//server.js  
  
require('dotenv').config()

const express = require('express')

const mongoose = require('mongoose')

const cors = require('cors')

const cookieParser = require('cookie-parser')

const authRouter = require('./controllers/auth')

const userRouter = require('./controllers/user')

const { verifyToken } = require('./middleware/jwtUtils')

const app = express()

const PORT = process.env.PORT || 3000

mongoose.connect(process.env.MONGODB\_URI)

mongoose.connection.on('connected', () => {

  console.log(`Connected to MongoDB ${mongoose.connection.name}.`)

})

app.use(express.json())

app.use(cookieParser())

app.use(cors({ origin: process.env.FRONT\_END\_URL, credentials: true }))

app.use('/auth', authRouter)

app.use('/user', verifyToken, userRouter)

app.use((err, req, res, next) => {

  console.error(err.stack)

  res

    .status(err.status || 500)

    .json({ error: err.message || 'Internal Server Error' })

})

app.listen(PORT, () => {

  console.log(`The express app is ready on port ${PORT}!`)

})

.env

MONGODB\_URI=mongodb+srv://@cluster0.vlh04.mongodb.net/MFA?retryWrites=true&w=majority&appName=Cluster0

PORT=3000

JWT\_SECRET=sdjfh

SALT=10

FRONT\_END\_URL=http://localhost:5173

//user.js

const mongoose = require('mongoose')

const userSchema = new mongoose.Schema(

  {

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

    username: { type: String, unique: true, required: true },

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

    avatar: { type: String },

    createdAt: { type: Date, default: Date.now }

  },

  { timestamps: true }

)

module.exports = mongoose.model('User', userSchema)

//jwtUtils.js

const jwt = require('jsonwebtoken')

const signToken = (user) => {

  return jwt.sign({ \_id: user.\_id }, process.env.JWT\_SECRET, {

    expiresIn: '7d'

  })

}

const verifyToken = (req, res, next) => {

  const token =

    req.cookies?.token || req.headers.authorization?.split('Bearer ')[1]

  if (!token) {

    return res.status(401).json({ error: 'Unauthorized. Token missing.' })

  }

  try {

    const decoded = jwt.verify(token, process.env.JWT\_SECRET)

    req.user = decoded

    next()

  } catch (error) {

    console.error('JWT Verification Error:', error.message)

    return res.status(401).json({ error: 'Unauthorized. Invalid token.' })

  }

}

module.exports = { signToken, verifyToken }

//user.js

const User = require('../models/user')

const router = require('express').Router()

const { verifyToken } = require('../middleware/jwtUtils')

router.get('/profile', verifyToken, async (req, res) => {

  try {

    const user = await User.findById(req.user.\_id).select('-password')

    if (!user) {

      return res.status(404).json({ error: 'User not found.' })

    }

    res.status(200).json(user)

  } catch (error) {

    console.error('Error fetching profile:', error.message)

    res.status(500).json({ error: 'Internal Server Error' })

  }

})

module.exports = router

//auth.js

const { signToken } = require('../middleware/jwtUtils')

const User = require('../models/user')

const bcrypt = require('bcrypt')

const router = require('express').Router()

const jwt = require('jsonwebtoken')

router.post('/signup', async (req, res) => {

  try {

    const { username, password, email } = req.body

    if (!username || !password || !email)

      return res.status(400).json({ error: 'Missing required fields.' })

    const userExist = await User.findOne({ username })

    if (userExist)

      return res.status(409).json({ error: 'Username already taken.' })

    const hashedPassword = await bcrypt.hash(password, +process.env.SALT || 10)

    const user = await User.create({

      username,

      password: hashedPassword,

      email

    })

    const token = signToken(user)

    return res.status(201).json({ message: 'User created successfully', token })

  } catch (error) {

    console.error(error)

    res.status(500).json({ error: 'Something went wrong!' })

  }

})

router.post('/signin', async (req, res) => {

  try {

    const { username, password } = req.body

    if (!username || !password) {

      return res.status(400).json({ error: 'Missing required fields.' })

    }

    const user = await User.findOne({ username })

    if (!user) {

      return res.status(401).json({ error: 'Invalid Credentials!' })

    }

    const matched = await bcrypt.compare(password, user.password)

    if (!matched) {

      return res.status(401).json({ error: 'Invalid Credentials!' })

    }

    const token = jwt.sign({ \_id: user.\_id }, process.env.JWT\_SECRET, {

      expiresIn: '7d'

    })

    const age = 1000 \* 60 \* 60 \* 24 \* 7 *// 7 days*

    res

      .cookie('token', token, { httpOnly: true, maxAge: age })

      .status(200)

      .json({ message: 'Signin successfully', token })

  } catch (error) {

    console.error('Signin Error:', error.message)

    res.status(500).json({ error: 'Failed to Signin!' })

  }

})

router.post('/signout', async (req, res) => {

  try {

    res

      .clearCookie('token')

      .status(201)

      .json({ message: 'Signout Successfully!' })

  } catch (error) {

    console.error(error)

    res.status(500).json({ error: 'Failed to login!' })

  }

})

module.exports = router