**📄 Proje Dokümanı: DailyHub - Daily Journal & Goal Tracker App**

**1. Proje Tanımı**

DailyHub, kullanıcıların günlük yazıları oluşturabileceği, hedefler belirleyebileceği ve ruh hallerini takip edebileceği bir web tabanlı uygulamadır. Kullanıcılar kayıt olabilir, giriş yapabilir ve sadece kendi içeriklerini yönetebilirler. Uygulama full-stack olarak geliştirilecektir.

**2. Teknoloji Yığını**

**Frontend:**

* React.js (Vite ile oluşturulmuş proje)
* React Router
* Context API veya useReducer (auth state yönetimi için)
* Tailwind CSS veya sade CSS
* Axios (API istekleri için)

**Backend:**

* Node.js
* Express.js
* MongoDB (Mongoose ODM ile)
* JWT Authentication
* bcrypt (şifre şifreleme için)

**3. Ana Özellikler**

| **Özellik** | **Açıklama** |
| --- | --- |
| Kayıt Ol (Register) | Kullanıcı e-posta ve şifre ile kayıt olur. |
| Giriş Yap (Login) | Kullanıcı giriş yapar ve JWT token alır. |
| Günlük Yazısı Ekleme | Kullanıcı tarih, başlık, içerik ve ruh hali seçerek günlük kaydı ekler. |
| Hedef Belirleme | Kullanıcı başlık, açıklama ve tarih belirleyerek hedef ekler. |
| CRUD İşlemleri | Günlük ve hedefler için oluşturma, okuma, güncelleme ve silme işlemleri yapılır. |
| Ruh Hali Takibi | Dashboard'da ruh halleri istatistik olarak gösterilir. |
| Profil Sayfası | Kullanıcı kendi bilgilerini görüntüler ve düzenler. |
| Responsive Design | Mobil ve masaüstü uyumlu tasarım. |

**4. Kullanıcı Akışı (User Flow)**

1. Kullanıcı ana sayfaya gelir ve giriş yapar / kayıt olur.
2. Başarılı giriş sonrası Dashboard ekranına yönlendirilir.
3. Dashboard'da:
   * Yeni bir günlük girişi yapabilir.
   * Yeni bir hedef ekleyebilir.
   * Mevcut günlüklerini ve hedeflerini listeleyebilir ve düzenleyebilir.
4. Profil sayfasından bilgilerini güncelleyebilir.
5. Logout ile oturumunu sonlandırır.

**5. API Tasarımı**

**Auth Routes**

* POST /api/auth/register → Yeni kullanıcı oluşturur
* POST /api/auth/login → Giriş yapar ve token döner
* GET /api/auth/profile → Kullanıcı bilgilerini döner

**Journal Routes**

* POST /api/journals → Yeni günlük ekler
* GET /api/journals → Kullanıcının günlüklerini listeler
* PUT /api/journals/:id → Günlüğü günceller
* DELETE /api/journals/:id → Günlüğü siler

**Goal Routes**

* POST /api/goals → Yeni hedef ekler
* GET /api/goals → Kullanıcının hedeflerini listeler
* PUT /api/goals/:id → Hedefi günceller
* DELETE /api/goals/:id → Hedefi siler

**6. Veritabanı Modelleri**

**User Model**

javascript

KopyalaDüzenle

{

username: String,

email: String,

password: String (hashed),

createdAt: Date

}

**Journal Entry Model**

javascript

KopyalaDüzenle

{

userId: ObjectId,

date: Date,

title: String,

content: String,

mood: String ("happy", "sad", "neutral", etc.),

createdAt: Date

}

**Goal Model**

javascript

KopyalaDüzenle

{

userId: ObjectId,

title: String,

description: String,

targetDate: Date,

completed: Boolean,

createdAt: Date

}

**7. Sayfa ve Component Yapısı**

**Sayfalar**

* /login
* /register
* /dashboard
* /profile
* /new-journal
* /new-goal

**Ana Bileşenler (Components)**

* Navbar
* Footer
* JournalList
* JournalForm
* GoalList
* GoalForm
* MoodSelector
* ProfileEditor
* PrivateRoute (korumalı route)

**8. Kullanılacak Önemli React Hook'lar**

* **useState:** Form verilerini, modal durumlarını yönetmek
* **useEffect:** API çağrıları ve lifecycle yönetimi
* **useContext:** Kullanıcı Auth verilerini globalde tutmak
* **useReducer:** Daha kompleks state yönetimi (isteğe bağlı)
* **useRef:** Form inputları veya API isteklerinde kontrol için
* **useNavigate (React Router v6):** Sayfa yönlendirmeleri için

**9. İlerleme Planı (Sprint Planı)**

| **Aşama** | **Görev** | **Süre** |
| --- | --- | --- |
| 1 | Proje kurulumları ve temel yapı | 2 gün |
| 2 | Backend Auth Sistemi (register/login) | 2 gün |
| 3 | Backend Journal ve Goal API'ları | 3 gün |
| 4 | Frontend Auth Sayfaları ve Context kurulumu | 2 gün |
| 5 | Dashboard ve CRUD işlemleri | 3 gün |
| 6 | Profil sayfası ve düzenlemeler | 2 gün |
| 7 | Responsive tasarım + son testler | 2 gün |
| 8 | Deploy ve bitiş | 1 gün |

Toplam: **~17 gün (günde 2-3 saat çalışarak rahat yapılır.)**

**10. Gereksinimler**

* Node.js yüklü olmalı (v18+)
* MongoDB Atlas hesabı veya local MongoDB kurulumu
* Postman (API testleri için önerilir)
* Vercel veya Netlify hesabı (frontend deploy için)
* Render veya Railway hesabı (backend deploy için)

**🔥 Bitti!**

Bu dokümanla gerçek bir freelance veya iş projelerinde kullanılacak şekilde planlı bir şekilde ilerleyebilirsin.

İstersen şimdi:  
✅ Sana **görev görev tasklist** çıkartabilirim (mesela Trello kartı gibi ilerlemek için).  
✅ Veya **bir sonraki adımda Backend'in dosya yapısını nasıl oluşturman gerektiğini** gösterebilirim.

**Nasıl devam edelim? 🚀**  
(İstersen başka bir projeye de geçeriz tabii.)

4o

**Şunu dedin:**

tasklist cikart

**ChatGPT:**

Süper! Hemen **DailyHub** projesi için detaylı bir **tasklist** çıkarıyorum.  
Bunu adım adım işleyebilirsin; sanki bir Trello veya Jira board'una yapıştırılmış gibi düşünebilirsin.

**📋 DailyHub - Proje Tasklist**

**0. Başlangıç**

* Proje klasörü oluştur (frontend ve backend ayrı klasörler)
* Github repo oluştur (istersen ayrı ayrı veya monorepo şeklinde)

**1. Backend Kurulum ve Yapı**

* Backend için npm init yap
* Express ve temel kütüphaneleri kur (express, mongoose, cors, dotenv, bcryptjs, jsonwebtoken)
* server.js dosyasını oluştur ve çalıştığından emin ol
* MongoDB bağlantısı kur (local veya MongoDB Atlas)

**2. Backend - Authentication Modülü**

* User modeli oluştur (username, email, password hash)
* Kayıt ol (Register) API'si yaz
* Giriş yap (Login) API'si yaz
* JWT token üretimi ve doğrulaması
* Auth middleware oluştur (korumalı route için)

**3. Backend - Journal API**

* Journal modeli oluştur (userId, title, content, mood, date)
* Yeni günlük ekleme (POST /journals)
* Günlükleri listeleme (GET /journals)
* Günlük düzenleme (PUT /journals/:id)
* Günlük silme (DELETE /journals/:id)

**4. Backend - Goal API**

* Goal modeli oluştur (userId, title, description, targetDate, completed)
* Yeni hedef ekleme (POST /goals)
* Hedefleri listeleme (GET /goals)
* Hedef düzenleme (PUT /goals/:id)
* Hedef silme (DELETE /goals/:id)

**5. Frontend Kurulum ve Yapı**

* Vite ile yeni React projesi başlat
* Tailwind CSS entegre et (veya sade CSS kullan)
* React Router kur (react-router-dom)
* Axios kur (backend ile iletişim için)

**6. Frontend - Authentication**

* Login sayfası oluştur
* Register sayfası oluştur
* Auth Context oluştur (useContext + useReducer)
* Token'ı localStorage'a kaydet ve kullan
* ProtectedRoute componenti yap (kullanıcı girişi olmadan dashboard'a erişilmesin)

**7. Frontend - Dashboard ve CRUD Sayfaları**

* Dashboard ana ekranı oluştur
* Journal ekleme formu oluştur
* Journal listeleme ve düzenleme sayfası
* Goal ekleme formu oluştur
* Goal listeleme ve düzenleme sayfası
* Mood Selector bileşeni oluştur (günlük için)

**8. Frontend - Profil ve Ayarlar**

* Profil ekranı oluştur
* Profil düzenleme imkanı sağla
* Logout işlemi ekle (token silinsin)

**9. UI/UX İyileştirmeleri**

* Loading spinner ekle (API isteklerinde)
* Hata ve başarı mesajları göster (toast, alert vs.)
* Responsive tasarımı tam optimize et (mobil - tablet - desktop)

**10. Son Testler**

* API'leri Postman ile test et
* Frontend form validasyonlarını kontrol et
* Kötü senaryoları test et (hatalı giriş, boş form vs.)

**11. Deployment**

* Backend'i Render veya Railway üzerinde deploy et
* Frontend'i Vercel veya Netlify üzerinde deploy et
* Çevresel değişkenleri (.env) doğru ayarla (özellikle production için)

**12. Ekstra (Opsiyonel Ama Süper Olur)**

* Kullanıcı şifresi sıfırlama (Forgot Password)
* Kullanıcının ruh hali grafiğini gösteren küçük bir Chart.js dashboard
* Günlükler ve hedefler için arama ve filtreleme özellikleri

**🎯 Kısa Özet:**

İlk adım:  
✅ Backend basit çalışsın (auth + journal CRUD) → sonra  
✅ Frontend giriş-kayıt ekranı + dashboard gelsin → sonra  
✅ Geliştirme ve güzelleştirme :)

**🗂️ Backend (Node.js + Express) Dosya ve Klasör Yapısı**

pgsql

KopyalaDüzenle

backend/

│

├── controllers/

│ ├── authController.js

│ ├── journalController.js

│ └── goalController.js

│

├── models/

│ ├── User.js

│ ├── Journal.js

│ └── Goal.js

│

├── middleware/

│ ├── authMiddleware.js

│

├── routes/

│ ├── authRoutes.js

│ ├── journalRoutes.js

│ └── goalRoutes.js

│

├── config/

│ └── db.js

│

├── utils/

│ └── generateToken.js

│

├── .env

├── server.js

├── package.json

└── README.md

**Özet:**

* controllers/: API işlemlerini burada yazacaksın (login, register, journal CRUD).
* models/: MongoDB şema dosyaları.
* middleware/: Özel middleware'ler (örnek: sadece giriş yapmış kullanıcı görebilir gibi).
* routes/: API endpointlerini burada tanımlıyorsun.
* config/: MongoDB bağlantısı gibi ayarlar.
* utils/: Token üretimi gibi küçük yardımcı fonksiyonlar.
* .env: Gizli bilgilerin (JWT\_SECRET, MongoDB URI vs.)

**🗂️ Frontend (React) Dosya ve Klasör Yapısı**

pgsql

KopyalaDüzenle

frontend/

│

├── public/

│ └── index.html

│

├── src/

│ ├── components/

│ │ ├── Navbar.jsx

│ │ ├── ProtectedRoute.jsx

│ │ ├── MoodSelector.jsx

│ │ └── LoadingSpinner.jsx

│ │

│ ├── contexts/

│ │ └── AuthContext.jsx

│ │

│ ├── pages/

│ │ ├── Login.jsx

│ │ ├── Register.jsx

│ │ ├── Dashboard.jsx

│ │ ├── Journals.jsx

│ │ ├── Goals.jsx

│ │ ├── Profile.jsx

│ │ └── NotFound.jsx

│ │

│ ├── services/

│ │ ├── authService.js

│ │ ├── journalService.js

│ │ └── goalService.js

│ │

│ ├── hooks/

│ │ └── useAuth.js

│ │

│ ├── App.jsx

│ ├── main.jsx

│ ├── index.css (veya Tailwind)

│

├── .env

├── package.json

└── README.md

**Özet:**

* components/: Tekrar kullanılacak ufak parçalar (navbar, spinner, korumalı route).
* contexts/: React Context API kullanarak login bilgisini yönet.
* pages/: Sayfa bazlı bileşenler (Login, Dashboard gibi).
* services/: Axios ile backend API çağrılarını burada yap.
* hooks/: Özel hook'lar (mesela useAuth gibi).
* .env: Frontend için API base URL.

**🚀 Yol Haritası Önerisi (Detaylı Versiyon)**

1. **Backend Temel Çalışır Hâle Getir**  
   ➔ Express kur, server ayağa kalksın, MongoDB bağlansın.
2. **Authentication Bitir**  
   ➔ Register + Login + JWT sistemi backend'de çalışır hale gelsin.
3. **CRUD İşlemleri Yap**  
   ➔ Journals ve Goals modelleri, route'ları ve controller'ları tamamla.
4. **Frontend Kurulumu**  
   ➔ Vite + Tailwind entegre, React Router ayarla.
5. **Login/Register Sayfalarını Yap**  
   ➔ Auth context ile birlikte login/register olayı tamamlansın.
6. **Dashboard ve CRUD UI'ları Kur**  
   ➔ Journal ekle, listele, güncelle, sil  
   ➔ Goal ekle, listele, güncelle, sil
7. **UI Güzelleştir ve Responsive Yap**  
   ➔ Loading spinner, toast mesajlar, mood selector gibi detaylar ekle.
8. **Test Et ve Hataları Düzelt**
9. **Deployment**  
   ➔ Backend: Render  
   ➔ Frontend: Vercel veya Netlify