Create the User Model (models/User.js):

js

Copy code

const mongoose = require('mongoose');

const bcrypt = require('bcryptjs');

const UserSchema = new mongoose.Schema({

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

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

password: { type: String, required: true }

});

// Hash password before saving

UserSchema.pre('save', async function (next) {

if (!this.isModified('password')) return next();

const salt = await bcrypt.genSalt(10);

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

});

// Compare entered password with hashed password

UserSchema.methods.matchPassword = async function (password) {

return await bcrypt.compare(password, this.password);

};

module.exports = mongoose.model('User', UserSchema);

3.3. Create the Bookmark Model (models/Bookmark.js):

js

Copy code

const mongoose = require('mongoose');

const BookmarkSchema = new mongoose.Schema({

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

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

description: { type: String },

tags: [String],

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

user: { type: mongoose.Schema.Types.ObjectId, ref: 'User' }

});

module.exports = mongoose.model('Bookmark', BookmarkSchema);

3.4. Create Authentication Routes (routes/auth.js):

js

Copy code

const express = require('express');

const bcrypt = require('bcryptjs');

const jwt = require('jsonwebtoken');

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

const router = express.Router();

// Register User

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

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

try {

const userExists = await User.findOne({ email });

if (userExists) return res.status(400).json({ message: 'User already exists' });

const user = new User({ username, email, password });

await user.save();

const token = jwt.sign({ userId: user.\_id }, process.env.JWT\_SECRET, { expiresIn: '1h' });

res.status(201).json({ token });

} catch (err) {

res.status(500).json({ message: 'Server Error' });

}

});

// Login User

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

const { email, password } = req.body;

try {

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

if (!user) return res.status(400).json({ message: 'Invalid credentials' });

const isMatch = await user.matchPassword(password);

if (!isMatch) return res.status(400).json({ message: 'Invalid credentials' });

const token = jwt.sign({ userId: user.\_id }, process.env.JWT\_SECRET, { expiresIn: '1h' });

res.json({ token });

} catch (err) {

res.status(500).json({ message: 'Server Error' });

}

});

module.exports = router;

3.5. Create Bookmark Routes (routes/bookmark.js):

js

Copy code

const express = require('express');

const Bookmark = require('../models/Bookmark');

const router = express.Router();

const { authenticate } = require('../middleware/auth');

// Add Bookmark

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

const { url, title, description, tags, category } = req.body;

const newBookmark = new Bookmark({ url, title, description, tags, category, user: req.userId });

try {

await newBookmark.save();

res.status(201).json(newBookmark);

} catch (err) {

res.status(400).json({ message: 'Error adding bookmark' });

}

});

// Get Bookmarks

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

try {

const bookmarks = await Bookmark.find({ user: req.userId });

res.json(bookmarks);

} catch (err) {

res.status(400).json({ message: 'Error fetching bookmarks' });

}

});

// Search Bookmarks

router.get('/search', authenticate, async (req, res) => {

const { query } = req.query;

try {

const bookmarks = await Bookmark.find({

$or: [

{ title: { $regex: query, $options: 'i' } },

{ tags: { $regex: query, $options: 'i' } },

{ category: { $regex: query, $options: 'i' } },

],

user: req.userId,

});

res.json(bookmarks);

} catch (err) {

res.status(400).json({ message: 'Error searching bookmarks' });

}

});

module.exports = router;

3.6. Create Middleware for Authentication (middleware/auth.js):

js

Copy code

const jwt = require('jsonwebtoken');

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

const token = req.header('Authorization') && req.header('Authorization').split(' ')[1];

if (!token) return res.status(401).json({ message: 'No token, authorization denied' });

try {

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

req.userId = decoded.userId;

next();

} catch (err) {

res.status(401).json({ message: 'Token is not valid' });

}

};

module.exports = { authenticate };

3.7. Create the Server File (server.js):

js

Copy code

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const dotenv = require('dotenv');

const authRoutes = require('./routes/auth');

const bookmarkRoutes = require('./routes/bookmark');

dotenv.config();

const app = express();

app.use(bodyParser.json());

// Connect to MongoDB

mongoose.connect(process.env.MONGO\_URI, { useNewUrlParser: true, useUnifiedTopology: true })

.then(() => console.log('MongoDB connected'))

.catch(err => console.log(err));

// Routes

app.use('/api/auth', authRoutes);

app.use('/api/bookmarks', bookmarkRoutes);

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => {

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

});