



softITo YAZILIM VE BİLİŞİM AKADEMİSİ

4. DÖNEM FRONTEND DEVELOPER EĞİTİMİ

GLIDE UYGULAMASI

YAPAY ZEKA DESTEKLİ PROJE YÖNETİM SİSTEMİ (PMO)

BİTİRME PROJESİ RAPORU

Proje Ekibi / Öğrenciler

**Onur Baha Koç • Beyza Mensur • Esra Sağın
Ahsen Dinler • Aslinur Gündüzalp**

Proje Danışmanı ve Değerlendirme Kurulu

Temmuz, 2026

softITo Akademi Yerleşkesi, İstanbul

İÇİNDEKİLER

1. GİRİŞ	3
1.1. Projenin Amacı ve Hedefleri	3
1.2. Proje Kısıtları ve Sınırları	3
2. MATERYAL VE YÖNTEM	4
2.1. React 19 ve Vite 8 Altyapısı	4
2.2. Redux Toolkit ile Global Durum Yönetimi	4
2.3. Tailwind CSS v4 ve Glassmorphic Tasarım Yönergeleri	5
2.4. JSON-Server REST API ile Veri Modelleme	5
2.5. Yapay Zeka Servisi (OpenRouter API) Entegrasyonu	6
3. KURULUM VE UYGULAMA YAPISI	7
3.1. Geliştirme Ortamının Hazırlanması	7
3.2. Proje Dosya ve Dizin Yapısı Analizi	7
3.3. REST API ve JSON Veritabanı Yapılandırması	8
3.4. Uygulamanın Başlatılması	8
3.5. Temel Modüllerin Arayüz Analizleri ve Mockup'lar	9
3.5.1. Giriş Ekranı (Login View)	9
3.5.2. Dashboard ve Metrik Analitiği	10
3.5.3. İş Öğeleri ve Sürükle-Bırak Kanban Panosu	11
3.5.4. Otomasyon Kural Motoru (Automations)	12
3.5.5. Akıllı Yapay Zeka Asistanı (AI Assistant)	13
3.5.6. Döngüler ve Sprint Planlama (Cycles)	14
3.5.7. Modül ve Epik Takibi (Modules)	14
3.5.8. Talep Toplama ve Triage Yönetimi (Intake)	15
3.5.9. Kullanıcı Yönetimi, Rol Yetki Matrisi (Users & Roles)	16
4. SONUÇ	17
5. KAYNAKÇA	17

1. GİRİŞ

Proje yönetimi; rekabetin ve dinamizmin sürekli arttığı modern iş dünyasında operasyonel verimliliğin en temel belirleyicisidir. Günümüzde iş takip ve süreç yönetimi pratikleri sadece yazılım geliştirme ekipleriyle sınırlı kalmayıp; pazarlama kampanyalarından finansal operasyonlara, stratejik yönetim süreçlerinden start-up ölçeklendirme modellerine ve kurumsal Proje Yönetim Ofisi (PMO) departmanlarına kadar tüm sektör ve iş kollarında hayati önem taşımaktadır. Geleneksel proje yönetim sistemleri (Jira, Asana, Azure DevOps vb.) veri yığınları altında hantallaşan yapıları, ekipleri yavaşlatan karmaşık konfigürasyon süreçleri ve yapay zeka teknolojilerini sadece pasif bir sohbet arayüzü olarak barındıran yetersiz entegrasyonları nedeniyle kurumlara fayda sağlamaktan ziyade ek bir operasyonel iş yükü getirmektedir. Bu durum, sürtünmesiz, hızlı ve süreçleri otonom hale getirebilen yeni nesil akıllı PMO araçlarına olan ihtiyacı zirveye taşımıştır.

1.1. Projenin Amacı ve Hedefleri

Bu projenin amacı; farklı iş kolları ve ölçeklerdeki ekiplerin iş süreçlerini uçtan uca, sıfır gecikmeli ve proaktif yapay zeka desteğiyle yönetmesini sağlayan "Glide" adlı web tabanlı akıllı proje yönetim platformunu geliştirmektir. Glide, geleneksel araçların hantal arayüzlerini ve karmaşık iş akışı oluşturma engellerini aşarak, modern kullanıcı deneyimi trendleri ile derinlemesine entegre üretken yapay zeka yeteneklerini tek bir potada eritir. Projenin temel hedefleri şunlardır:

- Yüksek Performanslı ve Sıfır Gecikmeli Arayüz:** Vite tabanlı React mimarisi kullanılarak, sayfa yenilemeleri olmadan (SPA) çalışan, geleneksel araçların aksine milisaniyeler içerisinde anlık veri güncellemeleri sağlayan akıcı bir arayüz sunmak.
- Proaktif ve Bağlam Duyarlı (Context-Aware) AI Entegrasyonu:** Yapay zekanın sadece soru-cevap yapan bir sohbet robotu olarak kalmayıp, arka plandaki tüm ilişkisel proje verilerini (projeler, geciken işler, döngüler, ekip yükü) okuyarak otonom risk analizi yapması, tek tıkla eklenebilen görev kartları üretmesi ve canlı durum özetleri sunması.
- Sürtünmesiz Kodsuz (No-code) Otomasyon Motoru:** Tekrarlayan idari ve operasyonel süreçleri insan müdahalesine gerek kalmadan otomatikleştiren, esnek ve kolayca yapılandırılabilen kural motoru altyapısı kurmak.
- Triage (Eleme) Tabanlı Süreç Güvenliği:** Müşteri veya dış paydaşlardan (Guest) gelen talepleri bağımsız bir Triage Inbox katmanında toplayarak, proje backlog'una kontrolsüz iş girişlerini engellemek ve sistem güvenilirliğini artırmak.

1.2. Proje Kısıtları ve Sınırları

Geliştirilen Glide Proje Yönetim Sistemi'ne ait kısıtlar şu şekildedir:

- Kısıt 1 (Veritabanı Katmanı):** Bu proje kapsamında veri saklama işlemleri ve REST API istekleri `json-server` (Mock DB) kullanılarak simüle edilmiştir. Gerçek bir SQL veya NoSQL veritabanı kurulumu kapsam dışıdır.
- Kısıt 2 (Yapay Zeka Servisi):** Sistemde OpenRouter API bağlantısı entegre edilmiştir. Ancak, geçerli bir API anahtarı girilmediğinde veya internet bağlantısı kesildiğinde asistanın çevrimdışı (Simülasyon/Mock) modunda çalışması hedeflenmiştir.
- Kısıt 3 (Güvenlik Altyapısı):** Oturum yönetimi ve rol bazlı erişim denetimleri (RBAC) frontend katmanında simüle edilmiş olup, sunucu tarafında JWT (JSON Web Token) veya SSL/TLS gibi üretim seviyesi güvenlik protokolleri kapsam dışında tutulmuştur.

2. MATERYAL VE YÖNTEM

Glide uygulamasının geliştirilmesinde, modern web standartlarına uygun, yüksek hızlı ve modüler kütüphaneler tercih edilmiştir. Bu bölümde, sistem mimarisinde kullanılan temel araçlar ve bu araçların çalışma mantıkları detaylandırılmıştır.

2.1. React 19 ve Vite 8 Altyapısı

React 19, bileşen bazlı (Component-based) arayüz tasarımı sağlaması ve Sanal DOM (Virtual DOM) yapısı sayesinde kullanıcı etkileşimlerinde yüksek hız sunması nedeniyle tercih edilmiştir. Geliştirme ortamı ve derleme aracı olarak Vite 8 seçilmiştir. Vite, geleneksel Webpack gibi araçların aksine ES modüllerini (ESM) yerel olarak destekler. Bu durum, geliştirme aşamasında anlık kod güncellemelerinin (HMR) milisaniyeler seviyesinde gerçekleşmesini ve üretim (production) çıktısının minimum dosya boyutuyla optimize edilmesini sağlar.

Tablo 2.1: React ve Vite Geliştirme Parametreleri

Teknoloji	Versiyon	Rolü	Sağladığı Avantaj
React Core	v19.2.7	Bileşen Hiyerarşisi ve DOM Yönetimi	Sanal DOM ile yüksek hızlı ekran çizimi.
React Router	v7.18.1	Sayfa Yönlendirme ve Rotalama	Rol bazlı Protected Rotalar ile erişim denetimi.
Vite	v8.1.1	Derleyici ve Paketleyici	Hızlı HMR ve hızlı canlı önizleme sunucusu.

2.2. Redux Toolkit ile Global Durum Yönetimi

Uygulamanın durumu (state), karmaşık bileşen hiyerarşilerinde veri taşıma problemlerini (prop drilling) engellemek amacıyla Redux Toolkit ile tek bir merkezden (Single Source of Truth) yönetilmektedir. Asenkron API çağrıları (RESTful GET/POST/PUT/DELETE işlemleri) Redux Toolkit Thunk yapısı kullanılarak gerçekleştirilir. Bu sayede veri sunucuda güncellendiği anda global state de senkronize olarak güncellenir ve Kanban panosu gibi dinamik ekranlara anında yansır.

Çalışma Mantığı:

- Bileşen Etkileşimi:** Kullanıcı Kanban panosunda bir kartı sürükleyip başka bir sütuna bıraktığında arayüzde bir olay tetiklenir.
- Action Dispatch:** Bileşen, `updateWorkItem` thunk action'ını yeni durum parametreleriyle tetikler (dispatch eder).
- API İsteği (Thunk):** Thunk, arka planda JSON-Server'a HTTP PUT isteği gönderir.
- Store Güncellemesi:** Sunucudan başarılı yanıt geldiğinde, reducer devreye girerek Redux Store'daki görev bilgisini günceller ve ilgili React bileşenleri otomatik olarak yeniden çizilir (re-render).

2.3. Tailwind CSS v4 ve Glassmorphic Tasarım Yönergeleri

Arayüz tasarımında Tailwind CSS v4 kullanılmıştır. Tailwind CSS, yardımcı sınıflar (utility classes) aracılığıyla harici CSS dosyaları yazmadan doğrudan HTML/JCX içinde stil tanımlama imkanı sunar. Uygulama genelinde premium ve modern bir görünüm elde etmek için **Glassmorphism (Buzlu Cam Efekti)** tasarım dili benimsenmiştir. Bu efekt, arka plan resimlerinin hafifçe görünmesini sağlayan yarı saydam beyaz katmanlar, ince kenarlıklar ve bulanıklaştırma filtreleri ile oluşturulmuştur.

Tasarım Sisteminin CSS Değişkenleri ve Özellikleri:

- **Bulanıklık Katmanı (Backdrop Blur):** `backdrop-blur-lg` sınıfıyla arka plandaki görseller 16px oranında bulanıklaştırılarak içerik okunabilirliği artırılmıştır.
- **Renk Paleti:** Koyu mavi (`#0f172a`), teal (`#0d9488`), cyan (`#06b6d4`) ve gri tonlarından oluşan dengeli ve kurumsal bir renk paleti seçilmiştir.
- **Karanlık Tema (Dark Mode):** CSS değişkenleri ve Tailwind konfigürasyonu üzerinden `html.dark` sınıfı aktif edildiğinde tüm renkler koyu temaya uygun şekilde esner. Arka plan ışıık taşmaları ve kart gölgeleri karanlık modda yumuşatılarak göz yorgunluğu önlenmiştir.

2.4. JSON-Server REST API ile Veri Modelleme

Projede backend veritabanını ve RESTful API servislerini simüle etmek amacıyla `json-server` kütüphanesi yapılandırılmıştır. JSON-Server, projenin kök dizininde bulunan `db.json` dosyasını bir veritabanı gibi okuyup yazarak HTTP GET, POST, PUT, PATCH ve DELETE isteklerine tam işlevli yanıtlar üretir. Bu sayede veritabanı işlemleri gerçeğe en yakın şekilde simüle edilir ve global state güncellemeleri veri tutarlılığıyla sürdürülür. Sistemdeki ilişkisel şema 19 temel veri tablosundan (modülden) oluşmaktadır.

Tablo 2.2: db.json Veritabanı Modülleri ve İlişkisel Şema Endpoints

Endpoint	Veri Türü	Temel Alanlar (Fields)	Açıklama ve İlişki Tanımı
`/workspaces`	Array (Object)	`id`, `name`, `settings`	Sistemdeki üst çalışma alanları (Workspace) yapılandırması.
`/users`	Array (Object)	`id`, `name`, `email`, `password`, `role`, `avatar`	Kullanıcı profilleri, kimlik bilgileri ve sistemdeki RBAC rolleri.
`/workspaceMembers`	Array (Object)	`id`, `workspaceId`, `userId`, `role`	Kullanıcıların çalışma alanları ile üyelik ilişkilerini tutan köprü tablosu.
`/projects`	Array (Object)	`id`, `name`, `key`, `description`, `color`, `status`, `health`, `ownerId`	Workspace altındaki projeler. Sağlık indeksi ('health') ve durum takibini barındırır.
`/projectMembers`	Array (Object)	`id`, `projectId`, `userId`, `role`	Kullanıcıların projelere katılımını ve proje bazlı yetkilerini tutan köprü tablosu.
`/cycles`	Array (Object)	`id`, `name`, `startDate`, `endDate`, `status`, `projectId`	Projeye bağlı çevik (Agile) döngüler / sprintler. Proje tablosuyla `projectId` üzerinden ilişkilidir.
`/modules`	Array (Object)	`id`, `name`, `status`, `projectId`, `description`	Epik seviyesinde mantıksal ürün modülleri (örn: Kimlik Doğrulama, Ödeme Sistemi).
`/workItems`	Array (Object)	`id`, `title`, `desc`, `status`, `priority`, `projectId`, `assigneeId`, `cycleId`, `moduleId`, `dueDate`	Sistemin kalbini oluşturan görevler (Kanban kartları). Proje, kullanıcı, döngü ve modüllerle ilişkilidir.
`/comments`	Array (Object)	`id`, `workItemId`, `userId`, `text`, `timestamp`	İş öğelerine (Work Items) yapılan yorumlar ve tartışma geçmişi kayıtları.
`/wikiPages`	Array (Object)	`id`, `projectId`, `title`, `content`, `creatorId`	Proje bilgi birikimini ve dokümantasyonunu saklayan Wiki sayfaları veri modeli.
`/intakeRequests`	Array (Object)	`id`, `title`, `description`, `status`, `clientName`, `clientEmail`, `projectId`	Dış paydaşların (Guest) Triage (Eleme) kutusunda incelenmeyi bekleyen ilk talep ve hata bildirimleri.
`/automationRules`	Array (Object)	`id`, `projectId`, `trigger`, `condition`, `action`, `isActive`	Kodsuz otomasyon kural motorunun kural tanımlamaları (Tetikleyici, Koşul ve Aksiyon).
`/notifications`	Array (Object)	`id`, `userId`, `title`, `message`, `read`, `timestamp`	Kullanıcılara sistem içi bildirimleri ileten veri tablosu.
`/aiInteractions`	Array (Object)	`id`, `userId`, `projectId`, `prompt`, `response`, `actionType`, `timestamp`	Yapay Zeka asistanı ile yapılan sohbetlerin ve rapor taleplerinin analiz geçmişi günlükleri.
`/reports`	Array (Object)	`id`, `projectId`, `title`, `content`, `generatedAt`	AI asistanı tarafından üretilen ve arşive kaydedilen proje raporları ve özetleri.
`/automations`	Array (Object)	`id`, `ruleId`, `firedAt`, `status`, `details`	Tetiklenen otomasyon kurallarının tarihsel çalışma ve hata log günlükleri.
`/docs`	Array (Object)	`id`, `projectId`, `title`, `url`, `uploadedBy`	Proje dosyaları, ekler ve dokümantasyon eklerini saklayan veri tablosu.

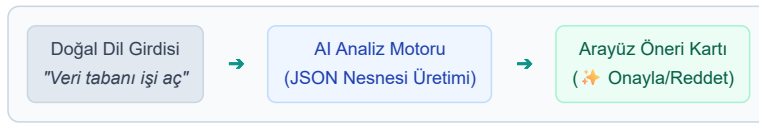
Endpoint	Veri Türü	Temel Alanlar (Fields)	Açıklama ve İlişki Tanımı
`/requests`	Array (Object)	`id`, `title`, `description`, `status`, `type`	Workspace genelindeki geribildirim ve bilet (ticket) verileri.
`/settings`	Array (Object)	`id`, `projectId`, `workflowStates`, `labels`	Projeye özel dinamik iş akışı durumları (states) ve küresel etiket (label) tanımları.

2.5. Yapay Zeka Servisi (OpenRouter API) Entegrasyonu

Glide uygulamasının yapay zeka asistanı, OpenRouter altyapısı üzerinden doğal dil modellerine (örn: Mistral, GPT-4o mini veya Llama 3) bağlanır. Asistanın çalışması için asenkron bir streaming Fetch API entegrasyonu yazılmıştır. Kullanıcı bir soru sorduğunda, istek sistemdeki mevcut proje verileri (aktif görevler, geciken işler, döngü bilgileri) ile zenginleştirilerek (Context Injection) API'ye gönderilir.

Sistemde ayrıca bir ****Fallback (Mock) AI Katmanı**** bulunmaktadır. Yerel ortamda API anahtarı tanımlanmadığında asistan çevrimdışı moda geçer ve kullanıcı taleplerini analiz eden regex tabanlı kurallar çalıştırılır. Çevrimdışı modda bile asistan, görev oluşturma ve risk raporu hazırlama işlevlerini simüle ederek arayüzün çalışmasını kesintisiz sürdürür.

Şekil 2.1: AI Asistanı Doğal Dil Analiz ve Görev Dönüşüm Akışı



AI Asistanının Yapılandırılmış Yanıt Şeması:

Asistan bir görev oluşturmak istediğinde, konuşma akışını bozmamak için normal cevabının sonuna özel bir JSON formatı ekler. React katmanındaki `AIAssistantView` bileşeni, gelen metindeki JSON desenini tespit ederek arayüze özel bir kart çizer. Bu kurgusal yapı, kullanıcının tek tıkla yapay zeka tarafından önerilen görevi projeye kaydetmesini sağlar.

3. KURULUM VE UYGULAMA YAPISI

Bu bölümde Glide uygulamasının yerel geliştirme ortamında nasıl kurulacağı, dizin yapısının detayları ve temel işlevsel modüllerin tasarımı ekran şablonlarıyla gösterilmektedir.

3.1. Geliştirme Ortamının Hazırlanması

Glide, JavaScript çalıştırma ortamı Node.js tabanlıdır. Geliştirme ve derleme adımları için aşağıdaki paketlerin yüklenmesi gerekmektedir:

- 1. Node.js Kurulumu:** Node.js'in resmi web sitesinden LTS (Uzun Süreli Destek) sürümü indirilir ve kurulur. Kurulum bittikten sonra terminalden `node -v` komutuyla sürüm kontrol edilir.
- 2. Bağımlılıkların Yüklenmesi:** Terminalde proje dizinine gidilerek bağımlılıklar kurulur:

```
npm install
```

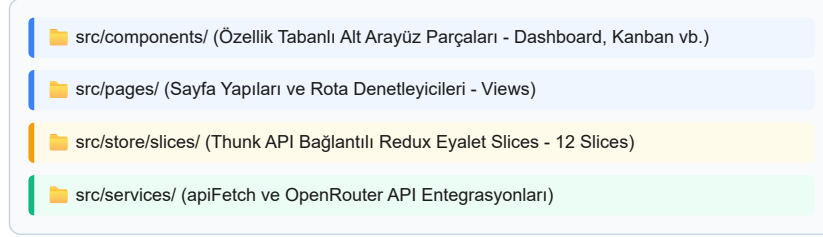
3.2. Proje Dosya ve Dizin Yapısı Analizi

Glide uygulamasının kaynak kodları `/src` dizini altında modüler, sürdürülebilir ve endüstriyel standartlara uygun bir Feature-Folder (özellik tabanlı klasörleme) mimarisiyle yapılandırılmıştır. Bu yapı sayesinde her ana fonksiyonel alan (görev yönetimi, AI asistanı, otomasyonlar vb.) kendi arayüz elemanlarını, veri dilimlerini (Redux slices) ve sayfa denetleyicilerini bağımsız yönetir. Katmanlı dosya hiyerarşisi aşağıdaki gibidir:

- `/src/components` (Tekrar Kullanılabilir ve Modüler UI Bileşenleri):** Uygulamanın her görünümüne ait alt arayüz parçaları burada saklanır. Modüler bir yapıda bölünmüştür:
 - `layout/`:** Uygulamayı sarmalayan `Sidebar` (yan menü), `Header` (üst araç çubuğu) ve RBAC yetki denetimini frontend seviyesinde yürüten `ProtectedRoute` bileşeni.
 - `dashboard/`:** KPI özet kartları, son güncellemeler akışı (`DashboardRecentUpdates`), proje sağlık durum kartları, çark grafikleri ve ****DashboardAiAssistant**** canlı analiz paneli.
 - `workitems/`:** Kanban kartları (`WorkItemKanban`), tablo listeleri (`WorkItemTable`), toplu işlem araç çubuğu ve görev ekleme/düzenleme modalları.
 - `requests/`:** Dış talepleri karşılayan `CreateRequestForm` ve PM eleme ekranı olan `RequestTable`.
 - `ai/`:** AI sohbet arayüzü (`AiChatInterface`), mesaj balonları ve akan metin (typewriter) bileşenleri.
 - `projects/`, `cycles/`, `modules/`, `people/`, `profile/`, `settings/`, `automations/`, `docs/`, `reports/`:** İlgili iş nesnelerine ait görsel kartlar ve alt form elemanları.
- `/src/pages` (Sayfa Denetleyicileri - Route Controllers):** Rotalama (Routing) tablosuna doğrudan bağlanan ana sayfa bileşenleridir. Bu bileşenler alt bileşenleri sarmalar ve sayfa yüklendiğinde gerekli Redux asenkron eylemlerini (actions) dispatch eder. (Örn: `AIAssistantView.jsx`, `DashboardView.jsx`, `ProjectsView.jsx`, `LoginView.jsx`).
- `/src/store` (Redux Global State Yönetimi):** Uygulamanın durum yönetim kalbidir.
 - `store.js`:** Redux store ayarlarının yapıldığı ve slice'ların birleştirildiği ana dosya.
 - `slices/`:** Redux Toolkit ile yazılmış 12 adet veri dilimi (slices). Thunk eylemleriyle backend REST API'den verileri çeker (fetch) ve store'u günceller (örn: `authSlice.js`, `workItemsSlice.js`, `settingsSlice.js`).
- `/src/services` (Network ve Yapay Zeka Entegrasyon Katmanı):** Dış dünya bağlantıları.

- ``api.js``: Tüm API isteklerinde kullanılan, token ekleme ve global hata yakalama (global error-catching) işlemlerini yürüten Fetch API tabanlı ``apiFetch`` modülü.
- ``aiService.js``: OpenRouter API sunucuları ile kurulan streaming (akan veri) bağlantısını yöneten, API anahtarı olmadığında otomatik devreye giren çevrimdışı fallback (simülasyon) modunu barındıran yapay zeka servisi.
- ``/src/Utils`` & ``/src/hooks`` & ``/src/config``: Yardımcı zaman ve renk algoritmaları, ekran veri gruplayıcıları (``dashboardUtils.js``), temalandırmayı yürüten custom hook'lar (``useTheme.js``) ve global yapılandırma ayarları.

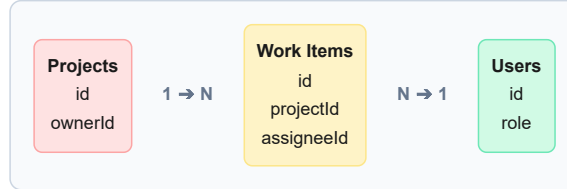
Şekil 3.1: Glide Detaylı Klasör ve Kod Organizasyon Şeması



3.3. REST API ve JSON Veritabanı Yapılandırması

Glide, ilişkisel veritabanı simülasyonunu `db.json` üzerinden yürütür. Sunucu ayarları `package.json` içerisindeki `server` script'ine tanımlanmıştır. JSON Server varsayılan olarak 3001 portunda ayağa kalkar ve HTTP POST/PUT gibi isteklerle `db.json` dosyasını anlık olarak günceller.

Şekil 3.2: db.json Veri İlişki Şeması



3.4. Uygulamanın Başlatılması

Yerel geliştirme ortamında uygulamayı test etmek için iki farklı terminal sekmesinde sunucular çalıştırılır:

Adım 1: Backend Sunucusunun Çalıştırılması:

```
PS C:\Users\Onur\repos\project-management-system> npm run server
> glide@0.0.0 server
> json-server --watch db.json --port 3001
JSON Server started on PORT :3001
Watching db.json...
```

Adım 2: Frontend Sunucusunun Çalıştırılması:

```
PS C:\Users\Onur\repos\project-management-system> npm run dev
VITE v8.1.1 ready in 450 ms
→ Local: http://localhost:5173/
```

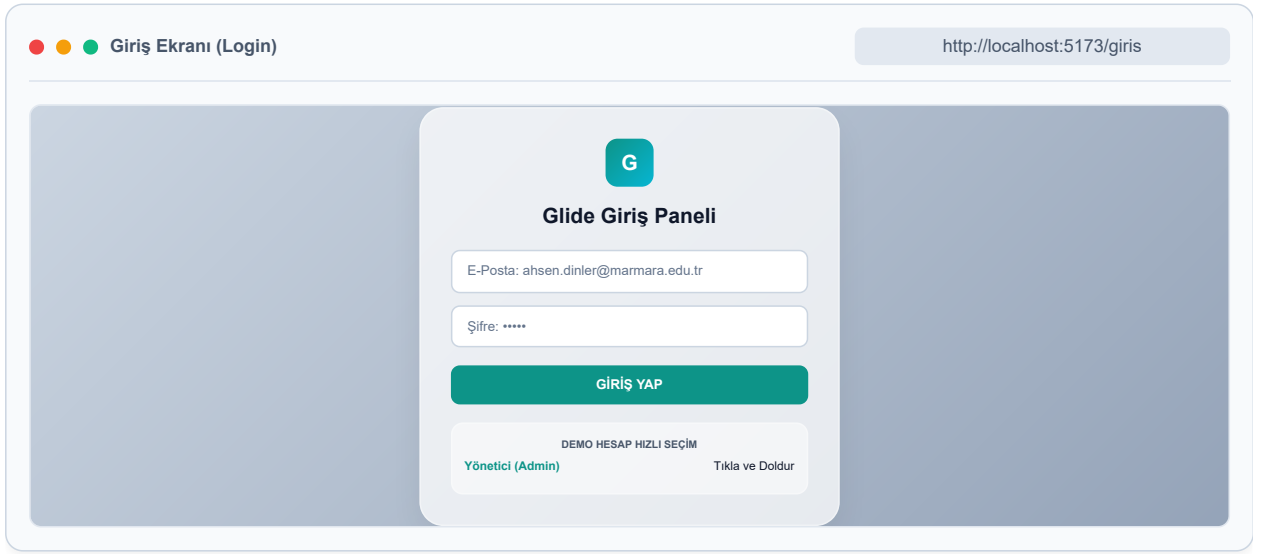
Derleme sonrasında tarayıcı üzerinden giriş ekranına otomatik yönlendirme gerçekleştirilir.

3.5. Temel Modüllerin Arayüz Analizleri ve Mockup'lar

Bu alt bölümde, sistemin ana sayfaları, iş akışları ve kullanıcı deneyimini artırmak için geliştirilen cam efektli arayüzler şablon tasarımlarla incelenmektedir.

3.5.1. Giriş Ekranı (Login View)

Giriş ekranı, arka planda dinamik parlayan ışık süzmeleri (blur blobs) ve yarı saydam glassmorphism kart tasarımına sahiptir. Kullanıcılar kendi kimlik bilgileriyle giriş yapabildikleri gibi, alt kısımda bulunan "Demo Hesaplar" kartlarına tıklayarak sisteme Admin, PM, Member veya Guest rollerinde hızlıca bağlanabilirler. Bu özellik, değerlendirme aşamasında farklı yetkilerin kolayca test edilmesini sağlar.



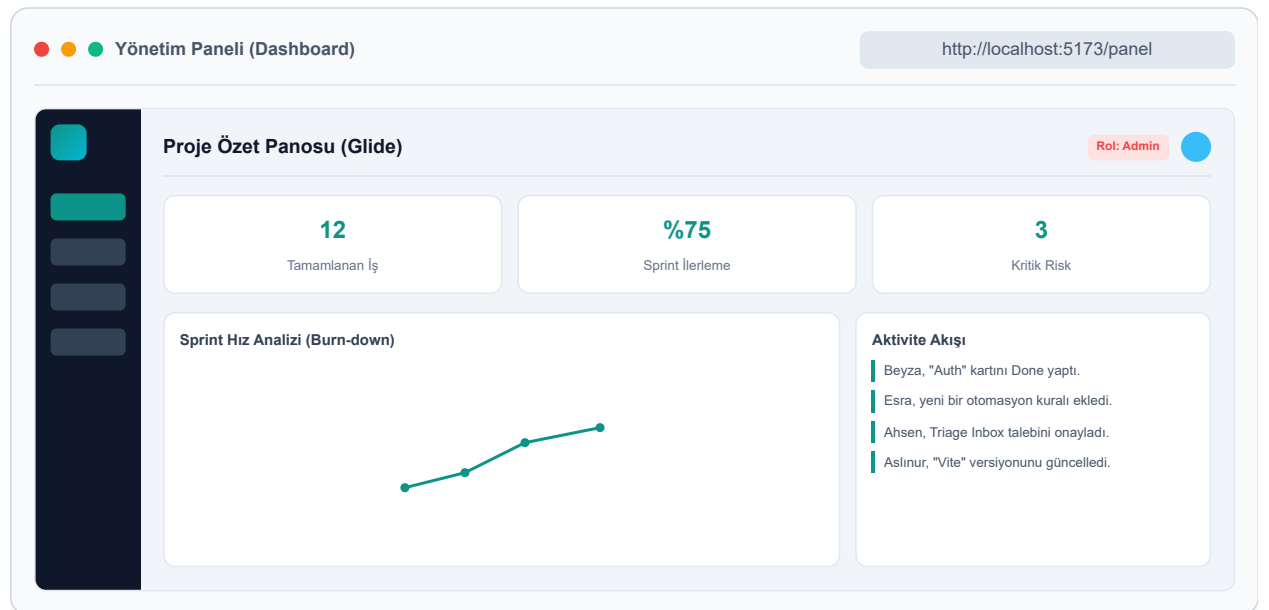
Şekil 3.3: Cam Efektli (Glassmorphic) Login Sayfası Mockup Şablonu

3.5.2. Dashboard ve Metrik Analitiği

Kullanıcı giriş yaptıktan sonra otomatik olarak /panel (DashboardView) sayfasına yönlendirilir. Dashboard, projedeki durum analizlerini ve iş akışını tek bir ekranda derleyen, alt bileşenlerden oluşan son derece kapsamlı ve dinamik bir kontrol panelidir. Dashboard modülü şu temel bileşenlerden oluşmaktadır:

- **DashboardHeader:** Oturum açan kullanıcının ismini karşılar ve sistemdeki etkin rolünün (Admin, PM, Member, Guest) etiketini dinamik olarak gösterir.
- **DashboardStatsCards (5 Dinamik KPI Özet Kartı):** Sayfanın üst kısmında yer alan; Aktif Projeler, Açık Görevler, Kritik Geciken İşler, Aktif Sprint (Cycle) İlerleme Yüzdesi ve Yüksek Öncelikli İş adetlerini listeleyen KPI kartlarıdır. Bu kartlara tıklamak, AI PMO asistanının ilgili KPI odağında (Risk raporu, haftalık ilerleme raporu) anlık özet analizler üretmesini tetikler.
- **DashboardAiAssistant (Glide AI Çalışma Alanı Analizi):** Dashboard'un üst kısmında tam genişlikte yer alan akıllı yapay zeka analiz şerididir. Sayfa yüklendiğinde otomatik olarak (`useEffect` kancası ile) tetiklenir ve o anki aktif projeler, geciken iş riskleri ve ekip verilerini OpenRouter API üzerinden analiz ederek kullanıcıya özel kısa bir "Mevcut Durum Eylem Planı" (AI Özet ve tavsiye raporu) hazırlar.
- **DashboardMyTasks:** Giriş yapan kullanıcıya atanmış ve henüz tamamlanmamış (status != done) olan kişisel görev listesini ve bunların teslim tarihlerini sunar.
- **DashboardActiveCycle:** Çevik (Agile) sprint süreçlerinin durumunu ve tamamlanma barını (Progress) görselleştirir.
- **DashboardProjectHealth:** Çalışma alanındaki projelerin ilerleme oranlarını ve geciken görev sayılarını hesaplayarak proje sağlığını dinamik olarak "Yolunda", "Riskli" veya "Kritik" olarak sınıflandırır.
- **DashboardMemberWorkload:** Ekip üyelerinin aktif iş yüklerini (Low, Medium, High) üstlerindeki görev sayılarına göre analiz eden görsel kapasite panelidir.
- **DashboardRecentUpdates:** Proje genelinde kullanıcıların yaptığı yorumlar, göreve başlama ve tamamlama hareketlerini kronolojik bir akış halinde (Activity Stream) listeler.

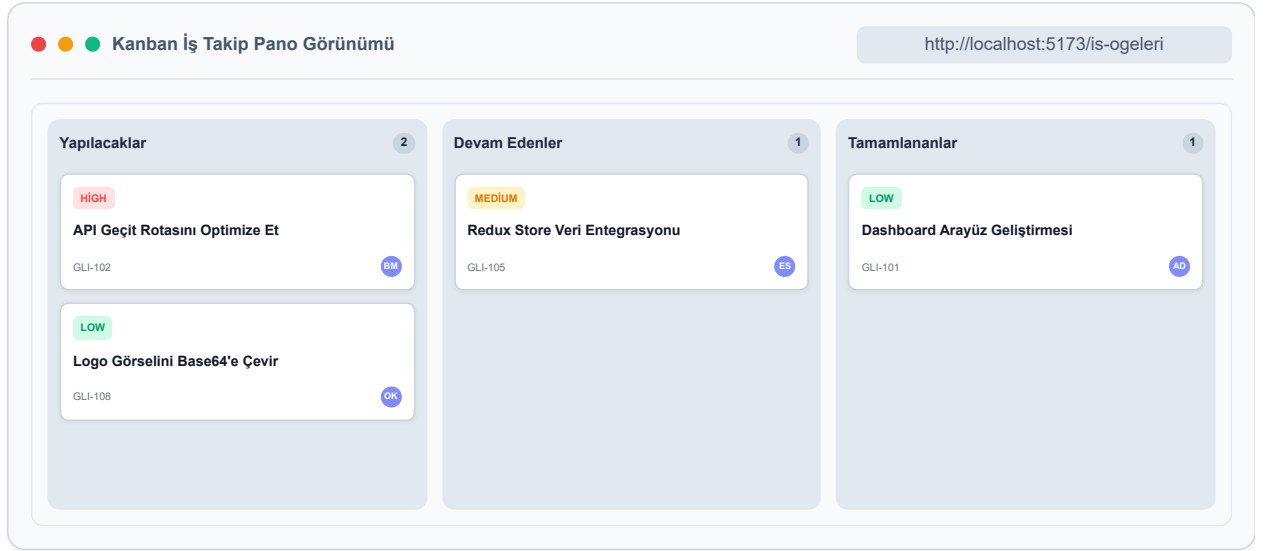
Rollerin yetki sınırlarının test edilebilmesi için sistemde dinamik bir rol değiştirme butonu bulunmamaktadır. Kullanıcılar rolleri değiştirmek için profil menüsünü kullanarak çıkış yapmakta ve giriş ekranındaki "Demo Hesaplar" kartları (Yönetici, PM, Ekip Üyesi, Misafir) yardımıyla farklı yetki sınırlarına sahip kimliklerle sisteme bağlanarak rol duyarlılığını test edebilmektedir.



3.5.3. İş Öğeleri ve Sürükle-Bırak Kanban Panosu

İş öğelerinin yönetildiği Kanban panosu (`WorkItemsView.jsx`), projenin günlük operasyonel akışını düzenleyen merkezdir. Sistem, dinamik sürükle-bırak (drag-and-drop) mekanizmasını kullanarak görevlerin durum geçişlerini görsel olarak yürütür. Kanban panosunun sütun yapısı tamamen dinamiktir; yani Admin panelinden veya veri tabanından (`settings.json` şeması) yeni bir süreç durumu (örn: "In Review") eklendiğinde, Kanban panosu otomatik olarak esneyerek yeni bir sütun kazanır.

Kanban Kart Mimarisi ve Etkileşim: Her görev kartı (`WorkItemKanban.jsx`) üzerinde görevin benzersiz anahtarı (örn: GLI-102), öncelik derecesi (renk kodlu HIGH, MEDIUM, LOW etiketleri), teslim tarihi ve atanan ekip üyesinin avatırı bulunur. Karta tıklandığında açılan `EditWorkItemModal.jsx` detayı üzerinden; görev açıklamaları düzenlenebilir, sprint (cycle) ilişkisi kurulabilir, epik modül atamaları yapılabilir ve iş öğesine yorumlar eklenerek ekip içi tartışma başlatılabilir. Arayüzün üstünde yer alan arama ve çoklu filtreleme barı sayesinde; atanan kişi, öncelik, sprint ve modül bazında anlık süzme işlemleri gerçekleştirilir. Toplu işlemler çubuğu (`BulkActionsBar.jsx`) ise seçilen birden fazla görevin durumunun veya önceliğinin tek tıkla güncellenmesini sağlar.



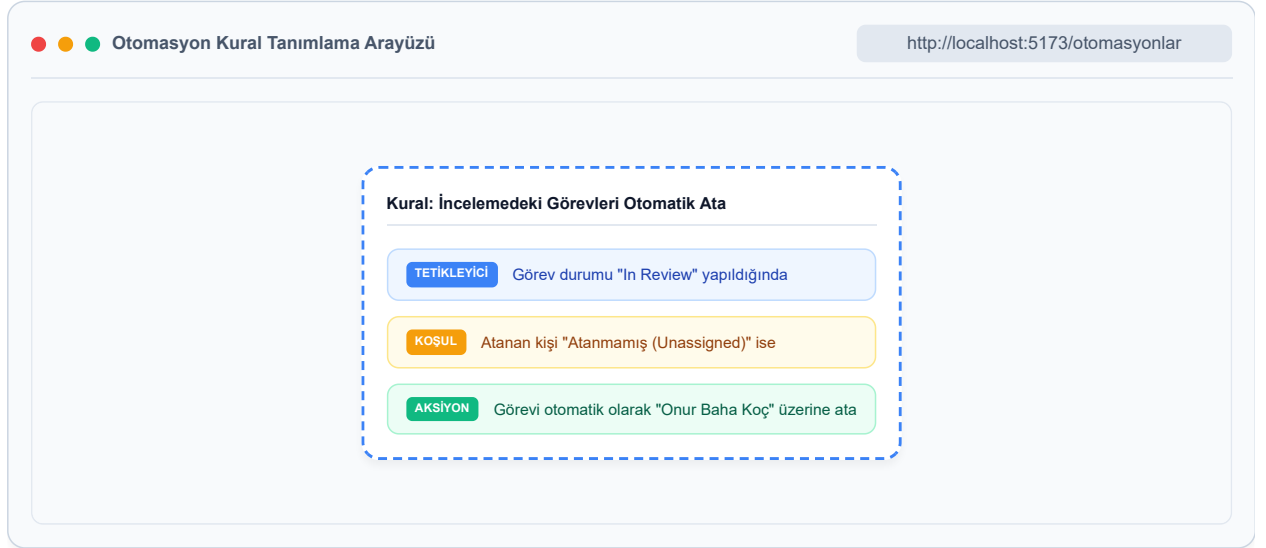
Şekil 3.5: Kanban Pano ve İş Öğeleri Dinamik Sütun Yapısı

3.5.4. Otomasyon Kural Motoru (Automations)

Otomasyon motoru (`AutomationsView.jsx`), ekiplerin günlük yönetim operasyonlarındaki el yordamıyla yaptığı adımları sıfıra indirmek ve insan kaynaklı unutma/hata payını yok etmek için geliştirilmiştir. Sistem, kod yazma gereksinimi olmayan (No-code) görsel bir kural sihirbazıyla çalışır. Kural yapısı; Tetikleyici (Trigger), Koşul (Condition) ve Aksiyon (Action) mantıksal katmanlarından oluşur.

Çalışma Algoritması ve Senaryolar: Bir görev durumu güncellendiğinde veya yeni bir iş oluşturulduğunda kural motoru arka planda tetiklenir. Eğer kuralda tanımlanan mantıksal koşullar sağlanıyorsa, belirlenen aksiyon anında veri tabanına (`json-server`) yansıtılır ve ilgili değişiklikler store üzerinden arayüze dağıtılır. Tetiklenen her otomasyon kuralının başarı durumu ve zaman damgaları `/automations` endpoint'inde loglanarak sistem yöneticisine sunulur.

Örnek Senaryo A (Darboğaz Önleme): "Bir görevin durumu 'In Review' (İncelemede) aşamasına çekildiğinde [Tetikleyici], eğer göreve atanmış bir geliştirici yok ise [Koşul], görevi otomatik olarak Proje Yöneticisine ata [Aksiyon]." Bu sayede inceleme bekleyen görevlerin sahihsiz kalması önlenir.



Şekil 3.6: No-code Otomasyon Kuralları Akış Şeması Mockup Görünümü

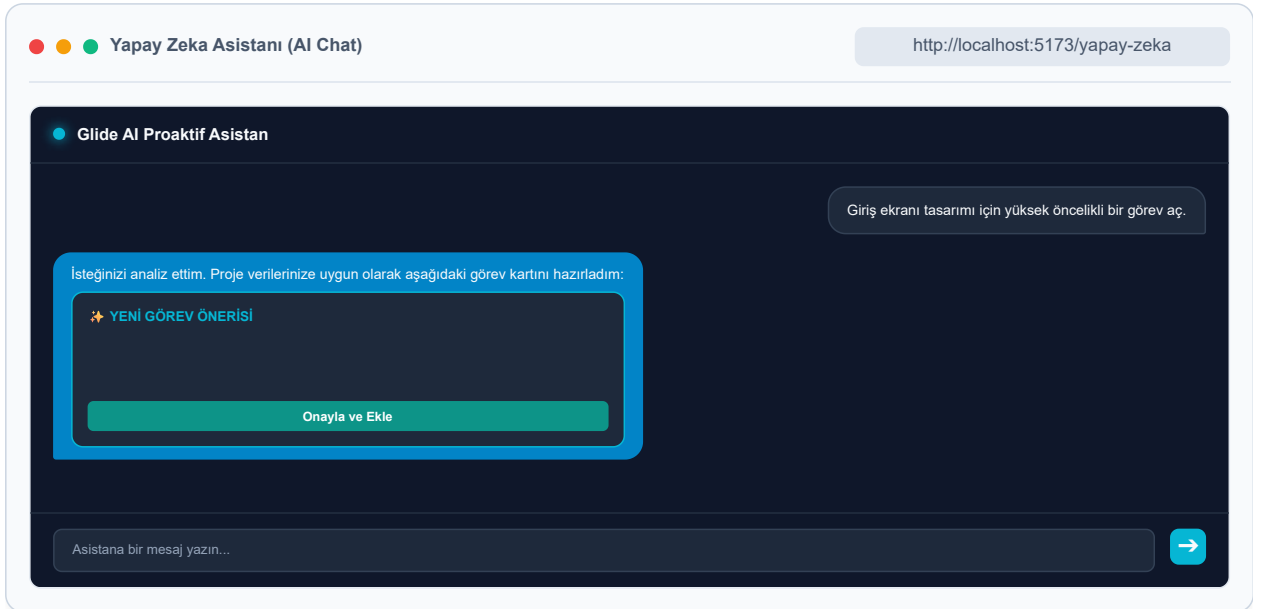
3.5.5. Akıllı Yapay Zeka Asistanı (AI Assistant)

Akıllı Yapay Zeka Asistanı (`AIAssistantView.jsx`), Glide platformunun en belirgin ve yenilikçi özelliğidir. Klasik proje yönetim araçlarının sunduğu basit arama motorlarının aksine, sisteme gömülü ve bağlam duyarlı (Context-Aware) bir co-pilot olarak çalışır. Sayfa açıldığında veya proje seçildiğinde, mevcut projenin iş listeleri, sprint tarihleri, ekip kapasitesi ve geciken riskli işler toplanarak sistem talimatlarına (Prompt Injection) eklenir.

Doğal Dilden Eyleme Dönüşüm: Kullanıcı asistan arayüzüne doğal dilde bir komut girdiğinde (örn: *"Giriş ekranı tasarımı için yüksek öncelikli bir görev aç"*), OpenRouter üzerinden çalışan model bu girdiyi işler. İstek, görev oluşturma niyetindeyse yapay zeka doğrudan metin cevabı yerine yapılandırılmış bir JSON şeması (`CREATE\_TASK` eylemi) döndürür:

```
{ "type": "CREATE_TASK", "title": "Giriş Ekranı Tasarımı", "description": "Giriş ekranı mockup ve React 19 kodlaması.", "priority": "HIGH" }
```

Arayüz bu JSON çıktısını yakalar ve kullanıcıya özel bir **"Yeni Görev Önerisi"** kartı çizer. Kullanıcı "Onayla ve Ekle" butonuna bastığı an Redux action tetiklenerek görev veritabanına ve Kanban panosuna milisaniyeler içinde kaydedilir.



Şekil 3.7: Doğal Dil İşleme Destekli Yapay Zeka Asistan Sohbet Arayüzü

3.5.6. Döngüler ve Sprint Planlama (Cycles)

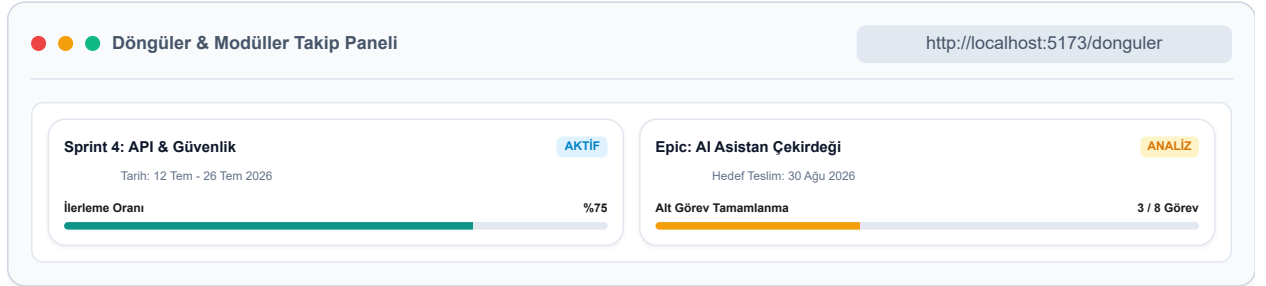
Glide, çevik (Agile/Scrum) yazılım metodolojilerini desteklemek amacıyla gelişmiş bir Döngü (sprint/cycle) yönetim arayüzü barındırır. Her projenin altında bağımsız döngüler tanımlanabilir. Döngüler; isim, başlangıç tarihi, bitiş tarihi ve durum (Yaklaşan, Aktif, Tamamlandı) özniteliklerine sahiptir.

Sprint İzleme ve İlerleme: Aktif olan döngü, projedeki görevlerin tamamlanma yüzdelerini ve yakılan iş (burndown) eğilimini otomatik hesaplar. `CyclesView.jsx` sayfasında, aktif döngüye ait toplam görev sayısı, tamamlanan işlerin oranı ve bitişe kalan gün sayısı dinamik grafiklerle görselleştirilir. Geliştiriciler ve PM'ler, görev oluştururken veya düzenlerken ilgili iş kartını döngüye atayarak sprint kapsamını belirler. Süresi biten döngüler tamamlandı durumuna alınarak arşivlenir ve bitmeyen görevler backlog'a veya bir sonraki döngüye aktarılır.

3.5.7. Modül ve Epik Takibi (Modules)

Büyük ölçekli projelerin uzun vadeli stratejik hedeflerini mantıksal bölümlere ayırmak için Modüller (Epics/Features) kullanılır. Örneğin bir e-ticaret projesindeki "Ödeme Geçidi", "Üyelik Sistemleri" veya "Arama Algoritması" gibi büyük iş kırılımları birer modüldür.

Modüler İlerleme Takibi: `ModulesView.jsx` ve `ModuleTimeline.jsx` bileşenleri, her modülün altındaki görevlerin tamamlanma durumunu (done/total) analiz ederek modülün gerçek zamanlı tamamlanma oranını çıkarır. Görsel arayüzde, modüllerin başlangıç ve bitiş tarihleri Gantt benzeri bir zaman çizelgesinde (Timeline) listelenir. Böylece yöneticiler, makro düzeyde hangi özelliğin geciktiğini ve hangi ekiplerin hangi modüllere odaklandığını anında tespit edebilirler.



Şekil 3.8: Çevik Döngüler ve Proje Epik Modülleri Durum Kartları

3.5.8. Talep Toplama ve Triage Yönetimi (Intake)

Glide proje yönetim platformunu benzerlerinden ayıran en önemli mimari özelliklerden biri, dış paydaşların (müşteriler, destek ekibi) taleplerini toplayan bağımsız bir **Triage (Talep Eleme) Gelen Kutusu** (`RequestsView.jsx`) katmanıdır. Geleneksel sistemlerde harici talepler doğrudan geliştirici backlog'una düşer ve planlama kirliliğine neden olur. Glide, bu süreci izole bir ara katmanla çözer.

Triage İş Akışı (Triage Workflow): Müşteriler veya Misafir (Guest) rolündeki kullanıcılar basitleştirilmiş bir talep formu doldurarak sisteme bilet (ticket) ekler. Bu biletler, ``intakeRequests`` veri tablosuna düşer. Proje yöneticileri ve yöneticiler, gelen kutusunda bu talepleri inceler. Her talep için üç seçenek bulunur:

- **Approve (Onayla):** Talep onaylandığında yöneticiye bir popup açılır. Yönetici, bu talebin hangi projeye dahil edileceğini, önceliğini ve atanan kişiyi seçer. Sistem arka planda bu talebi ``/intakeRequests`` tablosundan siler ve ilişkisel verileriyle doğrudan ``/workItems`` (görevler) tablosuna yeni bir Kanban kartı olarak aktarır.
- **Decline (Reddet):** Talep reddedilir ve gerekçesi sisteme işlenerek durum "Reddedildi" olarak etiketlenir.
- **Archive (Arşivle):** Talep, aktif işleri kirliletmemek adına arşiv tablosuna taşınır.

Bu akış, iç ekip ile dış müşteriler arasında güvenli ve kontrollü bir iş köprüsü kurarak planlama bütünlüğünü korur.

Talep Eleme ve Triage Yönetim Paneli

http://localhost:5173/talepler

Gelen Triage Talepleri (Triage Inbox)

Talep Sahibi	Konu Başlığı	Durum	İşlemler
Aslınur (Müşteri)	Ödeme adımında donma hatası alıyorum	BEKLEMEDE	<button>Onayla</button> <button>Reddet</button>

Şekil 3.9: Gelen Dış Paydaş Talepleri ve Eleme (Triage) Karar Mekanizması

3.5.9. Kullanıcı Yönetimi, Rol Yetki Matrisi (Users & Roles)

Glide platformunun güvenliği ve rol hassasiyeti, frontend katmanında kurgulanan **Rol Tabanlı Erişim Denetimi (RBAC)** mekanizmasıyla yönetilir. Sistemde tanımlı 4 temel rol (Admin, PM, Member, Guest) bulunmaktadır. `ProtectedRoute.jsx` bileşeni rotaları sarmalayarak yetkisiz girişleri engellerken, görünümlerdeki bileşenler de `activeRole` durumuna göre CRUD düğmelerini ve menüleri dinamik olarak gizler veya pasifleştirir.

Rollerin Genel Sınırları ve Davranışları:

- Yönetici (Admin):** Sistemdeki en üst düzey yetkili hesaptır. Workspace oluşturma/düzenleme, tüm projeleri ve görevleri yönetme, otomasyon kuralları tanımlama, Triage kutusundaki talepleri onaylayıp silme ve sistemdeki diğer kullanıcıların yetki derecelerini değiştirme hakkına sahiptir.
- Proje Yöneticisi (PM):** Sadece sahibi veya üyesi olduğu projelerde tam yetkiye sahiptir. Görevler (Kanban), Döngüler (Sprints) ve Modüller (Epics) üzerinde ekleme, silme ve düzenleme yapabilir. Otomasyon kuralları oluşturabilir. Workspace genel ayarlarını değiştiremez.
- Ekip Üyesi (Member):** Atandığı projelerdeki görevleri görüntüleyebilir. Sadece kendi oluşturduğu veya üzerine atanmış iş öğelerini güncelleyebilir ve bunlara yorum yazabilir. Yeni proje oluşturma, sprint tanımlama, otomasyon kuralı yazma veya triage taleplerini yönetme yetkisi bulunmaz.
- Müşteri / Misafir (Guest):** En kısıtlı roldür. Sadece projenin bitmiş veya aktif durumlarını izleyebilir (Read-only). Triage formunu kullanarak yeni bir dış talep (Intake Request) oluşturabilir ve yetki verilen iş öğelerine yorum yazarak geribildirim iletebilir.

Tablo 3.1: Glide Rol-Yetki Matrisi (RBAC Matrix)

Yetki Alanı / Fonksiyon	Yönetici (Admin)	Proje Yöneticisi (PM)	Ekip Üyesi (Member)	Müşteri / Misafir (Guest)
Proje / Workspace CRUD	✓ Evet	⚠ Sınırlı (Sadece Kendi)	✗ Hayır	✗ Hayır
Döngü (Sprint) & Modül Yönetimi	✓ Evet	✓ Evet	✗ Hayır	✗ Hayır
Görev (Kanban) CRUD	✓ Evet	✓ Evet	⚠ Sınırlı (Atanan/Açan)	✗ Hayır (Read-only)
Otomasyon Kuralı Tanımlama	✓ Evet	✓ Evet	✗ Hayır	✗ Hayır
Triage Talep Eleme (Approve/Reject)	✓ Evet	✓ Evet	✗ Hayır	✗ Hayır
Talep / Bilet Oluşturma (Intake)	✓ Evet	✓ Evet	✓ Evet	✓ Evet (Yegâne Yetki)

4. SONUÇ

Bu bitirme projesi kapsamında, modern yazılım ekiplerinin proje süreçlerini verimli yönetmesini amaçlayan yapay zeka ve no-code otomasyon destekli "Glide" proje yönetim sistemi başarıyla geliştirilmiştir. Geliştirilen sistem, Vite derleme aracı ve React kütüphanesinin sağladığı anlık sayfa güncellemeleri sayesinde yüksek performanslı bir kullanıcı deneyimi sunmaktadır.

Proje kapsamında kurgulanan no-code otomasyon yapısı, ekiplerin el yordamıyla yaptığı durum güncellemelerini ve atamaları otomatikleştirerek süreç içi hata oranlarını minimize etmiştir. AI asistanının proaktif görev önerme ve risk analizi yetenekleri, proje yöneticilerinin idari iş yükünü azaltmada yenilikçi bir yaklaşım sergilemiştir. Gelecek geliştirmelerde, JSON Server mock altyapısının yerini mikroservis mimarisinde çalışan gerçek bir backend servisine ve WebSocket tabanlı anlık veri senkronizasyon mekanizmalarına bırakması hedeflenmektedir.

5. KAYNAKÇA

[1] React Resmi Dökümantasyonu: <https://react.dev/blog/2024/12/05/react-19> (Erişim Tarihi: Temmuz 2026).

[2] Vite Derleme Aracı Kılavuzu: <https://vite.dev/guide/> (Erişim Tarihi: Temmuz 2026).

[3] Redux Toolkit Geliştirici Kılavuzu: <https://redux-toolkit.js.org/introduction/getting-started> (Erişim Tarihi: Temmuz 2026).

[4] Tailwind CSS v4 Tasarım Sistemi Belgelendirmesi: <https://tailwindcss.com/docs/v4-beta> (Erişim Tarihi: Temmuz 2026).

[5] OpenRouter API Referans Dökümantasyonu: <https://openrouter.ai/docs> (Erişim Tarihi: Temmuz 2026).

[6] MDN Web Docs - Using the Fetch API: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch (Erişim Tarihi: Temmuz 2026).