Увод

Настоящата дипломна работа е посветена на уеб базирана система за управление на онлайн магазин, реализирана с помощта на HTML, CSS, JavaScript и Firebase.Основната задача на проекта е насочена ключови функционалности като потребителско удостоверяване, управление на продукти, оформление на потребителски интерфейс и прилагане на актуални front-end технологии за създаване на респонсив и интерактивен дизайн.

Първо да се запознаем какво означава понятието “онлайн магазин”.Представлява уеб платформа, която позволява на потребителите да гледат, селектират и купуват стоки и услуги. Така този тип платформи осигуряват удобен и ефективен начин за търговия чрез интернет, като дават възможност на бизнеса да достигне до по-широка аудитория и на потребителите – бърз, лесен и достъпен начин до различни продукти, сравнение на цени и доставка на стоки до дома.

Изборът на темата за анализ на онлайн магазин е поради развитото търсене на електронна търговия в днешната икономика. С напредъка на интернет технологиите и променящото се потребителско поведение, онлайн магазините стават важен възел за осъществяването на продажби и използването на ефективни маркетингови стратегии. Изследването на технологичните платформи в така нареченият “онлайн магазин” позволява премахването на грешките и предизвикателствата и създава добри практики, които могат да се използват в нови проекти

Основната цел на тази дипломна работа е да се оцени структурата и вграждането на системата, както и да се оценят предимствата и минусите на използваните технологии. В този контекст се извършва детайлен преглед на HTML структурата, CSS стилизацията и JavaScript логиката, които директно влияят върху потребителското изживяване и ефективността на системата.

1.1 Началото на технологичната революция и ранните концепции

Първите компютъри са създадени (като ENAIC, 1946) през 40-те и 50-те годиини до микрокомпютърите(Altair 8800, 1975), прогреса на иновациите рязко изменя начина, по който обществето работи с информацията. В началото на създаването на тези “електронни мозъци” са се ползвали за воеено дело и сложни научни изчисления - например за прогноза за времето както и симулиране на ядрени експлозии. Преди изчисления били изпълнявани само от математици, именно това води до създаването на първите компютърни езици като FORTRAN (1957) и COBOL (1959), които автоматизират и улесняват процесите.

60-те години на миналия век инженери като Едгар Код създават релационния модел за база данни, слагайки основите за управление на информация. Друг такъв проект бил ARPANET (прародител на интернета, 1969) демемонстрира възможностите на мрежовата комуникация, макар че от налото били предоставени само за академични институции и военни организации. През това време започват и първите дискусии за стабилността и сигурността на системите - пример за това е атаката от червея Morris (1988), която излага на риск новосъздадените мрежи.

Централизираните системи имат изключителни плюсове и минуси, като те предлагат непозната до сега изчислителна сила, но техните минуси биват - скъпата подръжка, ненадежни и изискват професионален персонал. Това води до търсенето и създаването на персоналнтите компютри.

След създаването на индивидуалната машина се създават и първите операционни системи, които позвололяват на потребителите невиждана до сега многофункционалност. Тези въведения в информациония свят не просто са документирани, а водят технологичния процес.

1.2 Възникване на информационните системи и основни понятия

В началните години на 80-те и 90-те в ИТ секторът става значителна реформа. Това време се отбелязва от прогресивното сменяне от централизирани мрежи към персонални компютри. Свързването им в една обща система става чрез ново въведените локални мрежи (LAN) и протоколи като TCP/IP, което помага на всеки отделен компютър да бъде едновременно и получател и доставчик на информация.

Важни термини като дигитализация (трансформиране на аналогови процеси в цифрови), автоматизация и управлението на информация заемат главно място в дебатите за подобряване на бизнес процесите. Системата оценка се превръша в главна съпка в ключова дисциплина, която обединява различни теоритични модели (като стратегическо управление на проекти и системен анализ) в практиката на ИТ иновациите. Иновационен за времето си е напредакът на грагичните средства - диаграми на потока от данни (DFD) и обектни модели (), които организират комуникацията между екипа разработчици и заинтересованите страни. Едновременно се появяват **обектно-ориентираният дизайн** (пример като C++ и Smalltalk) и модулният подход, които правят програма по-динамична. Това за пример са първоначалните ERP мрежи,където разделни модули за инвентар, финансови операции и човешки ресурси се свързват в една обща система.

В центъра на всички нововъведения стои мисълта за оптимизация – процес, при който една система се подобрява, така че да се намали натоварването и производствен капацитет, изискванията по отношение на паметта или друга характеристика. Вдъхновението идва от япоските бизнеси за управление на качество (като Kaizen), това слага началото на скоростният технологичен напредък изискава сисмети, които са пригодени за бързо еволюиране без драстични промени. Именно тези години поставят базата за интернета (чрез ARPANET) и онлайн търговията.

1.3 Теоретични рамки и моделите за анализ на информационните системи

Изграждането на теоретични модели за анализ на информационните системи е нова неизследвана област. Amazon разработват техните системи за управление на складове с помощта на UML, езикът за моделиране. Това не само показва как изглеждат архитектурите, а превръща абстрактни бизнес цели в конкретни софтуерни части. Например, ER диаграмите от 90-те години помогнаха на банките да избегнат хаоса от ръчните записи, като показаха ясно как са свързани „клиент“, „транзакция“ и „сметка“.

Теорията на системния анализ не е само теоретична област – тя е инструмент за развитие. Когато Spotify използва машинно обучение, за да предскаже кой ще е следващият хит, това не е чудо. Това е реално приложение на модели, които съчетават компютърни науки и психология (тъй като музикалните предпочитания са и социален феномен).

Въпреки това, не всичко е идеално. Критиците, като Едсгер Дейкстра, твърдят, че прекалената зависимост от формални модели може да ограничи иновациите. „Теорията трябва да води, а не да ограничава“, предупреждава той. Въпреки това, без тази теория нямаше да съществува Linux – операционната система, изградена на принципите за свързаност на компонентите и отворен код, което е доказателство, че добрите модели не само създават системи, но и цели екосистеми.

1.4 Ролята на цифровата ера в преформатирането на теориите за развитие

Има драстична промяна в света след появяването на интернета и мобилните устройства, което води до основни промени в центъра на териотичните модели, използвани за изследването на човешките тенденции и размишлявания, бизнес действията и социалните контакти. Преди на теориите били само на хартия, като били главно идейни структури, но отхвърляни поради лиспата на възможността по това време. Днес дигиталната среда изисква по – гъвкави, живи и динамични идеи способни анализират потока от данни в реално време. От платформи като X и Facebook до управляването на алгоритми за пазаруване.

Пример за това е една компания като Nike, те не само искат физическо, но и онлайн присъствие. През тяхното приложения, клиентът може да пазарува и да ги върне в най-близкия физически магазин. Разчитат и на персонализацията на база на предишни покупки, те имат сложни алгоритми обработвани от изкуствен интелект, който предлага оферти на купувачите.

Друг пример са Starbucks, те също разчитат на изкуствения интелект като не искат да превърнат купуването на кафе в нещо уникално, разчитайки на фактори като геолокация, спецеални индивидуални оферти и програма за лоялност. Или IBM Watson, който показва ползата от изкуственият интелект в здравеопазването - в процеса на диагностициране на много редки заболявания и вземане на медицински решения.

1.5 Влиянието на глобализацията и съвременните иновации върху информационните модели

Основните ни разбирания за технологиите се променя с развитието и с модернизирането ни карат да преусмисляме основните ни разбирания за днешните информационни мрежи. Новите въведения и прогресивното ни разбиране от тях в тази динамична среда, в която класическите модели вече не покриват нуждите ни. Нужно е адаптиране към новите нужди на света, където хората и машини комуникират по – нови и по – сложни начини.

Новият напредък е свързан с иконимиката, културата, социалните изменения и потребителските нагласи. Затова е нужно по – широко мислене, „извън рамката на мислене“ за разрастване и подпомагане на бизнеса и обществото. Как това би повлияло на хората, органицациите, дори и начина, по който разсъждаваме.

Днес целта на специалистите е намиране на по-бързи и по- ефективни компотри, за да можем да разберем обществото. Затова са нужни по- мощни компютри, за да могат да обработят толкова голямо количество от информация. Това помага на обществото за да имат по-информирани избори.

Нужно е да се изгради нова визия на света, чрез изпозлването на нови инструменти, идеи и въведения. Именно така компаниите, които искат да успеят, трябва да се научат на ново нестандратно мислене, по-динамично и по-гъвкаво когато се налага. Това включва и да се учиш опита и да избереш най-добрата и стабилна опция.

Информационните технологии не са само технически иновации. Те влияят на начина, по който мислиш, живеем и работим. Няма как да отрежем този важен аспект от нашия дневен живот. Именно затова е нужно осъзнаем, че това част от нашата реалност и да се опитваме да я подобряваме всеки ден.

2. Обща концепция на уеб сайта

2.1. Основна идея на проекта

Основната идея на проекта е създаването на уеб сайт за електронна търговия, наречен MyFootballStore, който предлага на клиентите си да закупят футболни стикери или карти по избор, също така колекционерски футболни продукти. Сайтът таргетира любителите на футбола, които искат неповторими и несрещани артукули към тяхната колекция. Като могат да се запупят лесно и удобно от онлайн магазина.

Задачата е да се практично и сигурно потребителско изживяване, като се гарантира надежност и сигурност при избора или покупката на продукти, чрез пълна сигурност и защита на личните данни с помощта на модерни технологии.

2.2. Основни функционалности

MyFootballStore разполага със следните функции, които осигуряват лесна и удобна навигиция за потребителите:

Динамично зареждане на продукти: Продуктите се показват динамично в страницата с католога и категориите, като реално време бива заредена от Firebase Firestore с всичката информация. Това осигурява лесния достъп за добавяне и обновяване на продуктите, без да е нужна ръчна намеса.

Филтриране и сортиране: Потребителите ще могат да избират различни филтри спрямо категорията по различни изискавания като цена (от ниска към висока и обратното) и име (по азбучен ред). Това позволява лесно навигиране и намиране на продуктите, които потребителите търсят.

Търсене на продукти: Сайтът предлага и опция за търсене по ключови думи, което предоставя възможност на потребителите да открият лесно конкретни артикули

Добавяне в кошница и любими: Опцията потребителите да запазят нещо, което ги е заинтересувало в кошницата или в списъка с любими, като могат да се върнат по-късно за бъдеща покупка.

Профили на потребители и история на поръчките: Потребителите могат да се регистрират и влизат в профила си благодарение на Firebase Authentication. Също вече влезли с сайта могат да променят паролата си или да излязат от профила си. И всеки клиент може да види историята на покупките си и да ги следи състоянието им в реално време.

**Интерфейс с адаптивен дизайн**: Онлайн мазинът има адаптивен дизайн, което означава, че ще работи еднакво добре както и на мобилни усторйства, така и на декстоп машини. Това позволява на потребителя да пазарува и отвсякъде и по всяко време.

**Анимации и модерни визуални ефекти**: Благодарение на AOS (Animate on Scroll), биват интегрирани леки анимации при скролване, което прави изживяването на потребителя по-интерстно и динамично. Тези анимации се ползват при зареждане на нов продукт и останалата част от сайта.

2.4. Разположение на елементите и навигация

Структурата на сайта е направена, така че да е лесно достъпна и разбираема за всеки потребител, като основите елементи са:

Главна навигация: В горната част на онлайн магазина има основното навигационно меню, което съдържа линкове към Главната страница, Магазин, Любими, Кошница, Профил и Админ Панел (показващ се динамично само за администратори)

Херо секция: Тази секция е разположена на началната страница, която те отвежда към най-новите артикули в магазина.

Филтри и сортиране: Филтрите са подрени в центъра страницата с продукти, отдясно на търсачката, където може да се забележи безпроблемно. Като потребителя си избира как да сортира филтрите по категория.

2.5. Уникалност и добавена стойност

Проектът осигурява лесна работа с интерфейса, който съдържа страхотна комбинация от опции и функции. Предлага и допълнителни удобства като добавя към списък с любими продукти, уникални предложения и лесно филтрирране и сортиране. Firebase се грижи за динамичното обновяване на продуктите с лесна и удобна подръжка.

Сайтът предлага удобно и лесно ползване, чрез използването на най-последните технологии и интерфейси и надежност при пазаруване.

3. Избор на технологии и обосновка

За разбработката на уебсайта е избрана комбинация от технологии, които предоставят необходимата производителност, гарантирайки висока производителност, лесна поддръжка и удобен потребителски интерфейс чрез: HTML5, CSS3, JavaScript, Firebase, AOS, Font Awesome, Bootstrap и jQuery

3.1. HTML5: Основата на структурата на сайта

HTML5 (HyperText Markup Language) е основният език за създаване на една уеб страница. Осигурява необходимите тагове, чрез който създаваш основполагащите конструкции на една страница като: залгавия, параграфи, линкове, избображение и форми. Служи като напречна греда за всички останали компоненти на уеб сайта.

За създаване на проекта е използван HTML5, като той служи за базовия скелет на сайта и включва следните елементи:

-Структурни елементи: Таговете <header>, <main> и <footer> разполагат цялата логика на сайта, цялостната архитектура на уеб сайта.

<header> се ползва в горната част на сайта, обикновено има елементи като лого,заглавие и др.

<main> - основната част на страницата

<footer> - долната част на страницата, ползва се линкове, контакти и др.

Навигация: Елементът <nav> се използва за съдаржане на списък от линкове отвеждащ към основните страници(„Магазин“,“Кошница“ „Любими“ и др.).

**<nav** class="shop-nav"**>**

**<ul** class="nav"**>**

**<li** class="nav-item"**><a** class="nav-link active" href="index.html"**><i** class="fas fa-home"**></i>** Home**</a></li>**

**<li** class="nav-item"**><a** class="nav-link" href="shop.html"**><i** class="fas fa-store"**></i>** Shop**</a></li>**

**<li** class="nav-item"**>**

**<a** class="nav-link" href="favorites.html"**>**

<nav> ползва се за линкове към другата част от сайта

Форми за регистрация и вход: Жизнено важни за един сайта са елементите <form>, <input>, <button>, те позволяват направата на количката, формулярите за регистрация и вход възможни.

**<nav** class="shop-nav"**>**

**<ul** class="nav"**>**

**<li** class="nav-item"**><a** class="nav-link active" href="index.html"**><i** class="fas fa-home"**></i>** Home**</a></li>**

**<li** class="nav-item"**><a** class="nav-link" href="shop.html"**><i** class="fas fa-store"**></i>** Shop**</a></li>**

**<li** class="nav-item"**>**

**<a** class="nav-link" href="favorites.html"**>**

<form> се ползва за архивиране на информация от потрибителя, като имейл, парола, име и др.

<input> - се използва за въвеждане на данни

<button> - бутон, който може да натискаш, може да задейства скриптове и др.

HTML е сърцето на един уеб сайт, необходим, защото той определя какво ще се покаже на една страница. Работещ заедно с CSS и JavaScript, осигурявайки завършен вид на цялостния сайт.

3.2. CSS3: Стилово оформление и визуален дизайн

CSS3 (Cascading Style Sheets) е от съществено значение за изграждането на стилистичната идентичност на уеб сайта, като дава възможност за персонализация която става лесно и бърз. Елементите които дават образа на сайта са: цветове, шрифтове и разположение на съдържанието. CSS разполага също с респосив дизайн, позволяващ употреба от всякакъв размер екран или различни устройства.

Използвани CSS технологии: Мрежи (Grid/Flexbox): Тези технологии позволяват разполагането на елементите в различни колони и редове, което е особено полезно за организацията на продуктите. В примера с филтърната лента и продуктовите карти, CSS Grid и Flexbox се ползват за визуалното разпределение и осигуряване на динамичен дизайн.

**shop-filter-bar.modern-filter** {

**display**: flex;

flex-wrap: wrap;

align-items: **center**;

gap: 1rem;

**padding**: 1.5rem;

**background**: #fff;

**border**: 1px **solid** rgba(255, 255, 255, 0.2);

**border**-radius: 15px;

box-shadow: var(--shadow-md);

**margin**: -50px **auto** 30px;

**position**: **relative**;

**z-index**: 10;

**Адаптивен дизайн (Responsive Design)**: CSS Media Queries позволява оптимизацията на сайта, независимо от какво устройсто или екран. Това прави сайта да изглежда еднакво добре както и на мобилни устройства и на декстоп машини.

**Responsive** **Adjustments** **-** **Updated** \*/

**@media** (**max-width**: **768px**) {

**.shop-hero** {

**min-height**: 50vh; */\* Adjusted height for mobile \*/*

}

**.shop-hero** **.hero-title** {

**font-size**: 2.5rem; */\* Adjusted size for mobile \*/*

}

**.shop-hero** **.hero-subtitle** {

**font-size**: 1rem; */\* Adjusted size for mobile \*/*

}

**Трансформации и преходи (Transitions/Transforms)**: За да се постигне гъвкав и завладяващ ефект при работа с други елементи, като например при задържане на курсора върху бутон, осигурява гладко анимиране приятно за окото.

**product-card:hover** {

transform: translateY(-10px);

box-shadow: var(--shadow-hover);

}

Без CSS сайта ще изглежда като един празен текстов документ. Благодарение на нега сайта става по-жив и цветен, осигурявайки по приятно изживяване.

3.3. JavaScript: Динамично поведение на страниците

JavaScript е третият основополагащ език, който гради сайта, отговорен за динамичността на сайта. Той позволява извършването на действия в реално време без нужда да се рестартира страницата. Това включва опции като динамично зареждане на продукти, добавяне в кошницата и филтриране на продукти и др.

Примери за динамични взаимодействия:

- Добавяне на продукти в кошницата: Когато потребителят избере опцията „Добави в кошницата“, JavaScript е отговорен за добавянето на продукта в локалната база данни и да промени облика на количката.

document.querySelectorAll("[data-remove-id]").forEach(btn => {

btn.addEventListener("click", e => {

removeFromCart(e.target.dataset.removeId, uid);

});

});

})

.**catch**(err => {

console.error("Error loading cart:", err);

cartList.innerHTML = "<p>Error loading cart items.</p>";

});

}

Филтриране на продукти: Когато потребителят селектира избрана от него категория или ценови обсег, JavaScript обработва филтрите в реално време и актуализира новите резултати.

document.addEventListener("DOMContentLoaded", () => {

**const** productsListEl = document.getElementById("productsList");

**const** searchInput = document.getElementById("searchInput");

**const** categoryFilter = document.getElementById("categoryFilter");

**const** sortFilter = document.getElementById("sortFilter");

**const** clearFiltersBtn = document.getElementById("clearFilters");

**let** productsData = [];

**if** (!productsListEl) **return**; *// Ensure we're on shop page*

*// Add this at the beginning of your DOMContentLoaded event handler*

document.addEventListener("categoryUpdated", () => {

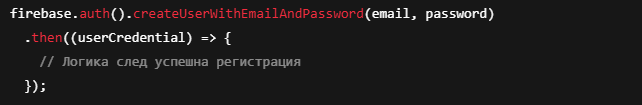
loadCategoriesForShop();

});

3.4. Firebase

Firebase предлага разнообразие от бекенд услуги, които значително опростяват разработката на уеб приложения, без нужда от настройка на собствен сървър. За MyFootballStore, Firebase предоставя три основни услуги:

Аутентикация на потребителите: Firebase Authentication е опция, която осигурява сигурното регистране и вход на потребители чрез различните методи на идентификация: имейл, парала, Google и др.



**База данни (Firestore)**: Firestore е клауд база данните осигуряваша лесната синхронизация между потребителя и сървъра. Използва се за архивиране на информация за продукти, поръчки, категории, потребители и др

**Съхранение на файлове (Firebase Storage)**: Firebase Storage се ползва за запазване на файлове като изброжение на продукт или др.

3.5. **AOS (Animate On Scroll)**

**AOS** (Animate On Scroll) е библиотека, позволяща на елементите от сайта да се анимират когато потребител скролва през дадена страница. Правейки посещаването на магазина по-приятно и интригуващо.

Пример за анимация:

*!-- HERO SECTION -->*

**<section** class="shop-hero"**>**

**<div** class="hero-content"**>**

**<h2** class="hero-title display-3 text-white mb-4" data-aos="fade-down"**>**

Explore Premium Football Gear

**</h2>**

**<p** class="hero-subtitle lead text-white mb-5" data-aos="fade-up" data-aos-delay="200"**>**

Discover authentic jerseys, limited editions, and exclusive collections

**</p>**

**</div>**

**</section>**

*<!-- MODERN FILTER BAR -->*

**<section** class="container"**>**

**<div** class="shop-filter-bar modern-filter" data-aos="fade-up" data-aos-delay="300"**>**

Тази функция прави навигацията при разглеждане на сайта целенасочена и привлича вниманието на клиента.

3.6. **Font Awesome**

**Font Awesome** предлага голям подбор от икони, които могат да се използват за удобство на потребителите. Те се забелязват по-лесно при действия като „Добавяне в кошница“ или „Добавяне в любими“ и др., правейки сайта по user-friendly.

**<ul** class="nav"**>**

**<li** class="nav-item"**><a** class="nav-link" href="index.html"**><i** class="fas fa-home"**></i>** Home**</a></li>**

**<li** class="nav-item"**><a** class="nav-link active" href="shop.html"**><i** class="fas fa-store"**></i>** Shop**</a></li>**

**<li** class="nav-item"**>**

**<a** class="nav-link" href="favorites.html"**>**

**<i** class="fas fa-heart"**></i>** Favorites **<span** class="badge badge-pill badge-warning" id="favoritesCount"**>**0**</span>**

**</a>**

**</li>**

**<li** class="nav-item"**>**

**<a** class="nav-link" href="cart.html"**>**

**<i** class="fas fa-shopping-cart"**></i>** Cart **<span** class="badge badge-pill badge-warning" id="cartCount"**>**0**</span>**

**</a>**

**</li>**

**<li** class="nav-item"**><a** class="nav-link" href="profile.html" id="profileNavItem" style="display:none;"**><i** class="fas fa-user"**></i>** Profile**</a></li>**

**<li** class="nav-item"**><a** class="nav-link" href="admin.html" id="adminNavItem" style="display:none;"**><i** class="fas fa-user-shield"**></i>** Admin**</a></li>**

**<li** class="nav-item"**><a** class="nav-link" href="login.html" id="loginLink" style="display:none;"**><i** class="fas fa-sign-in-alt"**></i>** Login**</a></li>**

**<li** class="nav-item"**><a** class="nav-link" href="register.html" id="registerLink" style="display:none;"**><i** class="fas fa-user-plus"**></i>** Register**</a></li>**

**</ul>**

**</nav>**

**</div>**

**</header>**

Иконите позволяват по-бързото навигиране из сайта и по-лесно разбираемият интерфейс.

3.7. Bootstrap и jQuery

За реализиране на интуитивен и отзивчив потребителски интерфейс са използвани **Bootstrap** и **jQuery**. Тези технологии осигуряват бързо и ефективно изграждане на съвременен уеб интерфейс с богата функционалност.

**Bootstrap**

Bootstrap е широко използван фронт-енд фреймворк, който позволява бързото създаване на адаптивни и естетични уеб страници. В проекта е използван за:

* изграждане на **отзивчиво оформление**, което се адаптира към различни резолюции и устройства (десктоп, таблет, мобилен);
* прилагане на **готови компоненти** като навигационни менюта, бутони и модали;
* осигуряване на **консистентен стил** чрез вградената си система от CSS класове.

jQuery е лека JavaScript библиотека, която опростява работата с DOM и обработката на събития. В рамките на проекта, jQuery е използван за:

* **динамична манипулация на DOM** – добавяне, премахване и обновяване на елементи в реално време;
* **обработка на събития**, като например клик върху бутони за добавяне на продукти в количката;
* реализиране на **интерактивно поведение** – скриване/показване на елементи, анимации и известия.

**4. Структура на сайта**A diagram of a company

AI-generated content may be incorrect.

**Регистрирани потребители/Нерегестрирани потребители:**

**Регистрация / Вход: Потребител, който е създал акаунт може да влезе.**

**Преглед на продукти: Може да разгледа каталога от продукти**

**Филтриране и търсене: Търсене по филтри или категории и достъп до търсачката за търсене по ключови думи.**

**Продуктов детайл: Има достъп до детайлите на конкретен продукт.**

**Добавяне в количка/любими: Избиране на продукти за запазване в списъка за любими или кошница.**

**Профил и История: Преглед на личните данни и история на поръчките**

**Checkout: Финализиране на плащането**

**Администраторски панел**

**Регистрация/Вход: Потребител, който е само админ.**

**Добавяне на продукти: Само от администратор, когато е нужно**

**Редактиране/Изтриване: Актуализиране на магазина.**

**Преглед на всички продукти: Вижда всички артикули**

**Преглед на поръчки: Следенето на всички направени поръчки**

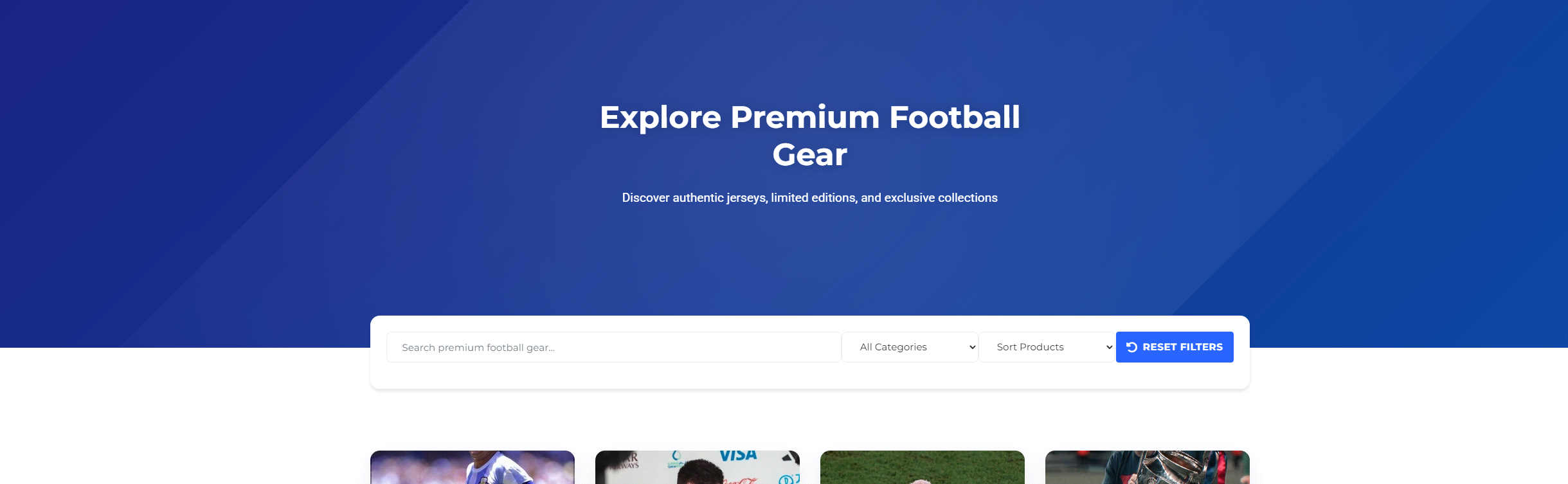
**Структурата на проекта е лесна и интерктивнo проектирана за потребителя. Добър дизайн с добро разположение на всички главни елементи. Също така е обърнато и внимание да отговоря за нуждите на потребители и администратори.**

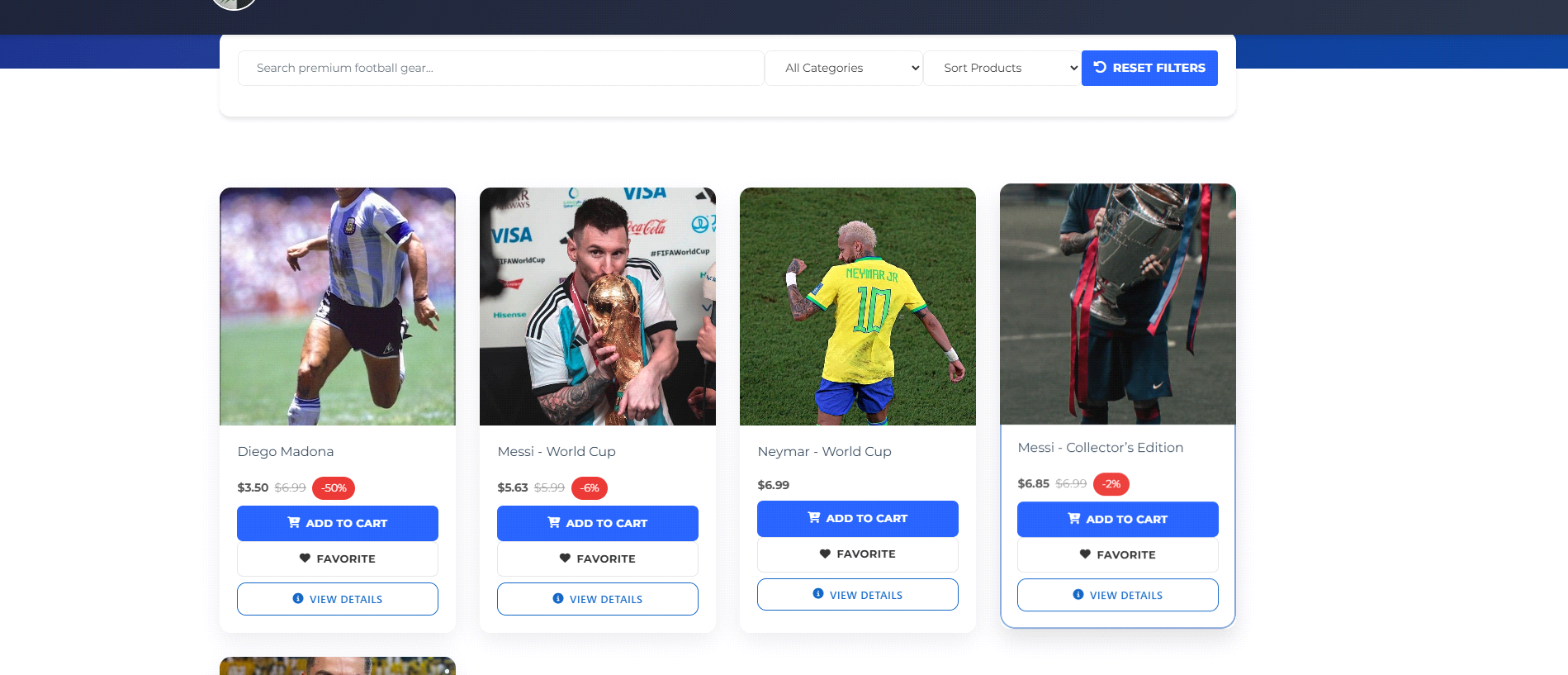
**4.1. Начална страница (Index Page)**

**Страницата е насочена да привлече вниманието на потребителя като има лесен достъп до всички страници, напревени лесно забележими, като още влезеш има опция да посетиш магазина. А други основни елементи са:**

* **Навигационно меню: Съдържа опции за „Магазин“, „Любими“, „Кошница“ и „Вход/Регистрация“. Когато потребителят се е регистрирал или влязъл всички други опции като: Профил, Административен Панел (само за админи)**.
* **<li** class="nav-item"**><a** class="nav-link active" href="index.html"**><i** class="fas fa-home"**></i>** Home**</a></li>**
* **<li** class="nav-item"**><a** class="nav-link" href="shop.html"**><i** class="fas fa-store"**></i>** Shop**</a></li>**
* **<li** class="nav-item"**>**
* **<a** class="nav-link" href="favorites.html"**>**
* **<i** class="fas fa-heart"**></i>** Favorites **<span** class="badge badge-pill badge-warning" id="favoritesCount"**>**0**</span>**
* **</a>**
* **</li>**
* **<li** class="nav-item"**>**
* **<a** class="nav-link" href="cart.html"**>**
* **<i** class="fas fa-shopping-cart"**></i>** Cart **<span** class="badge badge-pill badge-warning" id="cartCount"**>**0**</span>**
* **</a>**
* **</li>**
* **<li** class="nav-item"**><a** class="nav-link" href="profile.html" id="profileNavItem" style="display:none;"**><i** class="fas fa-user"**></i>** Profile**</a></li>**
* **<li** class="nav-item"**><a** class="nav-link" href="admin.html" id="adminNavItem" style="display:none;"**><i** class="fas fa-user-shield"**></i>** Admin**</a></li>**
* **<li** class="nav-item"**><a** class="nav-link" href="login.html" id="loginLink" style="display:none;"**><i** class="fas fa-sign-in-alt"**></i>** Login**</a></li>**
* **<li** class="nav-item"**><a** class="nav-link" href="register.html" id="registerLink" style="display:none;"**><i** class="fas fa-user-plus"**></i>** Register**</a></li>**
* **<li** class="nav-item" id="adminAccess" style="display: none;"**>**
* **<a** href="admin.html" class="nav-link"**>**
* **<i** class="fas fa-user-shield"**></i>** Admin Panel
* **Продуктови предложения: Избор на най-новите и актуални колекции**

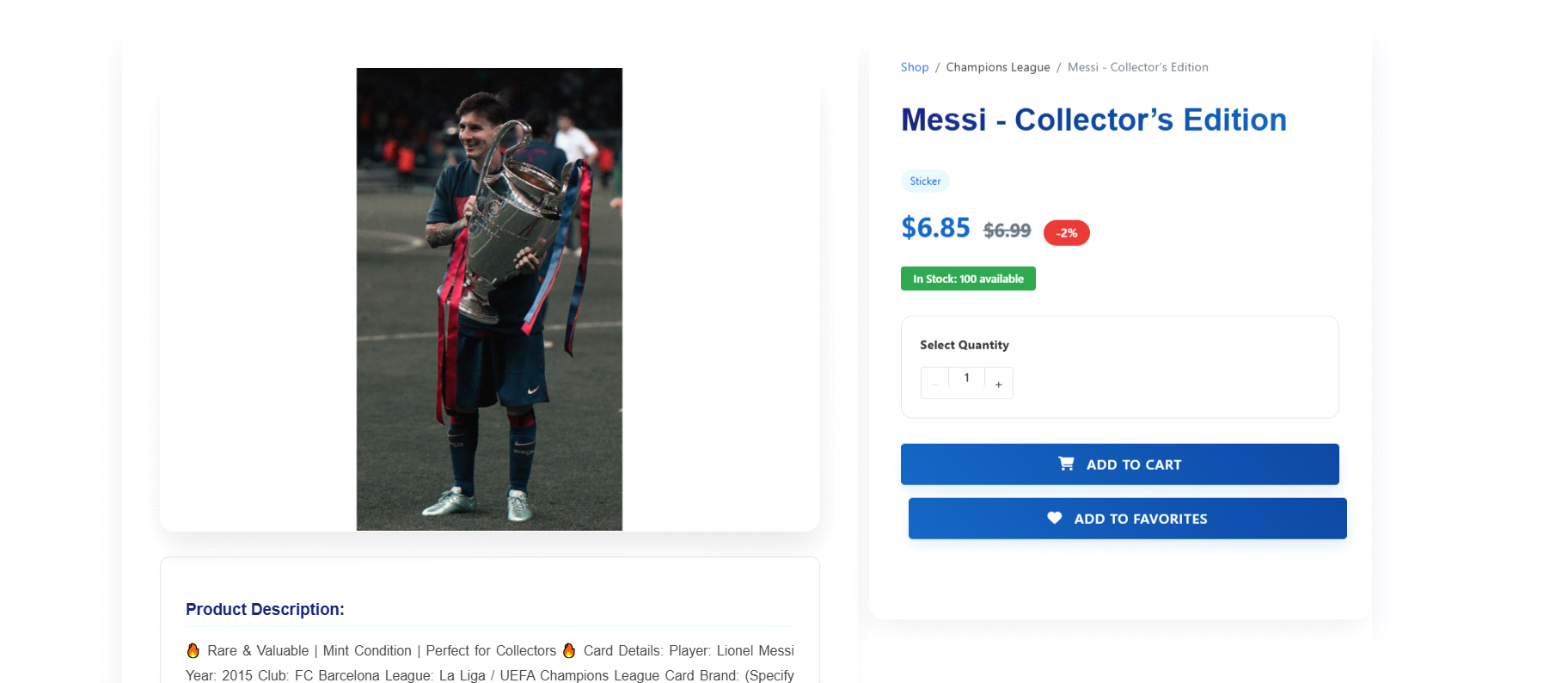
**4.2. Страница за магазин (Shop Page)**



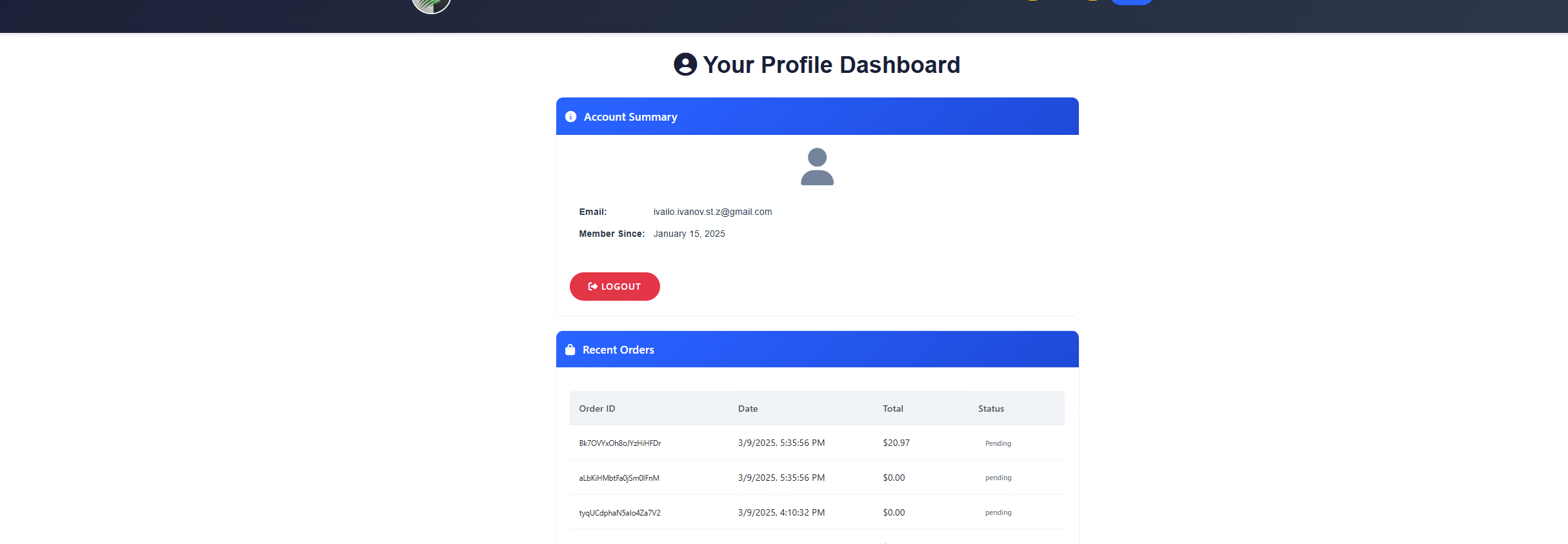


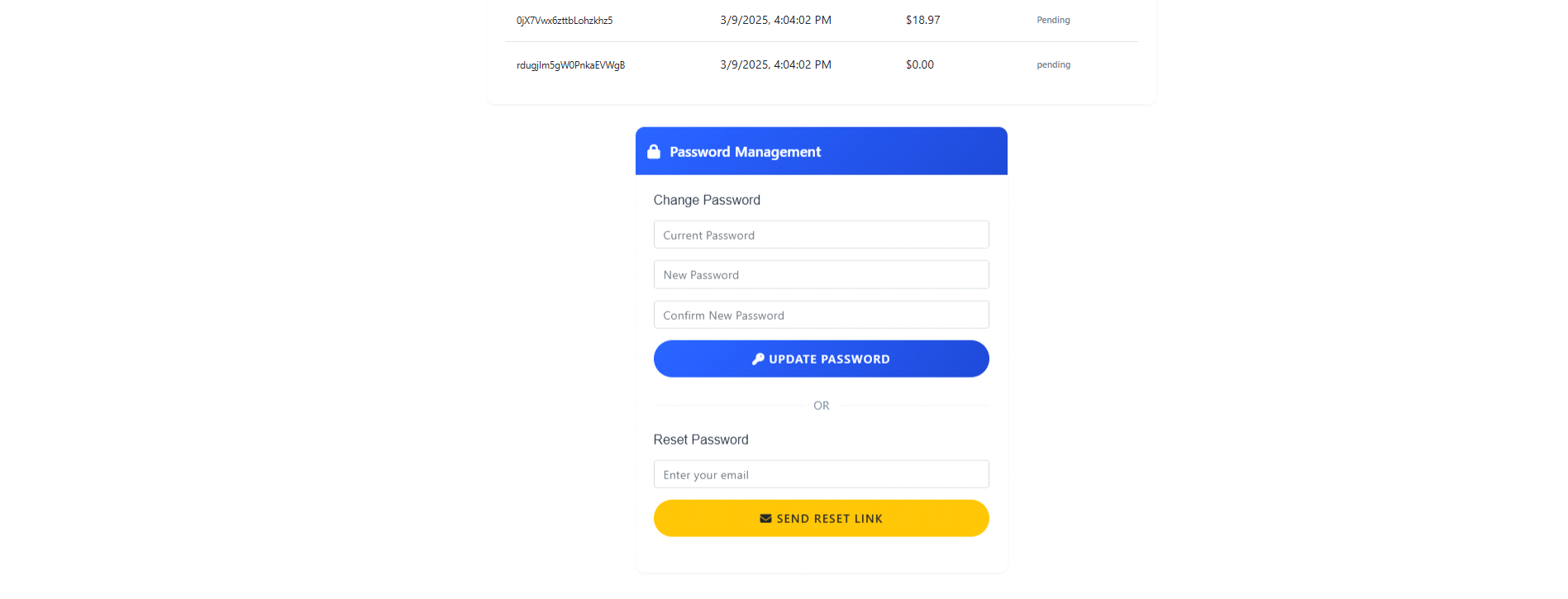
**На тази страница потребителите разглеждат продуктите, която съдържа следните елементи:**

* **Филтри и търсене: Възможност за търсене по категории и филтриране на по ценови диапазон и азбучен ред.**
* *!-- MODERN FILTER BAR -->*
* **<section** class="container"**>**
* **<div** class="shop-filter-bar modern-filter" data-aos="fade-up" data-aos-delay="300"**>**
* **<div** class="d-flex flex-wrap gap-4 align-items-center w-100"**>**
* **<div** class="flex-grow-1 position-relative"**>**
* **<i** class="fas fa-search position-absolute" style="left: 15px; top: 50%; transform: translateY(-50%);"**></i>**
* **<input** type="search" id="searchInput" class="form-control form-control-lg ps-5" placeholder="Search premium football gear..."**>**
* **</div>**
* **<div** class="d-flex gap-3"**>**
* **<select** id="categoryFilter" class="form-select form-select-lg"**>**
* **<option** value=""**>**All Categories**</option>**
* **<option** value="Legends"**>**Legends**</option>**
* **<option** value="WorldCup"**>**World Cup**</option>**
* **<option** value="ChampionsLeague"**>**Champions League**</option>**
* **</select>**
* **<select** id="sortFilter" class="form-select form-select-lg"**>**
* **<option** value=""**>**Sort Products**</option>**
* **<option** value="price-asc"**>**Price: Low to High**</option>**
* **<option** value="price-desc"**>**Price: High to Low**</option>**
* **<option** value="name-asc"**>**Name: A to Z**</option>**
* **<option** value="name-desc"**>**Name: Z to A**</option>**
* **</select>**
* **<button** id="clearFilters" class="btn btn-lg shop-reset-btn"**>**
* **<i** class="fas fa-undo-alt"**></i>** Reset Filters
* **Продуктови карти: Всеки продукт е показан като картинка с описание, цена и бутон за добавяне в количката и любими, както и спецификации.**



**4.3. Страница за профил на потребителя (Profile Page)**

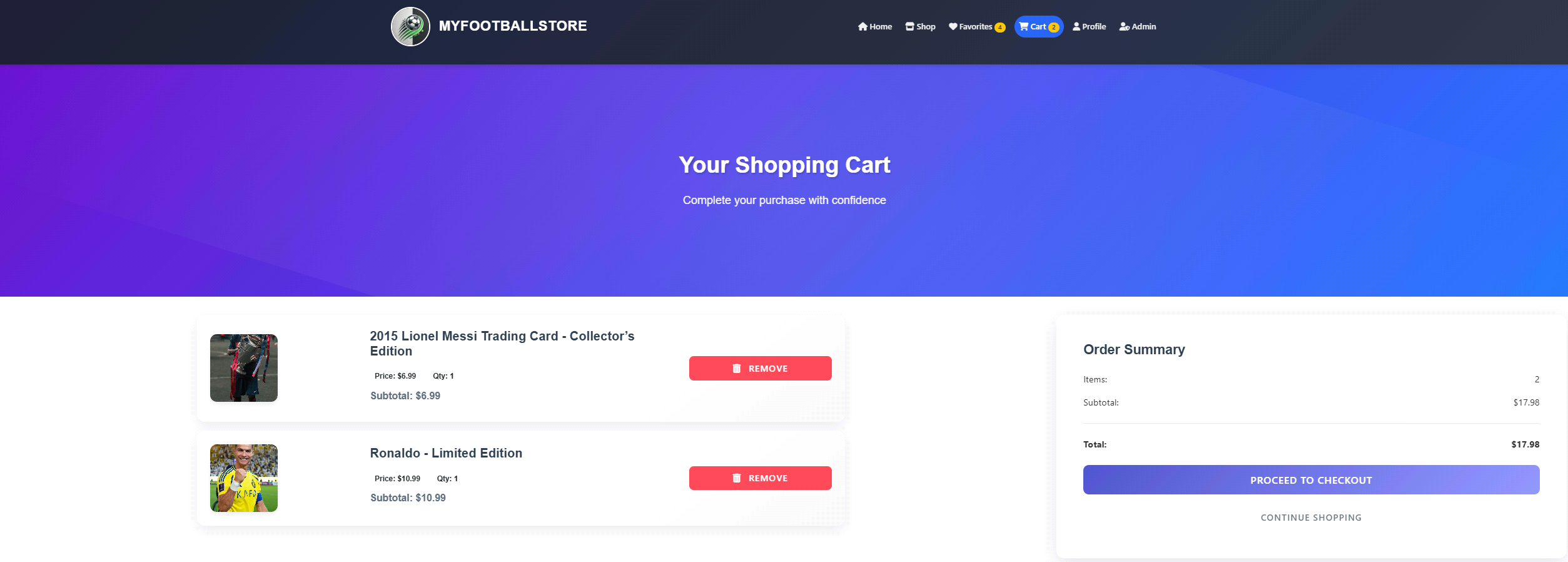




**Профилната страница има опция за проследяване на поръчки, за излизане от сайта и управление на личните данни:**

* **Лични данни: Показване на информация за потребителя, като име, имейл и други данни, които могат да бъдат променяни.**
* *!-- Password Management Card -->*
* **<div** class="profile-card"**>**
* **<div** class="card-header"**>**
* **<i** class="fas fa-lock"**></i>** Password Management
* **</div>**
* **<div** class="card-body"**>**
* **<form** id="changePasswordForm" class="password-form"**>**
* **<h4>**Change Password**</h4>**
* **<div** class="form-group"**>**
* **<input** type="password" id="currentPassword" class="form-control"
* placeholder="Current Password" required **/>**
* **</div>**
* **<div** class="form-group"**>**
* **<input** type="password" id="newPassword" class="form-control"
* placeholder="New Password" required minlength="6" **/>**
* **</div>**
* **<div** class="form-group"**>**
* **<input** type="password" id="confirmNewPassword" class="form-control"
* placeholder="Confirm New Password" required minlength="6" **/>**
* **</div>**
* **<button** type="submit" class="btn btn-primary btn-block"**>**
* **<i** class="fas fa-key"**></i>** Update Password
* **</button>**
* **</form>**
* **<div** class="divider"**>**
* **<span>**OR**</span>**
* **</div>**
* **<form** id="resetPasswordForm" class="password-form"**>**
* **<h4>**Reset Password**</h4>**
* **<div** class="form-group"**>**
* **<input** type="email" id="resetEmail" class="form-control"
* placeholder="Enter your email" required **/>**
* **</div>**
* **<button** type="submit" class="btn btn-warning btn-block"**>**
* **<i** class="fas fa-envelope"**></i>** Send Reset Link
* **</button>**
* **</form>**
* **</div>**
* **</div>**
* **</div>**
* **</main>**
* **История на поръчки: Всеки клиент има възможност да прегледа предишните съ поръчки, също така и тези в реално време.**

**4.4. Страница за кошницата (Cart Page)**



**Страницата за кошницата потребилите могат да видат добавите от тях продукти, както и да ги премахнат, да видат общата сума или да продължат към страницата за разплащане.**

*<!-- Updated MAIN CONTENT -->*

**<main** class="container-fluid px-4 fade-in"**>**

**<div** class="row"**>**

*<!-- Cart Items Column -->*

**<div** class="col-lg-8"**>**

**<div** id="cartList" class="fade-in"**>**

*<!-- Dynamic cart items -->*

**</div>**

**</div>**

*<!-- Summary Column -->*

**<div** class="col-lg-4"**>**

**<div** class="summary-card sticky-top" style="top: 2rem;"**>**

**<div** class="card-body"**>**

**<h5** class="mb-4"**>**Order Summary**</h5>**

**<div** class="summary-details mb-4"**>**

**<div** class="d-flex justify-content-between mb-3"**>**

**<span>**Items:**</span>**

**<span** id="cartItemCount"**>**0**</span>**

**</div>**

**<div** class="d-flex justify-content-between mb-3"**>**

**<span>**Subtotal:**</span>**

**<span** id="cartSubtotal"**>**$0.00**</span>**

**</div>**

**<hr** class="my-4"**>**

**<div** class="d-flex justify-content-between mb-4"**>**

**<strong>**Total:**</strong>**

**<strong** id="cartTotal"**>**$0.00**</strong>**

**</div>**

**</div>**

**<a** href="checkout.html" class="btn btn-primary w-100 ripple-button"**>**

Proceed to Checkout

**</a>**

**<a** href="shop.html" class="btn btn-outline-secondary w-100 mt-3 ripple-button"**>**

Continue Shopping

**</a>**

**</div>**

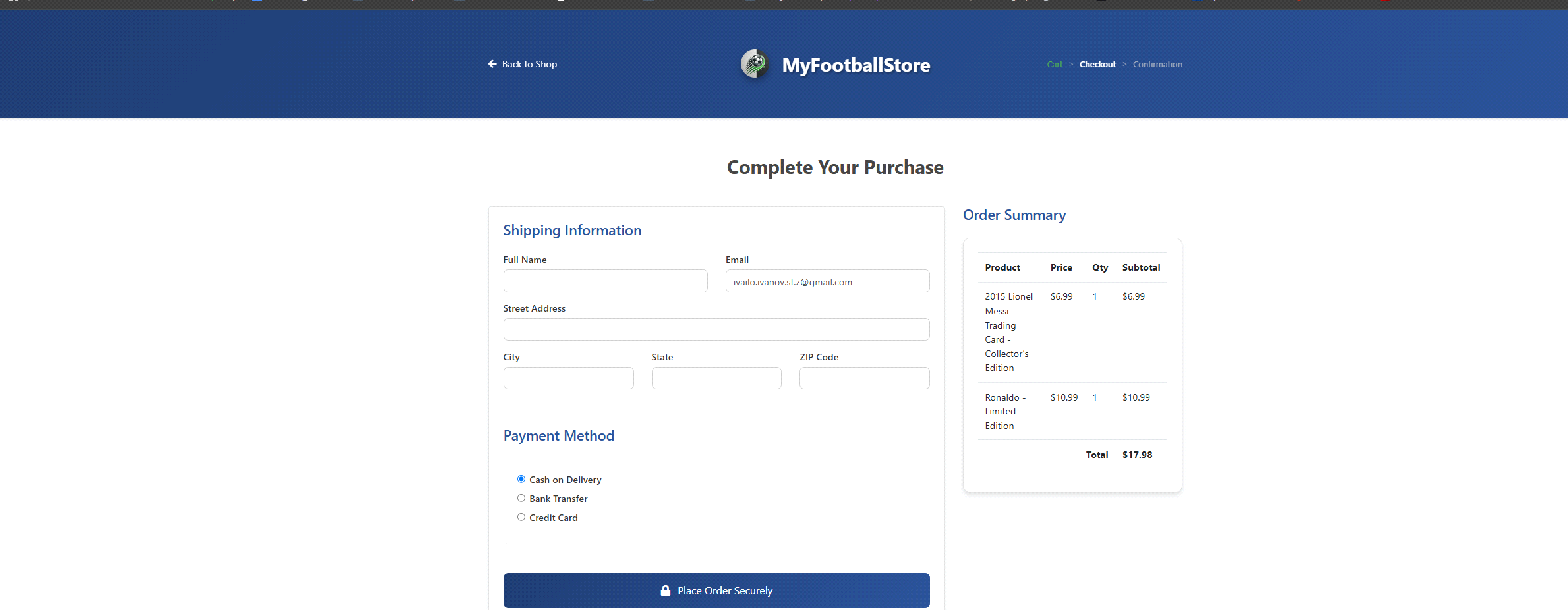
**</div>**

**</div>**

**</div>**

**</main>**

**4.5. Страница за поръчка (Checkout Page)**



**Тази страница позволява на потребителя да завърши поръчката. Тя включва следните елементи:**

* **Формуляр за доставка: Потребителят, трябва да въведе данни за доставка като име, адрес, телефонен номер и имейл.**
* **Обобщение на поръчката: Преглед на всички продукти, които ще бъдат закупени, тяхната цена и общата сума.**

*!-- Checkout Form Column -->*

**<div** class="col-md-8 order-md-1"**>**

**<form** id="checkoutForm" class="bg-white p-4 rounded-lg shadow-sm"**>**

**<h2** class="section-title"**>**Shipping Information**</h2>**

**<div** class="row"**>**

**<div** class="col-md-6"**>**

**<div** class="form-group"**>**

**<label** for="fullName"**>**Full Name**</label>**

**<input** type="text" class="form-control" id="fullName" required **/>**

**</div>**

**</div>**

**<div** class="col-md-6"**>**

**<div** class="form-group"**>**

**<label** for="email"**>**Email**</label>**

**<input** type="email" class="form-control" id="email" required **/>**

**</div>**

**</div>**

**</div>**

**<div** class="form-group"**>**

**<label** for="shippingAddress"**>**Street Address**</label>**

**<input** type="text" class="form-control" id="shippingAddress" required **/>**

**</div>**

**<div** class="row"**>**

**<div** class="col-md-4"**>**

**<div** class="form-group"**>**

**<label** for="city"**>**City**</label>**

**<input** type="text" class="form-control" id="city" required **/>**

**</div>**

**</div>**

**<div** class="col-md-4"**>**

**<div** class="form-group"**>**

**<label** for="state"**>**State**</label>**

**<input** type="text" class="form-control" id="state" required **/>**

**</div>**

**</div>**

**<div** class="col-md-4"**>**

**<div** class="form-group"**>**

**<label** for="zipCode"**>**ZIP Code**</label>**

**<input** type="text" class="form-control" id="zipCode" required **/>**

**</div>**

**</div>**

**</div>**

*<!-- Payment Method Section -->*

**<h2** class="section-title mt-5"**>**Payment Method**</h2>**

**<div** class="payment-methods mb-4"**>**

**<div** class="form-check mb-2"**>**

**<input** class="form-check-input" type="radio" name="paymentMethod" id="cashOnDelivery" value="cash" checked **/>**

**<label** class="form-check-label" for="cashOnDelivery"**>**Cash on Delivery**</label>**

**</div>**

**<div** class="form-check mb-2"**>**

**<input** class="form-check-input" type="radio" name="paymentMethod" id="bankTransfer" value="bank" **/>**

**<label** class="form-check-label" for="bankTransfer"**>**Bank Transfer**</label>**

**</div>**

**<div** class="form-check"**>**

**<input** class="form-check-input" type="radio" name="paymentMethod" id="creditCard" value="card" **/>**

**<label** class="form-check-label" for="creditCard"**>**Credit Card**</label>**

**</div>**

**</div>**

*<!-- Bank Transfer Details (Initially hidden) -->*

**<div** id="bankTransferDetails" class="payment-details rounded-lg" style="display: none;"**>**

**<div** class="alert alert-info"**>**

**<h4>**Bank Transfer Details**</h4>**

**<p>**Please transfer the total amount to:**</p>**

**<p>**

Bank: MyBank**<br** **/>**

IBAN: BG18RZBB91550123456789**<br** **/>**

BIC: RZBBBGSF**<br** **/>**

Recipient: MyFootballStore Ltd

**</p>**

**<small>**\* Please use your order number as payment reference**</small>**

**</div>**

**</div>**

*<!-- Credit Card Details (Initially hidden) -->*

**<div** id="creditCardDetails" class="payment-details" style="display: none;"**>**

**<div** class="form-group"**>**

**<label** for="cardNumber"**>**Card Number**</label>**

**<input** type="text" class="form-control" id="cardNumber" pattern="[0-9]{16}" **/>**

**</div>**

**<div** class="row"**>**

**<div** class="col-md-6"**>**

**<div** class="form-group"**>**

**<label** for="expiryDate"**>**Expiry Date**</label>**

**<input** type="text" class="form-control" id="expiryDate" placeholder="MM/YY" pattern="(0[1-9]|1[0-2])\/([0-9]{2})" **/>**

**</div>**

**</div>**

**<div** class="col-md-6"**>**

**<div** class="form-group"**>**

**<label** for="cvv"**>**CVV**</label>**

**<input** type="text" class="form-control" id="cvv" pattern="[0-9]{3,4}" **/>**

**</div>**

**</div>**

**</div>**

**</div>**

**<button** type="submit" class="btn btn-primary btn-lg btn-block mt-5" id="placeOrderBtn"**>**

**<i** class="fas fa-lock mr-2"**></i>** Place Order Securely

**</button>**

**</form>**

**</div>**

**</div>**

**</main>**

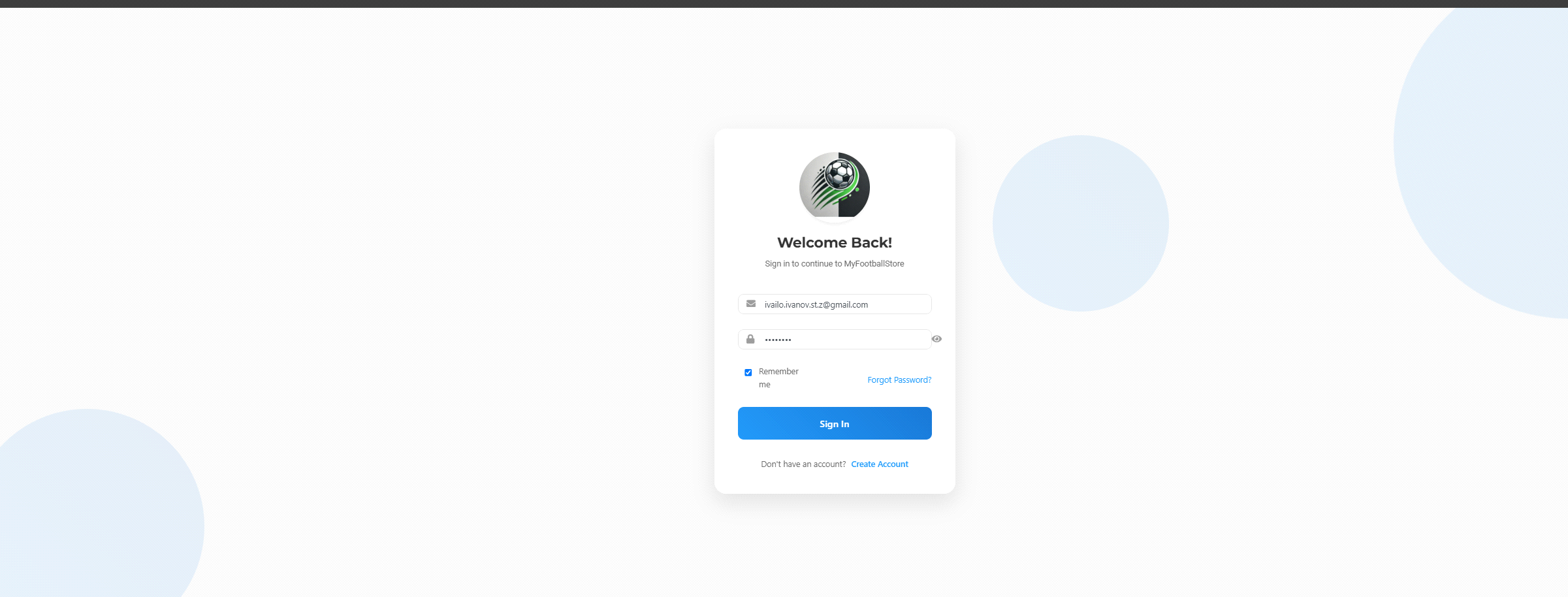
**4.6. Административен панел (Admin Panel)**

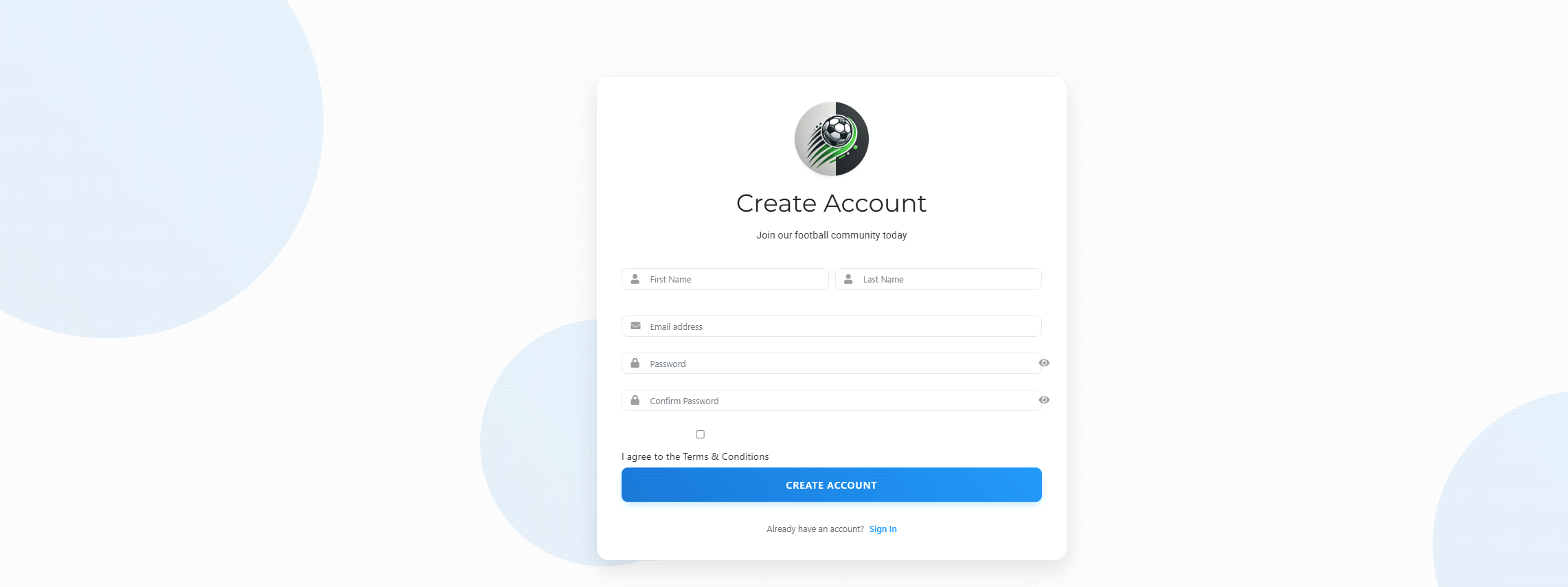


**Административният панел е нужен за управлението на продуктите и поръчките и цялостен мониторинг на сайта. Той предоставя следните възможности:**

* **Управление на продукти: Администраторите могат да добавят нови продукти, да ги редактират или да ги изтриват. Формулярът за добавяне на продукт включва полета за име, цена, категория и изображение на продукта.**
* **Проследяване на поръчки: Администраторите могат да виждат всички направени поръчки и да актуализират техния статус.**

**4.7. Страница за вход/регистрация (Login/Registration Page)**





**Тази страница позволява на потребителите да влязат в своя акаунт или да създадат нов. Формулярите са лесни за използване и включват полета за имейл и парола.**

**Формуляра за регистация:** **<form** id="registerForm" class="register-form"**>**

**<div** class="form-row"**>**

**<div** class="form-group col-md-6"**>**

**<div** class="input-icon-wrapper"**>**

**<i** class="fas fa-user"**></i>**

**<input** type="text" id="firstName" class="form-control" placeholder="First Name" required**>**

**<span** class="focus-border"**></span>**

**</div>**

**</div>**

**<div** class="form-group col-md-6"**>**

**<div** class="input-icon-wrapper"**>**

**<i** class="fas fa-user"**></i>**

**<input** type="text" id="lastName" class="form-control" placeholder="Last Name" required**>**

**<span** class="focus-border"**></span>**

**</div>**

**</div>**

**</div>**

**<div** class="form-group"**>**

**<div** class="input-icon-wrapper"**>**

**<i** class="fas fa-envelope"**></i>**

**<input** type="email" id="email" class="form-control" placeholder="Email address" required**>**

**<span** class="focus-border"**></span>**

**</div>**

**</div>**

**<div** class="form-group"**>**

**<div** class="input-icon-wrapper"**>**

**<i** class="fas fa-lock"**></i>**

**<input** type="password" id="password" class="form-control" placeholder="Password" required**>**

**<span** class="focus-border"**></span>**

**<button** type="button" class="toggle-password" id="togglePassword"**>**

**<i** class="fas fa-eye"**></i>**

**</button>**

**</div>**

**</div>**

**<div** class="form-group"**>**

**<div** class="input-icon-wrapper"**>**

**<i** class="fas fa-lock"**></i>**

**<input** type="password" id="confirmPassword" class="form-control" placeholder="Confirm Password" required**>**

**<span** class="focus-border"**></span>**

**<button** type="button" class="toggle-password" id="toggleConfirmPassword"**>**

**<i** class="fas fa-eye"**></i>**

**</button>**

**</div>**

**</div>**

**<div** class="form-options"**>**

**<label** class="remember-me"**>**

**<input** type="checkbox" id="agreeTerms" required**>**

**<span** class="checkmark"**></span>**

I agree to the Terms & Conditions

**</label>**

**</div>**

**<div** id="errorMessage" class="alert alert-danger" style="display:none;"**></div>**

**<button** type="submit" class="register-button" id="registerButton"**>**

**<span** id="registerButtonText"**>**Create Account**</span>**

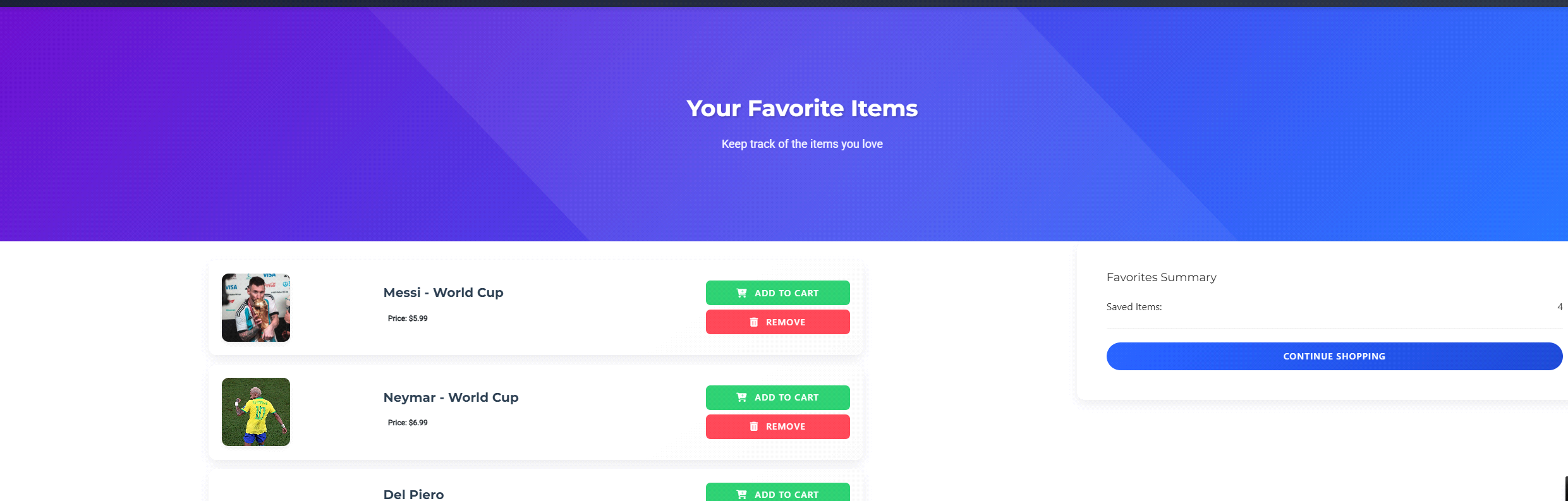
**<span** id="registerButtonSpinner" class="spinner" style="display:none;"**></span>**

**</button>**

**</form>**

**4.8. Страница за любими продукти (Favorites Page)**

**Страница позволява на потребителите да запазят любими продукти и да се върнат по-късно.**



*<!-- Main Content -->*

**<main** class="container-fluid px-4 fade-in"**>**

**<div** class="row"**>**

*<!-- Favorites Items Column -->*

**<div** class="col-lg-8"**>**

**<div** id="favoritesList" class="fade-in"**>**

*<!-- Dynamic favorites items -->*

**</div>**

**</div>**

*<!-- Summary Column -->*

**<div** class="col-lg-4"**>**

**<div** class="summary-card sticky-top" style="top: 2rem;"**>**

**<div** class="card-body"**>**

**<h5** class="mb-4"**>**Favorites Summary**</h5>**

**<div** class="summary-details mb-4"**>**

**<div** class="d-flex justify-content-between mb-3"**>**

**<span>**Saved Items:**</span>**

**<span** id="favoritesItemCount"**>**0**</span>**

**</div>**

**<hr** class="my-4"**>**

**</div>**

**<a** href="shop.html" class="btn btn-primary w-100 ripple-button"**>**

Continue Shopping

**</a>**

**</div>**

**</div>**

**</div>**

**</div>**

**</main>**

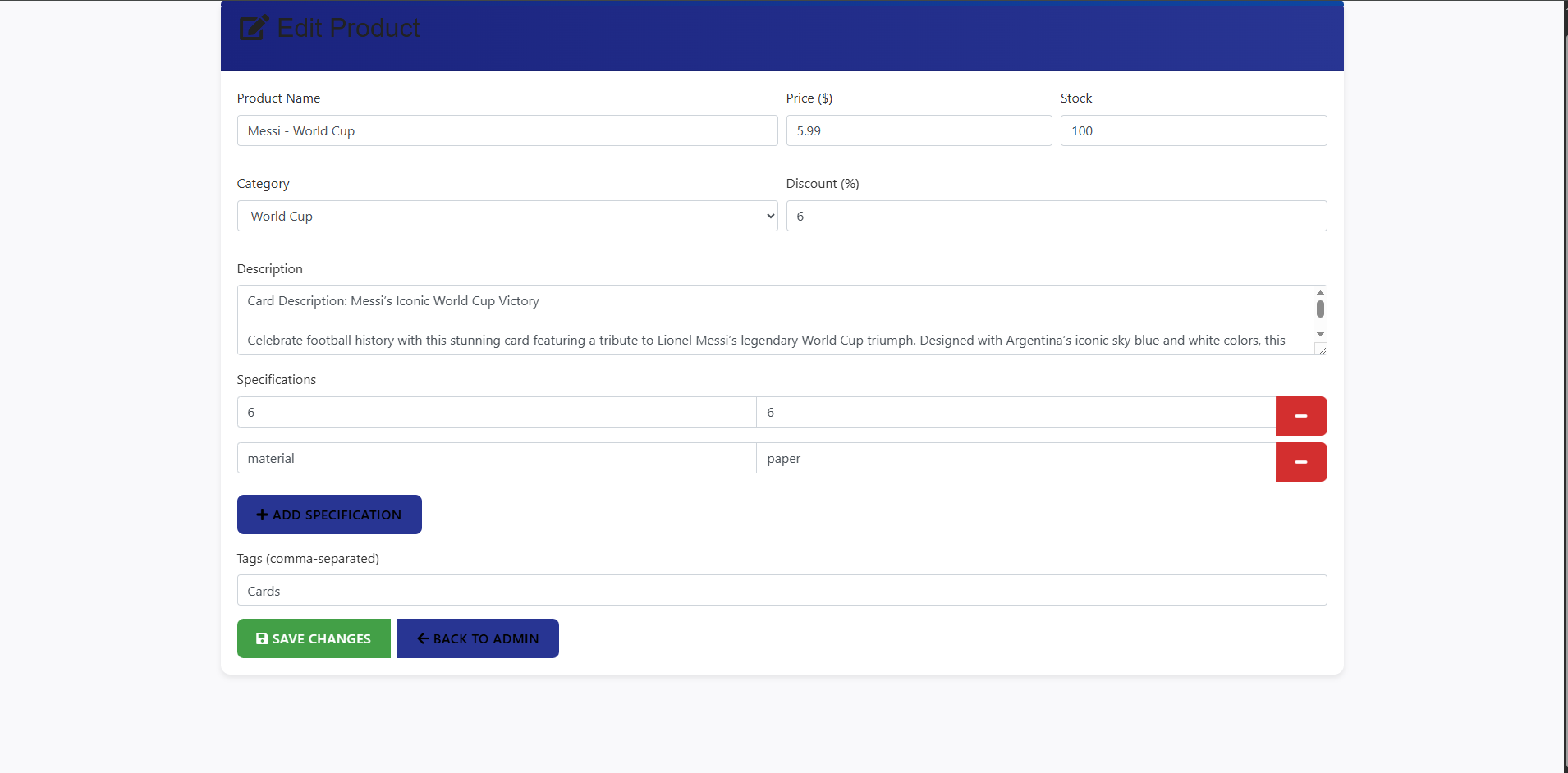
**Основни компоненти на страницата за любими продукти:**

* **Преглед на любими продукти: Страницата показва всички продукти, които потребителят е добавил към списъка си любими. За всеки продукт се показват само най-важните елементи като изображение, име, цена и бутон за добавяне в кошницата и опция за премахване.**
* **Съобщение, когато потребителят не е добавил продукт в любими.**
* **Бутон за добавяне на всички продукти в кошницата.**

**4.9 Страница за редактиране на продукт (Product-Edit Page)**

**Страница за редактиране и актуализиране на продукти, помагащ за мониторинга на сайта, като основните елементи са:**

* **Полета за заглавие, описание, категория, наличност на стоката и промоция**
* **Бутон за запазване на промените**
* **Поле за цена**
* **Избор на параметри и други описания, както и тагове**



5. Проектиране на функционалностите

Firebase и JavaScript отговарят за основните функции и за връзката между сървър потребител. JavaScript помага с всяко действие на потребителите, Firebase е за база данни, индентификация при регистриране и вход, съхранение на всякаква информация свързана с продуктите.

5.1 Динамичнота визуализация на продукти (JavaScript + Firebase Firestore)

В страницата с shop.html, JavaScript зарежда продуктите с Firestore от колекцията products и ги визуализира динамично там.

**function** loadAllProducts() {

db.collection("products").get()

.then(snapshot => {

productsData = [];

snapshot.forEach(doc => {

**const** p = doc.data();

productsData.push({

id: doc.id,

name: p.name,

price: p.price,

image: p.image,

category: p.category || "",

description: p.description || "",

discount: p.discount || 0

});

});

**function** renderProducts(products) – тази функция, прави възможно показването на съдържанието без необходимост от презареждане

5.1.1 Филтриране и търсене на продукти

**if** (searchInput) searchInput.addEventListener("input", applyFilters);

**if** (categoryFilter) categoryFilter.addEventListener("change", applyFilters);

**if** (sortFilter) sortFilter.addEventListener("change", applyFilters);

**if** (clearFiltersBtn) {

clearFiltersBtn.addEventListener("click", () => {

**if** (searchInput) searchInput.value = "";

**if** (categoryFilter) categoryFilter.value = "";

**if** (sortFilter) sortFilter.value = "";

applyFilters();

});

searchInput: input **– се аквтивира, когато се въвежда нещо в полето и след това функцията е** applyFilters **извикана.**

clearFiltersBtn **– изчиства филтрите.**

categoryFilter и sortFilter**:** change **– променя стойноста, след което извиква функцията** applyFilters.

5.1.3 Преглед на продукт (Product Details Page)

document.addEventListener("DOMContentLoaded", async () => {

**const** productId = **new** URLSearchParams(window.location.search).get("id");

**if** (!productId) {

window.location.href = "shop.html";

**return**;

}

**try** {

*// Извличане на данни за продукта от Firestore*

**const** productDoc = await db.collection("products").doc(productId).get();

**if** (!productDoc.exists) **throw** **new** Error("Продуктът не е намерен");

**const** productData = { id: productDoc.id, ...productDoc.data() };

*// Попълване на HTML елементи с извлечената информация*

**if** (elements.title) elements.title.textContent = productData.name;

**if** (elements.image) {

elements.image.src = productData.image;

elements.image.alt = productData.name;

}

**if** (elements.price) elements.price.textContent = formatCurrency(productData.price);

**if** (elements.category) elements.category.textContent = productData.category;

updateUI(productData);

} **catch** (error) {

console.error(error);

showNotification('error', "Неуспешно зареждане на продукта");

}

});

Така страницата зарежда даден продукт чрез id-то на продукта от url параметъра. Чрез JS и Firebase се показват данните от базата данни за продукта в страницата.

5.2 Функции за регистрирани потребители

**5.2.1 Регистрация и вход (Firebase Auth)**

**try** {

**let** cred = await auth.createUserWithEmailAndPassword(emailVal, passVal);

await cred.user.sendEmailVerification();

await db.collection("users").doc(cred.user.uid).set({

firstName: firstNameVal,

lastName: lastNameVal,

email: emailVal,

role: "user",

active: **true**,

createdAt: firebase.firestore.FieldValue.serverTimestamp()

});

* Въвеждане на имейл и парола в register.html.
* Чрез Firebase се създава нов потребител с функцията auth.createUserWithEmailAndPassword(emailVal, passVal)
* Изпращане на имейл за потвърждение с функцията cred.user.sendEmailVerification().
* Като потребителят се регистрира данните се записват в колекцията users.

5.2.2 Добавяне в количка

Този код отговаря за добавянето на продукт към кошницата, като го записва в users/{userId}/cart/{productId}.

**function** addToCart(productId, uid) {

*// Get product data from favorites*

db.collection("users").doc(uid).collection("favorites").doc(productId).get()

.then(doc => {

**if** (!doc.exists) **return**;

**const** product = doc.data();

*// Add to cart*

**return** db.collection("users").doc(uid).collection("cart").doc(productId).set({

...product,

quantity: 1

});

})

.then(() => {

updateCartCount();

alert("Item added to cart!");

})

.**catch**(err => {

console.error("Error adding to cart:", err);

alert("Error adding item to cart");

});

}

});

**5.2.3 Преглед и редакция на количка**

* Динамично зареждане на продуктите от под-колекцията cart.

**function** loadCart(uid) {

db.collection("users").doc(uid).collection("cart").get()

* Намаляне и увеличавене на броя продукти.

snapshot.forEach(doc => {

**let** product = doc.data();

**let** sub = (product.price || 0) \* (product.quantity || 1);

total += sub;

itemCount += (product.quantity || 1);

html += `

* Изтриване на продукт от количката.

**function** removeFromCart(cartId, uid) {

db.collection("users").doc(uid).collection("cart").doc(cartId).**delete**()

.then(() => {

loadCart(uid);

})

.**catch**(err => {

console.error("Error removing cart item:", err);

});

}

});

5.2.4 Добавяне в любими

Този код отговаря за добавянето на продукт в любими, като го записва в users/{userId}/favorites.

**function** addToFavorites(productId) {

db.collection("users").doc(user.uid).collection("favorites").doc(productId).set({});

}

**5.2.3 Преглед и редакция на количка**

* Динамично зареждане на продуктите от под-колекцията favorites.

**function** loadFavorites(uid) {

db.collection("users").doc(uid).collection("favorites").get()

* Добавяне в количката
* **function** addToCart(productId, uid) {
* Премахване от любимите

**function** removeFromFavorites(favoriteId, uid) {

db.collection("users").doc(uid).collection("favorites").doc(favoriteId).**delete**()

.then(() => {

loadFavorites(uid);

})

.**catch**(err => {

console.error("Error removing from favorites:", err);

});

}

**5.2.5 Checkout (завършване на поръчка)**

* Попълване на резюмето на поръчката

async **function** populateOrderSummary(uid) {

**try** {

**let** cartSnapshot = await db.collection("users").doc(uid).collection("cart").get();

**if** (cartSnapshot.empty) {

orderSummary.innerHTML = "<p>Your cart is empty.</p>";

**return**;

}

**let** total = 0;

**let** html = `

<table **class**="table">

<thead>

<tr>

<th>Product</th>

<th>Price</th>

<th>Qty</th>

<th>Subtotal</th>

</tr>

</thead>

<tbody>

`;

cartSnapshot.forEach(doc => {

**const** item = doc.data();

**let** st = (item.price || 0) \* (item.quantity || 1);

total += st;

html += `

<tr>

<td>${item.name}</td>

<td>${formatCurrency(item.price)}</td>

<td>${item.quantity}</td>

<td>${formatCurrency(st)}</td>

</tr>

`;

});

html += `

</tbody>

<tfoot>

<tr>

<th colspan="3" **class**="text-right">Total</th>

<th>${formatCurrency(total)}</th>

</tr>

</tfoot>

</table>

`;

orderSummary.innerHTML = html;

} **catch** (err) {

console.error("Error populating order summary:", err);

orderSummary.innerHTML = "<p>Error loading cart summary.</p>";

}

}

* Събиране на данни за доставката

**function** collectCheckoutData() {

**return** {

fullName: document.getElementById("fullName").value.trim(),

address: document.getElementById("shippingAddress").value.trim(),

city: document.getElementById("city").value.trim(),

postalCode: document.getElementById("zipCode").value.trim(),

country: document.getElementById("state").value.trim() *// Adjust as needed*

};

}

* Валидация на данни за поръчка

**function** collectCheckoutData() {

**return** {

fullName: document.getElementById("fullName").value.trim(),

address: document.getElementById("shippingAddress").value.trim(),

city: document.getElementById("city").value.trim(),

postalCode: document.getElementById("zipCode").value.trim(),

country: document.getElementById("state").value.trim() *// Adjust as needed*

};

}

* Създаване на поръчка

async **function** createOrder(uid, data) {

**let** cartSnapshot = await db.collection("users").doc(uid).collection("cart").get();

**if** (cartSnapshot.empty) **throw** **new** Error("Your cart is empty.");

**let** items = [];

**let** totalAmount = 0;

cartSnapshot.forEach(doc => {

**let** item = doc.data();

**let** st = (item.price || 0) \* (item.quantity || 1);

totalAmount += st;

items.push({

productId: doc.id,

name: item.name,

price: item.price,

quantity: item.quantity

});

});

**let** orderData = {

userId: uid,

fullName: data.fullName,

address: data.address,

city: data.city,

postalCode: data.postalCode,

country: data.country,

items,

totalAmount,

status: "Pending",

createdAt: firebase.firestore.FieldValue.serverTimestamp()

};

**let** orderRef = await db.collection("orders").add(orderData);

**return** orderRef.id;

}

* Изчистване на количката

async **function** clearCart(uid) {

**let** cartRef = db.collection("users").doc(uid).collection("cart");

**let** snapshot = await cartRef.get();

**if** (!snapshot.empty) {

**let** batch = db.batch();

snapshot.forEach(doc => {

batch.**delete**(doc.ref);

});

await batch.commit();

}

}

* Превключване между методите на плащане

**const** paymentMethods = document.getElementsByName('paymentMethod');

**const** bankTransferDetails = document.getElementById('bankTransferDetails');

**const** creditCardDetails = document.getElementById('creditCardDetails');

**const** cardInputs = creditCardDetails ? creditCardDetails.querySelectorAll('input') : [];

**function** togglePaymentDetails() {

**const** selectedRadio = document.querySelector('input[name="paymentMethod"]:checked');

**const** selectedMethod = selectedRadio ? selectedRadio.value : "";

**if** (bankTransferDetails) bankTransferDetails.style.display = 'none';

**if** (creditCardDetails) creditCardDetails.style.display = 'none';

cardInputs.forEach(input => {

input.required = **false**;

});

**if** (selectedMethod === 'bank' && bankTransferDetails) {

bankTransferDetails.style.display = 'block';

} **else** **if** (selectedMethod === 'card' && creditCardDetails) {

creditCardDetails.style.display = 'block';

cardInputs.forEach(input => {

input.required = **true**;

});

}

}

**if** (paymentMethods.length > 0) {

paymentMethods.forEach(radio => {

radio.addEventListener('change', togglePaymentDetails);

});

}

togglePaymentDetails();

5.2.6 Профил и история на поръчки

* показва имейл и кога се е регистрирал потребителя;

auth.onAuthStateChanged(user => {

**if** (user) {

*// Показване на имейл*

**if** (userEmail) userEmail.textContent = user.email;

*// Показване на дата на регистрация*

**if** (memberSince) {

**const** creationDate = **new** Date(user.metadata.creationTime);

memberSince.textContent = creationDate.toLocaleDateString('en-US', {

year: 'numeric',

month: 'long',

day: 'numeric'

});

}

* показва актуалните поръчки от колекцията orders

**if** (ordersContainer) {

loadUserOrders(user.uid);

}

} **else** {

window.location.href = 'login.html';

}

});

*// Зареждане на последни поръчки*

**function** loadUserOrders(uid) {

ordersContainer.innerHTML = `<div **class**="loading-spinner"><i **class**="fas fa-spinner fa-spin"></i> Loading orders...</div>`;

db.collection("orders")

.where("userId", "==", uid)

.orderBy("createdAt", "desc")

.limit(5)

.get()

.then(snapshot => {

**if** (snapshot.empty) {

ordersContainer.innerHTML = `<div **class**="alert alert-info"><i **class**="fas fa-info-circle"></i> You have no orders yet.</div>`;

**return**;

}

**let** html = `

<div **class**="table-responsive">

<table **class**="table table-hover">

<thead **class**="thead-light">

<tr>

<th>Order ID</th>

<th>Date</th>

<th>Total</th>

<th>Status</th>

</tr>

</thead>

<tbody>

`;

snapshot.forEach(doc => {

**const** order = doc.data();

**const** date = order.createdAt ? order.createdAt.toDate().toLocaleString() : "N/A";

**const** status = order.status || "Processing";

**const** statusClass = {

'Processing': 'text-warning',

'Shipped': 'text-info',

'Delivered': 'text-success',

'Cancelled': 'text-danger'

}[status] || 'text-secondary';

html += `

<tr>

<td><small>${doc.id}</small></td>

<td>${date}</td>

<td>${formatCurrency(order.totalAmount)}</td>

<td><span **class**="badge ${statusClass}">${status}</span></td>

</tr>

`;

});

html += `</tbody></table></div>`;

ordersContainer.innerHTML = html;

})

.**catch**(err => {

console.error("Error loading orders:", err);

ordersContainer.innerHTML = `<div **class**="alert alert-danger"><i **class**="fas fa-exclamation-circle"></i> Error loading orders.</div>`;

});

}

* изпраще на имейл за смяна на паролата

**if** (resetPasswordForm) {

resetPasswordForm.addEventListener("submit", async e => {

e.preventDefault();

**const** resetEmail = document.getElementById("resetEmail").value.trim();

**if** (!resetEmail) {

alert("Enter your email.");

**return**;

}

**try** {

await auth.sendPasswordResetEmail(resetEmail);

alert("Password reset email sent. Check your inbox.");

resetPasswordForm.reset();

} **catch** (err) {

console.error("Error sending reset email:", err);

alert("Error sending password reset email. Check the email address.");

}

});

}

});

* смяна на паролата след удостоверение

**if** (changePasswordForm) {

changePasswordForm.addEventListener("submit", async e => {

e.preventDefault();

**const** currentPassword = document.getElementById("currentPassword").value.trim();

**const** newPassword = document.getElementById("newPassword").value.trim();

**const** confirmNewPassword = document.getElementById("confirmNewPassword").value.trim();

**if** (newPassword !== confirmNewPassword) {

alert("New passwords do not match.");

**return**;

}

**if** (newPassword.length < 6) {

alert("New password must be at least 6 characters.");

**return**;

}

**const** user = auth.currentUser;

**if** (!user) {

alert("No user logged in.");

**return**;

}

**const** cred = firebase.auth.EmailAuthProvider.credential(user.email, currentPassword);

**try** {

await user.reauthenticateWithCredential(cred);

await user.updatePassword(newPassword);

alert("Password updated successfully.");

changePasswordForm.reset();

} **catch** (err) {

console.error("Error changing password:", err);

alert("Error updating password. Check your current password or re-login.");

}

});

}

5.3 Функции за администратори

5.3.1 Достъп до admin панел

Само потребител с роля admin може да има достъп до админстаторския панел

auth.onAuthStateChanged(user => {

**if** (!user) {

window.location.href = "login.html";

} **else** {

db.collection("users").doc(user.uid).get().then(doc => {

**if** (!doc.exists || doc.data().role !== "admin") {

alert("Access denied. Admin only.");

window.location.href = "index.html";

}

});

}

});

5.3.2 Навигация в админ панела

**if** (adminPanelToggle) {

adminPanelToggle.addEventListener("click", e => {

e.preventDefault();

sidebar.classList.toggle("active");

sidebarTrigger.classList.toggle("active");

});

5.3.3 Управление на продукти

await db.collection("products").add({

name: nameVal,

price: priceVal,

stock: stockVal,

discount: discount,

category: catVal,

image: imageLink,

description: descVal,

specifications: specs,

tags: tagsArray,

createdAt: firebase.firestore.FieldValue.serverTimestamp()

});

Визуализцията на продуктите става динамично:

db.collection("products").get().then(snapshot => {

snapshot.forEach(doc => {

**const** p = doc.data();

*// Визуализация в HTML*

});

});

За единично и групово изтриване:

bulkDeleteBtn.addEventListener("click", () => {

**let** batch = db.batch();

selected.forEach(id => {

**let** ref = db.collection("products").doc(id);

batch.**delete**(ref);

});

batch.commit().then(() => {

alert("Selected products deleted.");

fetchProducts();

});

});

5.3.4 Управление на категории

Админстратори могат да добават категории в shop.html

await db.collection("categories").add({

name: name,

icon: icon,

description: description,

createdAt: firebase.firestore.FieldValue.serverTimestamp()

});

5.3.5 Управление на поръчка

Администратора отговаря за статуса на поръчката

db.collection("orders").doc(orderId).update({ status: newStatus })

5.3.6 Редактиране на продукт

document.addEventListener("DOMContentLoaded", () => {

**const** form = document.getElementById("addProductForm");

**if** (!form) **return**;

*// Проверка за автентикация и администраторски права*

auth.onAuthStateChanged(user => {

**if** (!user) {

window.location.href = 'login.html';

**return**;

}

db.collection('users').doc(user.uid).get()

.then(doc => {

**if** (!doc.exists || doc.data().role !== 'admin') {

alert('Access denied. Admin only.');

window.location.href = 'index.html';

**return**;

}

});

});

*// Обработване на формуляра при изпращане*

form.addEventListener('submit', async (e) => {

e.preventDefault();

**const** newProduct = {

name: document.getElementById('productName').value.trim(),

price: parseFloat(document.getElementById('productPrice').value),

stock: parseInt(document.getElementById('productStock').value),

category: document.getElementById('productCategory').value,

discount: parseFloat(document.getElementById('productDiscount').value) || 0,

description: document.getElementById('productDescription').value.trim(),

tags: document.getElementById('productTags').value.split(',').map(tag => tag.trim()).filter(tag => tag),

createdAt: firebase.firestore.FieldValue.serverTimestamp(),

specifications: {}

};

*// Събиране на спецификации*

**const** specNames = document.getElementsByName('specName[]');

**const** specValues = document.getElementsByName('specValue[]');

**for** (**let** i = 0; i < specNames.length; i++) {

**const** name = specNames[i].value.trim();

**const** value = specValues[i].value.trim();

**if** (name && value) {

newProduct.specifications[name] = value;

}

}

**try** {

*// Добавяне на продукта в базата данни*

await db.collection('products').add(newProduct);

alert('Product added successfully!');

window.location.href = 'admin.html#manageProducts';

} **catch** (error) {

console.error('Error adding product:', error);

alert('Error adding product: ' + error.message);

}

});

});

5.4 Допълнителни функции

Тук ще разгледаме глобални настройки за Firebase и допълни спомагателни функции за сайта

5.4.1 Инициализация на Firebase - Firebase Firestore, Authentication и Storage

**const** firebaseConfig = {

apiKey: "AIzaSyD7SisKsPIZhBioYjJ2MEK7iSXIXChwND4",

authDomain: "onlainshop-4382c.firebaseapp.com",

projectId: "onlainshop-4382c",

storageBucket: "onlainshop-4382c.firebasestorage.app",

messagingSenderId: "589934123101",

appId: "1:589934123101:web:baaf42e92852ccdb2d55b4",

measurementId: "G-WGC9HX2FKF"

};

firebase.initializeApp(firebaseConfig);

**const** db = firebase.firestore();

**const** auth = firebase.auth();

**const** storage = firebase.storage();

5.4.2 Актуализиране на броя в количката и любими продукти

**function** updateCartCount() {

**const** user = auth.currentUser;

**const** cartCountElem = document.getElementById("cartCount");

**if** (!cartCountElem) **return**;

**if** (!user) {

cartCountElem.innerText = "0";

**return**;

}

db.collection("users").doc(user.uid).collection("cart").get()

.then(snapshot => {

cartCountElem.innerText = snapshot.size;

})

.**catch**(err => {

console.error("Error updating cart count:", err);

});

}

**function** updateFavoritesCount() {

**const** user = auth.currentUser;

**const** favElem = document.getElementById("favoritesCount");

**if** (!favElem) **return**;

**if** (!user) {

favElem.innerText = "0";

**return**;

}

db.collection("users").doc(user.uid).collection("favorites").get()

.then(snapshot => {

favElem.innerText = snapshot.size;

})

.**catch**(err => {

console.error("Error updating favorites count:", err);

});

}

5.4.3 Актуализиране на навигационните линкове

**function** updateCartCount() {

**const** user = auth.currentUser;

**const** cartCountElem = document.getElementById("cartCount");

**if** (!cartCountElem) **return**;

**if** (!user) {

cartCountElem.innerText = "0";

**return**;

}

db.collection("users").doc(user.uid).collection("cart").get()

.then(snapshot => {

cartCountElem.innerText = snapshot.size;

})

.**catch**(err => {

console.error("Error updating cart count:", err);

});

}

**function** updateFavoritesCount() {

**const** user = auth.currentUser;

**const** favElem = document.getElementById("favoritesCount");

**if** (!favElem) **return**;

**if** (!user) {

favElem.innerText = "0";

**return**;

}

db.collection("users").doc(user.uid).collection("favorites").get()

.then(snapshot => {

favElem.innerText = snapshot.size;

})

.**catch**(err => {

console.error("Error updating favorites count:", err);

});

}

5.4.4Функция за изход от профила

**function** logoutUser() {

auth.signOut()

.then(() => {

alert("You have been logged out.");

window.location.href = "login.html";

})

.**catch**(err => {

console.error("Logout error:", err);

alert("Error logging out. Try again.");

});

}

5.4.5 Актуализиране на промени в състоянието на индентификацията

auth.onAuthStateChanged(user => {

**if** (user) {

updateCartCount();

updateFavoritesCount();

updateNavLinks(user.uid);

console.log("User logged in:", user.email);

} **else** {

console.log("No user logged in");

**const** adminNavItem = document.getElementById("adminNavItem");

**if** (adminNavItem) adminNavItem.style.display = "none";

**const** profileNavItem = document.getElementById("profileNavItem");

**if** (profileNavItem) profileNavItem.style.display = "none";

**const** logoutLink = document.getElementById("logoutLink");

**if** (logoutLink) logoutLink.style.display = "none";

**const** loginLink = document.getElementById("loginLink");

**if** (loginLink) loginLink.style.display = "inline-block";

**const** registerLink = document.getElementById("registerLink");

**if** (registerLink) registerLink.style.display = "inline-block";

}

});

6. Връзката между компонентите **HTML**, **CSS**, **JavaScript** и **Firebase**

A diagram of a software company

AI-generated content may be incorrect.

**Обяснение на компонентите:**

* **Потребител**: Този, който ползва страницата през браузър.
* **HTML**: основната структута на сайта: заглавия, бутони и т.н
* **CSS**: Оформяме стиловия вид на страницата.
* **JavaScript**: Отговаря за диманичната функционалност.
* **Firebase**: Осигурява сървърна поддръжка:
  + Authentication – регистрация, вход и роли.
  + Firestore – продукти, поръчки и потребители.
  + Storage – изображения на продуктите.
  1. HTMlL се свърза чрез таговете <link> и <script> с Css и JavaScript.

<link> - отговаря за свързване с css-файловете

<script>- отговаря за свързване с JavaScript

**<link** rel="stylesheet" href="css/style.base.css" **/>**

**<link** rel="stylesheet" href="css/style.auth.alternative.css" **/>**

**<link** rel="stylesheet" href="css/style.register.css" **/>**

**<link** rel="stylesheet" href="css/style.register.advanced.css" **/>**

**<script** src="js/global.js" defer**></script>**

*<!-- Page-specific JS -->*

**<script** src="js/register.js" defer**></script>**

* 1. Как JavaScript общува с Firebase

Ето един пример, как потребителят: добавя продукт и се записва в базата данни:

await db.collection("products").add({

name: nameVal,

price: priceVal,

stock: stockVal,

discount: discount,

category: catVal,

image: imageLink,

description: descVal,

specifications: specs,

tags: tagsArray,

createdAt: firebase.firestore.FieldValue.serverTimestamp()

});