**Flare Administration UI - Tasarım Dokümanı**

**📌 Amaç**

Flare Administration UI (Flare Admin), kullanıcıların Flare platformunda proje tanımlayıp yönetebileceği bir web arayüzüdür. Angular ile geliştirilir, backend olarak Python ile geliştirilmiş olan Flare backend'i kullanır.

Flare Admin ekranlarında; proje tanımlama, mevcut projeyi düzenleme, projeye versiyon çıkarma, versiyonu publish etme, projeyi enable-disable etme, proje tanımını silme gibi işlemler yapılabilir.

Document database olarak Flare'in service\_config.jsonc dosyasını kullanır. Tüm tanım, düzenleme, silme ve listeleme işlemlerini bu dosya üzerinde yapar.

Flare Admin, Flare backend servisi üzerinden Spark ile entegre çalışır.

**Genel Kurallar**

Flare Admin, login ekranıyla başlar. Kullanıcı login olduktan sonra tanım ekranları görüntülenir.

Tanım ekranları, tab window'lar şeklinde gruplandırılmıştır. Her tab'da birbiriyle alakalı işlemler yapılır.

Tüm hatalı işlemler, işlemin yapıldığı ekranın alt kısmında kırmızı font ile uyarı şeklinde görüntülenir ve uyarı 5 sn sonra kaybolur.

Boolean alanlar ekranda (hem input olarak, hem de tablo üzerinde) checkbox olarak görüntülenir.

Silme işlemleri öncesinde, silinecek nesnenin diğer nesneler altında kullanımda olup olmadığı kontrol edilir. Eğer kullanımdaysa, silinmesine izin verilmez ve hangi nesneler tarafından kullanıldığını içeren bir mesaj ve hata kodu ile döndürülür. Kullanımda olmayan nesneyi silme öncesinde açılacak bir dialog'da kullanıcıdan onay alınır. Onayın ardından silme işlemi yapılır.

Nesnenin **deleted** alanı varsa silme işlemi fiziksel olarak yapılmaz, deleted=true olarak değiştirilir.

Ortam çok kullanıcılı olduğundan kayıt düzenleme işlemlerinde race condition kontrolü yapılır. Bunun için service\_config.jsonc içinde config, project, version ve api nesnelerine **last\_update\_date** alanı eklenir. Kullanıcı, düzenlemek için nesneyi ekrana yüklediğinde nesne içindeki bu değeri de yükler. Save butonuna bastığında backend'e bu değeri de gönderir ve backend kayıt öncesi bu değerin service\_config.jsonc'deki ilgili nesnedeki değerle aynı olup olmadığını kontrol eder, aynı ise kaydı yapar ve kaydederken bu değeri o andaki tarih-saat-dakika-saniye-milisaniye değeri ile günceller. Eğer bir başka kullanıcı aynı kaydı düzenliyorsa ve ilk kullanıcıdan sonra save'e bastıysa dosyadaki bu değer değişmiş olacağından backend kayda izin vermez ve "Kayıt başka kullanıcı tarafından düzenlendi. Lütfen tekrar yükleyin." tarzı bir uyarı ile birlikte bir hata kodu döndürür. Bir diğer durum da, kullanıcı tarafından düzenlenen kaydın, bu kullanıcı kaydetmeden önce başka bir kullanıcı tarafından silinmiş olması durumudur. Backend kayıt öncesi dosyada bu nesneyi bulamazsa "Kayıt başka kullanıcı tarafından silindi." gibi bir uyarı ve bir hata kodu döndürür, kayda izin vermez.

**Auto-save ve Undo/Redo Mekanizması**

* Published olmayan version'lar üzerinde çalışırken her 30 saniyede bir otomatik draft kayıt yapılır
* Son 10 işlem için undo/redo desteği (Ctrl+Z / Ctrl+Y)
* Draft kayıtlar drafts/ klasöründe tutulur ve UI açıldığında "Recover unsaved changes?" dialog'u gösterilir

**Klavye Kısayolları**

* Ctrl+S: Save
* Ctrl+N: New (context'e göre project/version/intent)
* Ctrl+F: Search/Filter
* Ctrl+D: Duplicate
* Esc: Cancel/Close dialog

**Flare Backend Düzenlemeleri**

Flare backend'de Flare Admin UI işleyişi için bazı eklemeler yapılması gerekiyor.

**Eklenecek endpoint'ler:**

**Temel CRUD Endpoint'leri**

* POST /api/login - Kullanıcı girişi ve token üretimi
* GET /api/environment - Environment bilgileri alma
* PUT /api/environment - Environment bilgilerini kaydetme
* GET /api/projects - Proje listeleme (pagination, filter, sort destekli)
* POST /api/projects - Yeni proje oluşturma
* PUT /api/projects/{id} - Mevcut projeyi düzenleme
* DELETE /api/projects/{id} - Mevcut projeyi silme
* PATCH /api/projects/{id}/toggle - Proje enable/disable
* POST /api/projects/{id}/versions - Proje versiyonu oluşturma
* DELETE /api/projects/{id}/versions/{version\_id} - Versiyon silme
* PUT /api/projects/{id}/versions/{version\_id} - Versiyon düzenleme
* POST /api/projects/{id}/versions/{version\_id}/publish - Versiyon yayınlama
* GET /api/apis - API listeleme
* POST /api/apis - Yeni API ekleme
* PUT /api/apis/{name} - Mevcut API'yi düzenleme
* DELETE /api/apis/{name} - Mevcut API'yi silme

**Ek Yardımcı Endpoint'ler**

* POST /api/validate/regex - Regex pattern validasyonu
* POST /api/apis/test - API endpoint test etme
* GET /api/projects/{id}/versions/diff - Version karşılaştırma
* POST /api/projects/{id}/intents/bulk-delete - Toplu intent silme
* GET /api/activity-log - Aktivite logları
* POST /api/projects/import - Proje import
* GET /api/projects/{id}/export - Proje export
* POST /api/test/run-all - Tüm testleri çalıştır
* GET /api/test/status - Test durumu sorgula

**Service\_config.jsonc düzenlemeleri:**

{

 "config": {

 // Mevcut alanlar...

 "project\_id\_counter": 1, // Yeni

 "last\_update\_date": "2025-06-08T12:00:00.000Z", // Yeni

 "last\_update\_user": "admin" // Yeni

 },

 "projects": [{

 // Mevcut alanlar...

 "version\_id\_counter": 1, // Yeni

 "last\_update\_date": "2025-06-08T12:00:00.000Z", // Yeni

 "last\_update\_user": "admin", // Yeni

 "deleted": false, // Yeni

 "created\_date": "2025-06-08T12:00:00.000Z", // Yeni

 "created\_by": "admin", // Yeni

 "versions": [{

 // Mevcut alanlar...

 "last\_update\_date": "2025-06-08T12:00:00.000Z", // Yeni

 "last\_update\_user": "admin", // Yeni

 "deleted": false, // Yeni

 "created\_date": "2025-06-08T12:00:00.000Z", // Yeni

 "created\_by": "admin", // Yeni

 "publish\_date": null, // Yeni

 "published\_by": null // Yeni

 }]

 }],

 "apis": [{

 // Mevcut alanlar...

 "last\_update\_date": "2025-06-08T12:00:00.000Z", // Yeni

 "last\_update\_user": "admin", // Yeni

 "deleted": false, // Yeni

 "created\_date": "2025-06-08T12:00:00.000Z", // Yeni

 "created\_by": "admin" // Yeni

 }],

 // Yeni: Activity log

 "activity\_log": [{

 "id": 1,

 "timestamp": "2025-06-08T12:00:00.000Z",

 "user": "admin",

 "action": "CREATE\_PROJECT",

 "entity\_type": "project",

 "entity\_id": 1,

 "entity\_name": "airline\_agent",

 "details": "Created new project"

 }]

}

**Tab Ekranları**

**1. User Info Tab'ı**

Bu tab penceresinde login olan kullanıcı mevcut şifresini değiştirebilir. Ekrandaki alanlar:

* **Current Password** (maskeli)
* **New Password** (maskeli)
* **New Password** (tekrar, maskeli)
* **Password Strength Indicator**: Şifre girilirken güvenlik seviyesini gösterir (Weak/Medium/Strong)
* **Save** butonu

Şifre değiştirebilmek için kullanıcı, maskeli alanlardan mevcut şifresini, yeni şifresini ve ikinci bir alanda teyit amaçlı olarak tekrar yeni şifresini girer ve kaydet butonuna basar. Şifre en az 8 karakter, büyük/küçük harf ve rakam içermelidir.

**2. Environment Tab'ı**

Bu tab'da aşağıdaki alanlar bulunur:

* **Work Mode**: Seçenekleri hfcloud, cloud ve on-premise olan bir combo. Zorunlu.
* **Cloud Token** (maskeli): Work Mode on-premise seçildiğinde disable olur.
* **Spark Endpoint**: URL formatında. Zorunlu. Yanında "Test Connection" butonu.
* **Save** butonu
* **Reload from Spark** butonu: Spark'tan güncel config'i çeker

**3. APIs Tab'ı**

Tab açıldığında, üstte bir toolbar bulunur:

**Toolbar:**

* **New API** butonu
* **Import** butonu: JSON dosyasından API tanımları yükler
* **Export** butonu: Seçili API'leri JSON olarak export eder
* **Search** alanı: API adı veya URL'de arama
* **Display Deleted** checkbox

**API Listesi Tablosu:**

| **Name** | **URL** | **Method** | **Timeout** | **Auth** | **Deleted** | **Actions** |
| --- | --- | --- | --- | --- | --- | --- |
| book\_flight\_api | https://api.example.com/book | POST | 10s | ✓ | ✗ | 🖊️ 🧪 📋 🗑️ |

**Action İkonları:**

* 🖊️ Edit: Edit API dialog'unu açar
* 🧪 Test: API'yi test eder
* 📋 Duplicate: API'nin kopyasını oluşturur
* 🗑️ Delete: API'yi siler (kullanımda değilse)

**4. Projects Tab'ı**

Tab ekranına girildiğinde üstte toolbar bulunur:

**Toolbar:**

* **New Project** butonu
* **Import Project** butonu
* **Search** alanı
* **Display Deleted** checkbox
* **View**: List/Card view toggle

**Proje Listesi:** Kart görünümünde her proje için:

┌─────────────────────────────────┐

│ 🛩️ airline\_agent │

│ Havayolu Müşteri Temsilcisi │

│ Versions: 3 (1 published) │

│ Status: ✓ Enabled │

│ Last update: 2 hours ago │

│ [Edit] [Versions] [Export] [⚙️] │

└─────────────────────────────────┘

**5. Test Tab'ı**

Bu tab'da sistemin end-to-end testi yapılabilir:

**Test Kategorileri:**

* **UI Tests**: Component render, navigation, form validation
* **Backend Tests**: API endpoint'leri, data integrity
* **Integration Tests**: UI-Backend entegrasyonu
* **Spark Tests**: LLM bağlantısı ve response kalitesi

**Ekran Düzeni:**

┌─────────────────────────────────────────────┐

│ [Run All Tests] [Run Selected] [Stop] │

├─────────────────────────────────────────────┤

│ □ All Tests │

│ □ UI Tests (15 tests) │

│ □ Login Flow │

│ □ Project CRUD │

│ □ Version Management │

│ □ Backend Tests (22 tests) │

│ □ Authentication │

│ □ API Endpoints │

│ □ Race Conditions │

│ □ Integration Tests (18 tests) │

│ □ Spark Tests (8 tests) │

├─────────────────────────────────────────────┤

│ Test Results: │

│ ✓ Login with valid credentials - 120ms │

│ ✓ Create new project - 340ms │

│ ✗ Delete API in use - Expected behavior │

│ ⏳ Testing Spark connection... │

│ │

│ Progress: ████████░░ 80% (48/60) │

│ Passed: 45 | Failed: 3 | Running: 5 │

└─────────────────────────────────────────────┤

**Test Detayları:** Her test için:

* Test adı ve açıklaması
* Çalışma süresi
* Hata durumunda detaylı log
* "Re-run" butonu

**6. Activity Log (Sağ üst notification icon'a tıklandığında açılır)**

┌─────────────────────────────────────┐

│ 🔔 Recent Activities │

├─────────────────────────────────────┤

│ 2 min ago │

│ 👤 admin published version 2 │

│ of airline\_agent │

├─────────────────────────────────────┤

│ 15 min ago │

│ 👤 user1 updated flight\_info\_api │

├─────────────────────────────────────┤

│ 1 hour ago │

│ 👤 admin created new project │

│ hotel\_booking │

├─────────────────────────────────────┤

│ [View All Activities] │

└─────────────────────────────────────┘

**Dialog Ekranları**

**Edit API Dialog**

┌─ Edit API ─────────────────────────────────┐

│ Tabs: [General] [Headers] [Auth] [Test] │

├────────────────────────────────────────────┤

│ General Tab: │

│ Name\*: [flight\_booking\_api ] │

│ URL\*: [https://api.example.com/book] │

│ Method\*: [POST ▼] │

│ │

│ Body Template: │

│ ┌─────────────────────────────────────┐ │

│ │{ │ │

│ │ "origin": "{{variables.origin}}", │ │

│ │ "destination": "{{variables.dest}}" │ │

│ │} │ │

│ └─────────────────────────────────────┘ │

│ [Template Variables ▼] [Validate JSON] │

│ │

│ Timeout\*: [10] seconds │

│ │

│ Retry Settings: │

│ Count\*: [3] | Backoff\*: [2]s │

│ Strategy\*: [Static ▼] │

│ │

│ Response Prompt: │

│ ┌─────────────────────────────────────┐ │

│ │ Kullanıcıya dön... │ │

│ └─────────────────────────────────────┘ │

├────────────────────────────────────────────┤

│ [Test API] [Save] [Cancel] │

└────────────────────────────────────────────┘

**Headers Tab:**

Key: [Content-Type ] Value: [application/json] [Add]

Current Headers:

┌────────────────┬─────────────────────┬───┐

│ Key │ Value │ │

├────────────────┼─────────────────────┼───┤

│ Authorization │ Bearer {{auth...}} │ 🗑 │

│ Content-Type │ application/json │ 🗑 │

└────────────────┴─────────────────────┴───┘

**Auth Tab:**

☑ Enable Authentication

Token Endpoint\*: [https://api.example.com/auth]

Response Token Path\*: [access\_token ]

 (e.g., data.token or token)

Token Request Body:

┌─────────────────────────────────────┐

│ { │

│ "grant\_type": "client\_cred..." │

│ } │

└─────────────────────────────────────┘

☐ Enable Token Refresh

Token Refresh Endpoint: [\_\_\_\_\_\_\_\_\_\_\_\_]

Token Refresh Body: [\_\_\_\_\_\_\_\_\_\_\_\_]

**Test Tab:**

Test Request:

┌─────────────────────────────────────┐

│ { │

│ "origin": "Istanbul", │

│ "destination": "Ankara" │

│ } │

└─────────────────────────────────────┘

[Send Test Request]

Response:

Status: 200 OK | Time: 342ms

┌─────────────────────────────────────┐

│ { │

│ "booking\_id": "XYZ123", │

│ "status": "confirmed" │

│ } │

└─────────────────────────────────────┘

**Edit Project Dialog**

┌─ Edit Project ─────────────────────────┐

│ Name\*: [airline\_agent ] │

│ Caption: [Havayolu Müşteri Tem...] │

│ │

│ Created: 2025-06-01 by admin │

│ Modified: 2025-06-08 by user1 │

├───────────────────────────────────────┤

│ [Save] [Cancel] │

└───────────────────────────────────────┘

**Select Version Dialog**

┌─ Select Version ───────────────────────┐

│ Project: airline\_agent │

├────────────────────────────────────────┤

│ ○ Version 3 - Latest features │

│ Published: No | Modified: Today │

│ │

│ ○ Version 2 - Stable release │

│ Published: Yes | Date: 2025-06-01 │

│ │

│ ○ Version 1 - Initial version │

│ Published: Yes | Date: 2025-05-15 │

├────────────────────────────────────────┤

│ [Select] [Compare] [Cancel] │

└────────────────────────────────────────┘

**Edit Version Dialog**

┌─ Edit Version ──────────────────────────────────────┐

│ Tabs: [General] [Prompt] [LLM] [Intents] [Preview] │

├─────────────────────────────────────────────────────┤

│ General Tab: │

│ Version Number: 3 (locked) │

│ Caption\*: [Latest features with date support] │

│ Published: No (locked) │

│ │

│ ⚠️ Unpublished version - changes auto-saved │

├─────────────────────────────────────────────────────┤

│ [Save] [Publish] [Cancel] │

└─────────────────────────────────────────────────────┘

**Prompt Tab:**

General System Prompt\*:

┌─────────────────────────────────────────────┐

│ ⚠️ \*\*NEVER output "#DETECTED\_INTENT:" │

│ unless the user's request CLEARLY... │

│ │

│ You are a friendly, empathetic... │

└─────────────────────────────────────────────┘

[Prompt Variables ▼] [Validate] [Preview]

Template Variables:

• {{user\_input}} - Current user message

• {{chat\_history}} - Conversation context

• {{intents}} - Available intents

**LLM Tab:**

Model Configuration:

Repo ID\*: [ytu-ce-cosmos/Turkish-Llama-8b]

 [Browse Models]

Generation Config:

Max New Tokens\*: [256] (128-512)

Temperature\*: [0.2] (0.0-1.0)

Top P\*: [0.8] (0.0-1.0)

Repetition Penalty\*: [1.1] (1.0-2.0)

☐ Use Fine-Tuned Model

Fine Tune ZIP: [\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_] [Browse]

**Intents Tab:**

[New Intent] [Import] [Export]

Search: [\_\_\_\_\_\_\_\_\_\_\_\_\_🔍]

┌──────────────┬─────────────────────┬────────┬───┐

│ Name │ Caption │ Params │ │

├──────────────┼─────────────────────┼────────┼───┤

│ flight-book │ Uçak bileti rezrv. │ 4 │ 🖊🗑│

│ flight-info │ Uçuş durum bilgisi │ 2 │ 🖊🗑│

│ booking-cancel│ Rezervasyon iptali │ 2 │ 🖊🗑│

└──────────────┴─────────────────────┴────────┴───┘

**Preview Tab:**

Test Intent Detection:

User Message: [Yarın Ankara'ya gitmek istiyorum]

[Test]

Result:

✓ Intent Detected: flight-booking

Confidence: 0.92

Parameters that would be extracted:

• destination: "Ankara" ✓

• flight\_date: "2025-06-09" ✓

• origin: (missing)

• passenger\_count: (missing)

Next prompt would be:

"Kalkış şehri bilgisini alabilir miyim?"

**Edit Intent Dialog**

┌─ Edit Intent ────────────────────────────────────┐

│ Tabs: [General] [Parameters] [Examples] [Test] │

├──────────────────────────────────────────────────┤

│ General Tab: │

│ Name\*: [flight-booking ] │

│ Caption: [Uçak bileti rezervasyonu] │

│ Locale\*: [tr-TR ▼] │

│ │

│ Detection Prompt\*: │

│ ┌───────────────────────────────────────────┐ │

│ │ User wants to book, reserve, or buy │ │

│ │ a flight ticket with specific cities... │ │

│ └───────────────────────────────────────────┘ │

│ │

│ Action\*: [book\_flight\_api ▼] │

│ │

│ Fallback Prompts: │

│ Timeout: [Rezervasyon sistemine...] │

│ Error: [Rezervasyon işlemi sırasında...] │

├──────────────────────────────────────────────────┤

│ [Save] [Cancel] │

└──────────────────────────────────────────────────┘

**Parameters Tab:**

[Add Parameter] [Import] [Reorder]

┌────────┬──────────────┬──────┬─────────┬───┐

│ Name │ Caption │ Type │ Required│ │

├────────┼──────────────┼──────┼─────────┼───┤

│ origin │ Kalkış şehri │ str │ ✓ │ 🖊🗑│

│ dest │ Varış şehri │ str │ ✓ │ 🖊🗑│

│ date │ Uçuş tarihi │ date │ ✓ │ 🖊🗑│

│ count │ Yolcu sayısı │ int │ ✓ │ 🖊🗑│

└────────┴──────────────┴──────┴─────────┴───┘

Drag parameters to reorder extraction priority

**Examples Tab:**

Add Example: [İstanbuldan Ankaraya bilet] [Add]

Current Examples:

┌─────────────────────────────────────────┬───┐

│ 10 Temmuz'da İstanbul'dan Ankara'ya... │ 🗑 │

│ Yarın İzmir'den Berlin'e sabah uçuşu │ 🗑 │

│ Antalya'ya iki kişilik bilet lazım │ 🗑 │

│ 15 Temmuz'da İstanbul Londra bileti │ 🗑 │

└─────────────────────────────────────────┴───┘

[Generate More Examples] (AI-powered)

**Edit Parameter Dialog**

┌─ Edit Parameter ─────────────────────────────┐

│ Basic Settings: │

│ Name\*: [flight\_date ] │

│ Caption\*: [Uçuş tarihi ] │

│ Type\*: [date ▼] │

│ ☑ Required │

│ Variable Name\*: [flight\_date ] │

│ │

│ Extraction Prompt\*: │

│ ┌─────────────────────────────────────────┐ │

│ │ Extract the flight date from Turkish │ │

│ │ text and convert to ISO format... │ │

│ └─────────────────────────────────────────┘ │

│ │

│ Validation: │

│ Regex Pattern: [\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_] │

│ [Common Patterns ▼] │

│ • Email: ^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+$│

│ • Phone: ^(\+90|0)?5\d{9}$ │

│ • Date: ^\d{4}-\d{2}-\d{2}$ │

│ • PNR: ^[A-Z0-9]{5}$ │

│ │

│ Error Messages: │

│ Invalid: [Geçersiz tarih formatı...] │

│ Type Error: [Tarih bir metin olmalı...] │

├─────────────────────────────────────────────┤

│ [Test Extraction] [Save] [Cancel] │

└─────────────────────────────────────────────┘

**Version Comparison Dialog**

┌─ Compare Versions ──────────────────────────┐

│ Version 2 vs Version 3 │

├─────────────────────────────────────────────┤

│ General Changes: │

│ • Caption: "Stable release" → "Latest..." │

│ │

│ Intent Changes: │

│ + Added: flight-status intent │

│ - Removed: (none) │

│ ~ Modified: flight-booking │

│ - Added parameter: seat\_class │

│ - Updated detection prompt │

│ │

│ LLM Changes: │

│ • Temperature: 0.3 → 0.2 │

│ • Max tokens: 128 → 256 │

├─────────────────────────────────────────────┤

│ [Close] │

└─────────────────────────────────────────────┘

**Flare Admin Test Planı**

**1. Birim Testler (Unit Tests)**

**Login Modülü:**

* ✓ Doğru kullanıcı adı ve şifre ile başarılı giriş
* ✓ Yanlış kullanıcı adı ile başarısız giriş
* ✓ Yanlış şifre ile başarısız giriş
* ✓ Boş alanlarla giriş denemesi
* ✓ Token expire durumu kontrolü
* ✓ Logout sonrası token temizleme

**Environment Modülü:**

* ✓ Work mode değişiminde cloud token alanı enable/disable
* ✓ Invalid URL formatı validasyonu
* ✓ Boş zorunlu alan kontrolü
* ✓ Başarılı kayıt sonrası Spark notification

**API Modülü:**

* ✓ Yeni API ekleme - tüm alanlar dolu
* ✓ Yeni API ekleme - sadece zorunlu alanlar
* ✓ Duplicate API name kontrolü
* ✓ URL format validasyonu
* ✓ Headers tablosuna ekleme/silme
* ✓ Auth enabled/disabled durumu
* ✓ API silme - kullanımda olmayan
* ✓ API silme - intent'lerde kullanılan
* ✓ API test butonu işlevselliği

**Project Modülü:**

* ✓ Yeni proje oluşturma
* ✓ Duplicate project name kontrolü
* ✓ Proje silme - version'ı olmayan
* ✓ Proje silme - version'ları olan
* ✓ Enable/disable toggle
* ✓ Version oluşturma - ilk version
* ✓ Version oluşturma - mevcut version'dan kopyalama
* ✓ Published version üzerinde edit engeli
* ✓ Unpublished version silme
* ✓ Version publish işlemi

**Intent Modülü:**

* ✓ Yeni intent ekleme
* ✓ Intent parametre ekleme/silme
* ✓ Example ekleme/silme
* ✓ Action combo - sadece tanımlı API'ler
* ✓ Intent silme

**Parameter Modülü:**

* ✓ Parameter type validasyonu
* ✓ Required checkbox işlevselliği
* ✓ Regex pattern validasyonu
* ✓ Variable name uniqueness

**2. Entegrasyon Testleri (Integration Tests)**

**Multi-user Senaryoları:**

* ✓ İki kullanıcı aynı projeyi edit - race condition
* ✓ Kullanıcı A edit ederken, kullanıcı B silme
* ✓ Last update date/user bilgisi güncelleme
* ✓ Concurrent version publish attempts

**Data Consistency:**

* ✓ Project silme - tüm version'lar da siliniyor mu?
* ✓ API silme - kullanan intent'ler kontrol ediliyor mu?
* ✓ Version publish - diğer version'lar unpublish oluyor mu?
* ✓ Auto-save draft recovery

**Import/Export:**

* ✓ Project export - tüm bağımlılıklar dahil mi?
* ✓ Project import - ID çakışmaları handle ediliyor mu?
* ✓ API bulk import - validation çalışıyor mu?

**3. UI/UX Testleri**

**Responsive Design:**

* ✓ 1920x1080 desktop görünüm
* ✓ 1366x768 laptop görünüm
* ✓ Tablet görünüm (opsiyonel)
* ✓ Dialog açıkken resize

**Keyboard Navigation:**

* ✓ Tab order mantıklı mı?
* ✓ Enter/Esc tuşları çalışıyor mu?
* ✓ Kısayollar (Ctrl+S, Ctrl+N vb.)

**Error Handling:**

* ✓ Backend timeout - kullanıcıya bilgi
* ✓ Network hatası - retry seçeneği
* ✓ Validation hataları - field highlight
* ✓ Başarı mesajları - 3 saniye görünür

**4. Performans Testleri**

**Load Times:**

* ✓ İlk yükleme < 3 saniye
* ✓ Tab geçişleri < 500ms
* ✓ Dialog açılma < 300ms
* ✓ Auto-complete < 200ms

**Data Volume:**

* ✓ 100+ proje listesi pagination
* ✓ 50+ intent'li version performansı
* ✓ 1000+ satırlık activity log

**5. Güvenlik Testleri**

**Authentication:**

* ✓ Token header her request'te
* ✓ Invalid token = 401 response
* ✓ Token expiry handling
* ✓ XSS prevention in inputs

**Authorization:**

* ✓ Sadece kendi user bilgilerini görebilme
* ✓ Admin olmayan config değiştiremez
* ✓ SQL injection attempts

**6. End-to-End Test Senaryoları**

**Senaryo 1: Yeni Proje Oluşturma**

1. Login yap
2. Projects tab'ına git
3. New Project butonuna tıkla
4. "hotel\_booking" adında proje oluştur
5. Version 1 otomatik oluşturuldu mu kontrol et
6. General prompt gir
7. LLM config ayarla
8. 2 intent ekle (check-availability, make-reservation)
9. Her intent'e 3'er parametre ekle
10. API tanımlarını yap
11. Version'ı publish et
12. Spark'a notification gitti mi kontrol et

**Senaryo 2: Mevcut Projeyi Güncelleme**

1. Login yap
2. "airline\_agent" projesini seç
3. Yeni version oluştur (v2'den kopyala)
4. Yeni intent ekle: "seat-selection"
5. Parametreler: flight\_number, seat\_number
6. Test tab'ından intent detection test et
7. Version'ı publish et
8. Eski version unpublish oldu mu kontrol et

**Senaryo 3: Race Condition Testi**

1. İki browser'da farklı kullanıcılarla login
2. Aynı projenin aynı version'ını aç
3. User1: Intent adını değiştir
4. User2: Aynı intent'i silmeye çalış
5. Race condition hatası alındı mı kontrol et

**Senaryo 4: Import/Export Akışı**

1. Bir projeyi tüm detaylarıyla oluştur
2. Export et
3. Projeyi sil
4. Export edilen dosyayı import et
5. Tüm data'nın geldiğini kontrol et
6. ID'lerin yeniden oluşturulduğunu kontrol et

**Test Otomasyonu**

Test tab'ında çalışacak otomatik test script'leri:

// UI Test Example

async function testLoginFlow() {

 const result = {

 name: "Login Flow",

 steps: []

 };

 // Test 1: Successful login

 await fillInput('#username', 'admin');

 await fillInput('#password', 'correct\_password');

 await click('#loginBtn');

 if (await waitForElement('.project-list', 5000)) {

 result.steps.push({

 name: "Valid login",

 status: "PASS",

 time: getElapsedTime()

 });

 }

 // Test 2: Invalid login

 await logout();

 await fillInput('#username', 'admin');

 await fillInput('#password', 'wrong\_password');

 await click('#loginBtn');

 if (await elementExists('.error-message')) {

 result.steps.push({

 name: "Invalid login shows error",

 status: "PASS",

 time: getElapsedTime()

 });

 }

 return result;

}

// Backend Test Example

async function testAPIEndpoints() {

 const results = [];

 // Test project CRUD

 const projectData = {

 name: "test\_project\_" + Date.now(),

 caption: "Test Project"

 };

 // Create

 const createRes = await api.post('/projects', projectData);

 results.push({

 name: "Create project",

 status: createRes.status === 201 ? "PASS" : "FAIL",

 details: createRes

 });

 // Read

 const getRes = await api.get(`/projects/${createRes.data.id}`);

 results.push({

 name: "Get project",

 status: getRes.status === 200 ? "PASS" : "FAIL"

 });

 // Update

 const updateRes = await api.put(`/projects/${createRes.data.id}`, {

 ...projectData,

 caption: "Updated Caption"

 });

 results.push({

 name: "Update project",

 status: updateRes.status === 200 ? "PASS" : "FAIL"

 });

 // Delete

 const deleteRes = await api.delete(`/projects/${createRes.data.id}`);

 results.push({

 name: "Delete project",

 status: deleteRes.status === 204 ? "PASS" : "FAIL"

 });

 return results;

}

// Integration Test Example

async function testRaceCondition() {

 // Simulate two users editing same resource

 const user1Token = await login('user1', 'pass1');

 const user2Token = await login('user2', 'pass2');

 // Both get same project

 const project = await api.get('/projects/1', {

 headers: { Authorization: `Bearer ${user1Token}` }

 });

 // User1 updates

 const update1 = await api.put('/projects/1', {

 ...project.data,

 caption: "User1 Update"

 }, {

 headers: { Authorization: `Bearer ${user1Token}` }

 });

 // User2 tries to update with old last\_update\_date

 try {

 const update2 = await api.put('/projects/1', {

 ...project.data,

 caption: "User2 Update"

 }, {

 headers: { Authorization: `Bearer ${user2Token}` }

 });

 return {

 name: "Race condition detection",

 status: "FAIL",

 details: "Second update should have been rejected"

 };

 } catch (error) {

 if (error.response.status === 409) {

 return {

 name: "Race condition detection",

 status: "PASS",

 details: "Correctly rejected concurrent update"

 };

 }

 }

}

**Test Raporlama**

Test tab'ında her test run için:

* Başlangıç/bitiş zamanı
* Toplam test sayısı
* Başarılı/başarısız/skipped test sayıları
* Her test için detaylı log
* Export edilebilir JSON/HTML rapor
* Tarihsel test sonuçları grafiği

**Continuous Testing**

* Her deployment sonrası otomatik test run
* Scheduled daily test runs
* Test failure durumunda email/Slack notification
* Test coverage metrikleri

Bu test planı ile Flare Admin UI'ın tüm fonksiyonlarının düzgün çalıştığından ve yeni güncellemelerin mevcut özellikleri bozmadığından emin olabiliriz.