# Live Code Collaboration IDE Project Documentation

## Project Overview

A real-time collaborative code editor that allows multiple users to write, share, and execute code together in the browser. Think Google Docs meets VS Code, but lightweight and free to host.

## Core Features

### 1. Real-time Collaboration

* Multi-user editing with cursor presence
* Conflict resolution using CRDT (Conflict-free Replicated Data Types)
* Room-based collaboration with shareable URLs

### 2. Code Editor Features

* Syntax highlighting for multiple languages (Python, JavaScript, Java, C++)
* Real-time output display
* Basic IntelliSense-like features

### 3. Code Execution

Two approaches (choose based on complexity/hosting requirements):

#### Option A: Browser-based Execution (Simplest to host)

* JavaScript/TypeScript: Direct browser execution
* Python: Using Pyodide (WebAssembly Python runtime)
* HTML/CSS: Live preview
* Limitations: No system-level access, limited to browser capabilities

#### Option B: Serverless Execution (More powerful, still cheap)

* Use Judge0 CE (open source) for code execution
* Deploy on Railway’s free tier
* Support for 40+ programming languages
* Safe sandboxed environment

### 4. Development Environment

* File system for simple projects
* Console output
* Basic error handling
* Code sharing via URLs

## Technical Requirements

### Frontend Stack

* React.js for UI
* Monaco Editor (VS Code’s editor)
* Y.js for CRDT implementation
* Socket.io-client for real-time communication
* TailwindCSS for styling

### Backend Stack (All Free Tier)

* FastAPI on Railway (free tier)
* Redis Cloud (free 30MB)
* SQLite for database
* Judge0 CE (self-hosted on Railway free tier) if using Option B

## Deployment Costs Breakdown

Total Cost: $0-5/month

### Free Tier Options:

1. Frontend:
   * Vercel (free tier)
   * Netlify (free tier)
   * GitHub Pages (free)
2. Backend:
   * Railway (free tier includes 500 hours)
   * Fly.io (free tier includes 3 shared-cpu-1x apps)
   * Render (free tier)
3. Database:
   * SQLite (free, file-based)
   * Supabase (free tier)
4. Real-time:
   * Redis Cloud (free 30MB tier)

### Scaling Considerations

* Free tiers sufficient for development and small-scale use
* Can handle 50-100 concurrent users without paid upgrades
* Optional paid upgrades only needed for heavy usage

### Infrastructure

* WebSocket server for real-time communication
* Code execution service with isolation
* Authentication service
* Database service

## API Endpoints

### Authentication

POST /api/auth/register  
POST /api/auth/login  
POST /api/auth/logout

### Rooms

POST /api/rooms/create  
GET /api/rooms/{room\_id}  
POST /api/rooms/{room\_id}/join

### Code Execution

POST /api/execute  
GET /api/execution/{execution\_id}/status  
GET /api/execution/{execution\_id}/output

### WebSocket Events

room:join - User joins a room  
room:leave - User leaves a room  
cursor:update - Cursor position update  
text:update - Text content update  
execution:start - Code execution started  
execution:output - Execution output received

## Database Schema

### Users

CREATE TABLE users (  
 id UUID PRIMARY KEY,  
 username TEXT UNIQUE,  
 email TEXT UNIQUE,  
 password\_hash TEXT,  
 created\_at TIMESTAMP  
);

### Rooms

CREATE TABLE rooms (  
 id UUID PRIMARY KEY,  
 name TEXT,  
 language TEXT,  
 created\_by UUID REFERENCES users(id),  
 created\_at TIMESTAMP  
);

### RoomParticipants

CREATE TABLE room\_participants (  
 room\_id UUID REFERENCES rooms(id),  
 user\_id UUID REFERENCES users(id),  
 joined\_at TIMESTAMP,  
 PRIMARY KEY (room\_id, user\_id)  
);

## Implementation Details

### Real-time Sync

* Use Y.js for text synchronization
* Implement operational transformation for conflict resolution
* Use Redis pub/sub for WebSocket scaling

### Code Execution

* Docker containers for isolated execution
* Resource limits and timeouts
* Support for multiple languages
* Package management within containers

### Security Considerations

* Rate limiting for API and WebSocket connections
* Input sanitization for code execution
* Room access control
* Secure WebSocket connections

## Deployment Architecture

Client -> CloudFlare -> Load Balancer -> WebSocket Servers  
 -> API Servers  
 -> Code Execution Workers  
 -> Redis  
 -> Database

## Future Enhancements

1. Git integration
2. Custom themes
3. More language support
4. Advanced IDE features (debugging, breakpoints)
5. Collaborative drawings/diagrams
6. Voice/video chat

## Development Phases

### Phase 1: Basic Editor

* Basic editor setup with Monaco
* Room creation and joining
* Real-time text sync
* User presence

### Phase 2: Code Execution

* Code execution environment
* Output streaming
* Basic language support
* File system implementation

### Phase 3: Advanced Features

* More languages
* Package management
* Terminal access
* Version control

### Phase 4: Polish

* UI/UX improvements
* Performance optimizations
* Security hardening
* Documentation

## Resource Requirements

* Free tier services sufficient for development
* Minimal hosting costs ($5-10/month)
* No special hardware needed

This documentation serves as a blueprint for implementing a collaborative IDE. Each component can be built incrementally, and the system can scale based on needs.