AI Code Generation for Lesson Plans and Model Abstraction Layer

Sugar Labs - Google Summer of Code 2025

Personal Details

Full Name: Nideesh Bharath Kumar

Email: bknideesh@gmail.com

First Language: English

GitHub profile link: https://github.com/nb923

Location: New Jersey, United States of America

Time zone: EST (GMT-5)

University name: Rutgers University–New Brunswick

Program you are enrolled in (Degree & Major/Minor): B.S. Computer Science, Artificial Intelligence Track

Year: Junior Year (Third Year)

Expected graduation date: May 2026

Open Source Contributions:

- Kubeflow PR #4055 – Kubeflow pull request
- API Dash PR #688 – API Dash pull request

Sugar Labs Contributions:

- MusicBlocks PR #4569 – MusicBlocks pull request regarding double flat/sharp documentation

My interest and Skills

Why This Project?

I am highly passionate about music; I have played the saxophone for eight years, and I am currently expanding my music knowledge while studying dance, singing, and piano in university. I believe that MusicBlocks offers an exceptional way to learn music, similar to how Hour of Code opened up coding to many children around the world. By using AI code generation for lesson plans, this project can extend this ease of learning beyond prebuilt lesson plans, letting learners explore any music-related topic with ease.

My Experiences

Although I am relatively new to Open Source contributions, as demonstrated by the PRs listed above, I have lots of experience working on large code bases in a fast-paced environment. I'm a junior (third year) studying at Rutgers University–New Brunswick pursuing a B.S. in Computer Science on the Artificial Intelligence Track. I have a strong foundation in full stack development and AI engineering: I have project and internship experience in technologies like: RAG, Vector Databases, LLMs, FastAPI, Docker, Python, JavaScript, TypeScript, LangChain, AWS, Kubernetes, PostgreSQL, and other technologies that aid in developing scalable and AI-powered systems.

I have interned at Manomay Tech, IDEA, and Newark Science and Sustainability, developing scalable systems and managing AI systems; additionally, I've completed fellowships with Google and Codepath, developing my technical skills. I've also won awards in hackathons, achieving Overall Best Project in the CS Base Climate Hackathon for an eco-friendly routing solution and Best Use of Terraform in the HackRU Hackathon for an Computer Vision Smart Shopping Cart.

I have built both MusicBlocks and MusicBlocks-AI locally, tested them, and am actively working on solving issues in the repositories. I believe my skills in AI development and experience with RAG, Vector Databases, and LLMs will put me in a position to effectively contribute to this project.

Resume Link:
https://drive.google.com/file/d/1ajcP2LG4qC4k0HCHrqTacib-cK9jJmfQ/view?usp=sharing

Project Details

What are you making?

I am planning on making the AI Code Generation for Lesson Plans and Model Abstraction Layer project. Specifically, I will be enhancing the already existing lesson plan generator in MusicBlocks-AI and adding code generation capabilities for both block-based and JavaScript code. The broad implementation steps will be the following:

- Clean the A-MAPS and/or The Lakh Dataset (annotated music datasets) and split each large MIDI file into snippets
- Populate a VectorDB (Chroma) with these snippets and add a script for ANN in Python
- Create a FastAPI endpoint for a model-agnostic RAG pipeline (model queries the database through the endpoint if necessary, decided through LangChain)
- Create a script that converts MIDI files into JavaScript and block-based code that is importable
- Populate the current lesson plan VectorDB with lesson plans from Music Educators Toolbox (Carnegie Hall)
- Connect all of this to the frontend Chainlit interface that can use the endpoint to create lesson plans and code
- Create an import tool that can take the generated code and put it into the MusicBlocks environment

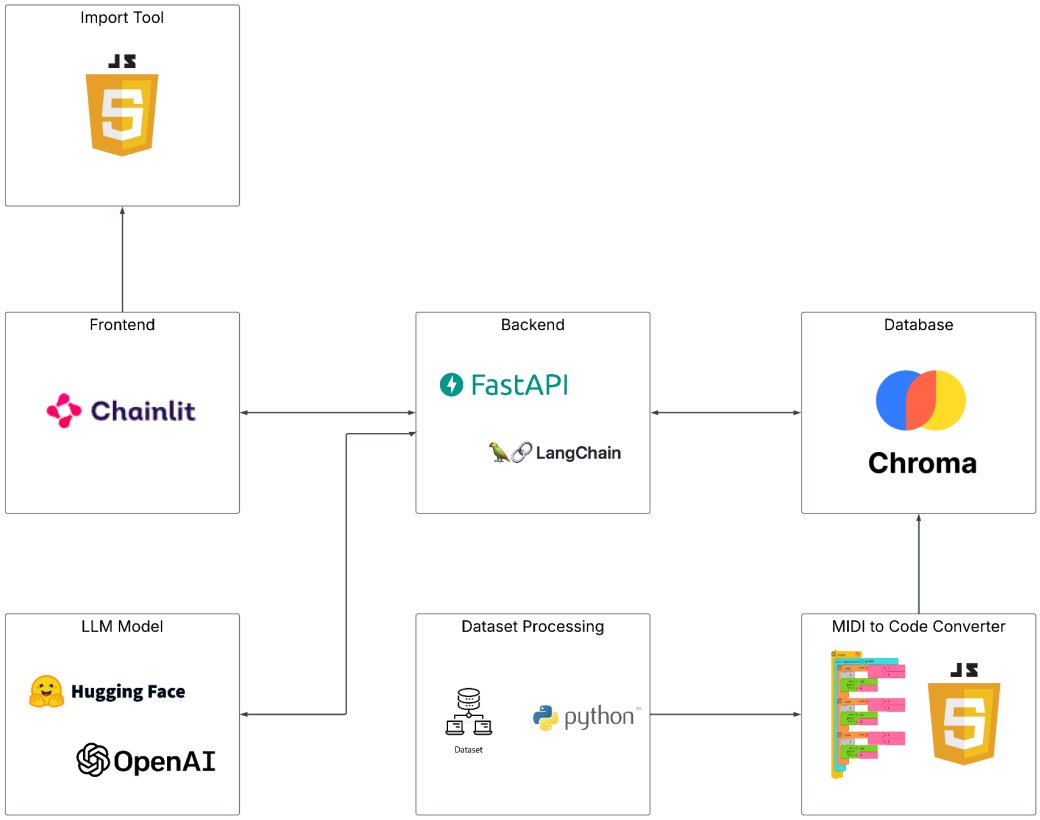
The final product will be the ability for the LLM (model-agnostic) to create more diverse lesson plans and also generate block-based and JavaScript code that can be used in MusicBlocks to supplement the lesson plan with examples.

How will it impact Sugar Labs?

As stated before, Sugar Labs already provides MusicBlocks, an amazing tool to exploring learning music through block-based code. Akin to Hour of Code, it makes learning music more accessible. By providing AI generated lesson plans and code, it allows students to

tailor their learning experience to their individual interests and also have a personal “tutor” to help them through their music learning journey in MusicBlocks.

What technologies (programming languages, etc.) will you be using?



Technologies/Languages Used: Python, JavaScript, Chainlit, FastAPI, LangChain, Hugging Face LLMs/OpenAI LLMs, Chroma DB, and Databases (A-MAPS/The Lakh Dataset)

- The user queries a lesson plan using the Chainlit interface
- The FastAPI backend checks if the query needs any external info from the database
    - If it does, then it queries the database for the necessary resources (lesson plans, code snippets)
- Then the queried files are sent to the LLM along with the prompt to get the response

- The LLM generates lesson plans and code through references provided from the database
- The response is submitted back into Chainlit for the user to view
- And the code can be imported into MusicBlocks using the import tool
- For dataset processing
    - The MIDI files will be split into snippets and then converted into code
    - The lesson plans will populate the lesson plan database after text extraction

Timeline

Pre-GSoC + Community Bonding

During this period, I will introduce myself through the Sugar Labs/MusicBlocks communication channels, make sure my current local builds of MusicBlocks and MusicBlocks-AI are right, explore the A-MAPS and The Lakh Dataset, submit small PRs to get more familiar with the Sugar Labs workflow, and discuss with the mentors and finalize implementation specifics of the project.

Week 1

Process the A-MAPS and/or Lakh dataset, split the MIDI files into snippets, and work on the code conversion tool to convert the MIDI snippets into JavaScript or Block-based code.

Week 2

Finish up working on the code conversion tool and generate embeddings for the snippets and populate a new Chroma DB with the annotated code snippets.

Week 3

Build the FastAPI backend with LangChain to determine if database retrieval is necessary and test the API call to see if it can fetch content from the database if required based on prompt.

Week 4

Add LLM support that is model agnostic to the backend by providing another endpoint for the LLM. This LLM will be provided with a prompt and potential database content and will be required to generate a lesson plan and code. This workflow should be tested to see if the LLM can do both of these tasks properly.

Week 5

Connect Chainlit frontend with the FastAPI backend and enable user prompts to connect to the FastAPI backend. Test if end-to-end workflow works, from fetching data from the database, to prompting the LLM, to receiving the results on the frontend.

Week 6

Start working on the code import tool for the MusicBlocks environment which can take both generated block-based and JavaScript code and import it into the editor. Additionally, present the working proof of concept so far for the midterm evaluation.

This will include:

- Basic Chainlit frontend
- Basic RAG and Vector DB for code generation
- Basic LLM integration that uses RAG

Week 7

Process the additional lesson plan data and add it to the respective database. Furthermore, if the A-MAPS or Lakh Dataset wasn't used earlier for the proof of concept, work on adding it in at this point.

Week 8

Create prompt templates for outputs to prevent misuse of the AI and also increase output quality by making it more relevant to the purpose of lesson plans and code generation. Also, add keywords to look out for in prompts and link it to specific templates.

Week 9

Improve the Chainlit UI for user usage, adding history and easy import features directly into the UI. Conduct user testing with mentors and community members to see which UIs work the best.

Week 10

Refine the unit, integration, and E2E tests created and clean up any messy code present inside the codebase.

Week 11

Start generating documentation and developer guides for both users and further developers of the tool. Make it comprehensive with demos and screenshots.

Week 12

Create the final version and add final polish to the project. Submit the final version and report for the final mentor evaluation.

This will include:

- Final code and lesson plan generation tool
- Code import tool
- Report and documentation

Week 13

Potential last week to fix up any minor bugs and work on features in case of any delay due to emergencies.

How many hours will you spend each week on your project?

If selected, I will be working on GSoC full-time. I am able to put in around 40 hours per week on this project, and I am open to putting in more hours if the project demands it.

How will you report progress between evaluations?

Regularly syncing up with the project mentors is perfectly fine with me. Any form of digital communication works for me. I can also ensure transparency in my work by updating a

shared public progress document that shows the mentors exactly how much progress has been made and provides resources for future documentation of the feature.

Furthermore, hitting the milestones provided in the timeline on time will show tangible proof of progress.

Discuss your post GSoC plans. Will you continue contributing to Sugar Labs after GSoC ends?

Yes, I plan to remain an active contributor to Sugar Labs, specifically MusicBlocks as I am interested in the music initiative it brings. I aim to enhance the project by testing it regularly, pointing out issues and enhancements, solving issues and submitting PRs, and supporting new contributor for the future community growth and hopefully future GSoC sessions.

Also, as I get further in my music studies, I want to use MusicBlocks as a tutoring tool to introduce music theory concepts to younger generations, gaining a deeper understanding of the platform along the way. This will help me point out more relevant issues and expand the community of MusicBlocks directly.