Yazılım Mühendisliği
Yazılım Yaşam Döngüsü

Dr. Öğr. Üyesi Emrah ÖZKAYNAK

Yazılım Yaşam Döngüsü Adımları

- ✓1. Gereksinim Analizi (Requirement Analysis)
- ✓2. Fizibilite Çalışması (Feasibility Study)
- ✓**3. Sistem Tasarımı (System Design)**
- ✓4. Kodlama (Coding)
- ✓5. Test Etme (Testing)
- ✓6. Kurulum (Deployment)
- ✓7. Bakım (Maintenance)
- ✓8. Sonlandırma (Retirement)

Sistem Tasarımı

- ✓**Sistem Tasarımı** adımı, yazılım yaşam döngüsünde kritik bir rol oynar ve genel olarak, geliştirilecek yazılımın mimarisini ve yapısını belirleme sürecidir.

- ✓Bu adım, yazılımın nasıl çalışacağını ve hangi bileşenlerden oluşacağını detaylı bir şekilde tanımlayarak projenin doğru ve verimli bir şekilde ilerlemesini sağlar.

Sistem Tasarımı

✓**Sistem Tasarımının Önemi**

- **Doğru planlama**: Sistem tasarımı aşaması, yazılımın nasıl çalışacağına dair doğru planlamalar yapılmasını sağlar ve sonradan ortaya çıkabilecek problemlerin önüne geçer.
- **Maliyet ve zaman yönetimi**: İyi bir tasarım, geliştirme sürecinde hataların azaltılmasına ve bu sayede maliyet ve zaman kayıplarının önlenmesine yardımcı olur.
- **Geleceğe yönelik genişletilebilirlik**: Tasarım aşamasında yapılan doğru seçimler, yazılımın ileride genişletilmesini veya modifiye edilmesini kolaylaştırır.

✓Sonuç olarak, sistem tasarımı, yazılımın başarılı bir şekilde geliştirilmesinin temelini oluşturan bir adım olup, yazılımın kalite, sürdürülebilirlik ve başarısı açısından kritik bir role sahiptir.

Sistem Tasarımı

- ✓Sistem tasarımı iki ana seviyede gerçekleştirilir:
- ✓**1. Yüksek Seviyeli Tasarım (High-Level Design - HLD)**
- ✓Bu aşama, yazılımın genel mimarisini, büyük bileşenlerin nasıl etkileşimde bulunacağını ve sistemin üst düzey yapılarını belirler.

- ✓**2. Detaylı Tasarım (Low-Level Design - LLD)**
- ✓Bu aşama, yüksek seviyeli tasarımın daha ayrıntılı hale getirildiği bölümdür. Her bir modülün iç yapısı, algoritmaları, veri yapıları, sınıfları ve işlevleri tanımlanır.

Sistem Tasarımı

✓**1. Yüksek Seviyeli Tasarım (High-Level Design - HLD)**

✓Bu tasarım aşamasında genel olarak şu unsurlar ele alınır:

- **Modüler yapı**: Yazılımın hangi ana modüllerden oluşacağı belirlenir.
- **Sistem mimarisi**: Sistemin hangi donanım ve yazılım bileşenleri üzerinde çalışacağı, kullanılan teknoloji yığınları tanımlanır.
- **Veri akışı**: Modüller arasındaki veri akışı ve etkileşimler belirlenir.
- **Dış sistemlerle entegrasyon**: Yazılımın diğer sistemlerle nasıl entegre olacağı tanımlanır.

✓Bu adım, proje ekiplerine genel bir bakış sunarak sistemin karmaşıklığını anlamalarına yardımcı olur. Genelde yazılım mimarları ve kıdemli mühendisler bu aşamada rol oynar.

Sistem Tasarımı

✓**2. Detaylı Tasarım (Low-Level Design - LLD)**

✓Bu aşama, yüksek seviyeli tasarımın daha ayrıntılı hale getirildiği bölümdür. Her bir modülün iç yapısı, algoritmaları, veri yapıları, sınıfları ve işlevleri tanımlanır. Şu detaylar üzerinde durulur:

- **Veritabanı tasarımı**: Tablolar, ilişkiler ve veri yapıları detaylandırılır.
- **Algoritmalar**: Her modülün içindeki işlemlerin nasıl yapılacağı tanımlanır.
- **Arayüzler ve API'ler**: Modüller arası iletişim için kullanılacak arayüzler belirlenir.
- **Hata yönetimi**: Yazılımda olası hata durumları nasıl ele alınacak, belirlenir.

Sistem Tasarımı

✓1. **Modül Tasarımı (Module Design)**

✓Her modülün iç yapısı detaylandırılır ve hangi sorumluluklara sahip olduğu belirlenir. Modülün işleyişi, diğer modüllerle nasıl etkileşimde bulunacağı ve veri akışını nasıl yöneteceği tanımlanır.

- **Modül Sorumlulukları**: Her bir modülün spesifik bir görevi vardır. Bu modül örneğin bir kullanıcı giriş sistemini, veri yönetimini veya raporlama süreçlerini yönetebilir.
- **Modüller Arası İlişkiler**: Modüller arasındaki bağımlılıklar, veri akışları ve etkileşimler detaylandırılır. Hangi modüllerin birbirleriyle nasıl iletişim kuracakları açıklanır.

Sistem Tasarımı

✓1. **Modül Tasarımı (Module Design)**

✓Her modülün iç yapısı detaylandırılır ve hangi sorumluluklara sahip olduğu belirlenir. Modülün işleyişi, diğer modüllerle nasıl etkileşimde bulunacağı ve veri akışını nasıl yöneteceği tanımlanır.

- **Modül Sorumlulukları**: Her bir modülün spesifik bir görevi vardır. Bu modül örneğin bir kullanıcı giriş sistemini, veri yönetimini veya raporlama süreçlerini yönetebilir.
- **Modüller Arası İlişkiler**: Modüller arasındaki bağımlılıklar, veri akışları ve etkileşimler detaylandırılır. Hangi modüllerin birbirleriyle nasıl iletişim kuracakları açıklanır.

Sistem Tasarımı

✓**2. Sınıf Diyagramları (Class Diagrams)**

✓Sınıf diyagramları, yazılımın hangi sınıfları ve nesneleri içereceğini, bu sınıfların birbirleriyle nasıl ilişkili olacağını ve sınıfların hangi özellik ve yöntemlere sahip olacağını gösterir. Nesneye dayalı programlama (OOP) metodolojisine dayalı olarak sınıflar şu şekilde detaylandırılır:

- **Sınıf İsimleri**: Her sınıfın adı ve sorumluluğu tanımlanır.
- **Nitelikler ve Metodlar**: Her sınıfın hangi özelliklere (nitelikler/attributes) ve hangi işlemlere (metodlar/methods) sahip olduğu belirtilir.
- **İlişkiler**: Sınıflar arasındaki ilişkiler (miras alma, bileşim, birleşim gibi) ve bu sınıfların birbirleriyle nasıl etkileşimde bulunduğu gösterilir.

Sistem Tasarımı

✓**3. Algoritma ve Akış Diyagramları (Algorithm and Flow Diagrams)**

✓Yazılımın belirli işlemleri nasıl gerçekleştireceği ve adım adım nasıl ilerleyeceği, algoritmalar ve akış diyagramlarıyla gösterilir. Bu aşamada:

- **Algoritmalar**: Modüllerin ya da fonksiyonların belirli bir görevi yerine getirmesi için kullanılacak adımlar, algoritmalar şeklinde tanımlanır. Örneğin, bir sıralama işlemi ya da veri doğrulama işlemi için algoritmalar belirlenir.
- **Akış Diyagramları**: Algoritmaların nasıl işlediğini göstermek için adım adım veri akışını ve işlem adımlarını gösteren diyagramlar kullanılır. Bu diyagramlar, geliştiricilerin süreci daha kolay anlamasını sağlar.

Sistem Tasarımı

✓**4. Veri Yapıları (Data Structures)**

✓Veri yapıları, yazılımda hangi veri türlerinin kullanılacağı ve bu verilerin nasıl organize edileceği ile ilgilidir. Detaylı tasarımda, kullanılan veri yapılarının türleri, ilişkileri ve yapıları tanımlanır:

- **Diziler, listeler, kümeler ve haritalar**: Kullanılacak temel veri yapılarının hangi durumlarda kullanılacağı belirlenir.
- **Veri ilişkileri**: Verilerin birbiriyle nasıl ilişkili olduğu ve hangi yapıların veriyi depolamak için en uygun olduğu açıklanır.

Sistem Tasarımı

✓**5. Veritabanı Tasarımı (Database Design)**

✓Detaylı tasarımın önemli bir kısmı da veritabanı tasarımıdır. Veritabanı yapıları, tablolar ve veri modelleri detaylandırılır:

- **Tablolar**: Hangi verilerin hangi tablolarda saklanacağı, bu tabloların alanları ve alan türleri belirlenir.
- **İlişkiler**: Tablolar arasındaki ilişkiler (birden bire, birden çoğa vb.) detaylı şekilde gösterilir. Bu ilişkiler, normalizasyon işlemleri ve veri tutarlılığı göz önünde bulundurularak düzenlenir.
- **SQL veya NoSQL kullanımı**: İhtiyaca göre hangi veritabanı türünün kullanılacağı, verilerin nasıl sorgulanacağı ve nasıl işleneceği tasarlanır.

Sistem Tasarımı

✓**6. Arayüzler ve API'ler (Interfaces and APIs)**

✓Modüller ve bileşenler arasındaki etkileşimi sağlamak için kullanılacak arayüzler ve API'ler belirlenir. Bu aşamada şunlar planlanır:

- **Fonksiyonlar ve Parametreler**: Her bir arayüzün hangi işlevleri sağlayacağı, hangi parametrelerle çağrılacağı belirlenir.
- **Giriş ve Çıkışlar**: Her bir fonksiyonun ya da API'nin aldığı girdiler ve sağladığı çıktılar detaylandırılır.
- **API Protokolleri**: REST, GraphQL veya SOAP gibi API'lerin kullanılacağı protokoller belirlenir.

Sistem Tasarımı

✓**7. Hata Yönetimi (Error Handling)**

✓Detaylı tasarım aşamasında, yazılımda karşılaşılabilecek hata durumlarında ne yapılacağı ve hataların nasıl ele alınacağı da belirlenir:

- **Hata Kodları**: Yazılımın karşılaşabileceği hata türleri ve bu hatalar için kullanılacak hata kodları (örn. HTTP 404, 500 hataları) belirlenir.
- **Hata Yakalama Mekanizmaları**: Hataların nasıl yakalanacağı, kayıt altına alınacağı ve yönetileceği (örneğin try-catch blokları) tanımlanır.
- **Hata Loglama**: Hataların takip edilmesi ve incelenebilmesi için loglama (kayıt tutma) mekanizmaları oluşturulur.

Sistem Tasarımı

✓**8. Performans Optimizasyonu (Performance Optimization)**

✓Yazılımın performansının optimize edilmesi için hangi stratejilerin uygulanacağı detaylı şekilde planlanır:

- **Önbellekleme (Caching)**: Hangi verilerin, hangi işlemlerin önbelleğe alınacağı ve hangi önbellekleme yöntemlerinin kullanılacağı (örn. Redis, Memcached) belirlenir.
- **Veri Yoğunluklu İşlemler**: Yüksek veri akışına sahip işlemlerde performansın nasıl optimize edileceği, örneğin indeksleme, sorgu optimizasyonu gibi stratejilerle açıklanır.
- **Yük Dengeleme**: Sistem üzerindeki yükün nasıl dengeleneceği ve sistem performansının nasıl sürdürülebileceği planlanır.

Sistem Tasarımı

✓**9. Güvenlik (Security)**

✓Yazılımın güvenliği, detaylı tasarımda teknik seviyede ele alınır. Kullanıcı verilerinin korunması ve sistemin güvenliğinin sağlanması amacıyla şu unsurlar detaylandırılır:

- **Kimlik Doğrulama ve Yetkilendirme**: Kullanıcıların kimlik doğrulama işlemleri (örneğin, JWT, OAuth) ve hangi kaynaklara erişim izni olduğu belirlenir.
- **Şifreleme**: Verilerin saklanması veya transfer edilmesi sırasında hangi şifreleme yöntemlerinin kullanılacağı (örn. AES, RSA, SSL/TLS) açıklanır.
- **Güvenlik Açıkları**: Potansiyel güvenlik açıklarına karşı önlemler (örn. SQL enjeksiyonu, XSS gibi) detaylandırılır.

Sistem Tasarımı

✓**10. Kapsamlı Test Planları (Comprehensive Testing Plans)**

✓Detaylı tasarım, yazılımın nasıl test edileceği ile ilgili ayrıntıları da içerir. Test süreçleri ve stratejileri detaylandırılır:

- **Birim Testleri (Unit Testing)**: Her modül ve fonksiyon için nasıl birim testlerinin yapılacağı ve bu testlerin kapsamı belirlenir.
- **Entegrasyon Testleri (Integration Testing)**: Modüller arası etkileşimlerin nasıl test edileceği ve olası sorunların nasıl önleneceği planlanır.
- **Sistem ve Performans Testleri**: Yazılımın genel performansının ve sistemin genel stabilitesinin nasıl test edileceği açıklanır.

Sistem Tasarımı

✓**11. Kodlama Standartları (Coding Standards)**

✓Yazılımın tutarlı ve sürdürülebilir olması için hangi kodlama standartlarına uyulacağı belirlenir. Bu aşamada:

- **İsimlendirme Kuralları**: Sınıflar, değişkenler, fonksiyonlar ve diğer kod elemanları için isimlendirme standartları belirlenir.
- **Yazım Stili**: Kodun nasıl yazılacağı, hangi format kurallarına uyulacağı (örneğin, girintileme, yorum ekleme) detaylandırılır.
- **Dokümantasyon**: Kodun nasıl dokümante edileceği ve geliştiricilere rehberlik edecek belgelerin nasıl hazırlanacağı planlanır.

Sistem Tasarımı

✓**12. Hata Ayıklama ve İzleme (Debugging and Monitoring)**

✓Sistemin geliştirilmesi ve çalışması sırasında ortaya çıkabilecek sorunları izlemek ve çözmek için kullanılacak hata ayıklama araçları ve izleme yöntemleri belirlenir:

- **Hata Ayıklama Araçları**: Kodun doğru çalışıp çalışmadığını görmek için kullanılacak hata ayıklama araçları belirlenir.
- **Sistem İzleme**: Uygulamanın performansını ve stabilitesini izlemek için kullanılacak izleme araçları (örn. Grafana, Kibana) ve metrikler tanımlanır.

Sistem Tasarımı – Örnek Proje : Online Kitap Satış Platformu

- ✓ **1. Yüksek Seviyeli Tasarım (High-Level Design - HLD)**
- ✓ Bu aşamada, sistemin genel yapısını ve ana bileşenlerini tasarlıyoruz. Bu, sistemin nasıl yapılandırılacağını ve hangi modüllere ayrılacağını belirlememizi sağlar.

Sistem Tasarımı – Örnek Proje : Online Kitap Satış Platformu

- ✓ **1. Yüksek Seviyeli Tasarım (High-Level Design - HLD)**
- ✓ Bu aşamada, sistemin genel yapısını ve ana bileşenlerini tasarlıyoruz. Bu, sistemin nasıl yapılandırılacağını ve hangi modüllere ayrılacağını belirlememizi sağlar.

Sistem Tasarımı – Örnek Proje : Online Kitap Satış Platformu

✓ **a) Sistem Mimarisi**

✓ Sistem üç ana bileşenden oluşur:

1. **Kullanıcı Arayüzü (UI/UX)**
    1. Web tabanlı olacak ve kullanıcılar bu arayüz üzerinden kitap satın alabilecek, yorum yapabilecek ve hesaplarını yönetebilecekler.
    2. Arayüz modern JavaScript framework'ü (örn. React.js veya Vue.js) ile geliştirilecek.
2. **Sunucu Katmanı (Back-End)**
    1. İş mantığının ve veri işleme süreçlerinin yönetileceği yer olacak.
    2. RESTful API ile kullanıcı talepleri işlenecek. Python Django veya Node.js kullanılabilir.
3. **Veritabanı Katmanı**
    1. Tüm kitap verileri, kullanıcı bilgileri, siparişler ve yorumlar bu katmanda saklanacak.
    2. SQL tabanlı (örn. PostgreSQL veya MySQL) bir veritabanı kullanılacak.
4. **Üçüncü Parti Servisler**
    1. **Ödeme Sistemi**: Stripe veya PayPal gibi üçüncü parti ödeme sağlayıcıları kullanılacak.
    2. **Tavsiye Motoru**: Kitap tavsiyeleri için makine öğrenimi modeli veya dış bir API entegrasyonu yapılacak.

Sistem Tasarımı – Örnek Proje : Online Kitap Satış Platformu

✓**b) Modüller**

✓Sistemi temel modüllere bölelim:

1. **Kullanıcı Yönetimi Modülü**: Kullanıcı kayıt, giriş, profil düzenleme, şifre sıfırlama işlemleri.
2. **Kitap Yönetimi Modülü**: Kitapları listeleme, arama, detaylarını görüntüleme.
3. **Alışveriş Sepeti ve Satın Alma Modülü**: Kullanıcıların kitapları sepete ekleyip ödeme işlemlerini yapabilmeleri.
4. **Yorum ve Puanlama Modülü**: Kullanıcıların kitaplar hakkında yorum yapıp puan verebilecekleri sistem.
5. **Yönetici Modülü**: Kitap ekleme, stok yönetimi ve siparişleri izleme.
6. **Bildirim ve Tavsiye Modülü**: Tavsiye edilen kitapları ve bildirimleri kullanıcılara gösterme.

Sistem Tasarımı – Örnek Proje : Online Kitap Satış Platformu

✓**c) Veri Akışı**

✓Kullanıcıların sistemi nasıl kullanacağını bir örnek üzerinden görelim:

1. Kullanıcı siteye giriş yapar.
2. Kitap araması yapar ve bir kitabın detay sayfasına gider.
3. Kitabı sepete ekler ve ödeme adımlarına geçer.
4. Ödeme başarılı olursa, sipariş tamamlanır.
5. Kullanıcı siparişini profili altındaki siparişler bölümünde görüntüler.

Sistem Tasarımı – Örnek Proje : Online Kitap Satış Platformu

- ✓ **2. Detaylı Tasarım (Low-Level Design - LLD)**
- ✓ Bu aşamada, her modül ve bileşen için daha ayrıntılı bir tasarım yapıyoruz.

Sistem Tasarımı – Örnek Proje : Online Kitap Satış Platformu

✓ **a) Modül Tasarımı**

✓ Her bir modülün nasıl çalışacağını ve alt modüllerini detaylandırmamız gerekiyor.

1. Kullanıcı Yönetimi Modülü

1. **Kayıt İşlemi**: Kullanıcı formu doldurur, e-posta doğrulaması yapılır.
2. **Giriş İşlemi**: JWT (JSON Web Token) ile kullanıcı oturumu güvenli şekilde yönetilir.
3. **Profil Düzenleme**: Kullanıcı kendi bilgilerini güncelleyebilir.

2. Kitap Yönetimi Modülü

1. **Kitap Listeleme**: Tüm kitaplar ana sayfada listelenir. Arama ve filtreleme fonksiyonları ile belirli türlerde kitaplar aranabilir.
2. **Kitap Detay Sayfası**: Kitap ismi, yazar, fiyat, kullanıcı yorumları ve puanlama bilgileri burada görüntülenir.

3. Alışveriş Sepeti ve Satın Alma Modülü

1. **Sepet Yönetimi**: Kullanıcılar bir veya daha fazla kitabı sepete ekleyebilir. Sepette kitap sayısı güncellenebilir veya kitap silinebilir.
2. **Ödeme İşlemi**: Kullanıcı Stripe API'si üzerinden ödeme yapar. Başarılı bir ödeme durumunda sipariş veritabanına kaydedilir ve kullanıcıya onay mesajı gönderilir.

4. Yorum ve Puanlama Modülü

1. **Yorum Ekleme**: Kullanıcılar sadece satın aldıkları kitaplara yorum yapabilir.
2. **Puanlama**: Kitaplar için 1-5 arasında puan verilebilir. Kitabın ortalama puanı görüntülenir.

Sistem Tasarımı – Örnek Proje : Online Kitap Satış Platformu

✓**b) Sınıf Diyagramları**

✓Sınıf diyagramları, sistemdeki sınıfları ve bunların birbirleriyle ilişkilerini tanımlar. Örnek olarak birkaç sınıf:

- **User**: Kullanıcıyı temsil eden sınıf. Attributeleri: kullanıcı adı, e-posta, şifre, sipariş geçmişi.
- **Book**: Kitap sınıfı. Attributeleri: kitap adı, yazar, tür, fiyat, stok durumu, yorumlar.
- **Order**: Sipariş sınıfı. Attributeleri: sipariş numarası, kullanıcı, kitaplar, toplam tutar, sipariş durumu.

Sistem Tasarımı – Örnek Proje : Online Kitap Satış Platformu

✓**c) Veritabanı Tasarımı**

✓Veritabanı yapısını tanımlayalım:

1. **Users** tablosu: Kullanıcı bilgilerini içerir (id, ad, soyad, e-posta, şifre).
2. **Books** tablosu: Kitap bilgilerini içerir (id, isim, yazar, fiyat, stok, kategori).
3. **Orders** tablosu: Sipariş bilgilerini içerir (id, kullanıcı_id, toplam_tutar, sipariş_tarihi).
4. **Reviews** tablosu: Yorum ve puanları içerir (id, kitap_id, kullanıcı_id, puan, yorum).

Sistem Tasarımı – Örnek Proje : Online Kitap Satış Platformu

✓**d) API Tasarımı**

✓RESTful API ile veritabanı ve arayüz arasındaki veri alışverişini yönetelim. Örnek bazı API uç noktaları:

- **POST /login**: Kullanıcı girişi.
- **GET /books**: Tüm kitapları listele.
- **POST /cart**: Sepete kitap ekle.
- **POST /checkout**: Ödeme yap ve siparişi tamamla.

Sistem Tasarımı – Örnek Proje : Online Kitap Satış Platformu

- ✓ **e) Algoritma ve Akış Diyagramları**
- ✓ Kullanıcı bir kitap satın aldığında şu adımları izleyen bir algoritma çalışır:

1. Kullanıcı sepetteki kitapları onaylar ve ödeme sayfasına yönlendirilir.
2. Stripe API ile ödeme doğrulanır.
3. Ödeme başarılı ise, sipariş veritabanına kaydedilir ve kullanıcıya e-posta ile onay gönderilir.

Sistem Tasarımı – Örnek Proje : Online Kitap Satış Platformu

✓**f) Hata Yönetimi**

- Ödeme başarısız olduğunda kullanıcıya hata mesajı gösterilir.
- Yetersiz stok durumunda sepet güncellenir ve kullanıcıya bildirilir.

Sistem Tasarımı – Örnek Proje : Online Kitap Satış Platformu

✓**g) Performans Optimizasyonu**

- **Önbellekleme**: Popüler kitaplar Redis kullanılarak önbelleğe alınır.
- **Sorgu Optimizasyonu**: Veritabanı sorguları için indeksleme yapılır.

Sistem Tasarımı – Örnek Proje : Online Kitap Satış Platformu

✓**g) Performans Optimizasyonu**

- **Önbellekleme**: Popüler kitaplar Redis kullanılarak önbelleğe alınır.
- **Sorgu Optimizasyonu**: Veritabanı sorguları için indeksleme yapılır.

Sistem Tasarımı – Örnek Proje : Online Kitap Satış Platformu

✓**h) Güvenlik**

- Kullanıcı şifreleri şifrelenir ve güvenli bir şekilde saklanır.
- API'ler JWT ile korunur.
- HTTPS kullanılarak veri transferleri güvenli hale getirilir.

Sistem Tasarımı – Örnek Proje 2 : Online Yemek Sipariş Platformu

- ✓ **Proje Hedefi:**
- ✓ Kullanıcıların restoranlardan online olarak yemek sipariş verebilecekleri, sipariş sürecini takip edebilecekleri ve restoranların menülerini yönetip siparişleri işleyebilecekleri bir platform geliştirmek.
- ✓ **1. Yüksek Seviyeli Tasarım (High-Level Design - HLD)**
- ✓ Bu aşamada, sistemin genel yapısı ve ana bileşenlerini tasarlıyoruz.

Sistem Tasarımı – Örnek Proje 2 : Online Yemek Sipariş Platformu

✓ **a) Sistem Mimarisi**

✓ Sistem üç ana bileşenden oluşacak:

1. Kullanıcı Arayüzü (Frontend)

1. Kullanıcılar ve restoranlar için web veya mobil arayüz olacak.
2. Kullanıcı arayüzü yemek siparişi verme, restoran listelerini görme, sepeti yönetme ve siparişi takip etme işlemlerini içerecek.
3. Restoranlar ise menü yönetimi ve sipariş takibi için bir panel kullanacak.
4. Angular, React.js veya mobil uygulamalar için Flutter gibi frontend teknolojileri kullanılabilir.

2. Sunucu Katmanı (Backend)

1. İş mantığının yönetileceği yer olacak.
2. Restoranlar, menüler, siparişler, kullanıcılar ve ödeme işlemlerini yönetecek bir API sağlayacak.
3. Java Spring Boot, Node.js veya Python Django gibi backend framework'leri kullanılabilir.

3. Veritabanı Katmanı

1. Kullanıcılar, restoranlar, yemekler, siparişler ve ödemelerle ilgili veriler saklanacak.
2. SQL tabanlı (PostgreSQL, MySQL) veya NoSQL (MongoDB) kullanılabilir.

4. Üçüncü Parti Servisler

1. **Ödeme Servisi**: Stripe, PayPal veya yerel bir ödeme sağlayıcı ile ödeme işlemleri yapılacak.
2. **Harita ve Adres Servisleri**: Google Maps API kullanılarak adresler doğrulanacak ve teslimat güzergahları belirlenecek.

Sistem Tasarımı – Örnek Proje 2 : Online Yemek Sipariş Platformu

✓**b) Modüller**

✓Sistemi temel modüllere bölelim:

1. **Kullanıcı Yönetimi Modülü**: Kullanıcı kayıt, giriş, profil düzenleme.
2. **Restoran Yönetimi Modülü**: Restoranlar menülerini yönetebilir, yeni yemekler ekleyebilir ve fiyatları güncelleyebilir.
3. **Yemek ve Menü Yönetimi Modülü**: Her restoranın menüsünde yer alan yemekleri listeleme, kategoriye göre filtreleme ve detaylarını görüntüleme.
4. **Sipariş Yönetimi Modülü**: Kullanıcılar sepeti yönetir, sipariş verir ve siparişin durumunu takip eder.
5. **Ödeme Yönetimi Modülü**: Kullanıcılar ödeme işlemlerini gerçekleştirir.
6. **Yorum ve Puanlama Modülü**: Kullanıcılar restoranlar ve yemekler hakkında yorum yapabilir ve puan verebilir.
7. **Restoran Paneli**: Restoranlar siparişleri görüntüler, onaylar ve teslimat sürecini yönetir.

Sistem Tasarımı – Örnek Proje 2 : Online Yemek Sipariş Platformu

✓**c) Veri Akışı**

✓Kullanıcıların sistemi nasıl kullanacağını basit bir örnekle inceleyelim:

1. Kullanıcı siteye giriş yapar.
2. Yakındaki restoranları görüntüler ve bir restorandan yemek seçer.
3. Seçilen yemekler sepete eklenir ve kullanıcı ödeme adımına geçer.
4. Ödeme işlemi tamamlandığında sipariş oluşturulur ve kullanıcı sipariş takibini başlatır.
5. Restoran siparişi onaylar ve yemek hazırlanmaya başlar.
6. Teslimat yapıldıktan sonra kullanıcı siparişi kapatabilir ve yorum yapabilir.

Sistem Tasarımı – Örnek Proje 2 : Online Yemek Sipariş Platformu

- ✓ **2. Detaylı Tasarım (Low-Level Design - LLD)**
- ✓ Bu aşamada, her modül ve bileşen için daha ayrıntılı bir tasarım yapıyoruz.

Sistem Tasarımı – Örnek Proje 2 : Online Yemek Sipariş Platformu

✓ **a) Modül Tasarımı**

✓ Her bir modülün nasıl çalışacağını ve alt modüllerini detaylandırmamız gerekiyor.

1. Kullanıcı Yönetimi Modülü

1. **Kayıt ve Giriş İşlemleri**: Kullanıcı, e-posta doğrulaması ile sisteme kayıt olabilir ve giriş yapabilir.
2. **Profil Düzenleme**: Kullanıcı, adres bilgilerini ve ödeme bilgilerini güncelleyebilir.

2. Restoran Yönetimi Modülü

1. **Menü Yönetimi**: Restoran, menüsüne yemek ekleyebilir, fiyatları güncelleyebilir ve yemekleri düzenleyebilir.
2. **Sipariş Yönetimi**: Restoran, aldığı siparişleri görüntüleyebilir, siparişi hazırlamaya başlayabilir ve teslimat sürecini yönetebilir.

3. Yemek ve Menü Yönetimi Modülü

1. **Yemek Listeleme**: Kullanıcı, restoranların menüsünü görüntüler. Yemekler fiyat, puan, kategori gibi filtrelerle sıralanabilir.
2. **Yemek Detayı**: Yemek ismi, malzemeler, fiyat, kalori bilgisi ve restoran bilgileri görüntülenir.

4. Sipariş Yönetimi Modülü

1. **Sepet Yönetimi**: Kullanıcılar bir veya daha fazla yemeği sepete ekleyebilir. Sepet güncellenebilir ya da temizlenebilir.
2. **Sipariş Takibi**: Siparişin durumu (hazırlanıyor, yolda, teslim edildi) kullanıcıya gösterilir.

5. Ödeme Yönetimi Modülü

1. **Ödeme İşlemleri**: Kullanıcılar, Stripe API veya benzeri bir ödeme servisi aracılığıyla ödemelerini tamamlayabilirler.
2. **Ödeme Onayı**: Ödeme başarılı olduğunda sipariş veritabanına kaydedilir ve kullanıcıya e-posta ile onay gönderilir.

6. Yorum ve Puanlama Modülü

1. **Yorum Yapma**: Kullanıcılar sadece sipariş verdikleri restoranlara ve yemeklere yorum yapabilirler.
2. **Puanlama**: Her yemek ve restoran için 1-5 arası puanlama yapılabilir.

Sistem Tasarımı – Örnek Proje 2 : Online Yemek Sipariş Platformu

✓**b) Sınıf Diyagramları**

✓Sınıf diyagramları, sistemdeki sınıfları ve ilişkilerini tanımlar. Örnek bazı sınıflar:

- **User**: Kullanıcıyı temsil eder. Attributeleri: kullanıcı adı, e-posta, şifre, adresler.
- **Restaurant**: Restoranı temsil eder. Attributeleri: isim, adres, menü, puanlama.
- **Order**: Siparişleri temsil eder. Attributeleri: sipariş numarası, kullanıcı, restoran, yemekler, durum.

Sistem Tasarımı – Örnek Proje 2 : Online Yemek Sipariş Platformu

✓**c) Veritabanı Tasarımı**

✓Veritabanı yapısını tanımlayalım:

1. **Users** tablosu: Kullanıcı bilgilerini içerir (id, ad, e-posta, şifre, adres).
2. **Restaurants** tablosu: Restoran bilgilerini içerir (id, isim, adres, puan).
3. **Dishes** tablosu: Yemek bilgilerini içerir (id, restoran_id, isim, fiyat, kategori).
4. **Orders** tablosu: Sipariş bilgilerini içerir (id, kullanıcı_id, restoran_id, toplam_tutar, sipariş_tarihi, sipariş_durumu).
5. **Reviews** tablosu: Yorum ve puanları içerir (id, yemek_id, kullanıcı_id, puan, yorum).

Sistem Tasarımı – Örnek Proje 2 : Online Yemek Sipariş Platformu

✓**d) API Tasarımı**

✓RESTful API ile veritabanı ve arayüz arasındaki veri alışverişini yönetelim. Örnek API uç noktaları:

- **POST /login**: Kullanıcı girişi.
- **GET /restaurants**: Restoranları listele.
- **POST /order**: Yeni sipariş oluştur.
- **GET /order/**
- : Sipariş durumunu görüntüle.

Sistem Tasarımı – Örnek Proje 2 : Online Yemek Sipariş Platformu

✓**e) Algoritma ve Akış Diyagramları**

✓Kullanıcı bir yemek siparişi verdiğinde şu adımları izleyen bir algoritma çalışır:

1. Kullanıcı sepetini onaylar ve ödeme sayfasına yönlendirilir.
2. Ödeme Stripe API ile doğrulanır.
3. Ödeme başarılıysa sipariş veritabanına kaydedilir ve restoran bilgilendirilir.
4. Restoran, sipariş durumunu günceller (hazırlanıyor -> teslim ediliyor -> teslim edildi).
5. Teslimat sonrası kullanıcıya teslimat bildirimi yapılır.

Sistem Tasarımı – Örnek Proje 2 : Online Yemek Sipariş Platformu

✓**f) Hata Yönetimi**

- Ödeme başarısız olduğunda kullanıcıya hata mesajı gösterilir.
- Yetersiz stok veya restoranın hizmet dışı olduğu durumlar kullanıcıya bildirilir.

Sistem Tasarımı – Örnek Proje 2 : Online Yemek Sipariş Platformu

✓ **g) Performans Optimizasyonu**

- **Önbellekleme**: Popüler restoranlar ve yemekler Redis kullanılarak önbelleğe alınır.
- **Sorgu Optimizasyonu**: Restoran aramaları ve yemek listeleri için verit

LAB Ödevi

- ✓Üniversite Öğrenci Bilgi Sistemi (ÖBS) Sistem Tasarım Adımlarını Uygulayınız