WorkFlow for Algorun:

1. Set Up Project Structure

- **Initialize Backend:**
  - Set up your backend server using Node.js and Express.
  - Configure environment variables and database (MongoDB).
- **Initialize Frontend:**
  - Create the React app for the frontend.
  - Set up routing, basic pages (Home, Login, Signup), and structure.

2. Build Basic Backend API

- **Set Up Routes and Controllers:**
  - Create basic API routes (e.g., for user authentication, saving code, running code).
  - Implement a simple **GET** route to check if the backend is working.
- **Set Up User Authentication:**
  - Implement **JWT authentication** for user login and signup.
  - Add password encryption using **bcrypt**.

3. Build Basic Frontend UI

- **Create User Interface:**
  - Design simple pages (Home, Login, Signup).
  - Build a basic layout with React (use TailwindCSS or ShadCN UI for styling).
- **Implement Authentication Forms:**
  - Build forms for user login and signup, and connect to the backend for authentication.

4. Integrate Code Editor (Frontend)

- **Set Up Code Editor:**
  - Integrate **Monaco Editor** or **CodeMirror** for the code editor component.
  - Add basic features like syntax highlighting and a language selector.

5. Set Up Code Execution (Backend)

- **Integrate Docker for Code Execution:**
  - Set up Docker containers for running code in multiple languages.
  - Ensure safe execution of code with timeouts and resource limits.
- **Create API to Run Code:**
  - Set up the backend to receive code via API and run it inside a container.
  - Return the execution output (success or errors).

6. Integrate Gemini API for Code Suggestions

- **Set Up API Integration:**
  - Integrate the **Gemini API** to fetch code suggestions (autocompletion, snippets).
  - Display suggestions in the editor as the user types.
- **Provide Documentation/Examples:**
  - Show relevant code documentation and examples based on the user's input.

7. Implement Real-time Collaboration (Socket.IO)

- **Set Up WebSockets (Socket.IO):**
  - Implement real-time collaboration where multiple users can edit the same code simultaneously.
  - Sync code changes in real time between users.

8. Implement Code Saving and Sharing (Optional)

- **Save Code Snippets:**
  - Allow users to save their code (store it in MongoDB).
  - Create a "Save" button and a user dashboard to view saved snippets.
- **Code Sharing:**
  - Implement a shareable link feature for users to share their code with others.

9. Implement Code History and Versioning (Optional)

- **Track Code Changes:**
  - Enable versioning of code snippets so users can revert to previous versions.

    - Store past code runs and their outputs.

10. Add UI/UX Enhancements

- **Polish the Interface:**
    - Add **Dark/Light Mode** toggle.
    - Improve error handling and display better messages.
- **Optimize User Experience:**
    - Implement undo/redo functionality in the editor.
    - Add performance improvements for smoother code execution.

11. Testing & Debugging

- **Test Features:**
    - Test each module (authentication, code execution, real-time collaboration).
    - Test the Gemini API integration and ensure it's working correctly.
- **Bug Fixes and Refinements:**
    - Fix any bugs or UI issues based on user feedback.

12. Deployment

- **Deploy Backend and Frontend:**
    - Deploy the backend on **Heroku**, **AWS**, or a similar platform.
    - Deploy the frontend (React app) on **Netlify** or **Vercel**.
- **Final Testing:**
    - Perform final tests on the live app to ensure everything is running smoothly.

13. Optional Features (Post-Launch)

- **Analytics Integration:** Track user behavior and code execution metrics.
- **Advanced Features:** Add features like **code refactoring** and **performance analysis**.

Key Order of Operations:

1. **Backend Setup** (Server, Authentication, API routes).
2. **Frontend Setup** (UI, Forms, Routing).
3. **Code Editor** (Monaco/CodeMirror, Syntax Highlighting).
4. **Code Execution** (Docker, Running Code).

5. **Gemini API Integration** (Suggestions, Autocompletion).
6. **Collaboration** (Real-time syncing with Socket.IO).
7. **Saving/Sharing Code** (Dashboard, Links).
8. **UI/UX Polishing** (Design Improvements).
9. **Testing and Deployment** (Bug Fixes and Launch).