Setup:

1: project setup [.gitignore]

npm init (entry point: src/server.ts)

scrum template for jira project management

yarn add -D typescript (installing ts as dev-dependency to build in js)

yarn add express mongoose

tsc -init

yarn add dotenv

yarn add ts-node-dev

whenever create a branch in jira, a git branch should also be created

git init

git checkout -b “LSE-3”

git push origin LSE-3

backlog to sprint in jira scrum template.

Create server.ts and app.ts in src folder, then create index.ts in config folder under src

yarn add -D mongoose

yarn add -D express

yarn add -D @types/express

yarn add -D @types/mongoose

yarn add -D @types/cors

yarn add -D dotenv

yarn add ts-node-dev –dev

yarn add cors

setup dotenv variables

DATABASE\_URL=mongodb+srv://little-star-admin:jzwLJne7yeOEsE1m@cluster0.nslo89v.mongodb.net/little-star-edu?retryWrites=true&w=majority

little-star-edu is the database user

yarn start

<https://blog.logrocket.com/linting-typescript-eslint-prettier/>

follow this article for setting up eslint

.eslintrc and .eslintignore create in root folder

"include": ["src"], // which files to compile

"exclude": ["node\_modules"], // which files to skip

Add this to tsconfig.json for eslint setup

* yarn add eslint @typescript-eslint/parser @typescript-eslint/eslint-plugin --save-dev

"lint:check": "eslint --ignore-path .eslintignore --ext .js,.ts .",

"lint:fix": "eslint --fix",

Add this in package.json in scripts

Now install prettier with yarn

* yarn add -D prettier

install prettier and eslint extension for vs code

.prettierrc create in root

"prettier:check":"yarn prettier --write .",

Add this in package in script

Now add the following to settings.json

"editor.defaultFormatter": "esbenp.prettier-vscode",

"editor.formatOnSave": true,

* yarn add -D eslint-config-prettier

Now finished. Run the scripts

Husky installation:

yarn add husky –dev

yarn husky install

A pre-commit file is added to husky folder

Add the commands to this pre-commit file

Package.json scripts are predefined commands, here we can add multiple commands linked by && operator. So to do that we can join these two commands like:

* “lint-prettier”: “yarn lint:check && yarn prettier:check”
* "lint:fix": "eslint . --fix",
* "prettier-fix":"prettier --fix .",

Now add lint staged

* yarn add -D lint-stage

added this script in package.json

"lint-staged": {

"src/\*\*/\*.ts":"yarn lint-prettier"

},

Now in pre-commit add this command: yarn lint-staged

Setup is done…. Congrats

Now use moduler pattern to create user interface, models etc.

Logger for better error management

Winston npm

const winston = require('winston');

const logger = winston.createLogger({

level: 'info',

format: winston.format.json(),

defaultMeta: { service: 'user-service' },

transports: [

//

// - Write all logs with importance level of `error` or less to `error.log`

// - Write all logs with importance level of `info` or less to `combined.log`

//

new winston.transports.File({ filename: 'error.log', level: 'error' }),

new winston.transports.File({ filename: 'combined.log' }),

],

});

//

// If we're not in production then log to the `console` with the format:

// `${info.level}: ${info.message} JSON.stringify({ ...rest }) `

//

if (process.env.NODE\_ENV !== 'production') {

logger.add(new winston.transports.Console({

format: winston.format.simple(),

}));

}

Add this to Winston file in src directory

const logger = winston.createLogger({

transports: [

new winston.transports.Console(),

new winston.transports.File({ filename: 'combined.log' })

]

});

Create two logger: logger and errorLogger

The follow this

<https://github.com/winstonjs/winston#combining-formats>

then add a library: Winston daily rotate file

there are some other types of errors also, that global error handler can not handle

like

unhandledRejection

uncaughtException

SIGTERM

The validation is done by zod

* yarn add zod

http-status for http status code

yarn add http-status