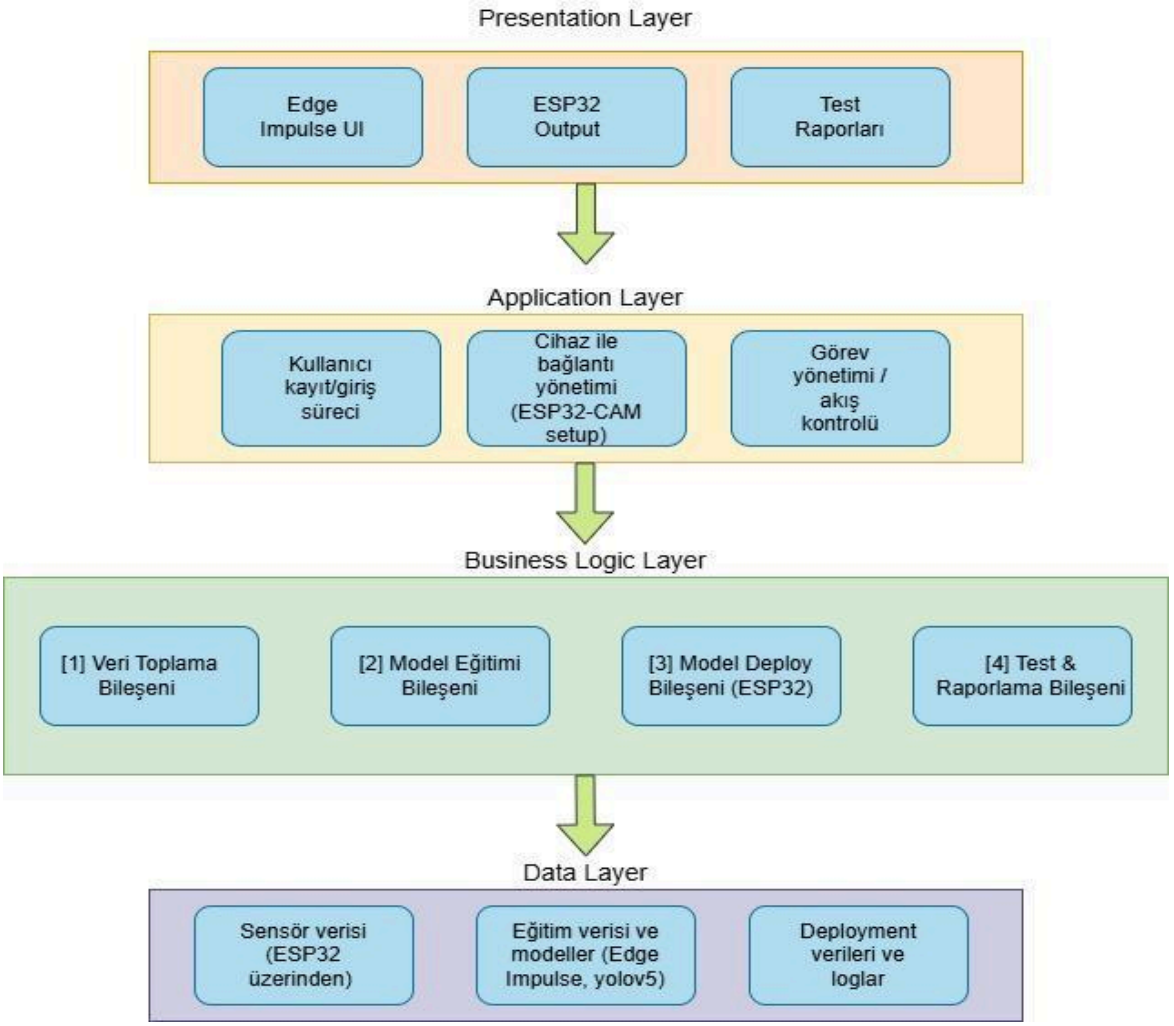
Sistem Mimarisi Tasarımı

Projemiz olan ESP32-CAM tabanlı, Edge Impulse ile geliştirilen nesne tanıma sistemi için Katmanlı Mimari ile Bileşen Tabanlı Mimarinin bir arada kullanıldığı hibrit bir mimari yapı tercih edilmiştir (Şekil 1.). Bu tercih, projenin modüler, sürdürülebilir ve geliştirilebilir bir yapıya sahip olmasını sağlamıştır.

- Katmanlı mimari sayesinde sistem; sunum, uygulama, iş mantığı ve veri katmanları şeklinde mantıksal olarak ayrılmış, her katmanın belirli sorumluluklara sahip olması sağlanmıştır. Bu yapı, görevlerin düzenli ilerlemesine ve katmanlar arası bağımsızlığa imkân tanımıştır.
- Aynı zamanda, her katman içerisinde yer alan işlevler bileşenler şeklinde modellenmiş ve birbirinden bağımsız, açık arayüzlerle etkileşim kuran yapı taşları hâline getirilmiştir. Böylece her görev (örneğin veri toplama, model eğitimi gibi) bağımsız geliştirilip test edilebilir hâle gelmiştir.

Bu mimari yapılar, projenin veri toplama, model eğitimi, dağıtımı ve test gibi ardışık ve tekrarlanabilir adımlara sahip olması nedeniyle tercih edilmiştir. Aynı zamanda yazılımın esnekliğini, yeniden kullanılabilirliğini ve bakım kolaylığını artırarak proje gereksinimlerine doğrudan hizmet etmiştir.



Şekil 1. Sistemin Katmanlı ve Bileşen Tabanlı Hibrit Mimarisi

1. Katmanlı Mimari (Layered Architecture)

Katmanlı mimarilerin yazılım geliştirmede en yaygın ve en çok kullanılan mimari çerçeve olduğu söylenir. Ayrıca n katmanlı mimari olarak da bilinir ve birlikte tek bir yazılım birimi olarak işlev gören birkaç ayrı yatay katmandan oluşan bir mimari deseni tanımlar . Bir katman, bileşenlerin veya kodun mantıksal bir ayrımıdır: Bu çerçevelerde, ilişkili veya benzer bileşenler genellikle aynı katmanlara yerleştirilir . Ancak, her katman farklıdır ve genel sistemin farklı bir bölümüne katkıda bulunur. Bu çerçevenin önemli bir özelliği, katmanların yalnızca doğrudan altlarındaki katmanlara bağlı olmasıdır.

Başka bir özellik ise yalıtım katmanları kavramıdır. Bu, katmanların değiştirilebileceği ve değişikliğin diğer katmanları etkilemeyeceği anlamına gelir. Kısacası, değişiklikler değiştirilen belirli katmana izole edilir.

Katmanlı mimarideki katman sayısı belirli bir sayıya ayarlanmamıştır ve genellikle geliştiriciye veya yazılım mimarına bağlıdır. Biz sistemimizi dört temel katmanda yapılandırdık:

• **Presentation Layer (Sunum Katmanı):** Kullanıcının etkileşim kurduğu arayüzleri içerir:

- Edge Impulse kullanıcı arayüzü
- ESP32'nin gerçek zamanlı çıktı ekranı
- Test ve raporlama ekranları

• **Application Layer (Uygulama Katmanı):** Sistemin genel akışını ve görev kontrolünü sağlar:

- ESP32-CAM bağlantı yönetimi
- Görev yönetimi (test başlatma, akış kontrolü)

• **Business Logic Layer (İş Mantığı Katmanı):** İşin çekirdek işlevselliğini barındırır. Bu katman içinde bileşen tabanlı yapı ile aşağıdaki görevler ayrıştırılmıştır:

- [1] Veri Toplama Bileşeni
- [2] Model Eğitimi Bileşeni
- [3] Model Dağıtımı Bileşeni
- [4] Test & Raporlama Bileşeni

• **Data Layer (Veri Katmanı):** Tüm veriler burada tutulur:

- ESP32 üzerinden gelen sensör verileri
- Eğitim verisi ve modeller (Edge Impulse, YOLOv5)
- Dağıtım sonrası loglar, çıktı verileri

Bu yapı, sistem bileşenlerinin birbirinden ayrılmasını ve katmanlar arası net bir sorumluluk dağılımını sağlamaktadır. Böylece sistem hem bakım kolaylığı, hem de yeniden kullanılabilirlik açısından avantaj kazanmaktadır.

2. Bileşen Tabanlı Mimari (Component-Based Architecture)

Yazılımın önceden tanımlanmış, yeniden kullanılabilir bileşenleri bir araya getirerek oluşturulduğu bir metodolojiyi ifade eder. Her bileşen, bileşenlerin birbirleriyle nasıl etkileşime gireceğini yöneten iyi tanımlanmış arayüzlerle belirli bir işlevsellik veya davranış parçasını kapsüller. Bu yaklaşım, yazılım geliştirmede modülerliği, esnekliği, soyutlamayı, bağımsızlığı ve yeniden kullanılabilirliği teşvik eder.

Örneğin İş mantığı katmanı, her biri belirli bir görevi gerçekleştiren dört bağımsız bileşene ayrılmıştır:

Bileşen No	Açıklama	Görev
[1]	Veri Toplama Bileşeni	Kaggle'dan çiçek sınıflarına göre görsellerin toplanması, Edge Impulse'a aktarılması
[2]	Model Eğitimi Bileşeni	Özellik çıkarımı, model eğitimi, test doğrulama
[3]	Model Deploy Bileşeni	Eğitilen modelin ESP32'ye uygun hale getirilmesi ve yüklenmesi
[4]	Test ve Raporlama Bileşeni	Modelin gerçek zamanlı test edilmesi ve çıktıların değerlendirilmesi

Her bileşen, sistemin tamamından bağımsız geliştirilebilir ve gerektiğinde yeniden kullanılabilir. Örneğin farklı bir TinyML projesi için yalnızca [2] Model Eğitimi Bileşeni yeniden entegre edilebilir.

Sonuç

- Veri Toplama, Model Eğitimi ve Deployment aşamaları, Edge Impulse üzerinde birbirini takip eden adımlar şeklinde yürütüldüğünden, katmanlı mimari bu yapıya doğal olarak uyum sağlar.

- ESP32 donanımı ile etkileşim, cihaz bağlantı yönetimi, test süreci gibi görevler açık bir akış içinde birbirini izler; bu da Application ve Business Logic Layer ayrımını anlamlı kılar.

- Bileşenlerin birbirinden ayrılması, farklı model türleriyle çalışmak ya da test sistemini değiştirmek gibi gereksinimlerde sistemi esnek hale getirir.

- Sunum, kontrol, iş mantığı ve veri yönetimi ayrımı, modülerlik, test edilebilirlik ve geliştirilebilirlik kriterlerine doğrudan katkı sunar.