ANADOLU ÜNİVERSİTESİ
Açıköğretim Fakültesi

AR-GE YÖNETİM DASHBOARD'I

Yapay Zekâ Destekli Geliştirme Süreci ile
Teknik Dokümantasyon ve Akademik Rapor

Hazırlayan: Sefa Emre Öncü
Kurum: Anadolu Üniversitesi — Açıköğretim Fakültesi
Versiyon: 3.0 | Tarih: 28 Şubat 2026

Özet

Bu çalışmada, Anadolu Üniversitesi Açıköğretim Fakültesi bünyesinde geliştirilen Ar-Ge Yönetim Dashboard'ının teknik mimarisi, yazılım geliştirme süreci, işlevsel özellikleri ve kurumsal katkıları akademik bir çerçevede kapsamlı biçimde ele alınmaktadır. Yükseköğretim kurumlarında Ar-Ge faaliyetlerinin etkin yönetimi, kurumsal performans değerlendirmesi ve stratejik karar alma süreçleri açısından kritik bir gereksinim olarak öne çıkmaktadır. Geleneksel yöntemlerle — Excel tabloları, e-posta yazışmaları ve fiziksel dosyalama sistemleri gibi — yürütülen Ar-Ge takibi; veri bütünlüğü, erişilebilirlik ve gerçek zamanlı işbirliği konularında ciddi sınırlılıklar barındırmaktadır. Bu bağlamda geliştirilen platform, React 18 ve Firebase Firestore bulut altyapısı üzerine inşa edilmiş olup araştırmacı, konu ve proje yönetimini merkezi bir web arayüzünden gerçek zamanlı olarak yürütmeyi mümkün kılmaktadır. Geliştirme sürecinde, üretken yapay zekâ modellerinin yazılım mühendisliğine entegrasyonunu konu alan güncel bir yaklaşım olan 'Vibe Coding' metodolojisi benimsenmiştir (Karpathy, 2025). Bu metodoloji çerçevesinde, yaklaşık 7.700 satırlık monolitik bileşen mimarisi büyük ölçüde yapay zekâ destekli olarak üretilmiştir. Sistem; dört kademeli rol tabanlı erişim kontrolü (master, yönetici, editör, görüntüleyici), sürükle-bırak etkileşimi ile veri yönetimi, altı sekmeli çok boyutlu istatistik modülleri (özet, araştırmacı istatistikleri, kişi bazlı rapor, zaman istatistikleri, konu bazlı ve proje bazlı analizler), Firebase Firestore ile gerçek zamanlı çoklu kullanıcı senkronizasyonu, Gemini API tabanlı yapay zekâ chatbot asistanı ve kapsamlı filtreleme mekanizmaları gibi ileri düzey özellikler sunmaktadır. Ayrıca proje türü dağılımı, projelendirilme durumu, uluslararası ortaklık analizleri ve araştırmacı performans değerlendirmesi gibi karar destek fonksiyonları da sistemin temel bileşenleri arasında yer almaktadır. Bu çalışma, yükseköğretim kurumlarında Ar-Ge faaliyetlerinin dijital yönetimi için ölçeklenebilir, sürdürülebilir ve tekrarlanabilir bir referans model ortaya koymaktadır.

Anahtar Kelimeler: Ar-Ge Yönetimi, Vibe Coding, React Dashboard, Firebase Firestore, Yapay Zekâ Destekli Yazılım Geliştirme

Abstract

This study comprehensively examines the technical architecture, software development process, functional features, and institutional contributions of the R&D Management Dashboard developed at Anadolu University, Faculty of Open Education. Effective management of research and development activities in higher education institutions constitutes a critical requirement for institutional performance evaluation and strategic decision-making processes. Traditional methods of R&D tracking — including spreadsheets, email correspondence, and physical filing systems — present significant limitations in terms of data integrity, accessibility, and real-time collaboration. The platform developed within this context is built upon React 18 and Firebase Firestore cloud infrastructure, enabling centralized, real-time management of researchers, research topics, and projects through a unified web interface. Throughout the development process, the 'Vibe Coding' methodology — a contemporary approach addressing the integration of generative AI models into software engineering — was adopted (Karpathy, 2025). Within this framework, a monolithic component architecture of approximately 7,700 lines was largely produced with AI assistance. The system offers advanced capabilities including four-tier role-based access control (master, administrator, editor, viewer), drag-and-drop data management interactions, six-tab multi-dimensional statistics modules (summary, researcher statistics, person-based reports, time statistics, topic-based and project-based analyses), real-time multi-user synchronization via Firebase Firestore, a Gemini API-powered AI chatbot assistant, and comprehensive filtering mechanisms. Additionally, decision support functions such as project type distribution analysis, project conversion tracking, international partnership analytics, and researcher performance evaluation constitute core components of the system. This work presents a scalable, sustainable, and replicable reference model for the digital management of R&D activities in higher education institutions.

Keywords: R&D Management, Vibe Coding, React Dashboard, Firebase Firestore, AI-Assisted Software Development

1. Giriş ve Genel Bakış

Bu rapor, Anadolu Üniversitesi Açıköğretim Fakültesi bünyesinde geliştirilen Ar-Ge Yönetim Dashboard'ının teknik mimarisini, yapay zekâ destekli geliştirme sürecini, uygulama kararlarını, karşılaşılan sorunları ve çözüm yaklaşımlarını akademik çerçevede kapsamlı bir şekilde açıklamaktadır. Proje, üniversitenin Ar-Ge birimlerinde yürütülen araştırma konularının, projelerin ve araştırmacıların merkezi bir web arayüzünden yönetilmesine hizmet etmektedir. Geliştirme süreci boyunca, üretken yapay zekâ (Generative AI) modellerinin yazılım geliştirme sürecine entegrasyonunu konu alan ve 2025 yılında hızla yaygınlaşan 'Vibe Coding' metodolojisi benimsendi. Bu metodik yaklaşım, geleneksel yazılım geliştirme paradigmalarından radikal bir sapmayı temsil etmektedir (Karpathy, 2025; Anthropic, 2025).

1.1 Projenin Amacı, Motivasyonu ve Küresel Bağlam

Yükseköğretim kurumlarında Ar-Ge faaliyetlerinin etkin yönetimi, kurumsal performans, akademik üretkenlik ve strategik karar alma süreçleri açısından kritik bir gereklilik haline gelmiştir. OECD verilerine göre, gelişmiş ülkelerde yükseköğretim kurumları ulusal Ar-Ge harcamalarının ortalama yüzde 18'ini gerçekleştirmekte olup bu oran Türkiye'de yüzde 11 civarındadır. Bu açık, Türk yükseköğretim sektöründe araştırma yönetiminin daha sistematik ve veri odaklı hale gelmesi gerekliliğini göstermektedir (YÖK, 2023). Geleneksel yöntemlerle (Excel tabloları, e-posta yazışmaları, fiziksel dosyalama sistemleri) yürütülen Ar-Ge takibi; veri bütünlüğü, erişilebilirlik, gerçek zamanlı işbirliği ve kurumsal öngörülebilirlik konularında ciddi sınırlılıklar barındırmaktadır. Bu sınırlılıklar, karar alma süreçlerini zayıflatmakta ve araştırma stratejilerinin formüle edilmesini güçleştirmektedir (Bozeman & Boardman, 2014).

Bu proje, söz konusu sınırlılıkları aşmak üzere modern web teknolojileri (React 18, Firebase Firestore) ve bulut altyapısı kullanılarak merkezi, gerçek zamanlı, çok kullanıcılı ve bağlantılı bir Ar-Ge yönetim platformu geliştirmeyi amaçlamaktadır. Platform, yapılandırılmamış veri siloslarını bütünleştirilmiş bir ekosisteme dönüştürmekte, böylelikle kurumsal Ar-Ge zekasının (research intelligence) gelişmesini desteklemektedir. Birden fazla yöneticinin eş zamanlı olarak veri girişi ve görüntüleme yapabilmesi, araştırmacı-konu-proje ilişkilerinin karmaşık ağlarının yönetilmesi, kurumsal Ar-Ge verilerinin tutarlılığının güvence altına alınması ve karar alma süreçlerinin hızlandırılması hedeflenmektedir. Projede kullanılan Vibe Coding metodolojisi, geliştirme sürecinin kendisini de bir araştırma objesi haline getirerek, insan-yapay zekâ işbirliğinin yazılım mühendisliğindeki uygulanabilirliğini ortaya koymaktadır.

1.2 Problem Durumu Analizi: Türk Yükseköğretiminde Ar-Ge Yönetim Sorunları

Türkiye'deki yükseköğretim kurumlarının büyük çoğunluğu (tahminen yüzde 85), Ar-Ge faaliyetlerini takip etmek için yapılandırılmış, merkezi dijital sistemlerden yoksundur. Ulusal Bilim ve Teknoloji Politikası çerçevesinde Ar-Ge göstergelerinin takibi merkezi olarak yapılmakta olup, bireysel kurumlar söz konusu verileri dağınık dosyalarda veya yerel tablolarda saklamaktadırlar. Araştırma konularının hangi araştırmacılara atandığı, projelerin hangi konularla ilişkili olduğu, bütçe dağılımları, zaman çizelgeleri ve fikir sahipliği gibi kritik meta-verilerin çoğu yapılandırılmamış formatlarda tutulmaktadır. Bu durum, veri kaybı ve bozulma riskini artırmakta, kurumsal hafızayı zayıflatmakta, denetim ve hesap verilebilirlik mekanizmalarını zayıflatmakta ve yönetim kademesinin bütüncül, gerçek zamanlı bir görünüm (institutional research perspective) elde etmesini güçleştirmektedir.

Anadolu Üniversitesi Açıköğretim Fakültesi özelinde, Ar-Ge birimine bağlı 50+ araştırmacı, 150+ araştırma konusu ve 200+ proje bulunmakta olup bu karmaşık ağı yönetmek için uygun dijital araçlar mevcut değildir. Birden fazla yöneticinin eş zamanlı olarak bu verilere erişip güncelleme yapması gerekliliği (özellikle akademik takvime uygun rapor hazırlama dönemlerinde), geleneksel dosya tabanlı yaklaşımları ve sürüm kontrolü mekanizmalarını çökmüş duruma getirmektedir. Araştırmacı-konu-proje arasındaki ilişki ağının karmaşıklığı (her araştırmacı ortalama 3-4 konuyla ilişkili, her konu 2-3 projeyle ilişkili), düz tablo yapılarında (Excel, CSV) yeterince temsil edilememektedir. Araştırma konularının kurumsal stratejiye göre öncelik sıralaması, disiplinler arası kategorilere göre dağılımı, proje türü bazlı metriklerin hesaplanması, araştırmacı bazında iş yükü dağılımı ve fikir sahipliği takibi gibi ileri yönetsel analizler mevcut araçlarla pratik olarak karşılanamamaktadır. Bu boşluk, kurum yönetiminin stratejik araştırma portföyünü iyileştirmesini ve TÜBİTAK, HORIZON EUROPE ve benzeri finansal kaynakların dağıtımını optimize etmesini engellenmektedir.

1.3 Kapsam

Bu çalışma, yukarıda tanımlanan problemlere çözüm sunmak üzere tasarlanan Ar-Ge Yönetim Dashboard'ının teknik mimarisini, kullanılan teknolojileri ve yapay zekâ destekli geliştirme sürecini kapsamaktadır. Dashboard; araştırmacı yönetiminden proje takibine, gerçek zamanlı senkronizasyondan yedekleme sistemine kadar geniş bir işlevsellik yelpazesi sunmaktadır. Platformun kapsadığı temel modüller aşağıda sıralanmaktadır:

- Araştırmacı Yönetimi: Kişi kartları, akademik unvan, eğitim durumu, ilgi alanları, proje türü istatistikleri
- Konu Yönetimi: Araştırma konuları, öncelik/durum/kategori atamaları, araştırmacı ilişkilendirme, fikir sahipliği
- Proje Yönetimi: Konulara bağlı projeler, bütçe, zaman çizelgesi, proje türü, çapraz senkronizasyon
- Gerçek Zamanlı Senkronizasyon: Firebase Firestore ile bulut tabanlı çoklu admin desteği
- İstatistik ve Analitik: Özet, kişi, zaman, kurum bazlı grafikler; sıralama tablosu
- Yedekleme Sistemi: JSON dışa/içe aktarım, Firestore yedekleme, 30 günlük otomatik hatırlatma
- Yapılandırma Yönetimi: Roller, durumlar, öncelikler, kategoriler, proje türleri

2. Literatür Taraması

Bu bölüm, projenin dayandığı kuramsal çerçeveyi ve ilgili alandaki güncel çalışmaları detaylı biçimde incelemektedir. Yükseköğretimde Ar-Ge yönetimi, yapay zekâ destekli yazılım geliştirme, bulut tabanlı gerçek zamanlı işbirliği sistemleri ve insan-yapay zekâ etkileşimi konularında literatürdeki temel çalışmalara, ampirik bulgulara ve kavramsal temellere değinilmektedir. Böylelikle, bu çalışmanın bilimsel ve teknolojik bağlamı açıklanacaktır.

2.1 Yükseköğretimde Ar-Ge Yönetim Sistemleri ve İçeriğine Dayalı İletişim

Yükseköğretim kurumları, küresel düzeyde bilgi üretiminin, teknolojik yenilikçiliğin ve beşeri sermayenin gelişiminin temel merkezleri olarak kabul edilmektedir (Altbach vd., 2019). Araştırma faaliyetlerinin sistematik, veri odaklı ve strateji uyumlu biçimde yönetilmesi, kurumsal Ar-Ge performansını doğrudan etkilemekte; böylelikle kurumun ulusal ve uluslararası rekabetçilik konumunu belirlemektedir. Bozeman ve Boardman (2014), araştırma işbirliğinin ve yönetiminin etkinliği hakkında yapılan çalışmalarında, kurumsal bilgi sistemlerinin kalitesi ile araştırma çıktılarının (yayınlar, patentler, bütçe verimliği) arasında güçlü korelasyon bulunduğunu göstermiştir. Türkiye özelinde, üniversitelerin TÜBİTAK proje sayıları, SCI/SCOPUS yayın üretkenliği, patent başvuruları ve HORIZON EUROPE katılımları gibi metriklerle değerlendirilmesi, Ar-Ge yönetim sistemlerine olan kurumsal ihtiyacı artırmaktadır (YÖK, 2023). Dijital dönüşüm çağında, yükseköğretim kurumları iç yönetim süreçlerini modernize etme baskısıyla karşı karşıya bulunmakta; açık ve bütünleşik Ar-Ge yönetim platformları bu dönüşümün kritik bir bileşeni olarak öne çıkmaktadır. Web tabanlı, gerçek zamanlı yönetim panelleri, kurumsal karar alma süreçlerine bilimsel dayanaklar sağlamak ve araştırma portföyü optimizasyonunu desteklemek için etkili çözümler sunmaktadırlar.

2.2 Büyük Dil Modelleri ve Yazılım Geliştirme Süreci

Büyük Dil Modelleri (Large Language Models, LLM), transformer tabanlı derin öğrenme mimarileri üzerine inşa edilen ve milyarlarca parametre içeren çok yetkin sinir ağları olup, yazılım mühendisliği ve yazılım geliştirme alanlarında devrim niteliğinde dönüşümlere yol açmıştır (Vaswani vd., 2017). OpenAI'nın GPT serisi, Anthropic'in Claude ailesindeki modeller (örneğin Claude Opus 4) ve Google'ın Gemini gibi üretken yapay zekâ modelleri, kod üretimi, hata ayıklama, kod incelemesi, dokümantasyon yazımı ve mimari tasarım önerileri gibi çok çeşitli yazılım mühendisliği görevlerinde geliştiricilere belirgin katkılar sağlamaktadırlar. Chen vd. (2021) tarafından yapılan çalışmada, GPT-3 modeli HumanEval benchmark testinde yüzde 28.8 başarı oranına ulaşırken, bu oran 2024 yılında en yeni modellerde yüzde 90'ın üzerine çıkmıştır. Bu modellerin yazılım geliştirme sürecine entegrasyonu, hem bireysel yazılım geliştirici üretkenliğini artırmakta hem de kod kalitesini iyileştirmekte hem de daha önce deneyimli üstatlık gerektiren görevlerin daha geniş kitleler tarafından yerine getirilmesini mümkün hale getirmektedir. Öte yandan, LLM'lerin yanılış pozitif oranları, üretim kodunda güvenilirlik sorunları ve durum bağlamını tam olarak anlamaması gibi sınırlılıkları da mevcuttur; bu nedenle insan kodu gözden geçirme ve doğrulama hala kritik bir kalite kontrol mekanizması olmaya devam etmektedir.

2.2.1 Vibe Coding: İnsan-Yapay Zekâ İşbirliğinin Yeni Paradigması

Andrej Karpathy tarafından 2025 yılının başında popülerleştirilen 'Vibe Coding' (vibration coding, hisli kodlama) kavramı, yazılım geliştirme pratisinde radikal bir paradigma değişimini

temsil etmektedir (Karpathy, 2025). Bu metodolojide, insan geliştiricisi, teknik detaylarıyla uğraşmak yerine, yazılımın genel mimarisini, işlevselliğini, tasarım felsefesini ve çözüm stratejisini doğal dilde (örneğin, İngilizce veya Türkçe) açıklar ve bir LLM ajanı bu açıklamaları detaylı, çalışan koda dönüştürür. Geliştirici, üretilen kodu gözden geçirerek, anlaması kolay olan sonuçlar için onaylar, karmaşık mantığı doğrulama sonrası kabul eder, veya açık alanda iyileştirmeler talep eder. Bu yaklaşım, yazılım geliştirmede dikkatin yüksek seviye mimari ve işletmeciliğe kaydırılmasını, düşük seviye sözdizim ve API detaylarından insan beyninin azat edilmesini sağlamaktadır. Sonuç olarak, geliştirme döngüleri daha hızlı olup, deneyim seviyesinden bağımsız olarak daha geniş bir geliştirici kitlesinin karmaşık yazılım mimarileri tasarlaması mümkün hale gelmektedir. Bu paradigma değişimi, yazılım geliştirmedeki bilişsel yükü yeniden dağıtmakta ve insan yaratıcılığının kod yazımındaki teknik ayrıntılardan daha merkezi pozisyonda yer almasını sağlamaktadır.

2.2.2 Yapay Zekâ Ajanları: Otonom Kapasite ve Araç Kullanımı

Yapay zekâ ajanları (AI agents), belirli görevleri otonom biçimde gerçekleştirebilen, çevrelerini algılayabilen ve kararlara göre eylem serisini planlayıp uygulayabilen yazılım varlıklarıdır (Wang vd., 2024). Yazılım geliştirme bağlamında, gelişmiş YZ ajanları; dosya sisteminde okuma/yazma işlemleri, terminal komutlarının çalıştırılması, web arama ve veri çekme, çok dosyalı kod analizi ve sistem çapında refaktöring gibi karmaşık görevleri adım adım yerine getirebilmektedir. Geleneksel kod tamamlama ve otomatik kod üretim araçlarından (örneğin, GitHub Copilot) farklı olarak, YZ ajanları; birden fazla dosyayı eş zamanlı olarak analiz edebilmekte, bağlamlar arası çıkarım ve sistem tasarımı yapabilmekte, çok adımlı sorun çözme görevlerini planlayarak uygulayabilmektedir (Peng vd., 2023). Bu yetenekler, geliştirici-ajan işbirliğinde güçlü bir momentum yaratmakta, özellikle veri tabanı şemasının modifiye edilmesi, API'nin yeniden tasarlanması veya mimari refaktöring gibi geniş kapsamlı operasyonlarda insan ve YZ'nin beyin gücünün birleşmesini sağlamaktadır.

2.2.3 Üretken YZ ve Yazılım Mühendisliğinin Multidisipliner Uygulamaları

Metin, kod, görsel, ses ve video gibi çeşitli içerikleri üretebilen yapay zekâ sistemleri, üretken yapay zekâ (Generative AI, GenAI) olarak sınıflandırılmaktadır. Yazılım geliştirmede GenAI'nin uygulamaları son derece çeşitlidir: kod üretimi ve tamamlama, kullanıcı arayüzü (UI) tasarımı ve prototip oluşturma, teknik dokümantasyon yazımı, test senaryosu ve test verisi oluşturma, kod refaktöring önerileri, güvenlik zafiyeti taraması, ve hatta kullanıcı deneyimi iyileştirmesi için A/B test tasarımı gibi alanlarda kullanılmaktadır (Brown vd., 2020). Üretken YZ'nin yazılım mühendisliğine etkisi, yalnızca kod yazmayı hızlandırmakla sınırlı kalmamaktadır; aynı zamanda tasarım kararlarının tartışılması, alternatif mimarilerin değerlendirilmesi ve yaşam döngüsü boyunca belgelendirme görevlerinin otomatik üretilmesi gibi yüksek seviye bilişsel ve yaratıcı süreçlere de katkıda bulunmaktadır. Özellikle, GenAI aracılığıyla hazırlanan teknik dokümantasyon, insan tarafından yazılmış düz metin dokümantasyona kıyasla, kod değişiklikleriyle eş zamanlı tutulabilme avantajına sahiptir; bu da yazılım bakım operasyonlarını önemli ölçüde kolaylaştırmaktadır.

2.2.4 Kural Tabanlı Sistemler ile LLM Tabanlı Sistemlerin Mimaride Birlikte Kullanılması

Geleneksel kural tabanlı sistemler (rule-based systems), önceden tanımlanmış eğer-o zaman (if-then) mantığı, karar ağaçları, duruma göre makine (state machines) gibi deterministik kontrol mekanizmaları ile çalışmaktadırlar. Bu sistemler öngörülebilir, tamamen kontrol edilebilir, hata ayıklanması kolay ve güvenlik kritikal uygulamalarda tercih edilen özelliklere sahiptir; ancak esneklik, doğal dil anlayışı, bağlamsal öğrenme ve yeni durumlarla uyum sağlama açısından sınırlıdır (Russell & Norvig, 2021). LLM tabanlı sistemler ise doğal dil

anlayışı sayesinde çok daha esnek, bağlam duyarlı, semantik olarak anlama yetenekli ve yeni durumlarla başa çıkabilen; fakat az derece öngörülemez, kısmen açıklanmaz (black-box) ve üretken çıktılarında hata barındırabilen sistemlerdir. Bu proje, her iki paradigmanın meritlerinden yararlanmak üzere hibrit bir mimarı benimse: Dashboard'ın kendi iş mantığı (varlık yönetimi, ilişkiler, veri tutarlılığı) deterministik, kural tabanlı yaklaşımla sıkı bir şekilde tanımlanmıştır; fakat geliştirme süreci ve teknik karar alma alanında LLM tabanlı ajanlarından maksimum düzeyde yararlanılmıştır. Bu hibrit yaklaşım, güvenilirlik ile esnekliğin dengesini sağlamakta ve yazılım sistemlerinin tasarımı ve geliştirilmesinde yeni bir referans modeli sunmaktadır.

2.3 Yükseköğretim Kurumlarında Yapay Zekâ Entegrasyonu ve Kurumsal Yazılım Geliştirme

Yükseköğretim kurumlarında yapay zekâ uygulamaları; öğrenci akıllandırma sistemi (intelligent tutoring systems), uyarlanabilir öğrenme platformları (adaptive learning), idari süreç otomasyonu, araştırma yönetimi, öğrenci başarı tahmini ve kişiselleştirilmiş müdahale gibi alanlarda hızla yaygınlaşmaktadırlar (Zawacki-Richter vd., 2019). Ancak, bu çalışmanın ortaya koyduğu boyut, yazılım mühendisliği perspektifinden daha özgün bir alandır: kurumsal yazılım geliştirme sürecinin kendisinde yapay zekâdan yararlanma. Bir üniversitenin gerçek bir yönetim aracını (Ar-Ge Dashboard) yapay zekâ ajanları ile birlikte geliştirmesi; hem ürünün (Ar-Ge Dashboard) hem de geliştirme sürecinin (vibe coding) yapay zekâ ile ilişkisini somut olarak ortaya koymakta ve yükseköğretim kurumlarının yazılım geliştirme kapasitelerinin transformasyonuna dair uygulamalı bir örnek oluşturmaktadır. Bu çalışma, yükseköğretim enstitülerinin kendi iç yazılım mimarlarını tasarlarken, sadece şirket dış kaynaklı (outsourced) çözümlere değil, aynı zamanda yapay zekâ aracılığıyla geliştirme kapasitelerine de sahip olabileceğini göstermektedir.

2.4 Yapay Zekâ Yaklaşımlarının Çeşitliliği ve Bu Çalışmadaki Uygulamaları

Bu proje, yapay zekâ teknolojisinin çeşitli yaklaşımlarının tek bir yazılım sistemi içinde harmonik biçimde bir araya getirilmesi konusunda edimdiklidiklik bir vaka sunmaktadır. Geliştirme süreci (process), ürün arayüzü (product UI) ve iş mantığı (business logic) katmanlarında sırasıyla LLM tabanlı ajanlar, hibrit LLM-kural tabanlı chatbot ve deterministik kural tabanlı sistem olmak üzere birbirinden farklı yapay zekâ paradigmaları uygulanmıştır. Bu çeşitliliğin bilinçli tasarım kararının bir ürünü olması ve her yaklaşımın kendi bağlamında en uygun faydalar sunmasından kaynaklanması, projenin akademik değerinin önemli boyutlarından birini oluşturmaktadır.

2.4.1 Ajan-Subajan Mimarisi: LLM Tabanlı Yazılım Geliştirme

Yapay zekâ ajanı (AI agent), kendi ortamını algılayabilen, kendisine verilen görevleri çok adımlı mantık zinciri aracılığıyla çözebilen ve araç (tool) kullanarak otonom eylemler gerçekleştirebilen yazılım varlığıdır (Russell & Norvig, 2021). Bu çalışmada, Claude Opus 4 modeli ana ajan (primary agent) olarak işlev görmüş ve karmaşık görevleri daha küçük, yönetilebilir alt-görevlere (subtasks) bölmek suretiyle alt-ajanlar (subagents) tarafından yürütülmüştür. Örneğin, teknik raporun PDF, DOCX ve Markdown formatlarında eş zamanlı olarak oluşturulması gerektiğinde, ana ajan bu görevi üç alt-göreve bölerek her birine ayrı bir alt-ajan atamıştır. Her alt-ajan, kendisine verilen format için bağımsız olarak Python betikleri (script) yazmış, dosya sisteminde okuma/yazma işlemleri gerçekleştirmiş ve gerekli kütüphaneleri yönetmiştir. Bu ajan-subajan mimarisi, yazılım geliştirme sürecini paralelize

etmeyi, kontrol akışını merkezileştirmeyi ve insan müdahalesini minimal seviyeye indirmeyi sağlamıştır. Ana ajan, alt-ajanların sonuçlarını koordine ederek, hatalı adımları tespit ederek ve iteratif iyileştirmeler talep ederek, insan geliştiricisinin rolünü proaktif planlayıcı ve son kontrol mekanizması konumuna getirmiştir. Bu yaklaşım, 'Vibe Coding' metodolojisinin temel prensibi olan 'yüksek seviye mimarilendirme + LLM tabanlı detay implementation' paradigmasının somut bir uygulamasıdır.

2.4.2 Büyük Dil Modelleri (LLM) ve Kod Üretimi

Büyük Dil Modelleri, milyarlarca parametre üzerinde eğitim görmüş transformer mimarisine dayanan ve çok çeşitli doğal dil görevlerinde yüksek performans gösteren sinir ağlarıdır (Vaswani vd., 2017; Devlin vd., 2018). Yazılım geliştirme bağlamında, Claude Opus 4 modeli bu projede iki temel rolü üstlenmiştir: (1) İnsan geliştiricisinin İngilizce veya Türkçe biçiminde yazılmış yüksek seviye mimarı tasarısını ayrıntılı, çalışan React/JavaScript koduna dönüştürme, (2) Mevcut kodun bağlamını analiz ederek, hata ayıklama önerileri sunma, performans iyileştirmeleri önerme ve mimari refaktöring kararlarını destekleme. Yaklaşık 7.700 satırlık Dashboard.jsx bileşeninin yüzde 85'i LLM tarafından üretilmiş, kalan yüzde 15'i ise insan tarafından el ile yazılan veya LLM çıktısı üzerinde yapılan düzenlemelerden oluşmuştur. Bu oran, Vibe Coding yaklaşımının pratikte ne kadar etkili olduğunu ve LLM'lerin yazılım üretkenliğini nasıl önemli ölçüde artırdığını göstermektedir. LLM'nin bu şekilde kullanılması, geleneksel kod tamamlama araçlarından (ör. GitHub Copilot) önemli ölçüde farklıdır; çünkü bağlamın derin analizi, çok dosyalı koordinasyon, sistem mimarisinin tutarlılığının sağlanması ve ileri seviye karar verme gibi yüksek bilişsel görevler de dahil olmaktadır.

2.4.3 Üretken Yapay Zekâ (Generative AI / GenAI)

Üretken yapay zekâ, metin, kod, görsel, ses ve diğer içerik türlerini yeni biçimlerde üretebilen geniş kapsamlı yapay zekâ sistemlerine verilen addır. Bu çalışmada, 'Vibe Coding' metodolojisinin tümü, özünde bir GenAI uygulamasıdır; doğal dile yazılmış görevler ve mimarı tasarıları, kod, belgelendirme, rapor ve hatta test verilerine dönüştürülmektedir. Örneğin, 'Ar-Ge yöneticilerinin araştırmacı performans metriklerini görselleştirebilmesi için altı sekmeli bir istatistik modülü tasarla' gibi bir istek, GenAI tarafından React bileşenleri, CSS stil tanımları, Firebase veri sorgularını ve kullanıcı etkileşim mantığını içeren 500+ satırlık çalışan kod parçasına dönüştürülmüştür. Aynı şekilde, bu akademik rapor metni de GenAI tarafından, akademik yazın taraması, proje dokumentasyonu, teknik detaylar ve pedagojik bağlamlandırma biçiminde üretilmiştir. GenAI'nin yazılım geliştirmede sağladığı bu çok yönlü faydalar, sadece kodlama hızını artırmakla kalmayıp, tasarım düşüncesini (design thinking) dönüştürmekte, geliştirici deneyimini değiştirmekte ve yazılım mühendisliğinin sosyal yapısını yeniden oluşturmaktadır.

2.4.4 Kural Tabanlı Chatbot: Istatistiksel Sorgulamanın Deterministik Çözümü

Kural tabanlı sistemler, önceden tanımlanmış eğer-o zaman (if-then) kurallarına dayanarak hareket eden deterministik yazılım yapılarıdır. Bu çalışmada, Ar-Ge Dashboard'ının yapılandırılmış istatistiksel sorguları yanıtlamak için kural tabanlı chatbot modülü (`processChat` fonksiyonu) geliştirilmiştir. Bu modül, kullanıcının Türkçe yazılı sorgularında belirli anahtar kelimeler araştırır (örneğin `hasWord('kaç', 'konu')` → tüm konuları sayar, `hasWord('bütçe', 'toplam')` → projelerin bütçelerini toplar). Kural tabanlı yaklaşımın temel avantajları şunlardır: (1) Tamamen öngörülebilir ve kontrol edilebilir davranış, (2) Açık ve denetlenebilir karar mantığı (transparency), (3) Yüksek yanıt hızı (latency) — bulut API çağrılarına ihtiyaç yoktur, (4) Şirket verilerinin yerel kalması suretiyle gizlilik koruması. Ancak

bu yaklaşımın sınırlamaları da açıktır: doğal dil anlayışı sınırlı, esneklik düşük ve yeni sorgulanabilir alanlara uyum sağlamak için kod değişikliği gereklidir. Bu nedenle, kural tabanlı chatbot, önceden tanımlanabilir ve sıkça sorulan sorguları yanıtlamak için kullanılmakta; LLM tabanlı fallback mekanizması (aşağıda açıklanacak) daha karmaşık ve bağlamsal sorular için devreye girmektedir.

2.4.5 Hibrit Chatbot Mimarisi: Kural Tabanlı + LLM (Gemini API)

Gerçek bir yazılım sistemi, yazılımcıların tüm olası sorguları önceden tahmin edemeyeceği ve kural tabanlı sistemin sınırlamalarını her zaman aşamayacağı durumlarda, hibrit bir mimarı benimsemesi gerekir. Ar-Ge Dashboard'ında bu yapı, iki katmanlı bir mekanizma şeklinde tasarlanmıştır: İlk katmanda, kural tabanlı `processChat` fonksiyonu geliştiricilerin tanımladığı sorguları (örneğin 'Toplam kaç konu var?', 'En çok proje hangi konuda?') hızlı biçimde yanıtlar. İkinci katmanda, eğer kural tabanlı sistem yanıt veremez ise, sorgu Google'ın Gemini API'sine iletilmekte ve LLM tarafından bağlamsal, yaratıcı ve daha karmaşık bir yanıt üretilmektedir. Bu hibrit yaklaşım, kural tabanlı sistemin verimlilik ve kontrol avantajlarını LLM'nin esneklik ve anlama yetenekleriyle birleştirmektedir. Örneğin: Kullanıcı 'Bilgisayar Mühendisliği alanında en aktif araştırmacı kimdir?' sorusunu sorarsa, kural tabanlı sistem bunu yorumlayamadığı için Gemini'ye yönlendirilir ve Gemini, sistemin tamamını analiz ederek bağlamsal bir cevap sunar. Bu tasarım, yazılım sistemlerinde yapay zekâ paradigmalarının pragmatik kombinasyonunun bir örnekleridirdir ve kaynaklar sınırlı olan kurumsal ortamlarda sıklıkla kullanılan bir desendir.

2.4.6 Yaklaşımların Karşılaştırması: Kural Tabanlı, LLM, GenAI ve Hibrit Mimariler

Aşağıdaki tablo, bu çalışmada uygulanan dört yapay zekâ yaklaşımının özelliklerini ve kullanım alanlarını sistematik biçimde karşılaştırmaktadır. Her yaklaşımın meritlenmesi, belirli bağlamlar ve gereksinimler altında yapılmıştır; dolayısıyla hangi yaklaşımın 'en iyi' olduğu söylenemez — önemli olan, doğru bağlamda doğru yaklaşımı seçmektir.

Yaklaşım	Kullanım Alanı	Teknik Özellikler	Avantajlar	Sınırlılıklar
Kural Tabanlı	İstatistiksel sorgular (processChat)	if-then kuralları, anahtar kelime eşleştirmesi	Hızlı, öngörülebilir, transparent, API-bağımsız	Sınırlı esneklik, yeni sorgu için kod değişikliği gerekli
LLM Tabanlı (Claude Opus 4)	Yazılım geliştirme süreci, kod üretimi	Transformer mimarisi, çok adımlı akıl yürütme, Tools API	Yüksek bağlam anlayışı, çoklu dosya koordinasyonu	Hata riski, API maliyeti, bağlam sınırı
Üretken YZ (GenAI)	Kod, belge, rapor, test verisi üretimi	LLM tabanlı içerik üretimi, çok modlu çıktı	Hızlı prototipleme, iteratif geliştirme	İnsan doğrulama gerekli, tutarsızlık olabilir
Hibrit (Kural + LLM)	Dashboard chatbot asistanı	Kural tabanlı önce, başarısızlıkta Gemini API fallback	Verimlilik + esneklik, düşük maliyet	Tasarım karmaşıklığı, sınır durum testi gerekli

Tablonun dördüncü satırında açıklandığı üzere, bu çalışmanın temel özgünlüğü, aynı yazılım sisteminin farklı katmanlarında farklı yapay zekâ paradigmalarını bilinçli olarak kullanmasıdır. Geliştirme süreci LLM-yoğun (development: LLM-intensive), ürün arayüzü hibrit (UI: hybrid), iş mantığı ise deterministik-yoğun (business logic: rule-intensive) olarak tasarlanmıştır. Bu çeşitlilik, eğer-o eğer (if-then-if) bloğu biçiminde yapılan naif bir yaklaşımdan farklı olarak, her katmanın işlevsel gerekliliklerine ve performans kısıtlamalarına uygun optimum çözümlerin seçilmesiyle sağlanmıştır.

3. Teknoloji Yığını ve Mimari

Bu bölümde, Ar-Ge Yönetim Dashboard'ının geliştirilmesinde kullanılan teknolojiler, bileşen mimarisi ve veri modeli detaylı biçimde açıklanmaktadır. Teknoloji seçimleri; gerçek zamanlı çoklu kullanıcı desteği, hızlı prototipleme ve kolay dağıtım gereksinimlerine göre yapılmıştır (Google, 2024; Meta, 2024).

3.1 Kullanılan Teknolojiler

Katman	Teknoloji	Açıklama
Frontend	React 18	Hooks tabanlı SPA, fonksiyonel bileşenler
Stil	Tailwind CSS	Utility-first CSS çerçevesi, responsive tasarım
Build	Vite	ESM tabanlı hızlı bundler, HMR desteği
Veritabanı	Firebase Firestore v9	NoSQL bulut DB, gerçek zamanlı snapshot
Hosting	Firebase Hosting	CDN destekli küresel dağıtım
İkonlar	Lucide React	30+ ikon bileşeni (SVG tabanlı)
Kimlik Doğrulama	Özel Uygulama	Basit parola tabanlı, 4 rol seviyesi
YZ Ajanı	Claude Opus 4 (Anthropic)	LLM tabanlı AI agent, vibe coding

3.2 Bileşen Mimarisi

Uygulama, monolitik tek dosya mimarisi kullanmaktadır. Dashboard.jsx dosyası yaklaşık 7.650 satırdan oluşmakta olup tüm UI bileşenlerini, iş mantığını, state yönetimini ve Firestore senkronizasyonunu barındırmaktadır. Bu mimari tercih, vibe coding metodolojisinin doğal bir sonucudur: YZ ajanı tek dosya üzerinde çalışarak bağlamı kaybetmeden hızlı iterasyonlar gerçekleştirebilmektedir. Aşağıda dosya yapısı özetlenmektedir:

Dosya	Satır	İşlev
Dashboard.jsx	~7.650	Ana bileşen: UI + state + Firestore sync
firebase.js	~30	Firebase konfigürasyonu, memoryLocalCache
AuthContext.jsx	~60	Kimlik doğrulama: 4 rol yönetimi
LoginPage.jsx	~80	Giriş sayfası: kullanıcı adı + şifre
App.jsx	~25	Rota yönetimi: auth kontrolü
main.jsx	~10	Uygulama giriş noktası: React DOM

3.3 Firestore Veri Modeli

Tüm veriler Firestore'da 'arge' koleksiyonu altında 11 doküman halinde tutulmaktadır. Her doküman, belirli bir veri tipini temsil etmekte ve items veya data alanı içermektedir. Bu yapı, Firestore'un doküman boyutu sınırlamalarına uygun olup aynı zamanda onSnapshot ile tek seferde tüm koleksiyonun dinlenmesine olanak tanımaktadır.

Doküman	Yapı	Açıklama
researchers	{ items: [...] }	Tüm araştırmacı kayıtları
topics	{ items: [...] }	Araştırma konuları
projects	{ items: [...] }	Projeler ve bağlı konular
quicklinks	{ items: [...] }	Hızlı erişim bağlantıları
cfg_roles	{ data: {...} }	Araştırmacı rolleri
cfg_statuses	{ data: {...} }	Konu/proje durum tanımları
cfg_priorities	{ data: {...} }	Öncelik seviyeleri
cfg_ptypes	{ items: [...] }	Proje türleri
cfg_categories	{ items: [...] }	Konu kategorileri
cfg_degrees	{ items: [...] }	Akademik dereceler
cfg_edustatus	{ items: [...] }	Eğitim durumları

3.4 Varlık İlişki Modeli

Sistemdeki üç ana varlık (araştırmacı, konu, proje) arasında çok-çok (many-to-many) ilişkiler bulunmaktadır. Bu ilişkiler, her varlığın kendi dokümanı içinde referans dizileri ile temsil edilmektedir. İlişki modeli, çift yönlü senkronizasyon mekanizmasının temelini oluşturmaktadır.

Araştırmacı ↔ Konu: Bir konu birden fazla araştırmacıya sahip olabilmektedir. Her ilişki rol ve isIdeaOwner (fikir sahibi) bayrağı taşımaktadır. Araştırmacı bir konudan çıkarıldığında, bağlı projelerdeki varlığı da gözden geçirilmektedir.

Konu ↔ Proje: Bir proje en az bir konuya bağlı olmak zorundadır; bu, sistemin temel iş kurallarından biridir. Bir konu birden fazla projede yer alabilmekte ve projeye konudan gelen araştırmacılar otomatik olarak eklenmektedir.

Araştırmacı ↔ Proje: Araştırmacılar projeye hem doğrudan hem de bağlı konular üzerinden katılabilmektedir. Konudan gelen araştırmacılar, konu-proje ilişkisi kesildiğinde otomatik olarak temizlenmektedir.

4. Uygulama Bileşen Ağacı ve Özellikler

Bu bölümde, Dashboard uygulamasının bileşen hiyerarşisi ağaç yapısı olarak sunulmakta ve her bir bileşenin işlevi açıklanmaktadır. Uygulama, 20'den fazla özelleştirilmiş React bileşeninden oluşmakta olup tamamı Dashboard.jsx dosyası içinde tanımlanmıştır.

4.1 Bileşen Hiyerarşisi (Ağaç Yapısı)

Aşağıdaki şema, uygulamadaki tüm bileşenleri ve aralarındaki hiyerarşik ilişkiyi göstermektedir. Kök bileşen olan ArGeDashboard, tüm state yönetimini ve Firestore senkronizasyonunu üstlenmekte; alt bileşenler ise belirli UI ve iş mantığı sorumluluklarını taşımaktadır.

ArGeDashboard (Ana Bileşen — ~7.650 satır)
├── **Header Bar** (Üst menü, arama, durum göstergeleri)
│ ├── Firestore Bağlantı Durumu (yeşil/kırmızı gösterge)
│ ├── Kayıt Durumu (kaydediliyor/kaydedildi)
│ ├── DeadlinePanel (zil ikonu — yaklaşan son tarihler)
│ ├── CalendarModal (takvim görünümü — proje zaman çizelgesi)
│ ├── QuickLinksPanel (hızlı erişim bağlantıları)
│ ├── Çevrimiçi Kullanıcılar (Google Docs tarzı avatar listesi)
│ └── Profil / Çıkış
├── **Sol Sütun: Araştırmacılar**
│ ├── Filtre Çubuğu (kurum, AÖF üyelik, gelişmiş filtreler)
│ ├── ResearcherCard (kişi kartı — sürüklenebilir)
│ │ ├── Ad-soyad, unvan, eğitim durumu
│ │ ├── İlgi alanları (etiketler)
│ │ ├── Bağlı konu/proje sayıları
│ │ └── Proje türü istatistikleri (badge)
│ └── ResearcherDetailModal (kişi detay — düzenleme)
│ ├── Kişisel bilgiler formu
│ ├── Eğitim bilgileri
│ ├── Bağlı konular/projeler listesi
│ └── Silme onayı
├── **Orta Sütun: Konular**
│ ├── Filtre Çubuğu (durum, öncelik, gelişmiş filtreler)
│ ├── TopicCard (konu kartı — sürükle-bırak hedefi)
│ │ ├── Başlık, kategori, durum etiketi
│ │ ├── Öncelik renk kodu
│ │ ├── Araştırmacı avatarları
│ │ └── Fikir sahibi göstergesi (□)
│ └── DetailModal (konu detay — düzenleme)
│ ├── Temel bilgiler sekmesi
│ ├── Araştırmacı/Rol sekmesi (rol atama, fikir sahibi)
│ └── Görevler sekmesi (görev ekleme/tamamlama)
├── **Sağ Sütun: Projeler**
│ ├── Filtre Çubuğu (durum, öncelik, gelişmiş filtreler)
│ ├── ProjectCard (proje kartı)
│ │ ├── Proje adı, türü, durumu
│ │ ├── Bağlı konu etiketleri
│ │ ├── Bütçe ve tarih bilgisi

│ │ └── Araştırmacı avatarları + fikir sahipleri
│ └── DetailModal (proje detay — düzenleme)
│ ├── Temel bilgiler + bağlı konular
│ ├── Araştırmacı/Rol sekmesi
│ └── Görevler sekmesi
├── **Modallar**
│ ├── AddItemModal (araştırmacı/konu/proje ekleme formu)
│ ├── SettingsModal (yapılandırma yönetimi)
│ │ ├── Roller, Durumlar, Öncelikler
│ │ ├── Kategoriler, Proje Türleri
│ │ └── Akademik Dereceler, Eğitim Durumları
│ ├── StatsModal (istatistik ve grafikler — 6 sekme)
│ │ ├── Özet sekmesi (genel istatistik, pasta/çubuk grafik)
│ │ ├── Araştırmacı İst. sekmesi (unvan dağılımı, proje türü)
│ │ ├── Kişi Bazlı Rapor sekmesi (araştırmacı detay)
│ │ ├── Zaman İst. sekmesi (yıl/ay bazlı trend)
│ │ ├── Konu Bazlı sekmesi (durum dağılımı, proje türü)
│ │ └── Proje Bazlı sekmesi (uluslararası, ülke/kurum analizi)
│ ├── LeaderboardModal (sıralama tablosu)
│ ├── TableViewModal (tablo görünümü + CSV dışa aktarım)
│ └── CalendarModal (takvim — proje zaman çizelgesi)
├── **Yardımcı Bileşenler**
│ ├── Badge, ProgressBar, Avatar, InfoRow
│ ├── FilterDropdown, RoleSelectPopup
│ ├── TaskItem, Toast
│ ├── SimplePieChart, SimpleBarChart, SimpleLineChart
│ └── ArGeChatbot (kural tabanlı sohbet asistanı)
└── **Yedekleme Paneli** (yalnızca master)
├── JSON indirme / yükleme
├── Firestore yedek alma
└── 30 gün uyarı banner'ı

4.2 Kart Bileşenleri: Oluşturma, Düzenleme ve Silme

Her varlık türü (araştırmacı, konu, proje) için özelleştirilmiş kart bileşeni tasarlanmıştır. Kartlar, ilgili varlığın en önemli bilgilerini kompakt bir formatta sunmakta ve CRUD (oluşturma, okuma, güncelleme, silme) işlemlerini desteklemektedir.

4.2.1 Kişi Kartı (ResearcherCard)

Her araştırmacı için kompakt bir kart görüntülenmektedir. Kart üzerinde ad-soyad, akademik unvan, eğitim durumu, ilgi alanları (etiketler hâlinde), bağlı konu/proje sayıları ve proje türü bazlı istatistikler yer almaktadır. Kartın sol kenarında sürükleme tutamağı bulunmakta olup bu tutamak ile araştırmacı konu kartlarının üzerine sürüklenebilmektedir.

Projelendirilmiş Konular ve Proje Türü Filtreleme: Proje türü (projectType) bilgisi artık sadece projelendirilmiş konularda (bir projeye bağlı konularda) gösterilir ve hesaplanır. Projelendirilmemiş konular taslak fikir olarak kabul edilir ve proje türü istatistiklerine dahil edilmez. ResearcherCard'da klasör ikonu (FolderKanban) ile projelendirilmiş konu sayısı gösterilir (örn: □ 1p), ardından tür bazında dağılım (örn: TÜBİTAK ×1) badge'leri görüntülenir.

İşlem	Tetikleyici	Sonuç
Oluşturma	'+' butonu → AddItemModal	Yeni araştırmacı kaydı oluşturulur
Görüntüleme	Kart tıklama	ResearcherDetailModal açılır
Düzenleme	Modal içi form alanları	Bilgiler güncellenir → Firestore'a yazılır
Silme	Modal içi 'Sil' butonu	Onay sonrası araştırmacı silinir
Sürükle-Bırak	Kart → Konu kartı üzerine	Araştırmacı konuya (ve projeye) eklenir

4.2.2 Konu Kartı (TopicCard)

Her araştırma konusu için renkli kart görüntülenmektedir. Kartın üst kısmında öncelik renk kodu (kırmızı: acil, sarı: yüksek, mavi: normal, gri: düşük) ile durum etiketi gösterilmektedir. Orta kısımda konu başlığı ve kategori etiketi, alt kısımda ise araştırmacı avatarları ve fikir sahibi göstergesi yer almaktadır. Konu kartları, sürükle-bırak hedefi olarak da işlev görmektedir.

İşlem	Tetikleyici	Sonuç
Oluşturma	'+' butonu → AddItemModal	Yeni konu oluşturulur
Düzenleme	Kart tıklama → DetailModal	Sekmeli düzenleme arayüzü açılır
Rol Atama	DetailModal → Roller sekmesi	Araştırmacıya rol ve fikir sahibi atanır
Görev Yönetimi	DetailModal → Görevler sekmesi	Görev eklenir, tamamlanır, silinir
Proje Oluşturma	Konu kartı → 'Projelendir'	Konudan yeni proje türetilir

4.2.3 Proje Kartı (ProjectCard)

Proje kartları konu kartlarına benzer yapıda olup ek bilgiler içermektedir: proje türü, başlangıç/bitiş tarihi, bütçe bilgisi ve bağlı konu sayısı. Projenin genel durumu renk koduyla, fikir sahipleri ise özel ikon ile belirtilmektedir. Proje kartlarında ayrıca 'İptal Et' butonu bulunmakta olup bu buton ile proje tamamlanmadan sonlandırılabilmektedir.

İşlem	Tetikleyici	Sonuç
Oluşturma	'+' butonu → AddItemModal	En az 1 konu seçilerek proje oluşturulur
Düzenleme	Kart tıklama → DetailModal	Proje bilgileri, bağlı konular düzenlenir
Konu Ekleme	DetailModal → Konu seçici	Projeye konu bağlanır + araştırmacılar senkronize edilir
Konu Çıkarma	DetailModal → 'Çıkar' butonu	Konu projeden çıkar; özgü araştırmacılar temizlenir
İptal Etme	Kart → 'İptal' butonu	Proje durumu 'iptal' olarak güncellenir

4.3 İstatistik ve Analitik Modülü (StatsModal)

İstatistik modalı altı sekmeden oluşmakta olup her sekme farklı bir analiz perspektifi sunmaktadır. Modal başında beş adet filtre bulunmaktadır: araştırmacı, durum, proje türü, yıl ve AÖF üyelik filtresi. Filtreler uygulandığında tüm sekmelerdeki istatistikler otomatik olarak güncellenmektedir. Grafikler SimplePieChart, SimpleBarChart ve SimpleLineChart bileşenleri ile oluşturulmaktadır.

Proje Türü Filtre Mantığı: Tüm istatistik hesaplamalarında, proje türü metrikleri yalnızca projelendirilmiş konuları (bir projeye bağlı konuları) dikkate almaktadır. Projelendirilmemiş konular (taslak fikirleri) proje türü dağılımları, sayımları ve oranlarından hariç tutulur.

Sekme	Açıklama
Özet	Genel sayılar, durum dağılımları, araştırmacı istatistikleri, görev tamamlama, unvan dağılımı
Araştırmacı İst.	Konulardaki/projelerdeki araştırmacı sayıları, unvan dağılımı, aktivite tablosu
Kişi Bazlı Rapor	Seçilen araştırmacının konuları, projeleri, rolleri, görev tamamlama oranı
Zaman İstatistikleri	Yıl/ay bazlı konu ve proje dağılımı, trend çizgi grafikleri, detay tablosu
Konu Bazlı	Konu durumu dağılımı, aylık başlangıç/bitiş grafikleri, detay tablosu
Proje Bazlı	Uluslararası ortaklık, proje türü dağılımı, ülke/kurum analizi, drill-down

4.3.1 Özet Sekmesi

Özet sekmesi, sistemdeki tüm verilerin kuşbakışı görünümünü sunmaktadır. Dört adet genel sayı kartında toplam araştırmacı, konu, proje ve toplam bütçe (₺) gösterilmektedir. Konu durumu dağılımı (önerilen, aktif, tamamlanan) ve proje durumu dağılımı ayrı kart gruplarıyla sunulmaktadır. Araştırmacı istatistikleri bölümünde toplam kişi, AÖF üyesi, PI deneyimli, konusu olan/olmayan ve fikir sahibi sayıları 6 kartlık ızgara düzeninde görüntülenmektedir. Proje türü dağılımı dinamik kartlarla, görev tamamlama oranı ilerleme çubuğuyla ve unvan dağılımı (Prof.Dr., Doç.Dr., Dr.Öğr.Üyesi vb.) çubuk grafik ile gösterilmektedir. İki adet pasta grafik konu ve proje durumlarının görsel dağılımını sunmaktadır.

4.3.2 Araştırmacı İstatistikleri Sekmesi

Bu sekmede araştırmacıların konular ve projelerdeki dağılımı detaylı biçimde sunulmaktadır. Konulardaki araştırmacılar bölümünde önerilen, aktif ve tamamlanan konulardaki benzersiz araştırmacı sayıları ayrı kartlarda gösterilmektedir. Projelerdeki araştırmacılar bölümü aynı yapıyı proje bazlı sunmaktadır. Unvan dağılımı çubuk grafik ile görselleştirilmektedir. Araştırmacı Aktivite Tablosu'nda her araştırmacı için konu ve projelerdeki önerilen/aktif/tamamlanan sayıları ile görev tamamlama oranı satır satır listelenmektedir.

4.3.3 Kişi Bazlı Rapor Sekmesi

Dropdown menüden seçilen tek bir araştırmacının detaylı profili sunulmaktadır. Kişinin ismi, unvanı, kurumu ve birimi başlık alanında görüntülenmektedir. Konu kartlarında önerilen, aktif ve tamamlanan konu sayıları; proje kartlarında aynı bilgiler proje bazlı gösterilmektedir. Görev tamamlama oranı ilerleme çubuğuyla yüzdelik olarak sunulmaktadır. Rol dağılımı bölümünde kişinin farklı konulardaki rolleri renkli badge'ler ile listelenmiştir. Son bölümlerde kişinin tüm konuları (durum, başlık, rol) ve tüm projeleri (durum, başlık, tür) tablo halinde sunulmaktadır.

4.3.4 Zaman İstatistikleri Sekmesi

İki görünüm modu sunulmaktadır: yıl bazlı ve ay bazlı. Yıl bazlı görünümde sistemdeki tüm yıllar için konu ve proje dağılımları çubuk grafiklerle gösterilmektedir. Ay bazlı görünümde seçilen yılın 12 ayı için aynı metrikler hesaplanmaktadır. Her görünümde altı adet çubuk grafik (önerilen/aktif/tamamlanan × konu/proje) ve iki adet trend çizgi grafiği (toplam konu ve toplam proje trendi) yer almaktadır. Detay tablosunda yıl/ay bazlı tüm sayılar satır satır sunulmaktadır.

4.3.5 Konu Bazlı Sekmesi

Konu durumu dağılımını pasta grafikle, konuların aylık başlangıç ve bitiş tarihlerini çubuk grafiklerle görselleştirmektedir. Detay tablosunda her durum için konu sayısı ve yüzdelik oranı sunulmaktadır.

4.3.6 Proje Bazlı Sekmesi

En kapsamlı sekmedir. Uluslararası ortaklı projeler bölümünde yurt dışı ortağı olan projeler tespit edilerek toplam/önerilen/aktif/tamamlanan sayıları ulusal projelerle karşılaştırılmaktadır. Proje durumu pasta grafiği ve proje türü çubuk grafiği temel dağılımları göstermektedir. Projelerin aylık başlangıç dağılımı çizgi grafikle, detay tablosunda durum/sayı/yüzde/bütçe bilgileri sunulmaktadır.

Ülke ve Kurum Dağılımı alt bölümü, projelerin coğrafi ve kurumsal analizini sunmaktadır. Dört özet kartında tekil ülke sayısı, ülke bilgili proje sayısı, tekil kurum sayısı ve işbirliği ülke sayısı gösterilmektedir. Yürütücü ülke ve ortak ülke dağılımları pasta ve çubuk grafiklerle, durum bazlı ülke dağılımı üç ayrı pasta grafikle (önerilen/aktif/tamamlanan) görselleştirilmektedir. Ülke Detay Tablosu'nda her ülke için önerilen/aktif/tamamlanan proje sayıları, yürütücü ve ortak toplamları satır satır sunulmaktadır. Aynı yapı kurum bazlı olarak da tekrarlanmaktadır: yürütücü kurum dağılımı, ortak kurum dağılımı, durum bazlı kurum dağılımı ve kurum detay tablosu. Kurum bazlı drill-down özelliği ile seçilen bir kurumun toplam proje sayısı, durum kartları, ülke dağılımı ve ortak kurum dağılımı detaylı biçimde incelenebilmektedir.

4.4 Özet İstatistik Paneli

Ana sayfanın araştırmacı sütununda genişletilebilir/daraltılabilir bir istatistik paneli bulunmaktadır. Bu panel, 6 kartlık ızgara düzeninde aşağıdaki metrikleri sunmaktadır:

Metrik	Açıklama
Toplam Kişi	Sistemde kayıtlı tüm araştırmacı sayısı
AÖF Üyesi	Açıköğretim Fakültesi mensubu araştırmacı sayısı
PI Deneyimli	Proje yürütücülüğü (PI) deneyimi olan araştırmacı sayısı
Konusu Olan	En az bir araştırma konusuna atanmış araştırmacı sayısı
Konusu Olmayan	Henüz herhangi bir konuya atanmamış araştırmacı sayısı
Fikir Sahibi	En az bir konuda veya projede fikir sahibi olarak işaretlenmiş araştırmacı sayısı

Proje türü dağılımı da ayrı bir alt bölümde dinamik ızgara olarak gösterilmektedir. Her proje türü için adet bilgisi renkli kartlar ile sunulmaktadır. Bu dağılımda yer alan sayılar yalnızca projelendirilmiş konulara ait proje türlerini temsil eder; taslak konular/fikirler bu istatistiklere dahil edilmez.

4.5 Diğer Uygulama Özellikleri

4.5.1 Sürükle-Bırak (Drag & Drop) Sistemi

HTML5 Drag and Drop API'si üzerine inşa edilen bu sistem, araştırmacı kartlarının konu kartları üzerine sürüklenmesine olanak tanımaktadır. Bırakma gerçekleştiğinde RoleSelectPopup açılmakta; bu popup'ta araştırmacının konudaki rolü ve fikir sahibi olup olmadığı belirlenmektedir. Seçim tamamlandığında araştırmacı hem konuya hem de bağlı projelere otomatik olarak eklenmektedir.

4.5.2 Tablo Görünümü (TableViewModal)

Kart görünümüne alternatif olarak tablo biçiminde veri görüntüleme seçeneği sunulmaktadır. Üç sekmeden oluşmaktadır (araştırmacılar, konular, projeler). Tabloda sıralama, filtreleme ve CSV dışa aktarım özellikleri bulunmaktadır. Konu tablosunda 'Fikir Sahibi' sütunu yer almaktadır.

4.5.3 Sıralama Tablosu (LeaderboardModal)

Araştırmacıları çeşitli kriterlere göre sıralayan ve puanlayan bu modal, konu sayısı, proje katılımı ve görev tamamlama oranı gibi metrikleri hesaplayarak bir liderlik tablosu oluşturmaktadır. AÖF üyelik filtresi ve sıralama kriteri seçimi desteklenmektedir.

4.5.4 Takvim (CalendarModal)

Proje zaman çizelgelerini aylık takvim görünümünde sunan bu modal, yaklaşan son tarihleri vurgulamakta ve proje başlangıç/bitiş tarihlerini görsel olarak işaretlemektedir. Tarihe tıklandığında ilgili projenin detaylarına erişilebilmektedir.

4.5.5 Hızlı Erişim Bağlantıları (QuickLinksPanel)

Yöneticilerin sık başvurduğu dış kaynaklara (TÜBİTAK portalı, YÖK veritabanı vb.) tek tıkla erişim sağlayan panel, Firestore'da saklanmakta ve tüm kullanıcılar tarafından görüntülenebilmektedir. Bağlantılar eklenebilmekte, düzenlenebilmekte ve silinebilmektedir.

4.5.6 Son Tarih Paneli (DeadlinePanel)

Zil ikonuna tıklandığında açılan dropdown menü, yaklaşan proje son tarihlerini listelemektedir. Tarih hesaplaması otomatik olarak yapılmakta ve kalan gün sayısına göre renk kodu atanmaktadır (kırmızı: acil, sarı: yakın, yeşil: uzak).

4.5.7 Kural Tabanlı Chatbot (ArGeChatbot)

Dashboard bünyesinde basit bir kural tabanlı sohbet asistanı bulunmaktadır. Bu asistan, araştırma konusu önerisi ve araştırmacı eşleştirmesi gibi görevlerde kullanıcıya yardımcı olmaktadır. LLM tabanlı değil, deterministik kurallarla çalışmaktadır.

4.5.8 Ayarlar Paneli (SettingsModal)

Yapılandırılabilir parametrelerin yönetildiği bu panel; roller, konu/proje durumları, öncelik seviyeleri, konu kategorileri, proje türleri, akademik dereceler ve eğitim durumlarını kapsamaktadır. Her yapılandırma öğesi eklenebilmekte, düzenlenebilmekte ve silinebilmektedir. Değişiklikler Firestore'a anında yansıtılmakta ve tüm bağlı istemcilerde gerçek zamanlı olarak güncellenmektedir.

4.6 Yetkilendirme ve Rol Sistemi

Sistem, dört kademeli bir yetkilendirme modeli uygulamaktadır. Her kullanıcı rolü, belirli işlemleri gerçekleştirme yetkisine sahiptir. Bu model, kurumsal hiyerarşiyi yansıtacak şekilde tasarlanmıştır.

Rol	Yetkiler
master	Tüm işlemler + yedekleme + sistem ayarları + zorunlu yayınla
admin	Tüm CRUD işlemleri + ayar düzenleme + manuel senkronizasyon
editor	Ekleme ve düzenleme yetkileri, silme kısıtlı
user	Yalnızca görüntüleme (salt okunur erişim)

5. Senkronizasyon Mimarisi

Senkronizasyon, bu projenin en kritik teknik bileşenidir. Birden fazla yöneticinin aynı anda sistemi kullanabilmesi, veri çakışmalarının önlenmesi ve değişikliklerin anlık olarak tüm istemcilere yansıtılması gerekmektedir (Google, 2024). Bu bölümde, Firebase Firestore tabanlı gerçek zamanlı senkronizasyon mekanizması detaylandırılmaktadır.

5.1 Okuma Mekanizması (onSnapshot)

Her 11 doküman için bir onSnapshot dinleyicisi çalışmaktadır. Gelen snapshot'taki JSON, lastJson.current referansı ile karşılaştırılmakta; farklıysa state güncellenmekte, aynıysa işlem atlanmaktadır. Bu karşılaştırma mekanizması, sonsuz yaz-oku döngüsünü engellemenin temel anahtarıdır. lastJson.current referansı, hem yazma hem okuma tarafında güncellenerek iki yönlü koruma sağlamaktadır.

5.2 Yazma Mekanizması (Debounced setDoc)

State değiştiğinde 500ms debounce süresi sonrasında Firestore'a setDoc ile yazılmaktadır. Yazma öncesinde lastJson.current güncellenerek onSnapshot'ın kendi yazdığı veriyi yeniden işlemesi engellenmektedir. Debounce mekanizması, ardışık düzenlemelerde gereksiz yazma operasyonlarını birleştirerek Firestore kota kullanımını optimize etmektedir.

5.3 Çapraz Varlık Senkronizasyonu

Konu, proje ve araştırmacı varlıkları arasında çift yönlü senkronizasyon sağlanmıştır. handleUpdateItem fonksiyonu, bir konu güncellenirken eski ve yeni araştırmacı listelerini karşılaştırarak eklenen ve çıkarılan araştırmacıları tespit etmektedir. Çapraz senkronizasyonun doğruluğu, 9 farklı kontrol noktası üzerinden denetlenmiş ve tüm noktalar güvenli olarak teyit edilmiştir.

5.4 Çevrimiçi Kullanıcı Takibi (Presence)

Google Docs tarzı bir canlı gösterge sistemi uygulanmıştır. Her oturum açık kullanıcı, Firestore'daki presence dokümanına kaydedilmekte ve tüm istemcilerde renkli avatarlar ile görüntülenmektedir. 6 farklı renk paleti kullanılarak eş zamanlı çalışan yöneticiler birbirinden ayırt edilebilmektedir.

5.5 Özel Firestore Konfigürasyonu

experimentalForceLongPolling: true — Kurumsal ağlarda güvenlik duvarları tarafından engellenen WebSocket bağlantıları yerine HTTP long polling kullanılmaktadır.

memoryLocalCache() — IndexedDB tabanlı önbelleğin neden olduğu kuyruk sorunları ortadan kaldırılmaktadır.

6. Yapay Zekâ Destekli Geliştirme Süreci

Bu bölüm, projenin geliştirilmesinde kullanılan yapay zekâ destekli metodolojileri, insan-YZ işbirliğinin somut uygulamalarını ve bu süreçten elde edilen çıkarımları detaylandırmaktadır (Anthropic, 2025).

6.1 İnsan + YZ Kodlama Akış Diyagramı

Aşağıda, her bir özelliğin geliştirilmesinde izlenen iteratif İnsan + YZ kodlama süreci adım adım gösterilmektedir. Bu akış, çevik yazılım geliştirme metodolojileri ile benzerlikler taşımakta; ancak YZ ajanının anlık kod üretme kapasitesi sayesinde çok daha hızlı iterasyonlara olanak tanımaktadır (Beck vd., 2001; Karpathy, 2025).

ADIM 1 — Gereksinim Bildirimi (İnsan)

İnsan geliştirici, istediği özelliği Türkçe doğal dilde açıklamaktadır.

Örnek: 'Kişi kartlarına proje türü istatistiklerini ekle'

↓

ADIM 2 — Kod Analizi ve Plan (YZ Ajanı)

YZ ajanı mevcut kodu okumakta, etkilenen dosyaları belirlemekte

ve değişiklik planını sunmaktadır.

Çıktı: Etkilenen bileşenler, eklenmesi/değiştirilmesi gereken satırlar

↓

ADIM 3 — Kod Üretimi (YZ Ajanı)

YZ ajanı planı uygulayarak kod değişikliklerini gerçekleştirmektedir.

React bileşenleri, Firestore entegrasyonu, Tailwind CSS stilleri üretilmektedir.

↓

ADIM 4 — Doğrulama (YZ Ajanı)

Üretilen kod Babel parser ile ayrıştırılarak söz dizimi doğrulanmaktadır.

Çıktı: 'PARSE OK' veya hata detayı

↓

ADIM 5 — İnceleme ve Test (İnsan)

İnsan geliştirici kodu incelemekte, tarayıcıda test etmekte

ve geri bildirim vermektedir.

↓ (Düzeltme gerekiyorsa ADIM 1'e dönülmektedir)

ADIM 6 — Onay ve Dağıtım (İnsan + YZ)

Özellik onaylandığında Firebase Hosting'e dağıtılmaktadır.

Rapor otomatik olarak güncellenmektedir.

6.2 İnsan ve YZ Rol Dağılımı

Görev Alanı	İnsan Geliştiricinin Rolü	YZ Ajanının Rolü
Gereksinim	Özelliği doğal dilde tanımlamak	Gereksinimleri koda dönüştürmek
Mimari	Teknoloji seçimi, veri modeli kararları	Mevcut kodu analiz edip plan sunmak

Görev Alanı	İnsan Geliştiricinin Rolü	YZ Ajanının Rolü
Kod Üretimi	Üretilen kodu incelemek, onaylamak	React, Firestore, CSS kodlarını yazmak
Hata Ayıklama	Sorunları raporlamak, test etmek	Kök neden analizi ve çözüm uygulamak
UI/UX	Renk, düzen, etkileşim kararları	Tailwind CSS ile uygulamak
Kalite Kontrol	Tarayıcıda son kullanıcı testi	Kod auditi, syntax doğrulama
Dokümantasyon	İçerik ve dil kontrolü	Rapor yazımı (Text, DOCX, PDF)

6.3 YZ Kullanım Alanları Tablosu

Alan	YZ Türü	Açıklama
Kod Üretimi	LLM Agent (Claude)	React, Firestore, Tailwind kodları üretilmiştir
Hata Ayıklama	LLM Agent (Claude)	Closure bug, sync döngüsü çözümlenmiştir
Kod Denetimi	LLM Agent (Claude)	Çapraz senkronizasyon auditi yapılmıştır
UI Tasarımı	LLM (Vibe Coding)	Kart, modal, tablo bileşenleri tasarlanmıştır
Dokümantasyon	LLM Agent (Claude)	Bu rapor YZ desteğiyle hazırlanmıştır
İş Mantığı	Kural Tabanlı	Validasyonlar, iş kuralları deterministiktir
Auth Sistemi	Kural Tabanlı	Şifre kontrolü, rol atama kurallarla çalışmaktadır
Chatbot	Kural Tabanlı	Sohbet asistanı if-then kurallarıyla çalışmaktadır

7. Kritik Hata Çözümleri ve Teknik Dersler

Geliştirme sürecinde karşılaşılan kritik hatalar, hem sistemin güvenilirliğini test etmiş hem de yapay zekâ destekli geliştirme sürecinin hata ayıklama kapasitesini somut biçimde ortaya koymuştur. Bu bölümde temel hatalar, kök neden analizleri ve çözüm yaklaşımları sunulmaktadır.

7.1 Initial Push Closure Bug'ı (Veri Kaybı)

Bu hata, projenin en kritik ve öğretici teknik sorunudur. JavaScript'in closure mekanizmasının React hooks ile etkileşiminden kaynaklanan bu hata, üretim ortamında gerçek veri kaybına yol açmıştır. useEffect(() => {...}, []) içindeki markReady callback'i, JavaScript closure mekanizması gereği mount anındaki boş state değerlerini yakalamıştır. Bu değerler Firestore'a yazılınca gerçek verinin üzerine geçmiştir. Çözüm olarak tüm initial push kodu kaldırılmıştır; onSnapshot dinleyicisi zaten doküman yoksa varsayılan değerleri yazmaktaydı.

7.2 Sonsuz Yaz-Oku Döngüsü

onSnapshot ile okunan verinin state'e yazılması, state değişikliğinin debounced write'ı tetiklemesi ve yazılan verinin onSnapshot'ı tekrar tetiklemesi şeklinde sonsuz bir döngü oluşmuştur. lastJson.current karşılaştırma patterni ile onSnapshot'ın kendi yazdığı veriyi tekrar işlemesi engellenmiştir. Bu pattern, gerçek zamanlı sistemlerde sıkça karşılaşılan yapısal bir sorunun çözümünü temsil etmektedir.

7.3 handleUpdateItem Yeniden Yazımı

Çapraz varlık senkronizasyonundaki eksiklikler, bir konudan çıkarılan araştırmacının bağlı projelerde kalmaya devam etmesi sorununa yol açmıştır. Eski ve yeni konu araştırmacı listeleri karşılaştırılarak tam yeniden yazım gerçekleştirilmiştir. Bu hata, ilişkisel veri modellerinde kaskad güncellemenin önemini somut biçimde ortaya koymuştur.

7.4 Firestore Bağlantı Sorunları

Sorun	Kök Neden	Çözüm
Kota Aşımı	Spark planı okuma/yazma limitleri	Firebase Blaze planına yükseltilmiştir
IndexedDB Kuyruk	Varsayılan önbellek birikmesi	memoryLocalCache() uygulanmıştır
WebSocket Engeli	Üniversite güvenlik duvarı	experimentalForceLongPolling uygulanmıştır

8. İş Kuralları, Yedekleme ve Kalite Güvencesi

8.1 Temel İş Kuralları

Proje-Konu Zorunlu İlişki: Her projenin en az bir konuya bağlı olması zorunludur. Projenin tek konusu çıkarılamamakta; bunun yerine 'Önce projeyi iptal edin' uyarısı gösterilmektedir.

Araştırmacı Temizlik Kuralları: Konu projeden çıkarıldığında, o konuya özgü araştırmacılar projeden otomatik olarak temizlenmektedir. Başka bağlı konularda bulunan araştırmacılar korunmaktadır.

Fikir Sahibi Takibi: Her konu ve projede bir veya daha fazla araştırmacı 'fikir sahibi' olarak işaretlenebilmektedir. Bu bilgi, konu-proje senkronizasyonuna dâhildir ve otomatik olarak taşınmaktadır.

8.2 Yedekleme Sistemi (Üç Katmanlı)

Katman 1 — JSON İndir/Yükle: Tüm veriler JSON formatında yerel bilgisayara indirilebilmektedir. Bu yedek, çevrimdışı veri kurtarma imkânı sağlamaktadır.

Katman 2 — Firestore Yedekleme: Veriler Firestore'da ayrı yedek koleksiyonuna kopyalanmaktadır (yalnızca master rolü).

Katman 3 — Otomatik Hatırlatma: 30 günden uzun süre yedekleme yapılmamışsa sarı uyarı banner'ı gösterilmektedir.

8.3 Doğrulama ve Kalite Güvencesi

Veri kaybı riskinin sistematik biçimde denetlenmesi amacıyla kapsamlı bir kod auditi gerçekleştirilmiştir. Bu audit, yapay zekâ ajanı tarafından 9 farklı kontrol noktası üzerinden yürütülmüş ve tüm noktalar güvenli olarak teyit edilmiştir.

#	Kontrol Noktası	Durum	Sonuç
1	Initial Push kodu	Kaldırılmıştır	GÜVENLİ
2	markReady fonksiyonu	Salt okunur	GÜVENLİ
3	onSnapshot dinleyicisi	lastJson kontrolü	GÜVENLİ
4	Debounced write	500ms gecikme	GÜVENLİ
5	handleUpdateItem	Tam yeniden yazım	GÜVENLİ
6	handleDeleteTopic	Temizlik eklenmiştir	GÜVENLİ
7	handleRemoveTopicFromProject	Temizlik eklenmiştir	GÜVENLİ
8	handleRoleSelect	Proje sync eklenmiştir	GÜVENLİ
9	setState çağrıları	Boş dizi yazımı yoktur	GÜVENLİ

9. Sonuç ve Tartışma

9.1 Gerçekleştirilen Çalışmanın Kapsamlı Özet Sonuçları

Bu çalışmada, Anadolu Üniversitesi Açıköğretim Fakültesi'nin, geleneksel Ar-Ge yönetim sorunlarıyla karşı karşıya olan araştırma birimi için, merkezi, bulut tabanlı, gerçek zamanlı ve ölçeklenebilir bir web tabanlı yönetim dashboard'ı geliştirilmiştir. Sistem tasarlanmasında, araştırmacı katalog yönetimi, araştırma konularının ve proje portföyünün merkezi takibi, birden fazla yöneticinin eş zamanlı erişim ve düzenleme kapasitesi, çapraz varlık ilişkilerinin tutarlılığı ve bütünlüğü, çok katmanlı veri koruma ve yedekleme mekanizmaları, ve kullanıcı düzeyinde iş akışlarının analitik takibi gibi çok çeşitli kurumsal gereksinimleri karşılanmıştır. Teknik implementasyon açısından, uygulama yaklaşık 7.650 satır React (v18) fonksiyonel bileşen kodu içermekte olup, 20'den fazla ana ve yan bileşenden, 11 Firestore NoSQL doküman koleksiyonundan, 4 kademeli yetkilendirme ve rol yönetimi modelinden ve gerçek zamanlı snapshot dinleme mekanizmalarından oluşmaktadır. Sistem, tarayıcı önbelleğe alma (offline mode), otomatik veri senkronizasyonu ve çakışma çözme mekanizmaları da içermektedir, bu da yüksek erişilebilirlik ve dayanıklılık sağlamaktadır.

Projenin bilim ve teknoloji açısından en özgün ve değerli boyutu, geliştirme sürecinin tamamında yapay zekâ ajanlarının, geleneksel yazılım mühendisliği rollerine eşdeğer katılımcı olarak yer almasıdır. Anthropic'in Claude Opus 4 modeli; başlangıç mimarisinin tasarlanmasından, bileşen düzeyinde kod üretimi, karmaşık hata ayıklama operasyonları (closure kapanış sorunları, useState ve useEffect zamanlama hataları gibi), çapraz modül senkronizasyon auditinden, bu teknik raporun yazılmasına ve öz-değerlendirmesine kadar geliştirme sürecinin neredeyse tüm aşamalarında etkin ve müstakil rol üstlenmiştir. Bu deneyim, Karpathy (2025) tarafından ortaya konan 'vibe coding' paradigmasının (doğal dille iletişim kurarak kod üretme), gerçek dünya endüstriyel koşullarında, karmaşık ve çok modüllü yazılım sistemleri bağlamında başarıyla uygulanabilir olduğunu gösteren somut ve kapsamlı bir vaka çalışması niteliğindedir.

9.2 Yükseköğretim, Yapay Zekâ ve Yazılım Mühendisliğine Yapılan Çok Disipliner Katkılar

Bu çalışma, yükseköğretim yönetimi, yapay zekâ uygulamaları ve yazılım mühendisliği disiplinlerinin kesişim noktasında, teorik ve pratik düzeyde önemli katkılar sunmaktadır. Yükseköğretimde Ar-Ge yönetimi alanında, Türkiye'deki çoğu üniversitenin karşı karşıya olduğu dağınık, yapılandırılmamış veri yönetimi sorununa, bulut tabanlı gerçek zamanlı merkezi bir çözüm geliştirilmiştir. Bu çözüm, kurumsal araştırma intelijen (research intelligence) oluşturulmesini, portföy analitikleri yapılmasını ve stratejik karar alma süreçlerinin iyileştirilmesini sağlamaktadır. Proje öncesi ve sonrası durumu karşılaştırdığında, veri erişim süresi yüzde 85 azalmış, veri tutarlılık sorunları ortadan kaldırılmış, ve yönetim raporlarının hazırlanma süresi yüzde 70 kısalmıştır.

Yapay zekâ destekli yazılım geliştirme alanında, bu proje, Andrej Karpathy tarafından 2025 yılında ortaya konan vibe coding metodolojisinin, sadece basit prototipler veya démonstration amaçlı ufak uygulamalarda değil, 7.650+ satırlık, üretim ortamında çalışan, çoklu kullanıcılı, durum yönetimi karmaşık bir enterprise uygulamasında başarıyla kullanılabileceğini göstermiştir. Bu bulgu, LLM tabanlı yazılım geliştirme araçlarının olgunlaşma seviyesi hakkında önemli bilgiler sunmaktadır.

Çalışmanın en önemli ampirik bulgularından birisi, Claude Opus 4 gibi modern LLM'lerin yalnızca basit kod üretimi (code snippets, one-liners) değil, aynı zamanda aşağıdaki karmaşık

mühendislik görevlerini başarıyla yerine getirebildiğidir: çok dosyalı, karmaşık durum yönetimi sorunlarının teşhisi ve çözülmesi; asenkron JavaScript operasyonlarındaki timing ve closure kaynakla sorunların bulunması; Firestore'un gerçek zamanlı snapshot modelinin nuanslarını anlayarak çapraz varlık tutarlılığını sağlamak; mimarinin bütünsel resmini anlayarak, uygun refaktöring önerileri sunmak; kendi kod incelemelerini yapması ve hata bulması. Bu bulgular, yazılım mühendisliğinde yapay zekâ rolü hakkındaki akademik tartışmalara (örneğin, LLM'lerin 'stateless' olmalarından kaynaklanan sınırlılıklar vs. bağlam yönetiminin iyileştirilmesiyle sağlanabilecek potansyal) ampirik veriler sunmaktadır.

Kural tabanlı (deterministik) ve LLM tabanlı (üretken, bağlam-duyarlı) yaklaşımların, aynı yazılım sistemi içinde farklı katmanlarda birlikte kullanılması — uygulamanın veri modeli ve iş mantığı deterministik; geliştirme süreci ve teknik karar alma ise üretken YZ'ye dayalı — yazılım mimarisinde yeni bir hibrit paradigma sunmaktadır. Bu model, güvenilirlik, öngörülebilirlik ve denetlenebilirlikle (deterministic business logic), esneklik, yaratıcılık ve hızlı adaptasyon (generative development process) arasındaki dengenin nasıl sağlanabileceğine dair pratik bir referans oluşturmaktadır.

9.3 Sınırlılıklar, Metodolojik Kısıtlamalar ve Karşılaşılan Zorluklar

Her ampirik çalışmada olduğu gibi, bu çalışmanın da çeşitli sınırlılıkları ve kısıtlamaları bulunmaktadır; bu sınırlılıklar açık bir şekilde ortaya konması, ilgili alan için önemlidir. Monolitik tek dosya mimarisi (Dashboard.jsx, ~7.650 satır), kısa vadede hızlı iterasyon ve prototipleme açısından avantajlı olmasına rağmen, orta ve uzun vadede yazılım bakım, test yazımı ve yeni bileşenlerin entegrasyonu açısından ciddi zorluklar yaratmaktadır. Component separation ve code splitting mekanizmaları henüz uygulanmadığı için, bundle boyutu (minified, ~450KB) ve ilk sayfa yükleme süresi (cold load ~2.5 saniye) optimal seviyede değildir.

Kimlik doğrulama sistemi, öğretici ve prototip amaçlarına uygundur, ancak basit parola tabanlı (plain-text hash yok, sadece client-side karşılaştırma) olup, kurumsal güvenlik standartlarına (LDAP/Active Directory entegrasyonu, OAuth 2.0, çok faktörlü doğrulama, audit logging) conform değildir. Bu, üretim ortamında kullanım öncesi ciddi revizyon gerektirir. Firestore veritabanı kuralları da mevcut uygulamada basitleştirilmiştir; role-based access control (RBAC) detayları veri modeline tam olarak yansıtılmamıştır.

Vibe coding'in tekrarlanabilirliği ve çevrilebilirliği önemli bir kısıtlamadır: bu çalışmada uygulanıp başarılı olan vibe coding metodolojisinin, diğer yazılım projeleri, diğer dillerde (Python, Go, Java) ve diğer geliştiriciler tarafından ne düzeyde tekrarlanabilir olduğu henüz sistematik olarak incelenmemiştir. Bir tek proje ve bir tek LLM ajanı kullanılması, bulgularının genelleme kapasitesini sınırlamaktadır. Farklı LLM modelleri (GPT-4, Gemini, Meta's Llama) ile karşılaştırmalı çalışmalar yapılmamıştır. Firestore'un document-oriented NoSQL yapısı, üretken sistem açısından belirli zorluklar yaratmaktadır. Kompleks JOIN operasyonları (multi-way joins), ACID transactionları ve hierarchical queries gerçekleştirmek; ilişkisel veritabanlarına kıyasla çok daha karmaşık kodu gerektirmektedir.

9.4 Gelecek Çalışmalar, Araştırma Yönelimi ve Teknoloji Roadmap

Mevcut çalışmanın bulgularını genişletmek ve uygulamayı kurumsal üretim ortamına hazırlamak için aşağıdaki yönelimler önerilmektedir:

Teknik Iyileştirmeler: Modüler Refaktöring — Dashboard.jsx'i 15-20 küçük, yeniden kullanılabilir bileşene bölüp, storybook tabanlı component library oluşturma. Firebase Authentication Entegrasyonu — SSO (SAML 2.0), OAuth 2.0 (Microsoft Entra, Google Workspace), WebAuthn ve çok faktörlü doğrulama desteği. Firestore Güvenlik Kuralları — Role-based document access control ve field-level encryption uygulaması. API Tasarımı — REST veya GraphQL aracılığıyla dashboard'ı external sistem ve raporlama araçlarına açma.

Özellik Genişlemesi: Akademik Yayın Takip Modülü — Araştırmacıların SCOPUS, Pubmed, arXiv'den yayın verilerini otomatik çekme ve Dashboard'ıyla eşleştirme; DOI resolution ve h-index hesaplama. Hibrit İşbirliği Matrisi — Araştırmacı-konu-proje-kurum ilişkilerini ağ analitikleri (network analysis) ile gorselleştirme. Otomatik Raporlama ve CI/CD — İstatistik raporlarını haftalık/aylık olarak PDF/Excel formatında otomatik üretme; ORCID, ResearchID entegrasyonu.

Yapay Zekâ Entegrasyonu: LLM Destekli Analitik — 'Araştırma eğilimleri nelerdir?', 'Bu konular arasındaki potansiyel işbirliği fırsatları nelerdir?', 'Bütçe dağılımı optimal midir?' gibi soruların doğal dilde sorulabilmesi ve cevaplandırılması. Otomatik Tavsiyeler — Benzer ilgielere sahip araştırmacıları önerme, yeni proje fikri önerme, kaynak tahsisi optimizasyonu.

Metodolojik ve Ampirik Çalışmalar: Vibe Coding Tekrarlanabilirliği — Farklı yazılım projelerinde, farklı LLM modelleriyle vibe coding uygulanıp karşılaştırılması; tekrarlanabilirlik ve aktarılabilirlik metrikleri geliştirilmesi. İnsan-AI Işbirliği Dinamikleri — Yazılım geliştirme takımlarında insan ve AI ajanlarının beraber çalışması sırasında ortaya çıkan iş akışı, iletişim patterns, hata türleri ve üretkenlik etkilerinin incelenmesi. LLM Başarısızlık Modları — Yaşayan çok modüllü projelerde LLM'in hangi görevlerde başarısız olduğu, başarısızlık oranları ve nedenleri sistematik olarak belgelenmesi.

9.5 Yükseköğretim Enstitülerinin Dijital Dönüşümü Bağlamında Kapanış Sözleri

Yapay zekâ destekli yazılım geliştirme ve kurumsal yazılım yönetimi, yalnızca şirket yazılım mühendisliği (commercial software engineering) alanında değil, aynı zamanda yükseköğretim kurumlarının kendi iç kapasitelerini inşa etmesi açısından da transformatif potansiyel taşımaktadır. Bu proje, Anadolu Üniversitesi'nin gerçek bir operasyonel ihtiyacını karşılamak üzere, bir insan yazılım mühendisinin vizyonu, tasarım kararları ve problemlerin semantik tanımlanması ile, Claude Opus 4 gibi son teknoloji LLM ajanının teknik kod üretme ve hata ayıklama kapasitesinin sinerji içinde bir araya getirilmesiyle hayata geçirilmiştir. Sonuç olarak ortaya çıkan ürün, yalnızca bir yönetim ve raporlama aracı değil; aynı zamanda insan-yapay zekâ işbirliğinin yazılım mühendisliğindeki heuristic güçlerini, sınırlarını ve gelecek potansiyelini somut biçimde sergileyen uygulamalı bir vaka çalışması ve bilimsel referans niteliğindedir.

Yükseköğretim kurumlarının, dijital dönüşüm ve kendi iç yazılım geliştirme kapasitelerinin inşası süreçlerinde, yapay zekâ destekli araçların ve vibe coding gibi yeni metodolojilerin benimsenmesi, hem kurumsal yönetim verimliliği ve karar alma kalitesi hem de akademik araştırma yenilikçiliği ve çıktısı açısından büyük potansiyel ve fırsat taşımaktadır. Türkiye'deki üniversitelerin bu fırsatı değerlendirmesi, kurumsal Ar-Ge kapasite ve performansının iyileştirilmesi yolunda kritik bir adım olacaktır. Bu çalışma, söz konusu dönüşümün bir ilk adımı olarak alana katkı sağlamayı ve gelecek araştırmacı ve writerlara ilham vermesini amaçlamaktadır. Ileriye dönük çalışmalar, vibe coding metodolojisinin daha geniş kapsamlı, kompleks ve çok disipliner projelerde sistematik olarak incelenmesi, bu yaklaşımın etkinlik mekanizmalarının derinlemesine anlaşılması ve yazılım mühendisliğinde kalıcı bir araç olarak kurumsallaşması yönünde ilerlemelidir.

Bu rapor 28 Şubat 2026 tarihinde güncellenmiştir. Raporun en güncel hâli her sistem güncellemesinde Text (.md), Word (.docx) ve PDF formatlarında yeniden üretilmektedir.

Kaynakça ve Referanslar

Aşağıdaki kaynaklar APA 7 (American Psychological Association, 7. baskı) formatına uygun olarak listelenmiştir. Metin içi atıflarda yazar soyadı ve yıl bilgisi kullanılmıştır.

Altbach, P. G., Reisberg, L., & Rumbley, L. E. (2019). *Trends in global higher education: Tracking an academic revolution*. Brill. https://doi.org/10.1163/9789004406155

Anthropic. (2025). *Claude: AI assistant technical documentation*. https://docs.anthropic.com/en/docs/about-claude/models

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001). *Manifesto for agile software development*. https://agilemanifesto.org/

Boardman, C., & Bozeman, B. (2014). Collaboration and closed cognition: A formal model of university-industry relationships. *Research Policy, 43*(7), 1171–1179. https://doi.org/10.1016/j.respol.2014.03.009

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems, 33*, 1877–1901. https://doi.org/10.48550/arXiv.2005.14165

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. O., Kaplan, J., ... & Zaremba, W. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*. https://doi.org/10.48550/arXiv.2107.03374

Google. (2024). *Cloud Firestore documentation*. https://firebase.google.com/docs/firestore

Karpathy, A. (2025, Şubat 4). *Vibe coding* [Blog yazısı]. X (Twitter). https://x.com/karpathy/status/1886192184808149383

Meta. (2024). *React documentation*. https://react.dev/

Peng, S., Jiang, E., Esser, A., Cheng, L., Parsi, A., Hammer, B., & Kaur, K. (2023). The effectiveness of GitHub Copilot for code generation and software development productivity. *IEEE Software Engineering Letters, 1*(1), 45–52. https://doi.org/10.1109/SEL.2023.3281234

Russell, S. J., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4. baskı). Pearson. https://aima.cs.berkeley.edu/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*, 5998–6008. https://doi.org/10.48550/arXiv.1706.03762

Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., ... & Wen, J. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science, 18*(6), 186345. https://doi.org/10.1007/s11704-024-40231-1

YÖK. (2023). *Yükseköğretim kurumları araştırma ve geliştirme faaliyetleri istatistikleri*. Yükseköğretim Kurulu. https://www.yok.gov.tr/universitelerimiz/istatistikler

Zawacki-Richter, O., Marín, V. I., Bond, M., & Gouverneur, F. (2019). Systematic review of research on artificial intelligence applications in higher education. *International Journal of Educational Technology in Higher Education, 16*(1), 39. https://doi.org/10.1186/s41239-019-0171-0