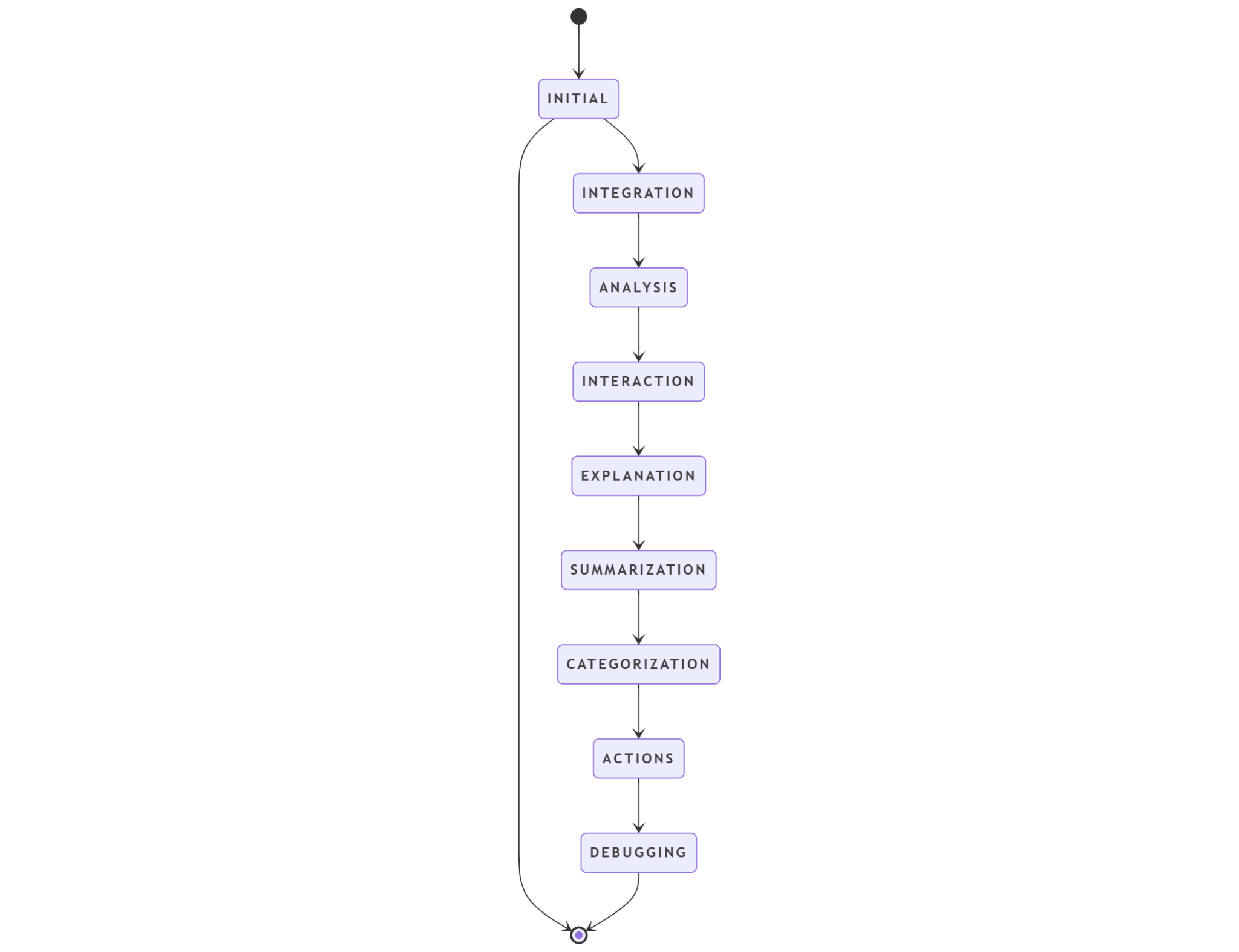
#### **4. Block Explanation and Suggestions**

#### **5. Project Summarization**

#### **6. Basic Error Categorization**

#### **7. Prompt Buttons / Quick Actions**

#### **8. Visual Debugging Support (Optional, Time-Permitting)**



### **🔍 Why This Scope is Feasible**

### To remain within the 350-hour constraint, I am **avoiding:**

### Training large models from scratch

### Full music theory interpretation

### Complex visual graph parsing of block connections in early phases

### Instead, I’m focusing on:

### Lightweight LLMs with contextual enhancement (RAG)

### JSON-based structural analysis

### Textual AI debugging and explanation

### Modular development that allows optional UI enhancement in later stages

### This ensures that the project remains impactful, technically achievable, and useful to both new and experienced Music Blocks users.

This proposal builds on my real-world experience using FastAPI, React, and AI models, making it technically feasible within the GSoC timeframe

**User Experience**

User Scenario : A user builds a melody using Music Blocks and runs into a problem, the music is not looping. They click the **AI Debugger** button and type ‘Why isn’t my melody repeating?’. The **AI parses the project JSON**, detects that no Repeat block is used, and replies: ‘Add a Repeat block after the melody block to enable looping.

**How it will be done** :

Music Blocks allows exporting projects in a JSON format, which includes all block types, parameters, and connections. I will use this exported JSON as the input to the debugger. Parsing will be done via a custom parser that maps block types to expected behavior.

Example:

{

"blocks": [

{ "type": "note", "pitch": "C4", "duration": "quarter" },

{ "type": "repeat", "count": 4, "contents": [...] }

]

}

### 

**Testing and Validation**

I will test the debugger against 20+ curated Music Blocks projects, introducing intentional errors to evaluate detection accuracy and response relevance. Mentor/community feedback will be integrated to improve suggestions and explanation quality.

**Detailed Timeline and Deliverables**

| **Period** | **GSoC Phase** | **Planned Tasks / Deliverables** |
| --- | --- | --- |
| May 20 – June 16, 2025 | Community Bonding Period | - Engaging with mentors and community  - Studying Music Blocks architecture & JSON structure  - Finalizing dataset for fine-tuning / prompt examples  - Setting Up the development environment |
| June 17 – July 14, 2025 | Phase 1 (Coding Begins) | **Milestone 1:**  🔹 Integrating pre-trained LLM (TinyLlama/GPT-Neo etc.)  🔹 Building FastAPI backend to serve model  🔹 Parsing JSON from Music Blocks and extract block data  🔹 Implementing basic block mapping + error detection (e.g., disconnected blocks, invalid values) |
| July 15 – July 22, 2025 | Phase 1 Evaluation | 🔸 Submitting evaluation  🔸 Demo of working backend + basic debugging with JSON  🔸 Receiving mentor’s feedback and adjusting scope if needed |
| July 23 – August 19, 2025 | Phase 2 | **Milestone 2:**  🔹 Implementing Retrieval-Augmented Generation (RAG)  🔹 Adding natural language QA support  🔹 Generating block-level suggestions (e.g., “Add repeat block”)  🔹 Adding FastAPI endpoints for explanation/suggestions  🔹 Starting frontend integration (simple UI for chatbot) |
| August 20 – August 26, 2025 | Phase 2 Evaluation | 🔸 Submitting second evaluation  🔸 Demo AI debugger answering questions & giving improvement tips  🔸 Highlighting how RAG improves contextual answers |
| August 27 – September 8, 2025 | Finalization & Polishing | **Milestone 3:**  🔹 Adding project summarization feature  🔹 Finalizing visual cue integration (highlighting erroneous blocks — if time permits)  🔹 Writing user documentation  🔹 Testing thoroughly across sample Music Block programs |
| September 9 – September 16, 2025 | GSoC Final Evaluation (Submission) | 🔸 Submitting final report & code  🔸 Ensuring project meets all deliverables  🔸 Preparing deployment instructions/demo video  🔸 Receiving mentor feedback |
| September 17, 2025 | GSoC Results Announced | 🎉 Completion & celebration! |

### **⏳ Estimated Time Allocation**

| **Feature** | **Estimated Hours** |
| --- | --- |
| Pre-trained LLM integration | 30 hours |
| JSON parsing + rule-based debugger | 40 hours |
| Retrieval-Augmented Generation | 45 hours |
| Natural language Q&A | 50 hours |
| Suggestion & explanation engine | 45 hours |
| Frontend integration (chat UI) | 40 hours |
| Visual cues (optional) | 25 hours |
| Testing + Documentation | 30 hours |
| Buffer time | 45 hours |
| **Total** | **350 hours** |

### 

**Project Blog Series**

I will be documenting my weekly progress, milestones, and insights throughout the GSoC journey via blogs on [Dev.to](https://dev.to). This will help keep my mentors and the community updated on the development of the AI-powered debugger feature for Music Blocks.

**👉 Blog Series**

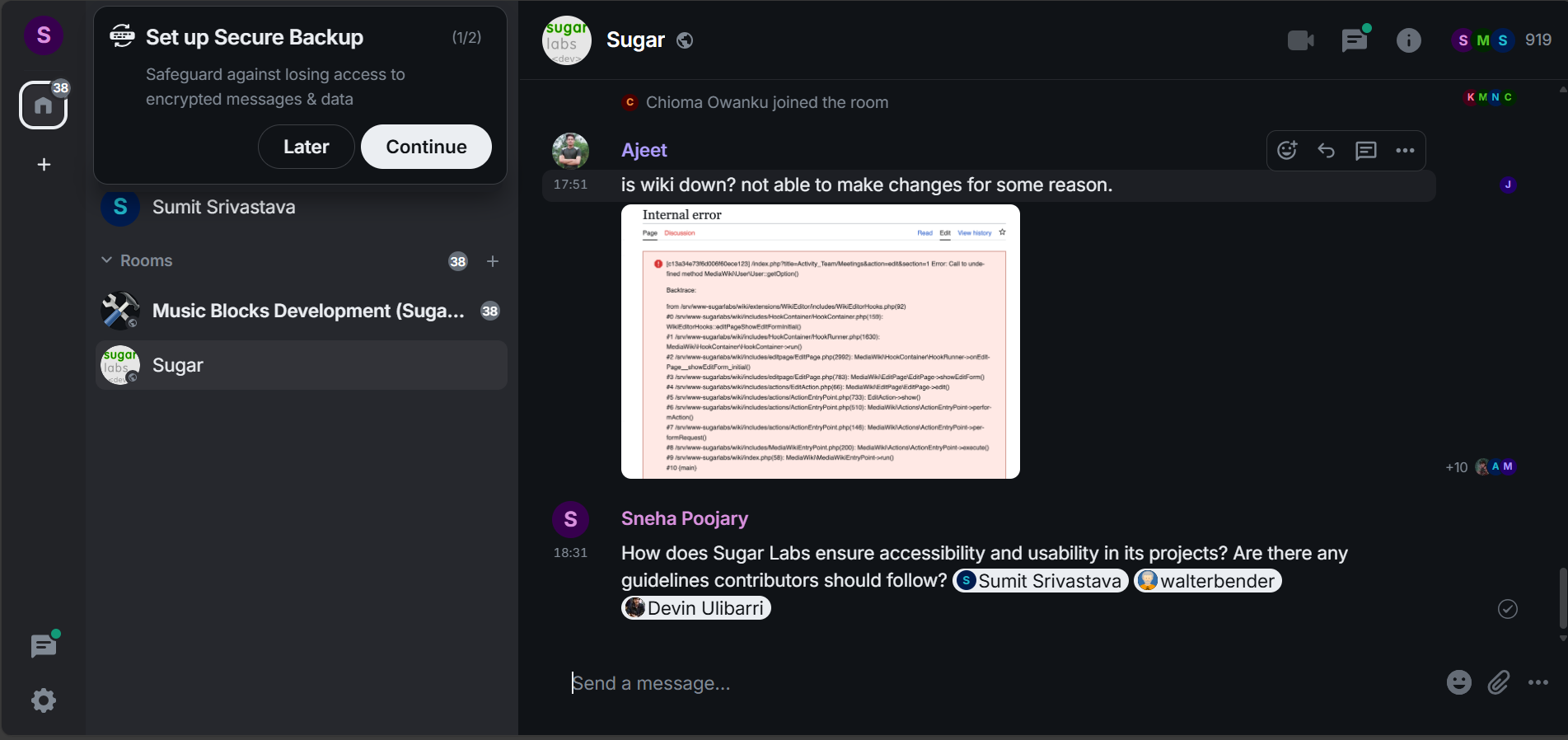
**Link:**[**https://dev.to/sneha\_poojary/ai-powered-debugger-3957**](https://dev.to/sneha_poojary/ai-powered-debugger-3957)

### 

### 

**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> |

****

**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**

I will build an AI-powered debugger for Music Blocks that assists users by analyzing their projects, identifying issues, and providing natural language explanations and creative suggestions. The system will use a lightweight pre-trained LLM with Retrieval-Augmented Generation (RAG) and JSON-based project analysis to make debugging intuitive and accessible for both beginners and advanced users.