Software Requirements Specification (SRS)

Project Title:- Real –Time Collaborative Cloud IDE with Project Repositories and Access Control

1. Introduction

1.1 Purpose

This SRS document defines the complete requirements for the project *"Real-Time Collaborative Cloud IDE with Project Repositories and Access Control."*

The purpose is to build a browser-based cloud IDE that allows users to:

a. Create and manage coding projects (repositories)

b. Collaborate in real time with access permissions

c. Use a built-in code editor and terminal

e. Enable secure login via Email, Google OAuth, or GitHub OAuth.

1.2 Stakeholders

Stakeholder	Role
End Users	→ Developers, students, or professionals collaborating on code in real time
System Admin	→Manages user data, repositories, and server configurations
Project Owner	→ Responsible for project design, architecture, and deployment
Developers	→ Build and maintain MEAN stack modules

1.3 Scope

The system provides an “online collaborative IDE” similar to VS Code Live Share, enabling:

a. User authentication (Email, Google, GitHub)

b. Project creation & GitHub integration

c. Real-time code editing and terminal access

d. Collaboration via email-based access permissions

e. View/Edit access control

f. Project history tracking and local project imports.

1.4 Definitions, Acronyms & References

Term	**Description**
IDE	→ Integrated Development Environment
OAuth	→ Open standard for access delegation
MEAN	→ MongoDB, Express, Angular, Node.js
Socket.io	→ Library enabling real-time, bi-directional communication
Monaco Editor	→ The editor engine used in Visual Studio Code

2. Overall Description

2.1 Product Perspective

The IDE functions as a **cloud-based platform** integrating:

Frontend: Angular for UI

Backend: Node.js with Express.js

Database: MongoDB for user/project data

Real-time communication: Socket.io

Code engine: Monaco Editor

System Flow:

1. User visits the base URL → https://codeCollab.com (landing page)
2. Landing page contains:

   * Navbar (Login/Register & more)
   * Hero Section
   * About IDE
   * Testimonials
   * “Trusted by” Company Logos
   * FAQ Section
   * Footer
3. After login → user redirected to personalized workspace
4. Workspace contains: Sidebar (Projects, GitHub Repo, History, Open Local Project), Editor area, Footer.

2.2 User Interfaces

1. Landing Page

* Responsive layout with sections: Hero, About, Testimonials, FAQ.
* Login/Register button on Navbar.

2. Workspace UI

Sidebar:→ Create Project, GitHub Repo, History, Open Local Project

Topbar:→ Search, Light/Dark mode toggle, Profile menu (e.g., PK)

Main Editor:→ Monaco-based coding area

Footer:→ Status display (e.g., MongoDB Connected, User ID)

2.3 System Interfaces

Database Interface:→ MongoDB for storing users, repositories, and access permissions.

External APIs:→ GitHub API for repo import/export, Google OAuth for login.

Socket Interface:→ Real-time data sync for collaborative editing.

2.4 Constraints

Requires stable internet connection.

Compatible with modern browsers only.

Real-time sync limited by Socket.io performance.

2.5 Assumptions and Dependencies

Assumes all users have valid email IDs.

Depends on GitHub and Google OAuth API availability.

Uses cloud storage (MongoDB Atlas / Firebase) for persistence.

3. System Features

3.1 Functional Requirements

ID	Feature	Description

a. FR-1 → User Registration & Login | New users can register via email or OAuth. Returning users log in with credentials.

b. FR-2 → Project Management | Users can create, open, edit, or delete projects.

c. FR-3 → Real-Time Collaboration | Multiple users can edit the same file simultaneously using Socket.io.

d. FR-4 → Access Control | Users can send collaboration invites to specific emails with permissions: View / Edit / Both.

e. FR-5 → GitHub Integration | Users can import or push projects to GitHub repositories.

f. FR-6 → File Explorer | Displays directory structure like VS Code (folders/files).

g. FR-7 → Code Editor | Monaco-based editor supporting syntax highlighting, themes, and autocompletion.

h. FR-8 → Built-in Terminal | Allows code execution or basic commands (Node/JS runtime).

i. FR-9 → History & Logs | Displays recent activities and changes.

j. FR-10 → Light/Dark Theme | User preference toggle saved in profile.

3.2 Use Case Scenarios

Use Case 1: New User Registration

Actors: New User

Steps:

1. Visit base URL → click "Register"
2. Enter email, password or select "Login with Google/GitHub"
3. Verify account → redirected to Workspace

Use Case 2: Project Collaboration

Actors: User A (owner), User B (invitee)

Steps:

1. User A opens project
2. Sends invite to User B’s email with “edit” access
3. User B receives mail → accepts
4. Both edit files in real-time via Socket.io

Use Case 3: Open Local Project

Actors: Existing User

Steps:

1. Click “Open Local Project” on sidebar
2. Choose local directory
3. Files load into Monaco editor

3.3 External Interface Requirements

Interface	Description
Google OAuth API	For authentication
GitHub API	For repo management
Socket.io	For real-time collaboration
MongoDB Atlas	For data storage

3.4 Logical Database Requirements

Collections:

1. *Users*

```
  {
 "_id": "ObjectId",
 "name": "Prashant",
 "email": "pk@gmail.com",
 "authProvider": "google | github | email",
 "passwordHash": "...",
 "projects": ["projectId1", "projectId2"],
 "createdAt": "2025-10-22T12:00:00Z"
}
```

2. *Projects*

```
  {
 "_id": "projectId",
 "name": "RealTime IDE",
 "owner": "userId",
 "description": "A real-time collaborative IDE",
 "structure": [
   { "path": "src/index.js", "fileId": "fileId1" },
   { "path": "src/utils/helper.js", "fileId": "fileId2" }
 ],
 "collaborators": [
   { "email": "x@gmail.com", "permission": "edit" }
 ],
 "createdAt": "2025-10-22T12:00:00Z",
 "updatedAt": "2025-10-22T15:00:00Z"
```

```
}
```

3. *Files Collection*

```
{
  "_id": "fileId1",
  "projectId": "projectId",
  "path": "src/index.js",
  "content": "console.log('Hello, world!');",
  "language": "javascript",
  "createdBy": "userId",
  "updatedBy": "userId",
  "createdAt": "2025-10-22T12:00:00Z",
  "updatedAt": "2025-10-22T13:00:00Z"
}
```

3. *History*

```
   {
  "_id": "ObjectId",
  "projectId": "projectId",
  "fileId": "fileId1",
  "timestamp": "2025-10-22T12:00:00Z",
  "action": "file_update | file_create | file_delete | collaborator_add | project_create",
  "userEmail": "pk@gmail.com",
  "changes": {
    "oldContent": "console.log('hi');",
    "newContent": "console.log('Hello, world!');"
  }
}
```

3.5 Non-Functional Requirements

Type	Description
Performance	Real-time editing latency < 200ms
Scalability	Should support 50+ concurrent sessions
Reliability	Auto-save every 10 seconds
Security	Encrypted JWT tokens, HTTPS, OAuth2
Availability	99.5% uptime expected
Usability	Responsive design, keyboard shortcuts, theme options
Maintainability	Modular MEAN architecture with RESTful APIs