Frontend: React, Angular veya Vue.js.

Backend: Node.js, Express.js veya Python (Django/Flask).

Veritabanı: PostgreSQL, MySQL veya MongoDB.

Authentication: JSON Web Tokens (JWT).

Deployment: Docker, Kubernetes, AWS, Azure.

Backend: Node.js + Express.j

mkdir is-takip-projesi

cd is-takip-projesi

npm init -y

npm install express mongoose dotenv body-parser jsonwebtoken bcrypt cors

/

├── server.js

├── config/

│ └── database.js

├── models/

│ ├── User.js

│ ├── Task.js

├── routes/

│ ├── userRoutes.js

│ ├── taskRoutes.js

├── middleware/

│ └── authMiddleware.js

├── controllers/

│ ├── userController.js

│ ├── taskController.js

└── .env

config/database.js

const mongoose = require('mongoose');

const connectDB = async () => {

try {

await mongoose.connect(process.env.MONGO\_URI, {

useNewUrlParser: true,

useUnifiedTopology: true,

});

console.log('MongoDB Connected...');

} catch (error) {

console.error(error.message);

process.exit(1);

}

};

module.exports = connectDB;

models/User.js

const mongoose = require('mongoose');

const bcrypt = require('bcrypt');

const UserSchema = new mongoose.Schema({

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

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

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

});

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);

next();

});

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

models/Task.js

const mongoose = require('mongoose');

const TaskSchema = new mongoose.Schema({

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

description: { type: String },

status: { type: String, default: 'Pending' },

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

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

});

module.exports = mongoose.model('Task', TaskSchema);

controllers/userController.js

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

const jwt = require('jsonwebtoken');

exports.registerUser = async (req, res) => {

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

try {

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

await user.save();

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

} catch (error) {

res.status(400).json({ error: error.message });

}

};

exports.loginUser = async (req, res) => {

const { email, password } = req.body;

try {

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

if (!user) return res.status(404).json({ message: 'User not found' });

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

if (!isMatch) return res.status(401).json({ message: 'Invalid credentials' });

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

res.status(200).json({ token });

} catch (error) {

res.status(500).json({ error: error.message });

}

};

controllers/taskController.js

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

exports.createTask = async (req, res) => {

const { title, description, assignedTo } = req.body;

try {

const task = new Task({ title, description, assignedTo });

await task.save();

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

} catch (error) {

res.status(400).json({ error: error.message });

}

};

exports.getTasks = async (req, res) => {

try {

const tasks = await Task.find().populate('assignedTo', 'name email');

res.status(200).json(tasks);

} catch (error) {

res.status(500).json({ error: error.message });

}

};

server.js

const express = require('express');

const connectDB = require('./config/database');

const dotenv = require('dotenv');

const userRoutes = require('./routes/userRoutes');

const taskRoutes = require('./routes/taskRoutes');

dotenv.config();

connectDB();

const app = express();

app.use(express.json());

app.use('/api/users', userRoutes);

app.use('/api/tasks', taskRoutes);

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

app.listen(PORT, () => console.log(Server running on port ${PORT}));