mkdir simple-blog-api

cd simple-blog-api

npm init -y

npm install express mongoose bcrypt jsonwebtoken dotenv

touch app.js

mkdir routes controllers models middlewares

// db/connect.js

const mongoose = require('mongoose');

require('dotenv').config();

function connectionToMongodb() {

mongoose.connect(process.env.MONGODB\_URL || 'mongodb://localhost:27017/blogDB');

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

console.log('MongoDB connection successful');

});

mongoose.connection.on('error', (err) => {

console.error(err);

console.log('MongoDB connection unsuccessful');

});

}

module.exports = { connectionToMongodb };

// config/logger.js

const { createLogger, transports, format } = require('winston');

const logger = createLogger({

level: 'info',

format: format.combine(format.timestamp(), format.json()),

transports: [

new transports.Console(),

new transports.File({ filename: 'error.log', level: 'error' }),

new transports.File({ filename: 'combined.log' }),

],

});

module.exports = logger;

/ models/users.js

const mongoose = require('mongoose');

const bcrypt = require('bcrypt');

const userSchema = new schema({

\_id: {

type: String,

default: shortid.generate

},

first\_name: { type: String, required: true },

last\_name: { type: String, required: true },

email: {

type: String,

required: true,

unique: true,

validate: {

validator: function (v) {

return /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(v);

},

message: props => `${props.value} is not a valid email address!`

}

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

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

profilePicture: { type: String }

})

//before save

userSchema.pre("save", async function (next) {

const user = this;

const hash = await bcrypt.hash(user.password, 10);

this.password = hash;

next();

})

userSchema.methods.isValidPassword = async function (password) {

const user = this;

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

return compare;

}

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

module.exports = userModel;

// models/blogs.js

const mongoose = require('mongoose');

const shortid = require('shortid');

const blogSchema = new schema({

\_id: {

type: String,

default: shortid.generate

},

title: { type: String, required: false },

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

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

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

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

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

publishedAt:{type: Date},

state: { type: String,

default: 'draft'},

user\_id:{type:mongoose.Schema.Types.String,

ref:"users"

},

read\_count: { type: Number, required: true, default: 0 },

reading\_time: { type: Number },

body:{type:String, required: false}

})

blogSchema.pre('save', function (next) {

this.updatedAt = new Date();

next();

});

const blogModel = mongoose.model("blogs", blogSchema);

module.exports =blogModel;

# **Implementing User Routes and Controllers**

1. User Routes: Create user routes in the routes/userRouter.js file.

// routes/userRouter.js

const express = require('express');

const controller = require('../controllers/userController');

const userRouter = express.Router();

userRouter.post('/signup', controller.createUser);

userRouter.post('/login', controller.login);

module.exports = userRouter;

2.User Controller: Implement user controller functions in the controllers/userController.js file.

// controllers/userController.js

const userModel = require('../models/users');

const jwt = require('jsonwebtoken');

require('dotenv').config();

const createUser = async (req, res) => {

const { first\_name, last\_name, email, password, country } = req.body;

try {

const existingUser = await userModel.findOne({

email: email,

});

if (existingUser) {

return res.redirect("/existinguser");

}

const user = await userModel.create({

first\_name: first\_name,

last\_name: last\_name,

email: email,

password: password,

country: country,

});

const JWT\_SECRET = process.env.JWT\_SECRET;

const token = await jwt.sign(

{ first\_name: user.first\_name, email: user.email, \_id: user.\_id },

JWT\_SECRET

);

res.status(302).redirect("/login");

} catch (error) {

console.log(error);

res.status(500).message("Internal Server Error");

}

};

const login = async (req, res) => {

const { email, password } = req.body;

try {

const user = await userModel.findOne({

email: email,

});

// console.log(user)

if (!user) {

return res.status(404).redirect("/signup");

}

const validPassword = await user.isValidPassword(password);

console.log(email);

if (!validPassword) {

return res.status(302).redirect("/unknown");

}

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

expiresIn: "1h",

});

res.cookie("token", token, { httpOnly: true }, { maxAge: 60 \* 60 \* 1000 });

res.status(200).redirect("/create");

} catch (error) {

console.log(error);

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

}

};

module.exports = {

createUser,

login,

};