Döngüsü (SDLC) Planı – Agile Yaklaşım

Giriş: ARColorCathes projesinin yazılım geliştirme süreci, çevik (Agile) yöntemler doğrultusunda planlanmıştır. Agile metodolojisi, gereksinimlerin ve çözümlerin ekipler arası işbirliğiyle ve iteratif yaklaşımla ele alındığı esnek bir geliştirme sürecidir. Bu bölümde, projenin fikir aşamasından bakım aşamasına kadar olan yaşam döngüsü **Planlama, Analiz, Tasarım, Geliştirme, Test, Dağıtım ve Bakım** adımları halinde açıklanacaktır. Her aşamada, Agile yaklaşımının nasıl uygulandığı, takım içi etkileşimler, geri bildirim döngüleri ve yinelemeli geliştirme vurgulanacaktır.

Planlama Aşaması (Proje Planlama ve Kapsam Belirleme)

Planlama, projenin başlangıç noktasıdır ve Agile felsefesinde proje planlaması büyük ölçüde **produkt arka lojik (product backlog)** oluşturma ve önceliklendirme şeklinde yapılır. ARColorCathes için planlama aşamasında yapılanlar:

- **Vizyon ve Amaç:** Öncelikle projenin vizyonu belirlendi. ARColorCathes'in vizyonu, "çocuklar için eğitsel ve eğlenceli bir artırılmış gerçeklik deneyimi sunmak" olarak tanımlandı. Bu vizyon, tüm geliştirme sürecine yön verecek bir kılavuzdur. Projenin kapsamı, artırılmış gerçeklik yoluyla renkler, şekiller, harfler gibi temel kavramların öğretilmesiyle sınırlandırıldı. Bu, proje ekibinin odağını netleştirdi ve "gereksinim kaymasını" önlemeye yardımcı oldu.
- **İlk Gereksinimler ve Backlog:** Takım, beyin fırtınası yaparak kullanıcı hikâyeleri (**user story**) formatında gereksinimleri ortaya koydu. Örneğin: "Bir kullanıcı olarak, kameramı kullanarak gerçek dünyada sanal bloklar yerleştirmek istiyorum ki kendi sanal kulemi inşa edebileyim." benzeri hikâyeler tanımlandı. Bu hikâyeler product backlog'a yazıldı. ARColorCathes özelinde backlog maddeleri, üç oyun modunun her birine ilişkin özellikleri, genel arayüz gereksinimlerini (menü, ayarlar, vs.) ve teknik gereklilikleri (ARCore entegrasyonu, kamera izin sistemi, vb.) içeriyordu.
- **Önceliklendirme:** Backlog oluşturulduktan sonra maddeler önceliklendirildi. Agile yaklaşımda müşteri değerine odaklanıldığı için, ilk olarak en fazla değer katacak ve en riskli konular ele alınır. Bu projede öncelikle **AR temel özelliği** (kamera ile yüzey algılayıp nesne yerleştirme) en kritik özellikti; bu olmadan diğer oyun modları çalışamazdı. Dolayısıyla ilk iterasyonlarda bu temel AR altyapısının sağlıklı çalışması planlandı. Ardından modların her birinin temel işlevleri (blok yerleştirme, renk eşleme, harf bilmece) önceliklendirildi. Daha az öncelikli işler (örneğin ekstradan eklenecek yeni bir oyun modu fikri veya gelişmiş ayarlar menüsü) backlog'un alt sıralarına konuldu.
- **Zaman Çizelgesi ve Sprint Planlaması:** Projenin zaman kısıtları (örneğin akademik takvimde teslim tarihi veya belirli bir demo tarihi) göz önüne alınarak, kaç adet **sprint** yapılacağı planlandı. Diyelim ki projenin toplam süresi 12 hafta, bu durumda 2 haftalık sprintler halinde 6 sprint öngörüldü. Her bir sprint için hedeflenen teslimatlar

belirlendi: 1. Sprint sonunda ARCore ile temel bir nesne yerleştirme demosu, 2. Sprint sonunda “Kendin Yap” modunun prototipi, 3. Sprint sonunda “Kırmızı Yeşil Mavi” modunun eklenmesi, vb. gibi. Bu şekilde, planlama aşaması sonunda yol haritası ve sprint hedefleri ortaya çıkmış oldu.

Analiz Aşaması (Gereksinim Analizi ve Detaylandırma)

Analiz aşaması, backlog’da tanımlanan kullanıcı hikâyelerinin ayrıntılandırılıp netleştirildiği süreçtir. Agile’da analiz, planlama ve geliştirme aşamalarından tamamen ayrı bir faz olmaktan ziyade **sürekli** bir faaliyet olsa da, projenin başında kritik kısımlar için daha yoğun bir analiz yapılmıştır:

- **Paydaş Analizi ve Hedef Kitle:** ARColorCathes için ana paydaşlar çocuk kullanıcılar ve dolaylı olarak ebeveynler/öğretmenlerdir. Analiz aşamasında “Kullanıcıların ihtiyaçları nelerdir?” sorusuna yanıt arandı. Çocukların dikkat süreleri, etkileşim biçimleri ve eğitimsel ihtiyaçları değerlendirildi. Örneğin, **renkleri öğrenme yaşı** ve bu yaştaki çocukların kavrama yöntemleri incelendi. Bu sayede her oyun modunun ne şekilde sunulursa etkili olacağına dair içgörüler elde edildi (örneğin renk eşleme oyununun çok uzun tutulmaması, başarı durumunda çocuğun motive edilmesi gibi).
- **Gereksinim Detaylandırma:** Backlog’daki her kullanıcı hikâyesi için kabul kriterleri (**acceptance criteria**) tanımlandı. Örneğin, “Kendin Yap modunda bir blok eklediğimde, blok zeminde stabil durmalıdır ve bir sonraki bloğu eklerken mevcut bloklar yerinde kalmaya devam etmelidir.” gibi kriterler yazıldı. Bu kriterler, modların tam olarak hangi koşullarda başarılı sayılacağını tanımlar ve testler için de temel oluşturur.
- **Teknik Analiz:** Projenin teknik açıdan analizi de bu aşamada yapıldı. ARCore ve Unity AR Foundation’ın yetenekleri ve sınırlamaları araştırıldı. Örneğin, aynı anda maksimum kaç tane sanal nesne performans sorununa yol açmadan sahnede tutulabilir? Işıklandırma koşulları AR nesnelerinin görünürlüğünü nasıl etkiler? Gibi soruların yanıtları arandı. Elde edilen bilgilere göre teknik riskler belirlendi ve bunlar için çözümler düşünülmeye başlandı. Diyelim ki analiz sırasında “düşük donanımlı cihazlarda çok sayıda nesne performansı düşürebilir” riski bulundu; buna karşı, nesne sayısına bir sınır koyma ya da nesnelerin çok uzakta oluşturulmasını engelleme gibi kararlar alındı.
- **Prototip ve Kavramsal Kanıt (PoC):** Agile yaklaşımında analiz, geliştirmeden tamamen kopuk değildir. Hatta belirsiz veya riskli görünen konularda hızlıca bir **Proof of Concept (Kavramsal Kanıt)** prototipi yapılması çok yararlı olur. ARColorCathes için örneğin, daha proje başında küçük bir Unity sahnesinde AR nesnesi yerleştirme denemesi yapıldı. Bu deneme, kamera izinleri, ARCore kurulumu gibi konularda karşılaşılabilecek sorunları önceden görmeye yardımcı oldu. Elde edilen sonuçlara göre analiz güncellendi (örn: uygulama açılışında ARCore kontrolü yapma gerekliliği belirlendi).

Analiz aşaması, projenin ne yapacağını netleştirmesini sağladı. Bu bilgiler, bir yandan planlamadaki backlog'u güncel tuttu (yeni keşfedilen gereksinimler eklendi, gereksiz olanlar çıkarıldı), diğer yandan gelecek tasarım ve geliştirme adımlarının sağlam temele oturmasına yardımcı oldu. Unutulmamalıdır ki Agile süresince analiz tamamen bitmez; her sprintin başında sprint planlaması yapılırken o sprintte ele alınacak backlog maddeleri tekrar detaylandırılır ve gerekirse müşteri/mentor geri bildirimi ile revize edilir.

Tasarım Aşaması (Sistem ve Mimari Tasarım)

Tasarım aşaması, uygulamanın hem **yazılım mimarisi** hem de **arayüz/deneyim tasarımı** boyutlarını kapsar. Agile projelerde "Big Design Up Front" (en başta kapsamlı tasarım) yerine evrimsel tasarım benimsenir; ancak ARColorCathes için temel mimari kararlar ve genel arayüz konsepti ilk aşamalarda belirlenmiştir:

- **Yazılım Mimarisi:** Projenin mimarisi, Unity oyun motorunun standart yapısı üzerine kuruludur. Bu bağlamda MVC (Model-View-Controller) veya MVVM gibi kalıplardan ziyade Unity'nin bileşen tabanlı mimarisi kullanılmıştır. Tasarım aşamasında, **her oyun modunun ayrı sahneler (scenes) olarak tasarlanması** kararlaştırıldı. Örneğin "Kendin Yap" modu DiyScene.unity, "Kırmızı Yeşil Mavi" modu ColorSortScene.unity, "Alfabe Bilmece" modu AlphabetRiddleScene.unity gibi ayrı sahnelerde geliştirildi. Ortak kullanılan bileşenler (kamera ayarları, AR yüzey algılama, temel UI çerçevesi) için sahneler arası taşınabilen prefab ve script'ler oluşturuldu. Bu modüler yaklaşım, bir mod üzerinde çalışırken diğerlerinin etkilenmesini en aza indirir.

Yazılım tasarımında ayrıca **AR yönetimi** için bir GameObject (örneğin ARSession) ve **AR içerik yerleştirme** için bir başka obje (ARPlacementManager gibi) planlandı. Bu objeler AR Foundation'ın sağladığı ARSession ve ARSessionOrigin bileşenlerini içerir. Oyun modlarının spesifik mantığı (blok dizme, renk eşleştirme, bilmece sorma) kendi controller script'lerinde yazıldı. Bu sayede, örneğin Alfabe Bilmece modunun nasıl çalıştığını değiştirmek gerektiğinde sadece ilgili script üzerinde düzenleme yapmak yeterli olacak, diğer modların koduna dokunulmayacak.

- **Kullanıcı Arayüzü Tasarımı:** ARColorCathes'in hedef kitlesi çocuklar olduğundan, arayüz tasarımı basitlik ve görsellik odaklı yapıldı. Tasarım aşamasında bir **wireframe** (kabataslak arayüz taslağı) çizildi. Ana menüde büyük ikonlarla temsil edilen üç mod butonu, üst tarafta uygulama logosu ve belki alt kısımda hakkında/ayarlar erişimi düşünüldü. Her bir oyun modunun içinde, minimal bir arayüz yaklaşımı benimsendi: Ekranın köşesinde ana menüye dön düğmesi, belki gerektiğinde ipucu veya tekrar başlatma butonları. Renk paleti olarak canlı ve çocuklara hitap eden renkler seçildi (arka plan yerine kamera görüntüsü olduğu için UI öğeleri kontrast renklerde olmalıydı). Örneğin butonlar sarı veya turuncu, metinler beyaz kenarlıklı siyah gibi belirgin tasarlandı.

Tasarım aşamasında çizimler ve gerekirse basit prototipler (örneğin Adobe XD veya kağıt üzerinde) hazırlandı ve ekip içi paydaşlarla (örneğin danışman öğretmen, hedef kullanıcı kitlesini temsil eden biri) paylaşıldı. Alınan geri bildirimlerle arayüz tasarımında düzeltmeler yapıldı. Örneğin başta tüm modlar için tek bir sahne düşünülmüşken, arayüz sadeliği adına modları tamamen ayırmanın daha iyi olacağına karar verildi (her mod kendi deneyimine odaklansın, mod geçişlerinde bellek temizlensin vs.). Bu tür tasarım kararları bu aşamada netleştirildi.

- **Deneyim Tasarımı (UX):** AR uygulamalarında kullanıcı deneyimi, sıradan 2D uygulamalardan farklıdır. Bu nedenle tasarım aşamasında **UX özelinde** bazı konular ele alındı: Kullanıcıya AR ortamını nasıl öğreteceğiz? Örneğin ilk kullanımda ekranda beliren yönlendirme okları veya kısa bir öğretici (tutorial) olacak mı? Takım, uygulamanın mümkün olduğunca sezgisel olmasını hedefledi. Bu yüzden, **eğitici ipuçları görsel öğelerle** verilecek şekilde tasarlandı. İlk mod açıldığında "telefonu hareket ettir" ikonları veya bir sanal karakterin rehberlik etmesi gibi fikirler tartışıldı. Zaman kısıtı nedeniyle tam bir tutorial uygulanamasa da, arayüzdeki simgelerin kendilerini açıklayıcı olmasına özen gösterildi (örn: çöp kutusu simgesi = sıfırla, ev simgesi = menüye dön).
- **Teknik Tasarım Detayları:** Özellikle AR tarafında, tasarım aşamasında bazı teknik yapılandırmalar belirlendi. Örneğin, **ölçek** konusu: Sanal nesnelerin gerçek dünyadaki boyutları ne olacak? Blokların bir kenarı 10 cm mi temsil edecek yoksa 1 metre mi? Bunlar belirlendi ki modlar arasında tutarlılık olsun. "Kendin Yap" modundaki blok boyutları ile "Alfabe Bilmece" modundaki nesnelerin boyutları orantılı ayarlandı. Yine, sanal nesnelerin görselleri/tasarımları (modellemeler) belirlendi: Bloklar için basit geometrik cisimler mi kullanılacak, yoksa harf blokları mı? Renk eşleme için hangi nesneler kullanılacak (belki renkli küpler veya meyve resimleri)? Bu içerik tasarımları, tasarım aşamasının parçası olarak konsept sanatçısı veya geliştiriciler tarafından kararlaştırıldı.

Tasarım aşamasının Agile karakteristiği olarak, unutmamak gerekir ki tasarım "donmuş" bir çıktı değildir. Geliştirme ilerledikçe ortaya çıkan yeni ihtiyaçlara göre tasarımın bazı yönleri güncellendi. Ancak bu aşamada yapılan iyi planlama, projenin daha sonraki kaotik değişikliklerden korunmasına yardımcı oldu. Özellikle mimari modülerlik kararı, Agile süreçte gelen değişikliklerin sadece lokal etkilerle uygulanabilmesini sağladı.

Geliştirme Aşaması (Kodlama ve İteratif Geliştirme)

Geliştirme aşaması, planlanan özelliklerin kodlanarak hayata geçirildiği süreçtir. Agile yaklaşım gereği ARColorCathes, **iteratif ve artımlı** bir şekilde geliştirilmiştir. Bu, projenin belirli aralıklarla çalışan sürümlerinin ortaya çıktığı ve her yinelemede (iterasyon) ürüne yeni özellikler eklendiği anlamına gelir. Geliştirme sürecinin özellikleri:

- **Sprintler Halinde Geliştirme:** Planlama aşamasında öngörülen sprint takvimine uygun olarak, ekip her sprintin başında bir **Sprint Planlama** toplantısı yaptı. Bu toplantılarda o sprintte tamamlanacak kullanıcı hikâyeleri seçildi. Örneğin 1. Sprint'te "ARCore ile düzlem algılama ve temel nesne yerleştirme", 2. Sprint'te "Kendin Yap modunda blok ekleme/taşıma", 3. Sprint'te "Renk eşleme modunun temel işlevleri" vb. Sprint planlamasında her hikâye için görevler belirlendi ve ekip üyelerine dağıtıldı. Görev dağılımı yapılırken ekip üyelerinin uzmanlıkları ve ilgi alanları dikkate alındı; ancak Agile ekiplerde herkesin gerektiğinde farklı işlere de destek olabileceği anlayışı benimsendi.
- **Günlük Toplantılar ve İşbirliği:** Sprint boyunca her gün kısa **Daily Scrum (Günlük)** toplantıları yapıldı. Bu toplantılarda ekip üyeleri bir önceki gün ne yaptıklarını, o gün ne yapacaklarını ve engelleri varsa neler olduğunu paylaştılar. Bu sayede takım içi iletişim güçlü tutuldu ve olası engeller hızlıca görünüp çözüldü. Örneğin, bir geliştirici "Renk eşleme modunda nesne sürükleme işini yaparken bir çökme sorunu alıyorum" dediğinde, belki diğer ekip üyesi benzer bir sorunla karşılaşıp çözdüğü için hemen yardımcı olabildi. Takım içi işbirliği en üst düzeyde tutuldu; kod paylaşımı, ortak kod oturumları (pair programming) gibi pratikler de kullanıldı.
- **Kodlama ve Versiyon Kontrol:** Geliştirme sırasında tüm kod değişiklikleri Git üzerinden yönetildi. Her özellik için gerekirse ayrı bir dal (branch) açıldı, üzerinde çalışmalar yapıldı ve tamamlandığında **Pull Request** ile ana koda entegre edildi. Kod gözden geçirme süreçleri uygulandı; en az iki kişi önemli değişiklikleri inceleyip onayladı. Bu sadece hataları yakalamak için değil, takımın herkesinin kod tabanına hakim olması için de yapıldı. Kodlama yapılırken Agile'ın **basitlik** ilkesi akılda tutuldu: "Gereğinden fazla karmaşık çözüm yapma, işini görecek en basit tasarımı uygula." Bu sayede hem hızlı ilerleme sağlandı hem de gelecekteki bakım yükü azaltıldı.
- **İteratif Teslimatlar:** Her sprint sonunda çalışan bir yazılım parçası ortaya kondu. Örneğin, 2. sprint sonunda takım uygulamanın bir APK derlemesini aldı ve gayri resmi olarak birkaç kullanıcı (örneğin ekip üyelerinin tanıdığı çocuklar veya danışman hoca) üzerinde denedi. Bu küçük teslimatlar, erken dönüt almak için önemliydi. Elde edilen geri bildirimler bir sonraki sprint planlamasına girdi olarak kullanıldı. Agile yaklaşımın ana avantajlarından biri, bu şekilde müşteri/son kullanıcı geri bildirimlerini geliştirme sürecine erken entegre edebilmektir. ARColorCathes için belki üniversite içindeki bir ara sunumda demo yapıldı ve jüriden/öğretmenden gelen öneriler sonraki sprintlerde uygulandı.
- **Değişikliklere Uyum:** Geliştirme sürecinde ortaya çıkan yeni fikirler veya gereksinimler esnek bir şekilde ele alındı. Örneğin, planlamada başta düşünülmeyen "sesli geri bildirim" özelliğinin önemli olduğu fark edildi diyelim (çocuklar henüz okumayı bilmeyebilir, bu yüzden uygulamanın sesli yönergeler vermesi gerekebilir). Bu, backlog'a eklendi ve önceliğine göre bir sonraki sprintte ele alındı. Agile metodoloji

sayesinde ekip, bu tür değişiklikleri sürece entegre ederken zorlanmadı; katı bir planı değiştirmek yerine zaten esnek olan backlog'u güncelliyordu. Gereksinim değişikliklerine geç bile olsa memnuniyetle cevap verme prensibi benimsendi, çünkü nihai hedef kullanıcı memnuniyetiydi.

- **Kod Kalitesi ve Refaktoring:** Her sprint sonunda (ve gerekiyorsa sprint ortalarında) ekip, yazdıkları kodu gözden geçirip iyileştirdi. Buna **refaktoring** denir. Refaktoring sırasında, davranışı değiştirmeden kod daha temiz ve optimize hale getirildi. Örneğin, "Kendin Yap" modundaki blok kontrol kodu çok karmaşıklaştıysa, bir sprint sonunda bu kod parçalanıp fonksiyonlara ayrıldı, gereksiz kısımlar atıldı. Bu tür sürekli temizlikler, geliştirme hızı yüksek tutulurken kalitenin de korunmasını sağladı. Agile'da teknik borç birikiminin önüne geçmek için bu uygulama şarttır; aksi halde ilerleyen sprintlerde verim düşebilir. Nitekim Agile prensiplerinden biri de ekiplerin sürdürülebilir bir tempoda çalışması ve teknik mükemmeliyete sürekli yatırım yapma.

Geliştirme aşamasının sonunda, planlanan tüm temel özellikler koda dökülmüş, test edilmiş ve entegre edilmiş oldu. Agile yaklaşımla ekip, projenin bitimine kadar potansiyel olarak her an çalışabilir durumda bir ürün sürümüne sahipti. Bu, proje sonunda sürprizlerle karşılaşmamak adına kritik bir yaklaşımdır; zira klasik yöntemlerdeki gibi en sona büyük bir entegrasyon yığılmadı, sürekli entegrasyon sağlandı.

Test Aşaması (Doğrulama ve Geçerleme)

Test aşaması, uygulamanın kalite kontrolünün yapıldığı, hata ayıklama ve doğrulama süreçlerinin yürütüldüğü kritik bir evredir. Agile modelde test etme, geliştirme ile iç içe geçmiş durumdadır – yani her iterasyonda testler yapılır, sadece sonunda değil. ARColorCathes projesinde test aktiviteleri şu şekilde gerçekleştirilmiştir:

- **Birimin Testleri (Unit Tests):** Unity ortamında saf C# kodu için birim testleri yazmak mümkündür (Unity Test Framework kullanılabilir). Projede kritik hesaplamalar veya oyun mantığı içeren fonksiyonlar için (özellikle modların kural setleri) birim testleri geliştirildi. Örneğin, **Alfabe Bilmece** modunda doğru harf seçimi kontrol eden fonksiyon varsa, bunun çeşitli senaryolar için (doğru harf, yanlış harf, büyük/küçük harf uyumsuzluğu vb.) birim testleri yazıldı. Bu testler otomatik olarak çalıştırılıp her derlemede kontrol edildi. Böylece, ileriki geliştirmelerde yanlışlıkla bu fonksiyonlar bozulursa hemen tespit edilebildi.

- **Entegrasyon ve Sistem Testleri:** ARColorCathes esasen bir mobil uygulama olduğu ve çok bileşenli (kamera, AR, UI vs.) yapıda olduğu için, bütünleşik testler daha önemliydi. Her sprint sonunda, ortaya çıkan ürünü takım üyeleri farklı cihazlarda manuel olarak test etti. Farklı marka ve model Android cihazlar denendi (Samsung, Huawei, Xiaomi vb. değişik donanımlar) ki uygulama genel sistem entegrasyonu olarak her yerde düzgün çalışıyor mu görülsün. Özellikle AR odaklı senaryolar test edildi:

- Kamera izni reddedilirse uygulama ne yapıyor? (Beklenen: Kibarca uyarı verip izinsiz AR'nin çalışmayacağını belirtmeli veya tekrar izin istemeli.)
- Düz zemin olmayan bir ortamda (örneğin halı veya dış mekânda) uygulamayı açarsak davranışı nedir? (Beklenen: Yüzey bulmakta zorlanabilir, bu durumda kullanıcıyı hareket ettirmeye yönlendirmeli veya bir mesaj göstermeli.)
- Uzun süre oyun oynandığında bellek kullanımı artıyor mu, uygulama yavaşlıyor mu? (Beklenen: Uzun kullanımda dahi kabul edilebilir performans.)
- Aynı anda birden fazla mod hızlıca açılıp kapatılırsa (mod değişimi) uygulama kararlı kalıyor mu? (Örneğin menüden hızlı şekilde modlar arası geçiş denendi.)

Bu gibi bütünsel testler, projenin farklı parçalarının birlikte uyum içinde çalıştığını doğruladı. Hatalar tespit edildikçe backlog'a **bug** olarak eklendi ve ilgili sprintte veya bir sonraki sprintte düzeltildi.

- **Kullanıcı Kabul Testleri:** Agile projelerde müşteri veya son kullanıcı da sürece dahil edilerek, belirli aralıklarla *kabul testleri* yapar. ARColorCathes akademik bir proje olsa da, kullanıcı rolünde düşünebileceğimiz danışman veya hedef kitleden pilot kullanıcılar ile denemeler yapıldı. Örneğin, proje belirli bir olgunluğa eriştiğinde bir ilkokulda birkaç öğrenciye uygulama denetildi. Çocukların uygulamayı anlama düzeyi, nerelerde takıldıkları, en çok neyi sevdikleri gözlemlendi. Bu gözlemler not alınarak projede iyileştirmeye açık alanlar belirlendi. Diyelim ki test sırasında fark edildi ki "Kırmızı Yeşil Mavi" modunda bazı çocuklar renkleri karıştırıyor; bu durumda tasarıma bir güncelleme yapılarak her nesneye şekil etiketleri de eklendi (kırmızı üçgen, yeşil kare gibi) ki renk körlüğü ihtimaline karşı veya pekiştirme amacıyla. Bu tür geri bildirimler projenin kullanıcı ihtiyaçlarına uygun hale gelmesini sağladı.
- **Otomatik Test ve Devamlı Entegrasyon:** Ekip, mümkün olduğunca otomasyonu da kullandı. GitHub üzerinde bir **CI (Continuous Integration)** ayarı yapılarak her koda gönderimde (push) projenin derlenebilir olduğu kontrol edildi ve birim testler otomatik koşturuldu. Unity için mevcut olan CI araçları sayesinde, ciddi hatalar anında görüldü. Örneğin, bir kütüphaneyi unutup import etmeme hatası ya da platforma özel bir derleyici hatası varsa, CI bunu yakaladı ve ekip düzeltene kadar ana sürüme entegre edilmedi. Bu, projenin her zaman çalışır durumda kalmasına yardımcı oldu.
- **Geribildirim Döngüsü:** Test aşamasının belki de en önemli yanı, hızlı geri bildirim sağlamasıdır. Agile yaklaşım gereği, hataların tespiti ve düzeltilmesi uzun süre bekletilmedi. Bulunan bir hata, ilgili geliştiriciye hemen atandı ve mümkünse aynı sprint içerisinde çözüldü. Örneğin, 4. sprintte bir bellek kaçağı (memory leak) tespit edildiyse, bu sprintin bitmesi beklenmeden acil öncelikle ele alındı; zira sonraki sprintlere sarkarsa katlanarak sorun çıkarabilirdi. Bu proaktif yaklaşım sayesinde final ürün daha sağlam hale geldi.

Test aşamasının sonunda (ya da daha doğru ifadeyle, proje tamamlanmaya yakın) ekip, projenin gereksinimlerine uygun çalıştığını, belirlenen tüm kabul kriterlerini sağladığını ve bilinen kritik hatalardan arınmış olduğunu doğruladı. Akademik bağlamda, bu aşama proje tesliminden önce danışmana yapılan son gösterimde de onaylandı. Böylece ARColorCathes'in hedeflenen kaliteye ulaştığı görülmüş oldu.

Dağıtım Aşaması (Deployment)

Dağıtım, uygulamanın son kullanıcılara ulaştırılmasını ifade eder. ARColorCathes için dağıtım aşaması, uygulamanın APK dosyası olarak paylaşılması ve potansiyel olarak bir uygulama mağazasına yüklenmesi adımlarını içerir. Agile projelerde dağıtım da yine yinelemeli olarak düşünülebilir (sık sık yeni sürüm yayınlama, kullanıcıdan geri bildirim alıp geliştirme döngüsüne devam etme şeklinde). Bu projedeki dağıtım faaliyetleri:

- **APK Oluşturma ve Sürümleme:** Geliştirme tamamlanıp testlerde başarılı sonuçlar alındığında, Unity üzerinden **Release Build** (yayınlama derlemesi) alındı. Bu APK dosyası, projenin son halini temsil eden **v1.0** sürüm numarasıyla etiketlendi (sürüm numaralandırma semantiği belirlenmişti: Örneğin v1.0.0 olarak ana sürüm, küçük güncellemeler için v1.0.1 vb.). Agile prensiplerinde, her iterasyonda çalışabilir bir ürün dilimimiz olduğundan aslında her sprint sonunda bir **iç dağıtım** yapılıyordu; fakat burada esas, bitişteki genel dağıtımdır. APK dosyası, proje paydaşlarıyla (danışman, sınıf arkadaşları, vs.) paylaşıldı.
- **Beta Dağıtım ve Geri Bildirim:** Zaman elverdiyse, proje bir **beta test** sürecine de sokuldu. Örneğin, proje Google Play Console'a "beta testi" kanalından yüklenerek, sınırlı sayıda kullanıcıya (örneğin proje ekibinin tanıdığı ebeveynler ve çocukları) dağıtıldı. Bu kullanıcılar uygulamayı gerçek ortamlarında deneyip görüş bildirdi. Beta testi, dağıtım öncesi son hataların ve iyileştirme fırsatlarının yakalanmasını sağlar. ARColorCathes için belki bu yolla birkaç cihaz uyumluluk sorunu veya küçük çeviri/metin hatası fark edildi ve düzeltilerek final sürüme yansıtıldı.
- **Resmi Dağıtım (Mağaza Yayını):** Projenin akademik teslimatı dışında gerçek dünyada da kullanıma sunulması hedeflendiyse, Google Play Store'a yükleme işlemleri yapıldı. Bunun için gerekli uygulama bilgileri (isim, açıklama, ekran görüntüleri, simge, kategori, yaş derecelendirmesi vs.) hazırlandı. Özelikle **çocuklara yönelik** bir uygulama olduğu için Play Store'un "Designed for Families" programı gibi ek gerekliliklerine dikkat edildi; içerik uygunluk beyanları doldurulduru. Yayınlama öncesi uygulamanın **Gizlilik Politikası** belgesi hazırlandı ve mağazaya eklendi (çocuk uygulamaları için gizlilik politikası şarttır, uygulama her ne kadar veri toplmasa da bunu beyan etmelidir). Ardından uygulama mağaza incelemesine gönderildi ve onaylandığında listelendi. Bu sayede daha geniş kitlelerin uygulamaya erişmesi sağlandı.
- **Dağıtım Sonrası İzleme:** Uygulama dağıtıldıktan sonra da iş bitmez; Agile düşüncede dağıtım sonrası gelen veriler yeni geliştirme döngülerini tetikleyebilir. Örneğin, Google

Play üzerindeki geri bildirimler (yorumlar, puanlar) veya crash raporları takip edildi. Eğer kullanıcılar belli bir modda zorluk yaşıyorsa veya bir cihazda çökme raporları geliyorsa, ekip hemen bunları backlog'a ekledi. Akademik proje olsa dahi, ekibin öğrenmesi ve projenin olgunlaşması için bu veriler değerlidir.

- **Son Teslimat (Dokümantasyon ile birlikte):** Akademik bağlamda, dağıtım aynı zamanda projenin jüriye/öğretmene sunulmasını da içerir. Bu nedenle, uygulamanın son derlemesi ile birlikte gereken tüm dokümanlar (kullanıcı kılavuzu, teknik rapor, sunum dosyası vs.) hazırlandı ve teslim edildi. Uygulamanın çalışır halde teslim edilmesi (örneğin bir Android cihaz üzerinde jüriye deneyimlettirilmesi) de projenin başarısı açısından önemli bir adımdı. Dağıtım aşaması, proje çıktılarının bütüncül bir paket haline getirilip ilgili kişilere ulaştırılmasıyla tamamlanmış oldu.

Bakım Aşaması (Maintenance ve Gelecek Geliştirmeler)

Bakım, uygulama yayınlandıktan veya teslim edildikten sonra devam eden iyileştirme, güncelleme ve hata düzeltme faaliyetlerini kapsar. Bir öğrenci projesi için bakım sınırlı ölçüde gerçekleşse de, ARColorCathes için gelecekte nelerin planlandığını ve projenin sürdürülebilirliğini nasıl sağladığını belirtmek önemlidir:

- **Hata Düzeltme ve Güncellemeler:** Dağıtımdan sonra tespit edilen hatalar için hızlı düzeltme güncellemeleri yayınlandı. Örneğin, kullanıcı raporlarına göre bazı cihazlarda kamera görüntüsünün siyah ekran kalması gibi bir sorun varsa, bunun nedeni araştırılıp (muhtemelen o cihaza özgü bir ARCore sorunu veya izin problemi olabilir) patch sürüm (v1.0.1 gibi) yayınlandı. Agile ortamda ekip, bakım döneminde de küçük sprintler yaparak biriken hataları ele alabilir. Bu, projenin canlı tutulmasını sağlar. Yine benzer şekilde, işletim sistemi güncellemeleri veya ARCore versiyon değişiklikleriyle uygulamada uyumluluk problemleri çıkarsa, kod güncellenerek yeni sürüm yayınlanacaktır.

- **İyileştirmeler ve Yeni Özellikler:** ARColorCathes, ileride genişletilebilecek bir projedir. Bakım aşamasında backlog'da kalan ancak zaman yetmediği için uygulanamayan bazı özellikler yeniden değerlendirmeye alınabilir. Örneğin, **yeni bir oyun modu** (sayıları öğretmeye yönelik bir mod gibi) ekleme fikri vardıysa, bu gelecek versiyonlar için planlanabilir. Ekip, kullanıcı geri bildirimlerinden öğrenerek belki en çok istenen geliştirmeleri önceliklendirir. Diyelim ki birden fazla kullanıcı "Keşke blokları boyayabilsek" diye yorum bıraktı; bu özellik backlog'da üst sıralara çıkarılıp üzerinde çalışılır. Agile yaklaşım, ürün yaşam döngüsü boyunca değer katmaya devam etmeyi öngörür, dolayısıyla her sürümde kullanıcı değerini artıracak bakımlar/güncellemeler planlanır.

- **Bakım Ekibi Organizasyonu:** Proje akademik olarak tamamlanmış olsa bile, açık kaynak yapıda GitHub'da kaldığı için, proje geliştiricisi(leri) dışındaki katkıda bulunmak isteyenler için zemin hazırlanmıştır. Kodun anlaşılır olması, dokümantasyonun

bulunması sayesinde başkaları projeyi fork edip geliştirebilir. Bu da bir nevi bakım/geliştirme sürecinin topluluk tabanlı devam etmesi demektir. Ekipten mezuniyet sonrası projeyi sürdürecek kimse olmazsa, kod deposunun açık olması projenin yaşamını uzatabilir.

- **Yasal ve Platform Değişiklikleri:** Bakım aşamasında ayrıca dış etkenlere de uyum sağlanır. Örneğin, Android işletim sistemi her yıl yeni sürümler çıkarır; uygulamanın yeni API seviyelerine uyumlu olması gerekir. Google Play politikaları değişebilir, çocuk uygulamalarına yeni gereklilikler gelebilir – uygulama bu gereksinimlere göre güncellenmelidir. ARCore'de büyük bir değişiklik olursa (API revizyonu gibi), uygulama kodu buna göre refaktör edilmelidir. Bu tür bakım faaliyetleri, uygulamanın ömrünü uzatır ve aktif kalmasını sağlar.
- **Son Kullanıcı Desteği:** Küçük çaplı bir proje olsa da, bakım kapsamında son kullanıcıların desteklenmesi de düşünülebilir. Web üzerinden veya mağaza üzerinden gelen sorulara yanıt vermek, gerektiğinde sorun yaşayan kullanıcılardan hata logları isteyip çözüm üretmek de bakımın parçasıdır. Bu, kullanıcı memnuniyetini yüksek tutar ve uygulamanın itibarını artırır.

Özetle, ARColorCathes projesinin SDLC planı Agile metodolojisi ile şekillendirilmiş ve her aşamada takım içi iletişim, geri bildirim odaklılık ve değişime uyum ön planda tutulmuştur. Bu sayede proje, dinamik bir süreç sonunda başarıyla tamamlanmış, kullanıcılarına değer sunan bir ürün haline gelmiştir. Gelecekte de Agile'ın prensiplerini uygulayarak güncelliğini ve faydasını korumaya devam edecektir.