

T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BSM 497 BİTİRME ÇALIŞMASI

**Yapay Zeka Tabanlı Akıllı Ev Asistanı Tasarımı ve
Gerçekleştirimi**

B231210371 – Furkan BARAN

Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ
Tez Danışmanı : Prof. Dr. Cüneyt Bayılmış

2024-2025 Güz Dönemi

T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

**Yapay Zeka Tabanlı Akıllı Ev Asistanı Tasarımı ve
Gerçekleştirimi**

BSM 497 - BİTİRME ÇALIŞMASI

Furkan BARAN

Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ

Bu tez .. / .. / ... tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

.....
Jüri Başkanı

.....
Üye

.....
Üye

ÖNSÖZ

Teknolojinin hızla geliştiğı günümüzde, evlerimiz artık sadece yaşam alanları olmaktan çıkıp, akıllı sistemlerle donatılmış, kullanıcılarının ihtiyaçlarını anlayabilen ve onlara yardımcı olabilen interaktif mekanlara dönüşmektedir. Bu dönüşüm sürecinde, yapay zeka ve doğal dil işleme teknolojilerinin ev otomasyonu ile birleşmesi, günlük yaşamımızı kolaylaştıran yeni nesil çözümlerin önünü açmıştır.

Bu bitirme tezi kapsamında, modern ev yaşamının gereksinimlerini karşılayacak, kullanıcılarıyla doğal dil üzerinden etkileşime girebilen ve ev otomasyonunu yapay zeka ile birleştiren bir akıllı ev asistanı geliştirilmiştir. Proje sürecinde, güncel teknolojileri kullanarak pratik bir çözüm oluşturmaya ve kullanıcı deneyimini ön planda tutmaya özen gösterilmiştir.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ.....	vii
ŞEKİLLER TABLOSU	viii
BÖLÜM 1. GİRİŞ.....	1
1.1. Projenin Amacı ve Önemi.....	1
1.2. Literatür Analizi.....	1
1.3. Piyasada Mevcut Çözümler	3
1.3.1. Ticari ses asistanları.....	3
1.3.2. Açık kaynak platformlar	3
1.3.3. Üretici spesifik çözümler.....	3
BÖLÜM 2. PROBLEM TANIMI VE ANALİZİ.....	4
2.1. Problemin Tanımı	4
2.2. Problemin Kapsamı	4
2.2.1. Sistem entegrasyonu sorunları	4
2.2.2. Kullanıcı etkileşimi zorlukları	5
2.2.2.1. Karmaşık komut yapıları ve kullanım prosedürleri	6
2.2.2.2. Yaşlı ve teknoloji konusunda deneyimsiz kullanıcıların adaptasyon sorunları	6
2.2.2.3. Doğal dil etkileşiminde yaşanan anlama ve yorumlama hataları	6
2.2.3. Veri yönetimi	7
2.2.3.1. Ev envanterinin takibi ve güncel tutulması	7
2.2.3.2. Kullanıcı görev ve aktivitelerinin kaydedilmesi.....	8
2.2.3.3. Sistem kullanım loglarının tutulması.....	8
2.2.3.4. Toplanan verilerin anlamlı şekilde organize edilmesi	8
2.3. Problemin Etkileri	9
2.3.1. Kullanıcı deneyimi üzerindeki etkiler	9
2.3.2. Ev yönetimi üzerindeki etkiler	9
2.3.3. Sağlık ve güvenlik üzerindeki etkiler	9
2.4. Mevcut Çözümlerin Sınırlamaları	9
2.4.1. Teknolojik sınırlamalar	9
2.4.2. Kullanılabilirlik sınırlamaları	10
2.4.3. Entegrasyon sınırlamaları	10
2.5. Problem Çözümünün Gereksinimleri	10
2.5.1. Doğal dil işleme gereksinimleri.....	10
2.5.2. Ev otomasyonu gereksinimleri	10

2.5.3.	Veri yönetimi gereksinimleri	11
2.5.4.	Kullanıcı arayüzü gereksinimleri.....	11
BÖLÜM 3.	KULLANILAN TEKNOLOJİLER VE YÖNTEMLER.....	12
3.1.	Geliştirme Ortamı	12
3.1.1.	Donanım ve işletim sistemi	12
3.1.2.	Konteynerizasyon	12
3.2.	Kullanılan Teknolojiler	12
3.2.1.	Home Assistant	12
3.2.2.	OpenAI GPT.....	13
3.2.3.	MongoDB	13
3.3.	Programlama Dilleri ve Kütüphaneler.....	13
3.3.1.	Python.....	13
3.3.2.	Temel kütüphaneler	13
3.3.3.	Sanal ortam (virtual environment).....	14
3.4.	İletişim Protokolleri.....	14
3.4.1.	REST API	14
3.4.2.	Veritabanı bağlantısı	14
3.5.	Güvenlik Önlemleri	15
3.5.1.	Kimlik doğrulama.....	15
3.5.2.	Veri güvenliği	15
BÖLÜM 4.	GELİŞTİRİLEN ÇÖZÜM VE UYGULAMA	16
4.1.	Sistem Mimarisi.....	16
4.2.	Sistem Akış Süreci.....	18
4.2.1.	Kullanıcı girdisinin alınması ve ilk işleme	19
4.2.2.	Ev durumu sorgusu.....	19
4.2.3.	GPT modeli işlemi.....	19
4.2.4.	Yanıt işleme ve eylem gerçekleştirme	21
4.2.5.	Sonuç iletimi ve iteratif iletişim	22
4.3.	Ana bileşenler.....	24
4.3.1.	Home Assistant, kurulum ve yapılandırma.....	24
4.3.1.1.	Kurulum ve başlatma.....	24
4.3.1.2.	Simülasyon ortamı.....	25
4.3.1.3.	API ve entegrasyon.....	27
4.3.2.	Python uygulama katmanı	27
4.3.2.1.	Ana kontrol modülü (main.py)	27
	İteratif İletişim Mekanizması	27

4.3.2.2.	Doğal dil işleme modülü (openai_integration.py)	29
4.3.2.3.	Konuşma geçmişi ve sistem prompt yönetimi (conversation_history.py) ..	31
4.3.2.4.	Home Assistant entegrasyon modülü (home_assistant.py).....	32
4.3.2.5.	Veritabanı altyapısı: Veritabanı kurulum modülü (DatabaseSetup.py)	34
4.3.2.6.	Veritabanı altyapısı: Veritabanı yönetim modülü (DatabaseManager.py) ..	36
4.3.2.7.	Sistem konfigürasyonu	37
4.3.3.	Doğal dil işleme servisi (OpenAI) katmanı	38
4.3.3.1.	Model seçimi ve alternatifler	38
4.3.3.2.	API erişimi ve yapılandırma	39
4.3.4.	MongoDB veritabanı katmanı	39
4.3.4.1.	Kurulum ve yapılandırma	40
4.3.4.2.	MongoDB Compass	40
4.3.4.3.	Veri yönetimi	40
BÖLÜM 5.	TEST VE SONUÇLAR	41
5.1.	Sistem Testleri	41
5.1.1.	Test senaryosu 1: Yeni kullanıcı kaydı ve sistem durumu sorgulama	41
5.1.2.	Test senaryosu 2: Envanter yönetimi	42
5.1.3.	Test senaryosu 3: Görev yönetimi	42
5.1.4.	Test senaryosu 4: Ev otomasyonu kontrolü	43
5.1.5.	Test senaryosu 5: Çoklu işlem ve hata düzeltme	44
5.1.6.	Test senaryosu 6: Günlük aktivite kaydı	45
5.1.7.	Test senaryosu 7: Çıkış bildirimi	46
5.1.8.	Test senaryosu 8: Kullanıcı ekleme ve sağlık durumu güncelleme	47
5.1.8.1.	Adım 1: Yeni kullanıcı ekleme	47
5.1.8.2.	Adım 2: Sağlık Durumu Güncelleme	47
BÖLÜM 6.	PROJE PLANI VE MALİYET ANALİZİ	50
6.1.	Proje Zaman Çizelgesi	50
6.2.	Maliyet Analizi	50
BÖLÜM 7.	SONUÇ VE GELİŞTİRME OLANAKLARI	52
7.1.	Genel Değerlendirme	52
7.2.	Temel Başarılar	52
7.3.	Geliştirme Önerileri	52
7.3.1.	Özel LLM entegrasyonu	52
7.3.2.	Dağıtık sistem mimarisi	53
7.3.3.	Ses arayüzü	53

SİMGELER VE KISALTMALAR LİSTESİ

API:	Application Programming Interface (Uygulama Programlama Arayüzü)
CRUD:	Create, Read, Update, Delete (Oluştur, Oku, Güncelle, Sil)
GPT:	Generative Pre-trained Transformer
HA:	Home Assistant (Ev Asistanı)
HTTP:	Hypertext Transfer Protocol
HTTPS:	Hypertext Transfer Protocol Secure
IoT:	Internet of Things (Nesnelerin İnterneti)
JSON:	JavaScript Object Notation
LLM:	Large Language Model (Büyük Dil Modeli)
NoSQL:	Not Only SQL
REST:	Representational State Transfer
URL:	Uniform Resource Locator
WSL:	Windows Subsystem for Linux
API:	Application Programming Interface (Uygulama Programlama Arayüzü)

ŞEKİLLER TABLOSU

Şekil 4.1. Sistem Mimarisi ve Bileşenler Arası İletişim	17
Şekil 4.2. Sistem Bileşenleri ve İletişim Akışı.....	18
Şekil 4.3. Yapay zeka modelinden beklenen yanıt formatı.....	20
Şekil 4.4. Örnek bir model yanıtı	21
Şekil 4.5. Modelin sistemle konuşarak yeni yanıt üretmesi sonucu verdiği çıktı	22
Şekil 4.6. Home Assitant paneli	26
Şekil 5.1. Komut sonrası Home Assitant panelindeki ilgili alanlar	44
Şekil 5.2. Test amaçlı asistan ile yapılan sohbet kaydı	49

ÖZET

Anahtar Kelimeler: Akıllı Ev, Yapay Zeka, Doğal Dil İşleme, Akıllı Ev Otomasyonu, OpenAI GPT, Aile ve Ev Yönetimi, Sağlık Takibi

Bu çalışmada, yapay zeka tabanlı doğal dil işleme teknolojilerini kullanan, MongoDB veritabanı ile entegre edilmiş ve Home Assistant altyapısı üzerine inşa edilmiş bir akıllı ev asistanı geliştirilmiştir. Sistem, ev sakinleriyle doğal dil üzerinden iletişim kurarak ev otomasyonu, envanter takibi, görev yönetimi ve günlük aktivite kaydı gibi temel ev yönetimi işlevlerini yerine getirmektedir.

Geliştirilen sistem, OpenAI'nin GPT-4o-mini modelini kullanarak kullanıcı isteklerini analiz etmekte ve bu istekleri uygun sistem komutlarına dönüştürmektedir. Kullanıcının kullandığı dili otomatik olarak algılayıp aynı dilde yanıt verebilme yeteneğine sahiptir. Ev sakinlerinin sağlık durumları, ev içi görevler, alışveriş listeleri ve envanter bilgileri MongoDB veritabanında tutulmakta, böylece sistem sürekli olarak ev ve kullanıcılarının durumunu takip edebilmektedir. Home Assistant entegrasyonu sayesinde ısıtma, aydınlatma gibi ev otomasyonu işlevleri de doğal dil komutlarıyla kontrol edilebilmektedir.

Test sürecinde kullanıcı ekleme, sağlık durumu güncelleme, envanter yönetimi, alışveriş listesi oluşturma, görev atama ve ev otomasyonu kontrolü gibi temel senaryolar başarıyla gerçekleştirilmiştir. Sistem, birden fazla işlemi tek bir doğal dil komutuyla gerçekleştirebilmekte ve hata durumlarında otomatik düzeltme yapabilmektedir.

Bu çalışma, modern ev yaşamının gereksinimlerini karşılayan, kullanıcı dostu ve genişletilebilir bir akıllı ev asistanı çözümü sunmaktadır. Docker konteyner teknolojisi kullanılarak geliştirilen sistem, yeni özelliklerin ve farklı ev otomasyon sistemlerinin entegrasyonuna olanak sağlayacak şekilde modüler bir yapıda tasarlanmıştır.

BÖLÜM 1. GİRİŞ

Modern yaşamın artan karmaşıklığı ve teknolojik gelişmeler, evlerimizin yönetimini giderek daha zorlu bir hale getirmektedir. Akıllı ev teknolojilerinin yaygınlaşmasıyla birlikte, ev sakinlerinin kontrol etmesi gereken cihaz ve sistemlerin sayısı artmakta, bu da kullanıcı deneyimini olumsuz etkilemektedir. Aynı zamanda, ev yaşamının düzenlenmesi, aile üyelerinin sağlık durumlarının takibi, ev envanterinin yönetimi gibi günlük yaşam görevleri de sürekli dikkat ve organizasyon gerektirmektedir.

1.1. Projenin Amacı ve Önemi

Bu projenin temel amacı, ev yaşamının farklı yönlerini tek bir sistem altında birleştiren, kullanıcılarıyla doğal dil üzerinden etkileşime girebilen kapsamlı bir akıllı ev asistanı geliştirmektir. Sistem, ev otomasyonu kontrolünü, aile üyelerinin sağlık takibini, ev envanteri yönetimini ve günlük aktivite kaydını entegre bir yapıda sunmaktadır.

Projenin özgün değeri, yapay zeka teknolojilerini ev yönetimi süreçleriyle birleştirerek, kullanıcıların günlük yaşamlarını kolaylaştırmasında yatmaktadır. Özellikle yaşlı ve teknoloji kullanımında zorlanan aile üyeleri için, doğal dil arayüzü sayesinde sistem ile etkileşim basitleştirilmiştir. Sistem, kullanıcının tercih ettiği dili otomatik olarak algılayıp o dilde yanıt verebilme yeteneğine sahiptir, bu da farklı dil tercihlerine sahip aile üyeleri için kullanım kolaylığı sağlamaktadır.

1.2. Literatür Analizi

Akıllı ev sistemleri ve doğal dil işleme teknolojilerinin entegrasyonu, son yıllarda hem akademik araştırmaların hem de endüstriyel uygulamaların odak noktası haline gelmiştir. Doğal dil işleme teknolojilerinin ev otomasyonunda kullanımı, kullanıcıların sistemlerle daha doğal ve sezgisel bir şekilde etkileşim kurmasını sağlamaktadır [1]. Yapılan sistematik literatür taramalarında, bu sistemlerin başarılı

bir şekilde uygulanmasında kullanıcı deneyiminin kritik bir faktör olduğu vurgulanmaktadır. Yapılan araştırmalar, özellikle kullanıcıların sistemle olan etkileşimlerinin doğallığının ve sistemin kullanım kolaylığının, adaptasyon sürecini doğrudan etkilediğini ortaya koymuştur [2].

Günümüz piyasasında mevcut çözümler incelendiğinde, üç ana kategori öne çıkmaktadır. Birinci kategoride, büyük teknoloji şirketleri tarafından geliştirilen ses asistanları ve doğal dil arayüzleri yer almaktadır. Araştırmalar, doğal dil işleme ve IoT teknolojilerini kullanan ses kontrollü ev otomasyon sistemlerinin kullanılabilirliğini detaylı şekilde analiz etmiş [3], özellikle yaşlı kullanıcıların bu sistemlerle etkileşimde yaşadığı zorlukları belgelemiştir [4].

İkinci kategoride, açık kaynak ev otomasyon platformları bulunmaktadır. Literatürdeki çalışmalar, bu platformların kullanıcılara sunduğu yüksek düzeyde özelleştirme ve esneklik özelliklerini incelemiş, açık kaynak sistemlerin farklı kullanıcı ihtiyaçlarına uyum sağlama konusunda sağladığı avantajları ortaya koymuştur [5].

Üçüncü ve giderek önem kazanan bir kategori ise sağlık takip sistemleridir. Yapılan araştırmalar, IoT teknolojilerinin gelişmesiyle birlikte, akıllı evlerde sağlık izleme sistemlerinin kullanımının yaygınlaşmasını ve etkilerini incelemiş, özellikle yaşlı bakımında ve kronik hastalıkların yönetiminde bu sistemlerin sağladığı faydaları kapsamlı bir şekilde belgelemiştir [6].

Literatürdeki çalışmalar incelendiğinde, bu üç kategorinin (ev otomasyonu, doğal dil işleme ve sağlık takibi) genellikle ayrı sistemler olarak ele alındığı görülmektedir. Her bir alan kendi içinde gelişmekte olup, tümünün tek bir platform altında entegrasyonu konusunda çalışmalar henüz başlangıç aşamasındadır. Bu entegrasyonun potansiyel faydaları olmakla birlikte, teknik zorluklar, güvenlik gereksinimleri ve kullanıcı gizliliği gibi önemli konuların çözüme kavuşturulması gerekmektedir.

1.3. Piyasada Mevcut Çözümler

Akıllı ev sistemleri pazarında çeşitli çözümler bulunmaktadır. Bu çözümleri üç ana kategoride incelemek mümkündür:

1.3.1. Ticari ses asistanları

- Amazon Alexa: Kapsamlı ev otomasyonu kontrolü, üçüncü parti entegrasyonları ve gelişmiş doğal dil anlama yetenekleri sunmaktadır.
- Google Assistant: Yapay zeka destekli ses kontrolü ve geniş cihaz uyumluluğu ile öne çıkmaktadır.
- Apple Siri: HomeKit ekosistemi üzerinden güvenli ve entegre bir ev otomasyonu deneyimi sağlamaktadır.

1.3.2. Açık kaynak platformlar

- Home Assistant: Geniş cihaz desteği, yüksek özelleştirilebilirlik ve aktif geliştirici topluluğu ile dikkat çekmektedir.
- OpenHAB: Java tabanlı mimarisi ve esnek yapılandırma seçenekleri sunan açık kaynak bir çözümdür.
- Domoticz: Hafif yapısı ve düşük sistem gereksinimleri ile küçük ölçekli uygulamalar için tercih edilmektedir.

1.3.3. Üretici spesifik çözümler

- Samsung SmartThings: Samsung ekosistemi içinde entegre çalışan, bulut tabanlı bir platformdur.
- Philips Hue: Aydınlatma odaklı, ancak diğer sistemlerle entegrasyon imkanı sunan bir çözümdür.
- IKEA TRÅDFRI: Uygun fiyatlı ve kullanımı kolay bir akıllı ev ekosistemi sunmaktadır.

Her bir çözümün kendine özgü avantajları ve kısıtlamaları bulunmaktadır. Ticari ses asistanları kullanım kolaylığı sunarken, açık kaynak platformlar daha fazla özelleştirme imkanı sağlamaktadır. Üretici spesifik çözümler ise genellikle belirli bir ekosistem içinde optimum performans göstermektedir.

BÖLÜM 2. PROBLEM TANIMI VE ANALİZİ

2.1. Problemin Tanımı

Modern ev yaşamında, akıllı cihazların ve sistemlerin sayısının artmasıyla birlikte, kullanıcılar için sistem yönetimi giderek karmaşık hale gelmektedir. Ev sakinleri, farklı arayüzler ve kontrol mekanizmaları arasında geçiş yapmak zorunda kalmakta, bu da kullanıcı deneyimini olumsuz etkilemektedir. Ayrıca, ev yaşamının önemli unsurları olan sağlık takibi, envanter yönetimi ve günlük aktivitelerin organizasyonu gibi işlevler genellikle birbirinden bağımsız sistemler üzerinden yürütülmektedir.

2.2. Problemin Kapsamı

Mevcut durum analiz edildiğinde, aşağıdaki temel sorun alanları öne çıkmaktadır:

2.2.1. Sistem entegrasyonu sorunları

Akıllı ev ekosisteminde, farklı üreticilere ait cihazların entegrasyonu önemli bir sorun teşkil etmektedir. Her üretici kendi protokollerini ve standartlarını kullanma eğilimindedir. Örneğin, Philips Hue aydınlatma sistemleri ZigBee protokolünü kullanırken, bazı akıllı prizler Wi-Fi üzerinden çalışmakta, kapı sensörleri ise Z-Wave protokolünü tercih etmektedir. Bu protokol çeşitliliği, cihazların birbiriyle doğrudan iletişim kurmasını zorlaştırmaktadır.

Çoklu arayüz problemi, kullanıcıların her cihaz için ayrı bir uygulama veya kontrol paneli kullanmak zorunda kalmasına yol açmaktadır. Bir kullanıcının ışıkları kontrol etmek için bir uygulama, termostatı ayarlamak için başka bir uygulama ve güvenlik sistemini yönetmek için üçüncü bir arayüz kullanması gerekmektedir. Bu durum, özellikle yaşlı kullanıcılar ve teknoloji konusunda deneyimsiz bireyler için önemli kullanılabilirlik sorunları yaratmaktadır.

Sistemler arası veri paylaşımı ve senkronizasyon konusunda da ciddi zorluklar yaşanmaktadır. Örneğin, hareket sensöründen gelen verinin hem aydınlatma sistemini hem de güvenlik sistemini tetiklemesi gerektiğinde, bu sistemler arasında güvenilir ve hızlı bir veri paylaşımı gerekmektedir. Ancak farklı sistemlerin kullandığı veri formatları ve iletişim protokolleri, bu tür entegrasyonları karmaşık hale getirmektedir.

Bu entegrasyon sorunları aşağıdaki spesifik zorluklara yol açmaktadır:

- **Protokol Uyumsuzluğu:** Farklı iletişim protokolleri (Wi-Fi, ZigBee, Z-Wave, Bluetooth) arasında çeviri yapılması gerekliliği
- **Veri Format Farklılıkları:** Her sistemin kendi veri yapısını kullanması ve bu yapılar arasında dönüşüm ihtiyacı
- **Gecikme Sorunları:** Protokol dönüşümleri ve veri aktarımları sırasında oluşan gecikmeler
- **Güvenlik Riskleri:** Farklı sistemler arasındaki veri paylaşımında ortaya çıkan potansiyel güvenlik açıkları
- **Yetkilendirme Karmaşası:** Her sistemin kendi kimlik doğrulama mekanizmasını kullanması
- **Senkronizasyon Problemleri:** Sistemler arasındaki durum güncellemelerinin eş zamanlı olarak yapılamaması

Bu sorunlar, ev otomasyonu sistemlerinin potansiyelinin tam olarak kullanılmasını engellemekte ve kullanıcı deneyimini olumsuz etkilemektedir. Örneğin, bir "evden çıkış" senaryosunda tüm ışıkların kapanması, termostatın ekonomi moduna geçmesi, güvenlik sisteminin devreye girmesi ve priz güç tüketiminin minimize edilmesi gibi birden fazla sistemin koordineli çalışması gerekmektedir. Ancak mevcut entegrasyon sorunları nedeniyle bu tür senaryoların güvenilir şekilde uygulanması zorlaşmaktadır

2.2.2. Kullanıcı etkileşimi zorlukları

Akıllı ev sistemlerinin yaygınlaşmasının önündeki en önemli engellerden biri, kullanıcı etkileşiminde yaşanan zorluklardır. Bu zorluklar, sistemlerin kullanım

karmaşıklığından doğal dil işleme sınırlamalarına kadar geniş bir yelpazede kendini göstermektedir.

2.2.2.1. Karmaşık komut yapıları ve kullanım prosedürleri

Mevcut sistemlerde, basit bir işlemin gerçekleştirilmesi bile birden fazla adım veya özel komut yapısı gerektirebilmektedir. Örneğin:

- Bir oda sıcaklığının programlanması için önce termostat uygulamasının açılması, ardından program menüsüne girilmesi, zaman dilimlerinin ayarlanması ve her dilim için sıcaklık değerlerinin girilmesi gerekmektedir.
- Otomatik senaryolar oluşturmak için kullanıcıların "if-then" mantığını anlamaları ve karmaşık tetikleyici-eylem ilişkilerini doğru şekilde kurgulamaları beklenmektedir.
- Cihaz kurulumları sırasında Wi-Fi ağ yapılandırması, cihaz eşleştirme, firmware güncelleme gibi teknik prosedürlerin takip edilmesi gerekmektedir.

2.2.2.2. Yaşlı ve teknoloji konusunda deneyimsiz kullanıcıların adaptasyon sorunları

Özellikle yaşlı kullanıcılar ve teknoloji konusunda sınırlı deneyime sahip bireyler için sistem adaptasyonu ciddi zorluklar içermektedir:

- Dokunmatik arayüzlerin kullanımında yaşanan fiziksel zorluklar
- Menü yapılarının karmaşıklığı ve navigasyon sorunları
- Teknik terimlerin anlaşılmasında yaşanan güçlükler
- Hata durumlarında çözüm üretme konusundaki yetersizlikler
- Sistem güncellemeleri ve değişikliklere adaptasyon zorluğu
- Çoklu uygulama ve arayüz kullanımında yaşanan karışıklıklar

2.2.2.3. Doğal dil etkileşiminde yaşanan anlama ve yorumlama hataları

Ses asistanları ve doğal dil arayüzleri, kullanım kolaylığı vaat etmelerine rağmen çeşitli sorunlar barındırmaktadır:

1) Dil İşleme Sorunları:

- a) Aksanlı konuşmaların yanlış anlaşılması
- b) Arka plan gürültüsünden kaynaklanan tanıma hataları
- c) Türkçe gibi sondan eklemeli dillerde yaşanan anlama zorlukları
- d) Bağlam bağımlı ifadelerin yanlış yorumlanması

2) Komut Yapısı Kısıtlamaları:

- a) Belirli kalıpların dışına çıkıldığında sistemin yanıt verememesi
- b) Kompleks isteklerin parçalara bölünmesi gerekliliği
- c) Doğal konuşma akışına uygun olmayan katı komut yapıları

3) Bağlam Yönetimi Sorunları:

- a) Önceki konuşmaların bağlamının korunamaması
- b) Çok turlu konuşmalarda tutarlılık eksikliği
- c) Kullanıcı tercihlerinin ve alışkanlıklarının öğrenilememesi

4) Geri Bildirim Mekanizması Eksiklikleri:

- a) Komutların doğru anlaşılıp anlaşılmadığının net olarak belirtilememesi
- b) Hata durumlarında yetersiz veya kafa karıştırıcı geri bildirimler
- c) Alternatif önerilerin sunulmaması

Bu zorluklar, kullanıcıların sistemle olan etkileşimlerini sınırlandırmakta ve sistemin potansiyel faydalarından tam olarak yararlanılamamasına neden olmaktadır. Özellikle yaşlı kullanıcılar için tasarlanan sistemlerde, bu etkileşim zorluklarının giderilmesi kritik önem taşımaktadır.

2.2.3. Veri yönetimi

Akıllı ev sistemlerinde temel veri yönetimi ihtiyaçları şunlardır:

2.2.3.1. Ev envanterinin takibi ve güncel tutulması

- Ev içindeki malzemelerin stok durumunun takibi
- Alışveriş listelerinin oluşturulması ve yönetimi
- Minimum stok seviyelerinin belirlenmesi ve takibi
- Ürünlerin kategorilere göre organizasyonu
- Envanter güncelleme işlemlerinin kayıt altına alınması

2.2.3.2. Kullanıcı görev ve aktivitelerinin kaydedilmesi

- Ev içi görevlerin atanması ve takibi
- Görev tamamlanma durumlarının güncellenmesi
- Düzenli yapılması gereken işlerin planlanması
- Kullanıcılara atanan görevlerin listelenmesi
- Görev önceliklerinin ve tarihlerinin yönetimi

2.2.3.3. Sistem kullanım loglarının tutulması

- Kullanıcı komutlarının kaydedilmesi
- Sistem yanıtlarının ve işlem sonuçlarının loglanması
- Hata durumlarının ve çözümlerinin kaydedilmesi
- Sistem performansının izlenmesi için gerekli verilerin toplanması

2.2.3.4. Toplanan verilerin anlamlı şekilde organize edilmesi

- Verilerin kategorilere göre sınıflandırılması
- Kullanıcı bazlı veri organizasyonu
- Tarih ve zaman bazlı veri düzenleme
- Verilere hızlı erişim için indeksleme
- Düzenli veri temizleme ve güncelleme mekanizmaları

Bu veriler, ev yaşamının düzenlenmesi ve sistem işlevselliği için gereklidir. Verilerin düzenli ve erişilebilir şekilde saklanması, sistemin etkin kullanımı için önem taşımaktadır. Örneğin, düzenli tutulan envanter kayıtları sayesinde ihtiyaç duyulan malzemelerin zamanında temin edilmesi sağlanabilir, veya görev takip sistemi ile ev içi sorumlulukların dengeli dağılımı ve zamanında tamamlanması mümkün olur.

2.3. Problemin Etkileri

2.3.1. Kullanıcı deneyimi üzerindeki etkiler

- Sistem kullanımında yaşanan zorluklar ve frustrasyon
- Teknoloji adaptasyonunda yaşanan direnç
- Sistemlerin sunduğu potansiyel faydalardan yeterince yararlanamama

2.3.2. Ev yönetimi üzerindeki etkiler

- Kaynakların verimsiz kullanımı
- Enerji tüketiminin optimize edilememesi
- Ev içi görevlerin etkin organizasyon eksikliği

2.3.3. Sağlık ve güvenlik üzerindeki etkiler

- Sağlık durumu takibinde yaşanan kesintiler
- Acil durumlarda müdahale gecikmesi riskleri
- Yaşlı ve özel bakım gerektiren bireylerin ihtiyaçlarının tam karşılanamaması

2.4. Mevcut Çözümlerin Sınırlamaları

2.4.1. Teknolojik sınırlamalar

Mevcut akıllı ev sistemlerinde, özellikle doğal dil işleme yetenekleri henüz istenilen seviyede değildir. Sistemler, karmaşık veya bağlam gerektiren komutları anlamakta zorlanmakta, çoğunlukla önceden belirlenmiş basit komut kalıplarıyla sınırlı kalmaktadır. Örneğin, "Hava kararmak üzere, perdeleri kapat ve ışıkları aç" gibi birden fazla işlem içeren ve duruma bağlı komutların işlenmesinde sorunlar yaşanmaktadır.

2.4.2. Kullanılabilirlik sınırlamaları

Kullanıcılar, sistemlerin kurulum ve yapılandırma süreçlerinde zorluklarla karşılaşmaktadır. Özellikle teknik bilgisi sınırlı olan kullanıcılar için sistem ayarlarının yapılması ve özelleştirilmesi karmaşık olabilmektedir. Hata durumlarında kullanıcıya sunulan geri bildirimler genellikle yetersiz kalmakta, sorun giderme süreci kullanıcı dostu bir şekilde yönetilememektedir.

2.4.3. Entegrasyon sınırlamaları

Farklı sistemlerin bir arada çalışması konusunda önemli kısıtlamalar bulunmaktadır. Her sistemin kendi veri formatını ve iletişim protokolünü kullanması, sistemler arası entegrasyonu zorlaştırmaktadır. Bu durum, kullanıcıların tek bir platform üzerinden tüm ev yönetimini gerçekleştirmesini engellemektedir.

2.5. Problem Çözümünün Gereksinimleri

Geliştirilen sistemin, belirlenen problemleri çözebilmesi için aşağıdaki gereksinimleri karşılaması gerekmektedir:

2.5.1. Doğal dil işleme gereksinimleri

- Günlük konuşma dilinde verilen komutları anlayabilme
- Birden fazla işlem içeren karmaşık komutları yorumlayabilme
- Bağlama uygun yanıtlar üretebilme
- Türkçe dil desteği

2.5.2. Ev otomasyonu gereksinimleri

- Temel ev cihazlarının kontrolü (ışıklar, sıcaklık vb.)
- Cihaz durumlarının takibi ve raporlanması
- Otomatik durum güncellemeleri
- Hata durumlarında uygun geri bildirim

2.5.3. Veri yönetimi gereksinimleri

- Ev envanterinin takibi
- Görev ve aktivite kayıtlarının tutulması
- Kullanıcı etkileşimlerinin loglanması
- Verilerin organize şekilde depolanması

2.5.4. Kullanıcı arayüzü gereksinimleri

- Basit ve anlaşılır komut yapısı
- Kullanıcı dostu etkileşim
- Açık ve net sistem yanıtları
- Kolay kullanım ve erişim

Bu gereksinimler, sistemin temel işlevselliğini ve kullanılabilirliğini sağlamak için minimum ihtiyaçları tanımlamaktadır.

BÖLÜM 3. KULLANILAN TEKNOLOJİLER VE YÖNTEMLER

3.1. Geliştirme Ortamı

3.1.1. Donanım ve işletim sistemi

- İşlemci: Intel Core i5-1240P
- RAM: 16 GB
- İşletim Sistemi: Windows 11 64-bit
- WSL2 (Windows Subsystem for Linux)

3.1.2. Konteynerizasyon

Docker konteyner teknolojisi, sistemin temel bileşenlerini izole ve taşınabilir bir şekilde çalıştırmak için kullanılmıştır. Bu yapı:

- Bileşenlerin bağımsız çalışmasını
- Kolay kurulum ve taşınabilirlik
- Kaynak yönetimi ve izolasyon sağlamaktadır.

3.2. Kullanılan Teknolojiler

3.2.1. Home Assistant

Açık kaynak kodlu ev otomasyon platformu olan Home Assistant:

- REST API desteği
- Geniş cihaz uyumluluğu
- Özelleştirilebilir yapı
- Aktif geliştirici topluluğu özellikleriyle tercih edilmiştir.

3.2.2. OpenAI GPT

Doğal dil işleme yetenekleri için GPT-4o-mini modeli kullanılmıştır. Bu model:

- Hızlı yanıt süresi
- Maliyet etkinliği
- Tutarlı çıktı üretimi avantajlarını sunmaktadır.

3.2.3. MongoDB

NoSQL veritabanı olarak MongoDB seçilmiştir. Tercih sebepleri:

- Esnek veri yapısı
- JSON benzeri döküman desteği
- Yüksek performans
- Ölçeklenebilirlik

3.3. Programlama Dilleri ve Kütüphaneler

3.3.1. Python

Ana geliştirme dili olarak Python kullanılmıştır. Tercih sebepleri:

- Zengin kütüphane ekosistemi
- API entegrasyonu kolaylığı
- Okunabilir ve sürdürülebilir kod yapısı

3.3.2. Temel kütüphaneler

- Requests: HTTP istekleri için
- PyMongo: MongoDB etkileşimi için
- JSON: Veri formatı işlemleri için
- Logging: Sistem logları için

3.3.3. Sanal ortam (virtual environment)

Proje için Python sanal ortamı (venv) kullanılmıştır. Bu tercih şu avantajları sağlamaktadır:

- Proje bağımlılıklarının izolasyonu
- Paket versiyonlarının yönetimi
- Çakışmaların önlenmesi
- Taşınabilirlik ve tekrar üretilebilirlik

Sanal ortam, aşağıdaki paketleri ve sürümlerini içermektedir:

- openai==1.58.1
- pymongo==4.10.1
- requests==2.32.3
- python-dateutil==2.9.0.post0
- python-dotenv==1.0.0

Bu yapılandırma, projenin farklı geliştirme ortamlarında tutarlı bir şekilde çalışmasını sağlamaktadır.

3.4. İletişim Protokolleri

3.4.1. REST API

Sistemler arası iletişim REST API üzerinden sağlanmaktadır:

- HTTP/HTTPS protokolü
- JSON veri formatı
- Stateless iletişim
- Standart HTTP metodları

3.4.2. Veritabanı bağlantısı

MongoDB ile iletişim için:

- TCP/IP bağlantısı
- BSON veri formatı
- İndeksleme ve sorgu optimizasyonu

3.5. Güvenlik Önlemleri

3.5.1. Kimlik doğrulama

- API token tabanlı güvenlik
- Veritabanı erişim kontrolü
- WSL izolasyonu

3.5.2. Veri güvenliği

- Şifreli iletişim (HTTPS)
- Konteyner izolasyonu
- Log yönetimi

Bu bölümde kullanılan teknolojiler ve altyapı bileşenleri genel hatlarıyla açıklanmıştır. Bir sonraki bölümde, bu teknolojiler kullanılarak geliştirilen çözümün detayları ele alınacaktır.

BÖLÜM 4. GELİŞTİRİLEN ÇÖZÜM VE UYGULAMA

4.1. Sistem Mimarisi

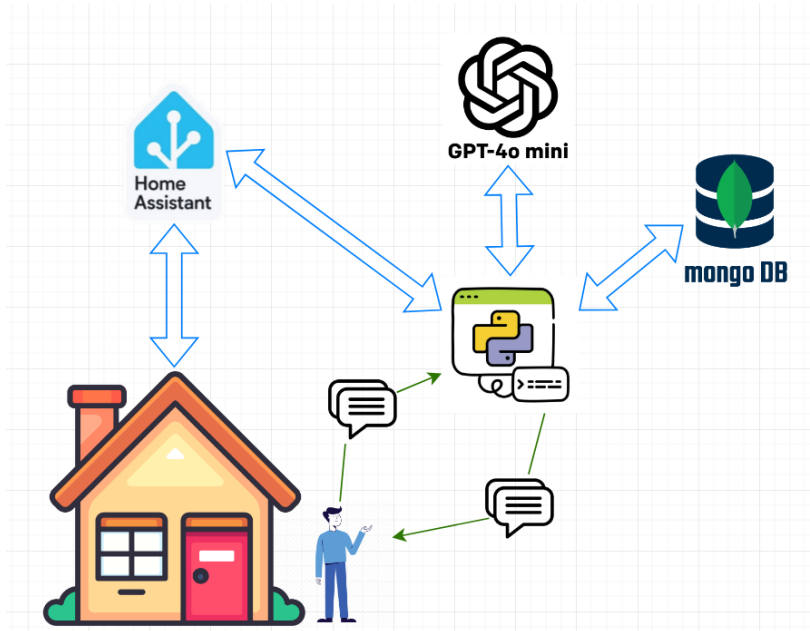
Sistemin doğal dil işleme yetenekleri, OpenAI'nin GPT-4o-mini modeli kullanılarak geliştirilmiştir. Bu model seçiminde, hız, maliyet etkinliği ve tutarlı yanıt üretimi gibi faktörler belirleyici olmuştur. Geliştirilen çözüm, kullanıcıların günlük konuşma dilinde verdikleri komutları anlayıp uygun sistem işlemlerine dönüştürmektedir. Örneğin, "Hava kararmak üzere, perdeleri kapat ve ışıkları aç" gibi bağlamsal ve çoklu komut içeren doğal cümleler, sistem tarafından anlaşılıp uygun API çağrılarına dönüştürülebilmektedir. Ayrıca sistem, kullanıcının hangi dilde komut verdiğini otomatik olarak algılayıp yanıtlarını aynı dilde üretebilmektedir, bu da çok dilli ev ortamlarında kullanım kolaylığı sağlamaktadır.

Sistemin doğal dil anlama yeteneği, özel olarak tasarlanmış bir prompt yapısı üzerine inşa edilmiştir. Bu yapı, modele ev otomasyonu bağlamını, mevcut sistem durumunu ve beklenen yanıt formatını sağlamaktadır. Model, bu bağlamı kullanarak kullanıcı komutlarını analiz etmekte ve JSON formatında yapılandırılmış yanıtlar üretmektedir. Bu yanıtlar, hem ev otomasyonu sistemine gönderilecek komutları hem de kullanıcıya iletilecek doğal dil mesajlarını içermektedir.

Sistemin en önemli özelliklerinden biri, çoklu görev işleme yeteneğidir. Tek bir kullanıcı komutundan birden fazla sistem işlemi üretilebilmekte ve bu işlemler sıralı şekilde yönetilebilmektedir. Ayrıca, model bağlamı koruyarak ardışık komutları anlayabilmekte ve önceki etkileşimleri göz önünde bulundurarak daha doğal bir iletişim sağlayabilmektedir.

Doğal dil işleme çözümü, hata toleransı da göz önünde bulundurularak tasarlanmıştır. Belirsiz veya eksik komutlar durumunda sistem, kullanıcıdan

açıklama isteyebilmekte veya en olası yorumu yaparak işlemi gerçekleştirebilmektedir. Bu sayede kullanıcılar, teknik detayları bilmeden veya özel komut yapılarını ezberlemeden sistemi etkin şekilde kullanabilmektedir.



Şekil 4.1. Sistem Mimarisi ve Bileşenler Arası İletişim

Geliştirilen sistemin temel bileşenleri ve aralarındaki iletişim akışı şematik olarak gösterilmiştir. Sistem dört ana bileşenden oluşmaktadır:

1. **Python Uygulama Katmanı:**
Sistemin merkezi kontrol birimidir ve kullanıcıdan gelen komutların işlenmesi, diğer bileşenlere yönlendirilmesi ve sonuçların kullanıcıya iletilmesi gibi işlemleri yürütür. Konsol tabanlı etkileşim doğrudan bu katman üzerinden sağlanır. Kullanıcı, komutlarını Python uygulamasına girdikten sonra, uygulama bu komutları işleyerek ilgili işlemleri gerçekleştirir ve çıktıyı yine konsol üzerinden sunar.
2. **GPT-4o mini:**
Doğal dil işleme ve komut yorumlama işlevini yerine getiren yapay zeka modeli, Python uygulama katmanı tarafından gönderilen girdileri analiz ederek sistem tarafından anlaşılabilir JSON formatında çıktılar üretir. Bu

çıktılar, API çağrıları ve veritabanı işlemleri gibi farklı işlemleri tetiklemek için kullanılır.

3. Home

Assistant:

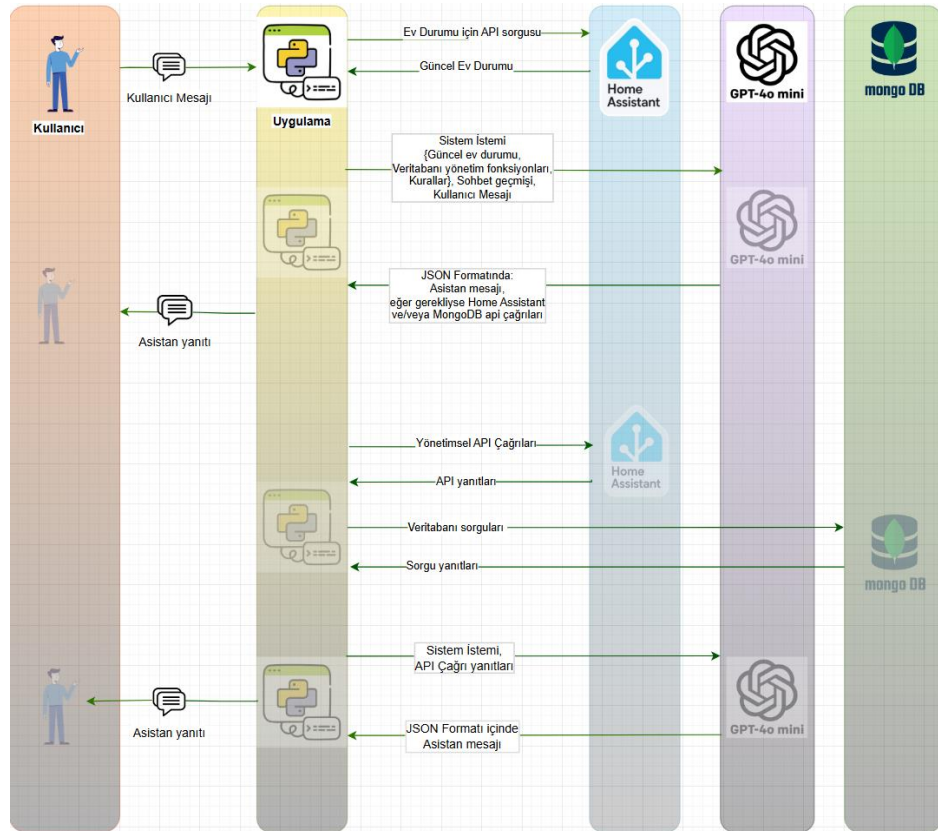
Akıllı ev cihazlarının yönetimini sağlayan platformdur. Python uygulama katmanı tarafından gönderilen API çağrılarını alır ve ilgili cihazları kontrol eder. Örneğin, ışıkların açılıp kapanması veya termostat sıcaklığının ayarlanması gibi işlemler Home Assistant aracılığıyla gerçekleştirilir.

4. MongoDB:

Kullanıcı verilerini, sistem loglarını ve ev envanterini saklayan veritabanıdır. Bu katman, sistemin geçmişe yönelik bağlamını oluşturmak için kritik bir rol oynar. Kullanıcıların geçmiş komutları, cihaz durumları ve görev bilgileri bu veritabanında depolanır ve gerektiğinde işlenir.

Mavi oklar HTTP/API iletişimini, yeşil oklar ise kullanıcı ile sistem arasındaki doğrudan etkileşimi göstermektedir.

4.2. Sistem Akış Süreci



Şekil 4.2. Sistem Bileşenleri ve İletişim Akışı

Bu diyagram, sistemin ana bileşenleri arasındaki etkileşimi ve veri akışını göstermektedir. Kullanıcıdan gelen doğal dil komutları, Python uygulama katmanı tarafından işlenir ve gerekli durumlarda Home Assistant, GPT modeli ve MongoDB veritabanı ile iletişim kurulur. Her bir istek-yanıt döngüsünde, sistem durumu sorgulanır, komutlar işlenir ve sonuçlar kullanıcıya iletilir. Yeşil oklar HTTP/API çağrılarını, konuşma balonları ise kullanıcı-sistem etkileşimini temsil etmektedir.

4.2.1. Kullanıcı girdisinin alınması ve ilk işleme

- Kullanıcı, doğal dil ile komutunu sisteme girer
- Python uygulama katmanı, gelen komutu alır ve ön işleme tabi tutar

4.2.2. Ev durumu sorgusu

- Uygulama, Home Assistant'a API sorgusu göndererek güncel ev durumunu alır
- Alınan durum bilgisi, komut bağlamını zenginleştirmek için kullanılır

4.2.3. GPT modeli işlemi

Sistem, GPT-4o mini modeline aşağıdaki bilgileri ve kuralları içeren yapılandırılmış bir prompt gönderir:

I. Bağlamsal Bilgiler:

- Güncel ev durumu (Home Assistant'tan alınan)
- Veritabanı yönetim fonksiyonları ve parametreleri
- Sohbet geçmişi
- Kullanıcı mesajı

II. Yanıt Format Kuralları:

```
{
  "message": "Doğal dil yanıtı",
  "api_calls": [{
    "action": "domain.service",
    "entity_id": "domain.entity_id",
    "parameters": {}
  }],
  "db_calls": [{
    "function": "function_name",
    "parameters": {}
  }],
  "need_response": boolean
}
```

Şekil 4.3. Yapay zeka modelinden beklenen yanıt formatı

III. Temel Kurallar:

- Sadece JSON formatında yanıt üretilmeli
- "message" alanında özel karakterler ve emojiler kullanılmamalı
- Home Assistant için sadece mevcut servis ve entity_id'ler kullanılmalı
- Veritabanı işlemleri için tanımlı fonksiyon isimleri kullanılmalı

IV. Veri İşlem Kuralları:

- Veritabanı okuma işlemleri öncelikli yapılmalı
- Veri durumu varsayılmamalı, her zaman sorgulanmalı
- Güncelleme öncesi okuma sonuçları beklenmeli (need_response: true)
- Yanıt bekleyen işlemlerde yeni yanıt üretilmeli

V. Zaman Yönetimi Kuralları:

- Gece yarısı sonrası işlemlerde hem güncel hem önceki gün logları kontrol edilmeli
- Kullanıcının günlük döngüsü takvim gününden öncelikli

VI. Loglama Kuralları:

Aşağıdaki olaylar her zaman kaydedilmeli:

- Eve giriş/çıkışlar
- Yemek aktiviteleri
- Sağlık ile ilgili durumlar
- Önemli konuşmalar
- Sistem değişiklikleri

Loglanmaması gereken durumlar:

- Selamlaşmalar
- Durum kontrolleri
- Basit sorgular

Model, bu kurallar çerçevesinde kullanıcı komutunu yorumlayarak yapılandırılmış JSON yanıtı üretir. Örneğin:

```
{
  "message": "Perdeleri kapatıyor ve ışıkları açıyorum. Ayrıca bu işlemi günlük aktivite loguna ekliyorum.",
  "api_calls": [
    {
      "action": "cover.close_cover",
      "entity_id": "cover.living_room"
    },
    {
      "action": "light.turn_on",
      "entity_id": "light.living_room"
    }
  ],
  "db_calls": [
    {
      "function": "add_daily_log",
      "parameters": {
        "title": "Evening Routine",
        "details": "Closed curtains and turned on lights"
      }
    }
  ],
  "need_response": false
}
```

Şekil 4.4. Örnek bir model yanıtı

4.2.4. Yanıt işleme ve eylem gerçekleştirme

JSON yanıtı içerisinde üç temel bileşen bulunabilir:

- Asistan mesajı (kullanıcıya iletilecek doğal dil yanıtı)
- Home Assistant API çağrıları (cihaz kontrolü için)
- MongoDB veritabanı sorguları (veri işlemleri için)

Sistem, yanıtı işlerken özellikle "need_response" parametresini kontrol eder. Bu parametre, yapay zeka modelinin veritabanı sorgularından gelecek bilgilere göre yeni bir yanıt üretmesi gerektiğini belirtir. Örneğin:

```
# İlk yanıt
{
  "message": "Bugünkü aktiviteleri kontrol ediyorum...",
  "db_calls": [
    {
      "function": "get_today_logs",
      "parameters": {}
    }
  ],
  "need_response": true # Veri bekleniyor
}

# Veritabanı yanıtı alındıktan sonra üretilen ikinci yanıt
{
  "message": "Bugün saat 14:00'te aile yemeği ve 16:30'da temizlik aktivitesi kaydedilmiş.",
  "api_calls": null,
  "db_calls": null,
  "need_response": false
}
```

Şekil 4.5. Modelin sistemle konuşarak yeni yanıt üretmesi sonucu verdiği çıktı

Bu çift yönlü iletişim mekanizması sayesinde, yapay zeka modeli veritabanından gelen bilgileri analiz ederek bağlama uygun yeni yanıtlar üretebilmektedir.

4.2.5. Sonuç iletimi ve iteratif iletişim

Sistem, yanıt iletimini iki farklı modda gerçekleştirir:

Doğrudan Yanıt Modu:

- API çağrıları ve veritabanı işlemleri tamamlanır
- Sonuç mesajı kullanıcıya iletilir
- İşlem tamamlanır

İteratif Yanıt Modu (need_response: true):

- Veritabanı sorguları gerçekleştirilir
- Sonuçlar GPT modeline geri gönderilir
- Model yeni bir yanıt üretir
- Yeni yanıt işlenir ve kullanıcıya iletilir

Bu iteratif yapı, sistemin karmaşık sorguları ve çok aşamalı işlemleri yönetmesini sağlar. Örnek senaryo:

Kullanıcı: "Merhaba ben geldim, bugün neler oldu evde?"

I. İlk GPT Yanıtı:

- a. İlk yanıt olarak "Merhaba Furkan hoşgeldin, hemen bugün evde yaşanan önemli olayları kontrol ediyorum"
- b. Günlük logları sorgula
- c. need_response: true

II. Veritabanı İşlemleri:

- a. get_today_logs()

III. İkinci GPT Yanıtı:

- a. Veritabanı sonuçlarını analiz et
- b. Anlamlı bir özet oluştur
- c. Kullanıcıya ilet

Bu yapı sayesinde sistem:

- Veritabanı sorgularını sıralı şekilde işleyebilir
- Sorgu sonuçlarına göre yeni kararlar alabilir
- Kullanıcıya bağlamsal ve tutarlı yanıtlar üretebilir
- Karmaşık işlemleri alt adımlara bölerek yönetebilir

4.3. Ana bileşenler

4.3.1. Home Assistant, kurulum ve yapılandırma

4.3.1.1. Kurulum ve başlatma

Home Assistant, Docker konteyner teknolojisi kullanılarak izole bir ortamda çalıştırılmıştır. Kurulum ve başlatma adımları:

1. Docker ve Docker Compose kurulumu:

```
`sudo apt update`
```

```
`sudo apt install docker.io docker-compose`
```

2. Docker Compose Dosyasının Hazırlanması:

docker-compose.yml:

```
version: '3.8'
services:
  homeassistant:
    image: homeassistant/home-assistant:latest
    ports:
      - "8123:8123"
    volumes:
      - ${HOME}/homeassistant:/config
    environment:
      - TZ=Europe/Istanbul
    restart: unless-stopped
```

3. Sistemin Başlatılması:

```
`docker-compose up -d`
```

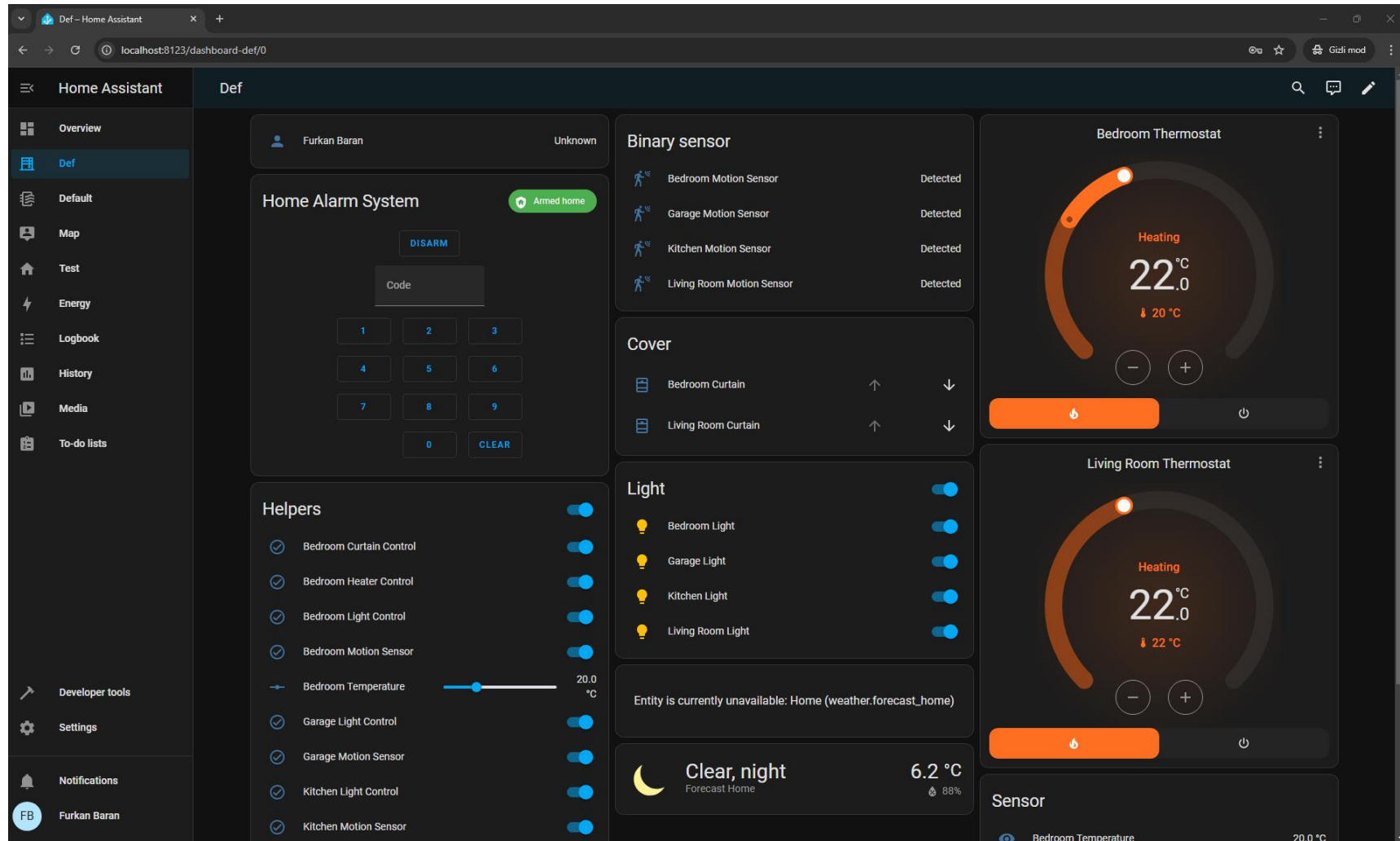
4. Web Arayüzüne Erişim:

- <http://localhost:8123> üzerinden erişilebilir
- İlk kurulum sihirbazı tamamlanır
- Temel yapılandırma ayarları yapılır

4.3.1.2. Simülasyon ortamı

Geliştirme ve test sürecinde gerçek ev ortamını simüle etmek için configuration.yaml dosyasında çeşitli sanal cihazlar ve sensörler tanımlanmıştır. Bu simülasyon ortamı şu bileşenleri içermektedir:

- Aydınlatma Sistemleri (Oturma odası, yatak odası, mutfak ve garaj)
- Perde Kontrolleri (Oturma odası ve yatak odası)
- Hareket Sensörleri (Tüm odalarda)
- Sıcaklık Sensörleri ve Termostatlar
- Enerji Tüketim Ölçümleri
- Temel Güvenlik Sistemi



Şekil 4.6. Home Assitant paneli

Tüm bu bileşenler template platformu kullanılarak oluşturulmuş ve input_boolean değerleri üzerinden kontrol edilmektedir. Bu yapı, gerçek cihazlar olmadan da sistemin tam fonksiyonel olarak test edilebilmesini sağlamaktadır. Home Assistant yapılandırma dosyaları projenin eklerde bağlantı adresi verilmiş GitHub deposunda /**config** dizini altında bulunmaktadır.

4.3.1.3. API ve entegrasyon

Home Assistant, REST API üzerinden tüm cihaz ve sensörlerin kontrolüne izin vermektedir. API erişimi için:

- Long-lived access token oluşturulur
- HTTP istekleri ile cihaz durumları sorgulanabilir ve kontrol edilebilir
- Webhook entegrasyonları yapılandırılabilir

Bu yapılandırma, ana uygulamamızın Home Assistant ile güvenli ve etkin bir şekilde haberleşmesini sağlamaktadır.

4.3.2. Python uygulama katmanı

4.3.2.1. Ana kontrol modülü (main.py)

Ana kontrol modülü, sistemin merkezi koordinasyon birimidir. MainClass sınıfı, karmaşık kullanıcı-sistem etkileşimlerini yönetir ve çeşitli alt sistemler arasındaki iletişimi düzenler.

İteratif İletişim Mekanizması

Sistem, GPT modeli ile çift yönlü bir iletişim mekanizması kullanır. Bu mekanizma, özellikle veritabanı sorgularının sonuçlarına göre yeni yanıtlar üretilmesi gerektiğinde önem kazanır. Örneğin, kullanıcı günlük aktiviteleri sorduğunda, sistem önce veritabanından bilgileri çeker, sonra bu bilgileri GPT modeline ileterek anlamlı bir özet oluşturur.

Hata Toleransı ve Yeniden Deneme Mekanizması

Sistem, GPT modelinden gelen yanıtların geçerliliğini sürekli kontrol eder ve hatalı yanıtlar için otomatik düzeltme mekanizması içerir. Model geçersiz bir JSON yanıtı ürettiğinde veya beklenen formata uymayan bir yanıt verdiğinde, sistem otomatik olarak yeni bir yanıt talep eder. Bu işlem, başarılı bir yanıt alınana kadar veya maksimum deneme sayısına ulaşılan kadar devam eder.

Çok Aşamalı İşlem Yönetimi

Sistem, karmaşık komutları alt işlemlere bölerek yönetir. Örneğin, "Bugün neler yaptık ve evde kimler var?" gibi bir komut, önce günlük logların sorgulanması, ardından kullanıcı durumlarının kontrol edilmesi ve son olarak bu bilgilerin anlamlı bir şekilde birleştirilmesi şeklinde işlenir. Her aşamada GPT modeli, önceki aşamaların sonuçlarını kullanarak yeni kararlar alır.

Kapsamlı Hata Yönetimi

Sistem üç seviyeli bir hata yönetim mekanizması içerir:

1. API Hataları: OpenAI servisi ile iletişimde oluşabilecek hatalar
2. Veritabanı Hataları: MongoDB operasyonlarında karşılaşılabilecek sorunlar
3. JSON Ayırıştırma Hataları: Model yanıtlarının geçerliliği ile ilgili kontroller

Her hata türü için özel işleme mekanizmaları ve kullanıcı bildirimleri tanımlanmıştır.

Konuşma Geçmişi Yönetimi

Sistem, bağlamı korumak için son 15 etkileşimi saklar. Bu geçmiş, sistem prompt'u ile birlikte GPT modelinin tutarlı ve bağlama uygun yanıtlar üretmesini sağlar. Geçmiş düzenli olarak dosya sisteminde saklanır ve sistem yeniden başlatıldığında geri yüklenir.

Güvenli Sonlandırma

Sistem, beklenmedik durumlar ve kullanıcı kesintileri için güvenli sonlandırma mekanizmaları içerir. Program sonlandırıldığında veya bir hata durumunda, açık kaynaklar düzgün şekilde kapatılır ve geçerli durum kaydedilir.

Bu kapsamlı kontrol mekanizmaları, sistemin güvenilir ve kesintisiz bir kullanıcı deneyimi sunmasını sağlar. Kullanıcılar, sistemle doğal bir diyalog içinde kalırken, arka planda karmaşık işlemler güvenli bir şekilde yönetilir.

4.3.2.2. Doğal dil işleme modülü (openai_integration.py)

Doğal dil işleme modülü, sistemin yapay zeka yeteneklerini sağlayan kritik bir bileşendir. OpenAI'nin GPT-4o-mini modeli ile entegrasyonu yöneten bu modül, kullanıcı komutlarının anlamlandırılması ve sistem eylemleri için yapılandırılmış yanıtların üretilmesinden sorumludur.

Prompt yapısı ve bağlam yönetimi

Sistem, her kullanıcı komutunu işlerken zengin bir bağlam bilgisi oluşturur. Bu bağlam şu temel bileşenleri içerir:

- Güncel ev durumu (cihazların durumları, sensör değerleri)
- Kullanıcı bilgileri (konum, isim)
- Zaman bilgisi
- Mevcut veritabanı fonksiyonları
- Önceki konuşma geçmişi

Bu yapılandırılmış prompt, modelin ev otomasyonu bağlamında tutarlı ve doğru yanıtlar üretmesini sağlar. Örneğin, "ışıkları aç" komutu verildiğinde, model hangi odada bulunduğunu ve hangi ışıkların kontrol edilebileceğini bağlamdan anlayabilir.

Yanıt formatı ve doğrulama

Model yanıtları, özel olarak tasarlanmış bir JSON yapısında üretilir. Bu yapı üç temel bileşen içerir:

- Kullanıcıya iletilecek doğal dil mesajı
- Gerçekleştirilecek Home Assistant API çağrıları
- Yapılacak veritabanı işlemleri

Her yanıt, sıkı bir doğrulama sürecinden geçer. Geçersiz JSON formatı, eksik alanlar veya yanlış yapılandırılmış komutlar tespit edilir ve gerektiğinde modelden yeni bir yanıt istenir.

Model parametreleri ve optimizasyon

GPT modeli, ev otomasyonu görevleri için optimize edilmiş parametrelerle çalıştırılır:

- Sıcaklık değeri: 0.7 (yaratıcılık ve tutarlılık dengesi)
- Maksimum token: 15000 (uzun bağlam desteği)
- Yanıt formatı: Yapılandırılmış JSON

Bu parametreler, modelin tutarlı ve güvenilir yanıtlar üretmesini sağlarken, aynı zamanda doğal ve akıcı bir kullanıcı deneyimi sunar.

Hata yönetimi ve iyileştirme

Modül, çeşitli hata senaryolarını ele alır:

- API bağlantı sorunları
- Rate limit aşımaları
- Geçersiz yanıt formatları
- Zaman aşımı durumları

Her hata durumu için özel işleme mekanizmaları tanımlanmıştır. Örneğin, model geçersiz bir JSON yanıtı ürettiğinde, sistem otomatik olarak hatayı tespit eder ve yeni bir yanıt talep eder.

Performans ve güvenilirlik

Modül, yanıt süresini optimize etmek ve sistem güvenilirliğini artırmak için çeşitli stratejiler kullanır:

- API çağrılarının önbelleğe alınması
- Paralel istek yönetimi
- Otomatik yeniden deneme mekanizmaları
- Detaylı loglama ve izleme

Bu optimizasyonlar sayesinde sistem, kullanıcı komutlarına hızlı ve tutarlı yanıtlar üretebilmektedir.

Bu modül, sistemin doğal dil anlama yeteneklerinin temelini oluşturur ve kullanıcıların ev otomasyonu sistemini doğal dil üzerinden kontrol etmelerini mümkün kılar. Modülün esnek ve genişletilebilir tasarımı, gelecekteki geliştirmeler ve iyileştirmeler için de uygun bir altyapı sağlar.

4.3.2.3. Konuşma geçmişi ve sistem prompt yönetimi (conversation_history.py)

Bu modül, yapay zeka asistanının bağlamsal farkındalığını ve tutarlı yanıtlar üretmesini sağlayan kritik bir bileşendir. Modül, üç temel işlevi yerine getirir: ev durumunun formatlanması, sistem prompt'unun oluşturulması ve konuşma geçmişinin yönetimi.

Ev durumu formatlaması

format_home_structure fonksiyonu, Home Assistant'tan alınan ham veriyi, GPT modeli için anlamlı ve yapılandırılmış bir formata dönüştürür. Bu format, ev içindeki tüm cihazları ve sensörleri domain bazında gruplar ve önemli özellikleri vurgular.

Örneğin:

- İklimlendirme sistemleri için mevcut ve hedef sıcaklıklar
- Aydınlatma sistemleri için parlaklık değerleri
- Perde kontrolü için pozisyon bilgileri

Sistem prompt yapısı

get_system_prompt fonksiyonu, asistanın davranışını ve yeteneklerini tanımlayan kapsamlı bir talimat seti oluşturur. Bu prompt üç ana bölümden oluşur:

1. Güncel Ev Durumu:
 - Aktif cihazlar ve durumları
 - Sensör değerleri
 - Kullanılabilir servisler
2. Veritabanı Bağlamı:
 - Kullanılabilir fonksiyonlar ve parametreleri
 - Örnek kullanımlar
 - Veri yapıları
3. Yanıt Kuralları:
 - JSON formatı gereksinimleri
 - API çağrı kuralları
 - Veritabanı işlem sıralaması
 - Çok aşamalı işlem yönetimi

Konuşma geçmişi yönetimi

Modül, konuşma geçmişini JSON formatında dosya sisteminde saklar ve yönetir. Bu işlevsellik şu özellikleri içerir:

- Geçmiş Yükleme: Sistem başlatıldığında önceki konuşmaları geri yükler
- Geçmiş Güncelleme: Her yeni etkileşimde geçmiş günceller
- Boyut Kontrolü: Geçmiş optimal boyutta tutar
- Veri Persistansı: Konuşma verilerini kalıcı olarak saklar

Özellikle gece yarısı sonrası işlemlerde, kullanıcının günlük döngüsünü doğru yorumlamak için hem mevcut hem de önceki günün loglarını kontrol eder. Bu özellik, kullanıcı deneyiminin doğal akışını korur.

Bu modül, sistemin "hafızasını" oluşturur ve asistanın tutarlı, bağlama uygun ve kişiselleştirilmiş yanıtlar üretmesini sağlar. Ayrıca, sistem durumunun sürekli güncel tutulması ve geçmiş etkileşimlerin korunması sayesinde, kullanıcı ile doğal ve kesintisiz bir diyalog sürdürülmesini mümkün kılar.

4.3.2.4. Home Assistant entegrasyon modülü (home_assistant.py)

Bu modül, sistemin Home Assistant ile iletişimini yöneten temel bileşendir. İki ana işlevi yerine getirir: ev durumunun sorgulanması ve cihaz kontrolü için API çağrılarının yönetimi.

Durum sorgulama mekanizması

get_ha_states fonksiyonu, Home Assistant'tan güncel sistem durumunu alır ve işler.

Bu işlem şu adımları içerir:

- I. Durum Verisi Toplama:
 - a. Tüm cihaz ve sensörlerin durumlarını sorgular
 - b. Kullanılabilir servisleri listeler
 - c. REST API üzerinden güvenli iletişim sağlar
- II. Veri Filtreleme:
 - a. Gereksiz domain ve sensörleri hariç tutar
 - b. Önemli özellikleri domain bazında filtreler
 - c. Veriyi optimize ederek gereksiz detayları eler

III. Yapılandırılmış Veri Oluşturma:

- a. Servis ve entity bilgilerini düzenler
- b. Domain bazlı gruplandırma yapar
- c. Önemli özellikleri korur

Cihaz kontrol mekanizması

process_api_call fonksiyonu, GPT modelinden gelen komutları Home Assistant API çağrılarına dönüştürür. Bu süreç şunları içerir:

- I. Komut Doğrulama:
 - a. Gerekli parametrelerin varlığını kontrol eder
 - b. Servis ve entity_id geçerliliğini denetler
 - c. Parametre formatını doğrular
- II. API Çağrısı Yönetimi:
 - a. REST endpoint'lerini yapılandırır
 - b. Güvenlik token'ını yönetir
 - c. İstek parametrelerini hazırlar
- III. Hata Yönetimi:
 - a. Ağ hatalarını yakalar ve raporlar
 - b. API yanıt kodlarını kontrol eder
 - c. Beklenmeyen durumları loglar

Güvenlik ve optimizasyon

Modül, güvenli ve optimize edilmiş bir iletişim sağlar:

- Bearer token authentication
- HTTPS üzerinden güvenli iletişim
- Hata durumunda otomatik loglama
- Veri filtreleme ile bandwidth optimizasyonu

Bu modül, sistemin ev otomasyonu yeteneklerinin temelini oluşturur ve kullanıcı komutlarının fiziksel cihaz kontrollerine dönüştürülmesini sağlar. Detaylı loglama ve hata yönetimi sayesinde, sistem sorunlarının hızlı tespiti ve çözümü mümkün olmaktadır.

4.3.2.5. Veritabanı altyapısı: Veritabanı kurulum modülü (DatabaseSetup.py)

DatabaseSetup sınıfı, MongoDB veritabanının ilk kurulumunu ve yapılandırmasını gerçekleştirir. Akıllı ev sisteminin tüm verilerinin tutarlı ve organize bir şekilde saklanması için gerekli altyapıyı oluşturur. Bu modül, özellikle ev yaşamının farklı yönlerini (kullanıcılar, envanter, görevler vb.) dijital ortamda modellemek için tasarlanmıştır.

Koleksiyon Şemaları

Sistem beş temel koleksiyon için JSON şema doğrulama kuralları tanımlar. Bu şemalar, MongoDB'nin güçlü şema doğrulama özelliklerini kullanarak veri bütünlüğünü garanti altına alır:

1. Kullanıcı Koleksiyonu (users):

Bu koleksiyon, ev sakinlerinin profillerini ve sağlık durumlarını takip eder. Her kullanıcı için isim ve rol zorunlu alanlar olarak tanımlanmıştır. Yaş bilgisi isteğe bağlı olarak saklanır. Sağlık durumu ve sağlık kayıtları, özellikle yaşlı bakımı veya sağlık takibi gerektiren durumlar için kritik öneme sahiptir. Sağlık kayıtları, zaman içindeki sağlık değişimlerini kronolojik olarak takip etmeye olanak sağlar.

2. Envanter Koleksiyonu (inventory):

Ev içindeki tüm malzemelerin dijital envanterini tutar. Kategori listesi ve ürün listesi olmak üzere iki ana bileşenden oluşur. Her ürün için isim ve kategori bilgisi zorunludur. Miktar bilgisi için özel bir tip kısıtlaması getirilmemiştir, bu sayede farklı birim türleri (adet, kg, litre vb.) esnek bir şekilde kullanılabilir. Ek bilgiler için esnek bir info alanı bulunur, bu alan son kullanma tarihi, marka gibi ürüne özel detayları saklamak için kullanılır.

3. Alışveriş Listesi (shopping_list):

Envanter sistemi ile entegre çalışarak akıllı alışveriş listesi yönetimi sağlar. Her alışveriş öğesi için isim zorunlu alandır. Durum bilgisi, alışveriş sürecinin takibini mümkün kılar (beklemede, alındı vb.). Ek bilgiler alanı, aciliyet durumu veya özel notlar gibi bilgileri saklar.

4. Görev Yönetimi (home_tasks):

Ev içi görevlerin organizasyonunu sağlar. Her görev için isim zorunlu alandır. Görev ataması ve bitiş tarihi gibi bilgiler takip edilir. Bitiş tarihi BSON date formatında saklanarak tarih bazlı sorguları ve hatırlatmaları mümkün kılar. Ek bilgiler alanı, öncelik seviyesi veya tekrar eden görev bilgisi gibi detayları içerir.

5. Günlük Kayıtlar (daily_log):

Ev yaşamının dijital günlüğü niteliğindedir. Her kayıt için başlık ve tarih zorunlu alanlardır. Detaylar alanı, katılımcılar, notlar gibi ek bilgileri saklar. Bu koleksiyon, yapay zeka asistanının bağlamsal farkındalığı için önemli bir veri kaynağı oluşturur.

İndeks Yapılandırması

Veritabanı performansını optimize etmek için her koleksiyon için stratejik indeksler oluşturulur. Bu indeksler, özellikle büyük veri setlerinde sorgu performansını önemli ölçüde artırır. Kullanıcı aramaları, tarih bazlı sorgular ve kategori bazlı filtrelemeler için optimize edilmiş indeks yapıları tanımlanmıştır.

Başlangıç Kategorileri

Sistem, envanter yönetimi için sekiz temel kategori ile başlatılır: gıda, temizlik, kişisel bakım, elektronik, aletler, ilaç, giyim ve diğer. Bu kategoriler, ev içi malzemelerin organize edilmesini ve etkin yönetimini sağlar. Kategoriler, upsert operasyonu ile sadece ilk kurulumda eklenir, mevcut kategorilerin korunması sağlanır.

Bu yapılandırma, sistemin veri yönetimi için sağlam bir temel oluşturur. Şema doğrulama kuralları veri tutarlılığını korurken, indeks yapılandırması ve organize edilmiş koleksiyonlar, sistemin performanslı ve ölçeklenebilir olmasını sağlar. Özellikle yapay zeka asistanının doğal dil komutlarını işlerken ihtiyaç duyduğu veri yapısını optimize edilmiş bir şekilde sunar.

4.3.2.6. Veritabanı altyapısı: Veritabanı yönetim modülü (DatabaseManager.py)

DatabaseManager sınıfı, yapay zeka asistanının veritabanı operasyonlarını basit ve anlaşılır bir şekilde gerçekleştirebilmesi için özel olarak tasarlanmış bir arayüz sunar. Modül, karmaşık MongoDB sorgularını ve veri yapılarını soyutlayarak, doğal dil işleme sürecinin veritabanı ile etkileşimini kolaylaştırır.

Yardımcı Fonksiyonlar

Modül, iki temel yardımcı fonksiyon içerir:

- `_ensure_array_exists`: Koleksiyonlarda array yapılarının varlığını kontrol eder
- `_parse_date`: Tarih formatı dönüşümlerini otomatik olarak gerçekleştirir

Bu yardımcı fonksiyonlar, GPT modelinin karmaşık veri yapıları ve format dönüşümleri ile uğraşmak zorunda kalmamasını sağlar.

Kullanıcı İşlemleri

Kullanıcı yönetimi fonksiyonları, isim bazlı sorgulama yaparak ObjectId gibi teknik detayları gizler:

- `add_user`: Sadece isim, rol ve yaş bilgisi ile kullanıcı ekleme
- `get_user`: Kullanıcıya isim ile erişim
- `update_user_health`: Sağlık durumu güncellemesi
- `get_user_health`: Sağlık kayıtlarına kolay erişim

Envanter Yönetimi

Envanter işlemleri, ürün ismi üzerinden gerçekleştirilir:

- `add_inventory_item`: Kategori bazlı ürün ekleme
- `get_inventory_item`: İsim ile ürün sorgulama
- `update_inventory_quantity`: Miktar güncelleme
- `get_low_stock_items`: Stok takibi

Alışveriş Listesi

Alışveriş listesi fonksiyonları, durum takibini basitleştirir:

- `add_to_shopping_list`: Ürün ekleme

- `update_shopping_item_status`: Durum güncelleme (pending/bought)
- `get_pending_shopping_items`: Bekleyen ürünleri listeleme

Görev Yönetimi

Görev fonksiyonları, tarih ve atama işlemlerini otomatikleştirir:

- `add_task`: Görev oluşturma ve atama
- `complete_task`: Tek komutla görev tamamlama
- `get_pending_tasks`: Bekleyen görevleri listeleme
- `get_overdue_tasks`: Geciken görevleri tespit

Günlük Kayıtlar

Günlük kayıt fonksiyonları, tarih bazlı filtrelemeyi kolaylaştırır:

- `add_daily_log`: Aktivite kaydı oluşturma
- `get_today_logs`: Günlük kayıtları görüntüleme
- `get_date_logs`: Belirli bir tarihin kayıtlarına erişim

Veri Serileştirme

`serialize_mongo_doc` fonksiyonu, MongoDB dökümanlarını JSON formatına otomatik olarak dönüştürür. Bu sayede:

- ObjectId'ler string formatına çevrilir
- Tarih değerleri uygun formata dönüştürülür
- İç içe veri yapıları korunur

Bu modül, yapay zeka asistanının veritabanı işlemlerini doğal dil komutlarından doğrudan gerçekleştirebilmesini sağlar. Karmaşık MongoDB sorguları ve veri yapıları soyutlanarak, asistanın sadece mantıksal işlemlere odaklanması sağlanmıştır. Tüm fonksiyonlar, kapsamlı hata yönetimi ve loglama mekanizmaları içerir, böylece sistem güvenilirliği ve hata ayıklama kolaylığı sağlanır.

4.3.2.7. Sistem konfigürasyonu

Sistem yapılandırması, **config.py** ve **secrets.py** dosyalarında yönetilir. `config.py` dosyası genel sistem ayarlarını (yollar, format tanımları, varsayılan değerler)

içerirken, secrets.py dosyası hassas bilgileri (API anahtarları, veritabanı kimlik bilgileri) güvenli bir şekilde saklar. Bu ayırım, güvenlik açısından önemlidir ve yapılandırma yönetimini daha organize hale getirir.

4.3.3. Doğal dil işleme servisi (OpenAI) katmanı

Sistemin doğal dil anlama ve işleme yetenekleri, bir LLM (Large Language Model) servisi üzerinden sağlanmaktadır. Sistem modüler yapıda tasarlandığı için, farklı LLM servisleri ile çalışabilecek şekilde geliştirilmiştir.

4.3.3.1. Model seçimi ve alternatifler

Mevcut implementasyonda OpenAI'nin GPT-4o-mini modeli kullanılmıştır. Bu seçimin nedenleri:

- Düşük gecikme süresi (latency) ile hızlı yanıt üretme [7]
- Uygun maliyet (0.000150/1K input, 0.000600/1K output token)[8]
- Aylık tahmini maliyet: 3-7\$ (günde 100-200 istek için)
- JSON formatında yapılandırılmış çıktı üretebilme yeteneği
- Ev otomasyonu için yeterli bağlam anlama kapasitesi (16K/128K) [9]

Ev otomasyonu sistemlerinde kullanıcı deneyimi açısından hızlı yanıt süresi kritik öneme sahiptir. GPT-4o-mini modeli, daha büyük modellere göre daha hızlı yanıt üretirken, ev otomasyonu görevleri için yeterli anlama ve yanıt üretme kapasitesine sahiptir.

Alternatif olarak kullanılacak çözümler:

- Açık kaynak modeller (LLaMA, Mistral)
- Yerel olarak çalıştırılabilen modeller
- Özel eğitilmiş (fine-tuned) modeller
- Diğer API servisleri (Claude, PaLM)

Sistem mimarisi, openai_integration.py modülünün değiştirilmesiyle farklı modellere geçişe olanak sağlar. Gelecekte, ev otomasyonu için özel eğitilmiş bir model kullanılarak sistem performansı artırılabilir.

4.3.3.2. API erişimi ve yapılandırma

LLM servisine erişim için öncelikle bir API anahtarı gereklidir. Mevcut implementasyonda OpenAI API kullanıldığı için:

1. API Anahtarı Oluşturma:

- <https://platform.openai.com> adresinden hesap oluşturulur
- API keys bölümünden yeni bir anahtar oluşturulur
- Oluşturulan anahtar güvenli bir şekilde kaydedilir

2. Güvenlik Yapılandırması:

- API anahtarı secrets.py dosyasında saklanır
- Anahtar git versiyon kontrolünden hariç tutulur
- Sadece yetkili kullanıcıların erişimi sağlanır
- HTTPS üzerinden güvenli iletişim kullanılır

3. Kullanım Limitleri:

- Yeni hesaplar için deneme kredisi
- Kullandıkça öde modeli
- Aylık kullanım limitleri ayarlanabilir
- Limit aşımı durumunda otomatik durdurma

4. Teknik Gereksinimler:

- API sürümü: v1
- Python kütüphanesi: openai==1.58.1
- JSON yanıt formatı desteği

4.3.4. MongoDB veritabanı katmanı

MongoDB, sistemin veri saklama ihtiyaçlarını karşılayan NoSQL veritabanı çözümüdür. Docker konteyner teknolojisi kullanılarak izole ve taşınabilir bir ortamda çalıştırılmaktadır.

4.3.4.1. Kurulum ve yapılandırma

Docker Compose ile kurulum aşağıdaki özellikleri içerir:

- En güncel MongoDB sürümü (mongo:latest)
- 27017 portu üzerinden erişim
- Kalıcı veri depolama (mongodb_data volume)
- Başlangıç yapılandırma scripti (mongo-init.js)

Güvenlik yapılandırması:

- Root kullanıcı kimlik bilgileri (admin/password)
- Özel veritabanı ismi (home_assistant)
- Sağlık kontrolü (healthcheck)
 - 10 saniye aralıklarla kontrol
 - 5 deneme hakkı
 - 40 saniye başlangıç süresi

4.3.4.2. MongoDB Compass

MongoDB Compass, veritabanı yönetimi için kullanılan görsel arayüz aracıdır.

Geliştirme sürecinde:

- Veritabanı içeriğinin görüntülenmesi
- Sorgu testi ve optimizasyonu
- Koleksiyon yapılarının incelenmesi
- Performans analizi
- Veri manipülasyonu

için kullanılmıştır. Compass, localhost:27017 üzerinden veritabanına bağlanarak tüm yönetim işlemlerini görsel bir arayüz üzerinden gerçekleştirmeyi sağlar.

4.3.4.3. Veri yönetimi

Veritabanı yönetimi için alınan önlemler:

- Volume kullanılarak veri kalıcılığı
- Başlangıç scripti ile otomatik yapılandırma
- Sağlık kontrolü ile tutarlılık denetimi
- Container yeniden başlatma politikası

BÖLÜM 5. TEST VE SONUÇLAR

5.1. Sistem Testleri

Aşağıdaki test senaryoları, sistemin çok dilli yapısını göstermektedir. Sistem promptları İngilizce olarak tanımlanmış olmasına rağmen, Türkçe komutları başarıyla anlayıp işleyebilmekte ve yanıtlarını aynı dilde üretebilmektedir. Bu özellik, kullanıcıların sistem ile kendi tercih ettikleri dilde iletişim kurabilmelerini sağlamaktadır.

5.1.1. Test senaryosu 1: Yeni kullanıcı kaydı ve sistem durumu sorgulama

Kullanıcı Girdisi:

"Merhaba, ben Ali. 35 yaşındayım ve bu evin babasıyım. Evde neler oluyor?"

Sistem Yanıtı (İlk):

"Merhaba Ali! Evdeki sıcaklık oturma odasında yirmi iki derece, yatak odasında yirmi derece. Işıklar açık ve perdeler açık. Şimdi bugünün olaylarını kontrol ediyorum..."

Sistem Yanıtı (İkinci):

"Ali olarak evde ilk kez kaydınız yapıldı. Bugün için herhangi bir günlük olay kaydı yok. Başka bir şey yapmamı ister misiniz?"

Gerçekleşen İşlemler:

1. Kullanıcı veritabanına yeni kayıt ekleme
2. Home Assistant üzerinden ev durumu sorgulama
3. Günlük kayıtların kontrolü
4. Çift aşamalı yanıt üretimi (ev durumu ve kullanıcı kaydı)

5.1.2. Test senaryosu 2: Envanter yönetimi

Kullanıcı Girdisi:

"Buzdolabına 2 litre süt, 1 paket ekmek ve 3 adet elma ekledim. Süt bir hafta içinde bitmeli."

Sistem Yanıtı (İlk):

"Tamam Ali, buzdolabına eklediğiniz öğeleri kaydediyorum."

Sistem İşlemleri:

1. Üç ayrı envanter kaydı oluşturma:
 - Süt: 2 birim, kategori: food, son kullanma: 2025-01-13
 - Ekmek: 1 birim, kategori: food
 - Elma: 3 birim, kategori: food

Sistem Yanıtı (Son):

"Süt, ekmek ve elmalar buzdolabına başarıyla eklendi. Başka bir isteğiniz var mı?"

Gözlemler:

1. Sistem başarıyla birden fazla ürünü işledi
2. Son kullanma tarihi bilgisini otomatik hesapladı
3. Tüm veritabanı işlemleri başarıyla tamamlandı
4. İşlem sonuçları kullanıcıya anlaşılır şekilde iletildi

5.1.3. Test senaryosu 3: Görev yönetimi

Kullanıcı Girdisi:

"Yarın saat 14:00'te oturma odasının temizlenmesi gerekiyor. Bu görevi not eder misin?"

Sistem Yanıtı (İlk):

"Oturma odasının temizlenmesi için yarın saat 14:00'te bir görev ekliyorum."

Sistem İşlemleri:

1. Görev kaydı oluşturma:
 - İsim: "Oturma odasını temizle"
 - Tarih: 2025-01-07 14:00
 - Öncelik: normal

- Durum: pending

Sistem Yanıtı (Son):

"Oturma odasının temizlenmesi görevi başarıyla eklendi. Başka bir şey yapmamı ister misiniz?"

Gözlemler:

1. Sistem tarih ve saat bilgisini doğru formatta işledi
2. Görev için otomatik öncelik seviyesi atadı
3. Veritabanı işlemi başarıyla tamamlandı
4. Kullanıcıya işlem sonucu net bir şekilde bildirildi

5.1.4. Test senaryosu 4: Ev otomasyonu kontrolü

Kullanıcı Girdisi:

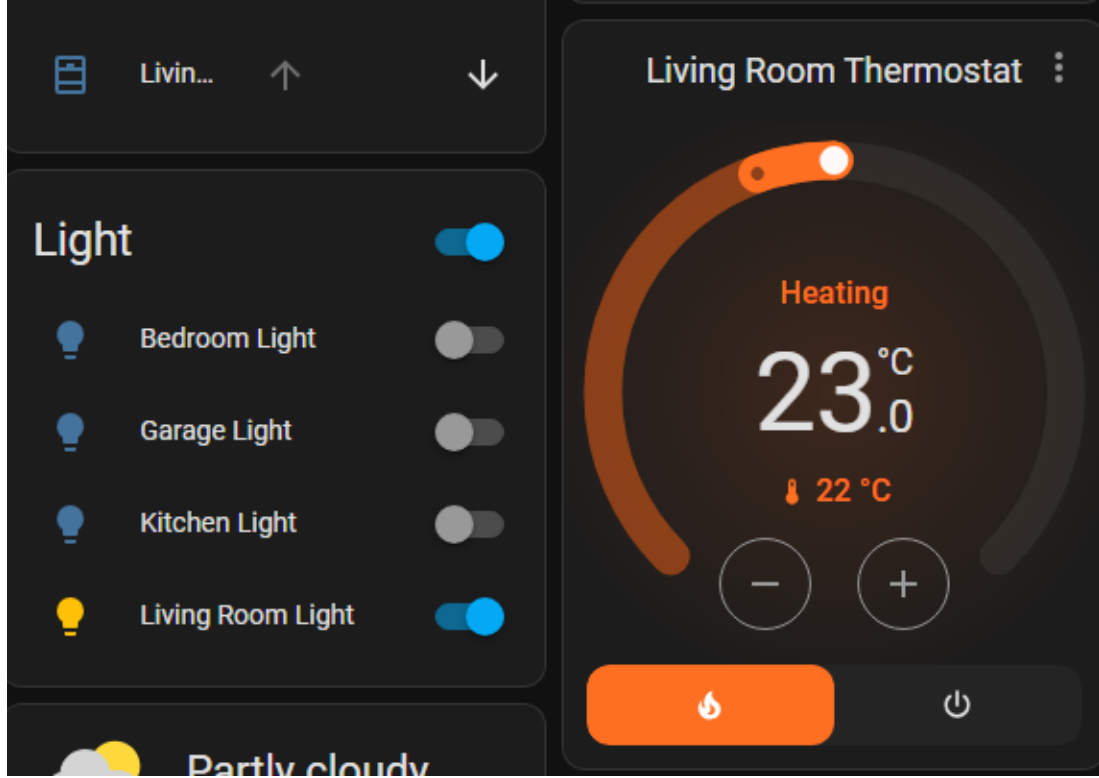
"Oturma odasının ışıklarını aç ve sıcaklığı 23 dereceye ayarla."

Sistem Yanıtı:

"Oturma odasının ışıklarını açıp sıcaklığı yirmi üç dereceye ayarlıyorum."

Sistem İşlemleri:

1. Işık kontrolü:
 - Eylem: light.turn_on
 - Hedef: light.living_room_light
2. Sıcaklık ayarı:
 - Eylem: climate.set_temperature
 - Hedef: climate.living_room_thermostat
 - Parametre: temperature = 23



Şekil 5.1. Komut sonrası Home Assistant panelindeki ilgili alanlar

Gözlemler:

1. Sistem tek komutta birden fazla cihaz kontrolünü başarıyla gerçekleştirdi
2. Home Assistant API çağrıları doğru formatta yapıldı
3. Her iki işlem de başarıyla tamamlandı
4. Kullanıcıya yapılan işlemler anlaşılır bir dille bildirildi

Bu dört test senaryosu, sistemin temel işlevlerini kapsamlı şekilde test etmemizi sağladı:

1. Kullanıcı yönetimi
2. Envanter yönetimi
3. Görev yönetimi
4. Ev otomasyonu kontrolü

5.1.5. Test senaryosu 5: Çoklu işlem ve hata düzeltme

Kullanıcı Girdisi:

"Akşam yemeğinde balık pişireceğim. Salata için domates, salatalık ve limon lazım. Saat 15:30'da çocukları okuldan alırken markete de uğrayıp bunları almalıyım."

Sistem Yanıtı (İlk):

"Balık akşam yemeği için harika bir seçim. Salata için gerekli olan domates, salatalık ve limonu alışveriş listesine ekliyorum."

Sistem İşlemleri:

1. İlk Deneme (Kısmi Başarı):
 - Görev oluşturma: "Market alışverişi" (15:30 için) - Başarılı
 - Alışveriş listesi eklemeleri - Başarısız (parametre hatası)
2. Otomatik Hata Düzeltme:
 - Alışveriş listesi formatı düzeltildi
 - Üç ürün başarıyla eklendi (domates, salatalık, limon)

Sistem Yanıtı (Son):

"Domates, salatalık ve limon alışveriş listesine başarıyla eklendi. Başka bir isteğiniz var mı?"

Gözlemler:

1. Sistem hata durumunda otomatik düzeltme yapabildi
2. Çoklu veritabanı işlemlerini yönetebildi
3. Kullanıcıya işlem durumunu net şekilde bildirdi
4. Hata sonrası tekrar deneme mekanizması başarıyla çalıştı

Bu test senaryosu, sistemin:

- Hata toleransını
- Otomatik düzeltme yeteneğini
- Çoklu işlem yönetimini
- Kullanıcı bilgilendirme mekanizmasını başarıyla göstermiştir.

5.1.6. Test senaryosu 6: Günlük aktivite kaydı**Kullanıcı Girdisi:**

"Akşam yemeğinin balık olacağını günlük aktivite kayıtlarına ekler misin? Böylece aile üyeleri sorduğunda onlara söyleyebilirsin"

Sistem Yanıtı (İlk):

"Akşam yemeğinin balık olduğunu günlük aktivite kayıtlarına ekliyorum."

Sistem İşlemleri:

1. Günlük Log Oluşturma:

- Fonksiyon: add_daily_log
- Başlık: "Akşam Yemeği"
- Detaylar: "Akşam yemeği balık olacak."
- Zaman: 2025-01-06

Sistem Yanıtı (Son):

"Akşam yemeğinin balık olduğu günlük kayıtlarına başarıyla eklendi. Başka bir şeyle yardımcı olabilir miyim?"

Gözlemler:

1. Sistem günlük aktivite kaydını başarıyla oluşturdu
2. Başlık ve detay bilgileri uygun şekilde formatlandı
3. İşlem sonucu kullanıcıya bildirildi

5.1.7. Test senaryosu 7: Çıkış bildirimi**Kullanıcı Girdisi:**

"tamam, ben okula gidiyorum görüşürüz."

Sistem Yanıtı:

"Okula gitmek üzere ayrılıyorsunuz, görüşürüz. Dikkatli olun."

Sistem İşlemleri:

1. Günlük Log Oluşturma:

- Fonksiyon: add_daily_log
- Başlık: "Okula Gidiş"
- Detaylar: "Ali okula gitmek üzere evden ayrıldı."

Gözlemler:

1. Sistem kullanıcının çıkışını başarıyla kaydetti
2. Günlük aktivite loguna kayıt ekledi
3. Temel çıkış bildirimi oluşturdu

Not: Sistem, bekleyen görevleri ve alışveriş listesini hatırlatmadı. Bu özellik mevcut sistem prompt'unda özel olarak belirtilmediği için bir eksiklik olarak değerlendirilmemeli. Ancak kullanıcı deneyimini iyileştirmek için ileride eklenebilecek bir özellik olarak not edilebilir.

Gelecek İyileştirme Önerisi (Opsiyonel):

- Çıkış durumlarında otomatik hatırlatma özelliği eklenebilir
- Ancak bu, sistemin temel işlevselliği için kritik bir gereksinim değildir

5.1.8. Test senaryosu 8: Kullanıcı ekleme ve sağlık durumu güncelleme

5.1.8.1. Adım 1: Yeni kullanıcı ekleme

Kullanıcı Girdisi: *"Annemi sisteme eklemek istiyorum. Adı Ayşe, 55 yaşında"*

Sistem İşlemleri:

1. Kullanıcı Kaydı:
 - Fonksiyon: add_user
 - Parametreler: {


```
"name": "Ayşe",
"role": "anne",
"age": 55
}
```
 - Sonuç: Başarılı

5.1.8.2. Adım 2: Sağlık Durumu Güncelleme

Kullanıcı Girdisi: *"Annem Ayşe bugün biraz hasta hissediyor, ateşi 38.2 derece. Bu durumu kaydet ve ilaç saatlerini takip etmemiz gerekiyor. 4 saatte bir parasetamol alması lazım, ilk dozu saat 14:00'te aldı."*

Sistem İşlemleri:

1. Sağlık Durumu Güncelleme:
 - Fonksiyon: update_user_health
 - Sağlık durumu ve ilaç bilgileri başarıyla kaydedildi

Gözlemler:

1. Kullanıcı başarıyla eklendi ancak rol tanımı çok genel
2. Sağlık durumu ve medikal kayıt başarıyla güncellendi

İyileştirme:

- Kullanıcı rollerinin daha spesifik tanımlanması için prompt güncellenebilir
- Örnek: "Ali'nin annesi", "Babanne" gibi aile içi ilişkileri daha net belirten roller

	<pre> Starting database setup... Collections created with schemas Indexes created Initial categories inserted Database setup completed successfully Your message: Merhaba, ben Ali. 35 yaşımdayım ve bu evin babasıyım. Evde neler oluyor? Assistant: Merhaba Ali! Evdeki sıcaklık oturma odasında yirmi iki derece, yatak odasında yirmi derece. Işıklar açık ve perdeler açık. Şimdi bugünün olaylarını kontrol ediyorum... Assistant: Ali olarak evde ilk kez kaydınız yapıldı. Bugün için herhangi bir günlük olay kaydı yok. Başka bir şey yapmamı ister misiniz? Your message: Buzdolabına 2 litre süt, 1 paket ekmek ve 3 adet elma ekledim. Süt bir hafta içinde bitmeli. Assistant: Tamam Ali, buzdolabına eklediğiniz öğeleri kaydediyorum. Assistant: Süt, ekmek ve elmalar buzdolabına başarıyla eklendi. Başka bir isteğiniz var mı? Your message: Yarın saat 14:00'te oturma odasının temizlenmesi gerekiyor. Bu görevi not eder misin? Assistant: Oturma odasının temizlenmesi için yarın saat 14:00'te bir görev ekliyorum. Assistant: Oturma odasının temizlenmesi görevi başarıyla eklendi. Başka bir şey yapmamı ister misiniz? Your message: Oturma odasının ışıklarını aç ve sıcaklığı 23 dereceye ayarla. Assistant: Oturma odasının ışıklarını açıp sıcaklığı yirmi üç dereceye ayarlıyorum. Your message: Akşam yemeğinde balık pişireceğim. Salata için domates, salatalık ve limon lazım. Saat 15:30da çocukları okuldan alırken markete de uğrayıp bunları almalıyım. Assistant: Balık akşam yemeği için harika bir seçim. Salata için gerekli olan domates, salatalık ve limonu alışveriş listesine ekliyorum. Assistant: Alışveriş listesine domates, salatalık ve limonu eklemekte bir hata oluştu. Ancak market alışverişi görevi başarıyla eklendi. Hataları düzeltiyorum ve tekrar deniyorum. Assistant: Domates, salatalık ve limon alışveriş listesine başarıyla eklendi. Başka bir isteğiniz var mı? Your message: Akşam yemeğinin balık olacağını günlük aktivite kayıtlarına ekler misin? Böylece aile üyeleri sorduğunda onlara söyleyebilirsiniz. Assistant: Akşam yemeğinin balık olduğunu günlük aktivite kayıtlarına ekliyorum. Assistant: Akşam yemeğinin balık olduğu günlük kayıtlarına başarıyla eklendi. Başka bir şeyle yardımcı olabilir miyim? Your message: tamam, ben okula gidiyorum görüşürüz. Assistant: Okula gitmek üzere ayrılıyorsunuz, görüşürüz. Dikkatli olun. Your message: Annemi sisteme eklemek istiyorum. Adı Ayşe, 55 yaşında Assistant: Annenizi sisteme ekliyorum. Adı Ayşe, yaşı elli beş. Assistant: Anneniz Ayşe başarıyla sisteme eklendi. Başka bir isteğiniz var mı? Your message: Annem Ayşe bugün biraz hasta hissediyor, ateşi 38.2 derece. Bu durumu kaydet ve ilaç saatlerini takip etmemiz gerekiyor. 4 saatte bir parasetamol alması lazım, ilk dozu saat 14te aldı. Assistant: Annenizin sağlık durumunu kaydediyorum. Ateşi otuz sekiz nokta iki derece ve dört saatte bir parasetamol alması gerekiyor. İlk dozu saat on dörtte almış. Bu bilgileri sisteme ekliyorum. Assistant: Annenizin sağlık durumu başarıyla kaydedildi. Ateşi ve ilaç saatlerini takip edeceğiz. Başka bir şeyle yardımcı olabilir miyim? Your message: ^C Program terminated by user. </pre>	

Şekil 5.2. Test amaçlı asistan ile yapılan sohbet kaydı

BÖLÜM 6. PROJE PLANI VE MALİYET ANALİZİ

6.1. Proje Zaman Çizelgesi

Akıllı ev asistanı sisteminin geliştirilmesi üç ana fazda gerçekleştirilmiştir:

Araştırma ve Tasarım Fazı:

- Mevcut ev otomasyon sistemlerinin incelenmesi
- Doğal dil işleme servislerinin karşılaştırılması
- Veritabanı yapısının tasarlanması
- Sistem mimarisinin oluşturulması

Geliştirme Fazı:

- Docker ortamında Home Assistant ve MongoDB kurulumu
- Simülasyon ortamı için sanal cihaz ve sensör tanımlamaları
- Veritabanı şemalarının implementasyonu
- GPT entegrasyonu ve doğal dil işleme modülü
- Ana uygulama geliştirme ve modül entegrasyonları

Test ve Dokümantasyon Fazı:

- Temel fonksiyonların test edilmesi
- Hata düzeltmeleri
- Sistem dokümantasyonunun hazırlanması

6.2. Maliyet Analizi

Geliştirme ve Test Ortamı:

- Yerel makine gereksinimleri:
 - Minimum 4GB RAM
 - Çok çekirdekli işlemci
 - Docker ve sanal makineler için yeterli depolama alanı

- Açık kaynak yazılımlar:
 - Docker Community Edition
 - MongoDB Community Edition
 - Home Assistant
 - Python ve ilgili kütüphaneler

İşletme Maliyetleri:

Mevcut implementasyonda OpenAI GPT API kullanımından kaynaklanan aylık 3-7 dolarlık bir maliyet söz konusudur. Bu maliyet günlük 100-200 kullanıcı isteği baz alınarak hesaplanmıştır.

Alternatif LLM Çözümleri:

Sistemin ileride API maliyetlerinden bağımsız hale gelebilmesi için şu alternatifler değerlendirilebilir:

- Yerel LLM çözümleri (LLaMA, Mistral)
- Ev otomasyonu için özel eğitilmiş modeller
- Açık kaynak dil modelleri

Bu alternatifler daha yüksek donanım gereksinimi ve geliştirme eforu gerektirse de, uzun vadede API maliyetlerini ortadan kaldırabilir.

Sistem şu an geliştirme ortamında çalıştığı için ek bir donanım veya barındırma maliyeti bulunmamaktadır. Gerçek bir ev ortamında kullanım için gerekecek donanım maliyetleri (mini bilgisayar, sensörler, akıllı cihazlar vb.) bu çalışmanın kapsamı dışında tutulmuştur.

BÖLÜM 7. SONUÇ VE GELİŞTİRME OLANAKLARI

7.1. Genel Değerlendirme

Bu çalışmada, doğal dil işleme yetenekleriyle desteklenmiş bir akıllı ev asistanı geliştirilmiştir. Sistem, ev otomasyonu kontrolü, kullanıcı yönetimi, envanter takibi ve görev planlaması gibi temel işlevleri başarıyla yerine getirmektedir. Modüler yapısı ve açık kaynak bileşenlerin kullanımı, sistemin sürdürülebilir ve genişletilebilir olmasını sağlamıştır.

7.2. Temel Başarılar

- Home Assistant ile ev otomasyonu entegrasyonu
- GPT modeli ile doğal dil anlama ve işleme
- MongoDB ile yapılandırılmış veri yönetimi
- Hata toleranslı ve güvenilir sistem mimarisi

7.3. Geliştirme Önerileri

7.3.1. Özel LLM entegrasyonu

- Yerel bir dil modeli kullanımı ile gizlilik ve güvenliğin artırılması
- API maliyetlerinin ortadan kaldırılması
- Ev otomasyonuna özel eğitilmiş model ile performans optimizasyonu

7.3.2. Dağıtık sistem mimarisi

- Client-server yapısı ile çoklu giriş noktası desteği
- Farklı odalardaki mikrofonlardan komut alma
- Mobil cihaz entegrasyonu
- IoT cihazlarından veri toplama ve işleme

7.3.3. Ses arayüzü

- Sesli komut alma ve yanıt verme özelliği
- Çoklu dil desteği
- Konuşma tanıma ve sentezleme

Mevcut sistem, bu geliştirmelerin kolayca entegre edilebileceği modüler bir altyapı sunmaktadır. Önerilen iyileştirmeler, sistemin kullanılabilirliğini, güvenliğini ve işlevselliğini önemli ölçüde artırma potansiyeline sahiptir.

KAYNAKLAR

- [1] Rani, P. J., et al. "Voice controlled home automation system using natural language processing (NLP) and internet of things (IoT)." International Conference on Science Technology Engineering Management, 368-373, 2017.
- [2] Marikyan, D., Papagiannidis, S., & Alamanos, E. "A systematic review of the smart home literature: A user perspective." Technological Forecasting and Social Change, 138, 139-154, 2019.
- [3] López, G., Quesada, L., & Guerrero, L. A. "Alexa vs. Siri vs. Cortana vs. Google Assistant: A comparison of speech-based natural user interfaces." International Conference on Applied Human Factors and Ergonomics, 241-250, 2018.
- [4] Kowalski, J., et al. "Older Adults and Voice Interaction: A Pilot Study with Google Home." Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems, 1-6, 2019.
- [5] Gunge, V. S., & Yalagi, P. S. "Smart home automation: a literature review." International Journal of Computer Applications, 975, 8887, 2016.
- [6] Majumder, S., et al. "Smart Homes for Elderly Healthcare—Recent Advances and Research Challenges." Sensors, 17(11), 2496, 2017.
- [7] <https://artificialanalysis.ai/> (30.12.2024)
- [8] <https://openai.com/api/pricing/> (30.12.2024)
- [9] <https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/> (30.12.2024)

EKLER

EK A: Proje GitHub adresi

<https://github.com/FurkanBaran/BSM497>

EK B: Test sırasında kaydedilmiş log dosyası

```

2025-01-06 06:15:25 - INFO - Starting application
2025-01-06 06:15:25 - INFO - MainClass initialized
2025-01-06 06:15:25 - INFO - Starting database setup...
2025-01-06 06:15:25 - INFO - Database setup completed
2025-01-06 06:15:32 - INFO - Processing user message: 'Merhaba, ben Ali. 35
yaşımdayım ve bu evin babasıyım. Evde neler oluyor?' at 2025-01-06 06:15 Monday
2025-01-06 06:15:32 - INFO - Attempt 1 to process message
2025-01-06 06:15:38 - INFO - Successfully parsed response: {'message': 'Merhaba
Ali! Evdeki sıcaklık oturma odasında yirmi iki derece, yatak odasında yirmi derece.
Işıklar açık ve perdeler açık. Şimdi bugünün olaylarını kontrol ediyorum...',
'db_calls': [{'function': 'add_user', 'parameters': {'name': 'Ali', 'role':
'father', 'age': 35}}, {'function': 'get_today_logs', 'parameters': {}}],
'need_response': True}
2025-01-06 06:15:38 - INFO - Processing assistant message: Merhaba Ali! Evdeki
sıcaklık oturma odasında yirmi iki derece, yatak odasında yirmi derece. Işıklar
açık ve perdeler açık. Şimdi bugünün olaylarını kontrol ediyorum...
2025-01-06 06:15:38 - INFO - Processing database calls: [{'function': 'add_user',
'parameters': {'name': 'Ali', 'role': 'father', 'age': 35}}, {'function':
'get_today_logs', 'parameters': {}}]
2025-01-06 06:15:38 - INFO - Successfully executed DB call: add_user
2025-01-06 06:15:38 - INFO - Successfully executed DB call: get_today_logs
2025-01-06 06:15:38 - INFO - Database operation results: [{'function': 'add_user',
'result': '677b4ada4295cab83a70d886'}, {'function': 'get_today_logs', 'result':
[]}]
2025-01-06 06:15:38 - INFO - Continuing with new message: Database operation
results: [{'function': 'add_user', 'result': '677b4ada4295cab83a70d886'},
{'function': 'get_today_logs', 'result': []}]
2025-01-06 06:15:38 - INFO - Attempt 1 to process message
2025-01-06 06:15:39 - INFO - Successfully parsed response: {'message': 'Ali olarak
evde ilk kez kaydınız yapıldı. Bugün için herhangi bir günlük olay kaydı yok. Başka
bir şey yapmamı ister misiniz?', 'api_calls': None, 'db_calls': None,
'need_response': False}
2025-01-06 06:15:39 - INFO - Processing assistant message: Ali olarak evde ilk kez
kaydınız yapıldı. Bugün için herhangi bir günlük olay kaydı yok. Başka bir şey
yapmamı ister misiniz?
2025-01-06 06:15:39 - INFO - Processing completed
2025-01-06 06:15:39 - INFO - Conversation history saved
2025-01-06 06:18:45 - INFO - Processing user message: 'Buzdolabına 2 litre süt, 1
paket ekmek ve 3 adet elma ekledim. Süt bir hafta içinde bitmeli.' at 2025-01-06
06:18 Monday
2025-01-06 06:18:45 - INFO - Attempt 1 to process message

```



```

2025-01-06 06:18:50 - INFO - Successfully parsed response: {'message': 'Tamam Ali,
buzdolabına eklediğiniz öğeleri kaydediyorum.', 'db_calls': [{'function':
'add_inventory_item', 'parameters': {'name': 'süt', 'category': 'food', 'quantity':
2, 'info': {'expiration': '2025-01-13'}}}], {'function': 'add_inventory_item',
'parameters': {'name': 'ekmek', 'category': 'food', 'quantity': 1}}, {'function':
'add_inventory_item', 'parameters': {'name': 'elma', 'category': 'food',
'quantity': 3}}], 'need_response': True}
2025-01-06 06:18:50 - INFO - Processing assistant message: Tamam Ali, buzdolabına
eklediğiniz öğeleri kaydediyorum.
2025-01-06 06:18:50 - INFO - Processing database calls: [{'function':
'add_inventory_item', 'parameters': {'name': 'süt', 'category': 'food', 'quantity':
2, 'info': {'expiration': '2025-01-13'}}}], {'function': 'add_inventory_item',
'parameters': {'name': 'ekmek', 'category': 'food', 'quantity': 1}}, {'function':
'add_inventory_item', 'parameters': {'name': 'elma', 'category': 'food',
'quantity': 3}}]
2025-01-06 06:18:50 - INFO - Successfully executed DB call: add_inventory_item
2025-01-06 06:18:50 - INFO - Successfully executed DB call: add_inventory_item
2025-01-06 06:18:50 - INFO - Successfully executed DB call: add_inventory_item
2025-01-06 06:18:50 - INFO - Database operation results: [{'function':
'add_inventory_item', 'result': True}, {'function': 'add_inventory_item', 'result':
True}, {'function': 'add_inventory_item', 'result': True}]
2025-01-06 06:18:50 - INFO - Continuing with new message: Database operation
results: [{'function': 'add_inventory_item', 'result': True}, {'function':
'add_inventory_item', 'result': True}, {'function': 'add_inventory_item', 'result':
True}]
2025-01-06 06:18:50 - INFO - Attempt 1 to process message
2025-01-06 06:18:51 - INFO - Successfully parsed response: {'message': 'Süt, ekmek
ve elmalar buzdolabına başarıyla eklendi. Başka bir isteğiniz var mı?',
'api_calls': None, 'db_calls': None, 'need_response': False}
2025-01-06 06:18:51 - INFO - Processing assistant message: Süt, ekmek ve elmalar
buzdolabına başarıyla eklendi. Başka bir isteğiniz var mı?
2025-01-06 06:18:51 - INFO - Processing completed
2025-01-06 06:18:51 - INFO - Conversation history saved
2025-01-06 06:21:07 - INFO - Processing user message: 'Yarın saat 14:00'te oturma
odasının temizlenmesi gerekiyor. Bu görevi not eder misin?' at 2025-01-06 06:21
Monday
2025-01-06 06:21:07 - INFO - Attempt 1 to process message
2025-01-06 06:21:10 - INFO - Successfully parsed response: {'message': "Oturma
odasının temizlenmesi için yarın saat 14:00'te bir görev ekliyorum.", 'db_calls':
[{'function': 'add_task', 'parameters': {'name': 'Oturma odasını temizle',
'due_date': '2025-01-07 14:00', 'info': {'priority': 'normal'}}}], 'need_response':
True}
2025-01-06 06:21:10 - INFO - Processing assistant message: Oturma odasının
temizlenmesi için yarın saat 14:00'te bir görev ekliyorum.
2025-01-06 06:21:10 - INFO - Processing database calls: [{'function': 'add_task',
'parameters': {'name': 'Oturma odasını temizle', 'due_date': '2025-01-07 14:00',
'info': {'priority': 'normal'}}}]
2025-01-06 06:21:10 - INFO - Successfully executed DB call: add_task
2025-01-06 06:21:10 - INFO - Database operation results: [{'function': 'add_task',
'result': True}]
2025-01-06 06:21:10 - INFO - Continuing with new message: Database operation
results: [{'function': 'add_task', 'result': True}]
2025-01-06 06:21:10 - INFO - Attempt 1 to process message
2025-01-06 06:21:11 - INFO - Successfully parsed response: {'message': 'Oturma
odasının temizlenmesi görevi başarıyla eklendi. Başka bir şey yapmamı ister
misiniz?', 'api_calls': None, 'db_calls': None, 'need_response': False}

```

```

2025-01-06 06:21:11 - INFO - Processing assistant message: Oturma odasının
temizlenmesi görevi başarıyla eklendi. Başka bir şey yapmamı ister misiniz?
2025-01-06 06:21:11 - INFO - Processing completed
2025-01-06 06:21:11 - INFO - Conversation history saved
2025-01-06 06:22:02 - INFO - Processing user message: 'Oturma odasının ışıklarını
aç ve sıcaklığı 23 dereceye ayarla.' at 2025-01-06 06:22 Monday
2025-01-06 06:22:02 - INFO - Attempt 1 to process message
2025-01-06 06:22:05 - INFO - Successfully parsed response: {'message': 'Oturma
odasının ışıklarını açıp sıcaklığı yirmi üç dereceye ayarlıyorum.', 'api_calls':
[{'action': 'light.turn_on', 'entity_id': 'light.living_room_light', 'parameters':
{}}, {'action': 'climate.set_temperature', 'entity_id':
'climate.living_room_thermostat', 'parameters': {'temperature': 23}}], 'db_calls':
None, 'need_response': False}
2025-01-06 06:22:05 - INFO - Processing assistant message: Oturma odasının
ışıklarını açıp sıcaklığı yirmi üç dereceye ayarlıyorum.
2025-01-06 06:22:05 - INFO - Processing API calls: [{'action': 'light.turn_on',
'entity_id': 'light.living_room_light', 'parameters': {}}, {'action':
'climate.set_temperature', 'entity_id': 'climate.living_room_thermostat',
'parameters': {'temperature': 23}}]
2025-01-06 06:22:05 - INFO - Successfully executed API call: {'action':
'light.turn_on', 'entity_id': 'light.living_room_light', 'parameters': {}}
2025-01-06 06:22:05 - INFO - Successfully executed API call: {'action':
'climate.set_temperature', 'entity_id': 'climate.living_room_thermostat',
'parameters': {'temperature': 23}}
2025-01-06 06:22:05 - INFO - Processing completed
2025-01-06 06:22:05 - INFO - Conversation history saved
2025-01-06 06:28:14 - INFO - Processing user message: 'Akşam yemeğinde balık
pişireceğim. Salata için domates, salatalık ve limon lazım. Saat 15:30'da çocukları
okuldan alırken markete de uğrayıp bunları almalıyım.' at 2025-01-06 06:28 Monday
2025-01-06 06:28:14 - INFO - Attempt 1 to process message
2025-01-06 06:28:24 - INFO - Successfully parsed response: {'message': 'Balık akşam
yemeği için harika bir seçim. Salata için gerekli olan domates, salatalık ve limonu
alışveriş listesine ekliyorum.', 'db_calls': [{'function': 'add_to_shopping_list',
'parameters': {'name': 'domates', 'status': 'pending', 'info': {}}}, {'function':
'add_to_shopping_list', 'parameters': {'name': 'salatalık', 'status': 'pending',
'info': {}}}, {'function': 'add_to_shopping_list', 'parameters': {'name': 'limon',
'status': 'pending', 'info': {}}}, {'function': 'add_task', 'parameters': {'name':
'Market alışverişi', 'due_date': '2025-01-06 15:30', 'info': {'priority':
'normal'}}}], 'need_response': True}
2025-01-06 06:28:24 - INFO - Processing assistant message: Balık akşam yemeği için
harika bir seçim. Salata için gerekli olan domates, salatalık ve limonu alışveriş
listesine ekliyorum.
2025-01-06 06:28:24 - INFO - Processing database calls: [{'function':
'add_to_shopping_list', 'parameters': {'name': 'domates', 'status': 'pending',
'info': {}}}, {'function': 'add_to_shopping_list', 'parameters': {'name':
'salatalık', 'status': 'pending', 'info': {}}}, {'function':
'add_to_shopping_list', 'parameters': {'name': 'limon', 'status': 'pending',
'info': {}}}, {'function': 'add_task', 'parameters': {'name': 'Market alışverişi',
'due_date': '2025-01-06 15:30', 'info': {'priority': 'normal'}}}]
2025-01-06 06:28:24 - ERROR - Error in add_to_shopping_list:
DatabaseManager.add_to_shopping_list() got an unexpected keyword argument 'name'
2025-01-06 06:28:24 - ERROR - Error in add_to_shopping_list:
DatabaseManager.add_to_shopping_list() got an unexpected keyword argument 'name'
2025-01-06 06:28:24 - ERROR - Error in add_to_shopping_list:
DatabaseManager.add_to_shopping_list() got an unexpected keyword argument 'name'
2025-01-06 06:28:24 - INFO - Successfully executed DB call: add_task

```

```

2025-01-06 06:28:24 - INFO - Database operation results: [{'function':
'add_to_shopping_list', 'error': "Error in add_to_shopping_list:
DatabaseManager.add_to_shopping_list() got an unexpected keyword argument 'name'"},
{'function': 'add_to_shopping_list', 'error': "Error in add_to_shopping_list:
DatabaseManager.add_to_shopping_list() got an unexpected keyword argument 'name'"},
{'function': 'add_to_shopping_list', 'error': "Error in add_to_shopping_list:
DatabaseManager.add_to_shopping_list() got an unexpected keyword argument 'name'"},
{'function': 'add_task', 'result': True}]
2025-01-06 06:28:24 - INFO - Continuing with new message: Database operation
results: [{'function': 'add_to_shopping_list', 'error': "Error in
add_to_shopping_list: DatabaseManager.add_to_shopping_list() got an unexpected
keyword argument 'name'"}, {'function': 'add_to_shopping_list', 'error': "Error in
add_to_shopping_list: DatabaseManager.add_to_shopping_list() got an unexpected
keyword argument 'name'"}, {'function': 'add_to_shopping_list', 'error': "Error in
add_to_shopping_list: DatabaseManager.add_to_shopping_list() got an unexpected
keyword argument 'name'"}, {'function': 'add_task', 'result': True}]
2025-01-06 06:28:24 - INFO - Attempt 1 to process message
2025-01-06 06:28:30 - INFO - Successfully parsed response: {'message': 'Alışveriş
listesine domates, salatalık ve limonu eklemekte bir hata oluştu. Ancak market
alışverişi görevi başarıyla eklendi. Hataları düzeltiyorum ve tekrar deniyorum.',
'db_calls': [{'function': 'add_to_shopping_list', 'parameters': {'item_data':
{'name': 'domates', 'status': 'pending'}}}, {'function': 'add_to_shopping_list',
'parameters': {'item_data': {'name': 'salatalık', 'status': 'pending'}}},
{'function': 'add_to_shopping_list', 'parameters': {'item_data': {'name': 'limon',
'status': 'pending'}}}], 'need_response': True}
2025-01-06 06:28:30 - INFO - Processing assistant message: Alışveriş listesine
domates, salatalık ve limonu eklemekte bir hata oluştu. Ancak market alışverişi
görevi başarıyla eklendi. Hataları düzeltiyorum ve tekrar deniyorum.
2025-01-06 06:28:30 - INFO - Processing database calls: [{'function':
'add_to_shopping_list', 'parameters': {'item_data': {'name': 'domates', 'status':
'pending'}}}, {'function': 'add_to_shopping_list', 'parameters': {'item_data':
{'name': 'salatalık', 'status': 'pending'}}}, {'function': 'add_to_shopping_list',
'parameters': {'item_data': {'name': 'limon', 'status': 'pending'}}}]
2025-01-06 06:28:30 - INFO - Successfully executed DB call: add_to_shopping_list
2025-01-06 06:28:30 - INFO - Successfully executed DB call: add_to_shopping_list
2025-01-06 06:28:30 - INFO - Successfully executed DB call: add_to_shopping_list
2025-01-06 06:28:30 - INFO - Database operation results: [{'function':
'add_to_shopping_list', 'result': True}, {'function': 'add_to_shopping_list',
'result': True}, {'function': 'add_to_shopping_list', 'result': True}]
2025-01-06 06:28:30 - INFO - Continuing with new message: Database operation
results: [{'function': 'add_to_shopping_list', 'result': True}, {'function':
'add_to_shopping_list', 'result': True}, {'function': 'add_to_shopping_list',
'result': True}]
2025-01-06 06:28:30 - INFO - Attempt 1 to process message
2025-01-06 06:28:33 - INFO - Successfully parsed response: {'message': 'Domates,
salatalık ve limon alışveriş listesine başarıyla eklendi. Başka bir isteğiniz var
mı?', 'api_calls': None, 'db_calls': None, 'need_response': False}
2025-01-06 06:28:33 - INFO - Processing assistant message: Domates, salatalık ve
limon alışveriş listesine başarıyla eklendi. Başka bir isteğiniz var mı?
2025-01-06 06:28:33 - INFO - Processing completed
2025-01-06 06:28:33 - INFO - Trimming conversation history
2025-01-06 06:28:33 - INFO - Conversation history saved
2025-01-06 06:33:13 - INFO - Processing user message: 'Akşam yemeğinin balık
olacağını günlük aktivite kayıtlarına ekler misin? Böylece aile üyeleri sorduğunda
onlara söyleyebilirsin' at 2025-01-06 06:33 Monday
2025-01-06 06:33:13 - INFO - Attempt 1 to process message

```

```

2025-01-06 06:33:23 - INFO - Successfully parsed response: {'message': 'Akşam
yemeğinin balık olduğunu günlük aktivite kayıtlarına ekliyorum.', 'db_calls':
[{'function': 'add_daily_log', 'parameters': {'title': 'Akşam Yemeği', 'details':
'Akşam yemeği balık olacak.'}], 'need_response': True}
2025-01-06 06:33:23 - INFO - Processing assistant message: Akşam yemeğinin balık
olduğunu günlük aktivite kayıtlarına ekliyorum.
2025-01-06 06:33:23 - INFO - Processing database calls: [{'function':
'add_daily_log', 'parameters': {'title': 'Akşam Yemeği', 'details': 'Akşam yemeği
balık olacak.'}}]
2025-01-06 06:33:23 - INFO - Successfully executed DB call: add_daily_log
2025-01-06 06:33:23 - INFO - Database operation results: [{'function':
'add_daily_log', 'result': True}]
2025-01-06 06:33:23 - INFO - Continuing with new message: Database operation
results: [{'function': 'add_daily_log', 'result': True}]
2025-01-06 06:33:23 - INFO - Attempt 1 to process message
2025-01-06 06:33:26 - INFO - Successfully parsed response: {'message': 'Akşam
yemeğinin balık olduğu günlük kayıtlarına başarıyla eklendi. Başka bir şeyle
yardımcı olabilir miyim?', 'api_calls': None, 'db_calls': None, 'need_response':
False}
2025-01-06 06:33:26 - INFO - Processing assistant message: Akşam yemeğinin balık
olduğu günlük kayıtlarına başarıyla eklendi. Başka bir şeyle yardımcı olabilir
miyim?
2025-01-06 06:33:26 - INFO - Processing completed
2025-01-06 06:33:26 - INFO - Trimming conversation history
2025-01-06 06:33:26 - INFO - Conversation history saved
2025-01-06 06:35:12 - INFO - Processing user message: 'tamam, ben okula gidiyorum
görüşürüz.' at 2025-01-06 06:35 Monday
2025-01-06 06:35:12 - INFO - Attempt 1 to process message
2025-01-06 06:35:18 - INFO - Successfully parsed response: {'message': 'Okula
gitmek üzere ayrılıyorsunuz, görüşürüz. Dikkatli olun.', 'db_calls': [{'function':
'add_daily_log', 'parameters': {'title': 'Okula Gidiş', 'details': 'Ali okula
gitmek üzere evden ayrıldı.'}], 'need_response': False}
2025-01-06 06:35:18 - INFO - Processing assistant message: Okula gitmek üzere
ayrılıyorsunuz, görüşürüz. Dikkatli olun.
2025-01-06 06:35:18 - INFO - Processing database calls: [{'function':
'add_daily_log', 'parameters': {'title': 'Okula Gidiş', 'details': 'Ali okula
gitmek üzere evden ayrıldı.'}}]
2025-01-06 06:35:18 - INFO - Successfully executed DB call: add_daily_log
2025-01-06 06:35:18 - INFO - Processing completed
2025-01-06 06:35:18 - INFO - Trimming conversation history
2025-01-06 06:35:18 - INFO - Conversation history saved
2025-01-06 06:45:44 - INFO - Processing user message: 'Annemi sisteme eklemek
istiyorum. Adı Ayşe, 55 yaşında' at 2025-01-06 06:45 Monday
2025-01-06 06:45:44 - INFO - Attempt 1 to process message
2025-01-06 06:45:47 - INFO - Successfully parsed response: {'message': 'Annenizi
sisteme ekliyorum. Adı Ayşe, yaşı elli beş.', 'db_calls': [{'function': 'add_user',
'parameters': {'name': 'Ayşe', 'role': 'anne', 'age': 55}], 'need_response': True}
2025-01-06 06:45:47 - INFO - Processing assistant message: Annenizi sisteme
ekliyorum. Adı Ayşe, yaşı elli beş.
2025-01-06 06:45:47 - INFO - Processing database calls: [{'function': 'add_user',
'parameters': {'name': 'Ayşe', 'role': 'anne', 'age': 55}}]
2025-01-06 06:45:47 - INFO - Successfully executed DB call: add_user
2025-01-06 06:45:47 - INFO - Database operation results: [{'function': 'add_user',
'result': '677b51eb4295cab83a70d887'}]
2025-01-06 06:45:47 - INFO - Continuing with new message: Database operation
results: [{'function': 'add_user', 'result': '677b51eb4295cab83a70d887'}]
2025-01-06 06:45:47 - INFO - Attempt 1 to process message

```

```

2025-01-06 06:45:49 - INFO - Successfully parsed response: {'message': 'Anneniz
Ayşe başarıyla sisteme eklendi. Başka bir isteğiniz var mı?', 'api_calls': None,
'db_calls': None, 'need_response': False}
2025-01-06 06:45:49 - INFO - Processing assistant message: Anneniz Ayşe başarıyla
sisteme eklendi. Başka bir isteğiniz var mı?
2025-01-06 06:45:49 - INFO - Processing completed
2025-01-06 06:45:49 - INFO - Trimming conversation history
2025-01-06 06:45:49 - INFO - Conversation history saved
2025-01-06 06:45:57 - INFO - Processing user message: 'Annem Ayşe bugün biraz hasta
hissediyor, ateşi 38.2 derece. Bu durumu kaydet ve ilaç saatlerini takip etmemiz
gerekıyor. 4 saatte bir parasetamol alması lazım, ilk dozu saat 14:00'te aldı.' at
2025-01-06 06:45 Monday
2025-01-06 06:45:57 - INFO - Attempt 1 to process message
2025-01-06 06:46:01 - INFO - Successfully parsed response: {'message': 'Annenizin
sağlık durumunu kaydediyorum. Ateşi otuz sekiz nokta iki derece ve dört saatte bir
parasetamol alması gerekiyor. İlk dozu saat on dörtte almış. Bu bilgileri sisteme
ekliyorum.', 'db_calls': [{'function': 'update_user_health', 'parameters': {'name':
'Ayşe', 'status': 'hasta', 'medical_record': {'temperature': 38.2, 'medication':
{'name': 'parasetamol', 'dosage': '4 saatte bir', 'first_dose_time': '14:00'}}}],
'need_response': True}
2025-01-06 06:46:01 - INFO - Processing assistant message: Annenizin sağlık
durumunu kaydediyorum. Ateşi otuz sekiz nokta iki derece ve dört saatte bir
parasetamol alması gerekiyor. İlk dozu saat on dörtte almış. Bu bilgileri sisteme
ekliyorum.
2025-01-06 06:46:01 - INFO - Processing database calls: [{'function':
'update_user_health', 'parameters': {'name': 'Ayşe', 'status': 'hasta',
'medical_record': {'temperature': 38.2, 'medication': {'name': 'parasetamol',
'dosage': '4 saatte bir', 'first_dose_time': '14:00'}}}]]
2025-01-06 06:46:01 - INFO - Successfully executed DB call: update_user_health
2025-01-06 06:46:01 - INFO - Database operation results: [{'function':
'update_user_health', 'result': True}]
2025-01-06 06:46:01 - INFO - Continuing with new message: Database operation
results: [{'function': 'update_user_health', 'result': True}]
2025-01-06 06:46:01 - INFO - Attempt 1 to process message
2025-01-06 06:46:03 - INFO - Successfully parsed response: {'message': 'Annenizin
sağlık durumu başarıyla kaydedildi. Ateşi ve ilaç saatlerini takip edeceğiz. Başka
bir şeyle yardımcı olabilir miyim?', 'api_calls': None, 'db_calls': None,
'need_response': False}
2025-01-06 06:46:03 - INFO - Processing assistant message: Annenizin sağlık durumu
başarıyla kaydedildi. Ateşi ve ilaç saatlerini takip edeceğiz. Başka bir şeyle
yardımcı olabilir miyim?
2025-01-06 06:46:03 - INFO - Processing completed
2025-01-06 06:46:03 - INFO - Trimming conversation history
2025-01-06 06:46:03 - INFO - Conversation history saved

```

ÖZGEÇMİŞ

Furkan BARAN, 03.11.2000 de Diyarbakır'da doğdu. İlk, orta ve lise eğitimini Diyarbakır'da tamamladı. 2019 yılında Gebze Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü'nü kazandı. 2023 yılında Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü'ne geçiş yaptı. 2023 yılında Otomatica Kontrol Otomasyon Yazılım Tic. Ltd. Şirketinde donanım stajını ve 2024 yılında da Medyasoft IT Group Şirketinde donanım stajını yapmıştır.

BSM 498 BİTİRME ÇALIŞMASI DEĞERLENDİRME VE SÖZLÜ SINAV TUTANAĞI

KONU : Yapay Zeka Tabanlı Akıllı Ev Asistanı Tasarımı ve Gerçekleştirimi
ÖĞRENCİLER (B231210371/FURKAN/BARAN):

Değerlendirme Konusu	İstenenler	Not Aralığı	Not
Yazılı Çalışma			
Çalışma klavuza uygun olarak hazırlanmış mı?	x	0-5	
Teknik Yönden			
Problemin tanımı yapılmış mı?	x	0-5	
Geliştirilecek yazılımın/donanımın mimarisini içeren blok şeması (yazılımlar için veri akış şeması (dfd) da olabilir) çizilerek açıklanmış mı?			
Blok şemadaki birimler arasındaki bilgi akışına ait model/gösterim var mı?			
Yazılımın gereksinim listesi oluşturulmuş mu?			
Kullanılan/kullanılması düşünülen araçlar/teknolojiler anlatılmış mı?			
Donanımların programlanması/konfigürasyonu için yazılım gereksinimleri belirtilmiş mi?			
UML ile modelleme yapılmış mı?			
Veritabanları kullanılmış ise kavramsal model çıkarılmış mı? (Varlık ilişki modeli, noSQL kavramsal modelleri v.b.)			
Projeye yönelik iş-zaman çizelgesi çıkarılarak maliyet analizi yapılmış mı?			
Donanım bileşenlerinin maliyet analizi (prototip-adetli seri üretim vb.) çıkarılmış mı?			
Donanım için gerekli enerji analizi (minimum-uyku-aktif-maksimum) yapılmış mı?			
Grup çalışmalarında grup üyelerinin görev tanımları verilmiş mi (iş-zaman çizelgesinde belirtilebilir)?			
Sürüm denetim sistemi (Version Control System; Git, Subversion v.s.) kullanılmış mı?			
Sistemin genel testi için uygulanan metotlar ve iyileştirme süreçlerinin dökümü verilmiş mi?			
Yazılımın sızma testi yapılmış mı?			
Performans testi yapılmış mı?			
Tasarımın uygulamasında ortaya çıkan uyumsuzluklar ve aksaklıklar belirtilerek çözüm yöntemleri tartışılmış mı?			
Yapılan işlerin zorluk derecesi?	x	0-25	
Sözlü Sınav			
Yapılan sunum başarılı mı?	x	0-5	
Soruları yanıtlama yetkinliği?	x	0-20	
Devam Durumu			
Öğrenci dönem içerisindeki raporlarını düzenli olarak hazırladı mı?	x	0-5	
Diğer Maddeler			
Toplam			

DANIŞMAN (JÜRİ ADINA): PROF. DR. CÜNEYT BAYILMIŞ

DANIŞMAN İMZASI: