# **Smart Doc Insight - Sunuma Hazır Proje Analiz Raporu**

**(Slayt 1: Başlık Slaytı)**

Herkese merhaba, ben Mehmet Said Hüseyinoğlu. Bugün sizlere, Yazılım Gereksinim Analizi dersi kapsamında geliştirdiğim "Smart Doc Insight" adlı projemin detaylı analizini sunacağım. Bu sunumda, projemin çıkış noktasından başlayarak gereksinimlerine, mimarisine ve teknik modellemesine kadar olan tüm süreci, sizden gelen istekler doğrultusunda eksiksiz bir şekilde inceleyeceğiz.

### **1. Proje Amacı, Sorun ve Hedef Analizi**

**(Slayt 2: Sorun Tanımı)**

Öncelikle bu projeyi neden geliştirmeye karar verdiğimden bahsetmek istiyorum. Değerli hocalarımız, her dönem onlarca öğrenci projesini değerlendiriyor. Bu süreç, her bir proje dokümanını manuel olarak açıp, "Proje amacı belirtilmiş mi?", "Literatür taraması var mı?" gibi belirli kriterleri tek tek kontrol etmeyi gerektiriyor. Bu yöntem, özellikle proje sayısı arttığında ciddi anlamda **zaman alıcı, yorucu ve hataya açık** bir hale geliyor. İşte benim projem tam olarak bu verimsizlik sorununa modern bir çözüm bulmayı hedefliyor.

**(Slayt 3: Proje Amacı ve Hedefler)**

**Projemin temel amacı;** yapay zeka destekli bir web uygulaması geliştirerek, eğitmenlerimizin proje değerlendirme süreçlerini otomatize etmek ve verimliliği en üst düzeye çıkarmaktır.

Bu ana amaç doğrultusunda belirlediğim **hedefler** ise şunlar:

* **Verimliliği Artırmak:** Değerlendirme süreçlerini hızlandırmak.
* **Zaman Tasarrufu Sağlamak:** Manuel incelemeye harcanan zamanı azaltmak.
* **Anlamlı Bilgi Çıkarmak:** Yapay zeka ile dokümanlardan sadece anahtar kelime değil, özetler ve anlamlı raporlar oluşturmak.
* **Kolay Erişim ve Yönetim Sağlamak:** Tüm projelere ve analiz sonuçlarına tek bir yerden ulaşım imkanı sunmak.
* **Akıllı Analiz Sunmak:** Kullanıcıya farklı yapay zeka modellerini kullanma esnekliği tanıyarak, analiz sürecini kişiselleştirilebilir kılmak.

### **2. Detaylı İhtiyaç Belirtimi**

**(Slayt 4: Fonksiyonel Gereksinimler)**

Sistemin sahip olması gereken işlevleri, yani **fonksiyonel gereksinimleri**, kullanıcı rollerini temel alarak şu şekilde tanımladım:

* **Tüm Kullanıcılar İçin:**
  + **FR-G-01:** Kullanıcılar e-posta ve şifre ile sisteme kayıt olabilmelidir.
  + **FR-G-02:** Kayıtlı kullanıcılar sisteme giriş yapabilmelidir.
  + **FR-G-03:** Kullanıcılar kendi yükledikleri dosyaları silebilmelidir.
* **Öğrenciler İçin:**
  + **FR-Ogrenci-01:** Öğrenci yeni bir proje oluşturabilmelidir.
  + **FR-Ogrenci-02:** Öğrenci projesine takım arkadaşı ekleyebilmelidir.
  + **FR-Ogrenci-03:** Öğrenci projesine PDF formatında doküman yükleyebilmelidir.
* **Öğretmenler İçin:**
  + **FR-Ogretmen-01:** Öğretmen sistemdeki tüm projeleri ve detaylarını görüntüleyebilmelidir.
  + **FR-Ogretmen-02:** Öğretmen, analiz için yerel veya harici bir dil modeli (LLM) seçebilmelidir.
  + **FR-Ogretmen-03:** Öğretmen, seçtiği harici API için kendi API anahtarını sisteme girebilmeli ve daha sonra kullanmak üzere kaydedebilmelidir.
  + **FR-Ogretmen-04:** Öğretmen, bir dokümanı varsayılan veya kendi belirlediği özel başlıklarla analiz ettirebilmelidir.
  + **FR-Ogretmen-05:** Sistem, analiz sonuçlarını öğretmene sunmalı ve veritabanına kaydetmelidir.

**(Slayt 5: Fonksiyonel Olmayan Gereksinimler)**

Sistemin kalitesini belirleyen **fonksiyonel olmayan gereksinimlerim** ise şunlar:

* **Performans:** Yüksek boyutlu PDF'lerin analizi dahil olmak üzere tüm işlemler makul bir sürede tamamlanmalıdır.
* **Güvenlik:** Kullanıcı şifreleri ve API anahtarları gibi hassas veriler, veritabanında mutlaka **şifrelenerek (hashed)** saklanmalıdır.
* **Kullanılabilirlik:** Arayüz, teknik bilgisi olmayan bir kullanıcının bile rahatlıkla anlayıp kullanabileceği kadar basit ve sezgisel olmalıdır.
* **Güvenilirlik:** Analiz sırasında bir API hatası olduğunda sistem çökmek yerine, hatayı yönetip kullanıcıya anlaşılır bir mesaj göstermelidir.
* **Uyumluluk:** Web uygulaması, günümüzdeki tüm modern web tarayıcılarında (Chrome, Firefox, Safari, Edge) sorunsuz çalışmalıdır.

### **3. User Story’ler (Kullanıcı Hikayeleri)**

**(Slayt 6: User Stories)**

Kullanıcılarımızın ihtiyaçlarını onların ağzından ifade etmek için bazı kullanıcı hikayeleri oluşturdum:

* **Öğrenci Rolü:** "**Bir öğrenci olarak,** projeme takım arkadaşlarımı ekleyebilmek istiyorum, **böylece** proje üzerinde birlikte verimli bir şekilde çalışabiliriz."
* **Öğretmen Rolü:** "**Bir öğretmen olarak,** bir dokümandaki belirli bilgilerin varlığını otomatik olarak kontrol ettirebilmek istiyorum, **böylece** değerlendirme sürecini manuel yapmaya göre çok daha hızlı tamamlayabilirim."

Bu hikayeler, projenin gerçek dünya problemlerine nasıl dokunduğunu net bir şekilde gösterir.

### **4. Use Case Tanımları ve Diyagramı**

**(Slayt 7: Use Case Diyagramı)**

Ekranda gördüğünüz **Use Case diyagramı**, sistemimle etkileşime girecek olan **Öğrenci** ve **Öğretmen** aktörlerini (çöp adamlar) ve onların sistem üzerinde gerçekleştirebileceği temel eylemleri (elipsler) özetliyor. Bu diyagram, sistemin genel işlevselliğini bir bakışta anlamamızı sağlar.

**(Slayt 8: Use Case Detay Tablosu)**

Diyagramdaki her bir senaryoyu daha da detaylandırabiliriz. İşte projenin en kritik işlevi olan **"Dokümanı Analiz Et"** senaryosu için hazırladığım sektörel yaklaşım tablosu:

| **Use Case Adı:** | **Dokümanı Analiz Et** |
| --- | --- |
| **ID:** | UC-01 |
| **Aktör(ler):** | Öğretmen |
| **Amaç:** | Bir öğrenci projesine ait PDF dokümanını, seçilen LLM sağlayıcısı ile belirli başlıklar altında analiz ederek sonuçları elde etmek. |
| **Ön Koşullar:** | 1. Öğretmen sisteme giriş yapmıştır.  2. Sistemde analiz edilecek bir doküman mevcuttur. |
| **Son Koşullar:** | 1. Analiz başarılı ise sonuçlar ekranda gösterilir ve veritabanına kaydedilir.  2. Analiz başarısız ise kullanıcıya bir hata mesajı gösterilir ve dokümanın durumu "Hata" olarak güncellenir. |
| **Ana Akış (Başarılı Senaryo):** | 1. Öğretmen, proje listesinden bir projeyi ve analiz etmek istediği dokümanı seçer.  2. Sistem, analiz ayarları panelini gösterir.  3. Öğretmen, LLM sağlayıcısını (yerel veya API) ve analiz başlıklarını seçer.  4. Öğretmen "Analizi Başlat" butonuna tıklar.  5. Sistem, PDF dosyasını metne dönüştürür.  6. Sistem, seçilen LLM sağlayıcısına analiz isteği yapar.  7. LLM sağlayıcısından gelen başarılı sonuçlar alınır.  8. Sistem, sonuçları kullanıcıya anlaşılır bir formatta sunar ve veritabanına kaydeder.  9. Dokümanın durumu "Analiz Edildi" olarak güncellenir. |
| **Alternatif Akışlar (İstisnalar):** | 6a. LLM sağlayıcısı ile iletişimde hata oluşursa (örn: geçersiz API anahtarı):      1. Sistem hatayı loglar.      2. Kullanıcıya "Analiz sırasında bir hata oluştu." mesajını gösterir.      3. Dokümanın durumunu "Hata" olarak günceller. Use case sonlanır. |

### **5. Zengin Görsel Gösterimler (Rich Picture & Context Diagram)**

**(Slayt 9: Rich Picture & Context Diagram)**

Sistemi daha geniş bir perspektiften anlamak için iki görsel modelleme tekniği kullandım:

* **Rich Picture (Zengin Resim):** Bu serbest çizim, projenin sadece teknik bileşenlerini değil, aynı zamanda sosyal ve insani yönlerini de gösterir. Öğretmenlerin "zaman yetersizliği" gibi endişeleri, öğrencilerin "takım çalışması" ihtiyacı gibi unsurlar bu resimde bir araya gelir.
* **Context Diagram (Bağlam Diyagramı):** Bu daha yapısal diyagram ise sistemimi merkezde tek bir süreç olarak ele alır ve dış dünyadaki varlıklarla (Öğrenci, Öğretmen, Harici LLM'ler) arasındaki veri alışverişini net oklarla gösterir. Örneğin, sistemimizin harici bir LLM sağlayıcısına Analiz Edilecek Metin gönderdiğini ve ondan İşlenmiş Sonuç aldığını bu diyagramda görebiliriz.

### **6. Sistem Mimarisi Tanımı**

**(Slayt 10: Sistem Mimarisi ve Teknolojiler)**

Projemin teknik altyapısını kurarken, modern, esnek ve ölçeklenebilir bir yapı olan **Katmanlı Mimari**'yi tercih ettim. Bu mimari, **İstemci-Sunucu** modeli üzerine kuruludur.

* **Genel Sistem Mimarisi:**
  + **İstemci (Client):** Kullanıcıların gördüğü web arayüzü (HTML/CSS/JS).
  + **Sunucu (Server):** Tüm iş mantığını yürüten Node.js (Express) uygulaması.
  + **Veri Depolama (Data Storage):** Esnek yapısı nedeniyle NoSQL bir veritabanı olan MongoDB.
* **Kullanılacak Teknolojiler:**
  + **Backend:** Node.js, Express.js
  + **Veritabanı:** MongoDB, Mongoose (ODM)
  + **PDF İşleme:** pdf-parse kütüphanesi
  + **Dosya Yükleme:** Multer middleware
  + **LLM Entegrasyonu:** Ollama, LM Studio, Gemini API, OpenAI API

### **7. ERD (Varlık-İlişki Diyagramı) ve Veri Sözlüğü**

**(Slayt 11: ERD ve Veri Sözlüğü)**

Sistemin bel kemiği olan veriyi nasıl saklayacağımızı ise **Varlık-İlişki Diyagramı (ERD)** ile mantıksal olarak modelledim. Bu diyagram Users, Projects ve Documents gibi temel varlıklar arasındaki ilişkileri gösterir.

Bu yapıyı daha da detaylandırmak için bir **Veri Sözlüğü** hazırladım. İşte bazı önemli koleksiyonlar ve alanları:

**Users Koleksiyonu**

| **Alan Adı** | **Veri Tipi** | **Açıklama** |
| --- | --- | --- |
| \_id | ObjectId | Eşsiz kimlik numarası. |
| name | String | Kullanıcının adı soyadı. |
| email | String | Eşsiz e-posta adresi (giriş için kullanılır). |
| password | String | Şifrelenmiş (hashed) kullanıcı parolası. |
| role | String | Kullanıcının rolü: 'student' veya 'instructor'. |

**Projects Koleksiyonu**

| **Alan Adı** | **Veri Tipi** | **Açıklama** |
| --- | --- | --- |
| \_id | ObjectId | Eşsiz proje kimliği. |
| title | String | Projenin başlığı. |
| description | String | Projenin detaylı açıklaması. |
| creator | ObjectId | Projeyi oluşturan öğrencinin Users'daki referansı. |
| members | Array[\_ObjectId] | Projeye dahil olan öğrencilerin Users'daki referansları. |

**Documents Koleksiyonu**

| **Alan Adı** | **Veri Tipi** | **Açıklama** |
| --- | --- | --- |
| \_id | ObjectId | Eşsiz doküman kimliği. |
| filename | String | Yüklenen dosyanın orijinal adı. |
| project | ObjectId | Dokümanın ait olduğu projenin referansı. |
| status | String | Dokümanın analiz durumu: 'awaiting\_analysis', 'analyzed', 'error'. |

### **8. UML Diyagramları**

**(Slayt 12: UML Diyagramları)**

Sistemin hem statik yapısını hem de dinamik davranışlarını daha derinlemesine incelemek için standart UML diyagramlarını hazırladım:

* **Sınıf Diyagramı:** User, Project, Document gibi ana sınıfları, niteliklerini ve metotlarını gösterir. Kodun iskeletini oluşturur.
* **Aktivite Diyagramı:** "Doküman Analizi" gibi bir iş akışının adımlarını, karar noktalarını ve akışını görselleştirir.
* **Sıralama (Sequence) Diyagramı:** Analiz sürecinde Öğretmen, Arayüz, Sistem ve LLMSağlayıcı nesneleri arasında gönderilen mesajları ve bunların zaman içindeki sırasını gösterir.
* **Durum (State) Diyagramı:** Bir Document nesnesinin yaşam döngüsünü, yani Yüklendi, Analiz Bekliyor, Analiz Edildi ve Hata Oluştu gibi durumlar arasındaki geçişleri modeller.
* **Bileşen Diyagramı:** Sistemin ana modüllerini (Kullanıcı Arayüzü, SmartDoc API, Veritabanı) ve aralarındaki arayüzleri gösterir.
* **Deployment Diyagramı:** Sistemin fiziksel olarak sunuculara nasıl dağıtılacağını (Uygulama Sunucusu, Veritabanı Sunucusu) ve donanım bileşenlerinin nasıl bağlandığını gösterir.

### **9. Anketler**

**(Slayt 13: Anketler)**

Gereksinimleri toplama ve doğrulama aşamasında potansiyel kullanıcılara yönelik anketler yapmak, projenin başarısı için kritik öneme sahiptir. Hazırlanabilecek bazı örnek anket soruları şunlar olabilir:

* **Öğretmenler İçin:**
  1. Bir proje dokümanını incelerken en çok hangi bilgileri ararsınız? (Örn: Amaç, Yöntem, Sonuçlar)
  2. Proje değerlendirme sürecinde en çok zamanınızı alan şey nedir?
  3. Böyle bir otomatik analiz aracının size sunacağı sonuçlara ne kadar güvenirsiniz?
* **Öğrenciler İçin:**
  1. Proje dokümanlarınızı ve ilerlemenizi yönetmek için bir platforma ihtiyaç duyuyor musunuz?
  2. Bir proje yönetim aracında sizin için en önemli özellik ne olurdu?

**(Slayt 14: Kapanış)**

Özetlemek gerekirse, "Smart Doc Insight" projesi, eğitimdeki önemli bir verimlilik sorununa modern teknolojilerle çözüm sunmayı amaçlayan, gereksinimleri ve mimarisi detaylıca analiz edilmiş kapsamlı bir sistemdir.

Beni dinlediğiniz için teşekkür ederim. Sorularınız varsa memnuniyetle cevaplayabilirim.