Project Overview & Setup

Goal: Collaborative code editor with P2P real-time sync, code execution, and a user-friendly UI. You'll use **JavaScript/Node.js** for networking and execution, **CodeMirror** (or **Monaco Editor**) for the code editor UI, and **WebRTC/Yjs** for syncing.

Day 1: Initial Setup and Research

1. **GitHub Setup**
    - **Everyone**: Create a GitHub repo and make separate branches:
        - Sanjay: `p2p-sync`
        - Rohit: `code-execution`
        - Dush: `frontend-ui`
    - Practice pulling, committing, and pushing code to get comfortable with Git.
2. **Learn the Basics**
    - **Sanjay**: Research **WebRTC** basics. Aim to understand how P2P connections work (check out Mozilla's WebRTC guide).
    - **Rohit**: Dive into the **Node.js VM** module to see how to run code in an isolated environment.
    - **Dush**: Start exploring **CodeMirror** or **Monaco Editor**. Try creating a simple editor that displays basic text.

End of Day Check: Each of you should have set up your branches on GitHub and pushed a basic "Hello World" setup for your component.

Day 2: Environment Setup and More Research

1. **Setup Automation**
    - **Sanjay**: Write a basic shell script that installs **Node.js, npm**, and any necessary packages. This script should work when any new device connects to ensure a consistent setup.
2. **Research P2P, Execution, and UI**
    - **Sanjay**: Look deeper into **WebRTC's data channels** for messaging between peers.
    - **Rohit**: Learn about **Python subprocesses** in case you decide to support Python code execution later.
    - **Dush**: Research how **WebRTC** integrates with front-end UI.

End of Day Check: Sanjay's setup script should be ready, and each of you should have a better understanding of your main technologies.

Day 3: Core Prototyping (Start Building Basics)

1. **Core Prototypes**
    - **Sanjay**: Build a simple WebRTC connection where peers can send text messages to each other.
    - **Rohit**: Write a basic **Node VM** function that runs simple JavaScript code and returns the output.
    - **Dush**: Set up the **CodeMirror** editor with a basic layout and syntax highlighting.
2. **GitHub Check-in**
    - Push prototypes to GitHub and test each other's setups by pulling code and running it locally.

End of Day Check: Each person should have a basic, working prototype of their part.

Day 4: Syncing Setup, Execution Sandbox, and UI Controls

1. **Develop Core Functionalities**
    - **Sanjay**: Integrate **Yjs** for collaborative editing, set up basic syncing between two peers.
    - **Rohit**: Improve the VM setup by adding **error handling** (catch and log errors in code execution).
    - **Dush**: Add an **Execute button** and a basic layout for displaying code output below the editor.
2. **Cross-check Progress**
    - Check each other's branches and leave comments. This early check will ensure everyone's on the same page.

End of Day Check: All members should have a working initial version of their component.

Day 5: Refining Core Components

1. **Expand Features and Testing**
    - **Sanjay**: Test the Yjs sync setup with three peers to see how edits propagate.
    - **Rohit**: Write tests for the sandbox to handle different types of code and errors.
    - **Dush**: Wire up the Execute button with a dummy function to mimic output, making the UI more interactive.

End of Day Check: You should each have a stable version with basic functionality.

Day 6: Initial Integration

1. **Connecting Components**
    - **Sanjay**: Set up **Yjs** with Dush's editor so that text changes sync in real-time.
    - **Rohit**: Work with Dush to link the Execute button to the code execution function, even if it's a basic setup.
2. **UI Testing and Adjustments**
    - **Dush**: Make any necessary adjustments for smoother integration with Sanjay and Rohit's modules.

End of Day Check: Everyone should see basic integration—editor changes sync, and code execution is triggered from the button.

Day 7: Refining and Improving Each Part

1. **Solidify Core Functionality**
    - **Sanjay**: Continue refining the P2P connection to avoid conflicts during syncing.
    - **Rohit**: Add **security measures** in the execution sandbox to prevent malicious code (simple checks to start).
    - **Dush**: Enhance the UI by adding a status indicator for "Executing..." and a place for results.
2. **GitHub Push**
    - Push changes and make sure each part is working after pulling.

End of Day Check: Each module should work smoothly and be ready for testing as a full system.

Day 8-9: Full Integration and Group Testing

1. **Connect All Pieces Together**
    - **Sanjay**: Finalize text syncing with Dush's editor.
    - **Rohit**: Ensure that sandboxed code results can be shared via P2P with Sanjay's sync.
    - **Dush**: Add a clear display for who is editing or executing code.
2. **Group Testing**
    - All: Test across devices to ensure that edits sync and code executes seamlessly between all peers.

End of Day Check: Your first fully integrated version should be ready by Day 9. Bugs or syncing issues can be noted for fixing.

Day 10-12: Performance Tweaks, Bug Fixing, and Minor Features

1. **Fix Bugs and Add Minor Features**
    - **Sanjay**: Add simple **conflict resolution** for text edits.
    - **Rohit**: Test various code inputs and improve performance, especially for repeated executions.
    - **Dush**: Improve UI for readability and add minor touches, like a loading indicator for execution.
2. **Daily Commits**
    - Push each day's work to GitHub so others can test updates as you make them.

End of Day Check: Each member's component should work well under different scenarios, ready for full use.

Day 13: Documentation, Code Comments, and Project Polish

1. **Documentation and Review**
    - **All**: Document each module—how it's set up, used, and any known limitations. Comment your code thoroughly.
2. **Run Full Tests**
    - Go through multiple editing and execution scenarios to ensure smooth functionality.

End of Day Check: Project documentation should be solid, and all modules should be fully ready.

Day 14: Final Bug Fixes and Testing

1. **Last-Minute Tweaks**
    - **All**: Identify any final bugs and work together to fix them. Ensure the sync, execution, and UI are fully stable.
2. **Run a Demo**
    - Each person should test a few full run-throughs of the application, from starting a session to executing and syncing code.

End of Day Check: Everything should work smoothly without major issues.

Day 15: Final Presentation and Wrap-Up

1. **Polish and Demo Preparation**
    - **All**: Prepare a short demo or presentation (optional) to showcase the project, if needed for submission or sharing.
    - Review the codebase, add any final comments, and make a final push to GitHub.
2. **Celebrate**
    - You've completed a collaborative, real-time code editor with P2P sync, code execution, and a cohesive UI in 15 days!

Summary of What Each Member Learned by the End

- **Sanjay**: Mastered P2P communication and sync logic using WebRTC and Yjs.
- **Rohit**: Built a secure code execution sandbox with Node.js VM.
- **Dush**: Created a fully functional editor UI with interactive buttons and real-time feedback.

GitHub Strategy Recap: Daily commits, dedicated branches, regular pulls, and merges as needed. This plan ensures that everyone works on their part every day while staying in sync for seamless integration.