**“E-ticaret Web Uygulaması”**

1. **Modüler ve Büyüyebilir Yapı**: E-ticaret sistemi, birçok modülden oluşur (ürün yönetimi, kullanıcı oturumu, ödeme sistemleri vb.), bu da ekip üyeleri arasında görevleri rahatça bölüştürmeyi sağlar. Herkes farklı bir modül üzerinde çalışabilir.

2. **Birçok Tasarım Desenini Uygulama İmkanı**: MVC, Factory, Singleton, Observer gibi desenler bu proje için doğal olarak kullanılabilir, bu da projenizin daha yapılandırılmış ve sürdürülebilir olmasını sağlar.

3. **Gerçek Dünya Problemleri ve Pratik Deneyim**: E-ticaret sistemleri gerçek dünyada sık karşılaşılan projelerdir. Bu nedenle proje tamamlandığında hem öğrenme açısından faydalı olacak hem de iş dünyasında kullanılabilir bir deneyim kazanmış olacaksınız.

4. **Zengin Özellik Seti**: Sipariş yönetimi, ödeme sistemleri, kullanıcı girişi gibi birden fazla özelliği olan bir uygulama, hem ekip içinde görev dağılımını iyi yapmanıza olanak sağlar hem de işinizi çeşitlendirir.

**Önerilen Görev Dağılımı:**

• **Backend (API ve Veritabanı Yönetimi)**: Sipariş, ürün ve kullanıcı yönetimi için API oluşturma.

• **Frontend (Kullanıcı Arayüzü)**: Kullanıcı dostu arayüz tasarımı ve veri gösterimi.

• **Ödeme Sistemleri ve Güvenlik**: Ödeme sistemleri ve kullanıcı oturumlarının güvenliği.

• **Test ve DevOps**: Testlerin yazılması, uygulamanın dağıtımı ve performans izleme.

**1. Proje Yapısı ve Modüller**

Projeyi modüler bir yapıda tasarlayarak her bir modül üzerinde ayrı ayrı çalışabilirsiniz. İşte temel modüller:

**1.1 Ürün Yönetimi Modülü (Product Management)**

• Ürünlerin eklenmesi, düzenlenmesi ve silinmesi gibi işlemleri yöneten bir sistem.

• Kategorilere göre ürün filtreleme, ürün arama özellikleri.

• Veritabanı tasarımı: Ürün bilgileri, stok durumu, fiyatlandırma ve indirimler.

**Tasarım Desenleri**:

• **Factory Pattern**: Farklı türde ürünler (örneğin elektronik, kıyafet, aksesuar) için fabrika sınıfları.

• **Repository Pattern**: Ürün verilerinin veritabanından erişimini soyutlamak için kullanılabilir.

**1.2 Kullanıcı Yönetimi Modülü (User Management)**

• Kullanıcı kaydı, giriş yapma ve kullanıcı profillerini yönetme.

• Kullanıcı yetkilendirme ve güvenlik politikalarının uygulanması.

**Tasarım Desenleri**:

• **Singleton Pattern**: Oturum yönetimi için kullanılabilir (her kullanıcı için tek bir oturum örneği).

• **Facade Pattern**: Kullanıcı kayıt ve giriş süreçlerini basitleştirmek için bir dış yüz kullanılabilir.

**1.3 Sipariş ve Sepet Yönetimi Modülü (Order & Cart Management)**

• Kullanıcıların ürünleri sepete ekleyip sipariş oluşturma işlemleri.

• Sipariş takip sistemi, kullanıcıya siparişin durumunu göstermek.

**Tasarım Desenleri**:

• **Observer Pattern**: Sipariş oluşturulduğunda stok ve ödeme sistemleri bilgilendirilir.

• **State Pattern**: Siparişin farklı durumlarını (işlemde, hazır, teslim edildi) yönetmek için kullanılabilir.

**1.4 Ödeme ve Faturalandırma Modülü (Payment & Billing)**

• Ödeme işlemleri, farklı ödeme yöntemleri (kredi kartı, banka havalesi vb.) ile entegre olma.

• Fatura oluşturma ve geçmiş siparişlerin saklanması.

**Tasarım Desenleri**:

• **Adapter Pattern**: Farklı ödeme sistemlerine (örneğin Stripe, PayPal) bağlanmak için kullanılabilir.

• **Strategy Pattern**: Farklı ödeme yöntemlerini yönetmek için uygulanabilir.

**1.5 Güvenlik Modülü**

• Kullanıcıların verilerini güvenli bir şekilde yönetmek.

• Veritabanı şifreleme, kullanıcı oturumları ve yetkilendirme süreçleri.

**Tasarım Desenleri**:

• **Decorator Pattern**: Kullanıcı bilgilerinin şifrelenmesi veya token doğrulaması gibi işlemleri eklemek için.

• **Singleton Pattern**: Güvenlik politikaları ve oturum yönetimi için kullanılabilir.

**2. Teknik Mimariler**

**2.1 Backend (API ve Veritabanı Yönetimi)**

• **Teknoloji Önerileri**:

• Flask veya Django REST Framework kullanarak API’ler geliştirebilirsiniz.

• Veritabanı olarak PostgreSQL veya MySQL tercih edilebilir.

**Görevler**:

• API tasarımı ve geliştirme.

• Veritabanı modelleme ve ilişkisel tablo yapıları oluşturma.

• Ürün yönetimi, sipariş yönetimi ve kullanıcı işlemleri API’lerinin geliştirilmesi.

**Tasarım Desenleri**:

• **MVC (Model-View-Controller)**: İş mantığını, veriyi ve görünümü ayırarak kodun daha düzenli olmasını sağlar.

• **Repository Pattern**: Veritabanı erişimini soyutlayarak veri yönetimi ve iş mantığını ayırır.

**2.2 Frontend (Kullanıcı Arayüzü)**

• **Teknoloji Önerileri**:

• React veya Vue.js ile modern, kullanıcı dostu bir arayüz geliştirebilirsiniz.

**Görevler**:

• Ürün listeleme sayfası, ürün detayları ve sepet işlemleri için UI geliştirme.

• Kullanıcı kayıt ve giriş işlemlerini sağlayan arayüzler oluşturma.

• Sipariş ve ödeme adımlarını yönetme.

**Tasarım Desenleri**:

• **MVC**: Kullanıcı arayüzü ve backend iletişimini düzenli şekilde yapılandırmak için.

**2.3 Ödeme Sistemleri ve Güvenlik**

• **Teknoloji Önerileri**:

• Stripe, PayPal gibi ödeme sağlayıcılarla entegrasyon.

• OAuth gibi yetkilendirme sistemleri.

**Görevler**:

• Ödeme işlemleri entegrasyonu.

• Kullanıcı kimlik doğrulama ve oturum yönetimi.

• Kullanıcı verilerinin güvenliğini sağlamak için şifreleme ve token doğrulama.

**2.4 Test ve DevOps**

• **Teknoloji Önerileri**:

• Otomatik testler için PyTest.

• CI/CD için Jenkins, GitLab CI.

**Görevler**:

• Her modül için otomatik testler yazmak.

• Sürekli entegrasyon (CI) ve dağıtım (CD) süreçlerini kurmak.

• Uygulamanın ölçeklenebilirliğini ve performansını izlemek için monitoring araçları kurmak.

**3. Görev Dağılımı**

Ekipteki kişilerin ilgisi ve becerilerine göre şu şekilde bir dağılım yapabilirsiniz:

• **Kişi 1: Backend ve Veritabanı Yönetimi**

• API geliştirme, veritabanı yönetimi, ürün ve sipariş modüllerini tasarlama.

• **Kişi 2: Frontend Geliştirme**

• Kullanıcı arayüzü geliştirme, ürün listeleme ve kullanıcı etkileşimi üzerine odaklanma.

• **Kişi 3: Ödeme Sistemleri ve Güvenlik**

• Ödeme entegrasyonları, oturum ve kimlik doğrulama sistemleri üzerine çalışma.

• **Kişi 4: Test ve DevOps**

• Testlerin yazılması, CI/CD süreçlerinin yönetimi ve uygulamanın dağıtımı.

**4. 4 Aylık Süreç İçin Çizelge**

Projeyi 4 aylık sürede yönetmek için bir yol haritası önerisi:

• **1. Ay**:

• Proje gereksinimlerinin belirlenmesi.

• Veritabanı tasarımı ve temel API’lerin oluşturulması.

• Ön uç (frontend) geliştirmeye başlanması.

• **2. Ay**:

• Ürün ve kullanıcı yönetimi modüllerinin tamamlanması.

• Kullanıcı oturumu ve güvenlik özelliklerinin entegrasyonu.

• **3. Ay**:

• Sipariş ve ödeme modüllerinin geliştirilmesi.

• Otomatik testlerin yazılması.

• CI/CD sürecinin kurulması.

• **4. Ay**:

• Hataların düzeltilmesi, testlerin tamamlanması.

• Performans ve güvenlik testleri.

• Proje dağıtımı ve son optimizasyonlar.

Bu şekilde projeyi daha detaylı ve planlı bir şekilde ilerletebilir, her ekip üyesinin görevlerini netleştirebilirsiniz.