**Index.js**

**import express from "express";**

**import AppRouter from "./src/routers/router.js";**

**import dotenv from "dotenv";**

**import cors from "cors";**

**import mongoose from "mongoose";**

**dotenv.config();**

**const app = express();**

**const PORT = Number(process.env.PORT);**

**app.use(cors());**

**app.use(express.json());**

**app.use("/", AppRouter);**

**app.get("/", (req, res) => {**

**res.send("Welcome to the Server! 🌐");**

**});**

**mongoose**

**.connect(`${process.env.DB\_URL}/${process.env.DB\_NAME}`)**

**.then(() => {**

**app.listen(PORT, () => {**

**console.log(`Server is running on port ${PORT}`);**

**});**

**})**

**.catch((error) => {**

**console.log(error);**

**});**

**In the src folder under common subfolder**

**Auth.js**

**import bcrypt from "bcryptjs";**

**import jwt from "jsonwebtoken";**

**const hashPassword = async (password) => {**

**let salt = await bcrypt.genSalt(Number(process.env.SALT\_ROUNDS));**

**let hash = await bcrypt.hash(password, salt);**

**return hash;**

**};**

**const hashCompare = async (password, hash) => {**

**return await bcrypt.compare(password, hash);**

**};**

**const createToken = async (payload) => {**

**const token = await jwt.sign(payload, process.env.JWT\_SECRET, {**

**expiresIn: process.env.JWT\_EXPIRE,**

**});**

**return token;**

**};**

**const decodeToken = async (token) => {**

**const payload = await jwt.decode(token);**

**return payload;**

**};**

**export default {**

**hashPassword,**

**hashCompare,**

**createToken,**

**decodeToken,**

**};**

**Validators.js**

**import { validationResult, body } from "express-validator";**

**const validate = (route) => {**

**switch (route) {**

**case "signup":**

**return [**

**body("firstName").notEmpty().withMessage("First Name is required"),**

**body("lastName").notEmpty().withMessage("Last Name is required"),**

**body("email").isEmail().withMessage("Invalid email address"),**

**body("password")**

**.matches(/^(?=.\*[a-zA-Z])(?=.\*\d).{8,}$/)**

**.withMessage(**

**"Password must contain at least one digit, one lowercase and one uppercase letter, and one special character (@#$%^&+=)"**

**),**

**];**

**case "signin":**

**return [**

**body("email").isEmail().withMessage("Invalid email address"),**

**body("password").notEmpty().withMessage("Password is required"),**

**];**

**case "forgotPassWord":**

**return [body("email").isEmail().withMessage("Invalid email address")];**

**case "resetPassWord":**

**return [**

**body("OTP").notEmpty().withMessage("OTP is required"),**

**body("password")**

**.isLength({ min: 8 })**

**.matches(/^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[@#$%^&+=]).{12,}$/)**

**.withMessage(**

**"Password must be at least 12 characters long and contain at least one digit, one lowercase and one uppercase letter, and one special character (@#$%^&+=)"**

**),**

**];**

**default:**

**return [];**

**}**

**};**

**const validationMiddleware = (req, res, next) => {**

**const errors = validationResult(req);**

**if (!errors.isEmpty()) {**

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

**}**

**next();**

**};**

**export default { validate, validationMiddleware };**

**Under controllers subfolder**

**User.controller.js**

**import userModel from "../models/User.models.js";**

**import auth from "../common/auth.js";**

**import nodemailer from "nodemailer";**

**import dotenv from "dotenv";**

**dotenv.config();**

**const signupController = async (req, res) => {**

**try {**

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

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

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

**}**

**const existingUser = await userModel.findOne({ email });**

**if (existingUser) {**

**return res.status(400).json({ message: "Email is already registered" });**

**}**

**const hashedPassword = await auth.hashPassword(password);**

**const newUser = new userModel({**

**firstName,**

**lastName,**

**email,**

**password: hashedPassword,**

**});**

**await newUser.save();**

**res.status(201).json({ message: "User registered successfully" });**

**} catch (error) {**

**console.error(error);**

**res.status(500).json({ message: "Internal server error" });**

**}**

**};**

**const signinController = async (req, res) => {**

**try {**

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

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

**if (!user) {**

**return res.status(404).json({ message: "User not found" });**

**}**

**const isPasswordValid = await auth.hashCompare(password, user.password);**

**if (!isPasswordValid) {**

**return res.status(401).json({ message: "Invalid password" });**

**}**

**const token = await auth.createToken({**

**id: user.\_id,**

**firstName: user.firstName,**

**lastName: user.lastName,**

**email: user.email,**

**role: user.role,**

**});**

**let userData = await userModel.findOne(**

**{ email: req.body.email },**

**{ \_id: 0, password: 0, status: 0, createdAt: 0, email: 0 }**

**);**

**res.status(200).json({ message: "Signin successful", token, userData });**

**} catch (error) {**

**console.error(error);**

**res.status(500).json({ message: "Internal server error" });**

**}**

**};**

**const forgotPassword = async (req, res) => {**

**const { email } = req.body;**

**try {**

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

**if (!user) {**

**return res.status(404).json({ message: "User not found" });**

**}**

**const generateOTP = () => {**

**const characters = "0123456789";**

**return Array.from(**

**{ length: 6 },**

**() => characters[Math.floor(Math.random() \* characters.length)]**

**).join("");**

**};**

**const OTP = generateOTP();**

**user.resetPasswordOtp = OTP;**

**user.resetPasswordExpires = Date.now() + 3600000;**

**await user.save();**

**const transporter = nodemailer.createTransport({**

**service: "gmail",**

**auth: {**

**user: process.env.USER\_MAILER,**

**pass: process.env.PASS\_MAILER,**

**},**

**});**

**const mailOptions = {**

**from: "eyegamers1234@gmail.com",**

**to: user.email,**

**subject: "Password Reset",**

**html: `**

**<p>Dear ${user.firstName} ${user.lastName},</p>**

**<p>We received a request to reset your password. Here is your One-Time Password (OTP): <strong>${OTP}</strong></p>**

**<p>Please click the following link to reset your password:</p>**

**<a href="https://password-reset-flows.netlify.app/reset-password">Reset Password</a>**

**<p>If you did not make this request, please ignore this email.</p>**

**<p>Thank you,</p>**

**<p>From Validation</p>**

**`,**

**};**

**await transporter.sendMail(mailOptions);**

**res.status(200).json({ message: "Password reset email sent successfully" });**

**} catch (error) {**

**console.error(error);**

**res.status(500).json({**

**message: "Internal Server Error",**

**error: error.message,**

**});**

**}**

**};**

**const resetPassword = async (req, res) => {**

**try {**

**const { OTP, password } = req.body;**

**const user = await userModel.findOne({**

**resetPasswordOtp: OTP,**

**resetPasswordExpires: { $gt: Date.now() },**

**});**

**if (!user) {**

**const message = user ? "OTP has expired" : "Invalid OTP";**

**return res.status(404).json({ message });**

**}**

**const hashedPassword = await auth.hashPassword(password);**

**user.password = hashedPassword;**

**user.resetPasswordOtp = null;**

**user.resetPasswordExpires = null;**

**await user.save();**

**res.status(200).json({ message: "Password changed successfully" });**

**} catch (error) {**

**console.error(error);**

**res.status(500).json({ message: "Internal Server Error" });**

**}**

**};**

**export default {**

**signupController,**

**signinController,**

**resetPassword,**

**forgotPassword,**

**};**

**Under models subfolder**

**User.model.js**

**import mongoose from "mongoose";**

**const validateEmail = (e) => {**

**var emailPattern = /^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;**

**return emailPattern.test(e);**

**};**

**const userSchema = new mongoose.Schema(**

**{**

**firstName: { type: String, required: [true, "First Name is required"] },**

**lastName: { type: String, required: [true, "Last Name is required"] },**

**email: {**

**type: String,**

**required: [true, "Email is required"],**

**validate: validateEmail,**

**},**

**password: { type: String, required: [true, "Password is required"] },**

**status: { type: Boolean, default: true },**

**role: { type: String, default: "customer" },**

**createdAt: { type: Date, default: Date.now() },**

**resetPasswordOtp: { type: Number },**

**resetPasswordExpires: { type: Date },**

**},**

**{**

**collection: "Users",**

**versionKey: false,**

**}**

**);**

**const userModel = mongoose.model("Users", userSchema);**

**export default userModel;**

**Under routers subfolder**

**User.router.js**

**import express from "express";**

**import UserController from "../controllers/User.controller.js";**

**import validators from '../common/validators.js';**

**const router = express.Router();**

**router.post("/signup", validators.validate("signup"), validators.validationMiddleware, UserController.signupController);**

**router.post("/signin", validators.validate("signin"), validators.validationMiddleware, UserController.signinController);**

**router.post("/forgot-password", validators.validate("forgotPassWord"), validators.validationMiddleware, UserController.forgotPassword);**

**router.post("/reset-password", validators.validate("resetPassWord"), validators.validationMiddleware, UserController.resetPassword);**

**export default router;**

**router.js**

**import express from "express";**

**import UserRoutes from "./User.router.js";**

**const router = express.Router();**

**router.use("/user", UserRoutes);**

**export default router;**