|  |
| --- |
| Server.js  const express = require('express');  const mongoose = require('mongoose');  const cors = require('cors');  const app = express();  app.use(cors());  app.use(express.json());  const port = 8000;  const dotenv = require("dotenv");  dotenv.config();  const dishStaticRouter = require("./routes/dishes.router.js");  const dishDBRouter = require("./routes/dishesDB.router.js");  const authRouter = require("./routes/auth.router.js");  const userRouter = require("./routes/user.router.js");  var mongoURL = "mongodb+srv://gauravraj:Mongo%40raj11@cluster0.cddlpyx.mongodb.net/FoodApp?retryWrites=true&w=majority" ;  mongoose.connect(mongoURL, {useNewUrlParser: true,useUnifiedTopology:true})  const con = mongoose.connection  con.on('connected', () => {  console.log('MongoDB connected...')  })  app.get('/', (req, res) => res.send('Server is working !' + port));  app.use('/api/dishes' , dishStaticRouter); //localhost:8000/api/dishes  app.use('/getdishes', dishDBRouter); //localhost:8000/getdishes  app.use('/api/auth', authRouter); //localhost:8000/api/auth  app.use('/api/users', userRouter); //localhost:8000/api/users  app.listen(port, () => console.log(`Food app is listening on port ${port}!`)) |
| auth.router.js  const express = require('express');  const signupHandler = require("../controllers/signupController");  const loginHandler = require("../controllers/loginController");  const router = express.Router();  router.route("/register") //localhost:8000/api/auth/register  .post(signupHandler)  router.route("/login") //localhost:8000/api/auth/login  .post(loginHandler)  module.exports = router; |
|  |
| userModel.js  const mongoose = require("mongoose");  const userSchema = mongoose.Schema({  username: { type: String, required: true },  number: { type: Number, required: true, unique: true },  email: { type: String, required: true, unique: true },  password: { type: String, required: true },    }, { timestamps: true });  const User = mongoose.model("users", userSchema);  module.exports = User; |
| signupController.js  const CryptoJS = require('crypto-js'); //bcrypt  const User = require("../models/userModel");  const signupHandler = async (req, res) => {  try{  const newUser = new User({  username: req.body.username,  number: req.body.number,  email: req.body.email,  // password: req.body.password  password: CryptoJS.AES.encrypt(req.body.password, process.env.PASSWORD\_SECRET\_KEY).toString()  // The CryptoJS.AES.encrypt() function is used to encrypt  // the password using the Advanced Encryption Standard (AES) algorithm.  });  const savedUser = await newUser.save();  console.log(savedUser);  // Return success response with a message and an identifier  res.status(201).json({ message: "User created successfully", redirectToLogin: true });  }catch(err){  console.log(err);  res.status(500).json({ message: "Error creating a user" })  }  }  module.exports = signupHandler; |
| loginController.js  const CryptoJS = require('crypto-js');  const jwt = require('jsonwebtoken');  const User = require("../models/userModel");  const loginHandler = async (req, res) => {  try {  // Find the user by their mobile number  const user = await User.findOne({ number: req.body.number });    // If user doesn't exist, return an error  if (!user) {  return res.status(401).json({ message: "Incorrect Mobile Number" });  }  // Decrypt the stored password and compare with the provided password  const decodedPassword = CryptoJS.AES.decrypt(user.password, process.env.PASSWORD\_SECRET\_KEY).toString(CryptoJS.enc.Utf8);    if (decodedPassword !== req.body.password) {  // If passwords don't match, return an error  return res.status(401).json({ message: "Incorrect Password" });  }  // If credentials are correct, generate JWT token  const accessToken = jwt.sign({ userId: user.\_id }, process.env.JWT\_SECRET\_KEY, { expiresIn: '1h' });    // Return the token as part of the response  // res.json({ accessToken });  res.json({ accessToken, userId: user.\_id });  } catch (err) {  // Handle errors  console.error(err);  res.status(500).json({ message: "Internal Server Error" });  }  }  module.exports = loginHandler; |