Gyulai SZC Szigeti Endre Technikum és Szakképző Iskola

**Vizsgaremek**

**Autós weboldal**

**Car Shop**

|  |  |  |
| --- | --- | --- |
| Kolozsvári Krisztina Janka |  | Pap László |
| 13/A |  | 13/A |

5520 Szeghalom, Ady Endre utca 3.  
2025.

***Tartalomjegyzék***

[**Bevezetés** 3](#_Toc195207092)

[**Fejlesztői dokumentáció** 4](#_Toc195207093)

[**Tesztdokumentáció** 24](#_Toc195207094)

[**Felhasználói dokumentáció** 33](#_Toc195207095)

[**Összefoglalás** 36](#_Toc195207096)

[**Irodalomjegyzék** 37](#_Toc195207097)

# **Bevezetés**

Az interneten található autós webshopok száma folyamatosan növekszik, és egyre több BMW-tulajdonos választja az online vásárlást, ha minőségi alkatrészekre, kiegészítőkre vagy tuning kiegészítőkre van szüksége. Egy BMW-specifikus webshop célja, hogy segítsen a vásárlóknak a megfelelő termékek kiválasztásában, legyen szó akár gyári alkatrészekről, akár prémium kiegészítőkről vagy egyedi tuningmegoldásokról. Ennek az oldalnak a dokumentációja bemutatja, hogyan lett kialakítva és fejlesztve egy olyan BMW autós webshop, amely kizárólag magas minőségű termékeket és hasznos tájékoztatásokat kínál a BMW-rajongók és autótulajdonosok számára.

# **Fejlesztői dokumentáció**

A hardverünk egy Lenovo ThinkCentre M75s második generációs személyi számítógép, amelyen végeztük a munkánkat.

Specifikációk:

* Processzor – AMD Ryzen 3 PRO 5350G with Radeon Graphics 4.00 GHz
* Memória - 8 GB
* Rendszer típusa - 64 bites operációs rendszer, x64-alapú processzor
* Operációs rendszer: Microsoft Windows 11 Pro N

Fejlesztői környezet:

A frontend illetve a backend programozást is a Visual Studio Code alkalmazás segítségével készítettük el.

A **Visual Studio Code (VS Code)** egy ingyenes, nyílt forráskódú, könnyű és rendkívül népszerű kódszerkesztő, amelyet a **Microsoft** fejlesztett. Főként programozók számára készült, és támogatja a legnépszerűbb programozási nyelveket, például a JavaScript-et, Python-t, C++, Java-t, és sok más nyelvet.

Főbb jellemzők:

1. **Könnyű és gyors**: A VS Code alacsony rendszerigénnyel rendelkezik, gyorsan indul és zökkenőmentes működést biztosít.
2. **Bővíthetőség**: Támogatja a kiterjesztéseket (extensions), amelyekkel új funkciók, témák, nyelvek és debuggerek adhatók hozzá.
3. **Beépített terminál**: Az integrált terminál lehetővé teszi a parancsok közvetlen futtatását a szerkesztőből.
4. **Git integráció**: Kiváló verziókezelési támogatás, közvetlenül a szerkesztőből végezhető Git műveletek.
5. **Intelligens kódsegédlet**: Automatikus kódkiegészítések, hibajelzések és dokumentációs javaslatok.
6. **Többplatformos**: Elérhető Windows, macOS és Linux rendszereken is.

A VS Code a fejlesztők számára ideális választás, mivel gyors, testreszabható és folyamatosan fejlődik a közösség és a Microsoft által egyaránt.

**Futtatási környezet:**

**Laravel:**

A weboldal a Laravel 11-es (vagy legfrissebb) verziójában készült. A legújabb verzió használatával biztosítottuk a modern, hatékony és biztonságos fejlesztési környezetet.

A Laravel egy nyílt forráskódú PHP keretrendszer, amelyet a webalkalmazások fejlesztésének megkönnyítésére terveztek. Az MVC (Model-View-Controller) architektúrán alapul, ami elősegíti a kód jól strukturált és könnyen karbantartható felépítését. Laravel modern megközelítései, mint az Eloquent ORM, a beépített autentikációs rendszerek, a queue kezelés, az ütemezett feladatok és az erőteljes migrációs eszközök, lehetővé teszik a gyors és rugalmas fejlesztést.

Főbb jellemzők:

1. **Elegáns szintaxis és tiszta kódszerkezet:** A Laravel kódja olvasható és karbantartható, támogatja a tiszta architektúrákat.
2. **Fejlett adatbázis-kezelés:** Az Eloquent ORM segítségével egyszerűen és hatékonyan kezelhetjük az adatbázis műveleteket.
3. **Biztonság:** Beépített védelmi mechanizmusok, például CSRF tokenek, jelszó hash-elés és SQL injekció elleni védelem.
4. **Erős közösség és ökoszisztéma:** A Laravel csomagok (Composer-en keresztül) és a hivatalos bővítmények (pl. Laravel Breeze, Jetstream, Nova) megkönnyítik és gyorsítják a fejlesztést.
5. **API fejlesztés:** Laravel ideális RESTful API-k létrehozására is, valamint könnyen integrálható SPA (Single Page Application) frontendekkel, például Vue.js vagy React segítségével.

**Programozási nyelvek:**

**JavaScript:**

* Egy dinamikus, magas szintű, értelmezett programozási nyelv, amelyet eredetileg weboldalak interaktivitásának növelésére fejlesztettek ki.
* Manapság már nemcsak a böngészőkben, hanem szerveroldalon (Node.js segítségével), mobilalkalmazásokban és asztaliprogramokban is használják.

Főbb jellemzők:

* **Interaktív weblapok** készítésére alkalmas (pl. animációk, felhasználói műveletek kezelése).
* **Aszinkron működés** támogatása (Promise, async/await).
* **Prototípus-alapú objektumorientált programozás**.
* **Hozzáférés a DOM-hoz** (Document Object Model), amely lehetővé teszi az oldalak tartalmának és struktúrájának módosítását.

**Felhasználási területek:**

* **Webfejlesztés** (front-end és back-end is)
* **Mobilalkalmazások** (React Native, Ionic)
* **Játékfejlesztés** (p5.js, Phaser)
* **Szerveroldali fejlesztés** (Node.js)

**Webfejlesztés** (Django, Flask).

* **Adatelemzés és mesterséges intelligencia** (Pandas, NumPy, Scikit-learn, TensorFlow).
* **Automatizálás és szkriptelés** (rendszeradminisztráció, web scraping).
* **Játékfejlesztés** (Pygame).

**Adatbázis – dbdiagram.io**

A dbdiagram.io alkalmazást használtuk az adatbázis kezeléséhez.

A **dbdiagram.io** egy online eszköz, amely lehetővé teszi az adatbázisok vizuális modellezését és dokumentálását. A program segít adatbázis-tervezőknek, fejlesztőknek és adatbázis-adminisztrátoroknak az adatbázis-sémák gyors és egyszerű megjelenítésében. Az alábbiakban részletesen bemutatom a **dbdiagram.io** működését és a főbb funkcióit.

Főbb jellemzők:

1. **Egyszerű használat**:
   * A dbdiagram.io felülete intuitív és könnyen használható. Nem szükséges telepíteni semmilyen szoftvert, mivel az eszköz böngészőben fut. Az adatok vizuális modellezéséhez nem szükséges mély technikai tudás, de lehetőséget ad a tapasztalt felhasználóknak is a testreszabásra.
2. **Támogatott adatbázisok**:
   * A dbdiagram.io képes támogatni különböző adatbázis-kezelő rendszerek szintaxisát, mint például a **MySQL**, **PostgreSQL**, **SQLite**, **SQL Server**, **Oracle**, és **MongoDB**. Így széles körben alkalmazható különböző típusú adatbázisokhoz.
3. **Diagramok készítése**:
   * Az adatbázis-sémák automatikus generálása a program által használt egyedi szintaxissal történik, amellyel a felhasználók egyszerűen leírhatják a táblákat, azok kapcsolatait és egyéb attribútumokat.
   * A diagramok vizuális reprezentációja az adatbázis struktúrájának könnyű megértését biztosítja, így könnyebben navigálhatunk a táblák és azok kapcsolatai között.
4. **Szintaxis támogatás**:
   * A dbdiagram.io lehetővé teszi a felhasználók számára, hogy egy egyszerű szöveges szintaxissal írják le az adatbázis sémákat. Ez a szintaxis nagyon hasonlít a SQL definíciókhoz, de a program saját szintaxist alkalmaz, amelyet könnyű elsajátítani.
   * Például egy egyszerű tábla létrehozása a következőképpen nézhet ki:

pgsql

Table users {

id int [pk]

name varchar

email varchar

}

Ez egy users nevű táblát definiál, három mezővel: id, name, és email. Az id mező elsődleges kulcs (pk).

1. **Relációk létrehozása**:
   * A táblák közötti kapcsolatok könnyen ábrázolhatók a dbdiagram.io-ban. A relációk létrehozásához a felhasználó egyszerűen hivatkozhat a másik tábla elsődleges kulcsára (vagy idegen kulcsára).
   * Például:

makefile

Ref: users.id > posts.user\_id

Ez azt jelenti, hogy a users tábla id mezője kapcsolatban áll a posts tábla user\_id mezőjével.

1. **Exportálás és megosztás**:
   * A diagramokat különböző formátumokban exportálhatjuk, például **PDF** vagy **PNG** formátumban.
   * Továbbá a diagramokat könnyen megoszthatjuk másokkal. A dbdiagram.io lehetőséget ad a diagramok URL-címének megosztására, így bárki, akinek van hozzáférése, megtekintheti vagy szerkesztheti a diagramot.
2. **Több adatbázis támogatása**:
   * A program lehetőséget ad arra is, hogy több adatbázist egyetlen diagramon ábrázoljunk. Ez különösen akkor hasznos, ha komplex rendszerekben dolgozunk, ahol több adatbázis-kezelő rendszer is jelen van.
3. **Automatikus generálás**:
   * A dbdiagram.io képes automatikusan generálni a diagramot egy meglévő adatbázisból, ha annak SQL sémáját beimportáljuk. Ezzel az eszközzel az adatbázis struktúrája könnyen ábrázolható anélkül, hogy kézzel kellene megadni minden táblát és kapcsolatot.
4. **Integrációk**:
   * A dbdiagram.io integrálható különböző verziókezelő rendszerekkel, mint például a **GitHub** vagy **GitLab**, így a csapatok közösen dolgozhatnak a diagramokon. Az eszköz API-ján keresztül automatizált munkafolyamatokat is létrehozhatunk.

Hogyan használhatjuk a dbdiagram.io-t?

1. **Regisztráció és belépés**:
   * Az eszköz használatához először regisztrálni kell egy fiókot. A regisztráció után lehetőség van a diagramok mentésére, megosztására és verziók kezelésére.
2. **Diagram létrehozása**:
   * A főképernyőn egy új diagramot hozhatunk létre, amelyhez szintaxissal leírhatjuk az adatbázis-táblákat és azok kapcsolatait.
3. **Diagram szerkesztése**:
   * A diagramok szerkesztésére lehetőség van a szöveges szintaxis frissítésével. Mivel minden változtatás valós időben látható a diagramon, a felhasználók gyorsan reagálhatnak az adatbázis változásaira.
4. **Diagram megosztása**:
   * A diagramok URL-címét könnyen megoszthatjuk másokkal. Emellett lehetőség van az egyes diagramok exportálására különböző formátumokban, például PNG, PDF vagy SQL kód formájában.

Előnyök:

* **Egyszerűség**: A dbdiagram.io könnyen használható, nem szükséges komoly technikai háttér az alapvető funkciókhoz.
* **Vizuális megjelenítés**: Az adatbázisok struktúrájának ábrázolása segít a jobban megértésében és az együttműködésben.
* **Támogatott adatbázisok**: Széles körű támogatás a legnépszerűbb adatbázis-kezelők számára.
* **Rugalmas megosztás és exportálás**: Lehetővé teszi a diagramok egyszerű megosztását és exportálását.

Hátrányok:

* **Korlátozott testreszabás**: Bár az eszköz rendkívül egyszerű, bizonyos fejlettebb testreszabási lehetőségek hiányoznak, mint például a diagramok színének és stílusának teljes mértékű testreszabása.
* **Internetkapcsolat szükséges**: Mivel online eszköz, internetkapcsolatra van szükség a használatához.

Összegzés:

A **dbdiagram.io** egy hasznos és könnyen használható eszköz az adatbázisok vizuális modellezéséhez. Ideális választás mindazok számára, akik egyszerű, átlátható módon szeretnék megjeleníteni az adatbázis sémáikat, és gyorsan szeretnék megosztani azokat másokkal. Az egyszerű szintaxis és a széleskörű adatbázis-támogatás lehetővé teszi, hogy kezdő és haladó felhasználók egyaránt hatékonyan dolgozzanak vele.

A táblák könnyen létrehozhatók és feltölthetők.

Egyszerűen kezelhető felülettel rendelkezik.

**Az adatbázisról bővebben:**

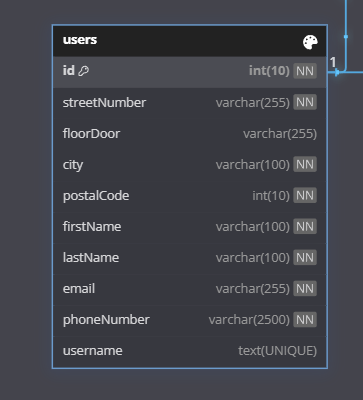
1. Általános magyarázat az adatbázis szerkezetéről

A **CarShop** adatbázis egy autóalkatrész webshop működéséhez szükséges adatokat tárolja. Az adatbázis normalizált szerkezetű, és külön táblákban kezeli a felhasználókat, termékeket, rendeléseket és a rendelési előzményeket. Az entitások között egyértelmű kapcsolatokat definiáltunk, amelyek biztosítják az adatok konzisztenciáját.

2. Táblánkként elemzés

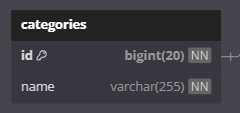
**Users (Felhasználók)**

* **id** – Egyedi azonosító
* **streetNumber** – Utca és házszám
* **floorDoor** – Emelet és ajtószám
* **city** – Város
* **postalCode** – Irányítószám
* **firstName** – Keresztnév
* **lastName** – Vezetéknév
* **email** – Email cím
* **phoneNumber** – Telefonszám
* **username** – Egyedi felhasználónév



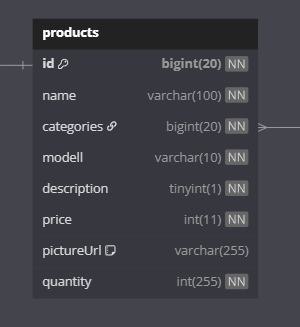
**Categories (Termékkategóriák)**

* **id** – Egyedi azonosító
* **name** – Kategória neve



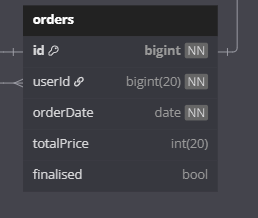
**Product (Termékek)**

* **id** – Egyedi azonosító
* **name** – Termék neve
* **categories** – Hivatkozás a termékkategóriára
* **modell** – Modellazonosító
* **description** – Rövid leírás
* **price** – Ár
* **pictureUrl** – Termékkép URL-je
* **quantity** – Raktárkészlet



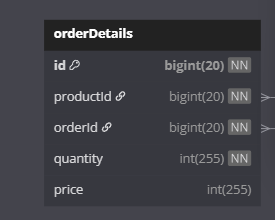
**Orders (Rendelések)**

* **id** – Egyedi rendelési azonosító
* **userId** – Felhasználó, aki a rendelést leadta
* **orderDate** – Rendelés dátuma
* **totalPrice** – Végösszeg
* **finalised** – Rendelés véglegesített állapota



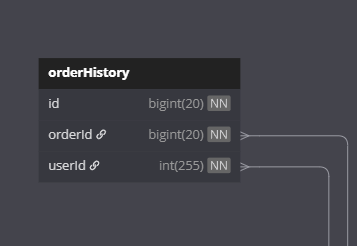
**OrderDetails (Rendelési Részletek)**

* **id** – Egyedi azonosító
* **productId** – Hivatkozás a termékre
* **orderId** – Hivatkozás a rendelésre
* **quantity** – Rendelt mennyiség
* **price** – Termék ára az adott rendelésben



**OrderHistory (Rendelési előzmények)**

* **id** – Egyedi azonosító
* **orderId** – Hivatkozás a rendelésre
* **userId** – Hivatkozás a felhasználóra



**Frontend programozás keretrendszere:**

A **React** egy népszerű, nyílt forráskódú JavaScript könyvtár, amelyet a **Facebook** fejlesztett ki és tart karban. React elsősorban a felhasználói felületek (UI) építésére szolgál, és lehetővé teszi az egyszerű, gyors és dinamikus webalkalmazások fejlesztését. A React nem egy teljes keretrendszer, hanem egy **UI könyvtár**, amely segít az alkalmazások komponens-alapú felépítésében.

1. **A React alapjai**

A React a **komponens-alapú fejlesztés** paradigmájára épít, ami azt jelenti, hogy a felhasználói felületet kisebb, újrahasználható komponensekre bontja. Mindez nagyban hozzájárul a karbantartás egyszerűsítéséhez és az alkalmazások skálázhatóságához.

A React főbb jellemzői:

* **Komponens-alapú felépítés**: Az alkalmazás UI-ját különálló komponensek alkotják, melyek egymásba ágyazódhatnak.
* **Virtuális DOM**: A React a változtatásokat először a virtuális DOM-ban hajtja végre, majd azokat az igazi DOM-ra alkalmazza. Ez javítja a teljesítményt, mivel csak a valóban változott elemek frissülnek.
* **Deklaratív megközelítés**: Ahelyett, hogy lépésről lépésre leírnánk, hogyan frissítsük a felhasználói felületet, a React azt mondja meg, hogy mi legyen a felület állapota, és automatikusan kezelni fogja a változásokat.

2. **React komponensek**

A React alkalmazásokat **komponensek** építik fel. Minden komponens egy JavaScript függvény vagy osztály, amely az alkalmazás egy részét képviseli, például egy gombot, egy űrlapot, vagy akár egy egész oldalt. A komponens két fő részből áll:

1. **Állapot (State)**: A komponensben tárolt adat, amely befolyásolja a renderelt tartalmat. Az állapot változása újrarendereli a komponenst.
2. **Tulajdonságok (Props)**: A komponensnek átadott adatok, amelyeket a szülő komponens biztosít a gyermek komponens számára. A props nem változtathatóak a gyermek komponensben.

Példa egy egyszerű React komponensre:

jsx

import React, { useState } from 'react';

function Counter() {

const [count, setCount] = useState(0);

return (

<div>

<p>Count: {count}</p>

<button onClick={() => setCount(count + 1)}>Increment</button>

</div>

);

}

export default Counter;

Ebben a példában:

* **useState** egy React Hook, amely a komponens állapotát kezeli.
* Az onClick eseménykezelő frissíti az állapotot, ami újrarendereli a komponenst, hogy tükrözze a változást.

3. **Virtuális DOM és teljesítmény**

A React a **virtuális DOM** koncepcióját használja, amely segít optimalizálni az alkalmazás teljesítményét. A valódi DOM frissítése drága művelet lehet, mivel a böngésző az egész oldalt újrarendereli. A virtuális DOM viszont egy könnyen módosítható JavaScript objektumot jelent, amely tükrözi az aktuális DOM struktúrát.

Amikor a React komponensének állapota vagy a props változik, a React először a virtuális DOM-ban frissíti az elemeket, majd összehasonlítja a változtatásokat a valódi DOM-mal. Csak a szükséges változtatásokat alkalmazza, így csökkenti a felesleges újrarendereléseket, ami javítja a teljesítményt.

4. **React Hooks**

A **React Hooks** egy viszonylag új funkció, amely lehetővé teszi a React funkcionális komponenseiben az állapot kezelését és egyéb funkciókat, amelyek korábban csak osztálykomponensekben voltak elérhetők. A leggyakrabban használt hookok:

* **useState**: A komponens állapotának kezelése.
* **useEffect**: A komponens élettartamának kezelésére szolgál, például aszinkron adatlekérésekhez vagy eseménykezelőkhöz.
* **useContext**: A React Context API-val való interakcióhoz, amely lehetővé teszi, hogy adatokat osszunk meg a komponensek között anélkül, hogy props-okkal kellene őket átadni.

Példa a useEffect használatára:

jsx

import React, { useState, useEffect } from 'react';

function DataFetcher() {

const [data, setData] = useState(null);

useEffect(() => {

fetch('https://api.example.com/data')

.then(response => response.json())

.then(data => setData(data));

}, []); // Üres tömb: csak egyszer, a komponens betöltésekor hívódik meg

return <div>{data ? JSON.stringify(data) : 'Loading...'}</div>;

}

export default DataFetcher;

5. **Routing a React alkalmazásokban**

A React önállóan nem tartalmaz routing megoldást, de a **React Router** könyvtár segítségével kezelhetjük az oldalak közötti navigációt egy egyoldalas alkalmazás (SPA) esetén. A React Router lehetővé teszi a dinamikus URL-kezelést, és az URL alapján a megfelelő komponens renderelését.

Példa a React Router használatára:

jsx

import React from 'react';

import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';

function Home() {

return <h2>Home Page</h2>;

}

function About() {

return <h2>About Page</h2>;

}

function App() {

return (

<Router>

<Switch>

<Route path="/about" component={About} />

<Route path="/" component={Home} />

</Switch>

</Router>

);

}

export default App;

6. **State Management (Állapotkezelés)**

A React alapértelmezett állapotkezelése a komponens szintjén történik, azonban nagyobb alkalmazások esetén szükség lehet globális állapotkezelésre. Erre számos könyvtár létezik, például:

* **Redux**: Egy nagyon népszerű állapotkezelő könyvtár, amely egy központi tárolót használ az alkalmazás állapotának kezelésére.
* **Context API**: A React saját megoldása, amely lehetővé teszi, hogy globálisan elérhető adatokat osszunk meg a komponensek között.

7. **React teljesítmény és optimalizálás**

Bár a React általában gyors, nagyobb alkalmazásoknál érdemes figyelni az optimalizálásra. Néhány technika, amely segíthet:

* **Code splitting**: A kódot kisebb darabokra oszthatjuk, így csak a szükséges részeket töltjük be, amikor azok valóban szükségesek.
* **React.memo**: Ez a funkció lehetővé teszi, hogy elkerüljük a felesleges újrarendereléseket a komponensek szintjén, ha azok props-a nem változott.
* **Lazy loading**: A komponensek késleltetett betöltése a teljesítmény javítására.

8. **Miért válasszuk a React-ot?**

A React előnyei közé tartozik:

* **Komponens-alapú fejlesztés**: A React az alkalmazásokat kisebb, újrahasználható komponensekre bontja, ami elősegíti a karbantartást és a kód újrahasznosítását.
* **Virtuális DOM**: A React hatékonyabb DOM manipulációt biztosít, javítva a teljesítményt.
* **React közösség**: A React nagy és aktív közösséggel rendelkezik, rengeteg dokumentációval, tutorialokkal és könyvtárakkal.
* **Bővíthetőség**: A React könnyen integrálható más könyvtárakkal és eszközökkel, mint például a Redux, React Router, stb.
* **Széles körű támogatás**: A React az egyik legnépszerűbb front-end könyvtár, és számos nagyvállalat (pl. Facebook, Instagram, Airbnb) használja.

Összegzés

A React egy rendkívül népszerű és erőteljes JavaScript könyvtár, amely segít modern, dinamikus felhasználói felületek építésében. A komponens-alapú fejlesztés, a virtuális DOM és a React Hooks révén a React lehetővé teszi a rugalmas és hatékony webalkalmazások fejlesztését. Az aktív közösség és a bővíthetőség miatt a React remek választás minden fejlesztő számára, aki dinamikus alkalmazásokat kíván létrehozni.

**Bootstrap:**

A **Bootstrap** egy népszerű, nyílt forráskódú front-end keretrendszer, amelyet a weboldalak gyors fejlesztésére terveztek. A keretrendszert először 2011-ben készítette el a Twitter csapata, és azóta széles körben elterjedt a webfejlesztők körében, mivel lehetővé teszi a reszponzív, modern, és felhasználóbarát felületek gyors létrehozását.

1. **Bootstrap alapjai**

A Bootstrap az alábbi három fő összetevőre épít:

* **HTML** – A Bootstrap előre definiált HTML struktúrákat és komponenseket biztosít, amelyek gyorsan testreszabhatók.
* **CSS** – A Bootstrap CSS fájljai előre meghatározott stílusokat tartalmaznak, amelyek segítenek a dizájn elemek gyors létrehozásában. A rendszer tartalmazza a rácsos elrendezést, tipográfiát, formázásokat és színeket.
* **JavaScript/jQuery** – A keretrendszer tartalmaz JavaScript (vagy jQuery) pluginokat, amelyek dinamikus elemeket és interaktivitást adnak az oldalhoz (pl. modális ablakok, dropdown menük, értesítések).

2. **Reszponzivitás és rácsos rendszer**

A Bootstrap egyik legfontosabb jellemzője, hogy **reszponzív**. Ez azt jelenti, hogy a weboldalak a különböző eszközökhöz (mobil, tablet, asztali számítógép) alkalmazkodnak, és a felhasználói élmény mindig optimális marad.

* **Rácsos rendszer**: A Bootstrap 12 oszlopos rácsos rendszert használ, ami azt jelenti, hogy az oldal elrendezése 12 oszlopra van felosztva. Az oszlopok szélessége rugalmasan változik az eszközök képernyőméretének megfelelően. A különböző oszlopok a következő osztályok segítségével állíthatók be: .col-xs-\*, .col-sm-\*, .col-md-\*, .col-lg-\*, ahol az \* helyére a kívánt oszlopok számát írjuk.

Példa a rács használatára:

<div class="row">

<div class="col-md-4">Egyharmad szélesség</div>

<div class="col-md-8">Két harmad szélesség</div>

</div>

3. **Komponensek**

A Bootstrap rengeteg előre elkészített komponenst tartalmaz, amelyeket gyorsan beilleszthetünk a projektünkbe. Néhány példa a leggyakrabban használt komponensekre:

* **Navigációs menük**: A navigációs sávak és menük könnyen testreszabhatók és reszponzívak.
* **Gombok**: Az előre definiált gombok egyszerűen hozzáadhatók az oldalon lévő interaktív elemekhez.
* **Formák**: A Bootstrap segít a formák kialakításában, tartalmaz előre definiált mezőket, validálásokat és egyéb elemeket.
* **Modális ablakok**: A modális ablakok egyszerűen hozzáadhatók és testreszabhatók.
* **Kártyák**: A kártyák lehetővé teszik a tartalom szép és rendezett megjelenítését.
* **Listák és menük**: A komponensek tartalmaznak előre meghatározott stílusokat a listákhoz és menükhöz.

4. **Javaslatok és Pluginok**

A Bootstrap különféle **JavaScript pluginokat** is tartalmaz, amelyek különféle dinamikus viselkedéseket biztosítanak:

* **Carousel (képforgató)**: Egy képforgató (carousel) rendszer, amely lehetővé teszi több kép vagy elem forgatását.
* **Tooltip (eszköztipp)**: Kis szöveges információk, amelyek akkor jelennek meg, ha a felhasználó az egérrel egy elem fölé viszi az egeret.
* **Popover**: Olyan felugró ablak, amely tartalmazhat szöveget, képeket vagy akár űrlapokat is.
* **Alert (értesítések)**: Figyelmeztető üzenetek, amelyek felhívják a figyelmet egy fontos információra vagy hibára.

5. **Testreszabás**

Bár a Bootstrap rengeteg előre meghatározott stílust kínál, könnyedén testreszabhatjuk azokat. A keretrendszer a **Sass (Syntactically Awesome Stylesheets)** segítségével testreszabható, így módosíthatók az alapértelmezett színek, betűtípusok, margók és egyéb stílusok.

* Az **alapértelmezett téma** módosítása: A Bootstrap a legfontosabb stílusokhoz szín- és méretezési változókat használ, így gyorsan létrehozhatunk egyedi dizájnt.
* **Sass fájlok** használata: Ha a Sass fájlokat használjuk, az egyszerűbbé teszi a testreszabást és a karbantartást.

6. **Bootstrap Verziók**

A Bootstrap folyamatosan fejlődik, és az új verziók különböző új funkciókat és javításokat hoznak. A legfontosabb verziók:

* **Bootstrap 3**: A Bootstrap 3 volt az első, amely a reszponzív tervezést támogatta.
* **Bootstrap 4**: Az új verzióban több új funkció is megjelent, mint a Flexbox alapú elrendezés és az új CSS Grid támogatás.
* **Bootstrap 5**: A legújabb verzió, amely eltávolította a jQuery-t, és további finomhangolásokat hozott a komponensekhez. Emellett a szintaxis és a funkcionalitás is átalakult a modernebb webes szabványoknak megfelelően.

7. **Miért használjuk a Bootstrapot?**

A Bootstrap előnyei közé tartozik:

* **Gyors fejlesztés**: Az előre definiált komponensek és stílusok lehetővé teszik, hogy gyorsan elkészítsünk egy szép és működő weboldalt.
* **Reszponzivitás**: A keretrendszer automatikusan alkalmazkodik az eszközök különböző képernyőméreteihez.
* **Közösségi támogatás**: A Bootstrapot széleskörű közösség használja, így könnyű segítséget találni és sokféle példa áll rendelkezésre.
* **Testreszabhatóság**: A keretrendszer testreszabható a projekt igényei szerint, így nem kell mindent nulláról építeni.

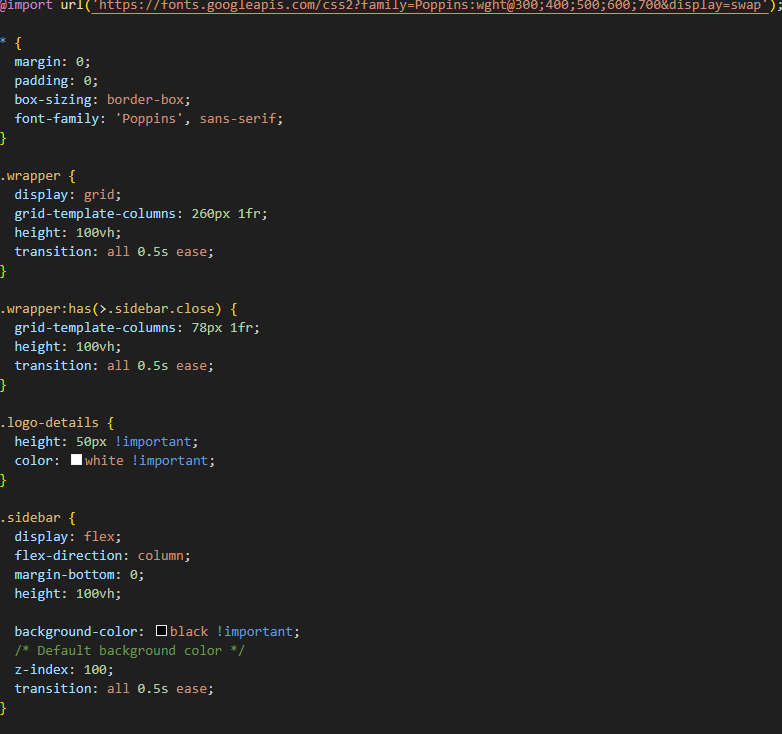
**Összefoglalás**

A Bootstrap egy erőteljes, rugalmas és könnyen használható front-end keretrendszer, amely lehetővé teszi a fejlesztők számára, hogy gyorsan készítsenek reszponzív weboldalakat és alkalmazásokat. A keretrendszer előre elkészített HTML, CSS és JavaScript komponenseket tartalmaz, amelyek segítségével a felhasználói élményt magas színvonalon lehet kialakítani. Az egyszerű integráció, a reszponzív tervezés, és a testreszabhatóság miatt a Bootstrap az egyik legnépszerűbb választás a webfejlesztők számára.

**Bootstrap főbb jellemzői:**

* **Reszponzív dizájn**: A Bootstrap **rácsrendszere** (grid system) lehetővé teszi, hogy az oldalak megfelelően jelenjenek meg különböző képernyőméreteken (pl. mobil, tablet, asztali gép).
* **Előre elkészített komponensek**: Tartalmaz rengeteg előre megírt elemet, mint például **gombok, űrlapok, kártyák, navigációs menük, modális ablakok** stb.
* **CSS és SASS támogatás**: Alapértelmezett stílusokat biztosít, de **SASS** segítségével könnyen testreszabható.
* **JavaScript komponensek**: Olyan interaktív elemeket tartalmaz, mint a **carousel (képváltó), dropdown (lenyíló menü), tooltip (buboréksúgó)** stb.
* **Könnyű használat**: Egyszerűen integrálható egy projekthez egy **CDN (Content Delivery Network)** link vagy a letöltött fájlok használatával.

Részletek a forráskódból:



# **Tesztdokumentáció**

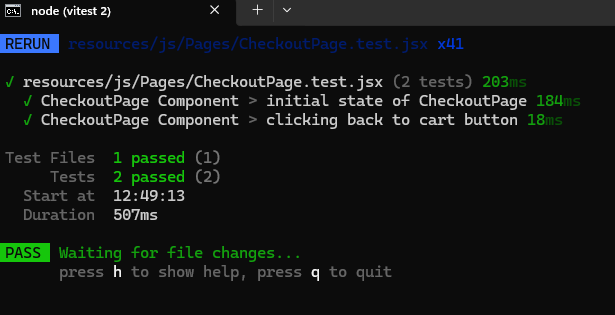
**A React tesztek általában a React alkalmazások vagy komponensek funkcionalitását és megjelenését tesztelik. A tesztek általában automatikusan futnak, és több fő típusra oszthatók: egységtesztek és integrációs tesztek.**

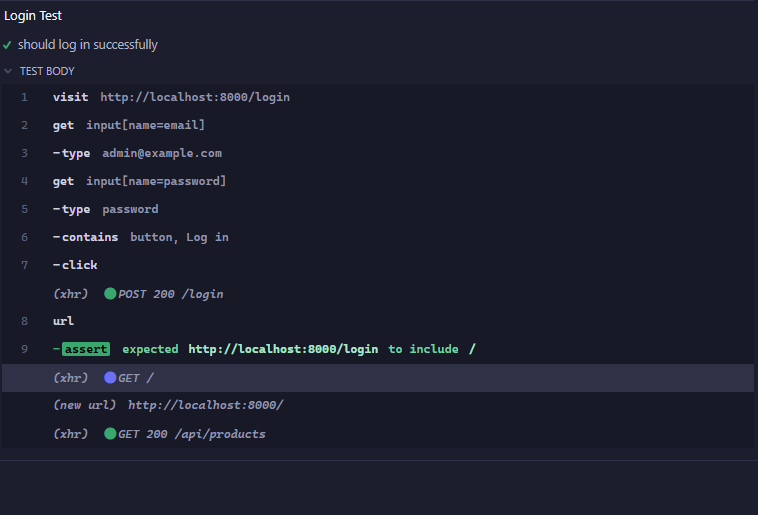
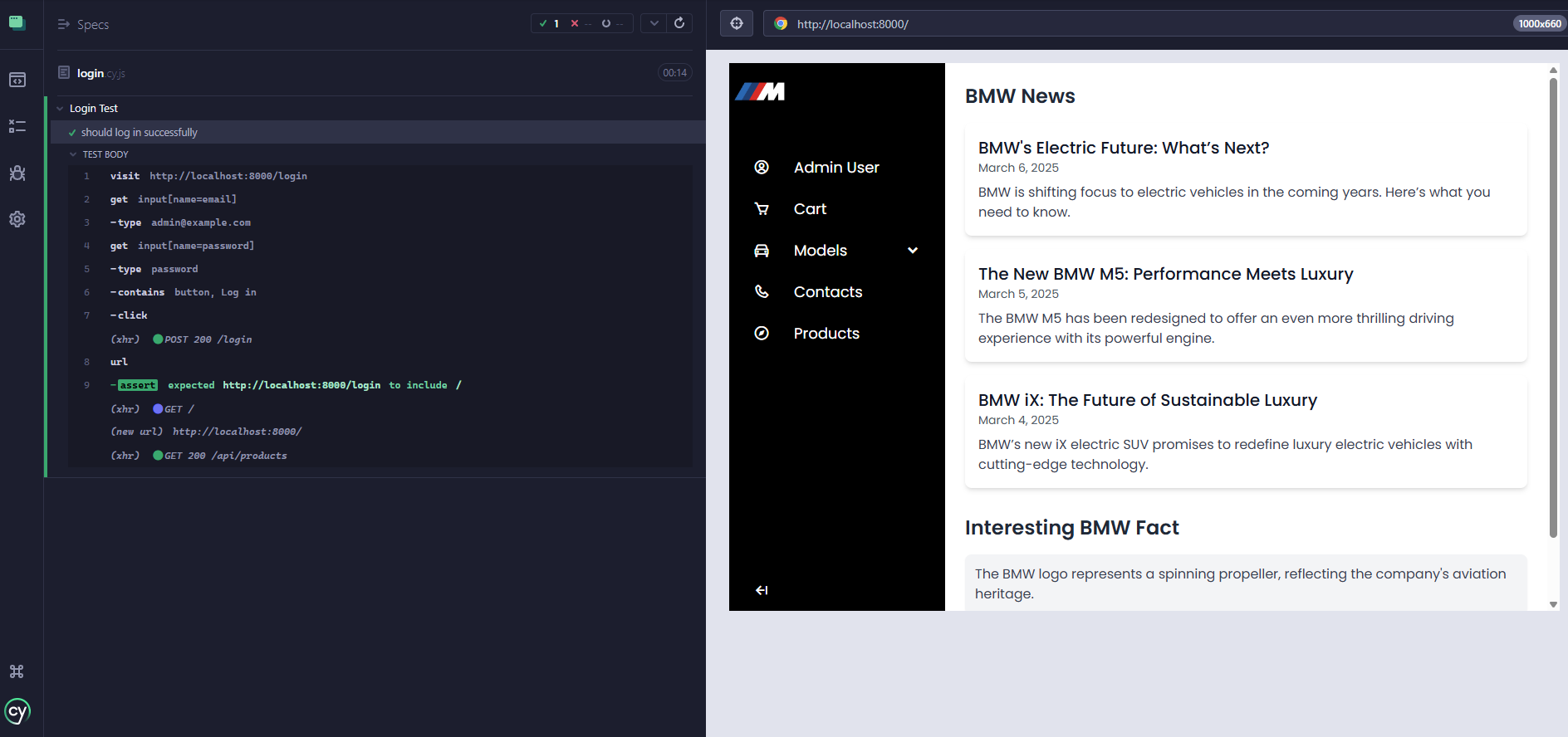
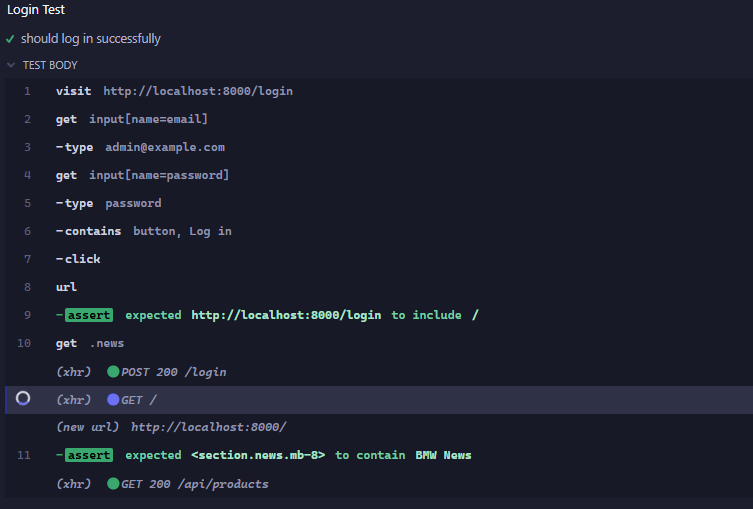
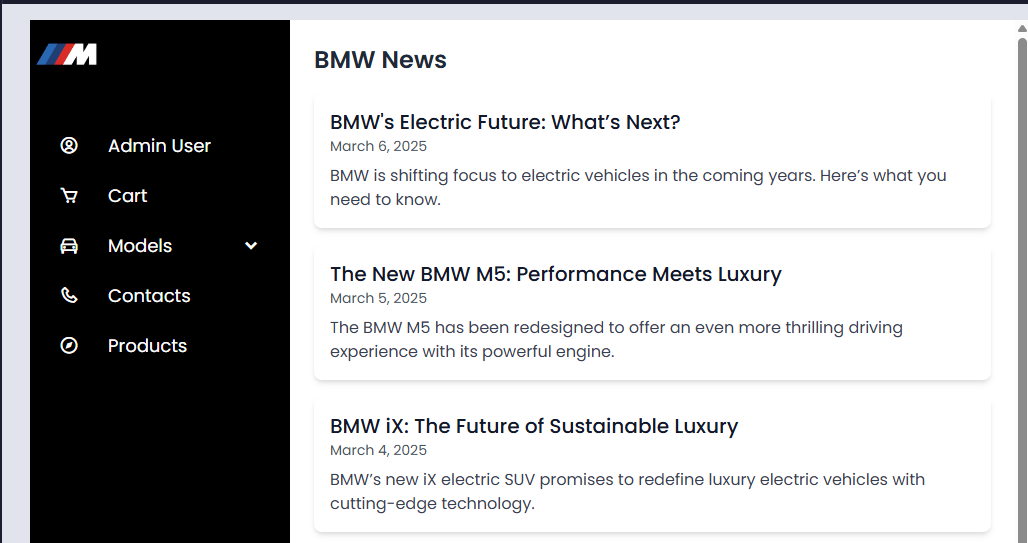
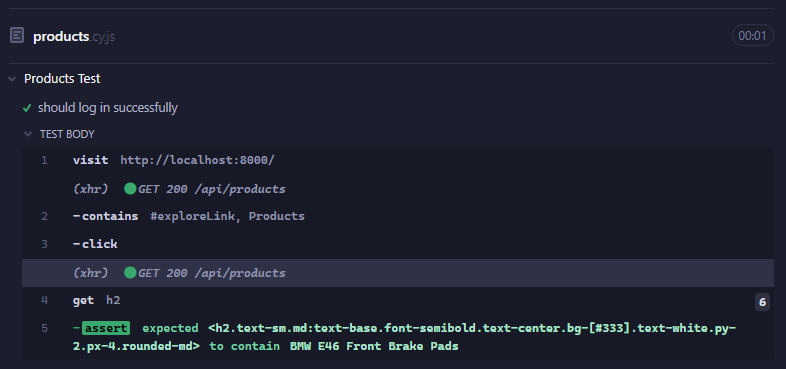
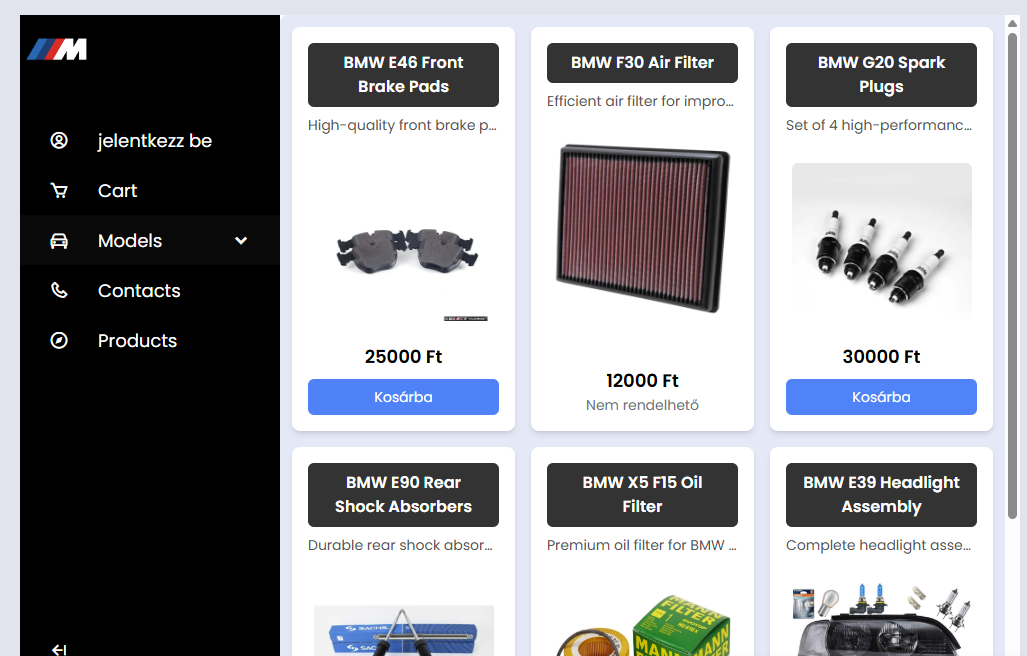
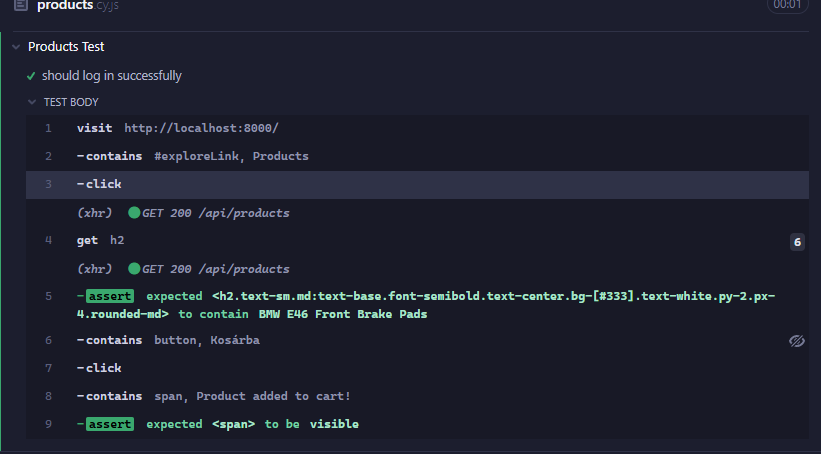
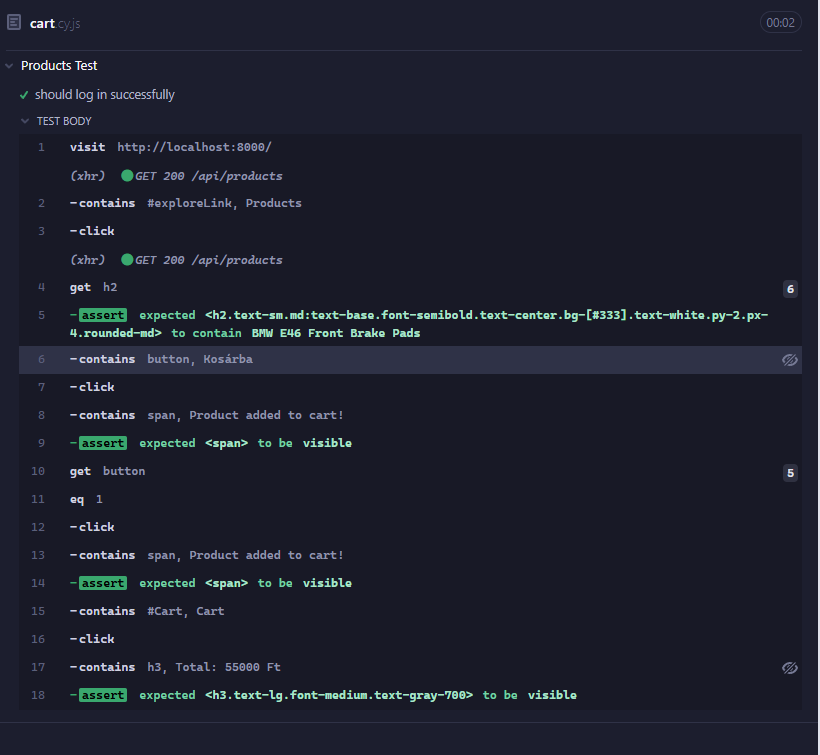
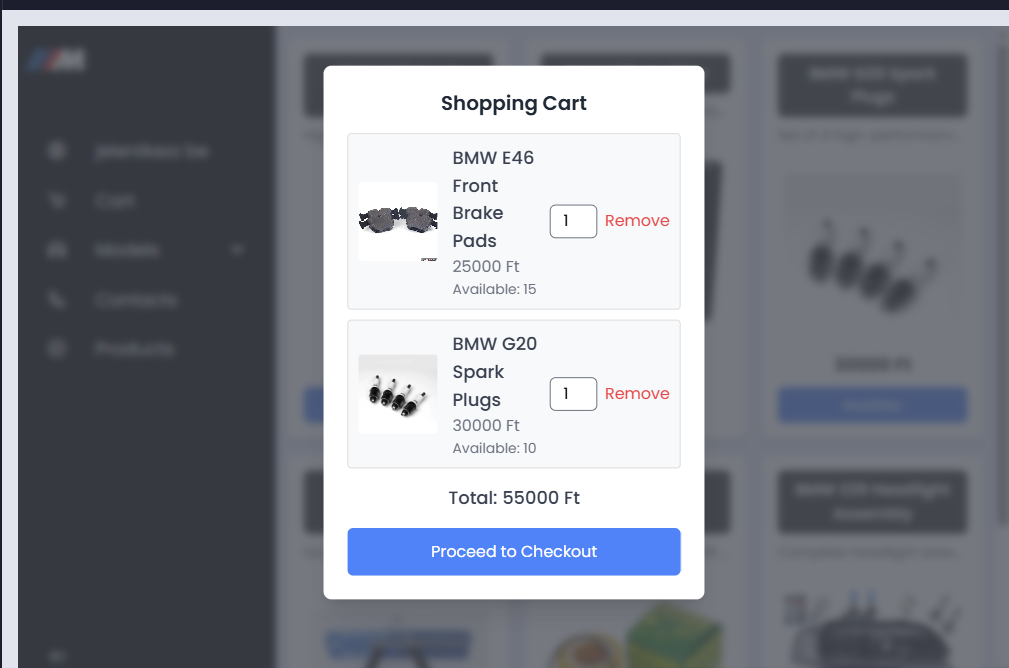
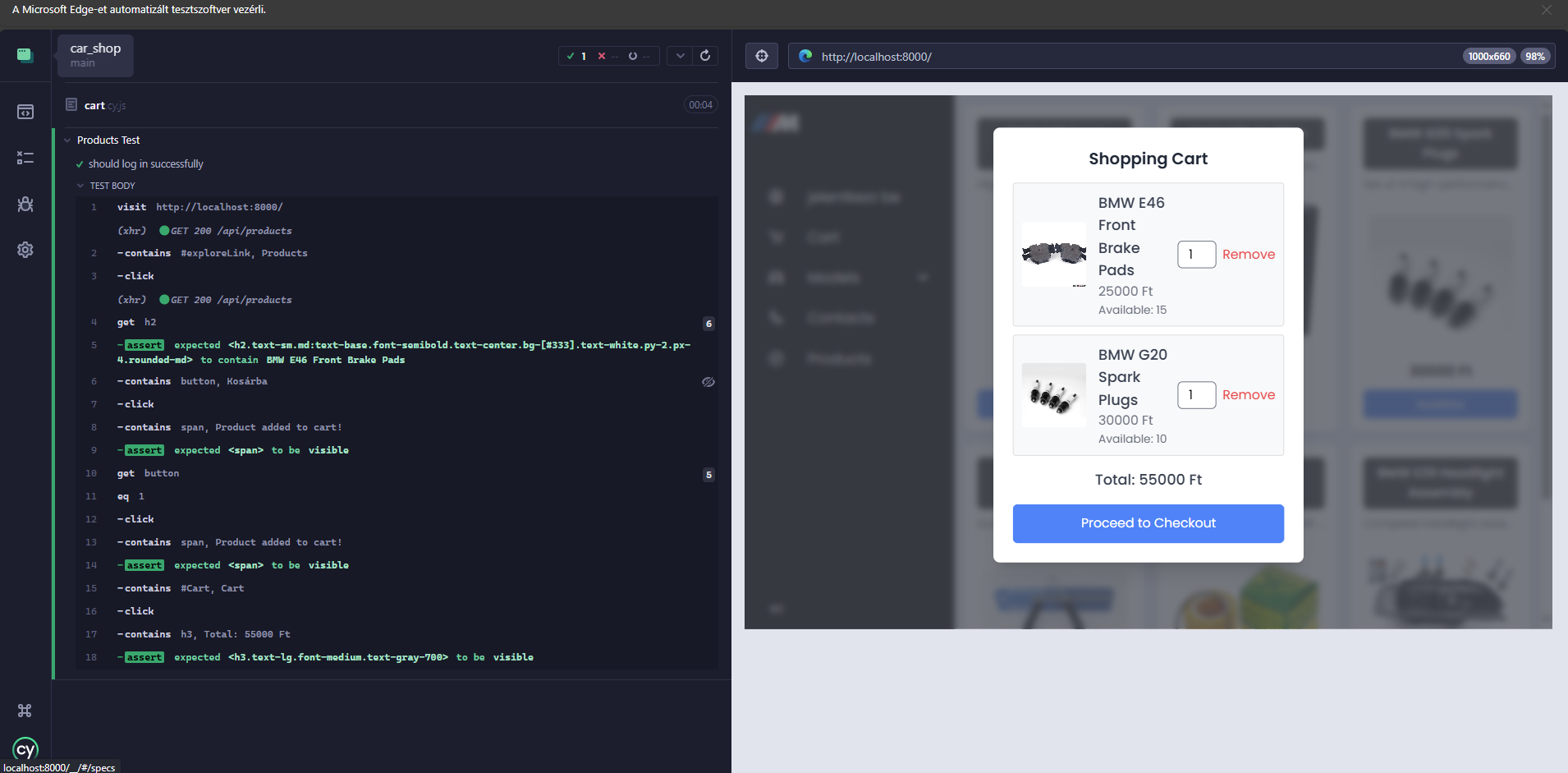
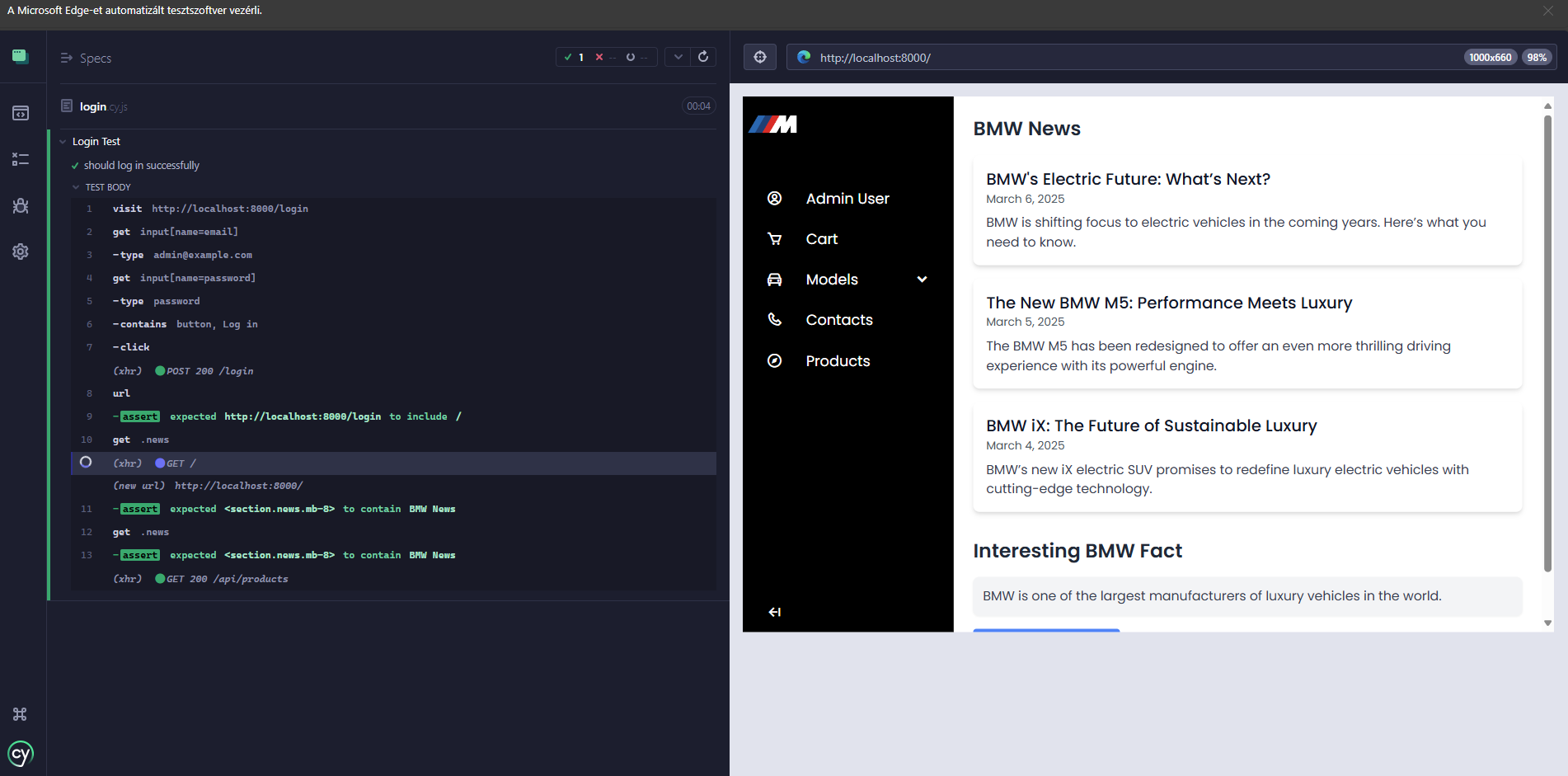
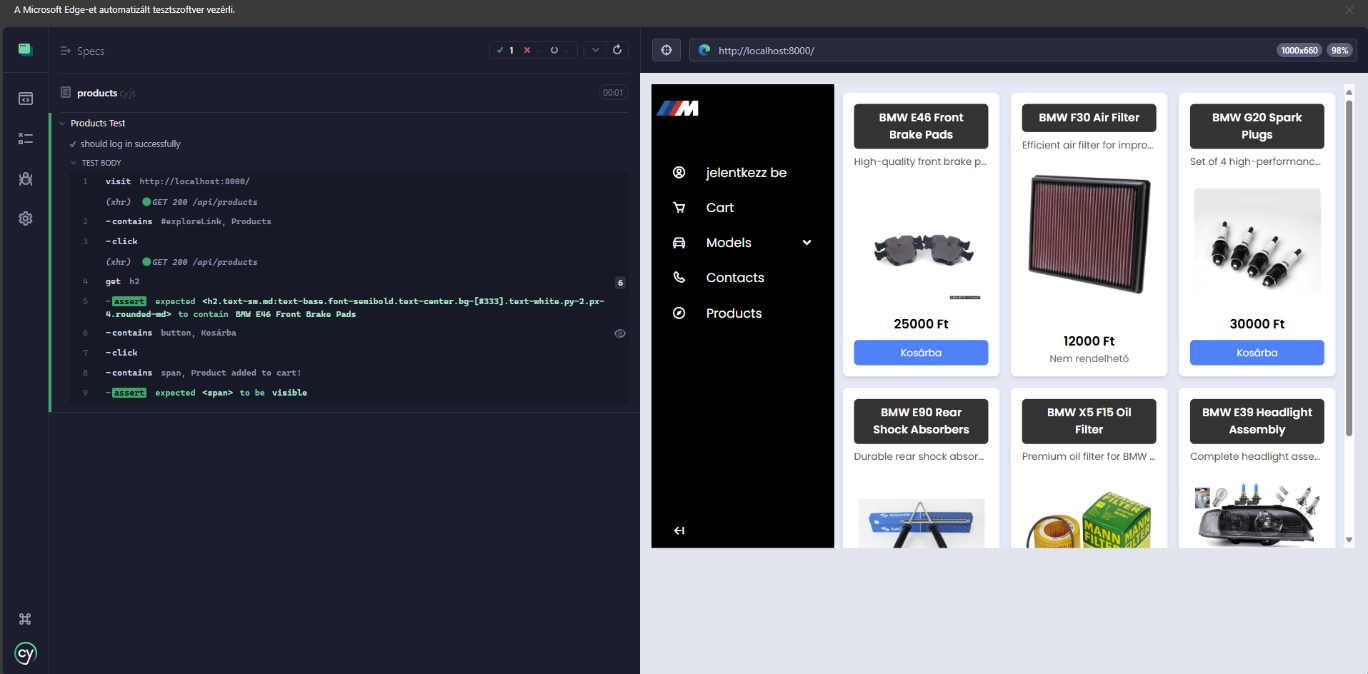
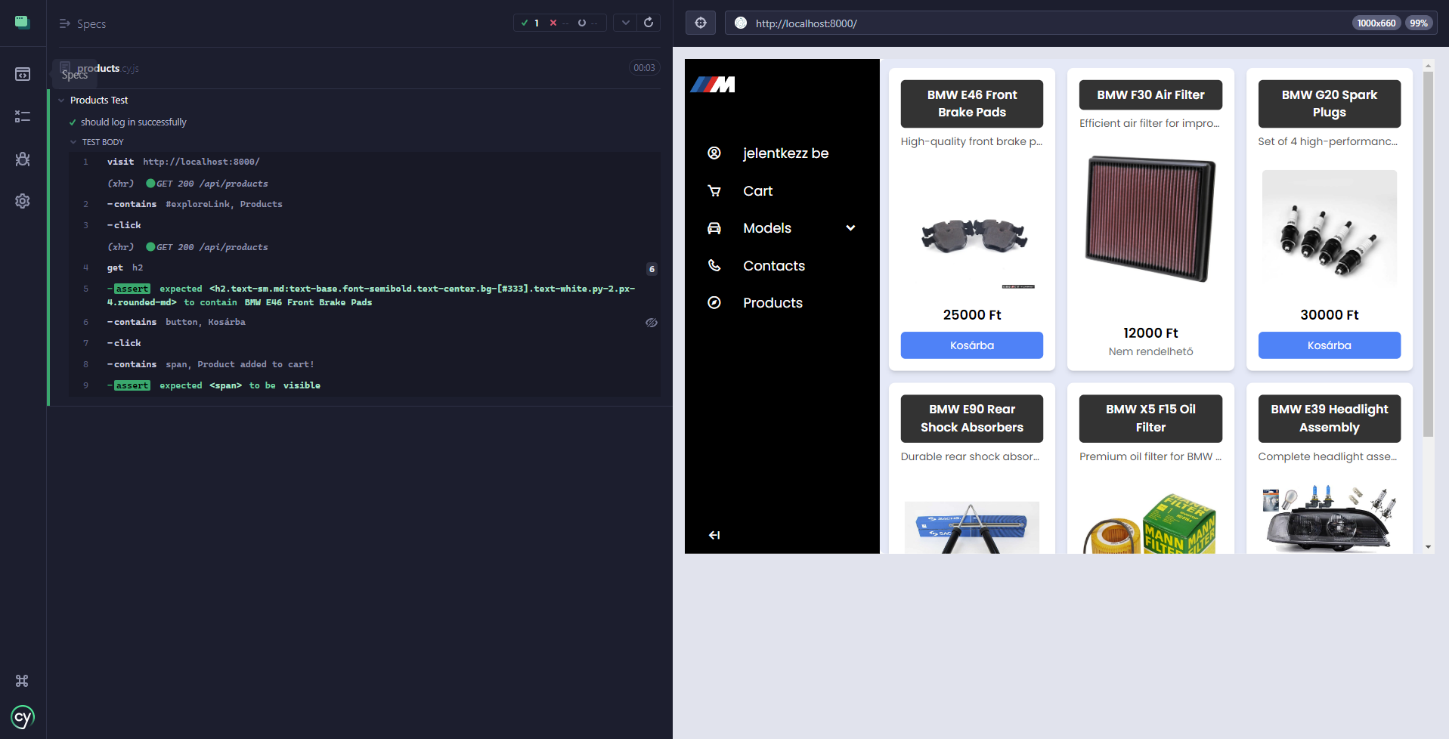
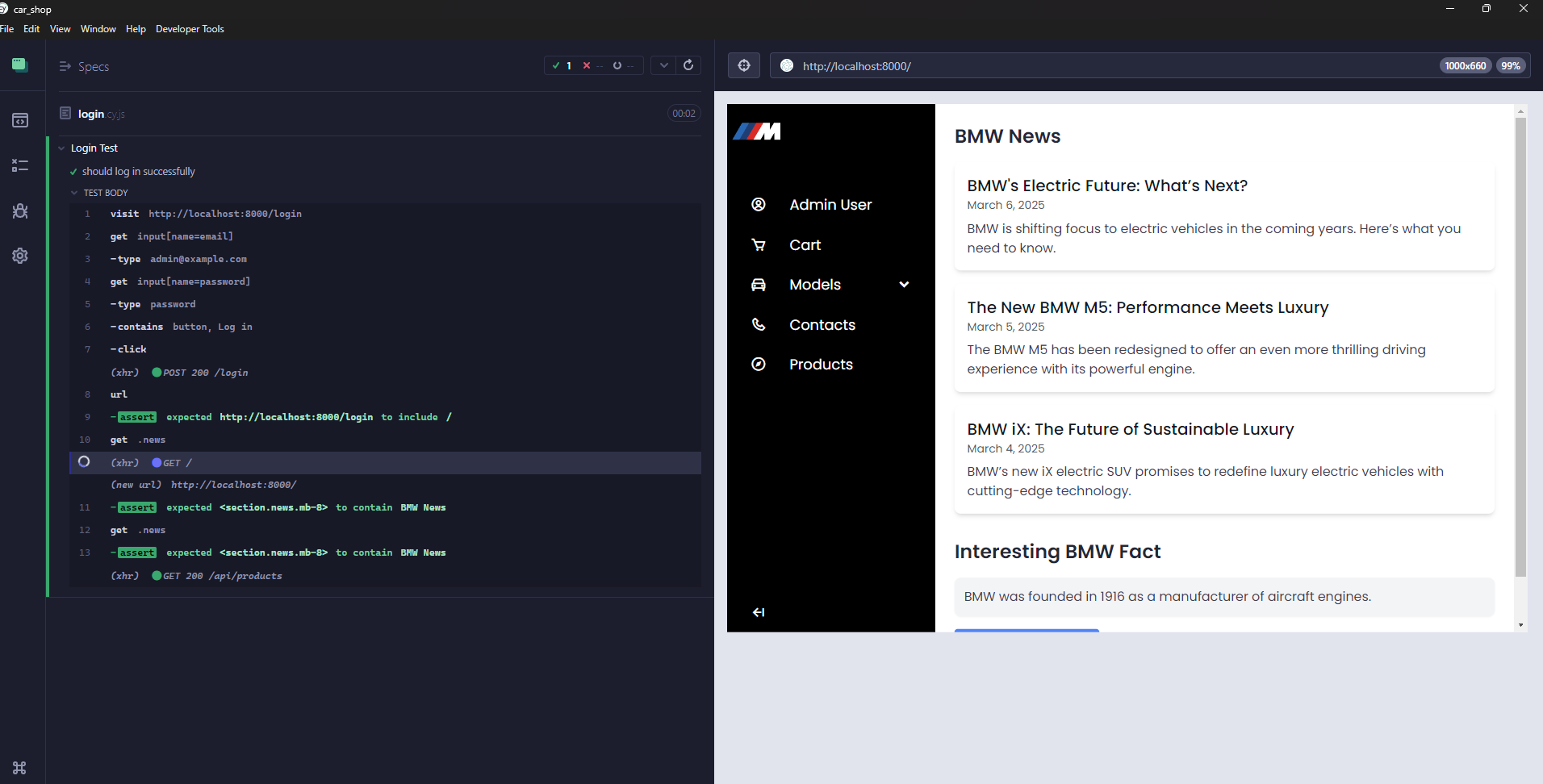
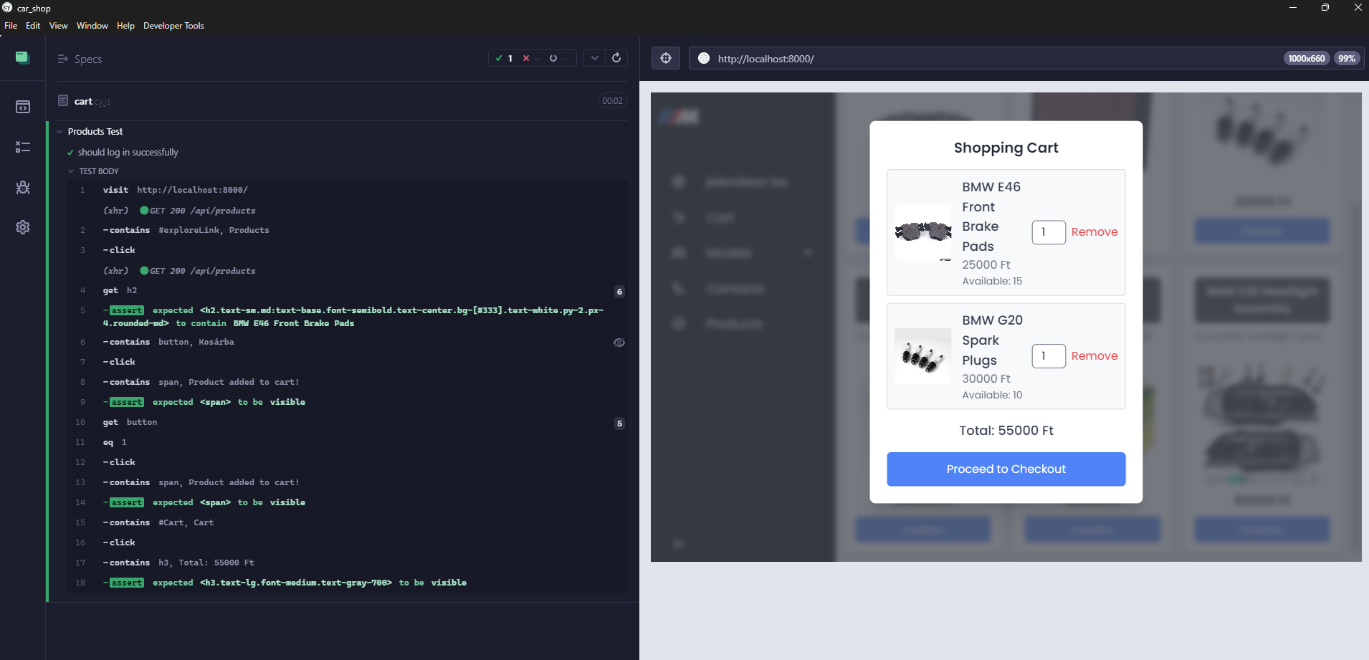
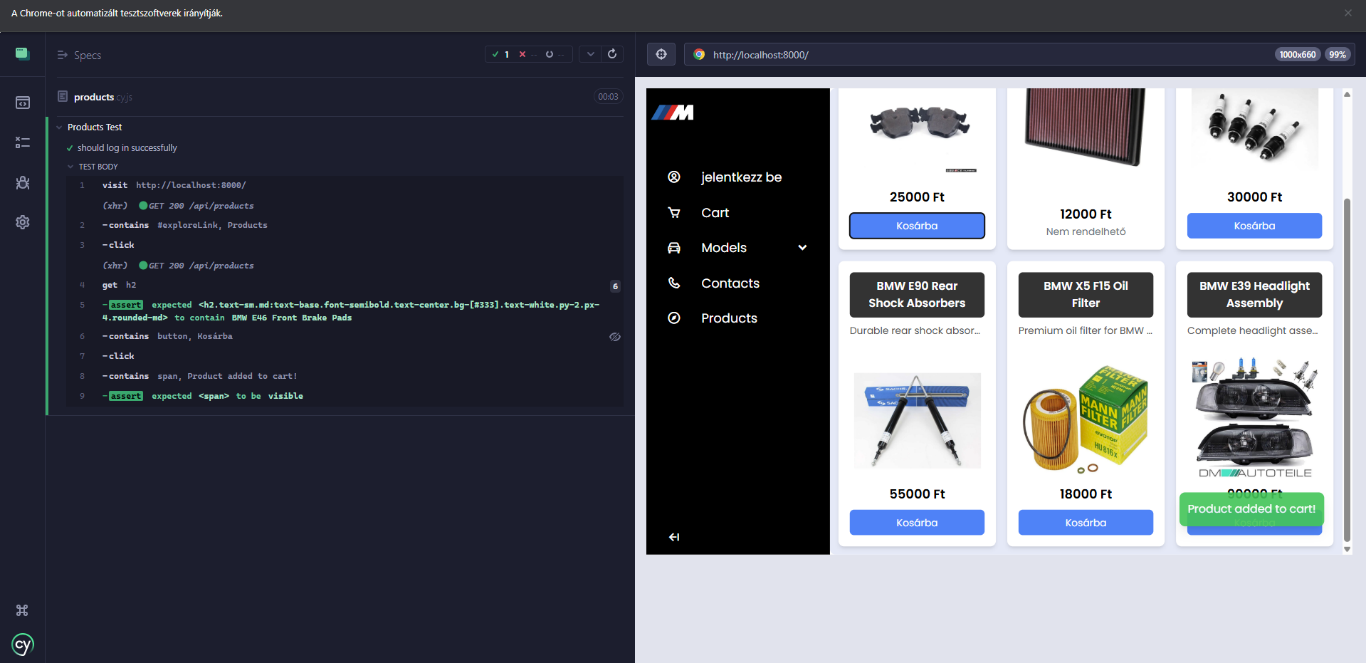
**Az egységtesztek a React komponensek vagy funkciók egyes részeit tesztelik, például a komponens állapotát vagy a funkció visszatérési értékét. Az egységtesztek segítségével ellenőrizhető, hogy a komponensek és funkciók helyesen működnek-e és helyes adatokat adnak-e vissza.**

**Az integrációs tesztek az egyes komponensek együttműködését és a rendszer egészének működését tesztelik. Az integrációs tesztek az egyes komponensek általános interakcióját, valamint a külső függőségeket, mint például az API-kat és az adatbázisokat is ellenőrzik.**

**A React tesztek futtatásához számos tesztelési keretrendszer áll rendelkezésre, például a Jest vagy az Enzyme. Az ilyen tesztelési keretrendszerek lehetővé teszik a tesztek egyszerű és hatékony létrehozását, futtatását és eredményeinek elemzését.**

**Összességében a React tesztek nagyon fontosak az alkalmazások minőségének biztosításához, és a fejlesztőknek érdemes részletesen megvizsgálniuk a tesztelési keretrendszereket és eszközöket a hatékony és megbízható tesztelés érdekében.**





Létrehoztunk egy teszt felhasználót, amivel a tesztelők tudnak majd tesztelni.

Email cím: admin@example.com

Jelszó: password

# **Felhasználói dokumentáció**

Az elkészült program futtatásához nemes egyszerűséggel csak egy webböngészőre lesz szükségünk. A programunkat futtathatjuk Windows alapú eszközökön, MacOS alapú eszközökön, illetve tetszőleges mobil alapú eszközökön is.

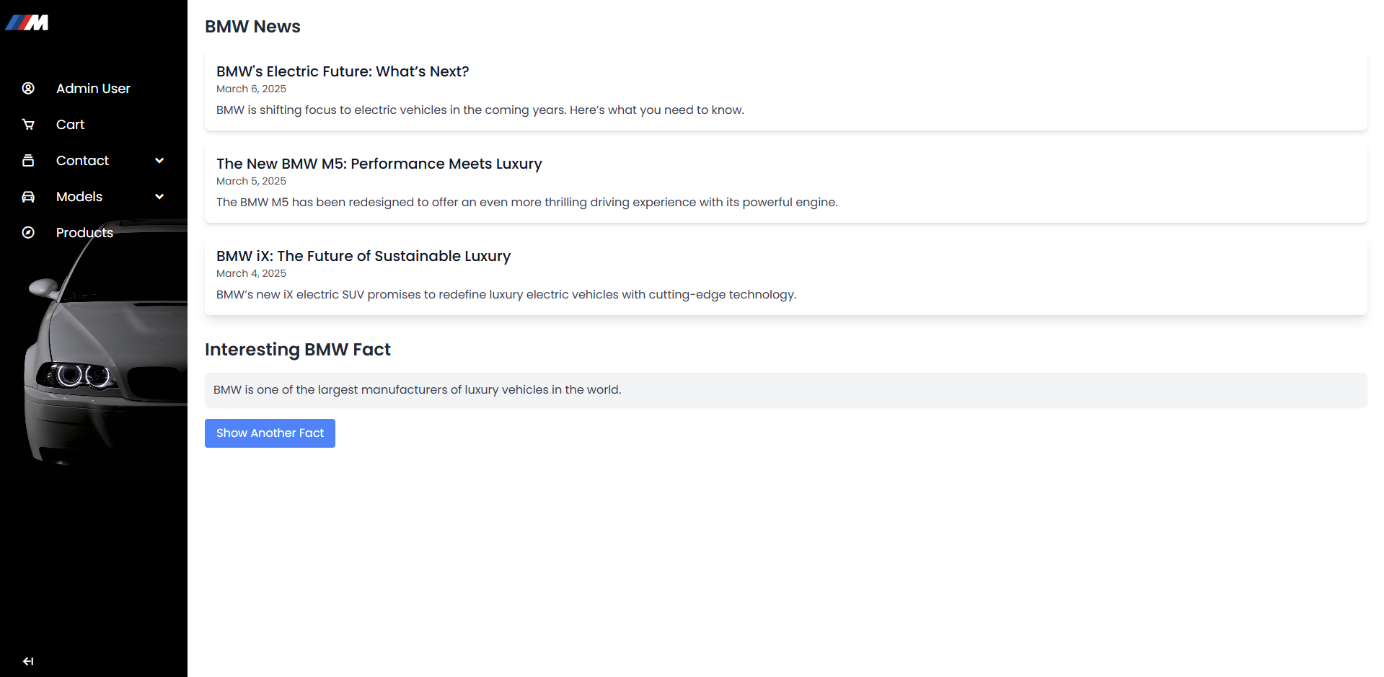
Néhány böngésző a program futtatásához:

Google Chrome

Microsoft Edge

Safari

Kezdőképernyő a program elindításakor:



Ez itt a legelső felület, amikor fellépünk a weboldalra.

A weboldal egy letisztult, modern felülettel fogadja a látogatókat, amelynek középpontjában a legfrissebb BMW-hírek állnak. A főoldalon három kiemelt cikk található, amelyek az elektromos jövőről, a legújabb BMW M5 modellről és az iX elektromos SUV-ról szólnak.

A bal oldali navigációs sáv lehetőséget biztosít a felhasználóknak arra, hogy elérjék a különböző menüpontokat, például a **Cart (Kosár), Contact (Kapcsolat), Models (Modellek)** és **Products (Termékek)** részeket. A menü ikonokkal és jól strukturált elrendezéssel segíti a gyors eligazodást.

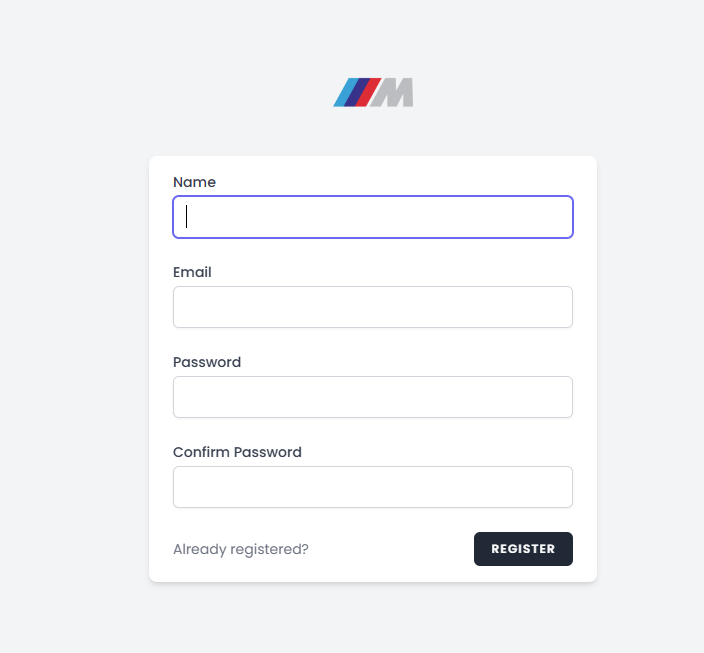
Az oldalon egy "Érdekes BMW tény" szekció is található, amely véletlenszerű információkat jelenít meg a márkáról, és egy gomb segítségével újabb érdekességeket kérhetünk.

Az oldal dizájnja elegáns, minimalista stílusú, amely illik a BMW prémium márkájának arculatához. A fekete-fehér színvilág és a háttérben látható BMW modell egy exkluzív és modern megjelenést kölcsönöz a felületnek.

**Regisztráció**

Ha szeretnénk a weboldalra funkciót használni, akkor ajánlott regisztrálni.

Regisztrációs felület:

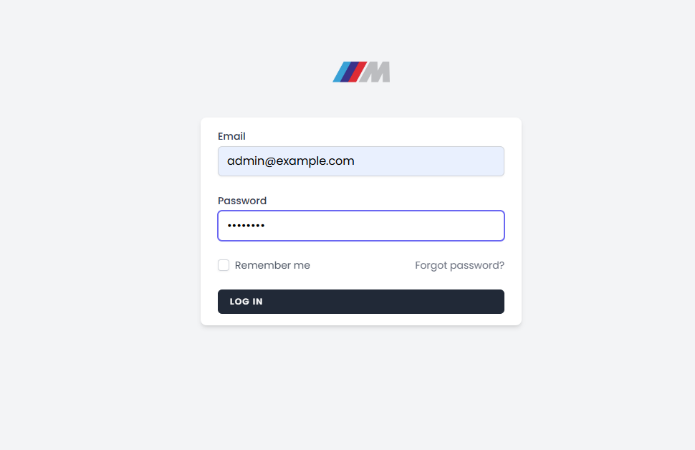


Ha már regisztráltunk a weboldalra, akkor bármikor be tudunk jelentkezni.

**Bejelentkezés**

A bejelentkezési felület egyszerű, csak az email címet és a jelszó megadását kéri.

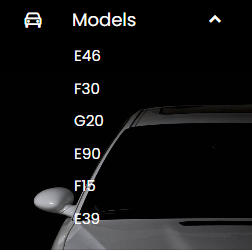
Bejelentkezési felület:



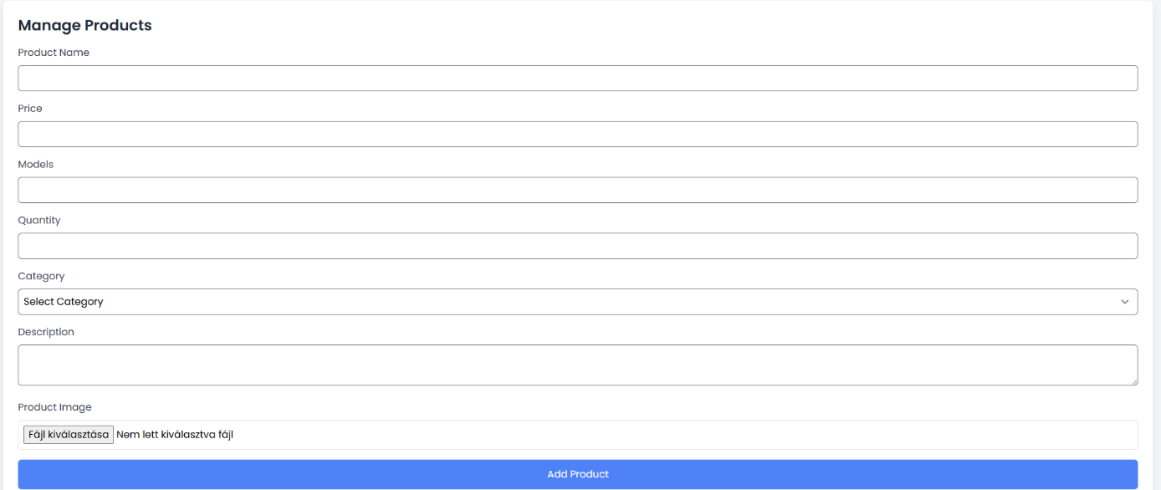
Csak azok férhetnek hozzá a szolgáltatásokhoz vagy a programhoz, akik rendelkeznek érvényes email címmel és jelszóval.

Modellek a weboldalon:

Ezen az oldalon találhatóak a modellek, melyek közül a vásárló illetve a weboldalt böngésző személy válogathat és vásárolhat.



Új termék hozzáadás:



Ezen a fülön a felhasználó új termékeket tud rögzíteni, ahol a program egyből a szerverhez navigál. A fájlokat és az adatok itt adhatók és tölthetők fel.

Itt tud a felhasználó újterméket hozzáadni, feltölteni, ami egyből elmegy a szerverhez. Ide feltudja tölteni a fájlt is, és meg tudja adni az adatokat is.

# **Összefoglalás**

A Car Shop projekt célja egy olyan online autóalkatrész-webshop létrehozása volt, amely kifejezetten BMW-tulajdonosokat céloz meg. A projektet csapatmunkában készítettük, a fejlesztést Kolozsvári Krisztina Janka és Pap László végezte a Gyulai SZC Szigeti Endre Technikum 13/A osztályában.

A fejlesztés során:

* Visual Studio Code volt a fő fejlesztői környezet,
* a backend futtatásához Node.js-t, a frontendhez pedig React-et használtunk,
* az adatbázis-tervezéshez dbdiagram.io-t alkalmaztunk.

A projekt során létrehoztunk:

* felhasználói és adminisztrátori funkciókat (regisztráció, bejelentkezés),
* termékek és modellek kezelésére alkalmas felületeket,
* egy letisztult, modern dizájnt, amely tükrözi a BMW márka arculatát.

A tesztelés során létrehoztunk tesztfelhasználót, és React tesztelési eszközöket használtunk (pl. Jest), hogy ellenőrizzük az alkalmazás helyes működését.

A projekt böngészőalapú, így bármilyen platformról elérhető (pl. Windows, macOS, mobil).

A munka során nagy hangsúlyt fektettünk az együttműködésre, feladatainkat megosztottuk, és közösen oldottuk meg a felmerülő problémákat. Ez a csapatmunka volt a projekt egyik legfontosabb alapja.

# **Irodalomjegyzék**

A projekt megtervezéséhez használt weboldal

<https://www.figma.com>

A React keretrendszer indulási alapok

<https://react.dev/learn>

A Node.js oktatási segédlete

<https://kodbazis.hu/node-js-tutorial>

A React oktatási segédlete gyakorló videókkal

<https://kodbazis.hu/react-kurzus>

W3Schools oldala

<https://www.w3schools.com/>

W3Schools HTML gyakorlata

<https://www.w3schools.com/html/default.asp>

W3Schools CSS gyakorlata

<https://www.w3schools.com/css/default.asp>

W3Schools Bootstrap ismeretek

<https://www.w3schools.com/bootstrap/bootstrap_ver.asp>

Bootstrap

<https://getbootstrap.com/>

A vizsga dokumentáció elkészítéséhez használt segédlet

<https://infojegyzet.hu/webszerkesztes/dokumentacio/>