BİL440 – Yapay Zeka Destekli Yazılım Geliştirme

Final Projesi Teknik Raporu

Proje: Akıllı Doküman Arama ve Özetleme Sistemi (Proje #2)
Öğrenciler: Damla Nur Alper (220601017) – Sidal Deniz Bingöl (210601009)
Tarih: 10 Ocak 2026

1. GİRİŞ VE PROJE FELSEFESİ

1.1 Problem ve Kapsam

Akademik ve profesyonel dünyada uzun dokümanlardan anlamsal bilgi çıkarımı yapmak klasik yöntemlerle (Ctrl+F) zordur. Bu proje; TF-IDF tabanlı anahtar kelime arama ile LLM tabanlı (RAG) soru-cevap ve özetleme fonksiyonlarını birleştiren web tabanlı bir çözüm sunar.

1.2 AI-Augmented Yaklaşımı

Proje, yazılım yaşam döngüsünün (SDLC) her aşamasında yapay zekayı bir karar destek mekanizması olarak konumlandırmıştır. Temel ilkeler şunlardır:

- AI önerilerini sorgusuz kabul etmemek, gerekirse reddetmek veya revize etmek
- Alınan her kararı AI Decision Log ile kayıt altına almak
- Deterministik işlemler (PDF parsing, arama) ile olasılıksal işlemleri (LLM) keskin bir sınırla ayırmak

2. SİSTEM MİMARİSİ VE TEKNOLOJİ STACK

Sistem, servis odaklı bir mimariyle geliştirilmiştir. Modüler yapı sayesinde AI bileşenlerinin hata etkisi minimize edilmiştir.

Katman – Teknoloji – Seçim Nedeni

- Backend: Python / FastAPI
  Yüksek performans (ASGI), otomatik OpenAPI ve LLM entegrasyonu
- Frontend: React 19 / Vite 7
  Hızlı iterasyon, modern state yönetimi ve bileşen yapısı
- Styling: Tailwind CSS 4.x
  Utility-first yaklaşımı ile hızlı UI prototipleme
- AI / LLM: Groq (Llama-3.1-8b)
  Düşük gecikme süresi ve RAG pipeline uygunluğu
- Metin İşleme: PyMuPDF & Scikit-learn
  PDF metin çıkarımı ve TF-IDF vektörizasyonu için güvenilir kütüphaneler

2.1 Sistem Mimarisi

Sistem mimarisi component diagram olarak tasarlanmıştır (rapora PNG olarak eklenecektir).

Mimari Bileşenler:

- Frontend (Web UI): React + Vite + Tailwind CSS
- Backend API: FastAPI (Python 3.11)
- Doküman Depolama: Local file system (data/uploads/, data/extracted/)
- Arama Motoru: TF-IDF (scikit-learn) – klasik yöntem, AI kullanılmaz
- Büyük Dil Modeli (LLM): Groq API (llama-3.1-8b-instant)

AI Kullanım Sınırları (Bilinçli Karar):

- AI kullanılan alanlar:
  Doküman özetleme, soru-cevap (RAG)
- AI kullanılmayan alanlar:
  PDF parsing, TF-IDF arama, kaynak atfı, indeksleme

Gerekçe:
Deterministik işlemlerde AI kullanımı gereksiz risk yaratır. PDF parsing klasik kütüphanelerle yapıldığında sonuç her zaman aynıdır ve debug edilebilir yapıdadır.

2.2 Sistem Performansı ve Kısıtlamalar

Performans Metrikleri (Test Ortamı):

- PDF upload ve text extraction: 2–5 saniye (sayfa sayısına bağlı)
- TF-IDF search: <100 ms (10 doküman için)
- AI summarization: 3–8 saniye (Groq API latency)
- Q&A (RAG): 5–10 saniye (search + LLM)

Sistem Kısıtlamaları:

1. Dosya boyutu: maksimum 16 MB (FastAPI default limit)
2. LLM context window: 128k token, 100k+ karakter dokümanlar truncate edilmektedir
3. Eşzamanlılık: tek kullanıcı demo (multi-user authentication yok)
4. Dil desteği: İngilizce ve Türkçe (test edilmiştir)
5. Arama yöntemi: TF-IDF (keyword-based), semantic search yoktur

Ölçeklenebilirlik:

- Mevcut yapı: tek sunucu, lokal storage
- Production ortamı için: Redis cache, PostgreSQL metadata ve S3 tabanlı storage gereklidir

3. AI KULLANIM STRATEJİSİ VE ANALİZİ

3.1 Araçlar ve Karar İstatistikleri

Proje sürecinde dört farklı AI aracı hibrit biçimde kullanılmıştır:

- GitHub Copilot: Kod yazımı ve helper function'lar
- ChatGPT: Mimari kararlar ve dokümantasyon
- Gemini: API kullanımı ve test generation
- Claude Code: Terminal işlemleri ve otomasyon

Karar Dağılımı (Toplam 47 Karar):

- Kabul edilen: %56 (26 adet)
- Reddedilen: %21 (10 adet)
- Revize edilen: %23 (11 adet)

Kritik çıkarım:
AI önerilerinin %44'sinde insan müdahalesi zorunlu olmuştur. Bu durum, "AI her zaman doğru" algısının yanlış olduğunu göstermektedir.

3.2 AI Decision Log (Özet Tablo)

- Karar 5 (Analiz):
  AI kullanım sınırları, ChatGPT tarafından önerildi ancak PDF parsing AI'ya bırakılmadı.
- Karar 13 (Geliştirme):
  LLM servis yapısı Gemini ve Claude önerileri birleştirilerek hibrit şekilde tasarlandı.
- Karar 21 (Geliştirme):
  Tailwind 4.x setup Claude tarafından yanlış önerildiği için reddedildi.
- Karar 42 (Geliştirme):
  Decommissioned model önerildiği için reddedildi.
- Karar 43 (Test):
  Edge-case senaryoları Gemini tarafından doğru öngörüldüğü için kabul edildi.

4. KRİTİK ETKİLEŞİM ÖRNEKLERİ VE HİBRİT YAKLAŞIM

Örnek 1: LLM Service Implementation (Hibrit)

Prompt:
"Groq API ile RAG tabanlı Q&A servisi yaz."

AI katkıları:

- Gemini: Doğru Groq API syntax
- Copilot: Hallucination prevention için NO_ANSWER_TEXT constant ve temperature=0.0
- Claude: Class-based yapı, strict system prompt'lar

İnsan kararı:
Üç farklı AI çıktısı birleştirilerek stabil ve güvenli bir servis oluşturulmuştur.

Örnek 2: Tasarım ve UX (İnsan Önceliği)

AI önerisi:
OCR, semantic search ve tüm aşamalarda AI kullanımı.

İnsan kararı:
Açıklanabilirlik ilkesi gereği kaynak atıfları ve anahtar kelime araması deterministik koda bırakılmıştır.

Gerekçe:
Kullanıcı, aramanın neden bu sonucu verdiğini TF-IDF skorları üzerinden anlayabilmelidir.

5. KULLANICI ARAYÜZÜ VE KULLANIM SENARYOLARI

5.1 UI Tasarım Felsefesi

"Academic Minimalism with Modern Polish"

- Neutral ve teal renk paleti (slate-50/200/900, teal-700)
- Geist Sans font (2025 trend, Vercel modern fontu)
- Quiet UI yaklaşımı: border ve shadow'lar hover durumunda aktifleşir

- Öncelik: bilişsel netlik, görsel gösterişten önce gelir

5.2 Ana Özellikler ve Ekranlar

1. Doküman Yükleme

- Drag & drop veya tıklayarak yükleme
- Desteklenen formatlar: PDF, TXT, MD
- Dosya validasyonu (MIME type ve uzantı kontrolü)

2. Doküman Listesi

- Grid layout ile tüm yüklenen dokümanlar
- Metadata: sayfa sayısı, dosya tipi, yüklenme tarihi
- İşlemler: view, summarize, delete

3. Arama (TF-IDF Keyword-Based)

- Enter tuşu veya buton ile arama
- Relevance score gösterimi (explainability)
- Highlight edilmiş snippet'lar

4. AI Soru-Cevap (RAG Pipeline)

- Doğal dilde soru girişi
- Top 5 ilgili doküman retrieval
- Kaynak atfı (hangi dokümanlardan geldiği açıkça gösterilir)
- "AI yanıtı hazırlanıyor" loading durumu

5. Özetleme (Kısa / Detaylı)

- Kısa özet: 1–2 paragraf, temperature=0.3
- Detaylı özet: 4–5 paragraf, temperature=0.7
- Kullanılan model bilgisi kullanıcıya gösterilir

5.3 Responsive Tasarım

- Mobil: tek kolon, kompakt spacing
- Tablet: iki kolon grid
- Desktop: üç kolon grid, geniş spacing

5.4 Accessibility

- Klavye navigasyonu (Tab / Enter)
- Focus state'ler (teal ring)
- ARIA label'lar ile ekran okuyucu desteği

6. SİSTEM KURULUMU VE DEPLOYMENT

6.1 Gereksinimler

Backend:

- Python 3.11+
- Kütüphaneler: PyMuPDF, FastAPI, scikit-learn, Groq SDK, pytest

Frontend:

- Node.js 18+
- Framework: React 19, Vite 7, Tailwind CSS 4.x

6.2 Kurulum Adımları

Backend:

cd backend

pip install -r requirements.txt

cp .env.example .env

Groq API key ekle: GROQ_API_KEY=xxx

uvicorn app.main:app --reload --port 8000

Frontend:

cd frontend

npm install

npm run dev

localhost:5173

6.3 API Endpoints

- POST /api/v1/documents/upload – Doküman yükleme
- GET /api/v1/documents – Doküman listesi
- GET /api/v1/documents/{id}/download – PDF indirme
- DELETE /api/v1/documents/{id} – Doküman silme
- POST /api/v1/search – TF-IDF arama
- POST /api/v1/ai/summarize – Doküman özetleme
- POST /api/v1/ai/qa – RAG tabanlı soru-cevap

7. TEST VE HATA AYIKLAMA

7.1 Test Stratejisi

Backend tarafında Pytest framework kullanılarak toplam 23 adet unit test yazılmıştır.

Test Dosyaları ve Kapsam:

- test_edge_cases.py
  Hallucination, scanned PDF ve büyük doküman senaryoları
- test_pdf_service.py
  PDF metin çıkarımı ve metadata parsing
- test_routers.py
  API endpoint testleri (upload, list, delete)

- test_search_service.py
  TF-IDF arama fonksiyonları
- test_ai_service.py
  LLM servisleri ve temperature doğrulamaları

Tüm testler başarıyla geçmiştir (23/23).

7.2 Test Kategorileri ve Kapsam

Backend Unit Testleri:

- API endpoint doğrulaması
- Service layer izolasyonu
- External dependency'lerin mock'lanması

Integration Testleri:

- FastAPI TestClient ile uçtan uca senaryolar
- PDF, TXT ve MD dosya formatlarının doğrulanması

Edge Case Testleri:

- Hallucination prevention (temperature=0.0)
- Scanned PDF (empty text durumları)
- Büyük doküman truncation (100k+ karakter)

Frontend Test Yaklaşımı:

- Manuel UI testleri
- Keyboard navigation ve focus state kontrolleri
- Responsive tasarım kontrolleri

Test Kısıtlamaları:

- Frontend için otomatik test suite yazılmamıştır
- E2E testler kapsam dışı bırakılmıştır
- Performans ve load testing yapılmamıştır

7.3 Test Sonuçları ve Analiz

Test Çalıştırma:

pytest tests/ -v

===== 23 passed in 5.57s =====

Kritik Test Senaryoları:

1. Hallucination Prevention
   Doküman dışı sorularda LLM yeterli bilgi yok yanıtı döndürmektedir.
2. Scanned PDF
   Image-only PDF'ler sistemin çökmesine neden olmamaktadır.
3. Large Document
   Context window taşması truncate mekanizması ile engellenmiştir.

Mock Stratejisi:

- Groq API mock'lanmıştır
- PyMuPDF read işlemleri simüle edilmiştir
- Mock iterator eksikliği tespit edilip düzeltilmiştir

7.4 AI'nin Test Yazarken Yaptığı Hatalar

Hata 1: Mock Iterator Eksikliği

- Gemini, PyMuPDF mock'unda for page in doc döngüsünü simüle etmemiştir
- Çözüm: __len__ ve __getitem__ metodları eklenmiştir

Hata 2: Gerçek API Çağrısı Önerisi

- Copilot canlı Groq API çağrısı önerdi
- Reddedildi, test izolasyonu bozulacağı için uygun görülmedi

Hata 3: Kapsam Dışı Dil Testi

- Gemini Fransızca özet testi önerdi
- Scope creep olduğu için reddedildi

8. AI EXECUTION ERRORS (16 ADET)

Toplam 16 adet AI kaynaklı hata bilinçli şekilde dokümante edilmiştir. Bu hatalar, AI'ye aşırı güvenmenin risklerini somut biçimde ortaya koymaktadır.

8.1 AI Hata Matrisi

- Konfigürasyon Hataları: 4 adet
- API ve Model Hataları: 3 adet
- Hallucination: 3 adet
- Mock ve Test Hataları: 3 adet
- Görsel ve Mantıksal Hatalar: 3 adet

8.2 Detaylı Hata Örnekleri

Hata 1: Tailwind CSS 4.x PostCSS Breaking Change

- AI: Claude Code
- Sorun: Tailwind 3.x konfigürasyonu önerildi
- Gerçek: Tailwind 4.x için @tailwindcss/postcss gereklidir
- Sonuç: UI tamamen stylesız çalıştı
- Çözüm: Konfigürasyon güncellendi

Hata 2: Decommissioned Model

- AI: Gemini, Copilot, Claude
- Sorun: Kullanımdan kaldırılmış model önerildi
- Sonuç: 500 Internal Server Error

- Çözüm: llama-3.1-8b-instant modeline geçildi

Hata 3: pytest-groq Hallucination

- AI: Gemini
- Sorun: Var olmayan kütüphane önerildi
- Ders: Pattern recognition her zaman gerçek değildir

Hata 4: UI Design Görsel Hatası

- AI: Claude Code
- Sorun: Ocean Blue ve Sand paleti başarısız oldu
- Çözüm: Academic Minimalism yaklaşımına geçildi

9. ETİK, GÜVENLİK VE LİSANS

9.1 Kod Lisansı Riski

- FastAPI, React, Vite, Tailwind CSS: MIT License
- Groq SDK: Apache 2.0
- Geist Font: SIL Open Font License

Projede viral lisans bulunmamaktadır.

9.2 Veri Gizliliği

- Yüklenen PDF içerikleri Groq sunucularına gönderilmektedir
- Groq API stateless çalışmaktadır
- Kullanıcılara hassas veri yüklememeleri konusunda uyarı yapılmaktadır

9.3 Güvenlik Önlemleri

- Dosya uzantı ve MIME type doğrulaması
- Prompt injection'a karşı strict system prompt
- Temperature=0.0 ve RAG context-only yaklaşımı

9.4 Etik ve Bias Analizi

- LLM bias riski RAG ile sınırlandırılmıştır
- Telif sorumluluğu kullanıcıya aittir
- Kaynak atfı ve TF-IDF skorları ile açıklanabilirlik sağlanmıştır

10. SONUÇ VE DEĞERLENDİRME

Bu proje, AI destekli yazılım geliştirme sürecinde AI ile kod yazan değil, AI ile yazılım geliştiren mühendislik yaklaşımını ortaya koymuştur.

10.1 Proje Başarı Kriterleri

- Doküman yükleme ve yönetimi
- TF-IDF tabanlı arama
- AI özetleme ve RAG soru-cevap
- 23 unit test ve %100 başarı oranı

- 16 AI execution error dokümantasyonu

10.2 Çıkarılan Dersler

- Çoklu AI kullanımı hata oranını düşürür
- Versiyon değişimleri manuel kontrol gerektirir
- Deterministik işlemler klasik kodla yapılmalıdır
- Görsel kararlar tarayıcıda test edilmelidir

10.3 Teknik Borç ve Gelecek İyileştirmeler

- Pydantic v2 ve FastAPI lifespan migration
- Vector search entegrasyonu
- Streaming LLM response
- Multi-user destek

10.4 Proje Metrikleri

- Backend: 2000+ satır Python
- Frontend: 1500+ satır React
- Test: 23 test, %100 pass rate

10.5 En Büyük Ders

AI güçlü bir asistandır ancak mimari kararlar, güvenlik ve etik sınırlar insan tarafından çizilmelidir.

11. KAYNAKLAR: FastAPI, React, Tailwind CSS, Groq API, PyMuPDF, Gemini, ChatGPT, Claude, Github Copilot

GitHub Repository:
https://github.com/damlalper/ai-document-search-system

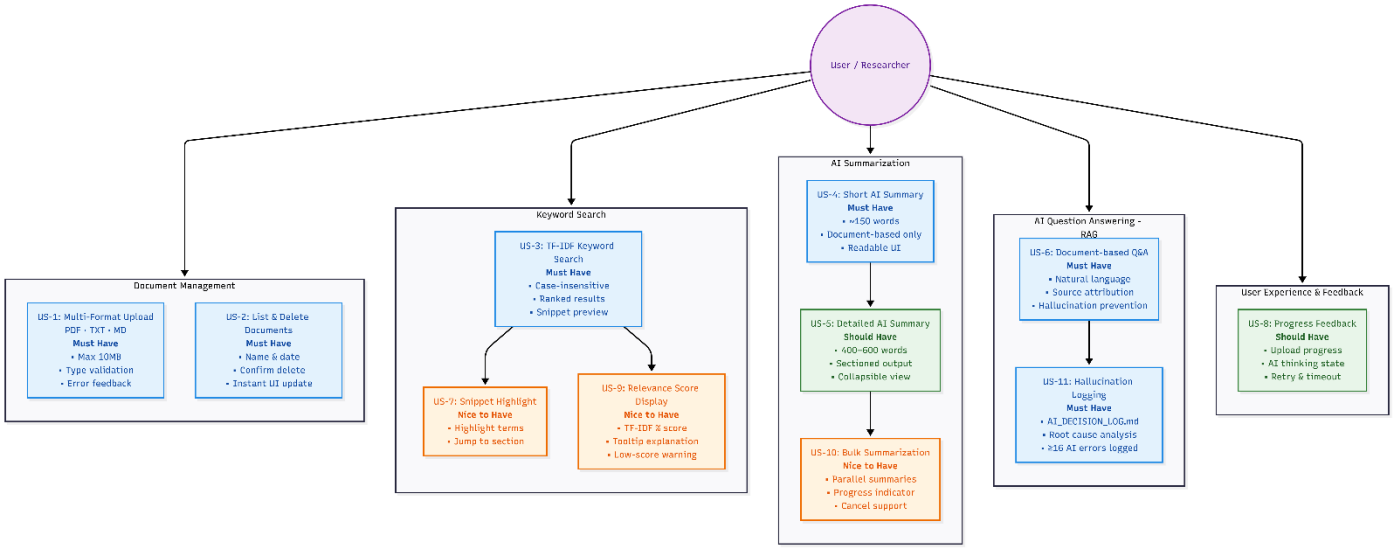
Bu teknik raporda yer sınırlamaları nedeniyle detaylı olarak ele alınamayan ancak proje sürecinin kritik parçalarını oluşturan dokümantasyon, GitHub repository'sindeki docs/ klasöründe bulunmaktadır.

https://github.com/damlalper/ai-document-search-system/tree/main/ai-document-search-system/docs

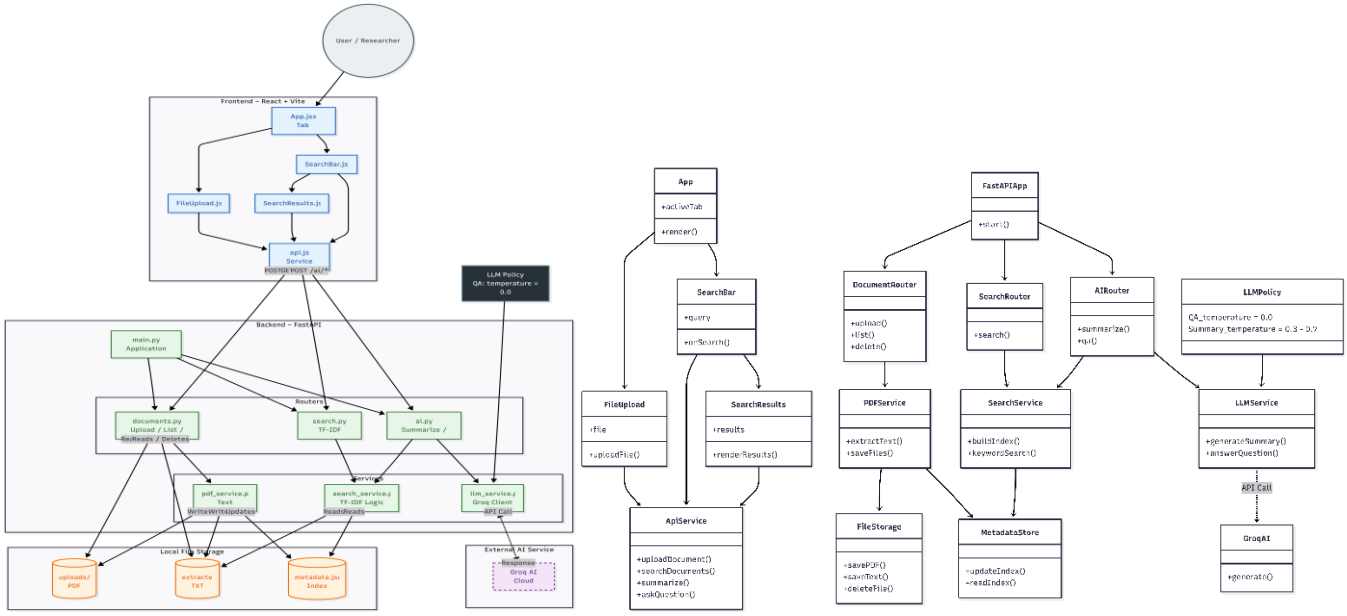
İçindekiler: Etik Güvenlik ve Lisans Değerlendirmesi, AI Araçları Kullanım Dokümantasyonu, Prompt Örnekleri ve Halüsinasyon Vakaları, AI Decision Log Tablosu, Kullanıcı Hikayeleri (User Stories), Test Stratejileri, Ortak Kararlar ve Proje Kuralları

12. EKRAN GÖRÜNTÜLERİ

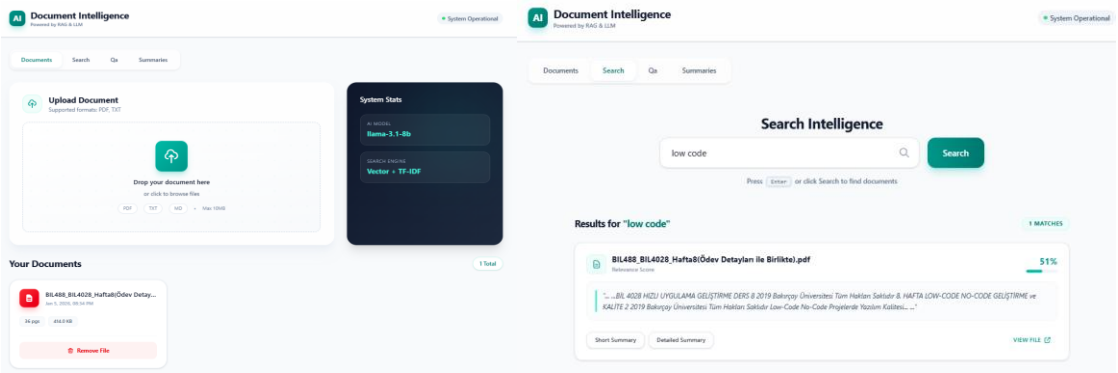
User Stories Diyagram

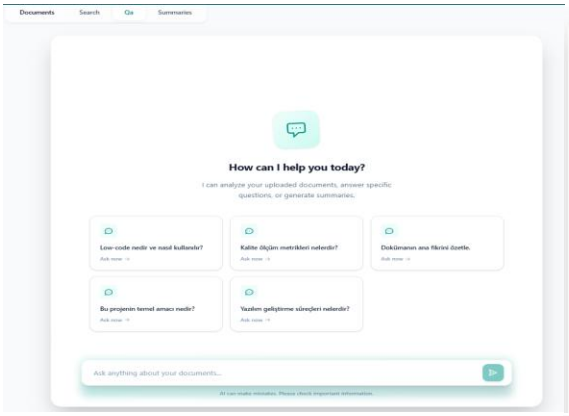


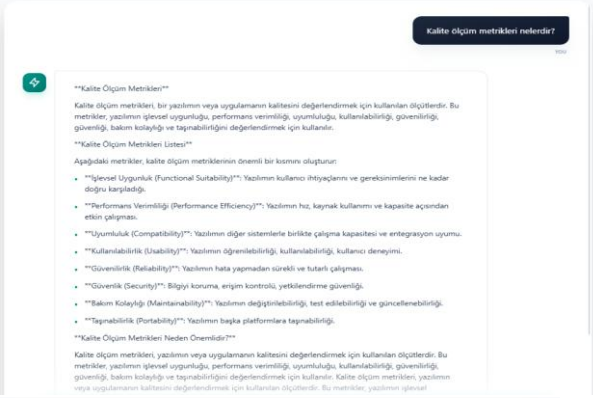
UML – ARCHITECTURE Diyagramları

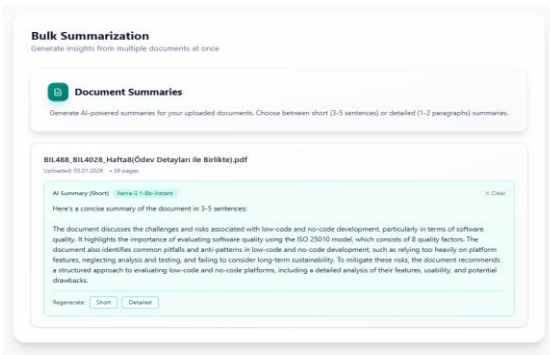


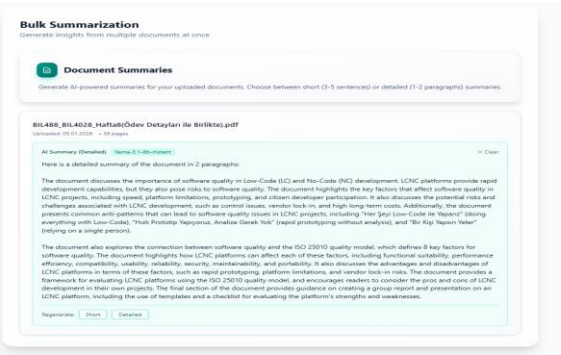
Frontend Ekran Görüntüleri




Documents
Search
QA
Summaries
How can I help you today?
I can analyze your uploaded documents, answer specific questions, or generate summaries.
Low-code nedir ve nasıl kullanılır?
Kalite ölçüm metrikleri nelerdir?
Dokümanın ana fikrini özetle.
Bu projenin temel amacı nedir?
Yazılım geliştirme süreçleri nelerdir?
Ask anything about your documents...
AI can make mistakes. Please check important information.


Kalite ölçüm metrikleri nelerdir?
Kalite Ölçüm Metrikleri
Kalite ölçüm metrikleri, bir yazılımın veya uygulamanın kalitesini değerlendirmek için kullanılan ölçütlerdir. Bu metrikler, yazılımın işlevsel uygunluğu, performans verimliliği, uyumluluğu, kullanılabilirliği, güvenilirliği, güvenliği, bakım kolaylığı ve taşınabilirliğini değerlendirmek için kullanılır.
Kalite Ölçüm Metrikleri Listesi
Aşağıdaki metrikler, kalite ölçüm metriklerinin önemli bir kısmını oluşturur:
İşlevsel Uygunluk (Functional Suitability): Yazılımın kullanıcı ihtiyaçlarını ve gereksinimlerini ne kadar doğru karşıladığı.
Performans Verimliliği (Performance Efficiency): Yazılımın hız, kaynak kullanımı ve kapasite açısından etkin çalışması.
Uyumluluk (Compatibility): Yazılımın diğer sistemlerle birlikte çalışma kapasitesi ve entegrasyon uyumu.
Kullanılabilirlik (Usability): Yazılımın öğrenilebilirliği, kullanılabilirliği, kullanıcı deneyimi.
Güvenilirlik (Reliability): Yazılımın hata yapmadan sürekli ve tutarlı çalışması.
Güvenlik (Security): Bilgiyi koruma, erişim kontrolü, yetkilendirme güvenliği.
Bakım Kolaylığı (Maintainability): Yazılımın değiştirilebilirliği, test edilebilirliği ve güncellenebilirliği.
Taşınabilirlik (Portability): Yazılımın başka platformlara taşınabilirliği.
Kalite Ölçüm Metrikleri Neden Önemlidir?
Kalite ölçüm metrikleri, yazılımın veya uygulamanın kalitesini değerlendirmek için kullanılan ölçütlerdir. Bu metrikler, yazılımın işlevsel uygunluğu, performans verimliliği, uyumluluğu, kullanılabilirliği, güvenilirliği, güvenliği, bakım kolaylığı ve taşınabilirliğini değerlendirmek için kullanılır. Kalite ölçüm metrikleri, yazılımın veya uygulamanın kalitesini değerlendirmek için kullanılan ölçütlerdir. Bu metrikler, yazılımın işlevsel


Bulk Summarization
Generate insights from multiple documents at once
Document Summaries
Generate AI-powered summaries for your uploaded documents. Choose between short (3-5 sentences) or detailed (1-2 paragraphs) summaries.
BIL488_BIL4028_Hafta8(Ödev Detayları ile Birlikte).pdf
Uploaded: 05.01.2026 • 26 pages
AI Summary (Short)
llama-3.1-8b-instant
Clear
Here's a concise summary of the document in 3-5 sentences:
The document discusses the challenges and risks associated with low-code and no-code development, particularly in terms of software quality. It highlights the importance of evaluating software quality using the ISO 25010 model, which consists of 8 quality factors. The document also identifies common pitfalls and anti-patterns in low-code and no-code development, such as relying too heavily on platform features, neglecting analysis and testing, and failing to consider long-term sustainability. To mitigate these risks, the document recommends a structured approach to evaluating low-code and no-code platforms, including a detailed analysis of their features, usability, and potential drawbacks.
Regenerate:
Short
Detailed


Bulk Summarization
Generate insights from multiple documents at once
Document Summaries
Generate AI-powered summaries for your uploaded documents. Choose between short (3-5 sentences) or detailed (1-2 paragraphs) summaries.
BIL488_BIL4028_Hafta8(Ödev Detayları ile Birlikte).pdf
Uploaded: 05.01.2026 • 26 pages
AI Summary (Detailed)
llama-3.1-8b-instant
Clear
Here is a detailed summary of the document in 2 paragraphs:
The document discusses the importance of software quality in Low-Code (LC) and No-Code (NC) development. LCNC platforms provide rapid development capabilities, but they also pose risks to software quality. The document highlights the key factors that affect software quality in LCNC projects, including speed, platform limitations, prototyping, and citizen developer participation. It also discusses the potential risks and challenges associated with LCNC development, such as control issues, vendor lock-in, and high long-term costs. Additionally, the document presents common anti-patterns that can lead to software quality issues in LCNC projects, including "Her Şeyi Low-Code ile Yapanı" (doing everything with Low-Code), "Hızlı Prototip Yapıyoruz, Analize Gerek Yok" (rapid prototyping without analysis), and "Bir Kişi Yapsın Yeter" (relying on a single person).
The document also explores the connection between software quality and the ISO 25010 quality model, which defines 8 key factors for software quality. The document highlights how LCNC platforms can affect each of these factors, including functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability. It also discusses the advantages and disadvantages of LCNC platforms in terms of these factors, such as rapid prototyping, platform limitations, and vendor lock-in risks. The document provides a framework for evaluating LCNC platforms using the ISO 25010 quality model, and encourages readers to consider the pros and cons of LCNC development in their own projects. The final section of the document provides guidance on creating a group report and presentation on an LCNC platform, including the use of templates and a checklist for evaluating the platform's strengths and weaknesses.
Regenerate:
Short
Detailed