Sugar Labs

Google Summer of Code 2025

Project Information:

Name: Git backend for Turtle Blocks and Music Blocks (link)
Length: 350 hours
Coding Mentors: Walter Bender, Sumit Srivastava
Assisting Mentor: Devin Ulibarri

Student Details:

Full Name: Nikhil Bhatt
Email: bhattnik442@gmail.com
Github: BeNikk
LinkedIn: Nikhil-Bhatt
Preferred Language: English
Location: New Delhi, India
Time Zone: Indian Standard Time (IST) (UTC +05:30)
Institution: Maharaja Agrasen Institute of Technology
Program: B.Tech in Information Technology (Expected 2026)

About Me

I have always been fascinated by the idea of **building things** that create real-world impact. The thought of solving problems and improving lives using just a laptop felt like magic. This idea sparked my interest in programming and motivated me to explore how technology is built to create a meaningful impact.

As my interest grew, I discovered the world of Open Source - where I could see **exactly how great tools are built**, and contribute to improving them. The thought that I could be part of this ecosystem and help in building something meaningful truly inspired me.

My open-source journey began when I discovered SyllabusX, a project that helped students at my university find exam resources. I reached out to the maintainer, who was my senior, and he encouraged me to contribute by making the project a Progressive Web App (PWA). With his guidance and open-source resources, I submitted my first pull request, marking the start of my contributions to a project I relied on.

Later, I participated in GirlScript Summer of Code, an open-source program where I contributed to another project by converting its entire frontend from JavaScript to TypeScript. The maintainer was happy with my work, and receiving appreciation from someone I didn't personally know helped me gain confidence in my skills.

As I continued exploring, I started working on personal projects to apply my knowledge and experiment with different technologies. These projects, which I have shared on my GitHub, with some deployed on my Portfolio, helped me understand real-world development challenges, best coding practices, and scalable software design.

Alongside building projects, I developed a passion for sharing knowledge. I recently conducted two sessions on HTML, CSS, and JavaScript, where I guided students in building their first web applications. Their enthusiasm and positive feedback reinforced my belief in the power of hands-on learning and accessible education. (Link)

Sugar Labs- My Journey So Far

I discovered Sugar Labs through Music Blocks in the summer of 2024 while exploring projects that align with both my skill set and my interest in education and open source learning. Music Blocks immediately stood out as a project that combined computational thinking with creativity.

I started by understanding the codebase (which felt huge at the time) and submitted my first pull request to Music blocks, which focused on fixing a bug where opening the hamburger menu caused the y-zero point to shift (Issue #3914). The process of contributing, receiving feedback, and collaborating with maintainers was a valuable and exciting learning experience.

Since then, I have been actively contributing to Music Blocks, working on bug fixes, feature improvements, and UI enhancements. I regularly attend community meetings and participate in the Sugar Labs and Music Blocks Matrix channel, discussing challenges, proposing ideas, and collaborating with maintainers.

My Contributions to Sugar Labs

Here is a detailed summary of the work and contributions that I did before the GSoC proposal period:

Features Added:

- **Introduced Cypress testing to Music Blocks (#4124)**
    - Set up the initial Cypress testing framework by creating the folder structure and writing the first set of test cases to automate UI testing. (Merged-#4362)

Bug Fixes:

- **Resolved *docById* undefined error (#4292)**
    - Resolved a race condition by explicitly ordering module dependencies in RequireJS. (Merged-#4295)
- **Fixed _onResize Execution Before Stage Initialization (#4300)**
    - Added a defensive check to ensure _onResize executes only after the stage is initialised. (Merged-#4301)

- **Ensured proper hiding of helpful search div (#4275)**
    - Implemented a check to ensure helpfulSearchDiv is closed before opening the Context menu. (Merged-#4276)
- **Resolved module export reference error (#4228)**
    - A conditional check was added to ensure the module is only exported when it exists, establishing a standard practice for handling exports in test case implementations. (Merged-#4229)
- **Fixed unintended helpfulWheel appearance (#4217)**
    - Modified the doContextMenus event handler to check if right click intersects with block coordinates, improving overall user experience. (Merged-#4219)
- **Added Custom Timeout for Block Deletion Notification** (#4094)
    - Implemented txtMsg with a timeout to display a notification whenever blocks are deleted. (Merged-#4097)
- **Added Notification for Drag-and-Drop Block Deletion (#4112)**
    - Extended the existing block deletion notification system to trigger for drag-and-drop deletions as well. (Merged-#4113)
- **Fixed Bottom Buttons Disappearing on Resize (#4053)**
    - Modified button container creation to always ensure visibility regardless of the previous state (Merged-#4054)
- **Added Close Button to Pie Menu (#3933)**
    - Added a close to the pie menu, allowing users to dismiss it more intuitively, enhancing user experience (Merged-#3934)
- **Fixed Y-Zero Shift Issue When Opening Hamburger Menu (#3914)**
    - Adjusted the grid positioning based on the hamburger menu toggle, ensuring it remains in sync with other elements. (Merged-#3924)

Testing Contributions:

- **Added Unit Tests for musicutils.js**
    - Improved test coverage by validating the correctness of key utility functions. (Merged-#4147)
- **Added Unit Tests for RhythmBlocks and DrumBlocks API**
    - Validated core functionalities to ensure expected behavior across different use cases. (Merged-#4208)

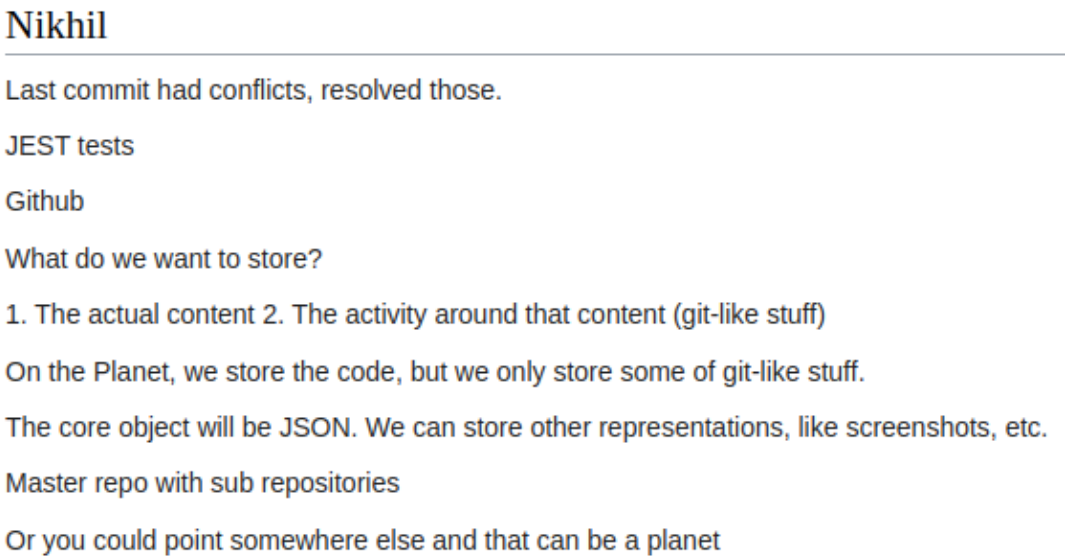
Documentation Improvements:

- **Added Music Blocks Documentation to Sugar Docs**
  - Provided clear navigation to essential Music Blocks links, making it easier for new contributors to get started. (Merged-#222)

Community Engagement & Meetings:

Meeting with fellow contributors and maintainers has helped me communicate clearly and collaborate effectively on ideas, bug fixes, and implementations. I always look forward to our discussions on Wednesdays and Sundays, where we brainstorm improvements and solve challenges together.
Links to the meeting notes can be found here, highlighting my active participation.

Below are excerpts from the meeting notes, documenting discussions on bug fixes and feature improvements.



Nikhil

Last commit had conflicts, resolved those.

JEST tests

Github

What do we want to store?

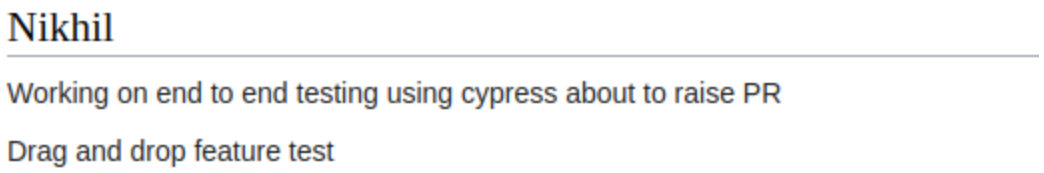
1. The actual content 2. The activity around that content (git-like stuff)

On the Planet, we store the code, but we only store some of git-like stuff.

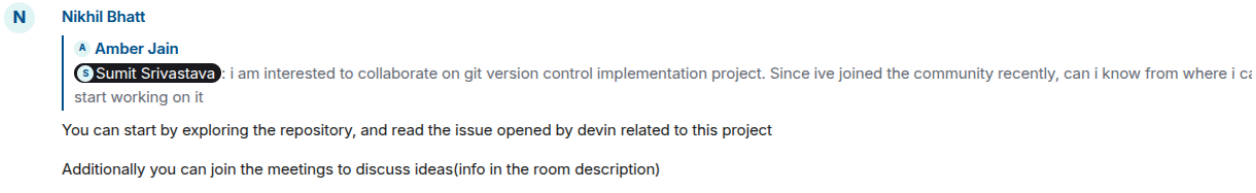
The core object will be JSON. We can store other representations, like screenshots, etc.

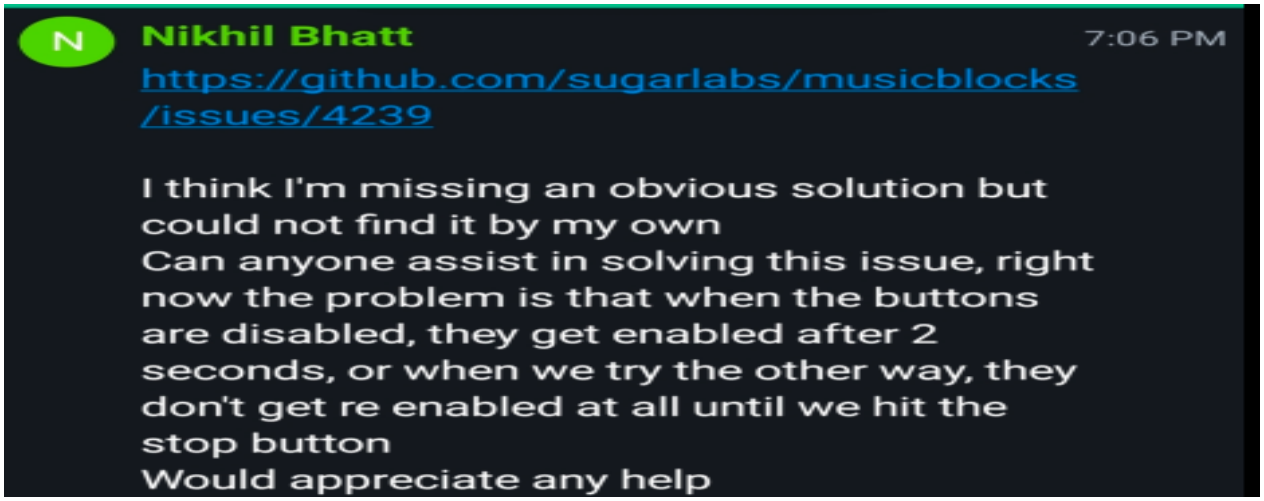
Master repo with sub repositories

Or you could point somewhere else and that can be a planet



Apart from the meetings, I'm also active in the Sugar Labs and Music Blocks Matrix channels, where I chat with contributors and maintainers, troubleshoot issues, exchange ideas to improve the project, and help new developers get started with Music Blocks. (Matrix username- @nikkhil:matrix.org)





Project Details

Introduction:

Music Blocks and Turtle Blocks are interactive learning platforms developed by Sugar Labs to help students learn computational thinking through creative coding. Currently, these platforms allow students to create and share projects via the **Planet server**, but they **lack version control**, making it difficult to track changes, experiment with different iterations, and collaborate effectively.

Version control is a powerful educational tool that fosters **collaboration, problem-solving, and structured thinking**. Git, the most widely used version control system, helps students track progress, experiment confidently, and work on projects collaboratively. Integrating Git into Music Blocks and Turtle Blocks will enable students to organize their learning, explore different approaches, and engage in open-source collaboration.

This project aims to integrate Git-based version control into Music Blocks and Turtle Blocks, introducing:

Anonymous and authenticated project publishing – Students can publish their projects to the central Sugar Labs repository without logging in.

Project history tracking – Students can commit changes, view past versions, and revert when needed.

Forking and branching – Students can create forks and branches to experiment without affecting the original project.

Merging contributions – Students can merge projects seamlessly, allowing collaboration and iterative improvements.

Pull requests and collaboration tools – Students can propose and review changes before integrating them.

Graphical history visualization – Students can view project history, branches, pull requests, and changes in an intuitive UI.

These features will help students learn structured collaboration, debugging, and reflective learning, **aligning with Sugar Labs' mission of making education open, transparent, and interactive.**

Detailed Breakdown:

Understanding the current system

Sugar Labs' Music Blocks and Turtle Blocks provide students with a creative platform to explore computational thinking. These tools allow students to create and share projects via the **Planet server**, a system where projects are uploaded and stored.

Currently, the *Planet* only stores projects that users have created, without any version control features. Without version control, students miss out on key learning experiences such as refining ideas, experimenting safely, collaborating effectively, and tracking their progress over time.

Current Limitations of the Planet Server:

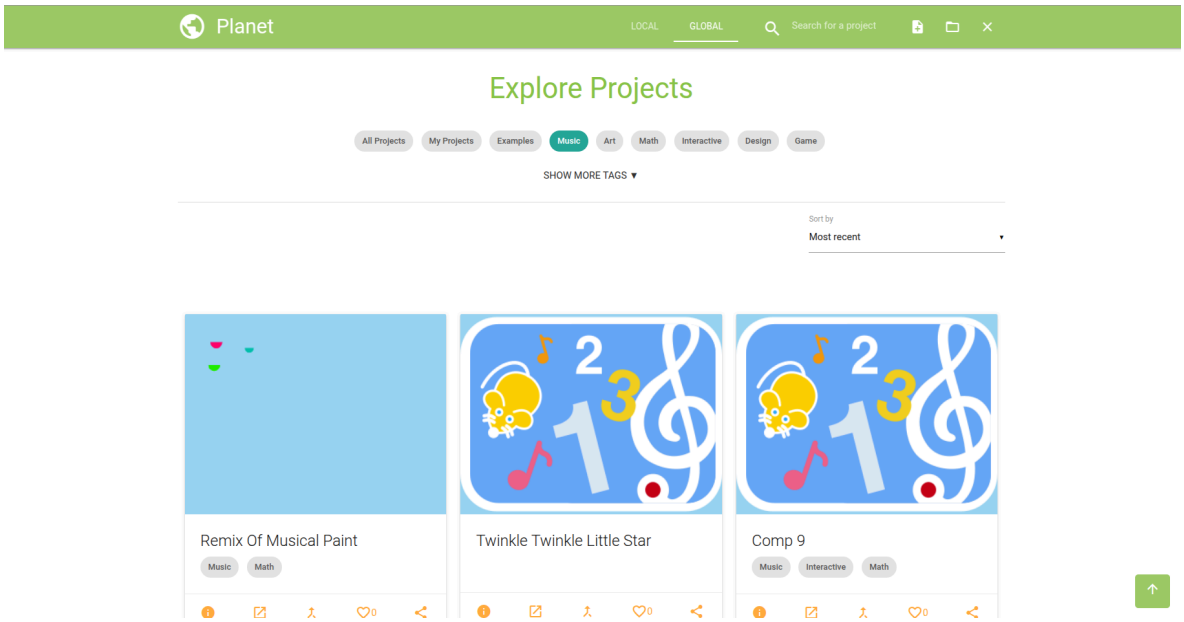
1- No Version History - Once uploaded, projects cannot be reverted to earlier states, making it hard for students to track progress and reflect on their learning. Without past versions, **students miss the opportunity to analyze how their work has evolved and learn from previous iterations.**

2- No Branching - Students can open and edit shared projects but cannot branch from them, limiting experimentation. Without isolated workspaces, students are discouraged from trying out unique ideas or tracking creative progress.

3- No Collaboration Tools - There is no structured way to contribute to a shared project or review changes. Without proper tools for feedback and peer review, **students miss out on learning from each other's approaches and refining their work through constructive discussions.**

4- No Structured Feedback & Review Process - There is no system for reviewing or suggesting improvements before changes are applied. **Students cannot engage in structured discussions, receive guided feedback, or iterate on their ideas based on constructive criticism, limiting their ability to refine their work.**

Current Planet Server – lacking version control features such as history tracking, branching, and collaboration tools.



Integrating Git-based version control to Music Blocks:

To address the limitations of the Planet server, this project introduces Git-based version control to Music Blocks and Turtle Blocks. Instead of projects being stored as static uploads, they will now be tracked, versioned, and open for collaboration.

By integrating Git, students will not only gain access to a **structured workflow for managing changes** but also develop skills essential for problem-solving, teamwork, and iterative learning. The introduction of version control will transform how students interact with their projects, allowing them to revisit past iterations, refine their ideas over time, and engage in meaningful collaboration.

The integration of Git will provide students with:

1- **A transparent learning journey** - Students will be able to track their progress over time, understanding how their projects evolve.

2- **A safe space for experimentation** - Through branching and forking, students can test new ideas without fear of losing previous work.

3- **Collaborative learning** - By implementing pull requests, students will be able to receive feedback and learn from their peers in a structured way.

4- **Visual History & Issue Resolution Workflow** - By engaging in an issue-resolution workflow, students will learn how to identify problems, break them into actionable steps, and implement solutions.

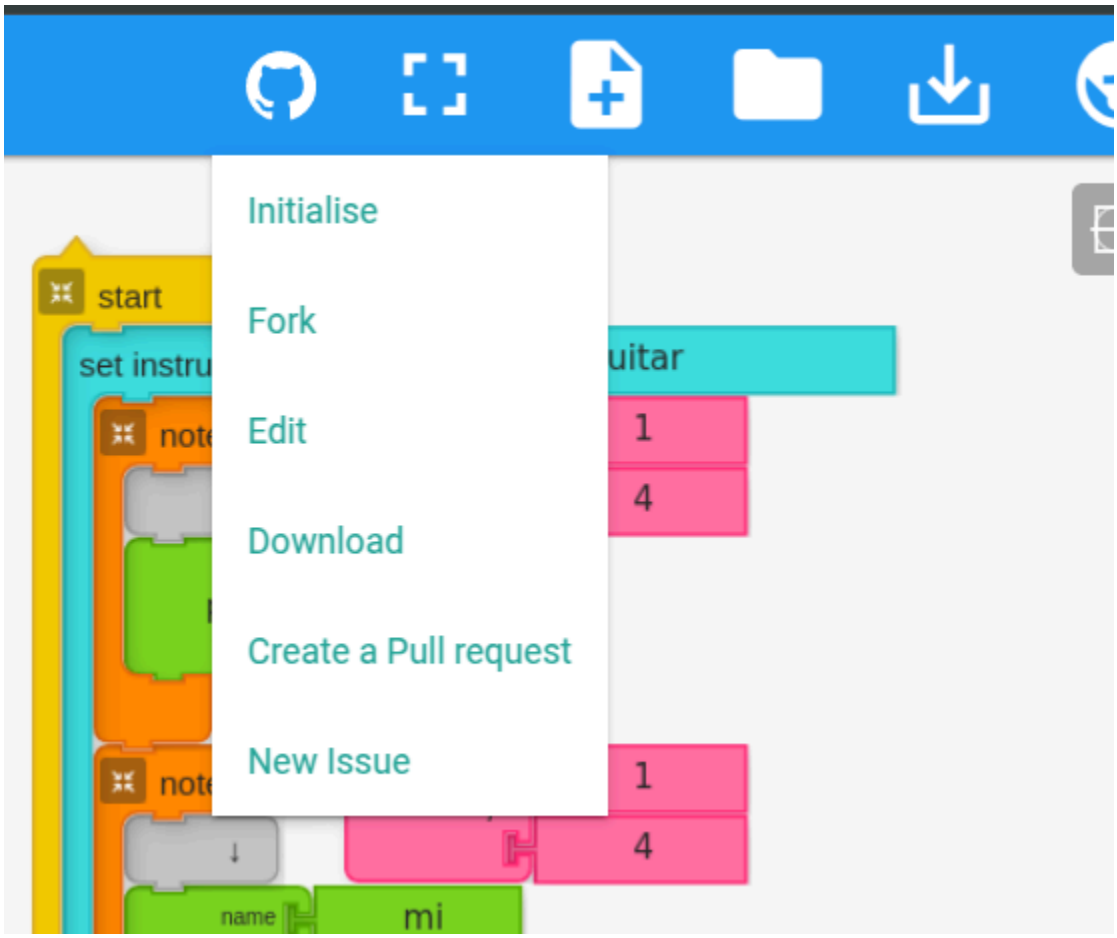
Technical Requirements:

The implementation of Git based version control in Music Blocks requires the following technologies:

- **Git for version control**
  - Used to track changes, manage branches, and enable collaboration through commits, forks, and pull requests.
- **Git Forge API (Github / GitLab Implementation)**
  - A backend API will interact with GitHub/GitLab to manage repositories, track commits, and handle pull requests, forking, and merging.
  - For example, Libraries such as Simple Git and Octokit can be used for interacting with repositories
- **Backend Implementation - (Node.js + TypeScript)**
  - A Node.js server (written in Typescript) will handle all Git-related operations, such as creating repositories, committing changes, and merging pull requests without storing any user information.
  - The backend will generate unique keys to identify project owners and manage permissions securely.
  - Typescript will ensure strong type safety, **reduce runtime errors**, and improve overall code maintainability and developer experience.
- **Frontend Integration - (HTML, CSS, AND JAVASCRIPT)**
  - The Music Blocks and Turtle Blocks interface will be updated to support Git operations via interactive UI components.

- Students will be able to commit changes, fork projects, create pull requests, and view project history directly within the platform.

Potential UI for Git integration in Music Blocks/Turtle Blocks (here)



A proposed Git integration menu for Music Blocks, suggesting options like initializing repositories, forking, editing, downloading, creating pull requests, and reporting issues.

System Architecture and Implementation Strategy:

The system integrates Git-based version control into Music Blocks and Turtle Blocks, enabling efficient project management **without requiring user authentication**. The architecture consists of three key components:

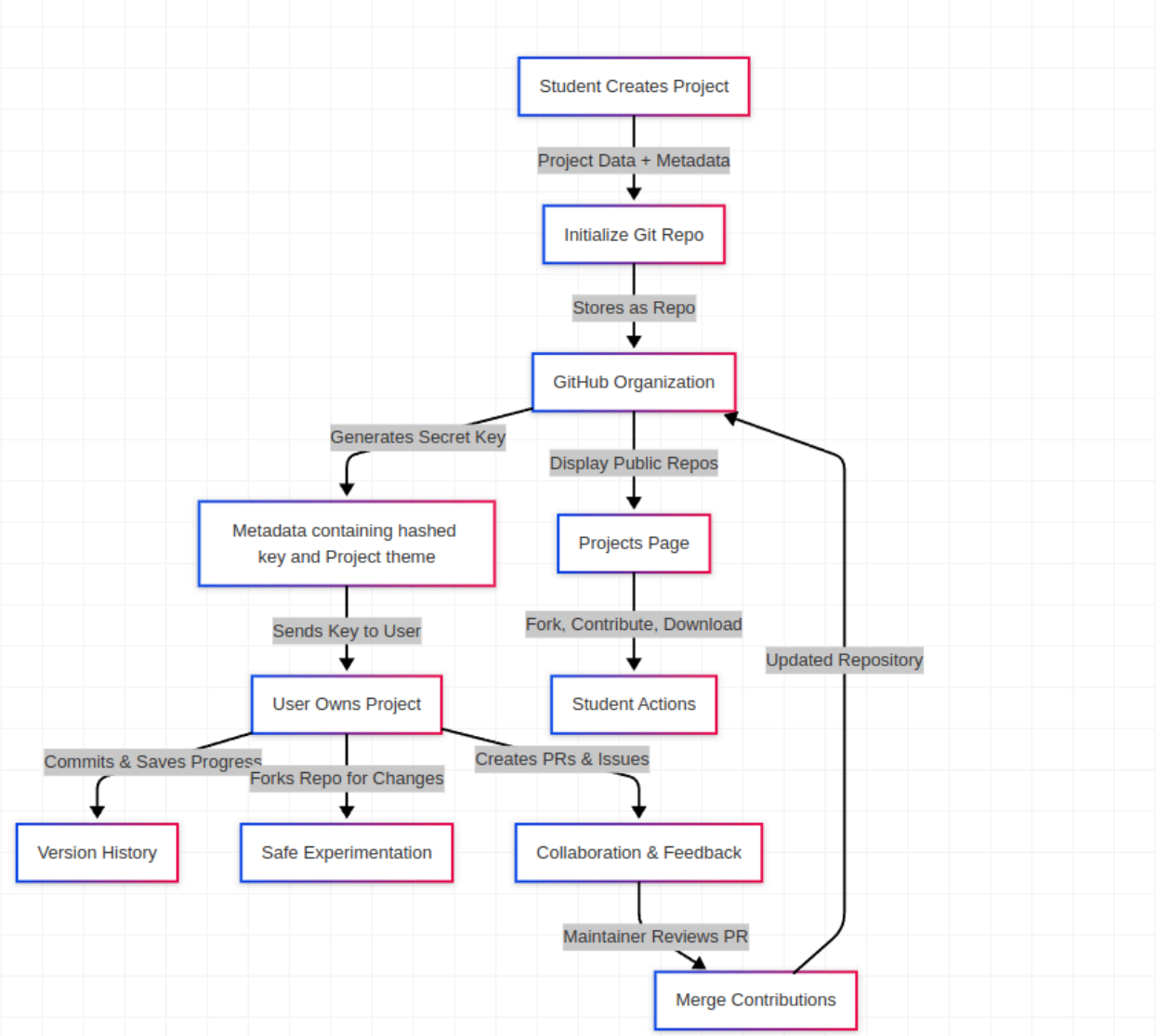
- **Project Storage and Repository Management**
  - Each project is stored as a **separate GitHub repository** under a designated **Music Blocks organization**.
  - The GitHub Repository will be storing
    - **Project Data:** Stored as a **JSON file**, representing the blocks and their arrangement.
    - **Metadata:** Contains additional information for project management:
      - **Hashed Key** - A unique identifier allowing access control without authentication.
      - **Theme Identification -** Projects are automatically categorized based on tags available in the Planet server.
      - **Other metadata** for managing project settings.
  - **Project Workflow:**
    - **Initialize Repository** - Users create a new GitHub repository for version control.
    - **Commit-Based Saving** - Changes are saved as commits, allowing history tracking.
    - **Forking & Collaboration** - Users can fork, modify, and submit PRs to contribute.
  - **Access Control (Without Authentication)**
    - Users with the hashed key can edit, merge PRs, and close issues.
    - Other users can fork projects and submit PRs, but need approval for modifications.
- **Backend Implementation**
  - The backend is built using Node.js with an Express.js server, written in **TypeScript** for better type safety, scalability, and maintainability. It handles all Git operations and interacts with Git forges like GitHub or GitLab to manage project lifecycle features.
    - **Git Management**: Uses libraries like **SimpleGit** for repository operations.

        - **API Integration:** Uses libraries like **Octokit.js** to communicate with GitHub's API for repository creation, commits, PRs, and issues.
        - **Metadata Storage:** The system maintains project data and metadata for efficient management.

- **Frontend Integration**
    - The frontend provides an intuitive interface for interacting with Git-based version control.
    - **Git Menu Integration:**
        - A Git menu in the UI enables users to perform essential Git operations within Music Blocks and Turtle Blocks:
            - Initialize a project repository
            - Fork an existing project to create a personal copy
            - Edit & Save changes with commit history
            - Download project files
            - Create Pull Requests for contributions
            - Raise Issues to report bugs or suggest improvements
    - **Dedicated Projects Page:**
        - A "Projects" page, similar to the current Planet page, will display all public repositories under the Music Blocks GitHub organization. This will include:
            - **Project Listings:** Show all available projects with details like name, theme (Music, Art, Math), and last update.
            - **Forking Option:** Users can fork any project to start their modifications.
            - **Contribution Actions:** Provides buttons to submit Pull Requests (PRs) and open Issues directly from the UI.

The proposed architecture and implementation strategy were designed based on discussions with Walter Bender and are subject to refinement as the project progresses. **Detailed notes from these discussions can be found here.**

A Systematic Flow chart showcasing the Architecture of the Github Backend Project



Deliverables

As part of GSoC, I will deliver a **fully functional Git-based version control system** integrated into Music Blocks and Turtle Blocks. This includes:

- **Git Backend for Version Control**
  - **Develop a backend system** using Typescript, Node.js, Express.js, and libraries like SimpleGit and Octokit.js to interact with GitHub and manage repositories.
  - **Implement anonymous project storage** - Users can publish to a central Music Blocks repository without authentication, having the option to **download** the project to publish on their own Github repositories.
  - **Enable project history tracking** - Students can commit changes, revert versions, and view progress over time.
  - **Introduce forking & branching functionality** to allow safe experimentation without modifying the original project.
- **Collaboration & Contribution Features**
  - **Integrate pull request functionality** - Students can propose changes and receive structured feedback.
  - **Develop an issue tracking system** - Users can report bugs, suggest improvements, and track discussions.
  - **Implement a merge approval system** - Only users with the project key (stored in metadata) can approve and merge contributions, ensuring access control without authentication.
- **Frontend Integration & UI Enhancements**
  - **Add a Git Menu** to the Music Blocks and Turtle Blocks UI for easy interaction with version control features.
  - **Create a dedicated 'Projects' Page** similar to the **current Planet page**, displaying all public repositories under the Music Blocks GitHub organization, with options to **fork, contribute, and view project history.**
- **Testing, Documentation & Deployment**
  - **Prototype testing** with partner schools to refine the system based on real-world student feedback.
  - **Write developer & user documentation** for setting up and using Git-based version control in Music Blocks.
  - Deploy the system & integrate it into Sugar Labs' infrastructure.

- Facilitate the transition away from the existing Planet Server once the new system is fully functional.

MVP Development & Initial Prototype

To demonstrate the feasibility of Git-based version control for Music Blocks, **I developed a Minimum Viable Product (MVP)** that implements core functionalities, including:

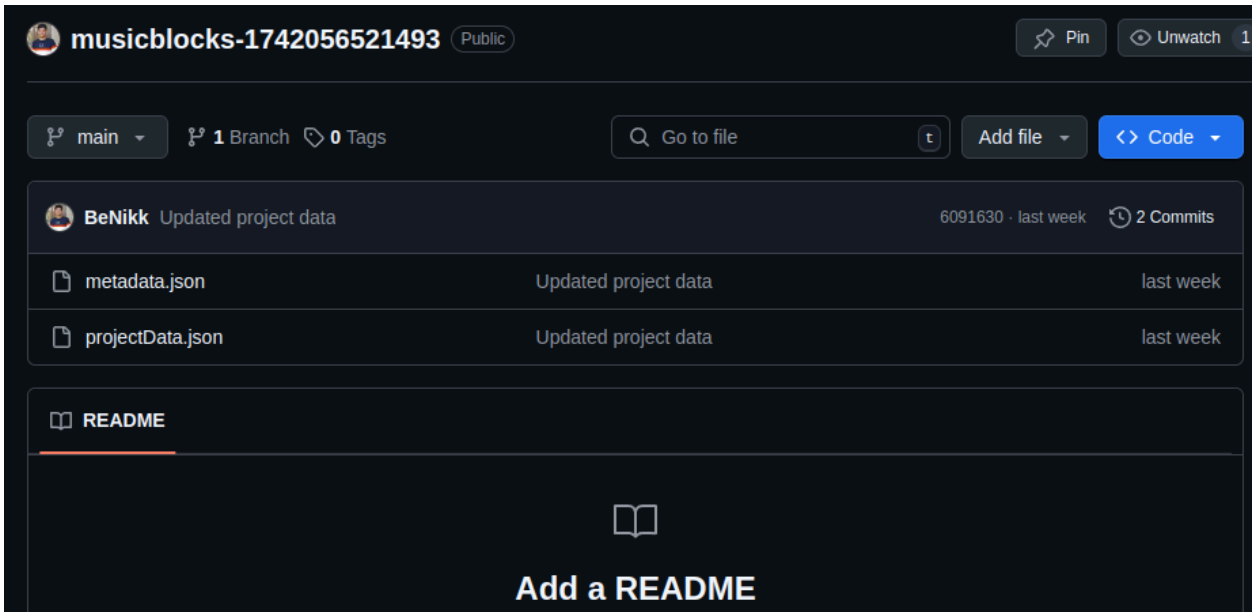
- **Project Initialization & Storage** - Each project is stored as a dedicated GitHub repository.
- **Versioning & Commit History** - Changes are tracked through commits, allowing users to review progress.
- **Forking & Pull Requests** - Users can fork projects and submit pull requests without modifying the original work.
- **Editing & Metadata Handling** - Project data (blocks JSON) is managed along with access control using hashed keys.
- **Visual Commit History** - A “Show History” feature displays all commits in a visual timeline, allowing users to preview and switch between past project versions.
- **Downloading Project** - Users can download the GitHub repository as a ZIP file directly from Music Blocks

This MVP was reviewed by Sugar Labs maintainers in our bi-weekly meetings, including Walter Bender and Devin Ulibarri, who provided valuable and positive feedback. Their insights will guide further refinements as development progresses.

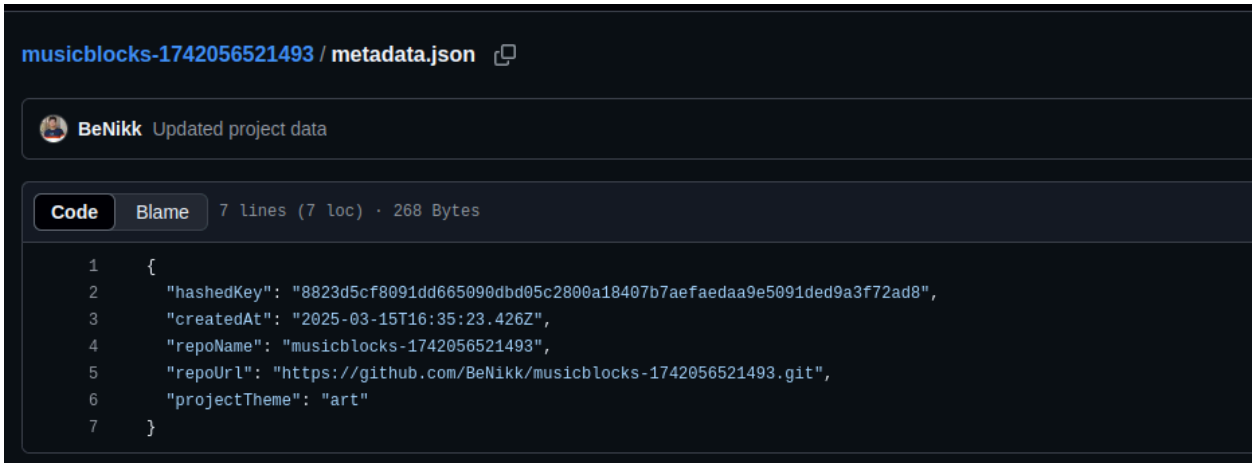
Github Repository of the server developed as a part of MVP - (Link)

Github Repository of the Project created through Musicblocks, containing the Project Data and its metadata - (Link)

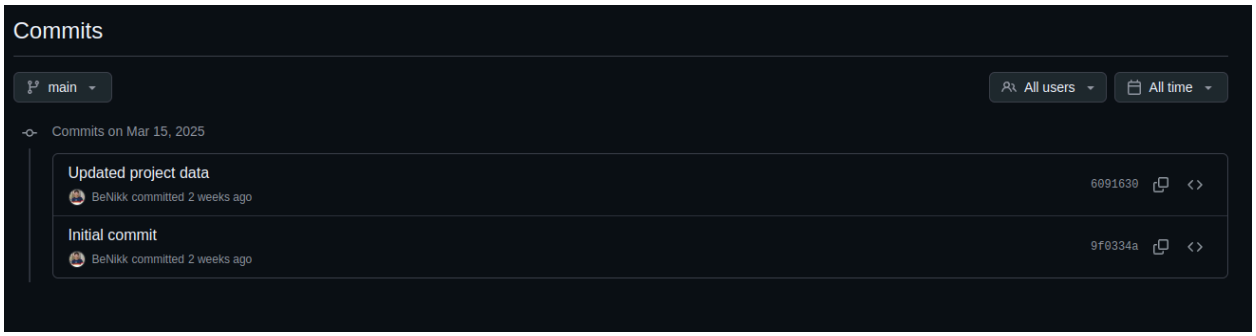
Screenshot of the Repository Created- containing the Project Data and its metadata



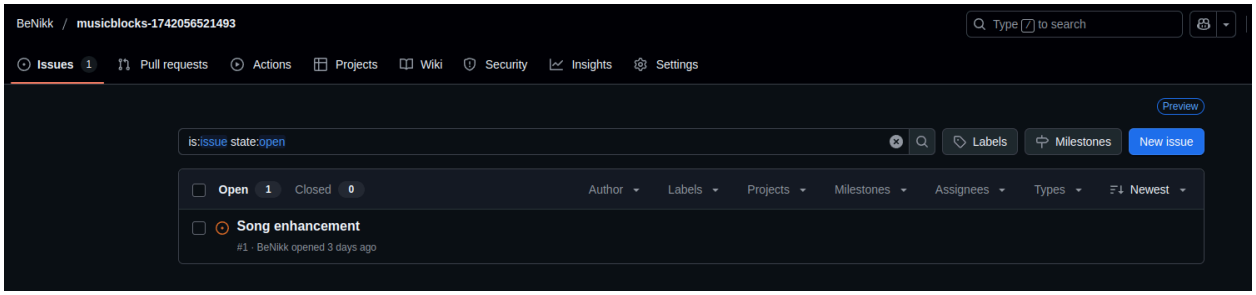
Example of the metadata stored in github repository, containing the Theme and hashed key



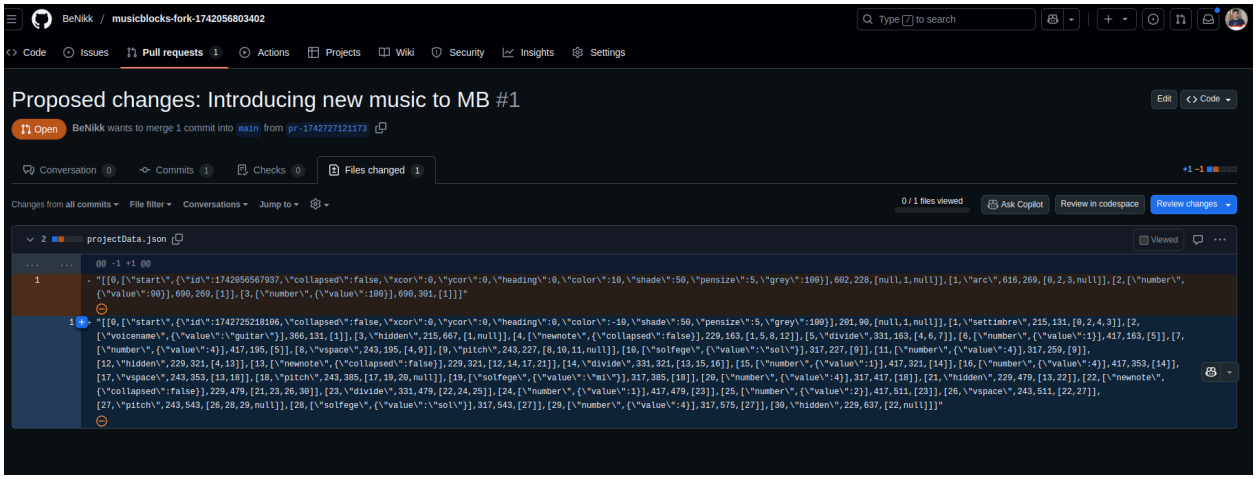
Commits to a project - ensuring version control and history tracking



Issues opened through Music Blocks UI, ensuring feedback is received



Pull request created directly through Music Blocks, with changed project data. (In the actual implementation, students will be able to directly see the changes visually in Music Blocks)



Some challenges:

Throughout the initial development of the MVP and discussions in community meetings, several challenges have come up. One major concern is how to **represent Git diffs meaningfully** within a block-based environment like Music Blocks.

Another challenge is integrating advanced Git features, like branching, merging, and pull requests, while keeping the interface simple and friendly for all age groups. Striking the right balance between functionality and usability will be the key.

I am also aware that more challenges will likely emerge as development continues. Some of them I may not have anticipated yet, but with constant feedback, support from mentors, and discussions with the community, I will tackle each issue iteratively and thoughtfully.

Timeline

This timeline provides a structured high-level roadmap for the project, outlining key milestones, deliverables, and checkpoints. While the plan may evolve based on feedback and testing, it ensures a clear progression from research and initial implementation to full integration and refinement.

- **Pre-GSoC Period (March - May 8)**
  - Continue refining the MVP and gathering feedback from maintainers.
  - Improve understanding of GitHub APIs and repository management.
  - Discuss implementation details with mentors to finalize the approach.
- **Community Bonding Period (May 8 - June 1)**
  - Engage with the community and participate in discussions.
  - Refine the implementation plan based on mentor feedback.
  - Set up the backend structure for Git integration.
  - Research best practices for handling project metadata (themes, keys, version tracking).
- **Phase 1: Backend Implementation & Core Features (June 2 - July 14)**
  - **Week 1-2 (June 2 - June 16)**

        - Finalize GitHub repository structure for storing Music Blocks projects.
        - Implement repository creation API (initialize, store metadata, commit structure).
    - **Week 3-4 (June 17 - July 14)**
        - Implement forking & branching workflows.
        - Develop backend support for Pull Requests & Issue creation.
        - Ensure Robust testing of backend APIs.
        - Begin drafting technical documentation for API endpoints.
- **Midterm Deliverables (July 14 - 18):**
    - Full backend support for Git operations.
    - API endpoints are ready for repository creation, commits, forks, and PRs.
    - Initial documentation of Git workflows for contributors.
- **Phase 2: Frontend Integration & UI Enhancements (July 14 - Aug 25)**
    - **Week 5-6 (July 15 - July 29)**
        - Integrate "Git Menu" for project management (initialize, fork, edit).
        - Implement contribution workflows (submit PRs, open Issues).
        - Begin user testing with community contributors & refine features.
    - **Week 7-8 (July 30 - August 25)**
        - Develop “Projects” page displaying repositories (similar to Planet).
        - Add UI for project visualization (history, commits, branches).
        - Start testing with partner schools, gather feedback, and iterate on improvements.
- **Final Deliverables (August 25 - September 1):**
    - Fully functional frontend integration.
    - UI allowing seamless project tracking & contributions.
    - Refined documentation for new contributors & educators.
    - Initial insights from school testing & planned improvements.

Availability:

I plan to dedicate 35-50 hours per week to this project. While I may have exams during the GSoC period, I will manage my time to ensure steady progress. My most active working days will be Monday to Wednesday, with working hours from 9 AM to 6-7 PM IST.

Progress Reporting:

To ensure transparency and steady progress, I will provide daily updates on the Matrix channel, sharing key developments and addressing any blockers. Additionally, I will participate in biweekly meetings with mentors to review implementation, discuss challenges, and modify the approach as needed. To document significant milestones and insights, I will also write blog updates every 2-3 weeks, detailing technical progress, key takeaways, and future steps.

Post-GSoC Plans:

After completing GSoC, I plan to continue improving and maintaining the Git-based version control system for Music Blocks and Turtle Blocks, ensuring it remains a valuable tool for students and educators. I will **actively review issues and pull requests**, assisting new contributors in understanding and enhancing the system. Additionally, I aim to refine the integration by gathering feedback from real users, particularly from schools adopting this workflow.

Beyond this project, I am deeply committed to **free and open-source education**. Sugar Labs provides an incredible platform to empower students through open tools, and I see this project as just the beginning. I plan to **explore expanding version control features across more Sugar Labs projects**, integrating them into Music Blocks v4 and improving Sugarizer's collaborative aspects. My long-term goal is to help Sugar Labs scale its educational initiatives and foster a stronger **community of student contributors**.

Conclusion

Thank you for taking the time to read my proposal. This project is not just about adding version control, it is about empowering students with structured

collaboration, safe experimentation, and real-world open-source practices. By implementing Git-based workflows into Music Blocks and Turtle Blocks, we are equipping learners with **essential problem-solving skills** while making open-source education more **interactive, transparent, and engaging**.

I am fully committed to Sugar Labs and this project. With my experience in Git, backend development, and frontend integration, I am confident in delivering a **scalable, impactful solution**. I look forward to working closely with my mentors and contributing to the long-term growth of Sugar Labs.

My focus is entirely on Sugar Labs, and I am fully committed to contributing to this project without any plans to submit proposals elsewhere.