Website: https://cipherschools.com | **Email**: support@cipherschools.com | **Contact**: +91-7206810246

CipherStudio - Helping Guide

Project Title: CipherStudio — Online React IDE

Objective

Build a simplified **React-based coding playground**, where users can write, run, and save React code directly in the browser — similar to *NextLeap.js* or *CodeSandbox*.

Tech Stack

Component	Recommended Tool
Frontend Framework	React/NextJS
Code Execution	Sandpack by CodeSandbox
Editor	Monaco Editor or Sandpack built-in editor
Database	MongoDB Atlas and AWS S3
Backend	Node.js / Express.js
Deployment	Vercel (frontend) + Render / Railway / Cyclic (backend)

Architecture Overview

1. Frontend (ReactJS/Next.js)
    - Displays an IDE-like interface with three primary sections:
        1. File Explorer (left) - shows user-created files.
        2. Code Editor (center) - allows editing code.
        3. Live Preview (right) - powered by Sandpack.
    - Handles login, project management, and API calls.

2. Backend (Node.js / Express)
    - REST APIs to handle:

    - Save project
    - Fetch saved projects
    - Update project
    - Delete project

3. Database (MongoDB & AWS S3)
    - Stores users, user projects, file structures, etc on mongodb.
    - Stores files on AWS S3 storage.

Suggested MongoDB Schema

1. **Users collection**

users

```
{
  _id: ObjectId,
  firstName: String,
  lastName: String,
  email: { type: String, unique: true, index: true },
  password: String,  // hashed password (bcrypt)
  mobile: String,
  createdAt:  timestamp,
  updatedAt: timestamp,
  lastLoggedIn: timestamp,
  settings: {
    theme: { type: String, enum: ["light", "dark"], default: "light" },
   }
}
```

2. **Projects collection**

projects

```
{
  _id: ObjectId,
  projectSlug: { type: String, unique: true, index: true }, //
public-friendly slug
  userId: { type: ObjectId, ref: "users" }, //null for non-auth
  name: String,
  description: String,
  rootFolderId: { type: ObjectId, ref: "files" }, // top-level folder
  createdAt: timestamp,
  updatedAt: timestamp,
  settings: {
    framework: { type: String, default: "react" },
    autoSave: { type: Boolean, default: true },
  },
}
```

3. Files collection

files

```
{
  _id: ObjectId,
  projectId: { type: ObjectId, ref: "projects" },
  parentId: { type: ObjectId, ref: "files", default: null }, // null for root folder
  name: String,
  type: { type: String, enum: ["folder", "file"], required: true },

  // Only applicable for files
  s3Key: String, // e.g."projects/<projectId>/src/App.js"
  language: String,       // "javascript", "jsx", "css", etc.
  sizeInBytes: Number,    // file size

  createdAt: timestamp,
  updatedAt: timestamp
}
```

Folder & File Example

Imagine a project structure like this:

```
MyProject/
│
├─ src/
│    ├─ index.js
│    ├─ App.js
│    └─ components/
│         ├─ Navbar.js
│         └─ Footer.js
├─ public/
│    └─ index.html
└─ package.json
```

How it maps to MongoDB files collection

_id	projectId	parentId	name	type	s3Key
f1	p1	null	MyProject	folder	-
f2	p1	f1	src	folder	-
f3	p1	f2	index.js	file	projects/p1/src/index.js
f4	p1	f2	App.js	file	projects/p1/src/App.js
f5	p1	f2	components	folder	-
f6	p1	f5	Navbar.js	file	projects/p1/src/components/Navbar.js
f7	p1	f5	Footer.js	file	projects/p1/src/components/Footer.js

f8	p1	f1	public	folder	-
f9	p1	f8	index.html	file	projects/p1/public/index.html
f10	p1	f1	package.json	file	projects/p1/package.json

Explanation:

- The Root folder (MyProject) has parentId: null.
- src and public are children of the root folder.
- components is a child of src.
- Each folder or file is a separate document → supports unlimited nesting.
- s3Key is used only for files; folders don't need S3 storage.

Visual ERD / Hierarchy

```
Users (1) ──< Projects (many)
Projects (1) ──< Files (many)
Files:
    type = "folder" → parentId = <folderId>
    type = "file" → s3Key points to S3 storage
```

UI Design Guidance

You can take UI inspiration from NextLeap.js or CodeSandbox layouts.

Sandpack Setup (Example Code)

```
npm install @codesandbox/sandpack-react
```

Example Component

```
import { Sandpack } from "@codesandbox/sandpack-react";
import { SandpackThemeProvider } from "@codesandbox/sandpack-react";

export default function IDE() {
  return (
    <Sandpack
      template="react"
      options={{
        showTabs: true,
        showLineNumbers: true,
        wrapContent: true,
      }}
    />
  );
}
```

This will instantly give you a working React playground inside your app.
You can later integrate your own file manager and connect it to APIs.

Sample API Endpoints

Endpoint	Method	Description
/api/users	POST	Create a new user
/api/users/login	POST	Authenticate user and return JWT token
/api/projects	POST	Create a new project
/api/projects/:userId	GET	Get all projects of user
/api/projects/:id	GET	Fetch project by ID
/api/projects/:id	PUT	Update project details or files
/api/projects/:id	DELETE	Delete a project
/api/files	POST	Create a new file or folder in a project
/api/files/:id	GET	Fetch file/folder details by ID
/api/files/:id	PUT	Update file content or rename folder/file
/api/files/:id	DELETE	Delete a file or folder