Kelemen Gábor & Visegrádi Tamás

Kivonat

[Ragadja meg az olvasó figyelmét egy izgalmas kivonattal. Ez többnyire a dokumentum rövid összegzése.

FRAGSTORE

The premier destination for games across all platforms and genres

**Tartalom**

Tartalomjegyzék

[Bevezetés 3](#_Toc219100687)

[1. Felhasználói dokumentáció - első szintű címsor 5](#_Toc219100688)

[1.1. címsor 2 5](#_Toc219100689)

[1.2. címsor 2 5](#_Toc219100690)

[1.2.1. címsor 3 5](#_Toc219100691)

[2. Fejlesztői dokumentáció – első szintű címsor 6](#_Toc219100692)

[2.1. Felhasznált technológiák 6](#_Toc219100693)

[2.1.1. PHP 6](#_Toc219100694)

[2.1.2. HTML 6](#_Toc219100695)

[2.1.3. CSS 6](#_Toc219100696)

[2.1.4. JavaScript 7](#_Toc219100697)

[2.1.5. TypeScript 7](#_Toc219100698)

[2.1.6. JSON 7](#_Toc219100699)

[2.1.7. SQL 7](#_Toc219100700)

[2.1.8. Bash 8](#_Toc219100701)

[2.2. Keretrendszerek 8](#_Toc219100702)

[2.2.1. React 8](#_Toc219100703)

[2.2.2. Angular 8](#_Toc219100704)

[2.2.3. Alpine.js 8](#_Toc219100705)

[2.2.4. p5.js 9](#_Toc219100706)

[2.2.5. Sketch 9](#_Toc219100707)

[2.2.6. GSAP 9](#_Toc219100708)

[2.2.7. Bootstrap 9](#_Toc219100709)

[2.2.8. Tailwind CSS 10](#_Toc219100710)

[2.3. API-k 10](#_Toc219100711)

[2.3.1. Chart.js 10](#_Toc219100712)

[2.3.2. Flags API 10](#_Toc219100713)

[2.3.3. Currency Exchange API 10](#_Toc219100714)

[2.3.4. EmailJS 11](#_Toc219100715)

[2.3.5. RAWG API 11](#_Toc219100716)

[2.4. Cloud service 11](#_Toc219100717)

[2.4.1. Firebase 11](#_Toc219100718)

[2.5. Fejlesztési eszközök 11](#_Toc219100719)

[2.5.1. Visual Studio Code 11](#_Toc219100720)

[2.5.2. Figma 12](#_Toc219100721)

[2.5.3. XAMPP 12](#_Toc219100722)

[2.5.4. GitHub 12](#_Toc219100723)

[2.5.5. Vite 13](#_Toc219100724)

[2.5.6. Discord 13](#_Toc219100725)

[2.6. Adatbázis 13](#_Toc219100726)

[2.6.1. Adatbázis tervezése 13](#_Toc219100727)

[2.6.2. Adatbázisok 14](#_Toc219100728)

[2.6.3. Továbbfejlesztési lehetőség 16](#_Toc219100729)

[3. Felmerült akadályok 16](#_Toc219100730)

[Összefoglalás 20](#_Toc219100731)

[4. Források: 21](#_Toc219100732)

# Bevezetés

A Fragstore webalkalmazás célja digitális termékek – elsősorban videojátékokhoz kapcsolódó gift cardok (például Call of Duty játékokhoz, illetve PlayStation gift cardok) – értékesítése egy modern, átlátható webes felületen. Az oldal egy webshopként működik, ahol a felhasználók böngészhetik a termékeket, kosárba helyezhetik őket, wishlistet hozhatnak létre, valamint regisztráció és bejelentkezés után különböző funkciókat érhetnek el. Bár az oldalon valódi fizetés nem valósul meg, a kosár működése teljes mértékben implementálva van.

A frontend megvalósításához JavaScriptet és annak keretrendszereit használtuk, míg a backend oldalon PHP és SQL technológiákra építettük fel a rendszert. Az alkalmazás egyik kiemelkedő funkciója egy AI chatbot, amely segíti a felhasználókat az oldalon való eligazodásban, valamint a wishlist rendszer, amely lehetővé teszi a kinézett termékek elmentését.

A projektet kétfős csapatban készítettük el. Kelemen Gábor (én) elsősorban az adatbázis tervezéséért és megvalósításáért, a fejlesztői dokumentáció elkészítéséért, az oldal reszponzivitásáért, valamint a tervezési folyamatok nagy részéért voltam felelős. Emellett jellemzően én készítettem el az egyes oldalak alap vázát (frame-jeit). A backend fejlesztésében mindketten részt vettünk: Visegrádi Tamással közösen dolgoztunk a bejelentkezési, regisztrációs és egyéb funkcionális elemek megvalósításán. Az easter egg játékok fejlesztésében kisebb mértékben vettem részt, ezek kidolgozásával főként csapattársam foglalkozott. A végső, igazán dizájnos megjelenés nagyban az ő munkájának köszönhető.

Az ötlet onnan származott, hogy mindketten szeretünk videojátékokkal játszani, és szerettünk volna egy összetettebb, kihívást jelentő projektet, amelyben bizonyíthatjuk a megszerzett tudásunkat. A fejlesztés során mentoraink – Drenkovics Péter, Barnáné Kiss Éva és Juhász Zoltán – folyamatos támogatást és motivációt nyújtottak számunkra, ami nagy segítséget jelentett a projekt megvalósítása során.

A projekt fejlesztése végig Git verziókezelő rendszerrel történt, a forráskód egy publikus GitHub repositoryban érhető el. A repository a https://github.com/VTom21/Fragstore-Webshop címen található meg, ahol a fejlesztési folyamat nyomon követhető, valamint a projekthez kapcsolódó dokumentáció is elérhető.

A Fragstore webalkalmazás fejlesztése során számos modern webes technológiát, keretrendszert és fejlesztői eszközt alkalmaztunk annak érdekében, hogy egy stabil, jól skálázható és felhasználóbarát rendszert hozzunk létre. A backend megvalósításához PHP-t és SQL adatbázist használtunk, amelyek lehetővé tették a felhasználói adatok, termékek, kosár- és wishlist funkciók biztonságos kezelését, valamint az üzleti logika hatékony megvalósítását. A frontend oldalon HTML, CSS és JavaScript alkotta az alapot, amelyet különböző JavaScript-alapú megoldásokkal egészítettünk ki annak érdekében, hogy az oldal dinamikus és interaktív működést biztosítson.

A JavaScript ökoszisztémán belül több különböző megközelítést is alkalmaztunk. A komplexebb, komponensalapú felületek kialakításában React és Angular keretrendszerek segítették a strukturált fejlesztést, míg az egyszerűbb interakciók és kisebb dinamikus elemek megvalósításához Alpine.js bizonyult hatékony megoldásnak. A TypeScript használata hozzájárult a kód minőségének javításához, mivel a típusosság révén csökkentette a futásidejű hibák számát. Az adatok strukturált továbbítását JSON formátum biztosította a frontend és a backend között.

A megjelenés és a felhasználói élmény fejlesztése során különös hangsúlyt fektettünk a reszponzivitásra és a vizuális élményre. Ebben nagy segítséget nyújtott a Bootstrap és a Tailwind CSS, amelyek lehetővé tették a különböző képernyőméretekhez való gyors és hatékony alkalmazkodást. Az animációk és látványos átmenetek megvalósításához GSAP könyvtárat használtunk, míg a kreatívabb, játékosabb elemek és easter egg funkciók fejlesztésében a p5.js játszott fontos szerepet. A felhasználói felület megtervezése és az oldal vizuális koncepciójának kialakítása Figma és Sketch segítségével történt, amelyek elősegítették az egységes dizájn kialakítását már a fejlesztés korai szakaszában.

Az alkalmazás funkcionalitását külső szolgáltatások és API-k is bővítették. A Chart.js lehetőséget biztosított adatok grafikus megjelenítésére, amely elsősorban statisztikai vagy adminisztrációs nézetekben volt hasznos. A Flags API segítségével országokhoz kötődő vizuális elemeket tudtunk megjeleníteni, míg a Currency Exchange API lehetővé tette az árak különböző pénznemek közötti átváltását. Az EmailJS integrálása biztosította az értesítések és kapcsolatfelvételi funkciók egyszerű megvalósítását, külön backend email szerver konfigurálása nélkül.

A fejlesztési folyamat során kiemelt szerepet kaptak a modern fejlesztői eszközök is. A Visual Studio Code biztosította az egységes fejlesztői környezetet, míg a Git és a GitHub lehetővé tették a verziókövetést és a hatékony csapatmunkát. A helyi fejlesztéshez XAMPP környezetet használtunk, amely megkönnyítette a backend tesztelését. A frontend buildelési és fejlesztési folyamatokat Vite gyorsította fel, míg egyes fejlesztési és automatizálási feladatok során Bash parancsokat alkalmaztunk.

Ezen technológiák együttes alkalmazása tette lehetővé, hogy a Fragstore egy összetett, mégis jól átlátható és bővíthető webalkalmazásként valósuljon meg, amely megfelelő alapot biztosít további funkciók és fejlesztések bevezetéséhez.

A Fragstore célja az, hogy

**Normál szöveg Times New Roman 12 pt, sorkizárt, 1,5 sortávolság**

# Felhasználói dokumentáció - első szintű címsor

## címsor 2

szöveg

## A home weboldal

### A navbar, hero és felső banner



#### A hero

A kezdőlap központi eleme a nagy méretű hero szekció, amely azonnal megragadja a látogatók figyelmét. A háttérben egy Xbox kontroller részlete látható, amely vizuálisan is közvetíti a gaming témát és modern, technológiai hangulatot teremt. Főbb elemek: Főcím: A képernyő közepén elhelyezett "UNLOCK THE FUTURE OF GAMING" felirat ciánkék színnel kiemelve, amely azonnal kommunikálja az oldal célját és brandidentitását. A nagybetűs szöveg látványos és figyelemfelkeltő, amely segít felkelteni a felhasználók érdeklődését. Alcím: A főcím alatt egy rövid leírás található: "Discover exclusive deals, rare collectibles, and your next favorite adventure — all in one place." Ez a szöveg tömören összefoglalja az oldal értékajánlatát, és elmagyarázza a felhasználóknak, hogy mit találhatnak az oldalon. CTA gombok: Két jól elkülönülő call-to-action gomb található a hero szekcióban: "Explore Games" - ciánkék kerettel, átlátszó háttérrel "Buy Gift Cards" - arany/sárga kerettel, átlátszó háttérrel Ezek a gombok közvetlen navigációs lehetőséget biztosítanak a felhasználók számára, hogy azonnal elérjék a számukra érdekes tartalmakat anélkül, hogy további menüpontokat kellene keresniük.

#### Navigációs menü

A banner alatt található a fő navigációs sáv, sötét, áttetsző háttérrel, amely lehetővé teszi a háttérkép átsejlését, miközben biztosítja az olvashatóságot.

**Bal oldal**: Az oldal logója (kék színű doboz ikon), amely egyben a főoldalra visszavezető link is.S

**Közép:** A fő menüpontok vízszintesen elrendezve: *Games, Gift, Cards Genres, Contact, About Us*

**Jobb oldal**: Felhasználói műveletek gombjai: "*Log in*" - ciánkék kerettel "*Sign up*" - ciánkék kitöltéssel

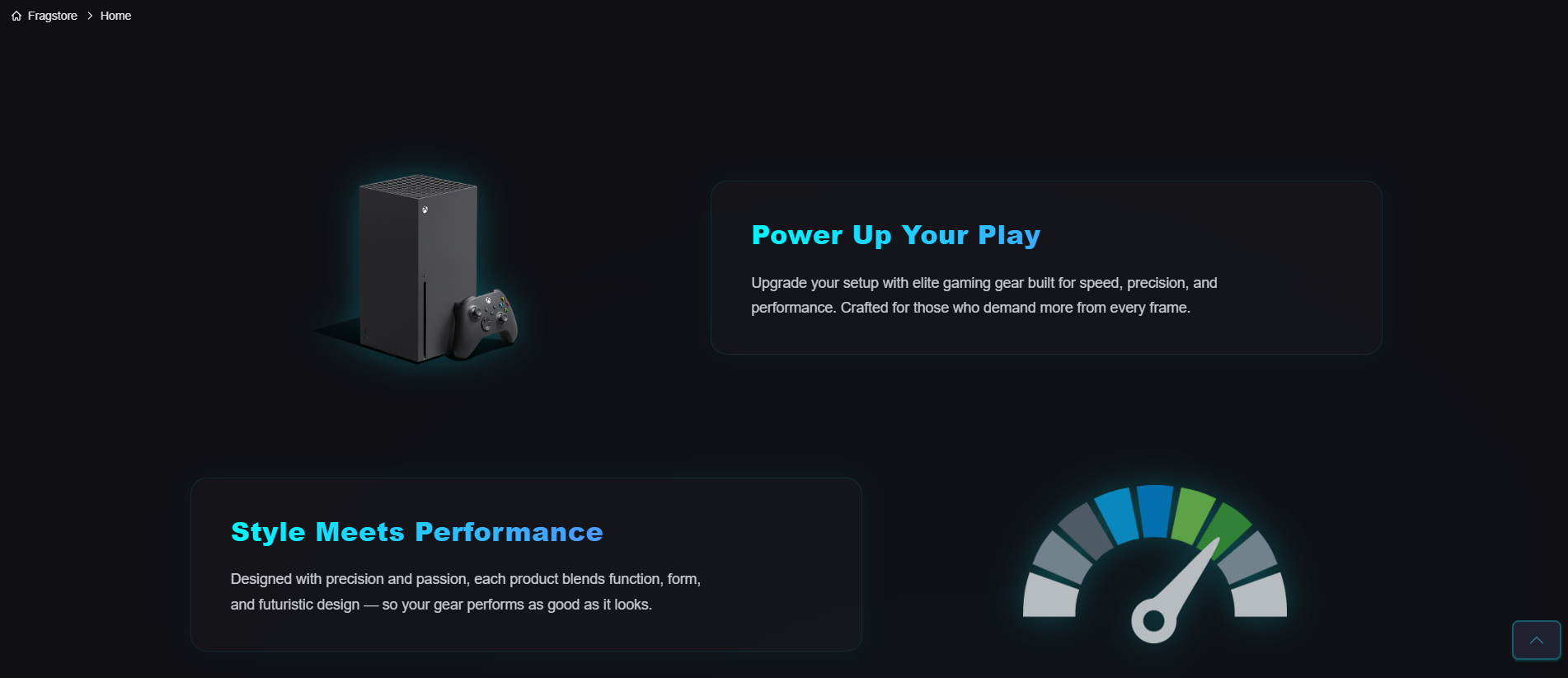
#### Felső banner

Az oldal legtetején egy figyelemfelkeltő banner található sötét háttérrel és piros díszítő elemekkel a széleken. A banner tartalma: "Call of Duty Franchise Sale - Grab the best cod games before they're gone!" Ez az elem időérzékeny akciókra hívja fel a figyelmet, és sürgősséget kelt a látogatókban, ösztönözve őket a gyors vásárlásra.

#### Nyelv választó

A navigációs sáv alatt, a bal felső sarokban található az amerikai zászlós nyelv választó ("English" felirattal), amely lehetővé teszi a felhasználók számára, hogy megváltoztassák az oldal nyelvét. Ez a funkció különösen hasznos nemzetközi közönség esetén, és javítja az oldal hozzáférhetőségét.

### About Us (Rólunk oldal)



#### Navigációs Fejléc (Fixed Position)

Az oldal tetején található navigációs sáv fix pozícióval rendelkezik, ami azt jelenti, hogy görgetés közben is a képernyő tetején marad. Ez a funkció jelentősen javítja a felhasználói élményt, mivel bármikor, az oldal bármely részéről azonnal elérhetők a fő menüpontok anélkül, hogy vissza kellene görgetni a lap tetejére.

**A fix navbar elemei:**

* **Bal oldal:** Oldal logója (kék doboz ikon)
* **Közép:** Fő menüpontok *(Games, Gift Cards, Genres, Contact, About Us)*
* **Jobb oldal:** "*Log in*" és "*Sign up*" gombok ciánkék színezéssel

A navbar sötét, félig átlátszó háttere biztosítja, hogy görgetés közben is jól látható és olvasható maradjon, miközben nem takarja el teljesen a hátteret.

#### Breadcrumb Navigáció

A fix navbar alatt található a breadcrumb (morzsanavigáció), amely megmutatja a felhasználó aktuális helyét az oldal struktúrájában. A példában látható: "Fragstore > Home", amely segít a felhasználóknak megérteni, hogy hol tartózkodnak az oldalon, és egyszerű visszalépési lehetőséget biztosít a korábbi szintekre. A kis ház ikon a kezdőlapra utal, amely egyértelmű vizuális jelzést ad.

#### Elrendezés és Design Elvek

**Szimmetrikus grid rendszer:** Az About Us oldal egy kétosztású grid rendszert használ, ahol a tartalom egyenletesen oszlik meg képi és szöveges elemek között. Az első szekcióban a kép bal oldalt, a második szekcióban jobb oldalt található, ami dinamikus, kiegyensúlyozott megjelenést eredményez.

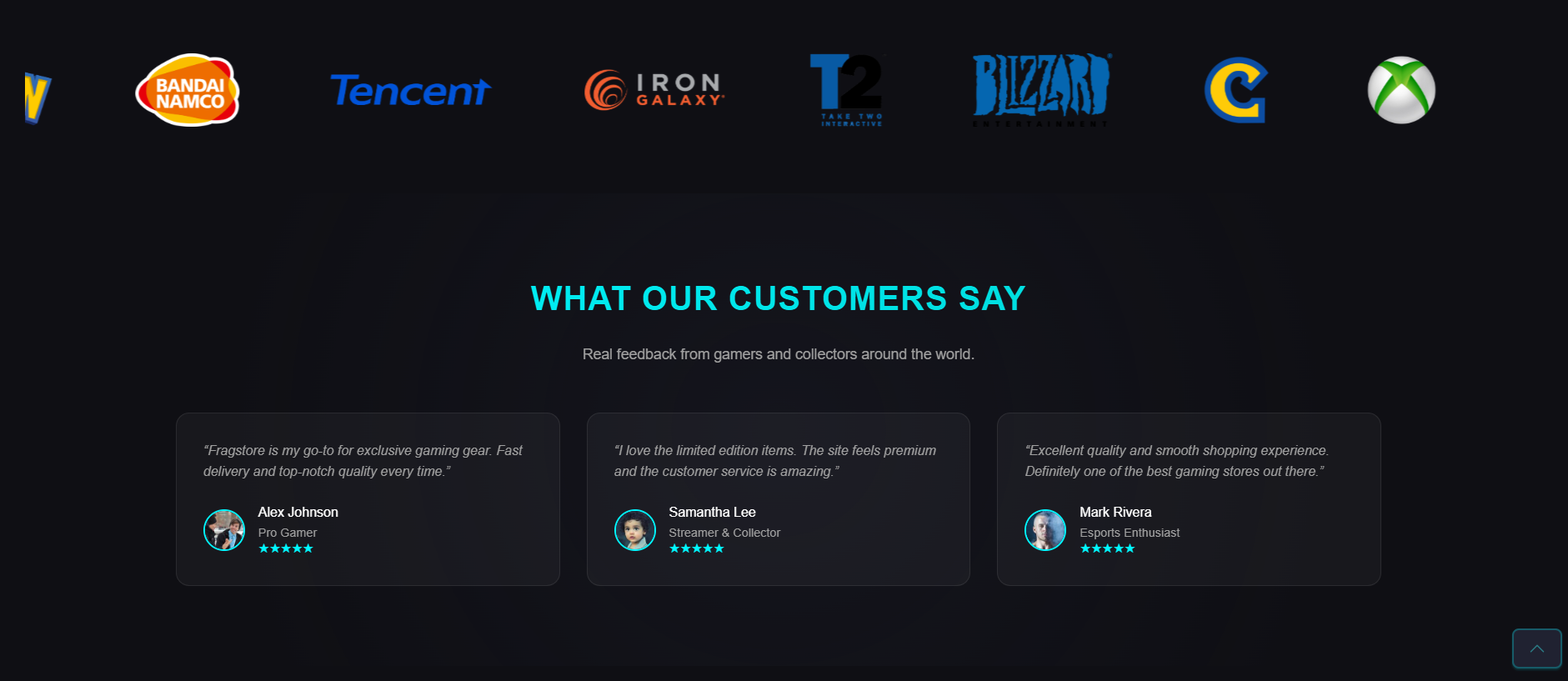
**Dark mode dizájn:** Az oldal következetesen sötét hátteret használ, amely:

* Kiemeli a színes elemeket (ciánkék címek, világos szöveg)
* Gaming-orientált, modern hangulatot teremt
* Csökkenti a szemek terhelését hosszabb böngészés esetén

**Tipográfia:** A címek nagybetűsek és ciánkék színűek, míg a törzsszöveg világosszürke, ami kellő kontrasztot biztosít az olvashatósághoz.

**Scrolling élmény:** A jobb alsó sarokban egy "vissza a tetejére" gomb látható ciánkék színnel, amely megkönnyíti a navigációt hosszabb oldalak esetén.

### ****Logo körhinta és értékelések****



#### Partnercégek Logó Körhintája

Az About Us oldal felső részén található egy automatikusan pergő logó körhinta (carousel), amely a platform partnercégeinek és együttműködő márkáinak logóit jeleníti meg vízszintes sorban. Ez a szekció vizuálisan erősíti az oldal hitelességét és megbízhatóságát azáltal, hogy bemutatja a nagy nevű gaming iparági szereplőkkel való kapcsolatokat.

**Megjelenő logók:**

* **Bandai Namco** - narancssárga-piros kerettel
* **Tencent** - kék márkalogó
* **Iron Galaxy** - narancssárga-szürke logó
* **Take-Two Interactive** - kék T2 logó
* **Blizzard** - kék márkanév logó
* **Capcom** - sárga-kék kerek logó
* **Xbox** - zöld-fehér X logó

**Működés:** A logók automatikusan, folyamatos loop-ban görgetnek végig a képernyőn, ami dinamikus és modern megjelenést biztosít. Ez a megoldás több logó megjelenítését teszi lehetővé korlátozott képernyőhelyen, miközben fenntartja a vizuális egyensúlyt és nem terheli túl az oldalt.

**Dizájn elemek:**

* Sötét, átmenetes háttér, amely finoman beolvad az oldal főhátterébe
* Egyenletes távolság a logók között
* Minden logó eredeti színében és formájában jelenik meg
* A logók mérete egységes, hogy vizuális harmóniát teremtsen

**Célja:** Ez a szekció azonnal kommunikálja a látogatók felé, hogy az oldal hivatalos kapcsolatban áll az iparág vezető szereplőivel, ami növeli a bizalmat és a brand presztízsét. A gaming közösség számára ezek az ikonikus logók felismerhetők és jelentéssel bírnak.

#### Vásárlói Értékelések Szekció

A logó körhinta alatt található a "WHAT OUR CUSTOMERS SAY" (Mit mondanak vásárlóink) című rész, amely közösségi bizonyítékokat (social proof) szolgáltat az oldal megbízhatóságának alátámasztására.

## Szekció felépítése:

**Főcím:** "WHAT OUR CUSTOMERS SAY" - nagybetűs, ciánkék színnel kiemelve, amely következetesen illeszkedik az oldal színsémájába.

**Alcím:** "Real feedback from gamers and collectors around the world." - világosszürke színnel, amely kontextust ad és hangsúlyozza az értékelések hitelességét.

**Értékelési Kártyák:**

A szekció három értékelési kártyát tartalmaz, amelyek vízszintesen, egyenlő távolságra vannak elrendezve. Minden kártya sötétszürke háttérrel rendelkezik, amely finoman kiemelkedik a fekete oldalhátterből.

## Dizájn Elemek:

**Kártya struktúra:**

* Lekerekített sarkok a modern megjelenésért
* Elegendő padding (belső térköz) az olvashatósághoz
* Kurzív idézetek a személyes jelleg hangsúlyozására
* Profilképek kerek alakban a bal alsó sarokban
* Csillagok ciánkék színben, konzisztens a brand identitással

**Elrendezés:** A három kártya egyenlő szélességű és responsive grid rendszerben van elhelyezve, amely kisebb képernyőkön valószínűleg egymás alá rendeződik a jobb olvashatóság érdekében.

**Funkció és Célközönség:**

**Diverzitás:** A három különböző vásárlói profil bemutatása azt közvetíti, hogy az oldal széles közönséget szolgál ki - a professzionális játékosoktól a streameren át az esports rajongókig.

**Hitelesség:** A konkrét nevek, profilképek és szerepkörök használata hitelesebbé teszi az értékeléseket, mint az általános, névtelen visszajelzések.

# Fejlesztői dokumentáció – első szintű címsor

## Felhasznált technológiák

### PHP

A PHP (Hypertext Preprocessor) egy szerveroldali szkriptnyelv, amelyet elsősorban dinamikus weboldalak és webalkalmazások fejlesztésére használnak. A PHP lehetővé teszi a szerver és a kliens közötti kommunikáció kezelését, valamint az adatbázisokkal való hatékony kapcsolattartást. A nyelv segítségével a fejlesztők dinamikusan generált tartalmakat hozhatnak létre, feldolgozhatják a felhasználói adatokat, valamint megvalósíthatják az üzleti logikát. A Fragstore projekt esetében a PHP kulcsszerepet játszott a backend működésében, például a bejelentkezési és regisztrációs folyamatok, a kosár és a wishlist funkciók, valamint az adatbázis-kezelés megvalósításában.

### HTML

A HTML (Hypertext Markup Language) a weboldalak szerkezetének kialakítására szolgáló jelölőnyelv. Segítségével határozható meg egy weboldal tartalma és annak hierarchikus felépítése, például szövegek, képek, űrlapok és egyéb elemek elhelyezése. A HTML a webfejlesztés alapját képezi, mivel minden weboldal erre épül. A Fragstore fejlesztése során a HTML biztosította az oldalak logikus és átlátható struktúráját, amelyre a további technológiák – például a CSS és a JavaScript – hatékonyan tudtak épülni.

### CSS

A CSS (Cascading Style Sheets) egy stíluslapnyelv, amely a weboldalak megjelenésének és vizuális stílusának meghatározására szolgál. A CSS lehetővé teszi a tartalom és a megjelenés elkülönítését, így a HTML kizárólag a szerkezetért, míg a CSS a formázásért felel. A CSS segítségével szabályozható az oldalak színvilága, betűtípusai, elrendezése és reszponzív viselkedése. A Fragstore esetében a CSS nagyban hozzájárult az esztétikus megjelenéshez és a felhasználói élmény javításához, különösen a modern, letisztult dizájn kialakításában.

### JavaScript

A JavaScript egy széles körben elterjedt programozási nyelv, amelyet elsősorban weboldalak és webalkalmazások interaktív funkcióinak megvalósítására használnak. A JavaScript jellemzően kliensoldalon fut, vagyis a felhasználó böngészőjében kerül feldolgozásra. Segítségével dinamikus tartalmak, eseménykezelések és felhasználói interakciók hozhatók létre. A Fragstore projektben a JavaScript biztosította az oldal élő működését, például a kosár frissítését, a wishlist kezelését, valamint az interaktív felhasználói elemek megvalósítását.

### TypeScript

A TypeScript a JavaScript egy kibővített változata, amely statikus típusosságot vezet be a nyelvbe. Ez lehetővé teszi a fejlesztők számára, hogy már a fejlesztési folyamat során felismerjék az esetleges hibákat, ezáltal növelve a kód megbízhatóságát és karbantarthatóságát. A TypeScript különösen hasznos nagyobb, összetettebb projektek esetén, ahol a kód átláthatósága kiemelt szempont. A Fragstore fejlesztése során a TypeScript hozzájárult a frontend kód minőségének javításához és a hosszú távon fenntartható kódbázis kialakításához.

### JSON

A JSON (JavaScript Object Notation) egy könnyen olvasható és strukturált adatformátum, amelyet leggyakrabban adatcserére használnak különböző rendszerek között. A JSON formátum egyszerűsége és rugalmassága miatt ideális a frontend és a backend közötti kommunikációhoz. A Fragstore alkalmazásban a JSON biztosította az adatok hatékony továbbítását, például a termékadatok, felhasználói információk és kosár tartalmának kezelésénél.

### SQL

Az SQL (Structured Query Language) egy adatbázis-kezelő nyelv, amely relációs adatbázisok kezelésére szolgál. Segítségével létrehozhatók, módosíthatók és lekérdezhetők az adatbázisban tárolt adatok. A Fragstore projektben az SQL kulcsszerepet játszott az adatbázis struktúrájának kialakításában, valamint a felhasználók, termékek, rendelések és wishlist adatok biztonságos és hatékony kezelésében.

### Bash

A Bash egy parancssori értelmező és szkriptnyelv, amelyet elsősorban fejlesztési és automatizálási feladatok során használnak. Lehetővé teszi különböző parancsok és folyamatok gyors végrehajtását, például buildelési, fájlkezelési vagy verziókezelési műveletek esetén. A Fragstore fejlesztése során a Bash hozzájárult a fejlesztési folyamatok egyszerűsítéséhez és hatékonyabbá tételéhez.

## Keretrendszerek

### React

A React egy JavaScript alapú felhasználói felület fejlesztésére szolgáló könyvtár, amelyet elsősorban dinamikus és komponensalapú webalkalmazások készítéséhez használnak. A React lehetővé teszi a felhasználói felület kisebb, újrahasznosítható komponensekre bontását, ami átláthatóbbá és könnyebben karbantarthatóvá teszi a kódot. A Fragstore projekt során a React alkalmazása hozzájárult a gyorsabb felületfrissítésekhez és az interaktív elemek hatékony kezeléséhez, ezáltal javítva a felhasználói élményt.

### Angular

Az Angular egy teljes körű, TypeScript alapú frontend keretrendszer, amelyet komplex webalkalmazások fejlesztésére használnak. Strukturált felépítése és beépített megoldásai – például adatkezelés és komponenskommunikáció – megkönnyítik a nagyobb alkalmazások fejlesztését. A Fragstore esetében az Angular használata segítette az alkalmazás logikus felépítését és a kód egységes kezelését, különösen az összetettebb felhasználói felületi funkciók esetén.

### Alpine.js

Az Alpine.js egy könnyű JavaScript keretrendszer, amelyet egyszerűbb interaktív funkciók megvalósítására terveztek. Elsősorban kisebb dinamikus elemek kezelésére alkalmas, ahol egy teljes keretrendszer használata túlzott lenne. A Fragstore projektben az Alpine.js hozzájárult az oldal gyorsabb betöltéséhez és az egyszerűbb felhasználói interakciók hatékony megvalósításához.

### p5.js

A p5.js egy JavaScript alapú grafikus könyvtár, amelyet kreatív kódoláshoz és vizuális elemek megvalósításához használnak. Különösen alkalmas animációk, interaktív grafikai megoldások és kisebb játékok fejlesztésére. A Fragstore alkalmazásban a p5.js az easter egg játékok és vizuálisan különleges elemek megvalósításában kapott szerepet, növelve az oldal egyediségét és élményszerűségét.

### Sketch

A Sketch egy felhasználói felületek tervezésére szolgáló grafikus eszköz, amelyet elsősorban webes és mobilalkalmazások dizájnjának megalkotására használnak. A Sketch segítségével pontosan megtervezhetők az oldalak elrendezései, komponensei és vizuális elemei. A Fragstore tervezési folyamatában a Sketch hozzájárult az egységes megjelenés kialakításához és a fejlesztési munka előkészítéséhez.

### GSAP

A GSAP (GreenSock Animation Platform) egy nagy teljesítményű JavaScript animációs könyvtár, amelyet komplex és látványos animációk létrehozására használnak. A GSAP pontos időzítést és zökkenőmentes animációkat biztosít, amelyek jelentősen javítják a felhasználói élményt. A Fragstore projekt során a GSAP segítségével valósultak meg az oldal vizuális átmenetei és animált elemei.

### Bootstrap

A Bootstrap egy széles körben elterjedt CSS keretrendszer, amely előre definiált komponenseket és rácsrendszert biztosít a reszponzív weboldalak gyors elkészítéséhez. A Bootstrap használata megkönnyíti a különböző képernyőméretekhez való alkalmazkodást. A Fragstore fejlesztése során a Bootstrap hozzájárult a gyorsabb frontend fejlesztéshez és az oldal reszponzív működéséhez.

### Tailwind CSS

A Tailwind CSS egy modern, utility-first szemléletű CSS keretrendszer, amely lehetővé teszi az egyedi dizájnok gyors és rugalmas kialakítását. A Tailwind segítségével a stílusok közvetlenül a HTML elemekhez rendelhetők, ami nagyfokú testreszabhatóságot biztosít. A Fragstore projektben a Tailwind CSS alkalmazása elősegítette a letisztult, modern megjelenés kialakítását.

## API-k

### Chart.js

A Chart.js egy JavaScript könyvtár, amely egyszerű és testreszabható grafikonok készítését teszi lehetővé weboldalakon és webalkalmazásokban. Segítségével vizuálisan átlátható módon lehet megjeleníteni adatokat, például statisztikákat vagy elemzéseket. A Fragstore projektben a Chart.js főként adminisztrációs nézetekben és statisztikai elemek megjelenítésében játszott szerepet, lehetővé téve a termékek és felhasználói aktivitás vizuális értékelését.

### Flags API

A Flags API egy külső szolgáltatás, amely országok zászlóinak megjelenítésére alkalmas. Segítségével egyszerűen lehet vizuális információt nyújtani a felhasználóknak, például nemzetközi termékekhez vagy felhasználói beállításokhoz kapcsolódóan. A Fragstore alkalmazásban a Flags API hozzájárult a felhasználói élmény gazdagításához és a vizuális tájékozódás megkönnyítéséhez.

### Currency Exchange API

A Currency Exchange API lehetővé teszi az árak különböző pénznemek közötti automatikus átváltását. Ez különösen hasznos nemzetközi felhasználók vagy több pénznemű árlisták esetén, mivel valós idejű és pontos átváltást biztosít. A Fragstore projektben a Currency Exchange API hozzájárult a nemzetközi felhasználók kiszolgálásához, valamint a termékek árainak rugalmas és naprakész megjelenítéséhez.

### EmailJS

Az EmailJS egy külső szolgáltatás, amely lehetővé teszi e-mailek küldését közvetlenül a frontendből, anélkül, hogy a fejlesztőnek külön szerveroldali e-mail kezelést kellene beállítania. A Fragstore projektben az EmailJS segítségével valósult meg a kapcsolatfelvételi és értesítési funkciók, például a felhasználói kérdések vagy visszajelzések egyszerű kezelése, anélkül hogy külön backend e-mail szervert kellett volna konfigurálni.

### RAWG API

A RAWG API egy nyilvános videojáték-adatbázisra épülő programozói felület, amely részletes információkat biztosít különböző videojátékokról. Az API segítségével lekérdezhetők többek között játéknevek, megjelenési dátumok, platformok, műfajok és egyéb, a játékokhoz kapcsolódó adatok. A RAWG API célja, hogy strukturált és könnyen feldolgozható formában biztosítson naprakész adatokat a videojátékok világából.

## Cloud service

### Firebase

A Firebase egy Google által fejlesztett felhőalapú platform, amely különböző backend szolgáltatásokat biztosít webes és mobilalkalmazások számára. A Firebase célja, hogy leegyszerűsítse az alkalmazások fejlesztését és üzemeltetését azáltal, hogy kész megoldásokat kínál többek között adatkezelésre, hitelesítésre és valós idejű kommunikációra. A szolgáltatás felhőalapú működése lehetővé teszi az adatok biztonságos tárolását és elérését anélkül, hogy a fejlesztőknek saját szerverinfrastruktúrát kellene fenntartaniuk.

## Fejlesztési eszközök

### Visual Studio Code

A Visual Studio Code egy könnyű, mégis nagy teljesítményű forráskódszerkesztő, amelyet széles körben használnak webes és szoftverfejlesztési projektek során. A program számos beépített és bővíthető funkciót kínál, mint például szintaxiskiemelés, automatikus kódkiegészítés, hibajelzések és integrált terminál. A Visual Studio Code lehetővé teszi a különböző programozási nyelvek – például PHP, JavaScript, TypeScript, HTML és CSS – hatékony kezelését egyetlen fejlesztői környezeten belül. A Fragstore projekt fejlesztése során ez az eszköz biztosította az egységes munkakörnyezetet, amely jelentősen hozzájárult a hatékonyabb kódoláshoz és a hibák gyorsabb felismeréséhez.

### Figma

A Figma egy modern, felhőalapú tervezőeszköz, amelyet elsősorban felhasználói felületek és weboldalak vizuális megtervezésére használjuk. A Figma lehetővé teszi a dizájnok valós idejű és közös szerkesztését, így különösen alkalmas csapatmunkára. Az eszköz segítségével már a fejlesztés korai szakaszában elkészíthetők a drótvázlatok (wireframe-ek), prototípusok és végleges látványtervek. A Fragstore esetében a Figma hozzájárult ahhoz, hogy az oldal egységes megjelenéssel, átgondolt, struktúrával és felhasználóbarát dizájnnal valósuljon meg.

### XAMPP

A XAMPP egy ingyenes, nyílt forráskódú szervercsomag, amely helyi fejlesztési környezetet biztosít webalkalmazások készítéséhez és teszteléséhez. A csomag tartalmazza az Apache webszervert, a MySQL/MariaDB adatbáziskezelőt, valamint a PHP futtatókörnyezetet. A XAMPP használata lehetővé teszi, hogy a fejlesztők internetkapcsolat nélkül, lokális környezetben teszteljék az alkalmazás backend működését. A Fragstore projektben a XAMPP kulcsfontosságú szerepet játszott a PHP-alapú backend és az adatbázis funkciók fejlesztésében és ellenőrzésében.

### GitHub

A GitHub egy verziókezelő és együttműködési platform, amely a Git rendszerre épül. Lehetővé teszi a forráskód változásainak nyomon követését, a fejlesztési folyamat dokumentálását, valamint a csapattagok közötti hatékony együttműködést. A GitHub használatával minden módosítás visszakövethetővé válik, így könnyen kezelhetők a hibák és a fejlesztési irányok. A Fragstore projekt teljes fejlesztési folyamata GitHub segítségével zajlott, ahol a publikus repository biztosítja az átláthatóságot és a projekt hosszú távú fenntarthatóságát.

### Vite

A Vite egy modern frontend build eszköz és fejlesztői szerver, amely a gyors betöltési időre és hatékony fejlesztési élményre helyezi a hangsúlyt. A hagyományos build eszközökkel szemben a Vite azonnali fejlesztői környezetet biztosít, így a változtatások szinte azonnal megjelennek a böngészőben. A Vite különösen jól együttműködik modern JavaScript és TypeScript alapú projektekkel, valamint keretrendszerekkel, mint például a React. A Fragstore fejlesztése során a Vite jelentősen felgyorsította a frontend fejlesztési folyamatot és hozzájárult a hatékonyabb munkavégzéshez.

### Discord

A Discord egy valós idejű kommunikációs platform, amelyet széles körben használnak fejlesztői csapatok együttműködésre és projektkommunikációra. A szolgáltatás lehetőséget biztosít szöveges és hangalapú kommunikációra, valamint fájlok és kódrészletek megosztására, ami különösen hasznos a közös fejlesztési munka során. A Discord felülete lehetővé teszi különböző csatornák létrehozását, amelyek segítik a feladatok, hibák és ötletek elkülönített kezelését.

A Fragstore projekt fejlesztése során a Discord szolgált a csapaton belüli kommunikáció elsődleges eszközeként. Segítségével a csapattagok folyamatos kapcsolatban tudtak maradni, gyorsan egyeztetni a fejlesztési irányokról, valamint hatékonyan megoldani a felmerülő problémákat. A Discord használata jelentősen hozzájárult a gördülékeny csapatmunkához és a fejlesztési folyamat szervezett lebonyolításához.

## Adatbázis

### Adatbázis tervezése

### Adatbázisok

#### videogames

##### datas

Egyedek:

* id int primary key (A játéknak az egyedi azonosítója)
* game\_pic varchar (A játék borító képe a kártya felső részében previewnak szolgál)
* name varchar (A játék neve)
* release\_date date (A játék kiadatásidátuma)
* genre varchar (A játék műfaja)
* platform varchar (A játékhoz tartozó platform vagy platformok)
* price decimal (A játék ára)
* publisher\_id int (A kiadónak az egyedi azonosítja egy külső kulcs)
* isDiscount tinyint kb olyan, mint egy boolen (Ellenőrzi, hogy a játék le e van akciózva vagy sem)

##### awards

* award\_id int primary key (A fejlesztő stúdióhoz tartozó díjak egyedi azonosítója)
* award\_name varchar (A díjnak a neve)
* award\_year int (A díj átadásának éve)

##### developers

* developer\_id int primary key (A fejlesztő egyedi kulcsa)
* developer name (A fejlesztő csapat neve)

##### publishers

* publisher\_id int primary key (A kiadó egyedi azonosítója)
* publisher\_name (A kiadó neve)

##### publisher\_awards (kapcsoló tábla)

* publisher\_id int idegen kulcs (A kiadó egyedi kulcsa)
* award\_id int idegen kulcs (A díj egyedi kulcsa)
* Ezzel fel tudjuk bontani a kiadó és a díj közötti N:M-es kapcsolatot

##### developer\_awards (kapcsoló tábla)

* developer\_id int idegen kulcs (A fejlesztő egyedi kulcsa)
* award\_id idegen kulcs (A díj egyedi kulcsa)
* Ezzel tudjuk fel bontani a fejlesztő és a díjak közötti N:M-es kapcsolatot

##### Kapcsolatok

* A videogames awards és a publisher táblák között N:M-es kapcsolat van. Mert egy kiadóhoz több díj is tartozhat és egy díj több kiadóhoz is tartozhat
* videogames awards és videogames developer között N:M-es kapcsolat van. Hiszen egy fejlesztőhöz több award is tartozhat és a kiadó miatt egy díj több fejlesztőhöz is tartozhat.
* A videogames publisher és a videgames developers között 1:N-es kapcsolat van hiszen egy cégen belül több fejlesztő is dolgozhat, de egy fejlesztő csak egy cégnél dolgozhat.

#### Users

##### Users

* id bigint primary key (A felhasználó egyedi azonosítója)
* email varchar (A felhasználó emailje)
* username varchar (Felhasználó név)
* password\_hash varchar (A felhasználó jelszava)
* crated\_at (A felhasznalói fiók létrehozásának ideje)

#### Giftcard

##### giftcard

* Cardid varchar primary key (A ajándékkártya egyedi azonosítója)
* Name varchar (Az ajándékkártyán szereplő cég neve pl: netflix)
* Img (Az ajándékkártyán szereplő cég logójáról kép)
* Price decimal (Az ajándékkártya összege)
* Region varchar (Az ajándékkártya régiója pl. kontinensekre lebontva)

#### Leaderboard

##### Datas

* id int primary key (A statisztika egyedi azonosítója)
* name (A leaderboardon szereplő név)
* score int (A leaderboardon szereplő eredmény)

### Továbbfejlesztési lehetőség

* A publisher és developer között lehetne egy az N:M-es kapcsolatot, hiszen néhány esetben egy fejlesztő dolgozhat több cégnél is
* A publishert bele lehetne kötni a videogames datas-ba egy idegen kulccsal ezzel a játék adatai között lenének a fejlesztő/fejlesztők ez, így N.M-es kapcsolat lehetne

# Felmerült akadályok

* Videogames adatbázis datas táblája tartalmaz többértékű mezőket, amiket több kód átírás ás adatbázis szerkesztés során sem sikerült külön táblákba szedni illetve biztosítani a weboldal megfelelő működését. Ezt sajnos nem tudtuk meg valósítani, mert ehhez az egész adatbázist újra kellett volna dolgozni és minden adatot az új séma szerint feltölteni a projekt akkori állapotában ez már nem volt sajnos megvalósítható.
* Reszponzivitási problémák. Ezekre jó példa a navbar problémák vagy esetleg kartyák kiíratása kisebb képernyőn (például.: telefon vagy táblagépes nézet). Ezeket CSS-sen belül media quaryket használtunk. Ezeket maximum szelesség alapján néztük és ez alapján osztottuk fel. Ezekre jó példa, hogy a telefonos nézetet 400px-eltől számoltuk és onnan volt reszponzív. A mini gameken belül ezek nehezebbek voltak, mert ott a map és minden megjelenítése javascripten belül történt, így azon belül kellett ifekkel és más módszerekkel eltávolítani vagy átalakítani elemeket.
* A pacman minigame-ben kilehet menni a map két oldalán de nincs le lockolva az x tengelyen (y tengelyen is lehet mozogni, ez azt eredményezni hogy a player ki tud menni a map határain kívűl). Ezt azzal oldottuk meg, hogy a mapon kívül, ha kiment a player a tunelen akkor lelockoltuk a mozgását az y-on tengelyen addig ameddig bennem ért a map másik oldalára.
* A szellemek AI – ja úgy van kivitelezve, hogy képtelenek a kezdő helyükről (dobozukról) kijönni. Megoldás az lenne, hogy a kiinduló irányok választásánál minden szellemet felfele toljuk. Eredetileg a mozgás random volt és, hogy ha az első mozgásuk nem felfele volt akkor egy végtelen körforgásba kerültek és köröztek a kezdő helyen belül, így minden ghost első mozgását felre állítottuk, így nem tudtak bebuggolni és sikeresen kijutottak a kezdő helyről.
* A checkoutnál nem történik tényleges kártyás / paypalos tranzakció (API)
* A pac man egyik powerupja – A teleport nem működik egyelőre megfelelően, mert képes blokkokba teleportálni a játékost. Ezt a játék frissítési funkciójában egy iffel avagy egy elágazással tudtuk megoldani, hiszen a teleport függvény az egész mapot nézte, így azokat a elemeket is ahova más elem volt rakva. Így az elágazással ellenőriztük, hogy a hely ahova teleportálja a Pac Mant vagy üres mező vagy pedig pellet avagy a pályán lévő kis pontok.
* A snake game-ben a kígyó (player) képes saját magán keresztül menni, de nem hal meg. Ezt egy collision logic avagy egy ütközési logikával ellenőrizzük. Ezen belül bekérjük a kígyó testét és megnézzük, hogy a fej avagy amit irányítunk nincs ugyan azon a mezőn amin már van a kígyónak az egyik testrésze, ha már ott van akkor meghívunk egy freeze függvényt ami megállítja a játékot, visszaállítja a kígyó eredeti méretét és eredeti helyét, megjeleníti a game over menüt és leállítja a játék folyamatokat. A gameover menün, ha megnyomjuk a Play again gombot akkor meghívja a Restart függvényt ezzel újra indítjuk a játékot.

* A felvehető kaja hitboxa és mérete révén, a pálya margójára is generálódhat. Ezt végül abban a függvényben oldottuk meg ami az étel megevését nézi és abban az étel random kordinátára helyezzük el, ehhez kell 2 változó a foodX és foodY ezek a kettő kordináta érték. Az X kordniátát a Math.random (Add egy random decimális számot 0 és 1 között) függvényt beszorozzuk a szélesség és a hitbox különbségével ezzel garantáljuk ne tudjon a kifolyni a mapról az X kordinátán az étel. At Y kordonítani is hasonlóan generálódik viszont itt aza különbség, hogy a szélesség helyet a magasság van, így a képlet (0 és 1 közötti szám) szorozva magasság és hitbox különbsége.

* A kígyó alap irányításai tükrözve vannak. Ez a hiba javítása egyszerű volt. Azért mert van egy controls függvény amely mindig fut és egy switchel ellenőrzi, hogy a játékos meg e nyom egy nyilat és ha igen akkor az alapján változtatja meg a kígyó sebességét abba az irányba itt van X és Y változó és ezek változtatásával indul el abba az irányba. Például, ha fel szeretnék menni akkor az X-en a sebesség 0 és Y-on -1 itt jön fel a probléma, hiszen eredetileg azt hittük 1 lesz itt a jó érték csak utólag jöttünk rá, hogy lentről lefelé nézi a játék az irányítást nem fordítva ezért kellett az értéket is megfordítani

* A High Score System nem megfelelően működik. A score system alapján oldottuk meg, amely szerint, ha a player megeszik egy ételt akkor lefut egy függvény amely megnöveli eggyel a scoret és ugyan ezen a függvényen belül egy elágazás ellenőrzi, hogy a mostani score nagyobb, mint az eddig vagy előző high score és ha igen akkor a mostani score lesz a high score és, hogy ez újrafrissítés után is megmaradjon local storageban eltároljuk ezt a változót. Majd a game canvasban ugyan ezt a változót megjelenítjük a high scorenak fenntartott  elemben.

* A snake játék border és food collision logikájának megírása. Ezt egy elágazással nézzük amelyben nézzük, hogy a food X és Y kordinátája egyenlő a snake X és Y kordinátájával. Azonban ezzel az volt a probléma, így önmagával, hogy a collison logika működött, de a food a játék borderébe is tudott újra spawnolni, így lehetetlen volt felszedni. Ezért a food kapott egy hitboxot amellyel  megtudtuk azt oldani, hogy a canvas határain belül tudjuk tartani és fixen egy cellába spawnoljon az étel és, hogy beleférjen a cellába az étel valamennyivel kisebb, mint maga a cella.

* A játékok custom assetjeinek betöltése különféle keretrendszereken keresztül. A snake játékhoz egy [p5.js](http://p5.js) keretrendszert használunk aminek egyedi előre megírt függvényei vannak ezzel megkönyíti egy játék fejlesztését ilyenek (pl.: Az update ami framenként frissíti a játékot: show is ami jelen esetben nekünk a kígyó testének kirajzolásában segít a frissítés után ezzel lesz látható az új testrész). A show függvény fontos az egész játékban nem csak a kígyó testét hanem az ételek megjelenítését is ez a függvéyn kezeli. Ezen a függvényen belül meghívtunk egy szintén [p5.js](http://p5.js)-es függvényt a creataImg függvényt amely a kép generálásért felelős. Itt rajzolódik ki a kígyónak a teste az assetsekből kiszedett képből ugyanígy generálódik az étel avagy az alma képe is. Majd egy for looplal végigmegyünk a kígyó testén és annyiszor fog lefutni amilyen hosszú a kígyó, így tudja legenerálni az összes testrészt a kígyónak.

* A kígyó alapvetően nyilakkal mozog, viszont tegyük fel ha felfele tart és megnyomjuk a lefele nyilat, beleütközik saját testébe. Ezt szintén a controls függvényben oldottuk meg a switchen belül amikor ellenőrizzük, hogy ha a lenyomott nyíl előtti nyíl vele ellentétes (pl.: felső nyíl majd alsó nyíl) akkor ezt kiszűrjük egy elágazással ezt nem egyelővel ellenőrizzük már az előbb említett X és Y sebességek alapján (pl.: ha a lenyomott nyíl az a fel gomb akkor ellenőrizzük, hogy a kígyó előző sebessége az Y-on tengelyen ne legyen 1 avagy lefelé megy a kígyó, ha ez igaz akkor az irány változtatás megtörténik, ha nem avagy az irány ellenkező a mostanival akkor átlép a másik elágazás részbe ami megszakítja az irányváltoztatást).
* A tetrominók alakzatainak eltárolása, azok x és y tengelybeli kezdeti értékének definiálása, illetve az alakzathoz tartozó színek megadása.
* A játék elindítása gombnyomásra (space bar)
* Egy fő függvény létrehozása amit frame-en ként meghívunk.
* Tetrominó alakzatok kitöltése a maguk színjével, majd kiírtatása a canvas képernyőjére. Ez egy Tetromino.tsx-ben történik. Itt vannak előre meghatározott mátrixokban alakzatok, innen kapja meg a random alakzatot a program. Egy listában vannak az előre meghatározott színek. Végül ezt átadja a tetris.tsxnek ahol beimportáljuk és ott adjuk át a canvasnak.
* A sidebar elkészítése a UI – on, majd kirajzolni a megfelelő színnel a következő alakzatot és meghatározni annak pozícióját a sidebaron belül.
* A tetromino collision logika megcsinálása a grid jobb és bal oldalán, az alján. Ugyanígy a már letett tetrominók esetében.
* A collision logika egyszerűsítése. Külön függvények létrehozása a jobb, bal oldali és a talaj collision-höz. Ezt végül egy újra gondolt logikával oldottuk meg. Egy ütközés avagy Collision ellenőrzővel oldottuk meg amelyet végül több fajta függvényben hívtuk meg. Erre jó példa a BottomCollision amely az alsó ütközést nézte. Ezen meghívunk 3 paramétert, az első az alakzat illetve 2 offset az x és az y. Itt az y az egy mivel nulla az a játék elején a keret alja a játék közben a sor amire érkezik. X meg 0 lesz, mert ezt itt nem nézzük. Ezeket majd még 2 különböző függvény nézi avagy a bal és a jobb oldali ellenőrzés.
* Egy megtöltött sor esetében az adott sor törlése. Újabb felmerült probléma, hogy egyszerre nem tudunk több telített sort eltávolítani. Ehhez végül egy ciklus kellett amely lefutáskor ellenőrzi, hogy ha megtel egy sor azt kitörli és mivel egy ciklus végig fut minden soron és addig nem megy tovább a játék ameddig miden megtelt sort ki nem törölte és a felette lévő sorokat lejebb vitte, így nem maradtak üres sorok.
* A tetromino alakzatok forgatása matematikai mátrix felhasználásával. Ezt érvényesíteni minden fokra. Erre jó példa az a logika amely nem hagyja, hogy az alakzatok kibuggoljanak a jaték canvasán kívülre ezt úgy csinálja meg, hogy az alakzat mátrixának a mérete nem változik az elforgása közben.
* A Controls fejlesztése: Meg kell vizsgálni a jobb és bal széleket, hisz amikor forgatás történik, az adott alakzat jobb és bal határai is változnak. Ugyanezt a logikát át kell ültetni a talaj collision logikára is.
* A discount logika átalakítása, ugyanis jelenleg a játékok leárazása úgy történik, hogy egy adott játék véletlenszerűen (5 – 90%) között kap egy leárazást és ezt az értéket meg is tartja (nem reális megoldás). Ezt egy jobb tovább fejlesztett logikával oldottuk meg. Amely most már nem csak random kiválasztott egy játékot, hanem most már a fejlesztő vagy a store adminja tudja meghatározni a leárazás mértékét, így például egy új játék nem 90%-kal lesz leárazva hanem csak 10-el.
* A wish list termékeinek eltárolása az oldal ujratöltése után. A wish list szív ikon megfelelő toggle logikájának kivitelezése. Ezt egyszerűen a local storagel lett eltárolva, így nem az oldaltól függött a tárolás hanem a felhasználó gépében tárolta ezt az adatot. A szív ikon egy svg volt javascripten belül a toggle logikán belül, ha bevolt nyomva az a gomb akkor lefutott egy elágazás ami megnézte a kettő közül melyik állapotban van bevan kapcsolva vagy pedig nincs, ha nem volt akkor az svg színét egy string változóban megtudtuk változtatni és visszaadni a html/php-nek. Fordítva visszaszínezte az eredeti színére az ikont.
* A filter rendszer megfelelő működése a learázott játékok esetében (ABC, ár szerinti rendezés, illetve ár range). Ezeket végül finomítottabb algoritmusokkal oldottuk meg például egy csúszka a kiadási évnek, így lehetet év szerint keresni a csúszkán szereplő adatot meg könnyen visszatudtuk adni egy javascript fájlnak. Az árat egy árrés alapján oldottuk meg volt egy minimális érték és egy maximális (pl.: 20$ minimumnak és maximumnak 50$) ezt szintén visszaadtuk a javascriptnek és onnan egy metódus megoldotta a keresést.
* A live stock feature működtetése firebase cloud service felhasználásával
* A shopping cart adatainak átvitele az URL – en keresztül a tranzakciós oldalakra
* Az alap ár (amerikai dollárban megadva) átváltása más pénznemekbe, azok igazítása a filterekhez, illetve azok megfelelő felvétele, tárolása a shopping cartban és azon túl. Ehhez egy API-t használtunk amely 5 percenként frissíti az árfolyamokat, így egy metóduson belül a select avagy az oldalon található pénznem lista elemét és annak nevét és ezt, így egy cikluson belül kikeresi az adott pénznemet és ezzel átváltja abba pénznembe a termék árát.

# Összefoglalás

Elért eredmények, tovább fejlesztési lehetőségek

közös munka értékelése

# Források:

abc betűrendben: Szerző, cím, megjelenés dátuma, ha van, link (excelben érdemes gyűjteni és sorba rendezni)

képek forrása külön