# Create back-end project

Initialize: npm init -y

In the initial setting below, “main”: descript the entry point to the project, changed it to “app.js” file (alternative main.js or index.js).

Text

Description automatically generated with medium confidence

## Install typescript

npm i typescript

Potential bug:

A screenshot of a computer

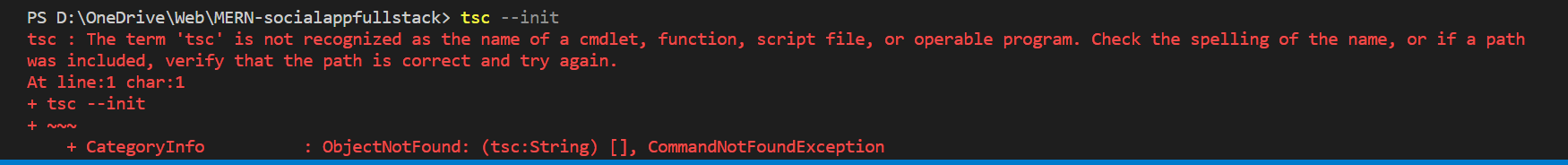
Description automatically generated

Fix: try turn off and on the wifi connection of your pc. <https://github.com/nodejs/node/issues/41056>

## Create typescript configuration file

tsc –init

Potential bug:



cause and fixed: typescript is not installed globally,

do npm install typescript@latest -g

<https://bobbyhadz.com/blog/typescript-tsc-is-not-recognized-as-internal-or-external>

## TSConfig setting

## Folder structure set up

src root folder

src\features to store all the features code

src\shared shared code

src\shared\global

src\shared\services all mongo db related services

src\shared\services\db code to connect to the mongo db

src\shared\sockets all thing related to socket io

src\shared\workers for redis message queue

src\app.ts entry point for this project, file name must be the same as "main": "app.js" in package.json file

src\config.ts configuration for this project

src\routes.ts define all the routes

src\setupDatabase.ts set up the mongo db

src\setupServer.ts set up the server

## Set up Server class

### Install express

npm i express

add following import:

import {Application, json, urlencoded, Response, Request, NextFunction } from 'express';

## Set up standard middleware

install

npm I cors helmet hpp cookie-session compression express-async-errors http-status-codes

helmet : a security library

store minimum data into cookie

need express-async-errors to cache error coming from async method

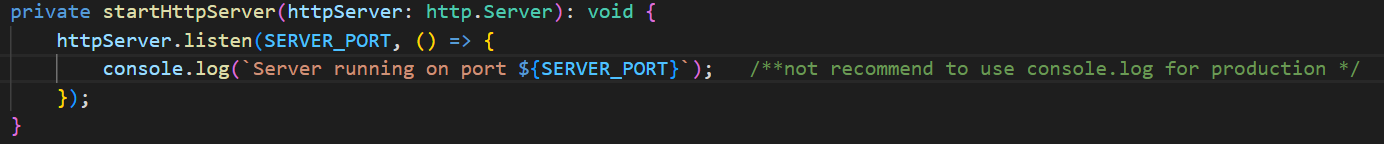
Text

Description automatically generated

## Set up http server

set server\_port constant to (any number after 1000 to 64000). except

Note: in typescript, use ` ${variable}` if want to include variable



set startServer to async, any async method will return Promise

Text

Description automatically generated

update app.ts



### install nodemon

npm i -g nodemon

add script to package.json, note: if we specify “dev”. later on if we call npm run dev, it will run the script at the end

A screenshot of a computer

Description automatically generated with medium confidence

### install ts-node

so that we can decode ts

npm i ts-node -D

### install tsconfig-paths

so all folder can be relative and still read by the typescript

<https://www.npmjs.com/package/tsconfig-paths>

npm install --save-dev tsconfig-paths

Text

Description automatically generated

## set up database

install mongoose:

npm i mongoose

Text

Description automatically generated

## set up environment config file

install dotenv

npm i dotenv

create a .env file in the root folder,

save your configurations constant

Text

Description automatically generated

## set up socket io

<https://www.npmjs.com/package/@socket.io/redis-adapter>

npm i @socket.io/redis-adapter redis socket.io

note: run command with @ in CMD, running in powershell will give error

<https://stackoverflow.com/questions/46107955/open-cmd-in-the-visual-studio-code-terminal>

add following imports to setupServer.ts

import { Server } from 'socket.io';

import { createClient } from 'redis';

import { createAdapter } from '@socket.io/redis-adapter';

add this constant to .env file : REDIS\_HOST = 'redis://localhost:6379'

update the method:

A screenshot of a computer

Description automatically generated

update startServer()

Text

Description automatically generated

## Set up route.ts file

A screenshot of a computer

Description automatically generated with medium confidence

update setupServer.ts

Text

Description automatically generated

## Set up Global error handler

create a file name error-handler under folder globals\helpers

define some custom error class as below  
Text

Description automatically generated

## logger set up

install bunyan

npm install bunyan

<https://www.npmjs.com/package/bunyan>

Graphical user interface, text

Description automatically generated

import Logger from ‘bunyan’;

add following const at the start of each ts file

/\*\* indicate the log is coming from the database \*/

const log: Logger = config.createLogger('setupDatabase');

then call the

log.error() or log.info to log the

## ESLint set up

install ESLint, editorconfig, prettier for vs code

create a file name .editorconfig under root folder

copy content from the website (<https://editorconfig.org/>) to editorconfig file.

npm i -D eslint eslint-config-prettier prettier @typescript-eslint/eslint-plugin @typescript-eslint/parser

create a file name .prettierrc.json, copy following config into the file

{

  "trailingComma": "none",

  "tabWidth": 2,

  "semi": true,

  "singleQuote": true,

  "bracketSpacing": true,

  "printWidth": 140

}

create a file name .eslintrc.json, copy the following config into the file

{

  "root": true,

  "parser": "@typescript-eslint/parser",

  "plugins": [

    "@typescript-eslint"

  ],

  "extends": [

    "eslint:recommended",

    "plugin:@typescript-eslint/recommended",

    "prettier"

  ],

  "parserOptions":  {

    "ecmaVersion":  2020,  // Allows for the parsing of modern ECMAScript features

    "sourceType":  "module"  // Allows for the use of imports

  },

  "rules": {

    "semi": [2, "always"],

    "space-before-function-paren": [0, {"anonymous": "always", "named": "always"}],

    "camelcase": 0,

    "no-return-assign": 0,

    "quotes": ["error", "single"],

    "@typescript-eslint/no-non-null-assertion": "off",

    "@typescript-eslint/no-namespace": "off",

    "@typescript-eslint/explicit-module-boundary-types": "off"

  }

}

add script to package.json file

## change absolute import

npm i ttypescript typescript-transform-paths

add following to tsconfig.json file

    "plugins": [

      { "transform": "typescript-transform-paths" },

      { "transform": "typescript-transform-paths", "afterDeclarations": true },

    ],                                                  /\* Declaration to use transform-path\*/

    "paths": {

      "@global/\*": ["src/features/shared/globals/\*"],

      "@service/\*": ["src/features/shared/services/\*"],

      "@socket/\*": ["src/features/shared/sockets/\*"],

      "@worker/\*": ["src/features/shared/workers/\*"],

      "@root/\*": ["src/\*"],   /\*make sure root directory is the last \*/

    }

# Note on DataFlow

The data will go to both Redis and Queue (to be stored in MongoDB).

Redis is like a memory cache (which respond faster) – hence good for storing data frequently used and expected immediate response.

A picture containing diagram

Description automatically generated

## Redis command and data type used:

* Strings
* Lists
  + LPUSH – prepends (add from beginning) one or multiple values to a list
  + LRANGE – gets a range of elements from a list
  + LINDEX – Gets an element from a list by its index
  + LLEN – gets the length of a list
  + LREM – removes elements from a list
  + LSET – sets the value of an element in a list by its index
  + RPUSH – appends (add from end) one or multiple values to a list
* Sets
* Hashes (field(key) : value)
  + HSET – sets the string value of a hash field
  + HGET – gets the value of a hash field stored at the specified key
  + HGETALL – gets all the fields and values stored at the specified key
  + HINCRBY – increments the number stored at the field in the hash stored at key by increment
  + HMGET – returns the values associated with the specified fields in the hash stored at key
* sorted sets
  + ZADD – adds one or more members to a sorted set
  + ZCARD – gets the number of members in a sorted set
  + ZCOUNT – returns the number of elements in the sorted set with a score between min and max
  + ZRANGE – returns a range of members in a sorted set
  + ZREM – removes one or more members from a sorted set

## Cloudinary set up

1. Create Cloudinary account and sign in
2. Create env variable for Cloud Name, API Key and API Secret
3. install cloudinary : npm i cloudinary
4. Create the function in config.ts and add it into loadConfig() function in app.ts
5. /\*\*
6. \* Set up cloudinary
7. \*/
8. public cloudinaryConfig(): void {
9. cloudinary.v2.config({
10. cloud\_name: this.CLOUD\_NAME,
11. api\_key: this.CLOUD\_API\_KEY,
12. api\_secret: this.CLOUD\_API\_SECRET
13. });
14. }

5. Create a cloudinary-upload.ts under global/helpers folder and use the following code:

Note: ? indicate the argument is optional,

{} for an object,

() for the call back

import cloudinary, { UploadApiResponse, UploadApiErrorResponse }from 'cloudinary';

export function uploads(

  file: string,

  public\_id?: string,

  overwrite?: boolean,

  invalidate?: boolean

): Promise<UploadApiResponse | UploadApiErrorResponse | undefined> {

  return new Promise((resolve) =>{

    cloudinary.v2.uploader.upload(

      file,

      {

        public\_id,

        overwrite,

        invalidate

      },

      (error: UploadApiErrorResponse | undefined, result: UploadApiResponse | undefined) => {

        if (error) resolve(error);

        resolve(result);

      }

    );

  });

}

# Authentication Features

## Validation scheme set up

install joi module

npm i joi

Create the signup, signin and password scheme

## Joi validator decorator

To perform the validation.