Dogrulama islemleri icin

express-validator

Sifreleri hashlemek icin

bcryptjs

Token olusturup kullanmak signin ve sign out islemlerini yapmak icin

jsonwebtoken

**routes / users.js**

router.post(

  "/",

express-validator ORNEK

  [

    check("name", "Please add name")

      .not()

      .isEmpty(),

    check("email", "Please include a valid email").isEmail(),

    check(

      "password",

      "Please enter a password with 6 or more characters"

    ).isLength({ min: 6 })

  ],

  async (req, res) => {

    const errors = validationResult(req);

    if (!errors.isEmpty()) {

      return res.status(400).json({ errors: errors.array() });

    }

    const { name, email, password } = req.body;

    try {

      let user = await User.findOne({ email });

      if (user) {

        return res.status(400).json({ msg: "User already exists" });

      }

      user = new User({

        name,

        email,

        password

      });

Bcryptjs ORNEK – genSalt ve HASHLEME

      //Hashing Password

      //Once salti aliriz sonra hashleriz

      const salt = await bcrypt.genSalt(10); //.genSalt() icine default olarak numara alir oda 10 dur

      //user.passwordu esitleriz. await yaparak bcrypt.hash ile hashleriz cunku bize deger donmesini istiyoruz.

      //hash icine 2 deger alir. 1.si gelen password ikincisi salt degeridir.

      user.password = await bcrypt.hash(password, salt);

      await user.save();

jsonwebtoken JWT ORNEK – SIGN IN

      //Kayit sonrasi girisi asagidaki sekilde saglariz

      //Once jwt yani jsonwebtokeni import ederiz buraya

      //Sonra payloadi olustur asagidaki gibi

      //jwt semasina bak, jwt 3 sey ister. payload, secretkey, ve expire yani acik kalma suresi

      //userid yi jwt de payloada gonderecegiz

      //secretkeyi configde olustur ve configi buraya import edip icinden al

      const payload = {

        user: {

          id: user.id

        }

      };

      jwt.sign(

        payload,

        config.get("jwtSecret"),

        {

          expiresIn: 360000 //Yani 3600 dersek 3600 saniye oda 60dakika yani 1 saat hesabin aktif kalacagi sure

        },

        //Ve en son olarakda istege bagli olarakda token koyabiliriz

        (err, token) => {

          if (err) throw err;

          res.json({ token });

        }

      );

    } catch (err) {

      console.error(err.message);

      res.status(500).send("Server Error");

    }

  }

);

HASHLENMIS PASSWORD ile INPUTTAN YANI REQ BODY DEN GELEN PASWORDU KARSILASTIRMA YAPMAK VE SIGN IN YAPMAK

**routes / auth.js**

const express = require("express");

const router = express.Router();

const bcrypt = require("bcryptjs");

const jwt = require("jsonwebtoken");

const config = require("config");

const { check, validationResult } = require("express-validator");

const User = require("../models/User");

router.get(

  "/",

express-validator ORNEK

  [

    check("email", "Please include a valid email").isEmail(),

    check("password", "Password is required").exists()

  ],

  async (req, res) => {

    const errors = validationResult(req);

    if (!errors.isEmpty()) {

      return res.status(400).json({ errors: errors.array() });

    }

    const { email, password } = req.body;

    try {

      let user = await User.findOne({ email });

      if (!user) {

        return res.status(400).json({ msg: "invalid Credentials" });

      }

Bcryptjs ORNEK – KARSILASTIRMA

      //Hashlenen passwordu karsilastirma Compare etmek

      const isMatch = await bcrypt.compare(password, user.password);

      if (!isMatch) {

        return res.status(400).json({ msg: "Invalid Credentials" });

      }

jsonwebtoken JWT ORNEK – SIGN IN

      //bcrypt ile karsilastirdiktan sonra ve jwt.sign ile girisini saglariz

      const payload = {

        user: {

          id: user.id

        }

      };

      jwt.sign(

        payload,

        config.get("jwtSecret"),

        {

          expiresIn: 360000 //Yani 3600 dersek 3600 saniye oda 60dakika yani 1 saat hesabin aktif kalacagi sure

        },

        //Ve en son olarakda istege bagli olarakda token koyabiliriz

        (err, token) => {

          if (err) throw err;

          res.json({ token });

        }

      );

    } catch (err) {

      console.error(err.message);

      res.status(500).send("Server Error");

    }

  }

);

// @route           POST api/auth

// @description     Auth user & get token

// @access          Public

router.post("/", (req, res) => {

  res.send("Log in user");

});

module.exports = router;

Ve artik korumak istedigimiz routes lari userlarin tokenleriyle dogrulayip karsilastirip bir AUTH middleware i olusturmamiz gerekiyor

**Middleware / Auth.js**

const jwt = require("jsonwebtoken");

const config = require("config");

module.exports = function(req, res, next) {

  //Get Token from Header

  const token = req.header("x-auth-token");

  //Check If not token

  //401 = Unauthorized demektir

  if (!token) {

    return res.status(401).json({ msg: "No token authorization denied" });

  }

  //Verify yani Dogrulama Yapmak

  //jwt.verify 2 argument alir icine. 1. si TOKEN 2. si secretKey

  try {

    const decoded = jwt.verify(token, config.get("jwtSecret"));

    //Dogruladiktan sonra req ile aldigimiz useri dogrulanmis kullanici olarak atiyoruz decoded.user diyerek

    req.user = decoded.user;

    next();

  } catch (err) {

    res.status(401).json({ msg: "Token is not valid" });

  }

};

Ve artik auth u ikinci parametre olarak routes larimiza ekleyip kullanabiliriz asagidaki gibi. Boylelikle Req isteklerinden once korumak istedigimzi routeslari user log in kontrolunu gerceklestirip req isteklerine cevap verecegiz

**routes / auth.js**

// @route           POST api/auth

// @description     Auth user & get token

// @access          Public

router.post("/", auth, (req, res) => {

  res.send("Log in user");

});

Veya

router.get("/", auth, async (req, res) => {

  try {

    const user = await User.findById(req.user.id).select("-pasword"); //Databaseden kullanicinin verilerini cekiyoruz password haric diyoruz

    res.json(user);

  } catch (err) {

    console.error(err.message);

    res.status(500).send("Server error");

  }

});