# Project Title: Online Code Editor

## Table of Contents

1. \*\*Introduction\*\*<br>

1.1 Purpose

1.2 Scope

1.3 Overview

2. \*\*Technologies Used\*\*<br>

2.1 Next.js

2.2 TypeScript

2.3 CodeMirror

2.4 Socket.IO

2.5 NextAuth.js

3. \*\*System Architecture\*\*<br>

3.1 Frontend Architecture

- 3.1.1 Next.js Pages and Components

- 3.1.2 CodeMirror Integration

3.2 Backend Architecture

- 3.2.1 Socket.IO Server

- 3.2.2 Authentication with NextAuth.js

4. \*\*User Authentication\*\*<br>

4.1 NextAuth.js Configuration

4.2 User Registration

4.3 Authentication Flow

5. \*\*Code Editor Features\*\*<br>

5.1 CodeMirror Configuration

5.2 Real-time Collaboration

- 5.2.1 Socket.IO Integration

- 5.2.2 Operational Transformations for Collaboration

6. \*\*Communication Flow\*\*<br>

6.1 User Connection

- 6.1.1 Authentication Flow

- 6.1.2 Socket.IO Connection

6.2 Real-time Code Editing

- 6.2.1 Sending and Receiving Code Changes

- 6.2.2 Broadcasting Changes to Connected Users

7. \*\*Data Storage\*\*<br>

7.1 Persistent Data Storage

- 7.1.1 Saving User Sessions and Code Changes

7.2 Database Integration (if applicable)

8. \*\*Conclusion\*\*<br>

8.1 Summary

8.2 Future Enhancements

## Introduction

### 1.1 Purpose

The purpose of this document is to provide a detailed design for the implementation of an online code editor using Next.js, TypeScript, CodeMirror, Socket.IO, and NextAuth.js.

### 1.2 Scope

The scope of this project includes the development of a real-time collaborative code editor that allows users to edit code simultaneously. The document outlines the architecture, technologies, and features required for the successful implementation of the project.

### 1.3 Overview

The online code editor will be built using Next.js as the frontend framework, TypeScript for type-checking and enhanced developer experience, CodeMirror as the core code editor component, Socket.IO for real-time communication, and NextAuth.js for user authentication.

## Technologies Used

### 2.1 Next.js

Next.js will serve as the frontend framework, providing server-side rendering, routing, and an optimized development experience.

### 2.2 TypeScript

TypeScript will be used to enhance code quality, provide static typing, and improve the overall maintainability of the codebase.

### 2.3 CodeMirror

CodeMirror is a versatile text editor implemented in JavaScript for the browser. It will be integrated to provide the core functionality of the code editor.

### 2.4 Socket.IO

Socket.IO will enable real-time bidirectional event-based communication between the server and clients, facilitating collaborative code editing.

### 2.5 NextAuth.js

NextAuth.js will handle user authentication, providing a secure and customizable authentication flow.

## System Architecture

### 3.1 Frontend Architecture

#### 3.1.1 Next.js Pages and Components

The frontend will consist of multiple pages and components built using Next.js. Key pages include the code editor page, authentication pages, and user dashboard. Components will include the CodeMirror editor and other UI elements.

#### 3.1.2 CodeMirror Integration

CodeMirror will be integrated into the project to provide a feature-rich and customizable code editing experience. Syntax highlighting, autocompletion, and other CodeMirror features will be utilized.

### 3.2 Backend Architecture

#### 3.2.1 Socket.IO Server

A Socket.IO server will be implemented on the backend to manage real-time communication between connected clients. It will handle events related to code changes and user collaboration.

#### 3.2.2 Authentication with NextAuth.js

NextAuth.js will be integrated into the backend to manage user authentication. It will provide a secure authentication flow, including social authentication options.

## User Authentication

### 4.1 NextAuth.js Configuration

NextAuth.js will be configured to use various authentication providers (email/password, github ).

### 4.2 User Registration

The user registration process will involve collecting necessary information and securely storing user data.

### 4.3 Authentication Flow

The authentication flow will be designed to ensure the security of user accounts and seamless integration with the online code editor.

## Code Editor Features

### 5.1 CodeMirror Configuration

CodeMirror will be configured to meet project requirements, including language support, theme customization, and integration with other functionalities.

### 5.2 Real-time Collaboration

#### 5.2.1 Socket.IO Integration

Socket.IO will be utilized to enable real-time collaboration features, allowing multiple users to edit code simultaneously.

## Communication Flow

### 6.1 User Connection

#### 6.1.1 Authentication Flow

The authentication flow will include secure user login and session management, ensuring that only authenticated users can access the online code editor.

#### 6.1.2 Socket.IO Connection

Once authenticated, users will establish a Socket.IO connection to the server for real-time communication.

### 6.2 Real-time Code Editing

#### 6.2.1 Sending and Receiving Code Changes

Users will send and receive real-time code changes via Socket.IO events, triggering updates on the CodeMirror editor.

#### 6.2.2 Broadcasting Changes to Connected Users

The server will broadcast code changes to all connected users, maintaining synchronization and collaboration.

## Data Storage

### 7.1 Persistent Data Storage

User sessions, code changes, and other relevant data will be persistently stored to ensure a seamless user experience.

### 7.2 Database Integration

Integrating MongoDB for storing user information such as name, email addresses and password.

## Conclusion

### 8.1 Summary

This document provides a comprehensive design for the implementation of an online code editor, covering architecture, technologies, features, and various aspects of development and deployment.

### 8.2 Future Enhancements

Potential future enhancements, such as code compilation, integrating webrtc for audio, including a whiteboard space for better discussions.