**1. Aşama: Temel React ve JavaScript Bilgisi (1-4. Hafta)**

**Hafta 1-2: JavaScript Temelleri**

**Öğrenmeniz Gerekenler:**

1. **ES6+ JavaScript Özellikleri:**
   * **Arrow Functions**: Fonksiyon yazımında kısa yol kullanımı.
   * **Destructuring**: Nesnelerden ve dizilerden veri çıkarma.
   * **Template Literals**: String interpolasyonu.
   * **Spread Operator**: Nesne ve dizileri kopyalamak için kullanılır.
   * **Rest Operator**: Parametrelerin bir dizi olarak toplanması.
   * **Async/Await**: Asenkron işlemler için daha temiz bir yapı.
2. **Array Methods:**
   * **map(), filter(), reduce(), forEach(), find()** gibi metodlar.

**Projeler:**

1. **To-Do List Uygulaması:**
   * JavaScript ES6 özelliklerini kullanarak temel bir todo listesi oluşturun.
   * Arrow Functions, Destructuring gibi özellikleri uygulama.
   * map() ve filter() kullanarak görevleri listeleyin ve filtreleyin.

**Mentorluk Notları:**

* **ES6 Kullanımı:** Her yeni özellik öğrenildiğinde, eski yöntemleri gözden geçirin ve yeni yöntemlerin avantajlarını anlayın.
* **Asenkron JavaScript:** async/await kullanarak asenkron işlerde hata yönetimini doğru yapın.

**Hafta 3-4: React Temelleri**

**Öğrenmeniz Gerekenler:**

1. **React Temelleri:**
   * **JSX ve Components:** React bileşenleri, props ve state.
   * **State ve Props Yönetimi:** State ve props’un farkları, state yönetimi.
   * **Event Handling:** Kullanıcı etkileşimleri (input değişiklikleri, buton tıklamaları).
   * **Conditional Rendering:** Koşullu render işlemleri.
   * **List Rendering:** Array ve obje veri yapılarını listeleme.
2. **React Hooks:**
   * **useState:** State yönetimi.
   * **useEffect:** Yan etkiler (side effects) ile çalışma.
   * **useRef:** DOM elementlerine doğrudan erişim.

**Projeler:**

1. **To-Do List Uygulaması (React Hooks):**
   * **useState** kullanarak todo listesi verilerini yönetme.
   * **useEffect** kullanarak localStorage’a veri kaydetme ve okuma.
   * **Conditional Rendering:** Görevlerin tamamlanma durumuna göre stil değiştirme.

**Mentorluk Notları:**

* **Props ve State Yönetimi:** Props bileşenler arasında veri taşırken, state ise bileşenin içindeki veriyi yönetir.
* **useEffect Kullanımı:** useEffect ile yan etkileri yönetin. Örneğin, API çağrıları veya DOM manipülasyonları gibi işlemler.

**2. Aşama: Redux ve Global State Yönetimi (5-8. Hafta)**

**Hafta 5-6: Redux Temelleri**

**Öğrenmeniz Gerekenler:**

1. **Redux Nedir?:**
   * **Actions:** Veri taşımak için kullanılan objeler.
   * **Reducers:** State’i değiştiren saf fonksiyonlar.
   * **Store:** Uygulamanın global state’ini tutan yer.
2. **Redux İleri Düzey Konular:**
   * **Reducers ve CombineReducers:** Birden fazla reducer’ı birleştirerek state yönetimi.
   * **Dispatch ve getState:** Redux store’a veri göndermek ve state’e erişmek.
   * **DevTools:** Redux DevTools ile uygulamanızdaki state değişimlerini takip etme.

**Projeler:**

1. **Todo List Uygulaması (Redux ile):**
   * **State Yönetimi:** Redux kullanarak todo listesi verilerini store’a taşıyın.
   * **Actions ve Reducers:** ADD\_TODO, REMOVE\_TODO, TOGGLE\_TODO gibi action türleri oluşturun.
   * **React-Redux Bağlantısı:** useDispatch ve useSelector kullanarak Redux store ile React bileşenlerini bağlayın.

**Mentorluk Notları:**

* **Redux Toolkit:** Redux Toolkit’i kullanarak uygulamanızdaki boilerplate kodları azaltın.
* **Global State:** Her bileşende state yönetmek yerine, merkezi bir store kullanarak uygulamanızın verilerini yönetin.

**Hafta 7-8: React-Redux ve Asenkron Veri Yönetimi**

**Öğrenmeniz Gerekenler:**

1. **React-Redux İleri Düzey:**
   * **Redux Toolkit ile Slicing:** createSlice ve createAsyncThunk kullanarak Redux işlemleri daha verimli hale getirin.
   * **Asenkron Veri Yönetimi:** API istekleri ve verileri Redux’a aktarma.
2. **Redux Thunk veya Redux-Saga:**
   * Asenkron işlemler için middleware kullanımı.

**Projeler:**

1. **Hava Durumu Uygulaması (Redux ile API Veri Yönetimi):**
   * **Redux Thunk ile API Çağrısı:** Hava durumu verisini çekmek için Redux Thunk kullanın.
   * **React-Redux Bağlantısı:** Redux store’dan hava durumu verisini çekin ve UI'yi güncelleyin.

**Mentorluk Notları:**

* **Asenkron İşlemler:** API çağrılarının Redux içinde nasıl yönetileceğini öğrenin.
* **Redux Thunk:** Asenkron işlemler için middleware kullanımı önemli bir beceridir.

**3. Aşama: HTTP İstekleri, URL İşlemleri ve İleri Düzey React Konuları (9-12. Hafta)**

**Hafta 9-10: HTTP İstekleri ve API Kullanımı**

**Öğrenmeniz Gerekenler:**

1. **Axios veya Fetch API ile HTTP İstekleri:**
   * **GET, POST, PUT, DELETE:** HTTP istekleri ile veri çekme ve gönderme.
   * **Error Handling:** Hataları doğru şekilde yakalama.
2. **React ile URL İşlemleri:**
   * **React Router:** URL bazlı yönlendirme yapma.
   * **URL Parametreleri:** URL’deki parametreleri çekme (useParams, useLocation).
   * **Query Strings:** URL query string’lerini okuma (useHistory, useLocation).

**Projeler:**

1. **Blog Uygulaması (Axios + React Router + Redux):**
   * **Axios Kullanımı:** Blog yazılarını çekmek ve oluşturmak için Axios kullanın.
   * **React Router ile Sayfa Geçişleri:** Blog yazılarını görüntülemek için React Router kullanarak dinamik sayfa geçişleri yapın.
   * **URL Parametreleri:** Blog yazılarına URL parametreleri ekleyin.

**Mentorluk Notları:**

* **API Entegrasyonu:** API’den veri çekerken hata yönetimini iyi yapın. try-catch blokları kullanın.
* **React Router ile Sayfa Yönlendirme:** URL yönetimini doğru yaparak kullanıcı deneyimini iyileştirin.

**Hafta 11-12: Backend Geliştirme (Node.js ve MongoDB)**

**Öğrenmeniz Gerekenler:**

1. **Node.js ve Express.js ile Backend Geliştirme:**
   * **RESTful API Tasarımı:** GET, POST, PUT, DELETE işlemleri.
   * **Routing ve Middleware:** Express Router ve middleware kullanımı.
2. **MongoDB ve Mongoose:**
   * **CRUD İşlemleri:** MongoDB’de verileri oluşturma, okuma, güncelleme ve silme işlemleri.
   * **Veritabanı Tasarımı:** Mongoose ile veritabanı şeması ve modelleri oluşturma.

**Projeler:**

1. **Kullanıcı Yönetim API’si (Node.js + MongoDB):**
   * **JWT ile Authentication:** Kullanıcı kaydı ve giriş için JWT tabanlı kimlik doğrulama ekleyin.
   * **CRUD İşlemleri:** Kullanıcıları MongoDB veritabanında yönetme.

**Mentorluk Notları:**

* **JWT ve Authentication:** Backend’de JWT tabanlı kimlik doğrulama işlemi için güvenlik önlemleri alın.
* **Mongoose İleri Seviye:** İleri düzey MongoDB işlemleri için populate() ve aggregation kullanımı.

**4. Aşama: Tamamlayıcı Projeler ve Performans İyileştirme (13-16. Hafta)**

**Hafta 13-14: Performans İyileştirme ve Güvenlik**

**Öğrenmeniz Gerekenler:**

1. **API Performans İyileştirme:**
   * **Caching:** Redis veya başka caching yöntemlerini kullanarak API performansını iyileştirin.
   * **Pagination:** Büyük veri setlerini işlerken sayfalama kullanımı.
2. **Web Güvenliği:**
   * **XSS, CSRF, SQL Injection Koruması:**
     + Güvenlik için input doğrulama.
     + Token tabanlı doğrulama ve güvenli oturum yönetimi.

**Projeler:**

1. **E-Ticaret Uygulaması (Full-Stack Proje):**
   * **React, Redux, Node.js ve MongoDB ile Tamamlanmış Uygulama:** Ürünleri listeleme, ödeme işlemleri, kullanıcı doğrulama.
   * **JWT Authentication ve Authorization:** Kullanıcı yönetimi ve admin paneli.

**Mentorluk Notları:**

* **Performans:** Veritabanı sorguları ve API yanıt sürelerini optimize edin.
* **Güvenlik:** Güvenliği ön planda tutarak kullanıcı verilerini koruyun.