בס"ד

Server.ts file:

import mongoose from "mongoose";

import express from "express";

import dotenv from "dotenv";

import userRouter from "./routes/user.route";

import cors from "cors";

dotenv.config(); //1

const app = express();

app.use(express.json()); //takes the req.body to json

app.use(cors({ origin: "\*" })); //2

app.use("/user", userRouter); //3

try {

  mongoose.connect(process.env.URI!).then(() => { //4

    app.listen(process.env.PORT, () => {

      console.log(

        `connected to mongo db and running on port ${process.env.PORT}`

      );

    });

  });

} catch (error) {

  console.log("hey something went wrong");

}

1. ע"י הוראה זו נעשה יבוא של env מ dotenv ומאפשר למשתנים המוסתרים ב env לבוא לידי שימוש בקובץ הזה.

2.שורה זו מאפשרת למשתמש מכל כתובת לקרוא ולהשתמש ב server. הכוונה היא ש cors הינו המנגנון המגן על השרת מקריאות חיצוניות \ לא מאשרות, ואנו מרשים לקבל קריאות מכל דומיין ע"י הסימון \*

3. אחרי שהבאנו את userRouter אנחנו משתמשים בו ומפנים את app להשתמש בו – ב routers שבו.

4. בחלק האחרון של הקובץ אנחנו מייבאים את ה url ואת ה port מהקובץ env בשביל להתחבר ל mongo db.

בס"ד

Controllers.ts file:

import { Request, Response } from "express";

import { UserModel } from "../models/user.schema";

import { isValidObjectId } from "mongoose";

import { checkPassword, generateToken, hashPassword } from "../utils/helpers";

export async function signup(req: Request, res: Response) {

  try {

    const { firstName, lastName, email, password } = req.body;

    //checking is all values are given

    if (!firstName || !lastName || !email || !password)

      return res.status(400).json({ message: "all fields are required" });

    //check if the user exists already

    const user = await UserModel.findOne({

      email,

    });

    if (user)

      return res

        .status(400)

        .json({ message: "the user already exist in the system" });

    //send the password to be hashed in order to secure the data in the db

    const hashedPassword = await hashPassword(password); //1

    const newUser = await UserModel.create({

      email,

      firstName,

      isVerified: false,

      lastName,

      password: hashedPassword, //2

      role: 1,

      techers: [],

    });

    return res.status(201).json({ message: "User created successfuly" });

  } catch (err) {

    return res.status(500).json({ message: err });

  }

}

1. זו היא הפונקציה השולחת את ה string ומצפינה את ערכו, ושומרת אותו ב dateBase
2. כאן אנחנו מכניסים את הערך המוצפן לתוך הערך password

בס"ד

export async function login(req: Request, res: Response) {

  try {

    const { email, password } = req.body;

    if (!email || !password)

      return res.status(400).json({ message: "all fields are required" });

    const user = await UserModel.findOne({ email });

    if (!user)

      return res.status(400).json({ message: "user dose not exist 😔" });

    //check if the passwords are the same

    const isMatch = await checkPassword(password, user.password);

    if (!isMatch)

      return res.status(400).json({ message: "incorrect password" });

    const token = generateToken(user.\_id as any);

    return res.status(200).json({ token, user });

  } catch (err) {

    return res.status(500).json({ message: err });

  }

}

export async function getUserList(req: Request, res: Response) {

  // getting the user id

  const { USER\_ID } = req.body;

  //verifying that that the user id and admin

  const user = await UserModel.findById(USER\_ID);

  if (!user) return res.status(400).json({ message: "user not found" });

  if (user.role != 3) return res.status(403).json({ message: "admin only" });

  // returning all of the users beside the admin itself

  const allUsers = await UserModel.find({});

  //filter the admin out of the array

  const users = allUsers.filter((user) => user.role != 3);

  return res.status(200).json(users);

}

חכחכחכ

בס"ד.

export async function getOneUser(req: Request, res: Response) {

  //get the user id from the params

  const { id } = req.params;

  const { USER\_ID } = req.body;

  //validate the the id is good

  if (!isValidObjectId(id))

    return res.status(400).json({ message: "the id is not valid" });

  //make sure that the USER\_ID belongs to the admin

  const user = await findOneUser(USER\_ID);

  if (!user) return res.status(400).json({ message: "user not found" });

  if (user.role != 3)

    return res.status(429).json({

      message: "Only admin have access to this data",

    });

  const SearchUser = await findOneUser(id);

  if (!SearchUser) return res.status(400).json({ message: "user not found" });

  return res.status(200).json(SearchUser);

}

export async function findOneUser(id: string) {

  if (!isValidObjectId(id)) return null;

  const user = await UserModel.findById(id);

  if (!user) return null;

  return user;

}