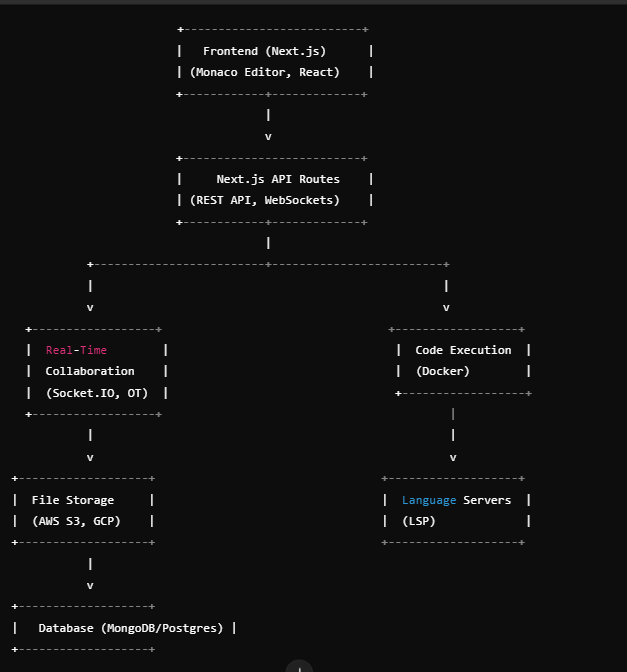
**Architecture**

****

**1. Client-Side (Frontend)**

* **Technologies: NextJs(React Based), Tailwind CSS, Monaco Editor**
* **Role:**
  + User interface for editing code, managing files, and accessing projects.
  + Real-time collaboration UI (showing other users' cursors, text changes, etc.).
  + Interaction with backend services through WebSockets for real-time communication.
  + **Communication with language servers for syntax highlighting, auto-completion, and error checking.**

**2. Backend**

* **Technologies: Next.js API routes, Node.js, WebSockets (Socket.IO)**
* **Role:**
  + Authentication and Authorization: Use NextAuth.js for handling user authentication, including OAuth providers like GitHub, Google, etc.
  + Store user sessions in cookies or JWT tokens.
  + **File Management API: Implement GraphQL APIs using Next.js API routes to handle CRUD operations (create, read, update, delete) on project files and folders.**
  + **Real-time Collaboration: Use Socket.IO within API routes to enable real-time communication for collaborative coding. Each WebSocket connection will track and broadcast changes to the shared code files.**
  + **Code Execution API: Create API routes to trigger code execution by making requests to the containerized environments (Docker) and returning the results (e.g., outputs, errors).3. Code Execution Layer**

**3. Code Execution Layer (Using Docker)**

* **Technologies: Docker, Kubernetes**
* **Role:**
  + **Containerization: Each code execution request from users spins up an isolated Docker container or Kubernetes pod for running the code in a secure environment.**
  + **Language Environments: Pre-configured Docker images for supported languages (Python, JavaScript, Java, C++) to run code within the containers.**
  + **Resource Management: Kubernetes manages the scaling and orchestration of these containers to ensure efficient resource utilization.**

**4. File System & Cloud Storage**

* **Technologies: AWS S3, Google Cloud Storage, or local file storage**
* **Role:**
  + **Store project files in a cloud storage service like AWS S3 or Google Cloud Storage to allow users to access and modify their files across sessions.**
  + **Use Next.js API routes to interact with the file storage service for saving/loading files.**

**5. Language Server Protocol (LSP) Integration**

* **Technologies: Language Server Protocol (LSP), Monaco Editor**
* **Role:**
  + **Integrate LSP (using Monaco Editor's built-in support) to provide features like syntax highlighting, autocomplete, and error detection for various programming languages.**
  + **You can host your own LSP servers or use existing ones and connect them to the frontend editor through WebSockets or HTTP requests.**

**6. Database Layer**

* **Technologies: MongoDB (for unstructured data), PostgreSQL/MySQL (for structured data)**
* **Role:**
  + **User Data Management: Store user information, session tokens, project metadata, and settings.**
  + **Version Control: Store file changes, commit history, and project versioning.**
  + **Collaboration Metadata: Store information on project collaborators, access permissions, and real-time activity logs.**

**7. Real-Time Collaboration**

* **Technologies: WebSockets, Operational Transformation (OT) or CRDT**
* **Role:**
  + **Syncing Changes: Enable multiple users to edit the same file in real-time with consistent updates and conflict resolution.**
  + **Presence Indicators: Show live cursor movements and code changes made by other users in real time.**

**8. DevOps, CI/CD, and Monitoring**

* **Technologies: Vercel, AWS, Kubernetes, Docker, Prometheus**
* **Role:**
  + **Vercel (the platform developed by the creators of Next.js) can be used to deploy the frontend and backend serverless functions (Next.js API routes).**
  + **Use Docker and Kubernetes for containerization and orchestration of code execution environments.**
  + **For monitoring, use Prometheus or Grafana to track API performance, container health, and real-time usage metrics.**

**9. Cloud Infrastructure**

* **Technologies: AWS (EC2, S3, EFS), Google Cloud, Azure**
* **Role:**
  + **Compute Resources: Host your backend and frontend on cloud services using virtual machines (e.g., AWS EC2) or serverless solutions.**
  + **Networking: Manage load balancing and routing through services like AWS Elastic Load Balancer or Cloudflare.**
  + **Auto-scaling: Automatically scale containers or VM instances up or down based on demand, ensuring cost-efficient usage.**

**How can use this technology freely**

Example Project Setup (Cost-Free)

1. Frontend & Backend: Next.js on Vercel (free)
2. Code Editor: Monaco Editor (open source)
3. Authentication: NextAuth.js
4. Real-Time Collaboration: Socket.IO on Vercel (free)
5. Code Execution: Replit API integration (free)
6. File Storage: Google Cloud Storage Free Tier
7. Database: MongoDB Atlas Free Tier
8. Version Control & Deployment: GitHub + Vercel

**Notes:**

* Give each user has its own instance (with the use of Docker)
* Each command of the user will be executing in the docker container
* **User creates a playground -> Server -> Docker Container**
* **System is secure**, as if he/she will try to crash the system then its own docker container will affect only.
* **Scalability**: System will be scalable because each user will have its own docker container.

A computer screen with text

Description automatically generated

**Implementations:**

* **Support Multi-language of programming**
* **Allow the collaboration among the users;**