

All steps are written properly inside backend and frontend folder:

BACKEND:

1-Install Node.js

2-Initialise an empty typescript project

npm init -y

npm install typescript

npx tsc --init

example : index.ts

npx tsc -> convert ts to js output to folder mentioned in tsconfig.json

3-Basic typescript configuration

Change rootDir to src

Change outDir to dist for the pro

4-Add Dependencies

npm install express @types/express nodemon ts-node aws-sdk dotenv node-pty socket.io cors

node-pty // helps creating pseudo terminal which browser can talks to

aws-sdk -> gives s3 object store

5-Initialize a simple express app in index.ts listening on port 3000

// build any project npx tsc -b TS->JS DIST mein convert

6-Build one /project POST method : Lines commented for understanding

takes parameters from frontend replID , baseCodeType

6-0 create http.ts file and inside this pass app:Express where all api will be made

6-0-0 create http Server using app Express and listen on that This server will be used by websocket later on

6-1 Copy S3 download repl/base/nodejs from s3 to backend and make code/replID on S3 as well

6-2 aws.ts create usme s3Dwonlaod function create

7-CopyS3Download SourcePrefix & DestinationPrefix Steps : Basically actual file s3 hi hai waha se s3 mein copy dusre destination pe : no need to download to my computer

await copyS3Folder(`base/${language}`, `code/${replId}`);

a) Take all keys for source prefix

bucket-repl

source prefix- /base/nodejs (passed)

get all keys inside this prefix await s3.listObjectsV2(listParams).promise();

b) Now for each key map and change its source address to destination addrees

// object key looks like /base/nodejs/index.ts from repl bucket

// change /base/nodejs to /code/replID/index.ts

// now this bucket repl mein copy the same content file from source to destination

// object key is /code/replID/index.ts

let destinationKey = object.Key.replace(sourcePrefix, destinationPrefix);

let copyParams = {

Bucket: "repl",

CopySource: `repl/${object.Key}`, // copy source is content file acutal file is repl/base/nodejs/index.ts this content copied to

Key: destinationKey // repl/code/replId/index.ts

};

console.log(copyParams);

\*\*\* await s3.copyObject(copyParams).promise(); // to call that params copy

8-Initialise websocket server -> Read socket.io.txt for backend Setup

initWs(httpServer); //pass for io connection

io.connection banaya // repl id frontend se connection url mein query mein passkar diya

9-Fetch S3Downloaded using repl ID and send this complete folder via socket emit loaded

replID se 7th step mein copyS3 mein kiye the code/replId on cloudfare code hai

await fetchS3Folder(`code/${replId}`, path.join(\_\_dirname, `../tmp/${replId}`));

//locally save that folder

//Code fetchS3Folder inside aws.ts waha se export

a)await s3.listObjectsV2(params).promise(); //get all keys

keys map file => file.key nikala const data = await s3.getObject(getObjectParams).promise();

//data store kiya

b)const fileData = data.Body; //actual data hai yeh

b)\*\*\*\* // fileData is S3.Body need to convert it to Buffer for fs.write to use

now file Data write inside local folder filePath = thoda edit kiya using path

c)await writeFile(filePath, fileData);

Create folder fs use karke fs.mkdir filePath recursively folder ke andar folder gaye

us filePath pe fs.write(filePaht,fileData)

10-Socket.emit(all Local folder downloaded)

//only dir pass kar rahe folder and files jo first layer mein hai

await fetchDir(local folder path) fs.ts

fs.readdir(folder path) => allFiles => map((file)=>{new fommat}) and return via resolve

file => ({ type: file.isDirectory() ? "dir" : "file", name: file.name, path: `${baseDir}/${file.name}` }))

// return type,file.name,path kya hai // use by frontend sidebar to distinguish between dir and file

{type: 'file', name: 'index.js', path: '/index.js'}

1

:

{type: 'file', name: 'package-lock.json', path: '/package-lock.json'}

2

:

{type: 'file', name: 'package.json', path: '/package.j

11-Frontend-> Socket more connection:

a)socket?.emit("fetchDir,filePath,(callback jo return from socket.on)=>{here data direct})

socket?.emit("fetchContent", { path: file.path }, (data: string)={})

OnSelect Frontend (file bhejega us path ka fileStructure(TypeDirectory) or actualContentFile (Type file))

i) fetchDir Calls function in fs.ts take filePath as argument

fs.readDir Upper Layer read ans same json fileStructure frontend ko samjh aaye send

resolve(files.map(file => ({ type: file.isDirectory() ? "dir" : "file", name: file.name, path: `${baseDir}/${file.name}` })));

ii) fetchContent Calls function in fs.ts filePath as argument

fs.readFile( actual data = read karega)

send as data:string to frontend => selectedFile (data) set kar do (Code Component Show karega)

b)Code Component showing selected File whenever value is changed

socket.emit("updateContent", { path: selectedFile.path, content: value });

// 1st Locally update file on path: new conent value

// 2nd on S3 update kar do

Backend -> scoket.on ("update content)

const fullPath = path.join(\_\_dirname, `../tmp/${replId}/${filePath}`); //local Path

await saveFile(fullPath, content);

await saveToS3(`code/${replId}`, filePath, content); //s3 Path

i) saveFile => uses fs.write(filePath,conent)

ii) saveToS3 => s3.putObject(params).promise() //Key/Path and Body:conent new kya hai

// update the existing filePath to new conent

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Terminal and Ouput Window\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

12- pty.ts direct mila xterm-socket-io github se

Imports fork and IPty from the node-pty library.

fork is used to create a new pseudo-terminal,

and IPty represents the pseudo-terminal interface.

const SHELL = "bash";: Defines the shell (in this case, bash) that will be used in the pseudo-terminal.

i) private session ={id,{terminal,repid}} initally constuctor ={ } empty

ii) createPty(socket.id,repid, And this method onData: (data: string, id: number) => void)){

create term = {SHELL, cwd: path.join (replId)//jaha kholna hai terminal}

}

$$$$ term.on('data', (data: string) => onData(data, term.pid)); // toh yeh method call kar rahe

iii) create a session this.sessions[id]={terminal: term jo banaye and replID hai}

iv) write(terminalId: string, data: string) {

this.sessions[terminalId]?. terminal.write(data); // acts like term.write

}

v) clear(terminalId: string) {

this.sessions[terminalId].terminal.kill(); //term.kill

delete this.sessions[terminalId];

}

13- Socket connection for terminal : Repld is present at the very beginning while starting io.connection

i) const terminalManager = new TerminalManager(); isko use karenge for this socket connection

ii) socket.on("requestTerminal") whenever frontend ready terminal xterm toh yeh bhejega so that backend apna terminal bana left

terminalManager.createPty(socket.id, replId, (data, id) => {

socket.emit('terminal', {

data: Buffer.from(data,"utf-8")

});

});

// create backend terminal and jo term.on hua backend mein us data ko socket.emit karke bhej do

iii) term.on('data', (data: string) => onData(data, term.pid)); STep12

isme Ondata is attached toh

backend terminal mein data change hoga uska term.on chaleg

OnData => socket.emit('terminal', {

data: Buffer.from(data,"utf-8")

});

// frontend terminal write hoga aur display hoga

This way term.on term.write both terminal backend and frontend

using socket iske saath communicate and backend terminal actual working and frontend terminal only displaying

FRONTEND:

1- Setup react + typescript project

npm create vite@latest

npm install

npm run dev // see pacakge.json

2-Create Landing Page

Send projectId, language to backend app.post("/project") await copyS3Folder() copy s3 base code to code/projectID

setLanguage -> options input

setReplID ->random generate or type

Button ->

used loading useState baar baar click na ho

onClick call to backend api with languge and replID

navigate to next page with reply id in http:url?=replID yeh waha us page pe extract kar lenge

const [searchParams] = useSearchParams();

const replId = searchParams.get('replId') ?? '';

3-Create Coding Page

a) // setup websocket for io connection created a useState<Socket>

whenever replId is changed creating new socket connection and saving that socket so that it can be used from anywhere

const socket= useSocket(replId); useSocket is using IO connection to backend

b) socket.on("loaded",takes files loaded from backend into its local folder )

{type: 'file', name: 'index.js', path: '/index.js'}

1

{type: 'dir', name: 'CheckDir', path: '/CheckDir'}

:

{type: 'file', name: 'package-lock.json', path: '/package-lock.json'}

2

:

{type: 'file', name: 'package.json', path: '/package.json)

// aisa backend ne map karke bheja hai all Files ko (type,name,path)

c) This fileContent ==== yahi type name path hai isme

export interface RemoteFile {

type: "file" | "dir";

name: string;

path: string;

}

d) const [fileStructure, setFileStructure] = useState<RemoteFile[]>([]);

//all files get from socket.load save here sabse imp all upper layer files and dir mile json mein

4- Editor Page(Open Source) SideBar + CodeWindow Basic useStates

-> CodeSandBox website se left panel and right panel JUST NEED fileStructure

https://github.com/codesandbox/codesandbox-client

https://github.com/hkirat/repl/tree/main/bad-code/frontend/src/components/external/editor

take socket, selectedFile(CodeWindow), setFileStructure(Sidebar/MoneroTree), onSelect(use for updating selected file and setFileStructure)

)

a) const [selectedFile, setSelectedFile] = useState<File | undefined>(undefined);

// this will be used inside code window

if(selected file == Type "file") socket.emit(fetchContent)

b) Made one method onSelect(file:File) and passed along with Editor everywhere

if(file==DIR) toh setFileStructure ko update more json load

else file==FILe toh content lao read karke backend se and

setSelectedFile : File hai

socket?.emit("fetchContent", { path: file.path }, (data: string) => {

IMP STEP file.content = data;

file update with actual content:string form

// us file mein data bhara as string passed from backend fs.readFile karke us path pe

setSelectedFile(file); //then set it

change used by code windows

c) interface File >= RemoteFile type name path hai aur data bhi hai

\*\*\* File has content : string

file.content= data(received from backend afetr fs.readFile)

5-Editor Page Open Source

a) using fileStructure [{type,name,path},{}] =>

const rootDir = useMemo(() => {

return buildFileTree(files); // since building time baar baar recalculation ho sakta hai

useMemo is used to memoize (like we do in Dynamic Programming, concept wise) and skip recalculation

// now use rootDir.files[0] : get file you want

Why RootDir Convert?

RootDir part of file-manager.tsx which is open source and jo bhi yeh file tree code banaya hai

woh file and direcrtoty TYPE DEFINED kiya hai file manager mein

That's why my file structure convert into their RootDir and simply use their code

// buildFileTree OpenSource => all RemoteFiles jo json tha unko hierachy plus apne defined DIRECTORY AND FILE CONVERT

The buildFileTree function constructs a hierarchical file tree from a flat list of RemoteFile objects

The buildFileTree function returns a Directory object representing the root directory of the constructed file tree.

This root directory includes nested directories and files, each with the correct parent-child relationships and depth values.

//Steps

first separates the input data into directories and files:

nitializes a cache (a Map) to store the constructed Directory and File objects and creates the root directory

\*\*\*\*//hamne pass type "dir" "file" yaha isne us array ko dir and files break aur inko map

apne interface type.directory and type.file mein kiya

//inputconst data: RemoteFile[] = [

{ type: "dir", name: "documents", path: "documents" },

{ type: "file", name: "file1.txt", path: "documents/file1.txt" },

{ type: "dir", name: "photos", path: "photos" },

{ type: "file", name: "photo1.jpg", path: "photos/photo1.jpg" }

];

//output RootDir=> json //hierachica;

{

"id": "root",

"name": "root",

"parentId": undefined,

"type": "DIRECTORY", //yeh define kar diya

"path": "",

"depth": 0,

"dirs": [

{

"id": "documents",

"name": "documents",

"path": "documents",

"parentId": "0",

"type": "DIRECTORY",

"depth": 1,

"dirs": [],

"files": [

{

"id": "documents/file1.txt",

"name": "file1.txt",

"path": "documents/file1.txt", //path wrt parents ho rahe

"parentId": "documents",

"type": "FILE", //yeh define kar diya

"depth": 2

}}

}

b)rootDir Mil gaya Json on hierachy Type.Directory=dir(backend) and Type.File(file Backend sended) hai ab iske andar

6-FileTree (CodeSandBox Code available)External Component Role: takes rootDir hierarchical order

Takes

selectedFile(here decide hoga),

onSelect Method(if folder then more hierarchy if file change selected File)

rootDir -> Convert this to visually Component

Process Summary-> FileTree

Initially, rootDir is passed to FileTree.

FileTree renders SubTree passing rootDir.

SubTree iterates over rootDir.dirs and rootDir.files, rendering each DirDiv and FileDiv.

Clicking on a FileDiv triggers onSelect(file) directly, setting selectedFile.

Clicking on a DirDiv toggles its open state and triggers onSelect(directory) to propagate the selection.

rootDir={

type, name ,dirs[],files[] array rakha tha inka

}

rootDir.dirs => UI Component DirDiv => OnClick Laga hai send click dir

rootDir.files => UI Component FileDiv =>OnClick Laga hai send click file

OnSelect (file mila)=>file type basis selectedFile || fileStructure change hoga

7-Code Component: CodeSandBox OpenSource :

takes selectedFile and Socket

import Editor from "@monaco-editor/react";

use <Editor as main component direct

code = selectedFile.content hai

value = {code}

onChange debounce(0.5seconds ka wait karke fir call inside)

=> USES setTimeout internally => Then jo debouce(function) 1st timeout then function use after delay

// change the fileContent with new value

// 1st locally then on s3

socket.emit("updateContent", { path: selectedFile.path, content: value });

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Terminal and Ouput Window\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

8-Output Component

<iframe hai src="localhost:3001" is 3001 pe run hoga repld wala app

Idea:can customize accordingly contenxt provide use karke port dynamic kar sakte

9- Terminal Component Coding Page -> Xterm with socket io Github

https://github.com/jpcweb/xtermjs-socketio

a) Terminal window is completely created by xterm

just create and return <div> ref={terminalRef} and give this ref to xterm and terminal will be

fit inside this div using FitAddon

Binds the terminalRef to the div element where the xterm.js terminal will be rendered (<div ref={terminalRef}>).

useRef Hook allows you to persist values between renders.

const terminalRef = useRef(); terminal.current initial value

b) Creating terminal using Xterm and fitting it inside div

import { Terminal } from "xterm";

const term = new Terminal(OPTIONS\_TERM) // creating new terminal //option terms terminal looks tells

term.loadAddon(fitAddon); //Adds the FitAddon to the terminal instance, which ensures the terminal fits within its container.

term.open(terminalRef.current); //using that <div component and set terminal in that

fitAddon.fit();

c)Communication term.on() and term.write main hai terminal input apne paas and output dena terminal ko

i) term.on(whatenever we write on terminal that (data)=>{

socket.emit(backend send kar do)

})

ii) Backend send ouput

1st socket.on("terminal", terminalHandler) catch

function terminalHandler({ data }) {

if (data instanceof ArrayBuffer) {

console.error(data);

console.log(ab2str(data)) //convert to string

$$$$$$$$$$$$$ term.write(ab2str(data)) // display that data to frontend terminal

}

}

iii) socket.emit("requestTerminal"); starting hua tab bata do

//tell backend that backend can start creating its own terminal

iv) return () => {

socket.off("terminal")

}

// when this terminal component closes

//socket close and send msg to backend to close its pseudo terminal

SUMMARY->   
Builded an online IDE where user will get ready made template of language they want to use, they can code live on the platform whose all data is managed using cloudfare s2, also a terminal is provided where they can actually run there code and test.