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

GitHub Link:
https://github.com/Apollo-Level2-Web-Dev/Level2-Batch-3-PH-university-server/tree/mastering-mongoose

In short cut : I never recommended to do that , think of this is a hole to lost yourself , rather suffer pain and sufferings and div dive from the documentation ocean.

If we face any issue regarding this clone this from github → https://github.com/Shafayathub/NoSQL-Backend-Template

Just go to a folder and clone the template code
Run →npm i

Npm i will install the node modules that we will having

Run → npm run lint

Lint will use to detect the error , and if any error occurred run

→ npm run lint:fix

Run → npm run prettier

Prettier will format the code

Run → npm run prettier:fix

This command will fix the code

Add a .env file in the root and make two variables

DB_URL= “”
PORT=

Why we have used this unethical motherfucker template code ?
Because the course is flowing and I am finding bugs setting up the eslint and prettier
If eslint is finding any issues then create a file named

.eslint.config.mjs →

```
import globals from "globals";
import pluginJs from "@eslint/js";
import tseslint from "typescript-eslint";

export default [
  { languageOptions: { globals: globals.node } },
  pluginJs.configs.recommended,
  ...tseslint.configs.recommended,
  {
    ignores: ["node_modules", "dist", ".src/config"],
    rules: {
      "no-unused-vars": "error",
      "no-undef": "error",
    },
  },
];
```

Paste the code there and some changes with lint script and prettier.

```
    "lint": "npx eslint .",
    "lint:fix": "npx eslint . --fix",
      "prettier": "prettier --ignore-path .gitignore --write
\"./src**/*.+(js|ts|json)\"",
    "prettier:fix": "npx prettier . --write",
```

Add dev dependencies this package.json file

```
"eslint": "^8.57.0",
"eslint-config-prettier": "^9.1.0",
"eslint-config-standard": "^17.1.0",
"eslint-plugin-import": "^2.29.1",
"eslint-plugin-n": "^16.6.2",
"eslint-plugin-promise": "^6.1.1",
"typescript-eslint": "^7.9.0"
```

Index

- Express
- Mongoose
- Typescript
- Dotenv
- Cors
- Software design pattern (MVC - Modular view controller)
- Interfaces to design object structures
- Creating a schema for a student
- Creating route, services and controllers
- Refactor existing schema

Mongoose: Mongoose is a powerful ODM (Object data modelling) library.

Why we need mongoose ? In mongodb we can send different structural data in one document because mongodb is a no sql database and it will not mind if we give different types of data.

Example : lets say I am going to the cantonment and the army checks me first , so the mongoose is the army which will validate the data.

Many methods of mongoose and mongodb are the same because mongoose is the upper layer of mongodb .

Why we use mongoose?

- Schema creating
- Model creation
- Data validation
- Querying
- Middleware support
- Population

3-2 Installing Express , Mongoose, Typescript, Dotenv ,Cors

https://web.programming-hero.com/l2-b1-b2-mission-success/video/l2-b1-b2-mission-success-8-2-installing-express-mongoose-typescript-dotenv-cors

Make a folder name first project → with cmd open the project

1) npm init -y ⇒ initializing first project

Installing Express , mongoose and typescript

Express: go to the express website → https://expressjs.com/en/starter/installing.html

Run the command → npm install express

Mongoose: Go to mongoose documentation →
https://mongoosejs.com/docs/

Run the command → npm install mongoose –save

Typescript: Go to the typescript documentation and install it

https://www.typescriptlang.org/download/
Run the command → npm install typescript --save-dev

Cors : we need to install cors on the project

Go to the documentations →
https://www.npmjs.com/package/cors

Run the command → npm install cors

Dotenv: we need to save the sensitive data stored in environment variable
Go to the documentation →
https://www.npmjs.com/package/dotenv

Run the command → npm install dotenv

Now we need to initialize a typescript json configuration file ,

Run the command → tsc -init

That will generate a tsconfig.json file. Now go to the tsconfig.json and press cntr+F → rootdir , outdir

“rootDir” : “./src”
“outDir” : “./dist”

Go to the express.js , helloworld example → https://expressjs.com/en/starter/hello-world.html

Now in the root create a folder called → src
inside /src → create a file app.ts

And copy and paste the template to app.ts

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})
```

We need to configure our application so that our all ts files can change to the js file

So we have to go to the package.json → add a command to the script

```
“scripts”:{
build: “tsc”,
}
```

Now run and test the command build using

npm run build

After running this command we can see a app.js file inside dist folder

Now run the code using

node ./dist/app.js

Now terminate the command using cntr+c

But we will not open our server like that , so what will we do ?
In src we will create a file called server.ts and cut and paste the line → app.listen from the app.js and paste it into server.ts

/src/server.ts

```
app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

Server.ts : Server.ts is the file where we make all the connectivity with server to other files.

Go to the mongoose documentation →
https://mongoosejs.com/docs/

And copy and paste the lines from the mongoose template to server.ts before app.listen is written→

```
const mongoose = require('mongoose');

async function main() {
  await mongoose.connect('mongodb://127.0.0.1:27017/test');
```

```
  // use `await
mongoose.connect('mongodb://user:password@127.0.0.1:27017/test');` if your
database has auth enabled

}
```

So now the server.ts will look like this

```
const mongoose = require('mongoose');

async function main() {
  await mongoose.connect('mongodb://127.0.0.1:27017/test');

  // use `await
mongoose.connect('mongodb://user:password@127.0.0.1:27017/test');` if your
database has auth enabled

}
```

app.listen(port, () => {
 console.log(`Example app listening on port ${port}`)
})

Now inside the

```
async function main() {
  await mongoose.connect('mongodb://127.0.0.1:27017/test');

  // use `await
mongoose.connect('mongodb://user:password@127.0.0.1:27017/test');` if your
database has auth enabled

}
```

Function we need to connect the mongodb atlas connection with the application.

GO to the mongodb atlas
https://cloud.mongodb.com/v2/64573d539de82c7252509290#/overview

Click on connect → Drivers → copy the connecting URLs

mongodb+srv://<username>:<password>@cluster0.zycuvps.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0

Now this is a sensitive data so we have to use the environment variable

So create a .env named file in the root and paste the url

DATABASE_URL=mongodb+srv://<username>:<password>@cluster0.zycuvps.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0

PORT = 5000

We have to create a new user to the mongodb atlas database. So go to database access → https://cloud.mongodb.com/v2/64573d539de82c7252509290#/security/database/users

Give user a new userName and password . change the role to atlas admin. Click on add user.

→ Now in .env, DATABASE_URL→ add the username and password ⇒
mongodb+srv://<username>:<password>@cluster0.zycuvps.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0

Add a new Database name inside the link like that

→
mongodb+srv://<username>:<password>@cluster0.zycuvps.mongodb.net/first-project?retryWrites=true&w=majority&appName=Cluster0

In the server.ts use the link to connect to the mongodb atlas like → process.env.DATABASE_URL

Now the server.ts will look like that →

```
const mongoose = require('mongoose');

require('dotenv').config()


async function main() {
  await mongoose.connect(process.env.DATABASE_URL);

  // use `await mongoose.connect('mongodb://user:password@127.0.0.1:27017/test');` if your database has auth enabled

}
```

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})

Error : we got a error `Cannot find name 'require' Do you need to install type definitions for node? Try `npm i --save-dev @types/node`.ts(2580)`

Solution: we have to run and save a command →
npm i --save-dev @types/node

Now the process.env each time writing to be a hazard so we have to do this in some systemic way like that →

Inside src we will create a folder name → app

Inside app we will create another folder name config → inside config folder we will create a file name index.ts

/src/app/config/index.ts

Index we are using to config the .env . So inside index.ts

import dotenv from “dotenv”

import path form ‘path’

dotenv.config({path : path.join((process.cwd(), ‘.env’))})

export default{

port: process.env.PORT,

database_url: process.env.DATABASE_URL

}

3-3 Installing Eslint, Refactor Code, Fix Errors Using Command

https://web.programming-hero.com/l2-b1-b2-mission-success/video/l2-b1-b2-mission-success-8-3-installing-eslint-refactor-code-fix-errors-using-command

Now since we have configured the config file so inside server.ts we have to replace all process.env to config. And import config from the required path

Import config from “./app/config”

Now the server.ts will look like that →

```
const mongoose = require('mongoose');

import config from “./app/config”

async function main() {
```

```
  await mongoose.connect(config.database_url);

  // use `await
mongoose.connect('mongodb://user:password@127.0.0.1:27017/test');` if your
database has auth enabled

}
```

```
app.listen(config.port, () => {
  console.log(`Example app listening on port ${config.port}`)
})
```

Now we can replace require from app.ts to import module, so the app.ts will look like this →

```
import express from 'express'
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})
```

Error: After changing the require to import an error will occurred says → can't find a declaration file for module `'express'`.

Solution: typescript don't do type in that way so we have to run a command

→ npm i –save-dev @types/express

Now we have to give the types of request and response for typescript that comes directly from the express

Now the app.ts looks like that →

```
import express, { Request, Response } from 'express'
```

```
const app = express()
const port = 3000

app.get('/', (req : Request , res : Response) => {
  res.send('Hello World!')
})
```

In server.ts we can change the require syntax to import

There could be an error that in server.ts → config.database_url types not defined, since we have imported it from .env file and we are sure about its types we can give string surely.

And we cut paste the app.listen function into the main() function.

Now the server.ts will look like that →

We have to change the DATABASE_URL to database_url since we are taking it from config file

```
const mongoose = require('mongoose');

async function main() {
  await mongoose.connect(config.database_url as string);

  // use `await mongoose.connect('mongodb://user:password@127.0.0.1:27017/test');` if your database has auth enabled

app.listen(config.port, () => {
  console.log(`Example app listening on port ${config.port}`)
})

}
```

Now the app doesn’t exists in server.ts file its existes in app.ts so we have to export app from app.ts and import app in server.ts

Now app.ts looks like that: →

```
import express, { Request, Response } from 'express'
const app = express()
const port = 3000

app.get('/', (req : Request , res : Response) => {
  res.send('Hello World!')
})


export default app;
```

Server.ts looks like that →

```
import app from “./app”

const mongoose = require('mongoose');

import config from ‘./app/config’

async function main() {
  await mongoose.connect(config.database_url as string);

  // use `await mongoose.connect('mongodb://user:password@127.0.0.1:27017/test');` if your database has auth enabled

app.listen(config.port, () => {
  console.log(`Example app listening on port ${config.port}`)
})

}
```

Try Catch: still there is a problem in server.ts

There could be an error and if that happens our server will be crushed so we have to use a try catch block inside the main() function

Server.ts looks like that →

import app from “./app”

```
const mongoose = require('mongoose');

async function main() {
try{
 await mongoose.connect(config.database_url as string);

  // use `await mongoose.connect('mongodb://user:password@127.0.0.1:27017/test');` if your database has auth enabled
```

app.listen(config.port, () => {
 console.log(`Example app listening on port ${config.port}`)
})

```
}
catch(err)
{
      console.log(err);
}
}
```

Parsers:

We have to use perser to perse the json and cors

Inside app.ts→

app.use(express.json())

app.use(cors())

If can’t find cors shown import cors from ‘cors’

Now when you want to import cors the typescript will give an error that cors need a typescript declaration file

So run the command → npm i –save-dev @types/cors

Now in this part either you can fix it or follow the process

create a file named

.eslint.config.mjs →

```
import globals from "globals";
import pluginJs from "@eslint/js";
import tseslint from "typescript-eslint";

export default [
  { languageOptions: { globals: globals.node } },
  pluginJs.configs.recommended,
  ...tseslint.configs.recommended,
  {
    ignores: ["node_modules", "dist", ".src/config"],
    rules: {
      "no-unused-vars": "error",
      "no-undef": "error",
    },
  },
];
```

Paste the code there and some changes with lint script and prettier.

```
    "lint": "npx eslint .",
    "lint:fix": "npx eslint . --fix",
      "prettier": "prettier --ignore-path .gitignore --write
\"./src**/*.+(js|ts|json)\"",
    "prettier:fix": "npx prettier . --write",
```

Add dev dependencies this package.json file

```
    "eslint": "^8.57.0",
    "eslint-config-prettier": "^9.1.0",
    "eslint-config-standard": "^17.1.0",
    "eslint-plugin-import": "^2.29.1",
    "eslint-plugin-n": "^16.6.2",
    "eslint-plugin-promise": "^6.1.1",
    "typescript-eslint": "^7.9.0"
```

First copy and paste the dev dependencies in package.json and run the command → npm i

create a file named .eslint.config.mjs →and copy and paste the upper code

So to setup the projects we need 2 packages eslint and prettier

Eslint : Es lint doest the works

- Code format
- Find errors
- Code quality check

Prettier : Helps to format the code

To setup the typescript eslint prettier setup we need to go to google and search → typescript eslint prettier setup and we can find a blog

The documentation → https://blog.logrocket.com/linting-typescript-eslint-prettier/

Now we have to change some code in tsconfig.json

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

"exclude": ["node_modules"], // which files to skip

“compilerOptions” :{

// code

}
```

We have to add this two lines before the compiler options

ESLint :

From the blog we have to install eslint to the project , run the command →

```
npm install eslint @typescript-eslint/parser @typescript-eslint/eslint-plugin --save-dev
```

Now we need a eslint config file so we have to run the command →

```
npx eslint --init
```

```
npm init @eslint/config
```

Any of them will work

After running this command the project will ask some questions

- Syntex check problem find = yes
- Module = import / export
- Which framework does your project use? - none of this
- Does your project use TypeScript? = yes
- Where does your code run? = Node
- Would you like to install them now? = Yes
- Which package manager do you want to use? = npm

After installation it will give us a file named eslint.rc

From the new version of .eslintrc.json we

In this eslint.rc file add rules

```
"rules": {
    "@typescript-eslint/no-unused-vars": "error",
    // to enforce using type for object type definitions, can be type or interface
    "@typescript-eslint/consistent-type-definitions": ["error", "type"],
  },
```

If this process does not work eslint has got some updated version. Keep going on

No unused vars refers to the non used vars doesn't give the error

Now some files doesn't need to be lintin so make a file name `.eslintignore` in the root directory and inside

.eslintignore → add two lines

```
node_modules

dist
```

We have to use a script to lintin files add a script to the package.json

```
"scripts": {

    "lint": "eslint src --ignore-path .eslintignore --ext .ts"

   },
```

We have add src and removed .js from lint script that refers that inside src lint every .ts file

When we run the script npm run lint the lint is finding the bugs.

Run it

I am geeting a error :

Invalid option '--ignore-path' - perhaps you meant '--ignore-pattern'?You're using eslint.config.js, some command line flags are no longer available. Please see https://eslint.org/docs/latest/use/command-line-interface for details.

x0d\x0a$

Lets paste it in google and see whats wrong

https://eslint.org/docs/latest/use/configure/ignore

Following this blog I have add this ignores config in eslint.config.mjs

```
export default [
    {
        ignores: [".config/*"]
    }
];
```

And also changed the lint script to

```
"scripts": {
    "lint": "eslint src --ignore-pattern .eslintignore --ext .ts"
  },
```

Now I am getting another error →

Invalid option '--ext' - perhaps you meant '-c'?

You're using eslint.config.js, some command line flags are no longer available. Please see https://eslint.org/docs/latest/use/command-line-interface for details.

x0d\x0a$

Lets copy and paste it into google

Error : main() is declared but never used in app.ts if main is not called then the server will not run so we have to call main()

The server.ts looks like that →

import app from “./app”

```
const mongoose = require('mongoose');

async function main() {
try{
 await mongoose.connect(config.database_url as string);

  // use `await mongoose.connect('mongodb://user:password@127.0.0.1:27017/test');` if your database has auth enabled
```

app.listen(config.port, () => {
 console.log(`Example app listening on port ${config.port}`)
})

```
}
catch(err)
{
      console.log(err);
}
}
```

main();

Solution: If we run the command

→ npx eslint src -fix

Eslint will automatically fixes all the problems so let it add to a script

Add that line to the script

“lint:fix” : “npx eslint src –fix”

Now if we run →

npm run lint:fix ⇒ this command will fix the bugs that eslint finds out.

Some Extra rules in eslint.rc →

```
"rules": {
    "@typescript-eslint/no-unused-vars": "error",
    // to enforce using type for object type definitions, can be type or interface
    "@typescript-eslint/consistent-type-definitions": ["error", "type"],
       “no-unused-expression” : “error”,
       “prefer-const”: “error”,
       “no-console”: “warn”,
       “no-undef”: “error”
  },
“globals”:{
“process” : “readonly”
}
```

No unused-expression : we don’t want any unused expression,

Prefer-const : we can’t use let

No-undef : we can’t except any undefined variable

```
“globals”:{
“process” : “readonly”
}
```

⇒ refers that we have a global variable called process which only we reads.

Now check if lintIn run perfectly or not

→ npm run lint

If any problem occurred run npm run lint:fix

Error: ‘process’ is not defined no-undef , because process.env is not a local variable its a global variable,

→Solution: We have to add the eslint.rc some global variable, go to eslint.rc, after rules add a property called globals

rules: {

},

“Globals”:{

“process”: “readonly”

}

Install prettier

We go to the documentation again → https://blog.logrocket.com/linting-typescript-eslint-prettier/

And run the command →

npm install --save-dev prettier

While installing prettier we are encountering a error:

After that we have to create a file in the root name prettierrc.json and add the rules →

```
// .prettierrc.json

{

  "semi": true, // Specify if you want to print semicolons at the end of statements

  "singleQuote": true, // If you want to use single quotes

  "arrowParens": "avoid", // Include parenthesis around a sole arrow function parameter

}
```

We can use a command to format our code to pretty app.ts

```
npx prettier --write src/app.ts
```

If we go down in the documentation there is a script given prettier

```
"prettier": "prettier --ignore-path .gitignore --write \"./src/**/*.+(js|ts|json)\""
```

Now we need to set some setting in vscode

Go to vscode setting.json →

Add this two lines in settings.json

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

  "editor.formatOnSave": true,
```

Now there are two ways to use eslint and prettier , one command and another one is extension

With command we can run the commands before pussing out

Now in the documents there might be sometimes prettier and eslint will conflict so we need to install another package

Run this command →

```
npm install --save-dev eslint-config-prettier
```

And we have to add another scripts in eslint.rc so we can replace extends from eslint.rc in this line

```
"extends": ["eslint:recommended", "plugin:@typescript-eslint/recommended",
"prettier"],
```

Now we can add a script in package.json to fix all files in prettier

Script: {

“prettier:fix” : “npx prettier --write src”,

}

Now we can run the command npm run prettier:fix

.gitignore: .gitignore will ignore the files that we don’t want to push in the github

node_modules

dist

Now we can run the project using this command two command

npm run build

node run dist/server.js

While i am trying to start the server with this command I am finding a error that path is not found I am keep going on the process

Error: In mezba bhai project there is a mongodb server is not white listed.

Solution :

Go to mongodb atlas → network access → add ip address → allow access from anywhere

Now everytime node modules run and server run is quite bit pathetic, So there is a package name tsnodedev we can use this , But we can not use it everytime we have to give production node javascript file

TS-Node-Dev:

https://www.npmjs.com/package/ts-node-dev

Now copy and run the command :

```
npm i ts-node-dev
```

Now go to its github →

https://github.com/wclr/ts-node-dev#readme

For using ts-node-dev we have to run another command →

npm run ts-node-dev --respawn --transpile-only src/server.ts

Add Script:

Each time to run this command is hatric so we can add a script in package,json →

“start:dev” : “ts-node-dev --respawn --transpile-only src/server.ts ”

“start:pod” : “node ./dist/server.js”

We can add a variable to the .env that can define either we are using production or development

So inside .env →

NODE_ENV = development

__ Basic project setup done

Step by step

1) Npm init -y ⇒ initialize package.json