DOLUNAY | Yazılım

Week 5 & 6 | Frontend

Introduction

Geldik projenin sonunaa!

Artık çıktılarımızı görselleştirmeye ve bir web-app haline getirmeye hazır mıyız?

Bu kısımda yapacaklarımız sayesinde kullanıcılarımız database'deki etkinlik verilerini web üzerinden görebilecek.

Belki de birçoğunuzun oldukça yabancı olduğu front-end dünyasına bu bölümde kısa bir giriş yapıyoruz.

Why we have it?

Frontend is the part of a software application or website that users interact with directly. It's the user interface, the visual elements, and the overall design that people see and interact with when using a digital product.

Think of it like this: when you visit a website or use a mobile app, everything you see and interact with on the screen—buttons, menus, images, forms, text, and more—is part of the frontend. It's what allows users to input information, click on buttons, navigate through pages, and experience the content or functionality of the application.

Why do we need frontend?

User Experience: Frontend development is crucial for creating a pleasant and intuitive user experience. It's responsible for the design and layout that make applications easy to use and visually appealing.

Interactivity: Frontend enables interactivity. Users can click on buttons, submit forms, view animations, and experience dynamic content—all made possible by frontend development.

Accessibility: It ensures that the application is accessible to users with different abilities and devices. Good frontend development considers factors like responsive design to make sure the interface works well on various screen sizes.

Branding and Design: Frontend allows developers and designers to bring a brand's visual identity to life. Colors, fonts, logos, and overall aesthetics contribute to the brand's identity and recognition.

Communication with Backend: While frontend deals with the user interface, there is often a need for communication with the backend (server-side) to fetch or send data. Frontend developers use programming languages like JavaScript to facilitate this communication.

Examples of Frontend Elements:

Buttons: The interactive elements users click on to perform actions.

Forms: Input fields, checkboxes, and buttons for users to submit data.

Navigation Menus: The menu bar or navigation links that help users move around the application.

Images and media: The visual elements that enhance the overall design and convey information.

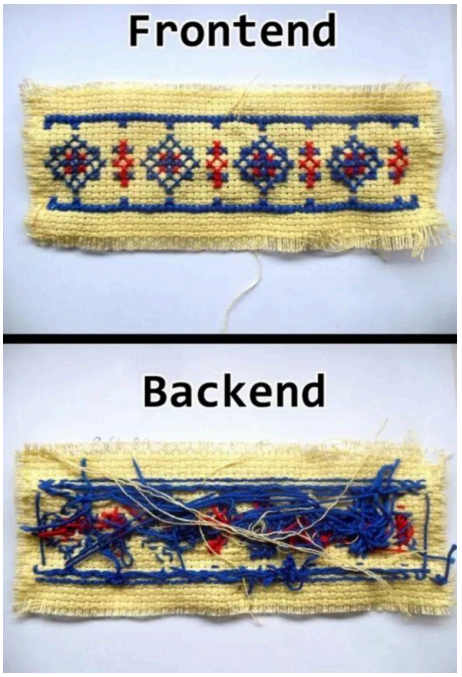
Text and Content: All the written information, articles, and descriptions presented to the users.

Animations and Transitions: Visual effects that make the user interface more engaging.

Özetle,

Front-end, ürünümüzde kullanıcıların gördüğü kısmın geliştirilmesine verilen addır. Frontend ve Backend arasında bir iletişim kurararak kullanıcının neye tıkladığını, şu anda ne yapmak istediğini anlar ve yazılımımızın buna göre çalışmasını sağlarız.

Yani şu anda, siz bu yazıyı okurken, bir frontend yazılımı sonucu ortaya çıkarılmış bir sayfaya bakıyorsunuz.

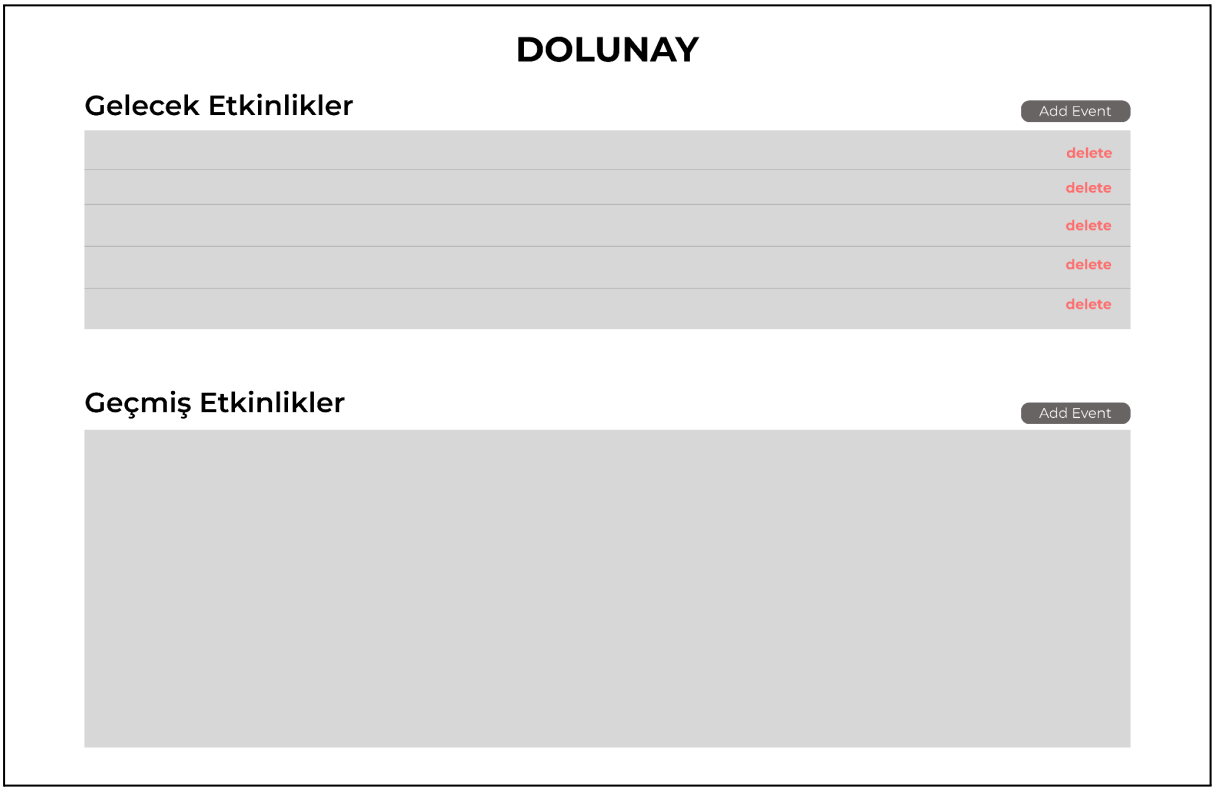


Tech stack

#React.js

Task definition

Objective: Create the following frontend interface for user interaction.



What will you learn? :

- Basics of React and component-based architecture.
- Developing user interfaces for event creation and user profiles.
- Implementing state management for data handling.
- Connecting the frontend with the backend via API calls.
- Basic styling and responsive design practices.

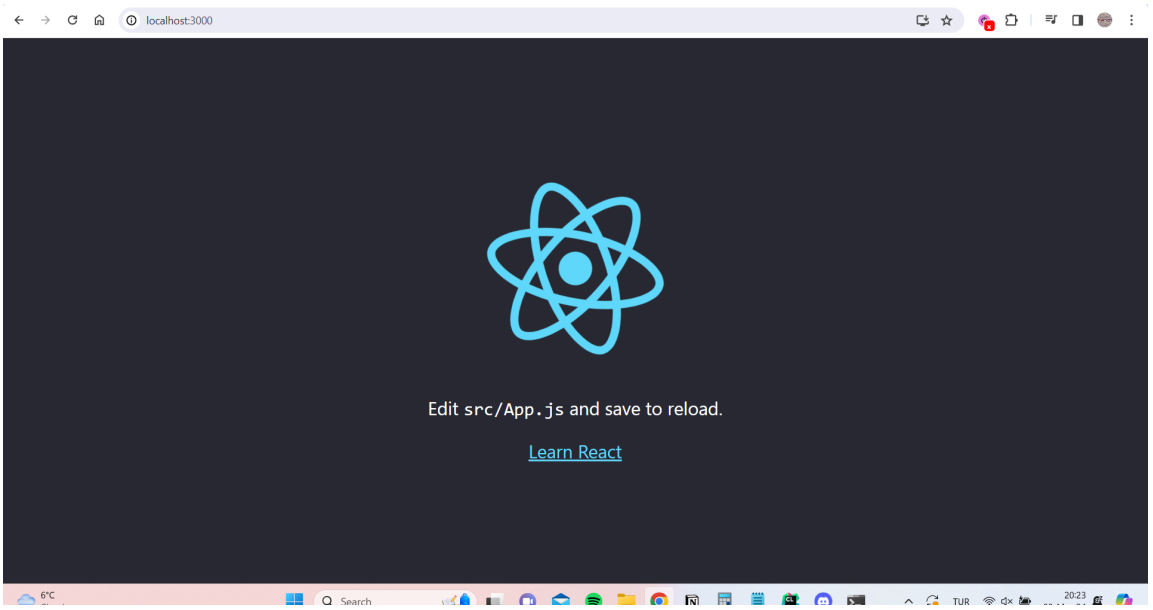
Recommended Tools & Languages:

- Node.js (React'ın çalışması için Node.js'e ihtiyaç var o yüzden ilk olarak Node.js indiriyoruz: https://nodejs.org/en)
- Front-end çalışmalarını yürüteceğimiz bir klasör açıyoruz ve o klasörün içinde terminali açıyoruz (cmd ya da IDE içinden bir terminal açabilirsiniz).
- Buradaki Quick Start başlığı altında yazan 3 komutu sırası ile çalıştırıp React'ı yüklüyor ve ilk React uygulamımızı çalıştırıyoruz: https://create-react-app.dev/docs/getting-started/#quick-start

Steps to-do:

1. **Installation**

   Yukarıda yazılan uygulamaları yükleyelim ve adımları uygulayalım. Eğer her şey doğru çalıştıysa http://localhost:3000/ adresine gittiğinizde şöyle bir ekran sizi bekliyor olacak:



2. **Understanding React Mechanism**

   Eğer oluşturduğumuz Frontend çalışmaları klasörüne gidersek React'ın bizler için bir sürü dosya oluşturduğunu göreceğiz. React.js bu dosyaları kullanarak bir web uygulaması oluşturur.

   src (yani source) klasörüne gidersek şu anda gördüğümüz sayfanın kodlarına ulaşırız. *App* dosyası içinde ana sayfanın kodları bulunuyor. Örneğin *App* dosyasını açarak oradaki "Edit src/App.js and save to reload" yazısını başka bir yazıyla ("DOLUNAY" 🙂) değiştirirsek ve dosyayı kaydedersek localhost:3000'deki web sayfamızın da değiştiğini göreceğiz.

   React.js uçsuz bucaksız bir alan 🙂 Anlamadığımız her noktayı ChatGPT'ye sormaya ve Google'lamaya devam.

   Bu frontend çalışmalarımız için zorunlu değil ama React'a ayrıntılı bir giriş yapmak isterseniz bu videoyu izleyebilirsiniz:
   https://youtu.be/SqcY0GlETPk?si=NrCrVfxR2RshC5-V

3. **Import UI Design Library**

   Web sayfalarında sürekli tekrar eden bazı component'lar vardır. Örneğin; butonlar, formlar, menüler, tablolar vb.

   İşte, bu component'ların kolay kullanımı ve her seferinde tekrardan baştan tasarımının/kodlamasının yapılmaması için çeşitli *UI Design Library*'ler üretilmiş.

Her bir UI Design Library'nin kendine has bir tasarımı var ve bu library'lerin dokümantasyonlarını inceleyerek birkaç satır kodla, örneğin, bir tablo oluşturabiliyoruz.

Bu kısımda aşağıda verdiğimiz UI Design Library'leri yüklemek ya da kullanmak zorunda değilsiniz, burası tamamen size kalmış. Aşağıdaki UI library'leri kullanarak ya da React'ın çeşitli library'lerini kullanarak sitemizi tasarlayabiliriz.

Bizce en estetik component'lara sahip library, Tailwind:

https://tailwindcss.com/docs/guides/create-react-app

Google'ın meşhur UI Library'si, Material UI:

https://mui.com/material-ui/getting-started/installation/

Bir diğer çok yaygın UI Library, Bootstrap:

https://getbootstrap.com/

Ve daha niceleri…

Biz bu projede özellikle tablo ve buton kullanacağız. Sizler de tablo ve buton componentlarına bakarak gözünüze hoş gözüken bir library'i seçebilirsiniz. Ya da seçmeyebilirsiniz ve React'ın kendisinde bulunanları otomatik kullanabilirsiniz 🙂

4. **Create Table (Gelecek Etkinlikler)**

   Bu adımda

   - Gelecek etkinlikleri gösteren bir tablo oluşturuyoruz. (Bu noktada yukarıdaki örnek UI library'lerden seçtiğiniz UI Library'i kullanarak table nasıl oluşturulduğunu öğrenebilirsiniz ya da react'ın kendi table oluşturma mekanizmasını kullanabilirsiniz)
   - Frontend'de Backend'de işlemlerini hallettiğimiz tabloları göstereceğiz. Bu yüzden bu iki yüzün birbirleriyle haberleşmesi lazım. Bunun için de öncelikle CORS mekanizmasını ayarlıyoruz. CORS'un ne olduğunu bu linkten öğrenebiliriz: https://fastapi.tiangolo.com/tutorial/cors/

     Backend tarafında yazdığımız main.py dosyasına aşağıdaki kodu eklememiz durumunda ayarları doğru bir şekilde yapmış olacağız:

```
from fastapi.middleware.cors import CORSMiddleware
...
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

- Gelecek etkinliklerin gösterileceği tabloda backend kısmında yazdığımız **get_upcoming_events** API'ını kullanıyoruz.

  React ve FastAPI'ı birbirine bağlamanın birkaç yolu var. Burada bütün yollar listelenmiş: https://www.freecodecamp.org/news/how-to-fetch-api-data-in-react/ Sektör tecrübelerimizden gördüğümüz kadarıyla en sık kullanılan yöntem axios. Bu yüzden axios kullanarak frontend ve backend bağlamayı öneririz. Örnek bu kodda events adlı endpoint den gelen bütün veriler frontend de data adında bir variable'a atanmış:

  ```
  const { data } = await axios.get(`${API_URL}/events`);
  ```

- Tablonun başına bir add_event butonu ekleyip bunu backend'de yazdığımız **add_event** API ile bağlıyoruz. Add event butonuna eğer admin rolüne sahip bir kullanıcı basarsa etkinlik bilgilerinin girilebileceği bir formu barındıran pop-up ekran açılmalı. Bu aşamada ise bir post işlemi yapacağız. Yazacağınız kod aşağı yukarı buna benzeyecek fakat aşağıdaki kod çalışır durumda değil tabii ki.

```
const authToken = 'your_actual_authentication_token';  // Retrieve from
where it's stored
const response = await axios.post(
  'http://localhost:8000/api/add_event',
  {<your data>},
  {
    headers: {
      Authorization: `Bearer ${authToken}`,
    },
  }
);
```

- Her bir etkinlik verisinin yanına delete_event butonu ekleyip bunu backend'de yazdığımız **delete_event** API ile bağlıyoruz. Kullanıcı admin rolünde ise delete event butonuna bastığında "are you sure?" diye soran bir pop-up belirmeli ve kullanıcı "evet" derse silinme işlemi yapılmalı.
- Projenin bu noktasında giderek spesifikleştiğimizden artık direkt olarak yapmak istediklerimizi kopyala yapıştır yaparak uygulayamıyoruz. FastAPI ile React nasıl bağlanıyor, tablo nasıl oluşturuluyor, buton nasıl oluşturuluyor, butonu yetkisi olmayanlara görünmez nasıl yapabilirim ya da butona yetkisi olmayanlar basarsa nasıl hiçbir işlem yaptırmam gibi soruları Google'layarak ya da ChatGPT ile konuşarak halletmemiz gerekiyor. Aşağıda bu çalışmaları yaparken size yardımcı olabilecek genel kaynakları listeledik.

  Kaynaklar:

  Bu videoda FastAPI & React kullanılarak bir table oluşturulup, butonlar eklenip, o table üzerinde çeşitli düzenlemeler yapılmasına olanak sağlanan bir kod üretiliyor: https://www.youtube.com/watch?v=Mxh67Vbibqk

Bu yazıda FastAPI & React ve HarperDB adında bir database kullanılarak table oluşturulup bu table'da çeşitli düzenlemeler yapılıyor: https://medium.com/@dennisivy/fast-api-react-crud-app-with-harperdb-5834af537c23

React üzerinden data çekmenin yolları üzerine bir yazı: https://www.freecodecamp.org/news/how-to-fetch-api-data-in-react/

Burada daha kapsamlı bir FastAPI & React uygulaması inşa ediliyor ve detaylıca anlatılıyor:

https://thepythoncode.com/article/fullstack-notes-app-with-fastapi-and-reactjs

5. **Create Table 2 (Geçmiş Etkinlikler)**

Bu adımda

- Geçmiş etkinlikleri gösteren bir tablo oluşturuyoruz.
- Bu tabloda backend kısmında yazdığımız **get_past_events** API'ını kullanıyoruz.
- Tablonun başına bir add_event butonu ekleyip bunu backend'de yazdığımız **add_event** API ile bağlıyoruz.
- Her bir etkinlik verisinin yanına delete_event butonu ekleyip bunu backend'de yazdığımız **delete_event** API ile bağlıyoruz.

🎉 Üçüncü bölümün sonuna geldikkk.
⭐ Soruların için her zaman discord'dayız, kocaman bir komünite seni bekliyor.