Cüzdan Uygulaması

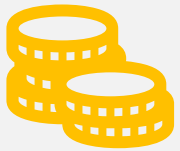
İbrahim Vahit AÇAR

1191602054

Elektronik Cüzdanım Uygulamasının Amacı

Online olarak bütçe takip imkanı.

Eski harcama ve gelirlerinizi tarihiyle beraber listeleyip kullanıcıya gösterebilmek.



Son durumdaki harcama ve gelir durumunu gösterebilmek.

Giriş Sayfamız

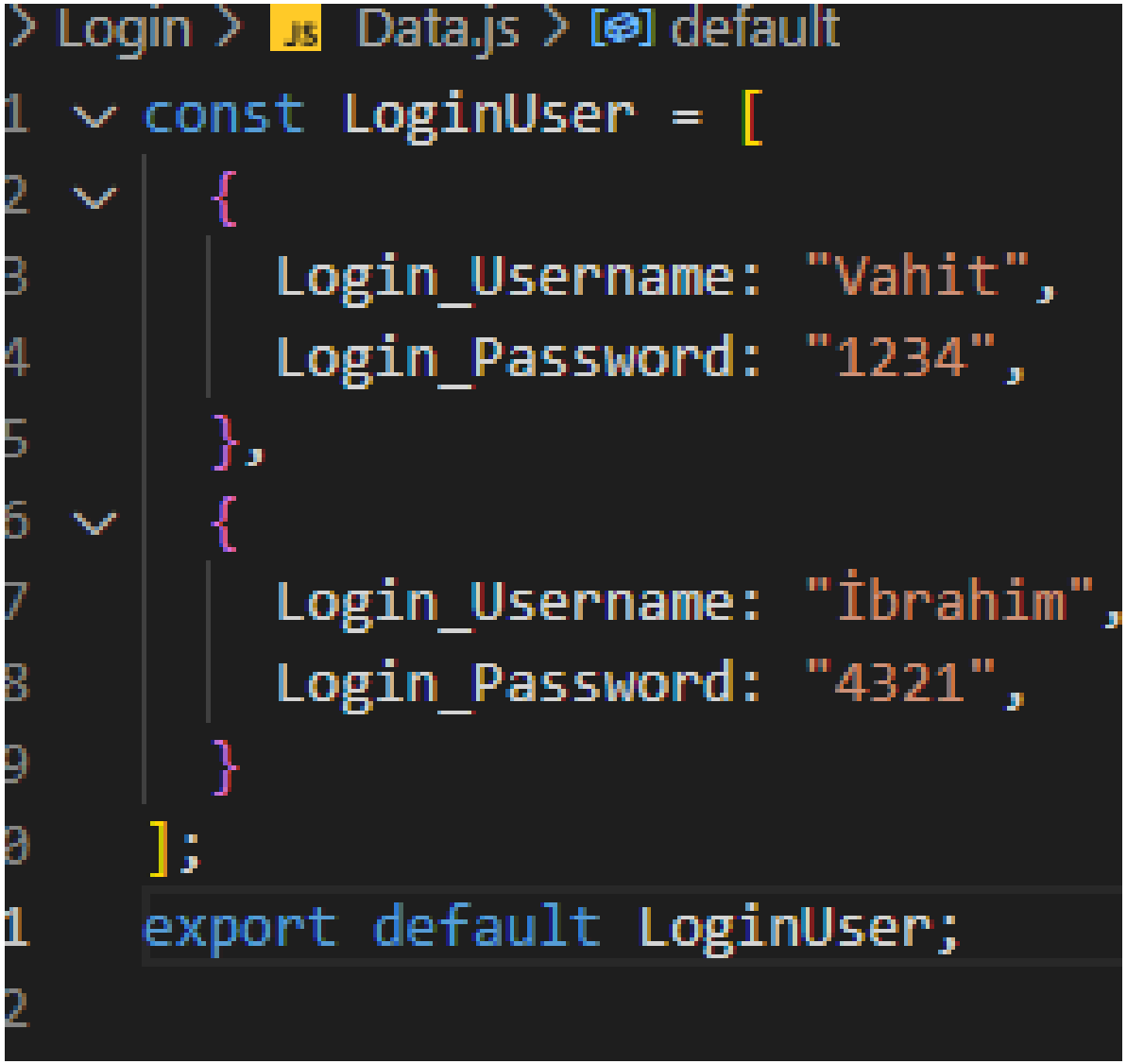
- Elektronik Cüzdanım uygulamamızın giriş arayüzüdür.

```
return (
  <div >
    <div className="login-full-page">
      <h2>Elektronik Cüzdanım</h2>
      <form className="form">
        <h1 className="log-title">Giriş Sayfası</h1>
        <div className="first-page">
          <label className="first-login-user"> Kullanıcı Adı</label>
          <input
            className="Text_css"
            type="text"
            onChange={(e) => setLoginUser(e.target.value)}
            autoComplete="off"
            maxLength={12}
            required
          ></input>
          <div className="space"></div>
          <label className="first-login-user"> Şifre </label>
          <input
            className="Text_css"
            type="password"
            onChange={(e) => setLoginPassword(e.target.value)}
            autoComplete="off"
            required
          ></input>
        </div>

        <div className="second-page">
          <button className="log-button" onClick={log_click}>
            Giriş
          </button>
        </div>
      </form>
```

Giriş Sayfamız

- Kullanıcı adında text tipinde ve şifre adında password tipinde veri alan inputlar belirledik.
- Giriş adında bizi doğru olduğunda /home adresine göndericek button belirledik.
- Görünüş içinde hepsinin classNamelerini adlandırıp css'lerini belirledik.



Giriş Datamız

- Giriş yapabilmek için kullanıcıların sahip olması gereken Datayı tutmasıiçin LoginUser'ı tanımladık ve export ettik.

```
import "../Login/Login.css";
const Login = () => {
  const Alert = () => {
    alert("Kullanıcı adı veya Şifre yanlış");
  };
  const navigate = useNavigate();
  const [loginUser, setLoginUser] = useState();
  const [loginPassword, setLoginPassword] = useState
  const log_click = (e) => {
    e.preventDefault();
    var user = LoginUser.find(
      (admin) =>
        admin.Login_Username === loginUser &&
        admin.Login_Password === loginPassword
    );
    if (user) {
      navigate("/home");
    } else {
      Alert();
    }
  };
```

Girilen Bilgileri Doğrulama

- UseState fonksiyonunu kullanarak kullanıcı adı ve şifreye yazdığımız datayı setLoginUser ve setLoginPassworda atadık.
- LoginUser.find ile girilen değerleri datamızla karşılaştırdık.
- Yanlışsa hata varmesi için Alert()'e doğruysa /home'a gitmesi için if else belirledik.

NavTab ve Sayfa geçişleri

Sayfanın üst kısmında geçebileceğimiz sayfalar ve adresleri.

```
  return (
    <div className="full-navbar">
      <div className="first-navbar">
        <img
          className="flat"
          src="https://avatars.mds.yandex.net/i?id=5854c5247cc767a215d24ab53a31e6b7-4571053-images-thumbs&n=13"
          alt=""
        ></img>
        <ul className="ul-css">
          <li onClick={Home} className={index === "0" ? "btn active" : "btn"}>
            Cüzdan
          </li>
          <li onClick={About} className={index === "1" ? "btn active" : "btn"}>
            Uygulama Hakkında
          </li>
        </ul>
      </div>
      <div className="second-navbar">
        <li onClick={quit} className="quit-btn">
          Çıkış
        </li>
      </div>
    </div>
  );
};

export default NavTab;
```

```
import { useNavigate } from "react-router-dom";
const NavTab = (props) => {
  const { index } = props;
  const navigate = useNavigate();
  const quit = () => {
    navigate("/login");
  };
  const Home = () => {
    navigate("/home");
  };
  const About = () => {
    navigate("/about");
  };
  return (
```

```jsx
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Home from "./Home/Home";
import Login from "./Login/Login";
import About from "./About/About";
const App = () => {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Login />} />
        <Route path="/login" element={<Login />} />
        <Route path="/home" element={<Home />} />
        <Route path="/about" element={<About/>} />
      </Routes>
    </BrowserRouter>
  );
};

export default App;
```

App.js Routes

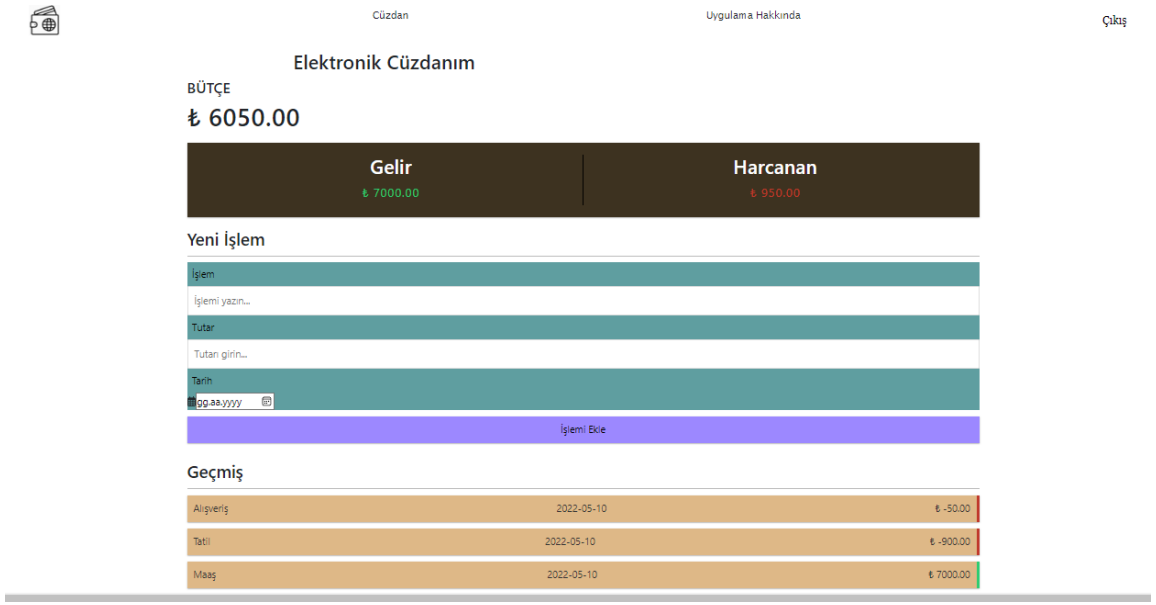
- Hangi adresin hangi sayfaya ait olduğunu belirledik

/home Sayfamız

src > Home > Home.js > Home

```jsx
import React from "react";
import NavTab from "../NavTab/NavTab";
import "./Home.css";
import BudgetApp from "../BudgetApp";
function Home() {
  return (
    <div className="home">
      <>
        <NavTab index="0"></NavTab>
        <hr></hr>
      </>
      <div>
        <BudgetApp />
      </div>
    </div>
  );
}

export default Home;
```



AppContext.js

- Aynı veriyi çok sayıda componentte kullanmamız gerektiği için Context Apı kullandık.Bunun için payşalabilir bir state oluşturmamız lazım.
- Paylaşılabilir bir state oluşturmak için React.createContext() metodunu kullanırız.
- AppContext Provider oluşturduğumuz contexti kullanacak componentlerin erişebilmesini sağlayan ,context güncellemelerinde bu componentlerin yeniden render olmasını sağlar.
- Value isminde bir prop alır ve bu prop ile context değerlerini child componentlere aktarılır.

```
import React, { createContext, useReducer } from 'react';
import AppReducer from './AppReducer';
const initialState = {
  transactions: [
    { id: Math.floor(Math.random() * 100000000 ), text: "Alışveriş",date:'2022-05
    { id: Math.floor(Math.random() * 100000000 ), text: "Tatil",date:'2022-05-10'
    {id: Math.floor(Math.random() * 100000000 ), text: "Maaş",date:'2022-05-10',
  ]
}
export const AppContext = createContext(initialState);
export const AppProvider = ({ children }) => {
  const [state, dispatch] = useReducer(AppReducer, initialState);
  function deleteTransaction(id) {
    dispatch({
      type: 'DELETE_TRANSACTION',
      payload: id
    });
  }
  function addTransaction(transaction) {
    dispatch({
      type: 'ADD_TRANSACTION',
      payload: transaction
    });
  }
  return (<AppContext.Provider value={{
    transactions: state.transactions,
    deleteTransaction,
    addTransaction
  }}>
    {children}
  </AppContext.Provider>);
}
```

AppReducer.js

useReducer dispatch ve state bilgisini alır.

dispatch fonksiyonu gönderdiğimiz “string” komutunu ilgili reducer (state, action) olarak iletilmesinden sorumludur

Sonrasında bileşenimiz tekrardan render edilir.

```
}
export const AppContext = createContext(initialState);
export const AppProvider = ({ children }) => {
  const [state, dispatch] = useReducer(AppReducer, initialState);
```

src > context > AppReducer.js > ...

```
1  const AppReducer =(state, action) => {
2      switch(action.type) {
3        case 'DELETE_TRANSACTION':
4          return {
5            ...state,
6            transactions: state.transactions.filter(transaction => transaction.id !== action.payload)
7          }
8        case 'ADD_TRANSACTION':
9          return {
10           ...state,
11           transactions: [action.payload, ...state.transactions]
12         }
13       default:
14         return state;
15     }
16   }
17   export default AppReducer;
18
```

BudgetApp.js

- Uygulamamızın componentlerini buraya çağırıp tek sayfalık uygulamamızın gösterilmesi gereken sırayla yerleştiririz.
- AppProvideri import edip componetleri çevreleriz.

src > BudgetApp.js > BudgetApp

```
1   import React from 'react';
2   import { Header } from './components/Header';
3   import { Balance } from './components/Balance';
4   import { IncomeExpenses } from './components/IncomeExpenses';
5   import { TransactionList } from './components/TransactionList
6   import { AddTransaction } from './components/AddTransaction';
7   import { AppProvider } from './context/AppContext';
8   import './App.css';
9
10  function BudgetApp() {
11    return (
12      <AppProvider>
13        < Header/>
14        <div className="container">
15          <Balance />
16          <IncomeExpenses />
17          <AddTransaction />
18          <TransactionList />
19
20        </div>
21      </AppProvider>
22    );
23  }
24  export default BudgetApp;
25
```

Balance.js

- useContext kullanarak dataya ulaşıp amounts.reduce ile değerlerin toplamını elde ederiz.
- moneyFormatter ile ne türde yazıcağımızı belirleriz.

src > components > Balance.js > Balance

```
1   import React, { useContext } from "react";
2   import { AppContext } from "../context/AppContext";
3
4   function moneyFormatter(props) {
5     let p = props.toFixed(2).split(".");
6     return "₺ " + p[0] + "." + p[1];
7   }
8   export const Balance = () => {
9     const { transactions } = useContext(AppContext);
10    const amounts = transactions.map((transaction) => transaction.amount);
11    const total = amounts.reduce((acc, item) => (acc += item));
12    return (
13      <>
14        <h4>Bütçe</h4>
15        <h1>{moneyFormatter(total)}</h1>
16      </>
17    );
18  };
19
```

IncomeExpenses.js

- transaction.amountdaki değerlerin ‘-’ veya ‘+’ olmasına göre gelir ve harcanan değerler belilenir.
- MoneyFormatter sayesinde düzgün bir şekilde gösterilir.

src > components > IncomeExpenses.js > IncomeExpenses

```
1  import React, { useContext } from "react";
2  import { AppContext } from "../context/AppContext";
3
4  function moneyFormatter(num) {
5    let p = num.toFixed(2).split(".");
6    return "₺ " + p[0] + "." + p[1];
7  }
8  export const IncomeExpenses = () => {
9    const { transactions } = useContext(AppContext);
10   const amounts = transactions.map((transaction) => transaction.amount);
11   const income = amounts.filter((item) => item > 0)
12     .reduce((acc, item) => (acc += item), 0);
13   const expense =
14     amounts.filter((item) => item < 0).reduce((acc, item) => (acc += item), 0) *
15     -1;
16   return (
17     <div className="inc-exp-container">
18       <div>
19         <h2>Gelir</h2>
20         <p className="money plus">{moneyFormatter(income)}</p>
21       </div>
22       <div>
23         <h2>Harcanan</h2>
24         <p className="money minus">{moneyFormatter(expense)}</p>
25       </div>
26     </div>
27   );
28 };
29
```

AddTransaction.js

- Yeni işlem eklemek için kullanırız.
- Aldığımız yeni değerleri useState kullanarak atarız.
- Yeni işlemin değerleri ana datamıza eklenmesi için addTransaction(newTransaction); yazılır.Buda appReducerdaki ‘Add Transaction’ durumunu çalıştırır.

AddTransaction.js

```jsx
export const AddTransaction = () => {
  const [text, setText] = useState("");
  const [amount, setAmount] = useState();
  const [date, setDate] = useState();
  const { addTransaction } = useContext(AppContext);
  const onSubmit = (e) => {
    e.preventDefault();
    const newTransaction = {
      id: Math.floor(Math.random() * 100000000),
      text,
      amount: +amount,
      date,
    };
    setDate("");
    setText("");
    setAmount("");
    addTransaction(newTransaction);
  };

  return (
    <>
      <h3>Yeni İşlem</h3>
      <form onSubmit={onSubmit}>
        <div className="new-transaction" >
          <label className="new">İşlem</label>
          <input
            type="text"
            value={text}
            onChange={(e) => setText(e.target.value)}
            placeholder="İşlemi yazın..."
          />
        </div>
        <div className="new-transaction">
          <label className="new">
            Tutar <br />
          </label>
          <input
            type="number"
            value={amount}
            onChange={(e) => setAmount(e.target.value)}
            placeholder="Tutarı girin..."
          />
        </div>
        <div className="new-transaction">
          <label className="new">
            Tarih <br />
          </label>
          <div>
            <FontAwesomeIcon icon={faCalendarDays} />
            <input
            type="date"
            value={date}
            onChange= {(e) => setDate(e.target.value) }
            />
          </div>
        </div>
        <button className="new-transaction-btn">İşlemi Ekle</button>
      </form>
    </>
  );
};
```

Transaction.js

- İşlemlerin yazılma şeklini belirledik.
- deleteTransactiona butona tanımlayıp çalıştığında transaction.id'yi 'Delete Transaction' durumuna gönderir.

src > components > Transaction.js > Transaction

```
1   import React, { useContext } from "react";
2   import { AppContext } from '../context/AppContext';
3   function moneyFormatter(num) {
4     let p = num.toFixed(2).split(".");
5     return "₺ " + p[0] + "." + p[1];
6   }
7
8   export const Transaction = ({ transaction }) => {
9     const { deleteTransaction } = useContext(AppContext);
10
11    return (
12      <li className={transaction.amount < 0 ? "minus" : "plus"}>
13        {transaction.text}
14        <span>
15          {transaction.date}
16        </span>
17        <span>
18          {moneyFormatter(transaction.amount)}
19        </span>
20        <button
21          onClick={() => deleteTransaction(transaction.id)}
22          className="delete-btn"
23        >
24          x
25        </button>
26      </li>
27    );
28  };
```

TransactionList.js

İşlemleri listeleme

src > components > TransactionList.js > TransactionList

```
1   import React, { useContext } from 'react';
2   import { Transaction } from './Transaction';
3   import { AppContext } from '../context/AppContext';
4   export const TransactionList = () => {
5     const { transactions } = useContext(AppContext);
6     return (
7       <>
8         <h3>Geçmiş</h3>
9         <ul className="list">
10          {transactions.map(transaction => (<Transaction key={transaction.id} transaction={transaction} />))}
11        </ul>
12      </>
13    )
14  }
```