**Text Instructions: Module 8**

GitHub Link: <https://github.com/Apollo-Level2-Web-Dev/Level2-Batch4-PH-University-Server/tree/mastering-mongoose>

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

**8-1 Introduction to mongoose**

Create folder : first-project > cmd > code . > npm init -y

Google > express js > getting started > npm install express >

typescript (download install typescript)> npm install typescript --save-dev

mongoose > getting started > npm install mongoose --save

cors npm > npm i cors

dotenv > npm i dotenv

tsc –init

NB :

**Adjust Execution Policy in PowerShell**

1. **Open PowerShell as Administrator:**
   * Search for **PowerShell** in the Start menu.
   * Right-click on it and select **Run as Administrator**.
2. **Change the Execution Policy:** Run the following command to allow scripts to run:

powershell

command:

Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned

* + **Explanation:**
    - CurrentUser: Changes the policy for the current user only.
    - RemoteSigned: Allows locally created scripts to run, but requires downloaded scripts to be signed.

1. **Confirm the Change:** If prompted, press **Y** and then **Enter**.

.env :

PORT=5000

DATABASE\_URL=mongodb+srv://amin-um:admin12345@cluster0.uuibjb3.mongodb.net/first-project?retryWrites=true&w=majority&appName=Cluster0

Tsconfig > modify “rootdir”, ”outdir”

Google > express js > getting started > hello world > Copy the code (first)

package.json > Debug > “build”: “tsc”,

terminal > npm run build [create ”dist” folder]

node ./dist/app.js

npm i --save-dev @types/node

ইন্ডাস্ট্রি তে এখনো .eslintrc.json এবং .eslintignore এই প্যাটার্ন ব্যবহার হতেই দেখা যায়, তবে .eslint.config.mjs তুলনামূলক নতুন যা সামনে ইন্ডাস্ট্রিতে এডাপ্ট হতে পারে   
  
ব্লগঃ  
  
<https://dev.to/shafayat/-express-typescript-eslint-prettiersetup-5fhg>

ভিডিওঃ <https://www.facebook.com/100004778526351/vide>

**8-3 Installing eslint, refactor code, fix errors using command**

npm i --save-dev @types/express

npm i --save-dev @types/cors

google > typescript eslint prettier setup > tsconfig.json > copy two lines :

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

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

goto vscode > tsconfig.json > paste them

now again go to blog-doc > eslint > copy the code :

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

terminal :

npx eslint --init

docs [ for setup ]:<https://dev.to/shafayat/-express-typescript-eslint-prettiersetup-5fhg>

**8-4 Install prettier,ts-node-dev,fix formatting issues**

Install : npm install --save-dev eslint-config-prettier

npm run prettier fix

mongodb atlas > network access > set > confirm

terminal : node ./dist/server.js

**8-5 Software design pattern , mvc vs modular, create an interface**

Src > app > config > modules > student > student.interface.ts

import { Schema, model, connect } from 'mongoose';

export type Guardian = {

    fatherName: string;

    fatherOccupation: string;

    fatherContactNo: string;

    motherName: string;

    motherOccupation: string;

    motherContactNo: string;

  };

  export type UserName = {

    firstName: string;

    middleName: string;

    lastName: string;

  }

  export type LocalGuardian = {

    name: string;

    occupation: string;

    contactNo: string;

    address: string;

  }

export type Student = {

  id: string;

  name: UserName,

  gender: "male"|"female";

  dateOfBirth: string;

  email: string;

  avatar?: string;

  contactNo: string;

  emergencyContactNo: string;

  bloodGroup: "A+" | "A-" | "B+" | "B-" | "AB+" | "AB-" | "O+" | "O-";

  presentAddress: string;

  permanentAddress: string;

  guardian: Guardian;

  localGuardian: LocalGuardian;

}

**8-6 Create an schema for a student**

**8-7 Refactor your schema**

student.model.ts:

import { Schema, model, connect } from 'mongoose';

import { Guardian, LocalGuardian, Student, UserName } from './student/student.interface';

const userNameSchema = new Schema<UserName>({

        firstName:{

            String,

            required: true,

        },

        middleName: {

            type: String

        },

        lastName: {

            type: String,

            required: true,

        }

})

const guardianSchema = new Schema<Guardian>({

    fatherName: {

        type: String,

        required: true,

    },

    fatherOccupation: {

        type: String,

        required: true,

    },

    fatherContactNo: {

        type: String,

        required: true,

    },

    motherName: {

        type: String,

        required: true,

    },

    motherOccupation: {

        type: String,

        required: true,

    },

    motherContactNo: {

        type: String,

        required: true,

    },

})

const localGuardianSchema = new Schema<LocalGuardian>({

    name: {

        type: String,

        required: true,

    },

    occupation: {

        type: String,

        required: true,

    },

    contactNo: {

        type: String,

        required: true,

    },

    address: {

        type: String,

        required: true,

    },

})

const studentSchema = new Schema<Student>({

    id: {type: String },

    name: userNameSchema,

    gender: ['male', 'female'],

    dateOfBirth: {type: String},

    email: {type: String, required: true},

    contactNo: {type: String, required: true},

    emergencyContactNo: {type: String, required: true},

    bloodGroup : ["A+", "A-", "B+", "B-", "AB+", "AB-", "O+", "O-"],

    presentAddress: {type: String, required: true},

    permanentAddress: {type: String, required: true},

    guardian:guardianSchema,

    localGuardian: localGuardianSchema,

    profileImg: {type: String},

    isActive: {'active': 'blocked'},

})

const Student = model<Student>('Student', studentSchema);

**8-8 Create route , service and controller**

Previous app.ts:

import express, { Application, Request, Response } from 'express';

import cors from 'cors';

const app: Application = express();

// parser

app.use(express.json());

app.use(cors());

app.get('/', (req: Request, res: Response) => {

  //   res.send('Hello World!')

  const a = 10;

  res.send(a);

});

// console.log(process.cwd()); // E:\web\Programming Hero\Level 2\Mission 01-Be A Typescript Technocrat\Module 8-Mastering The Core concept of Mongoose\first-project

export default app;

**8-9 Insert a student data into mongoDB**