

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Specijalistički diplomski stručni studij Primijenjeno računarstvo

KATARINA KOLAK

Z A V R Š N I R A D

Online trgovina proizvoda za uljepšavanje

Split, rujan 2023.

SVEUČILIŠTE U SPLITU
SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Specijalistički diplomski stručni studij Primijenjeno računarstvo

Predmet: Napredno programiranje web aplikacija

Z A V R Š N I R A D

Kandidat: Katarina Kolak

Naslov rada: Online trgovina proizvoda za uljepšavanje

Mentor: Marina Rodić, viši predavač

Split, rujan 2023.

Sadržaj

Sažetak	1
Summary	2
1. Uvod	3
2. Tehnologije	5
2.1. MongoDB	5
2.2. Node.js	6
2.3. React	6
2.4. Cascading Style Sheets	8
2.5. Mongoose	8
2.6. Express	9
2.7. Postman	9
3. Implementacija aplikacije	11
3.1. Modeli	11
3.2. Uloge	13
3.3. Korisničko sučelje za administratora	15
3.4. Korisničko sučelje za registriranog korisnika	31
3.5. Korisničko sučelje za neregistriranog korisnika	58
4. Zaključak	53
Literatura	54

Sažetak

Cilj diplomskog rada je izrada web aplikacije za kozmetičke proizvode. Razvojem i napretkom tehnologije mijenja se način života i navike ljudi. Web trgovine postaju sve zastupljeniji oblik kupovine gdje ljudi mogu iz udobnosti svoga doma pretraživati različite proizvode i kupovati iste. Web aplikacija za kozmetiku upravo bi to omogućila korisnicima. Bogata ponuda kozmetičkih proizvoda različitih brendova olakšala bi korisniku odlazak u trgovinu te bi korisnik pomoću nekoliko klikova na stranici kupio željeni proizvod.

Aplikacija je implementirana kombinacijom React okvira i Node.js poslužiteljskog okruženja. Za korisnički dio aplikacije korišten je React, a za poslužiteljsku stranu Node.js okruženje te Express okvir. Aplikacija se temelji na različitim ulogama koje prate različite funkcionalnosti. Administratorska uloga je uloga s najvećim pravima, u koja spadaju dodavanje, brisanje i uređivanje korisnika, proizvoda, upravljanje statusima narudžbe. Nakon navedene uloge slijedi registrirani korisnik koji može pregledati ponudu, stvarati listu omiljenih proizvoda, dodavati proizvode u košaricu, obaviti kupnju proizvoda, pratiti tijek narudžbe. Na koncu, gost je uloga s najmanjim pravima te može samo pregledati ponudu.

Ključne riječi: Express, narudžbe, React, web trgovina

Summary

Online beauty shop

The purpose of the graduation work is to create a web application for cosmetic products. With the development and progress of technology, people's way of life and habits are also changing. Online shops are becoming an increasingly common form of shopping, where people can browse and buy different products from the comfort of their home. The online shop for cosmetics would allow users to do just that. A rich offer of cosmetic products of distinct brands would make it easier for the user to go to the store and the user would buy the desired product with a few clicks on the page.

The application is implemented using a combination of the React framework and the Node.js server environment. React is used for the frontend, on the other side Node.js and Express were used for the backend. The application is based on different roles that follow different functionalities. The administrator role is the role with the most rights, which includes adding, deleting and editing users, products, managing order statuses. The next role is a registered user who can view the offer, create a wish list, add products to the cart, purchase products and add track the progress of the order. Finally, a guest is the role with the least rights and can only view the offer.

Keywords: Express, online shop, orders, React

1. Uvod

Diplomski rad prikazuje jednostavnu web trgovinu za kozmetiku. Motivacija za odabir teme je utjecaj web aplikacija i internet kupovine u današnjem svijetu. Ljudi sve češće biraju navedeni oblik kupovine kako bi uštedjeli vrijeme, izbjegli potencijalne gužve u prometu ili na blagajni. U situacijama kada čovjek jednostavno nije u mogućnosti fizički otići kupiti potrebne proizvode, ovakav način kupnje je jako koristan.

Za implementaciju aplikacije korištena je MongoDB baza podataka, za klijentsku stranu (eng. *frontend*) korištena je React biblioteka te za poslužiteljsku stranu (eng. *backend*) korišteno je Node.js poslužiteljsko okruženje. Aplikacija je uređena i oblikovana pomoću CSS (eng. *Cascading Style Sheets*) jezika.

Web aplikacija temelji se na tri osnovne uloge: administrator, registrirani korisnik i gost. Administrator je uloga koja ima najveće funkcionalnosti na stranici. Stvara ponudu aplikacije, odnosno dodaje, uređuje i briše proizvode, ali i korisnike, mijenja status dostave naručenih proizvoda, uređuje cijene i popuste u aplikaciji. Registrirani korisnik je korisnik s izrađenim korisničkim računom u aplikaciji te kao takav može pregledati, pretraživati, filtrirati ponudu, dodavati proizvode u listu omiljenih proizvoda ili u košaricu, uklanjati proizvod iz liste omiljenih proizvoda ili iz košarice, kupiti proizvod, pratiti status dostave proizvoda, potvrditi primitak naručenog proizvoda, mijenjati osobne podatke. Gost ima minimalne funkcionalnosti u aplikaciji. Može samo pregledati, pretraživati i filtrirati ponudu, ali nudi mu se i mogućnost izrade korisničkog računa kako bi imao veće funkcionalnosti.

U poglavlju 2 opisuju se korištene tehnologije koje su bile potrebne za implementaciju aplikacije. Kroz kratke opise pojasnit će se osnovna namjena i specifičnosti svake od korištenih tehnologija.

Poglavlje 3 detaljno prikazuje važnije funkcionalnosti aplikacije. Opisi funkcionalnosti praćeni su slikom ili kôdom koji je vezan za određenu funkcionalnost. Pošto nije moguće prikazati sve funkcionalnosti koje su implementirane u aplikaciji, neke koje su standardne neće biti detaljno opisane kao što su registracija, prijava i odjava u aplikaciju. U ovakvim slučajevima

spomenut će se ono što je nestandardno ili što je karakteristično za jezik u kojem je aplikacija implementirana.

Nakon poglavlja 3 slijedi zaključak same aplikacije, ali i osvrt na pisanje i implementiranje aplikacije, eventualne poteškoće, potencijalna poboljšanja koja su moguća u aplikaciji.

2. Tehnologije

Ovo poglavlje sadrži opise tehnologija koje su omogućile implementaciju aplikacije. U nastavku su posebno obrađene tehnologije za bazu podataka, za klijentsku, za poslužiteljsku stranu aplikacije te tehnologija za dizajn i oblikovanje.

2.1. MongoDB

MongoDB je NoSQL (eng. *Not only SQL*) program otvorenog koda za upravljanje bazom podataka. NoSQL je alternativa tradicionalnim relacijskim bazama podataka [1]. Koristan je za rad s velikim skupovima distribuiranih podataka. Umjesto tablica i redaka koje su karakteristične za relacijske baze, MongoDB koristi arhitekturu koja se sastoji od zbirki (eng. *collections*) i dokumenata. Kolekcije su ekvivalent tablicama i sadrže dokumente. Dokumenti sadrže podatke koje korisnik pohranjuje, a prikazani su u obliku parova polje i vrijednost. Polja su ekvivalent stupcima relacijske baze, a vrijednosti mogu biti različite vrste podataka. Dokument sadrži `_id` polje koje je jedinstvena identifikacija dokumenta automatski kreirana od strane MongoDB-a, kojom se može referencirati na konkretni dokument.

Navedena baza podataka pogodna je za integraciju ili skladištenje velike količine podataka, a performanse je moguće poboljšati dodavanjem indeksa. U nastavku su vidljive značajne prednosti MongoDB baze podataka:

- Bez sheme – ne zahtijeva unaprijed definirane sheme i tako povećava fleksibilnost.
- Skalabilnost – ova značajka je bitna za velike organizacije koje pokreću velike podatkovne aplikacije
- Podrška treće strane – podržava mehanizme za pohranu i priključne API-je pomoću kojih treća strana može razviti vlastite mehanizme pohrane.

2.2. Node.js

Node.js je besplatno, poslužiteljsko okruženje otvorenog koda [2]. Može se pokrenuti na različitim operacijskim sustavima. Pogodan je za rad nad datotekama, bazom podataka, različitim objektima. Neke od funkcionalnosti ovoga jezika su:

- Podržava različite operacije nad bazom podataka kao što su: dodavanje, čitanje, brisanje, uređivanje
- Moguće je koristiti i datoteke nad kojima podržava sljedeće operacije: kreiranje, čitanje, otvaranje, mijenjanje, brisanje i zatvaranje
- Asinkronost - poslužitelji nikad ne čekaju podatke API-ja (eng. *Application Programming Interface*) već se izravno prelazi na sljedeći API [3].

Node.js okruženje ima dosta prednosti zbog kojih se koristi. Neke od njih su: brzo izvršavanje naredbi, brzo pokretanje, dostupnost mnogih paketa koji se mogu instalirati i koristiti u projektima, učinkovit je za izradu web aplikacija s velikim brojem podataka koje rade u realnom vremenu, asinkron je, dobra je sinkroniziranost između poslužitelja i klijenta, razvojni programeri koji koriste JavaScript lako se prilagode ovom okruženju, kompatibilnost s drugim platformama.

2.3. React

React je okvir i biblioteka otvorenog koda koju je razvio Facebook [4]. Omogućuje učinkovitu i efikasnu izvedbu web aplikacija i korisničkih sučelja. Programeri stvaraju cjelovitu aplikaciju kreirajući pojedinačne komponente koje naposljetku rezultiraju dinamičnim aplikacijama. Nudi više funkcionalnosti te manje koda od JavaScripta [5]. React ima malu krivulju učenja pošto kombinira znanja HTML-a (eng. *HyperText Markup Language*) i JavaScript koncepte uz neke novitete. Glavni cilj React biblioteke je razviti brza korisnička sučelja.

Objektni model dokumenata (eng. *The Document Object Model*) ili DOM je prikaz podataka o objektu koji čine strukturu i sadržaj dokumenata na webu. [6] Manipulacija DOM-om može biti spora, pošto većina JavaScript okvira ažurira DOM više nego što je potrebno. Kad je potrebno ažuriranje jedne stavke većina okvira ažurirat će ponovno i sve ostale stavke, umjesto da promijene jednu, a druge ostave onakve kakve su bile prije. Ovo ne predstavlja velik problem web preglednicima, ali neučinkovito ažuriranje može biti problem. Kako bi se navedeni problem riješio React uvodi virtualni DOM.

U Reactu, za svaki DOM objekt postoji odgovarajući virtualni DOM objekt (predstavlja kopiju DOM objekta). Virtualni DOM ne može izravno promijeniti ono što je prikazano na zaslonu, ali ima ista svojstva kao stvarni DOM. Ovakva manipulacija virtualnim DOM-om je brža za razliku od prethodne. Cijeli virtualni DOM se ažurira, ali se spremi snimka virtualnog DOM-a prije ažuriranja. Nakon ažuriranja, uspoređuje se novi virtualni DOM s prethodno spremljenom snimkom. Promijenjeni objekti ažuriraju se na stvarnom DOM-u i tek onda je promjena vidljiva na zaslonu. Na ovaj način, React mijenja samo potrebne dijelove DOM-a, a ne cijeli DOM.

Aplikacija s jednom stranicom (eng. *Single Page Application*) ili SPA postaju sve popularnije među programerima. [7] React pruža programerima alate koji su potrebni za izradu brzih i učinkovitih jednostraničnih aplikacija. Ovakve aplikacije trebaju ažurirati samo trenutno prikazanu stranicu kako bi napravile neki korisnički zahtjev. Resursi se dinamički učitavaju i dodaju na stranicu prema potrebi, a korisnik se može pomoću poveznica ili drugih interaktivnih elemenata pomjerati po stranici. Ovakav koncept smanjuje potrebu za ponovnim i potpunim učitavanjem sadržaja. Početno vrijeme učitavanja ovakvih stranica može biti sporije, ali brzo korisničko iskustvo nakon učitavanja velika je prednost. Na ovaj način manja je opterećenost poslužitelja, a samo održavanje i ažuriranje kôda je pojednostavljeno.

2.4. Cascading Style Sheets

Cascading Style Sheets (CSS) je stilski jezik koji se koristi za prikaz dokumenta napisanog u HTML-u. Opisuje na koji način se elementi trebaju prikazati na ekranu [8]. Pomoću CSS-a moguće je stilizirati i rasporediti elemente web stranice kao što su određivanje boje, fonta, veličine, razmaka, pozadine i ostalih svojstava koji utječu na dizajn same stranice. Stil se formira unutar dokumenta koji ima .css ekstenziju. Za definiranje stila potrebno je označiti selektore unutar HTML dokumenta (označava nad kojim se elementom definira stil) te nakon toga slijedi deklaracijski blok [9]. Deklaracijski blok sastoji se od jedne ili više deklaracija odvojenih točkom i zarezom te svaka deklaracija sadrži CSS svojstvo i vrijednost odvojene zarezom.

2.5. Mongoose

Mongoose je biblioteka za modeliranje objektnih podataka za MongoDB i Node.js. [10] Omogućuje provjeru valjanosti sheme, upravlja odnosima podataka te objekte iz Node.js kôda preslikava u MongoDB zapise. Mongoose shema definira cjelokupnu strukturu ili oblik dokumenta. Modeli su konstruktori višeg reda koji pomoću sheme stvaraju instancu dokumenta koja je ekvivalent zapisima u bazi podataka. Kreiraju se tako da se definiraju polja zajedno sa zahtjevima za provjeru valjanosti i zadanim vrijednostima. Sheme se nakon kreiranja preslikaju u modele, a model se može koristiti za stvaranje, ažuriranje ili brisanje objekata. Mongoose nudi različite provjere pomoću kojih se može definirati dozvoljeni raspon, obavezno polje, najmanja/najveća moguća vrijednost polja i druge potrebne opcije.

2.6. Express

Express je minimalan i fleksibilan Node.js okvir koji pruža širok spektar značajki za razvoj aplikacija. [11] Karakterizira ga brz razvoj aplikacija. Neke od prednosti Express okvira su:

- Postavljanje međuprograma (eng. *middleware*) za odgovaranje na HTTP zahtjeve i obradu zahtjeva
- Definiranje tablice usmjeravanja i određivanje načina na koji aplikacija odgovara na zahtjev klijenta prema krajnjoj točki. Krajnja točka sastoji se od putanje i HTTP metode (kao što su GET, POST, PUT, DELETE). Svaka putanja sadrži. Tablica usmjeravanja koristi se za izvođenje različitih radnji na temelju HTTP metode i URL-a
- Dinamičko prikazivanje HTML stranica na temelju prosljeđivanja argumenata predlošcima.

Express okvir je minimalistički okvir, ali programeri su stvorili različite međuprogramske pakete za rješavanje problema prilikom razvoja web aplikacija. Tako postoje biblioteke za rad s kolačićima (eng. *cookies*), korisničkim prijavama, URL (eng. *Uniform Resource Locator*) parametrima, sigurnosnim zaglavljima i ostale korisne biblioteke. Navedeni okvir olakšava razvoj aplikacija, štedi vrijeme kodiranja i aplikacije čini učinkovitima. Može se koristiti za razvoj jedne stranice, više stranica ili hibridnih aplikacija.

2.7. Postman

Postman je jedan od najpopularnijih alata za testiranje aplikacija, a služi za testiranje API-ja. [12] To je grafičko sučelje za slanje i testiranje HTTP zahtjeva i odgovora. Može se koristiti za dizajn, izradu, testiranje, modificiranje ili dokumentiranje API-ja. Postman je praktičan i jednostavan alat za korištenje, a ovo su neke prednosti alata:

- Pristupačnost - može se koristiti bilo gdje i bilo kada
- Stvaranje i korištenje zbirke API poziva - moguće je organizirati pakete testova u mape i podmape
- Testiranje - omogućuje testiranje kontrolnih točaka te provjeru statusa testa

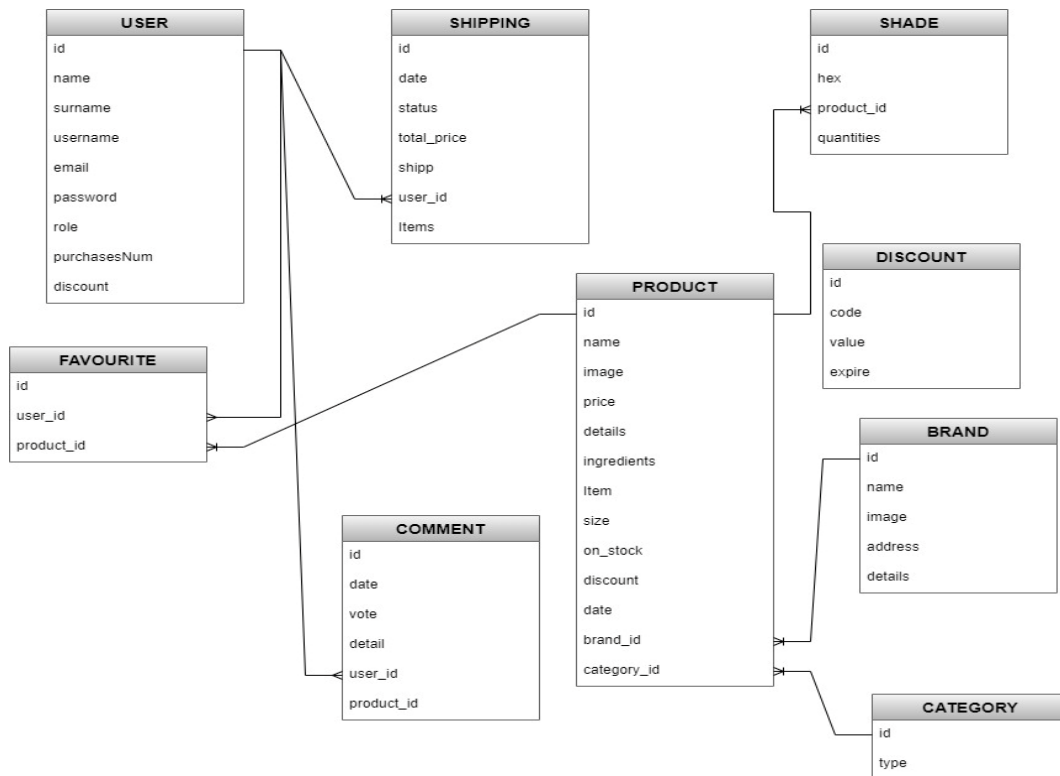
- Otklanjanje grešaka - konzola pomaže kod uklanjanja grešaka prilikom testiranja API-ja na način da generira izvješća u slučaju neuspjeha
- Kontinuirana integracija - može se integrirati s mnogim vrhunskim alatima, korisnim aplikacijama i platformama
- Podrška - ono što većina programera očekuje od nekog alata je podrška i dobra zajednica. Ima snažnu podršku i zajednicu kako bi poboljšao svoju uporabu na raznim platformama
- Automatizirano testiranje - testovi se mogu izvoditi u nekoliko iteracija pomoću posebnih programa.

3. Implementacija aplikacije

U ovom poglavlju prikazana je i opisana sama implementacija aplikacije. Kroz poglavlje su objašnjene glavne funkcionalnosti i način rada aplikacije uz pomoć slika, opisa te popratnog kôda.

3.1. Modeli

Kako bi stranica logički funkcionirala i prikazivala određene podatke bilo je potrebno definirati format podataka koji će se pohraniti u kolekcije MongoDB baze podataka. To je postignuto korištenjem modela iz paketa Mongoose. Svaki definirani model sadrži polja i vrijednosti koji su u bazu spremljeni u obliku dokumenta unutar pripadne kolekcije modela. Model sadrži primarni ključ koji je ujedno i jedinstveni identifikator modela, a on se može pojaviti u nekom drugom modelu kao strani ključ. Poslužiteljska strana aplikacije izvodi odgovarajuće operacije nad modelima kako bi se podaci spremali, dohvaćali, brisali ili dodavali u kolekcije. Za učinkovit rad aplikacije bilo je potrebno definirati 9 modela podataka, neki od tih modela su međusobno povezani pomoću stranih ključeva, a neki nisu. Shema modela vidljiva je na Slici 1.



Slika 1 – Prikaz sheme modela

Slika 1 prikazuje modele te njihova pripadna polja koja svaki model sadrži. Način na koji je definiran model Brand prikazan je u Ispisu 1, a na sličan način samo s drugim poljima su napisani i ostali modeli.

```

const mongoose = require('mongoose');
const brandModel = new mongoose.Schema({
  name:{type:String, required: true},
  image:{type:String},
  address:{type:String, required: true},
  details: {type:String}
})
module.exports = mongoose.model('Brand', brandModel, 'brands');
  
```

Ispis 1 – Primjer modela Brand

U Ispisu 1 vidljivo je na koji način se definiraju pojedini modeli. Potrebno je definirati ime modela i pripadna polja modela. Polja se definiraju tako da se navodi ime određenog polja i tip polja (postoje različiti tipovi kao što su string, broj, datum, niz i slično). Osim navedenog moguće je dodati i neka dodatna svojstva kao što su je li polje obavezno, može li biti prazno i druge opcije.

3.2. Uloge

Funkcionalnosti web trgovine ovise u ulozi koju korisnik ima u aplikaciji. Gost ima najmanje ovlasti u aplikaciji. Kao takav može samo pregledati, pretraživati i filtrirati proizvode dostupne u aplikaciji. Za njega nema evidentiranih podataka u bazi podataka, ali ima mogućnost registracije prilikom koje se njegovi korisnički podaci spremaju u bazu podataka te on dobiva ulogu korisnika.

Korisnik ima nešto veće funkcionalnosti u aplikaciji, odnosno on može u cijelosti obaviti kupnju u web trgovini. Osim funkcionalnosti koje su navedene za gosta, registrirani korisnik može: dodavati, uklanjati proizvode iz košarice (u slučaju da proizvod ima više nijansi može odabrati koju želi dodati), stvarati listu omiljenih proizvoda (kao i ukloniti proizvod iz navedene liste), mijenjati podatke korisničkog računa, ostvariti popust na svaku okruglu kupnju, koristiti popuste kojima je valjan datum trajanja odobrene od strane administratora, platiti proizvod, pratiti tijek dostave proizvoda, potvrditi primitak proizvoda na dostavi te ako je kupio proizvod može ostaviti recenziju u obliku ocjene (zvjezdica) i/ili komentara.

Administrator ima najveća prava u aplikaciji i upravlja cijelom aplikacijom. U njegove funkcionalnosti spadaju: dodavanje, uređivanje, brisanje korisnika i proizvoda, dodavanja aktualnih kodova za popust, uvid u korisnike i proizvode, mijenjanje statusa proizvoda iz dostave, određivanje postotka sniženja proizvoda i pregled statistike najprodavanijih proizvoda u danu i tjednu. Pošto određeni proizvodi mogu imati različite nijanse, administrator može dodavati i brisati nijansu proizvoda.

Kako bi se u aplikaciji znalo koju ulogu ima trenutno prijavljeni korisnik, nakon prijave osnovni podaci korisnika (identifikacijska oznaka, ime i uloga) spremaju se u prostor lokalne pohrane. Uz navedene podatke sprema se i JWT (eng. *JSON Web Token*) koji sadrži potrebne informacije o korisniku, a generira se prilikom uspješne prijave korisnika u aplikaciju. Napisane su posebne funkcije za generiranje JWT tokena pomoću tajnog ključa i za provjeru valjanosti istog. Na ovaj način može se lako saznati koji korisnik je trenutno prijavljen, koja je njegova uloga i kojim funkcionalnostima može pristupiti. Ispis 2 prikazuje dio kôda pomoću kojeg se provjerava je li korisnik prijavljen te ako je, posjeduje li administratorsku ulogu.

```
const role=JSON.parse(localStorage.getItem("user")).role;
if (role !== "admin"){
    window.location.href = '/';
}
```

Ispis 2 – Provjera uloge korisnika

Na isti način se može provjeriti ima li prijavljeni korisnik ulogu običnog korisnika. Kad u lokalnoj pohrani nema spremljenih podataka, riječ je o gostu aplikacije te on kao takav ima pravo pristupa malom broju funkcionalnosti. Ako uloga nije zadovoljavajuća korisnik se preusmjeri na neku drugu stranicu kojoj može pristupiti.

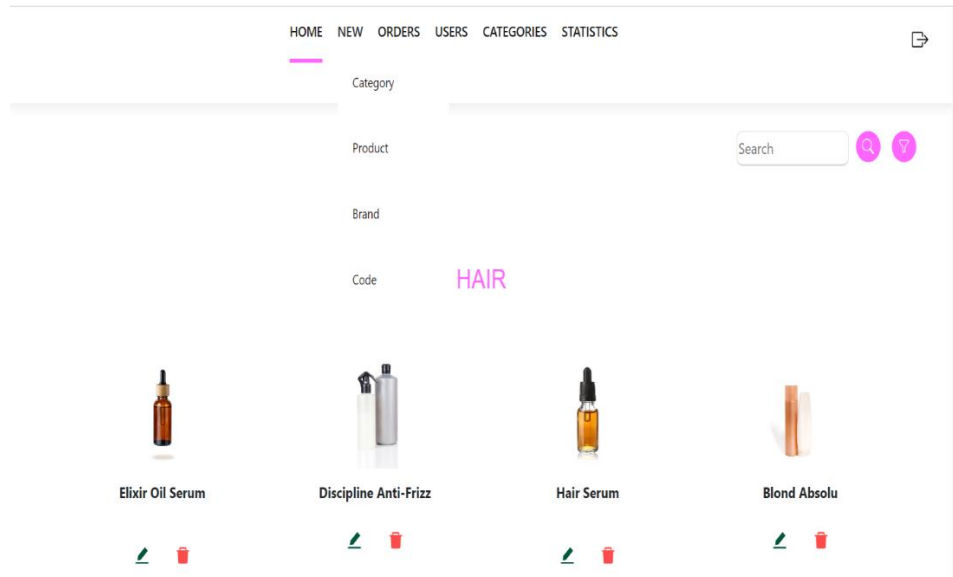
Nakon uspješne prijave korisnika u aplikaciju, korisnički podaci spremaju se u lokalnu pohranu. Kad korisnik prilikom prijave unese korisničko ime i lozinku, prvo se provjeri postoji li korisnik u bazi podataka. Ako ne postoji, ispiše se poruka da podaci nisu ispravni te može pokušati ponovno upisati nove podatke. Ukoliko su podaci ispravni, poslužiteljska strana aplikacije vrati JWT token korisnika te se taj token sprema u lokalnu pohranu. Osim JWT tokena spremi se identifikacijska oznaka korisnika, ime i uloga. Ispis 3 prikazuje dio kôda u kojem se spremaju podaci o korisniku u lokalnu pohranu.

```
if (data.accessToken) {  
    setIsCorrect(true);  
    localStorage.setItem("token", data.accessToken);  
    const myuser = JSON.stringify({  
        id: data.user_id,  
        name: data.user_name,  
        role: data.user_role  
    });  
    localStorage.setItem("user", myuser);  
    let myuse = JSON.parse(localStorage.getItem("user"));  
    setUser(myuser);  
    window.location.href = '/';  
} else {  
    setIsCorrect(false);  
    setMessage(data.message);  
}
```

Ispis 3 - Spremanje korisničkih podataka u lokalnu pohranu

3.3. Korisničko sučelje za administratora

Izgled administratorskog sučelja prikazan je na Slici 2. Administrator može dodavati nove proizvode, kategorije, brendove i kodove za popust, osim toga može upravljati korisnicima, proizvodima i narudžbama. Kako bi lakše upravljao navedenim dodani su botuni za uređivanje i brisanje. Pritiskom na botun za uređivanje korisnika se preusmjeri na stranicu na kojoj su vidljivi trenutno spremljeni podaci o proizvodu te ih može mijenjati.



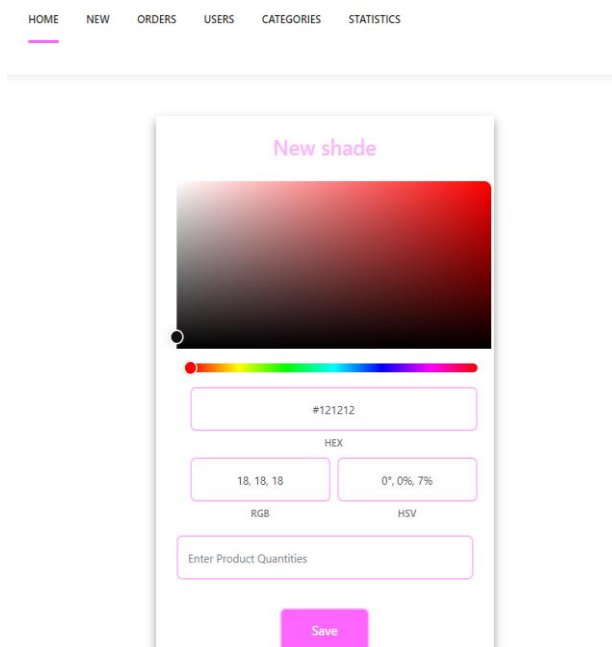
Slika 2 – Izgled administratorskog sučelja

U navigaciji pod opcijom „New“ administratoru se prikazu podaci koje može dodavati. Nakon što se odabere unos nekih od ponuđenih podataka, administrator se preusmjeri na odabranu stranicu na kojoj se prikazuje forma za unos odabranih podataka. Slika 3 prikazuje formu za unos podataka o proizvodu.

Slika 3 - Prikaz forme za unos proizvoda

Prilikom uređivanja podataka o proizvodu, administrator ima mogućnost dodavanja nijansi proizvoda. Neki proizvodi imaju univerzalnu nijansu, a neki su dostupni u više nijansi. Proizvodi koji imaju više nijansi imaju vezu na posebnu tablicu u kojoj se nalazi nijansa i količina. Korisniku se ispod ovakvih proizvoda prikazu sve dostupne nijanse i on može odabrati točnu nijansu koju želi. U slučaju da je neki proizvod snižen, administrator određuje popust sniženog proizvoda i uređuje novu cijenu.

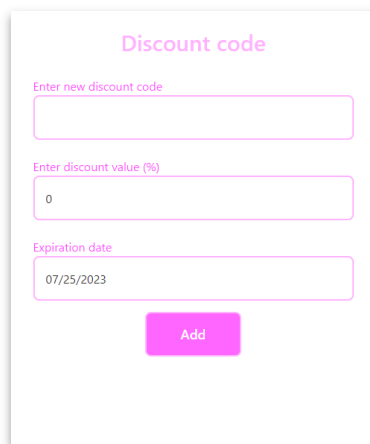
Dodavanje nijansi određenog proizvoda moguće je odabirom željene nijanse te unosom količine u kojoj je dostupan proizvod odabrane nijanse. Samo administrator može dodati nijansu nekom proizvodu. Nakon što nestane određene nijanse sa stanja korisniku se ta nijansa više ne prikazuje kao dostupna. Slika 4 prikazuje funkcionalnost dodavanja nijanse određenom proizvodu.



The screenshot shows a web application interface with a navigation bar at the top containing links: HOME, NEW, ORDERS, USERS, CATEGORIES, and STATISTICS. The 'NEW' link is highlighted with a red underline. Below the navigation bar is a form titled 'New shade'. The form includes a large color selection area with a gradient from black to red. Below this is a horizontal color bar with a red dot indicating the selected color. The form also contains input fields for the color code (showing '#121212' and labeled 'HEX'), RGB values (showing '18, 18, 18' and labeled 'RGB'), and HSV values (showing '0°, 0%, 7%' and labeled 'HSV'). There is a text input field labeled 'Enter Product Quantities' and a red 'Save' button at the bottom.

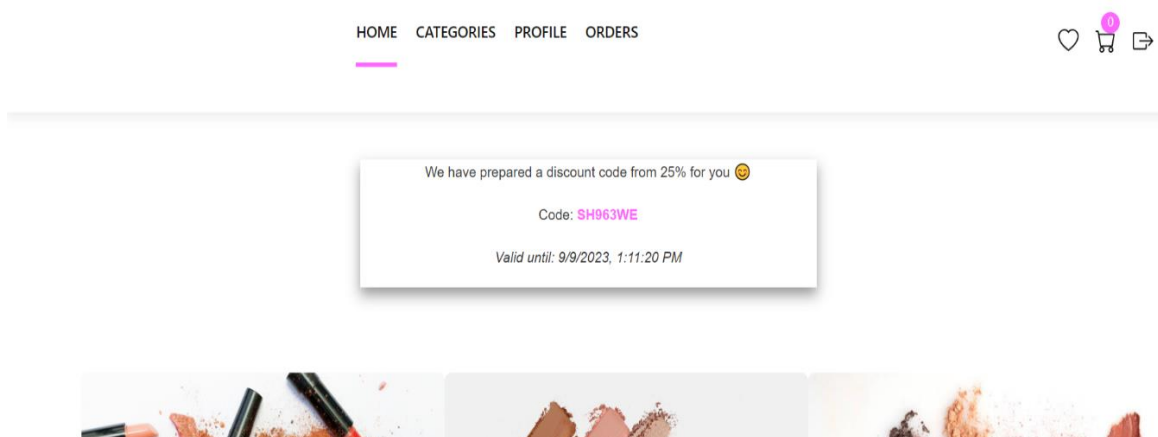
Slika 4 – Dodavanje nijanse određenom proizvodu

Osim toga, administrator može dodavati kodove za popust. Prilikom unosa koda za popust, određuje tekst koda pomoću kojeg će korisnik dobiti popust na ukupnu vrijednost košarice, unosi i vrijednost popusta u obliku postotka te datum do kojeg kod vrijedi. Slika 5 prikazuje izgled unosa novog popusta.



Slika 5 – Unos novog popusta za korisnike

Nakon što administrator unese novi kod za popust, na početnoj stranici korisnika pojavi se obavijest o novom popustu uz objašnjenje koja je valjana šifra koda i do kada se može iskoristiti. Obavijest je vidljiva do isteka valjanosti koda. Kad popust prestane vrijediti obavijest se više ne prikazuje. Slika 6 prikazuje izgled obavijesti o popustu na početnoj stranici registriranog korisnika.



Slika 6 - Prikaz obavijesti o popustu za registriranog korisnika

Kodovi za popust koje administrator unese spremaju se u bazu podataka. Aktivni kodovi prikazu se na stranici na način da se iz baze podataka dohvate kodovi čiji je datum valjan. Kodovi se filtriraju na poslužiteljskom dijelu aplikacije tako da se filtriraju kodovi čiji je datum valjanosti veći od trenutnog datuma. Ispis 4 prikazuje dio koda za filtriranje kodova za popust na poslužiteljskom dijelu aplikacije.

```
productRouter.get('/discounts', cors(), (req, res)=>{
    Discount.find({expire: { $gt: new Date() }}),
    (err, discount)=>{
        if(err){
            console.log(err);
        }
        else{
            return res.json(discount);
        }
    })
})
```

Ispis 4 - Filtriranje valjanih kodova za popust

Nakon što se dohvate valjani kodovi od poslužitelja, na klijentskom dijelu aplikacije potrebno je prikazati trenutno aktivne kodove. Kodovi se prikazuju samo registriranom korisniku, administratoru i neregistriranom korisniku se ne prikazuju. Prikaz je implementiran tako da se prije ispisa provjeri uloga trenutno prijavljenog korisnika. Ispis 5 prikazuje kôd u kojem se korisniku prikazuje obavijest o valjanim kodovima.

```









{ (discounts.length) && (JSON.parse(localStorage.getItem("user"))) &&
(JSON.parse(localStorage.getItem("user")).role === 'user') ? (
    <div className='codeDiv'>
        {discounts.map((item) =>{
            return(
                <div className='code-body'>
                    <p>We have prepared a discount code from
                    {item.value}% for you &#128522;</p>
                    <p><span>Code: </span> <span
                    style={{color:"#ff66ff"}}><b>{item.code}</b></
                    span></p>
                    <p><i><span>Valid until:</span> {(new
                    Date(item.expire)).toLocaleString()}</i></p>
                </div>)
            })}
        </div>
    ) : (<span></span>)}

```

Ispis 5 - Prikaz kôda za postavljanje obavijesti o popustu

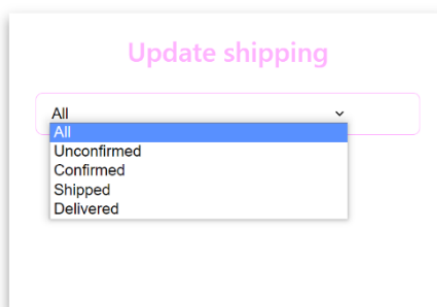
Administrator ima mogućnost pregleda liste narudžbi i korisnika tako da mu se prikažu sve narudžbe/korisnici i kraj njih botuni za uređivanje ili brisanje. Ovim funkcionalnostima administrator može pristupiti preko opcija „Orders“ i „Users“ koje se nalaze u navigaciji korisničkog sučelja za administratora. Preko opcije „Orders“ može se jednostavno mijenjati status narudžbi. Podaci korisnika/narudžbi formirane su u obliku tablice. Prikazani su neki važniji podaci o narudžbi, kao što su identifikacijska oznaka narudžbe, status te botuni za upravljanje podacima. Prilikom uređivanja narudžbe administrator zapravo uređuje samo status narudžbe. Status mijenja na način da odabire jednu od opcija iz padajućeg izbornika. Kod prikaza

liste korisnika prikaže se ime i prezime korisnika te botuni za uređivanje i brisanje. Slika 7 prikazuje izgled sučelja za prikaz svih narudžbi.

HOME	NEW	ORDERS	USERS	CATEGORIES	STATISTICS
ID	STATUS	EDIT	DELETE		
6475b57b05216c9d9ee5aaba	Shipped				
64ad99c18b2dea463bcc75a3	Shipped				
64ae477816ca227b2dbc8bef	Unconfirmed				
64aebe4fc8f5a52fbb320dd0	Unconfirmed				

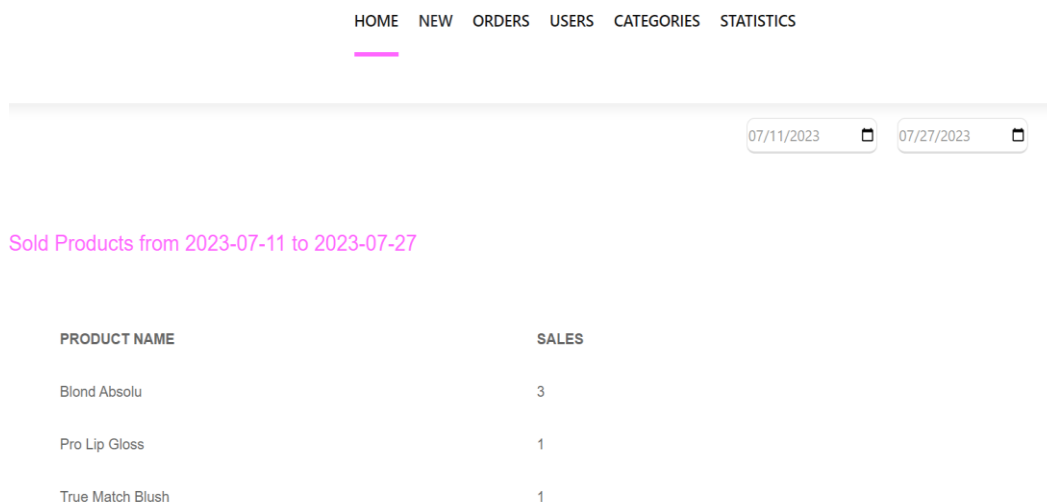
Slika 7 – Prikaz narudžbi za administratora

Pritiskom na botun za uređivanje, administratoru se prikaže stranica na kojoj je moguće urediti korisnika/narudžbu. Nakon pritiska botuna za brisanje, određeni korisnik ili narudžba briše se iz baze podataka i više neće biti prikazan. Slika 8 prikazuje način uređivanja statusa naručenog proizvoda.



Slika 8 – Uređivanje statusa narudžbi

Administrator može pratiti statistiku prodanih proizvoda u određenom vremenskom rasponu. Raspon određuje sam administrator tako da odabire početni i/ili završni datum u kojem želi pratiti prodaju. Odabirom samo početnog datuma prikaže se statistika od početnog do trenutnog datuma, a odabirom samo završnog datuma dobiju se svi prodani proizvodi do završnog datuma. Na koncu, odabirom oba datuma vidljiva je prodaja proizvoda unutar odabranog vremenskog raspona. Pregled statistike prikazan je na Slici 9.



Slika 9 – Prikaz statistike s odabranim vremenskim rasponima

Statistika je napravljena tako da su dohvaćeni svi naručeni proizvodi, a iteracijom kroz naručene proizvode zbrojeno je koliko puta je pojedini proizvod naručen. Važno je bilo paziti na parametre koje korisnik odabire. Pošto korisnik može odabrati samo početni datum, početni i završni ili samo završni datum, podatke je trebalo filtrirati ispravno prema unosu. U slučaju da je unesen samo početni datum, filtrirani su svi najprodavaniji proizvodi od odabranog početnog datuma do trenutnog datuma. Ako je korisnik odabrao i početni i završni datum, traženi su prodani proizvodi u rasponu dva odabrana datuma. Na koncu, ukoliko je odabran samo završni datum, prikazani su svi proizvodi koji su prodani do odabranog datuma. Nakon funkcija za filtriranje i brojanje prodanih proizvoda, napisana je i funkcija koja iz sortiranog niza uzima samo 10 najprodavanijih proizvoda. Kôd za sortiranje prikazan je u Ispisu 6.

```

function sortFun(filterArray){
    const productCounts = {};
    filterArray.forEach(sale => {
        const productName = sale.name;
        productCounts[productName]=(productCounts[productName]
            || 0) + 1; });
    const sortedProducts = Object.entries(productCounts)
        .sort(([ , countA], [ , countB]) => countB - countA)
        .map(([product, count]) => ({ product, count }));
    return sortedProducts.slice(0, 10);
}

```

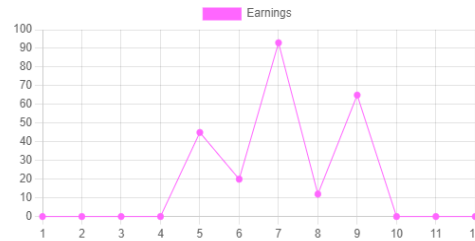
Ispis 6 – Funkcija sortiranja najprodavanijih proizvoda

Funkcija za sortiranje najprodavanijih proizvoda prikazana u Ispisu 6, iterira kroz prodane proizvode i izbroji koliko je puta svaki proizvod prodan. Nakon toga, proizvodi se sortiraju silazno po broju prodaja te se vrati samo prvih deset elementa. Pošto su proizvodi sortirani silazno, tih deset prvih elemenata predstavljaju deset najprodavanijih proizvoda.

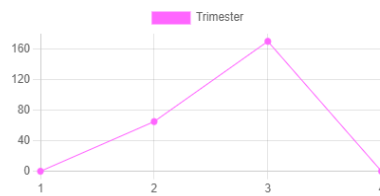
Osim navedenih najprodavanijih proizvoda, administrator može pratiti i statistiku zarade tijekom određenog vremenskog razdoblja. Odabirom početnog i/ili završnog vremenskog intervala prikaže se ukupna zarada trgovine u odabranom vremenskom rasponu. Osim toga, nudi mu se polje za unos godine te tako može vidjeti mjesečnu zaradu, zaradu po tromjesečjima ili ukupnu godišnju zaradu u odabranoj godini. Ako administrator ne unese godinu prikažu mu se statistički podaci o zaradi u tekućoj godini. Podaci su vidljivi u obliku tabličnog zapisa, ali i grafičkog zapisa. Na Slici 10 vidljivi su statistički podaci o zaradi u obliku tablice i grafa u tekućoj godini.

Total earnings in 2023. - 235€

Month	Earnings
1.	0€
2.	0€
3.	0€
4.	0€
5.	45€
6.	20€
7.	93€
8.	12€
9.	65€
10.	0€
11.	0€
12.	0€



Trimester	Earnings
1.	0€
2.	65€
3.	170€
4.	0€



Slika 10 - Mjesečna, tromjesečna i godišnja zarada u tekućoj godini

Za prikaz grafova korišten je modul „Line“ koji omogućuje iscrtavanje grafa pomoću podataka koji se definiraju za x i y os. Uz navedeni modul bili su potrebni i drugi moduli kao što su: „Chart“, „CategoryScale“, „LinearScale“, „PointElement“, „LineElement“ i ostali. Ove module bilo je potrebno uključiti u projekt te nakon toga registrirati pomoću metode `register` koja je dio „Chart“ modula. Nakon što su uključeni svi potrebni moduli, bilo je potrebno napisati funkciju koja pronalazi potrebne podatke za mjesečnu, tromjesečnu i godišnju zaradu. Ovi podaci su ključni za definiranje osi grafova te se pomoću njih iscrtava graf. Grafovi su prikazani za podatke o mjesečnoj i tromjesečnoj zaradi, a podaci o ukupnoj zaradi su samo ispisani, pošto se radi samo o jednom ukupnom broju. Funkcija koja računa zaradu prikazana je u Ispisu 7.

```

let mArray = [], tArray = [], totalSum = 0, totalMonth = 0, totalTrim =
0, trimCounter = 0;
const myYear = year > 0 ? Number(year) : Number(currYear);
for (let i = 0; i < 12; i++){
    if (trimCounter % 3 === 0 && i > 1){
        tArray.push(totalTrim);
        totalTrim = 0;
    }
    shippings.map((item, index) => {
        let itemDate = new Date(item.date);
        if (itemDate.getMonth() === i && itemDate.getFullYear() ===
myYear){
            totalMonth += item.total_price;
            totalSum += item.total_price;
            totalTrim += item.total_price
        }
    });
    trimCounter++;
    mArray.push(totalMonth);
    totalMonth = 0;
}

```

Ispis 7 - Prikaz kôda za računanje zarade

Nakon definiranja podataka koji će biti prikazani na grafu, definirana su i svojstva grafa. Svojstva koja su definirana su: podaci x i y osi, boja grafa i debljina linije grafa. Na kraju, svi ti podaci dodani su i uključeni u modul „Line“ kako bi se konačno prikazali na stranici. Primjer definiranja svojstava grafa i sam ispis grafa prikazan je u Ispisu 8.

```
const data = {
  labels: [1, 2, 3, 4],
  datasets: [{
    label: 'Trimester',
    data: trimester,
    backgroundColor: '#ff66ff',
    borderColor: '#ff66ff',
    borderWidth: 1,
  }],
};

<Line data={data} options={options} />
```

Ispis 8 - Primjer definiranja svojstava grafa

Administrator ima mogućnost dodavanja proizvoda na skladište za sljedeći mjesec, kako proizvodi ne bi bili u manjku i kako bi korisnici imali uvijek dostupne proizvode. Navedena funkcionalnost temelji se na činjenici koliko je proizvoda prodano isti mjesec prethodne godine. Potrebno je pronaći proizvode koji su prodani prethodne godine u tom mjesecu i količinu prodanih proizvoda tog mjeseca. Manjak se računa kao razlika prethodno izračunate količine i trenutnog stanja proizvoda na skladištu. Na ovaj način administrator ima uvid u proizvode koji bi potencijalno mogli biti u manjku sljedeći mjesec. Tako bi se problem proizvoda koji nisu trenutno na stanju trebao riješiti, ali naravno pošto je ovo samo predviđanje, ne znači da će biti točno za svaki mjesec. Za implementaciju ove funkcionalnosti potrebno je bilo pronaći proizvode koji su prodani prošle godine u mjesecu koji prethodi trenutnom (pošto se radi predviđanje za sljedeći mjesec ne gleda se trenutni mjesec već onaj koji prethodi trenutnom). Funkcija koja filtrira narudžbe koje zadovoljavaju prethodno napisani uvjet vidljiva je u Ispisu 9.

```

setShippingPerMonth(shipplings.filter(sale => {
    const saleDate = new Date(sale.date);
    const oneDayAgo = new Date();
    return saleDate.getMonth() + 2 === currentMonth &&
        saleDate.getFullYear() === currentYear - 1;
}));

```

Ispis 9 - Filtriranje proizvoda za određeni mjesec prethodne godine

Nakon što su proizvodi filtrirani izračunato je u kojoj je količini proizvod prodan ukupno taj mjesec. Na kraju, razlika izračunate količine i trenutnog stanja skladišta predstavlja eventualni manjak. Administrator može vidjeti popis proizvoda koji bi mogli biti u manjku sljedeći mjesec i povećati količinu proizvoda na stanju za iznos izračunatog manjka. Kako bi to što jednostavnije napravio, kraj svakog proizvoda ispisan je potencijalni manjak, ali i dodan botun „Add“. Klikom na botun „Add“ količina određenog proizvoda na skladištu poveća se za iznos manjka. Nakon toga se proizvod više ne nalazi na tom popisu pošto je manjak uklonjen dodavanjem nove količine na stanje. Slika 11 prikazuje kako izgleda korisničko sučelje za predviđanje stanja na skladištu za sljedeći mjesec.

The lack of products on stock				
LACK	ID	NAME	MESSAGE	
2	644a4d44b22cfc687f9ea4d1	Blond Absolu	Order	Order
1	6447b1ef6b591a753eebe028	Discipline Anti-Frizz	Order	Order
1	6418910eaeaba4879db6e3ea	Infallible Foundation	Order	Order

Slika 11 - Predviđanje stanja na skladištu za sljedeći mjesec

Ispis 10 prikazuje dio kôda koji služi za vizualni prikaz funkcionalnosti predviđanja u aplikaciji.

```
{soldProducts.map(({product, count}) => {
  return Object.keys(products).map((p) =>{
    if (product === products[p].name && (count -
      products[p].on_stock) > products[p].on_stock){
      return (
        <div className="row" >
          <div className='col-3'><p class="p-1">{count -
            products[p].on_stock }</p> </div>
          <div className='col-3'><p class="p-1">{
            products[p]._id}</p> </div>
          <div className='col-2'><p class="p-1">{
            products[p].name}</p></div>
          <div className='col-2'><p class="p-1">
            Order</p></div>
          <div className='col-1'><p class="p-1"><button
            onClick={()=>{addFormData(products[p], count -
              products[p].on_stock)}}
            className="orderBtn">Order</button></p></div>
        </div>)
      }
    })
  })
})
```

Ispis 10 - Prikaz kôda za ispisivanje podatka o manjku na skladištu

Administrator može pratiti statistiku deset kupaca koji su ostvarili najviše kupnji prethodni mjesec. Na temelju toga može im dodijeliti određeni popust. Ova statistika se svaki mjesec mijenja kako bi administrator mogao na kraju mjeseca nagraditi svoje kupce. Funkcionalnost je

implementirana na način da su pronađeni kupci koji su kupili nešto prethodni mjesec. U Ispisu 11 vidljiva je funkcija koja traži kupce koji su obavili kupnju prethodni mjesec.

```
function getTopUsers() {  
    setUsersPerMonth(shippings.filter(sale => {  
        const saleDate = new Date(sale.date);  
        const currDate = new Date();  
        if (saleDate.getMonth() === currDate.getMonth() - 1 &&  
            saleDate.getFullYear() === currDate.getFullYear()) {  
            return saleDate;  
        } else if (saleDate.getMonth() === currDate.getMonth() - 1  
            && (saleDate.getFullYear() === currDate.getFullYear() - 1)  
            && saleDate.getFullYear() === 12) {  
            return saleDate;  
        }  
    }))  
    var array = [];  
    usersPerMonth.map((item) => {  
        users.map((u) => {  
            if (u._id === item.user_id) {  
                array.push(u);  
            }  
        })  
    })  
    setTopMonthUsers(sortUserFun(array));  
}
```

Ispis 11 - Funkcija za filtriranje kupaca iz prethodnog mjesec

Nakon toga, iteracijom kroz kupce prebrojano je koliko kupnji je svaki kupac ostvario prošli mjesec. Napisana je funkcija koja sortira silazno kupce po broju kupnji i ta funkcija vrati deset prvih kupaca. Drugim riječima, deset kupaca koji su imali najveći broj kupnji prethodni mjesec. Prethodni mjesec dobije se tako da se uspoređi mjesec u kojem je kupljen proizvod s trenutnim

mjesecom umanjenim za jedan, a godine bi trebale biti iste. U slučaju da je prodaja ostvarena u prosincu, onda se trenutna godina umanjuje za jedan prilikom usporedbe godina. Funkcija za sortiranje kupaca prikazana u Ispisu 12.

```
const sortedProducts = Object.entries(productCounts)
  .sort(([ , countA], [ , countB]) => countB - countA)
  .map(([product, count]) => ({ product, count }));
return sortedProducts.slice(0, 10);
```

Ispis 12 - Funkcija za sortiranja kupaca silazno

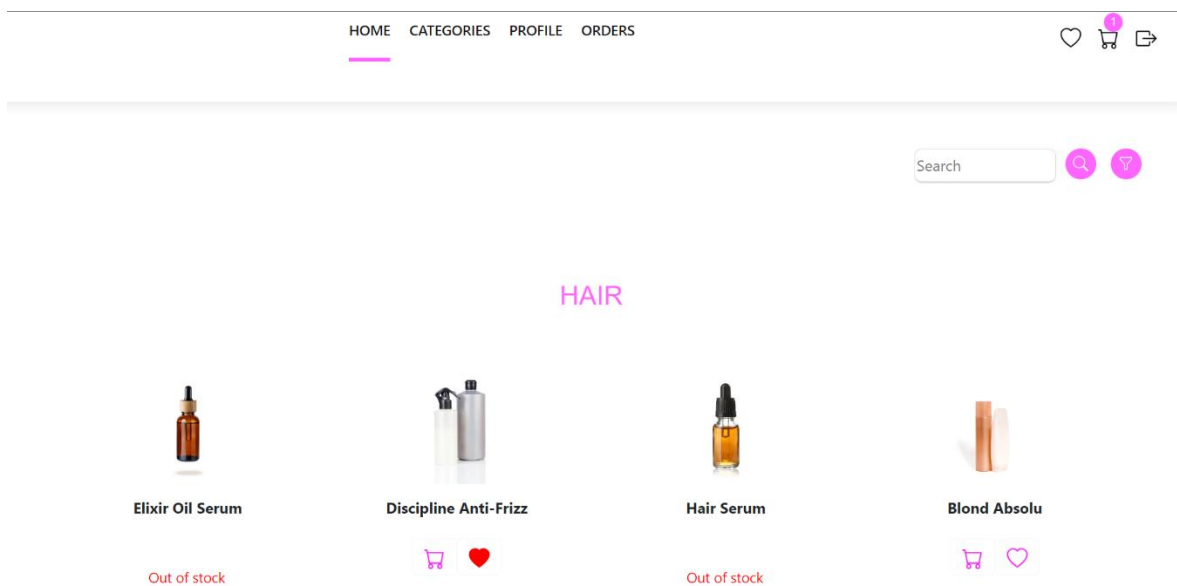
Nakon što su pronađeni kupci s najviše kupnji prikazani su na stranici. Sustav je unaprijed izračunao popust koji im administrator može dodijeliti tako da se pomnožio broj kupnji s brojem deset. Kraj svakoga kupca nalazi se i botun „Add“ te pritiskom na navedeni botun administrator može kupcu dodijeliti popust koji sustav predlaže, ali ne mora. Ukoliko mu se dodijeli taj popust, korisnik će ga moći iskoristiti kao dodatni popust koji će umanjiti ukupan iznos košarice. Pomoću navedene funkcionalnosti administrator može mjesečno nagrađivati svoje najvjernije kupce kako bi u budućnosti nastavili kupovati u aplikaciji. Slika 12 prikazuje prethodno opisanu funkcionalnost.

Popular products Earnings Customers					
ID	USER	PURCHASES	DISCOUNT	TOTAL DISCOUNT	
647214c4530d7193fa7ca235	Test2	3	30%	30%	ADD
647c9b7b7123dcc283b717f9	Test3	1	10%	10%	ADD

Slika 12 - Prikaz kupaca s najvećim brojem kupnji prethodni mjesec

3.4. Korisničko sučelje za registriranog korisnika

Sučelje registriranog korisnika izgledom je drugačije od sučelja za administratora. Razlike su vidljive već u samoj navigaciji. Prikaz proizvoda korisničkog sučelja vidljiv je na Slici 13.



Slika 13 – Prikaz korisničkog sučelja za registriranog korisnika

Slika 14 prikazuje ponudu i funkcionalnosti za registriranog korisnika. Kao takav korisnik može pretražiti ponudu ili filtrirati ponudu po cijeni (klikom na ikonu filtera prikaže se polje za unos minimalne i maksimalne cijene). Ponuda je sortirana abecedno po kategorijama. Ukoliko je proizvod dostupan korisnik može dodati proizvod u košaricu ili u listu omiljenih proizvoda (crveno srce označava da je proizvod već u listi omiljenih proizvoda, a bijelo da nije), u protivnom ispod proizvoda je ispisano „Out of stock“. Početna stranica proizvoda ispiše samo po četiri proizvoda unutar svake kategorije, ali klikom na „See all“ tekst, moguće je vidjeti sve proizvode željene kategorije.

Unutar košarice korisnik ima mogućnost upisati kod određenog popusta te ako je kod valjan smanji se iznos ukupne vrijednosti košarice. Kod za popust određuje administrator te se aktivni

i valjani kodovi prikazuju na početnoj stranici registriranog korisnika. Prilikom provjere valjanosti koda prvo se provjeri je li kod valjan, odnosno je li mu istekao datum do kojega vrijedi i je li unesena dobra šifra koda. Ako je kod valjan, provjerava se je li korisnik već prije iskoristio kod koji je trenutno aktivan. Uz navedeni popust, korisnik na svaku okruglu kupnju dobije popust u iznosu te okrugle kupnje koju može iskoristiti ako želi. Korisnik ne treba popust koji ostvari nakon određenog broja kupnje iskoristiti odmah, može u bilo kojem trenutku. Navedena vrsta popusta nije vremenski ograničena već korisnik ima mogućnost sakupljati popuste do trenutka kad ih želi iskoristiti. Nakon što korisnik ostvari popust, u košarici mu se pojavi mogućnost da pritiskom na botun iskoristi popust. Ako nema nikakvog popusta ne prikaže mu se navedena opcija. Funkcija za provjeru koda prikazana je u Ispisu 13.

```
function checkCode() {
  currentCode.length > 0 ?
  currentCode.map(async (item) => {
    if (item.code === discount){
      const options = {headers:{
        Authorization: "Bearer " + localStorage.getItem("token")
      }};

      const discountsList = await
      fetch(`http://localhost:3001/product/userDiscounts/${params.id}/${
        item._id}`, options);

      const discountsJson = await discountsList.json();
      if (discountsJson.length === 0){
        setExtraDiscount(item.value);
        setDiscountId(item._id);
      }else {
        setIsCorrect(false);
        setMessage("You have already used this code!");
      }
    }
  }) : setExtraDiscount(0);}
```

Ispis 13 - Provjera valjanosti koda za popust

U funkciji iz Ispisa 13 vidljivo je kako se prvo provjeri postoji li kod za popust kojeg je korisnik upisao. Ako postoji, potrebno je provjeriti nalazi li se u listi kodova koje je korisnik već iskoristio. Ukoliko se nalazi, korisniku nije dopušteno iskoristiti kod iznova. Nakon unosa valjanog koda smanjuje se ukupan iznos košarice. Ažuriraju se podaci, tako da se spremi korisnik koji je iskoristio trenutno aktivni kod, kako ga ne bi više puta mogao koristiti. U slučaju druge vrste popusta, nakon aktiviranja dodatnog popusta prilikom plaćanja i nakon uspješno plaćene narudžbe, popusti se poništavaju (pošto je iskoristio popust) i korisnik ponovno ostvaruje bodove nakon okruglih kupnji ili na neki drugi način.

Različiti popusti međusobno se ne zbrajaju, tijekom jedne kupnje može se koristiti samo jedna vrsta popusta. Ukoliko korisnik upiše šifru koda koji je valjan ili ako klikne „Yes“ da želi iskoristiti svoj dodatni popust koji je ostvario, ukupan iznos košarice smanjuje se za vrijednost ostvarenog popusta. Slika 14 prikazuje kako u košarici izgleda prethodno navedeni dio s popustima.



The screenshot displays a shopping cart summary with the following elements:

- Total product number:** 2
- Total price:** 38.95 €
- Discount rule:** "You can use only one type of discount!" with a sub-note "*Different discounts do not count."
- Discount code input:** A label "Dicount code:" followed by a text input field containing the placeholder "Type your dicount code" and a "Check" button.
- Extra discount notification:** "You have 20% extra discount. Do you want use it?" with a radio button and the label "Yes".

Slika 14 – Prikaz mogućih načina za korištenja popusta

Ispis 14 prikazuje na koji način je implementiran dio sa Slike 14 u kôdu.

```

<span>*Different discounts do not count.</span></span></b></p>
<p><b><span> Dicount code: </span></b>
<input type="text" placeholder="Type your dicount code"
onChange={ (e) => {setDiscount(e.target.value); checkCode();}}/>
<button className = "checkBtn" onClick={checkCode}>Check</button>
</p>
{ !isCorrect ? (<div class="alert alert-danger" role="alert">Error:
{ message }!</div>) : null }
{user.discount && activate !== true > 0 ? (
    <><p>You have {user.discount}% extra discount. Do you want
    use it?
    <span><input type="radio" value="true" name="activate"
    onClick={onChangeValue}/> Yes</span>
    </p>
    </>) : (<span></span>)
}

```

Ispis 14 - Dio kôda za odabir vrste popusta

Kao što je vidljivo u Ispisu 14, korisnik može pokušati napisati neki kod za popust, ali nakon klika na botun „Check“ provjerit će se je li upisani kod valjan. Ukoliko nije, ukupna vrijednost košarice ostaje nepromijenjena, korisnik ima mogućnost ponovno unijeti neki kod. U situaciji kad korisnik unese kod koji je iskoristio prikazat će mu se greška koja je vidljiva u Ispisu 14. Ako je korisnik ostvario dodatan popust, prikazat će mu se mogućnost da ga iskoristi, ali ako nije onda ova mogućnost nije vidljiva na stranici.

Slika 15 prikazuje situaciju kada se pokuša jedan kod iskoristiti više puta.

Total price: 54.00 €

You can use only one type of discount!

*Different discounts do not count.

Discount code: SH963WE

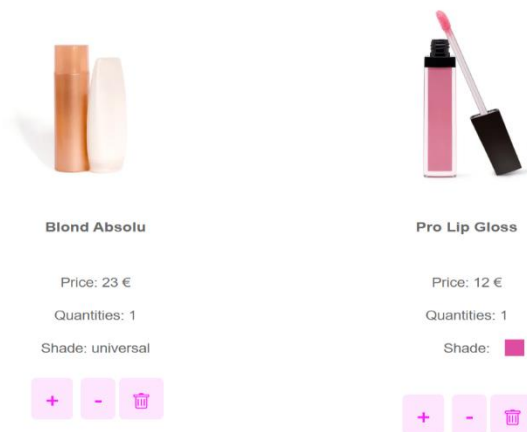
Check

Error: You have already used this code!!

You have 30% extra discount. Do you want use it? ☐ Yes

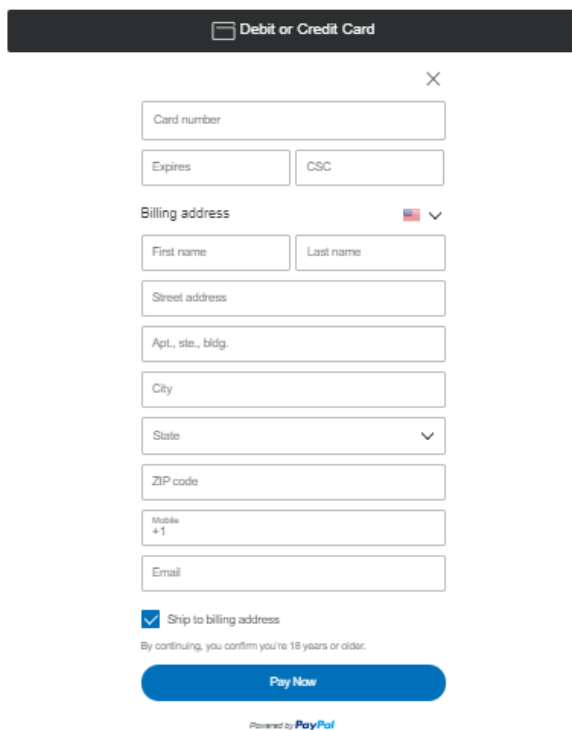
Slika 15 - Neuspjeli pokušaj korištenja koda za popust

Korisnik unutar košarice može povećavati/smanjivati količinu dodanih proizvoda ili u potpunosti ukloniti proizvod u košarici. Količinu proizvoda unutar košarice nije moguće povećati iznad količine u kojoj je taj proizvod dostupan. Trenutno dostupna količina proizvoda spremljena je u varijablu `on_stock`. Nakon kupnje proizvoda ažurira se vrijednost navedenog polja za količinu kupljenog proizvoda. Na Slici 16 je vidljivo kako je moguće upravljati količinom proizvoda unutar košarice.



Slika 16 – Prikaz proizvoda unutar košarice

Ispod proizvoda u košarici se nalazi botun „Pay“. Nakon klika na botun otvori se mogućnost odabira načina plaćanja. Moguće je platiti preko PayPal računa ili karticom. Prilikom plaćanja PayPal računom, potrebno je prijaviti se u PayPal račun te nakon toga unijeti kartične podatke. Ako se odabere plaćanje karticom, potrebno je unijeti ispravne podatke. Nakon uspješnog plaćanja sadržaj košarice se isprazni, a podaci o narudžbi se spremaju u bazu podataka. Slika 17 prikazuje formu za unos kartičnih podataka prilikom plaćanja narudžbe.



Slika 17 - Forma za unos podataka za plaćanje narudžbe

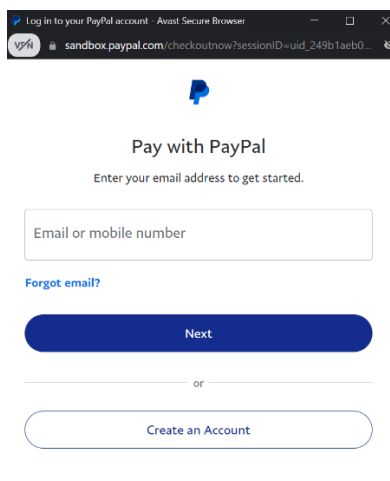
Za korištenje funkcionalnosti sa Slike 17 korišteni su moduli „PayPalScriptProvider“ i „PayPalButtons“. Uz navedeno, bilo je potrebno izraditi račun na „PayPal Developer“ stranici. Nakon izrade računa trebala se izraditi aplikacija. Pošto PayPal API, za dohvaćanje identifikacijskog broja koristi pristupni token kako bi se provjerila autentičnost zahtjeva, bilo je potrebno generirati „ClientId“ i „Secret“. Na stranici „PayPal Developer“ u poglavlju „My Apps & Credentials“ potrebno je bilo kopirati „ClientId“ i „Secret“ i dodati ih u React konfiguracijsku datoteku. Poslije postavljanja postavki na navedenoj stranici i nakon instaliranja potrebnih modula u React-u moglo se započeti s uvođenjem PayPal modula u aplikaciju. PayPal moduli

dolaze s dvije već napisane metode `createOrder` i `onApprove`. Ove metode trebalo je modificirati kako bi radile u skladu s potrebama aplikacije. Ispis 15 prikazuje način korištenja PayPal modula u aplikaciji.

```
<PayPalButtons className="paypalbtn"
    style={{ layout: "vertical", alignItems:"center",
    position:"relative", right:"20%"}}
    createOrder={createOrder} onApprove={onApprove} />
```

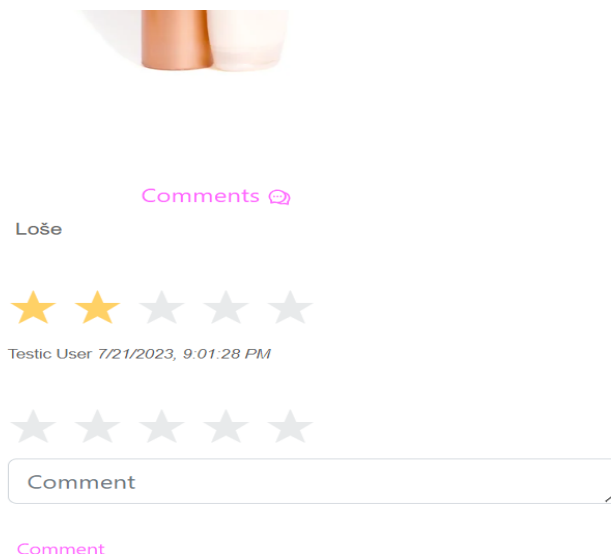
Ispis 15 - Korištenje PayPal modula u aplikaciji

Na Ispisu 15 vidljivo je korištenje „PayPalButtons“ modula. Taj modul prikaže odabir načina plaćanja. Korisnik može odabrati platiti PayPal računom ili kreditnom karticom. Ovisno o tijeku plaćanja i uspješnosti naručivanja pozivaju se funkcije `createOrder` i `onApprove`. Funkcije su prilagođene aplikaciji tako da su u njih dodane neke potrebne akcije. U funkciju `onApprove` dodano je stvaranje i spremanje narudžbe u bazu podataka. Također, bilo je potrebno pratiti je li korisnik iskoristio popust kako bi se ti podaci evidentirali i još neke slične stvari. Na Slici 18 vidljiv je prikaz nakon odabira plaćanja PayPal računom. Korisnik se prvo treba prijaviti u račun ili kreirati novi, a tek nakon toga može platiti proizvod.



Slika 18 - Prijava u PayPal račun

Nakon što korisnik kupi određeni proizvod ima mogućnost recenziranja tog proizvoda. Recenzija može biti u obliku ocjene od 1 do 5 (broj zvjezdica je ocjena) i/ili komentara. Klikom na određeni proizvod vidljivi su detalji proizvoda, ali i recenzije ostalih korisnika. Ukoliko korisnik nije kupio proizvod može samo čitati recenzije ostalih korisnika, u slučaju da je kupio može dodati vlastitu recenziju, ali ako je već recenzirao isti proizvod onda ne može dodati novi komentar već može urediti postojeću ocjenu i/ili komentar te izbrisati vlastitu recenziju. Recenzije su ograničene da ih može dodati samo korisnik koji je kupio taj proizvod kako bi recenzije bile što točnije i iskrenije. Korisnik koji nije kupio proizvod ne može ostaviti svoj komentar ni recenziju pošto možda taj proizvod nije ni koristio. Ispod recenzije vidljivo je koji korisnik je napisao recenziju i datum zadnje izmjene. Gost može čitati komentare i vidjeti recenzije ostalih korisnika, ali ne može napisati vlastitu recenziju za proizvod. Nakon dolaska na stranicu komentari nisu odmah prikazani, tek nakon što korisnik klikne na tekst „Komentari“ oni se prikažu. Na ovaj način korisnik može čitati komentare ako želi, u protivnom oni mu se neće ispisivati. Komentari su vidljivi nakon klika na tekst „Comments“. Slika 19 prikazuje kako izgleda recenziranje proizvoda.



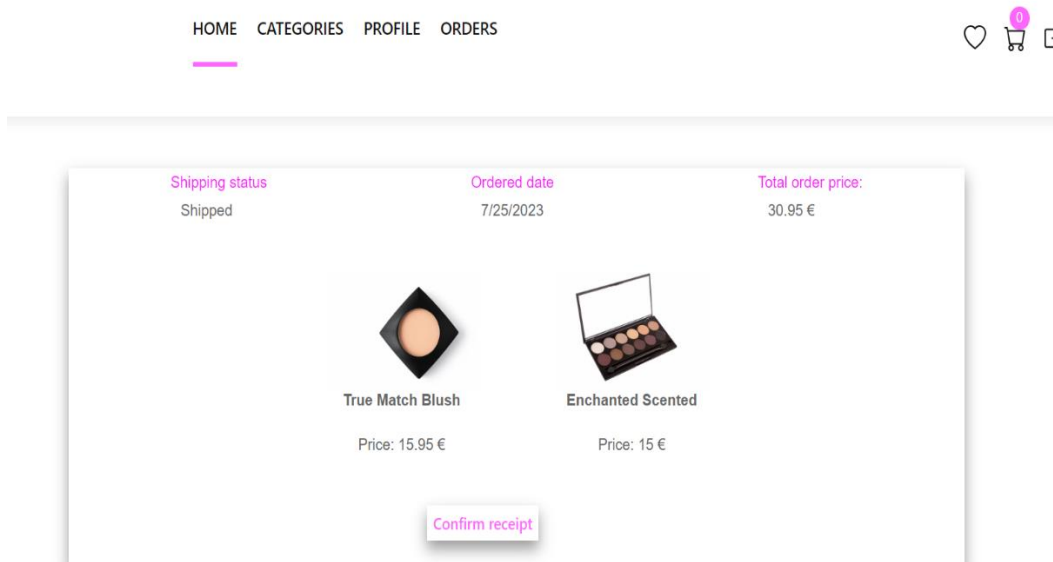
Slika 19 – Funkcionalnost recenziranja proizvoda

Za prikaz zvjezdica instaliran je i korišten modul StarsRating. Modulu je potrebno dodati vrijednost koja predstavlja broj zvjezdica koje će biti obojane žutom bojom, a ukupan broj zvjezdica je unaprijed definiran i postavljen na pet. Primjer korištenja navedenog modula nalazi se u Ispisu 16.

```
return (<div >
  <div className="allComments2">
    <p style={{fontSize:14}}>{com.detail}</p>
  </div>
  <StarsRating classNamePrefix="react-star-rate" value={com.vote}
  size="small" disabled />
  <p className="username">{u.name + " " + u.surname}
    <span className="dateCom">
      {(new Date(com.date)).toLocaleString()}
    </span>
  </p>
</div>)
```

Ispis 16 – Implementacija StarsRating modula

Korisnik može pregledati narudžbe, provjeriti status narudžbe ili neke druge informacije o narudžbi te potvrditi primitak narudžbe. Nakon što korisnik potvrdi primitak narudžbe proizvod mu se više ne prikazuje u naručenim proizvodima. Ako korisnik potvrdi primitak narudžbe, narudžba se ne briše iz baze podataka. U tom slučaju, posebno polje se postavlja na vrijednost `true` te se na ovaj način zna da je narudžba dostavljena. Nakon što korisnik klikne na botun „Confirm receipt“ poziva se funkcija koja mijenja polje `shipp` iz `false` u `true`. Prilikom prikaza naručenih proizvoda provjeri se je li navedeno polje `true`. Ako je, ne prikaže se narudžba, u protivnom se prikaže. Slika 20 prikazuje navedenu funkcionalnost.

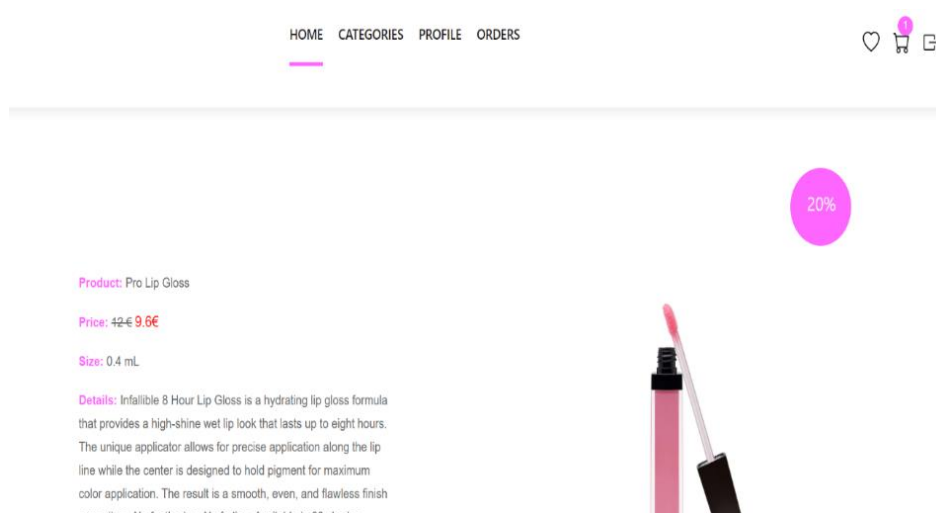


Slika 20 – Prikaz naručenih proizvoda

Kao što je vidljivo na Slici 20, korisnik može vidjeti naručene proizvode, sliku, ime i cijenu naručenih proizvoda, uz to dostupni su mu podaci o datumu narudžbe, ukupnoj cijeni narudžbe i statusu dostave. Status dostave nakon naručivanja je „Unconfirmed“. Administrator mijenja taj status u „Confirmed“ te tako potvrđuje narudžbu. Ovisno o tijeku dostave administrator može status promijeniti u „Shipped“ ili „Delivered“. Nakon što administrator postavi status na „Shipped“ korisnik ima mogućnost potvrditi da je primio narudžbu. Nakon što potvrdi primitak pritiskom na botun proizvod se briše iz narudžbi i korisnik više ne može pratiti tijek dostave za potvrđenu narudžbu.

Korisniku su vidljiva i sniženja određenih proizvoda. Klikom na detalje o proizvodu korisnik može vidjeti postotak sniženosti, ali i novu cijenu proizvoda. Na Slici 21 vidljivo je kako je korisniku prikazana snižena cijena proizvoda i postotak sniženja. Ovi podaci nalaze se u detaljima o proizvodu. Podaci su uvijek fiksni, jedino administrator može mijenjati prikazane podatke. Postotak sniženosti iznosi nula do trenutka dok administrator ne unese koliko je proizvod snižen. Nakon toga ažurira se i cijena proizvoda ovisno o postotku sniženja.

Cijena nakon popusta izračunata je i u aplikaciji prikazana crvenom bojom kako bi korisnik bolje uočio novu cijenu. Uz novu cijenu prikazana je i stara kako bi korisnik mogao bolje procijeniti isplativost nove s obzirom na staru cijenu. Osim cijene dostupni su i ostali podaci o proizvodu kao što su: sastojci, način uporabe, veličina ambalaže, ime i detalji o proizvodu. Ukoliko proizvod sadrži različite nijanse može pogledati u kojim nijansama dolazi željeni proizvod te u skladu s tim naručiti najbolju opciju za sebe.



Slika 21 – Prikaz detalja o proizvodu

Nova cijena proizvoda tijekom sniženja računa se prema formuli koja se nalazi u Ispisu 17.

```
function countPrice(value, discount) {
    const p = parseFloat(getFloat(value));
    const d = (parseFloat(getFloat(value)) *
        parseFloat(discount) / 100);

    return (parseFloat(p) - parseFloat(d));
}
```

Ispis 17 – Prikaz kôda za računanje cijene s popustom

U Ispisu 17 vidljivo je kako se dobije cijena s obzirom na postotak sniženja. Kako bi se mogli podaci zbrajati i dijeliti potrebno je sve podatke pretvoriti u isti decimalni oblik. Nakon toga se trenutna cijena pomnoži s kvocijentom koji je rezultat dijeljenja sniženja i broja sto. Konačna snižena cijena dobije se oduzimanjem stvarne cijene i prethodno dobivenog umnoška.

Prilikom prikaza detalja o proizvodu korisniku su vidljivi prijedlozi drugih proizvoda koji bi mu se mogli svidjeti. Na ovaj način moguće je privući korisnikovu pažnju da nastavi kupnju sličnih proizvoda. Navedeni proizvodi pripadaju istoj kategoriji kao i proizvod čije detalje korisnik pregledava. Funkcionalnost je implementirana na način da kad korisnik klikne na proizvod pronađe se kategorija proizvoda koji je korisniku privukao pažnju. Iz te kategorije prikažu mu se prijedlozi i nekih drugih proizvoda koji bi mu se mogli svidjeti. Ispis 18 prikazuje način dohvaćanja proizvoda iz određene kategorije.

```
async function getCategoryProducts() {  
  
    const options = {headers:{  
  
        Authorization: "Bearer " +  
        localStorage.getItem("token"),  
  
    }};  
  
    const productsList = await  
    fetch(`http://localhost:3001/product/productsByCategory/${  
    product.category_id}`, options);  
  
    const productsJson = await productsList.json();  
  
    setCategoryProducts(productsJson.slice(0, 4));  
  
}
```

Ispis 18 - Dohvaćanje proizvoda koji pripadaju istoj kategoriji

Vizualni prikaz ovog primjera vidljiv je na Slici 22. U ovom slučaju, korisnik pregledava proizvod iz kategorije „Hair“. Ispod su mu vidljivi prijedlozi nekih drugih sličnih preparata za kosu kako njegova pozornost ne bi ostala samo na jednom proizvodu. Ukoliko mu se neki od

predloženih proizvoda svidi, klikom na proizvod može pogledati detalje tog proizvoda i naručiti ga.

Might be interested:

Elixir Oil Serum



Hair Serum



Blond Absolu



Slika 22 - Preporuke proizvoda za korisnika

Kako bi u slučaju velikog broja proizvoda korisnik mogao lakše tražiti ono što mu je potrebno, u aplikaciji je implementirano pretraživanje. Prilikom pregleda narudžbe potrebno je u tražilicu upisati naziv ili dio naziva proizvoda i ako takav proizvod postoji u bazi podataka, korisnik će ga dobiti kao rezultat pretrage. Pretraga je implementirana tako da se nakon unosa naziva u tražilicu kao parametar šalje tekst unesen u tražilicu i čeka se odgovor od poslužiteljske strane kako bi se dobili podaci ako postoje u bazi. Izgled dijela koda s klijentske strane aplikacije prikazan je u Ispisu 19.

```
async function getProducts () {  
    const data = await fetch(`http://127.0.0.1:3001/  
product/searchProducts/${params.search}`);  
    const dataJson = await data.json();  
    console.log("INFO", dataJson);  
    setProducts(dataJson);  
}
```

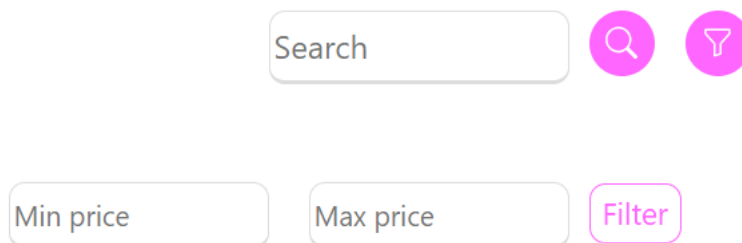
Ispis 19 – Prikaz kôda za pretraživanje proizvoda

Na poslužiteljskoj strani radi se pretraga baze podataka na temelju parametra koji se primi od klijentske strane aplikacije. U slučaju da se pronađu rezultati šalju se klijentskoj strani, a ako dođe do neke greške ispiše se pripadna poruka. Ispis 20 prikazuje izgled prethodno opisane funkcije.

```
productRouter.get('/searchProducts/:search',
cors(), (req, res)=>{
    Product.find({ name: { $regex:
req.params.search } }, (err, products)=>{
        if(err){
            console.log(err)
        }else{
            return res.json(products)
        }
    })
})
```

Ispis 20 – Pretraživanje na poslužiteljskom dijelu aplikacije

Korisnik ima mogućnost i filtrirati proizvode po željenoj cijeni od minimalne do maksimalne vrijednosti cijene. Na ovaj način je postignuto bolje korisničko iskustvo. Nakon klika na ikonu filtera u aplikaciji otvara mu se mogućnost unosa raspona cijena kao na Slici 23.



The image shows a user interface for product filtering. At the top, there is a search bar with the placeholder text "Search". To the right of the search bar are two circular icons: a magnifying glass and a funnel. Below the search bar, there are three input fields: "Min price", "Max price", and "Filter". The "Filter" button is highlighted with a pink border.

Slika 23 – Prikaz kôda za filtriranje proizvoda

Navedena funkcionalnost radi na sličan način kao i prethodna, razlika je samo u upitu koji se izvodi nad bazom podataka. Poslužiteljski dio aplikacije primi dva parametra koji predstavljaju raspon cijene i izvodi upit na bazu pomoću primljenih parametara. Traže se proizvodi čija cijena je veća od minimalne i manja od maksimalne cijene koje su primljene kao parametri. Funkcija je prikazana u nastavku, u Ispisu 21.

```
productRouter.get('/filterProduct/:min/:max', cors(),
(req, res) => {

    Product.find({$and: [{price: { $gte:
parseFloat(req.params.min) }}, {price: { $lte:
parseFloat(req.params.max) } }]}, (err, products)=>{

        if(err){

            console.log(err)

        }

        else{

            console.log(products)

            return res.json(products);

        }

    })

})
```

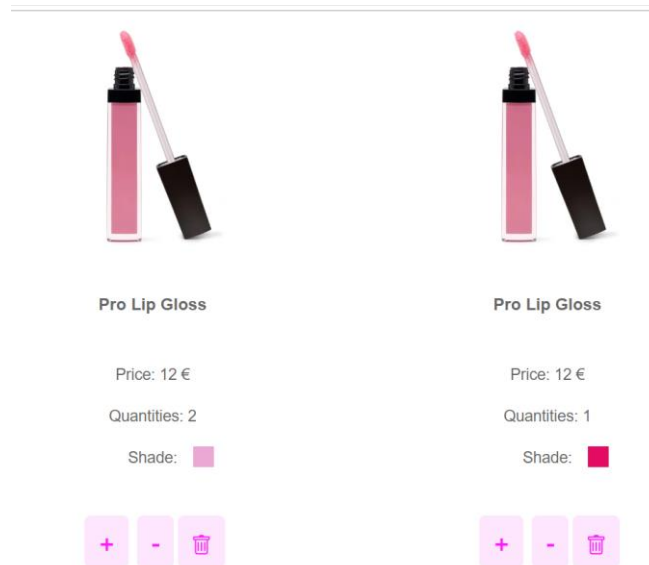
Ispis 21 – Filtriranje proizvoda po parametrima

Neki proizvodi dostupni su u više nijansi. U tom slučaju korisnik prilikom kupnje odabire nijansu koju želi kupiti i dodati u košaricu. Pošto svaka nijansa ima količinu u kojoj je dostupna, korisnik ne može dodavati količine nijanse koje prelaze količinu određenu na skladištu. Ukoliko neka nijansa više nije dostupna, neće biti prikazana ispod proizvoda. Slika 24 prikazuje proizvod i dostupne nijanse proizvoda koje su vidljive korisniku.



Slika 24 - Prikaz proizvoda dostupnog u više nijansi

Nakon što se doda proizvod određene nijanse u košaricu, bit će vidljiva nijansa proizvoda koja je odabrana. Ukoliko se ne odabere nijedna od dostupnih nijansi, u košaricu se dodaje univerzalna nijansa tog proizvoda. Isti proizvod različite nijanse u košaricu je dodan kao dva različita proizvoda kako bi korisnik lakše povećavao/smanjivao količinu proizvoda te nijanse u košarici, ali i uklonio proizvod iz košarice. Ukoliko se jedna nijansa ukloni, druga nijansa ostaje u košarici sve dok se i ona ne ukloni ili kupi. Isto tako, mijenjanje količine jedne nijanse ne utječe na količinu istog proizvoda druge nijanse. Proizvodi koji imaju nijanse gledaju se kao potpuno različiti proizvodi. U ponudi oni su ipak prikazani kao jedan proizvod, samo su ispod označene dostupne nijanse. Slika 25 prikazuje situaciju kad je isti proizvod različitih nijansi dodan u košaricu.



Slika 25 - Dodavanje proizvoda različite nijanse u košaricu

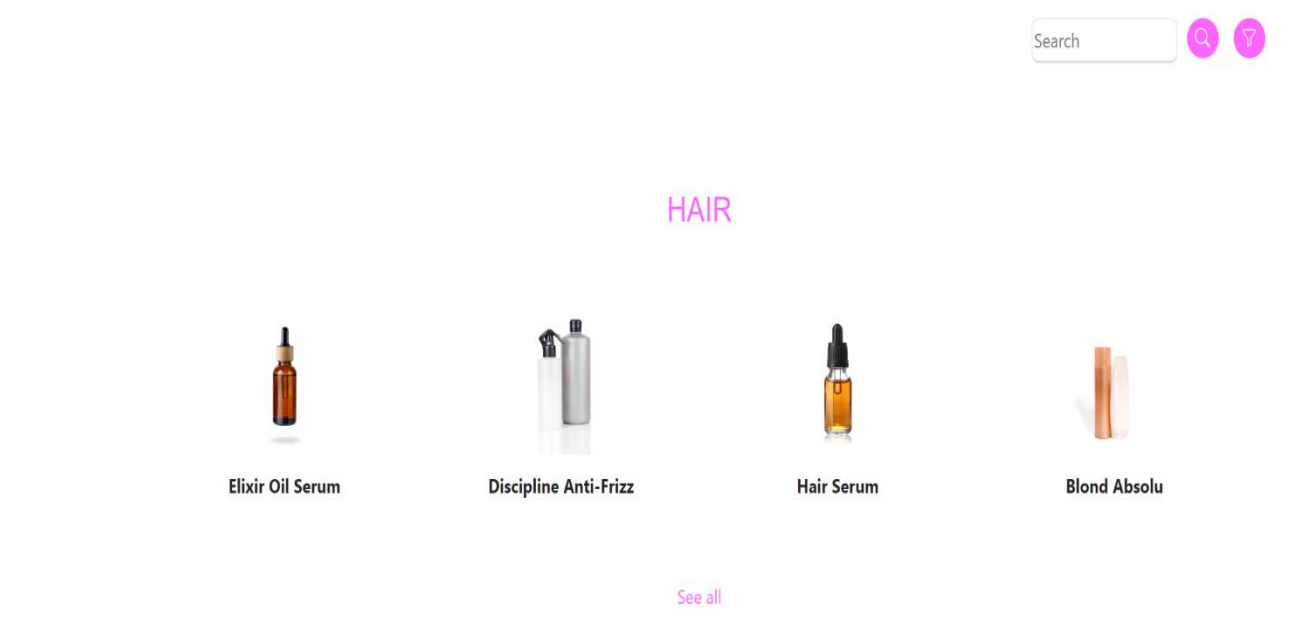
Ova funkcionalnost postignuta je na način da se ne uspoređuje samo identifikacijska oznaka proizvoda u košarici, već i nijansa proizvoda. Ovo je bilo važno kod uklanjanja, smanjivanja i povećavanja količine proizvoda u košarici. Sve ove funkcionalnosti trebale su se filtrirati po oba parametra, u protivnom kad se želi ukloniti jedna nijansa, uklonile bi se sve nijanse istog proizvoda. U Ispisu 22 vidljivo je na koji način je napisana funkcija za uklanjanje proizvoda iz košarice.

```
function removeFromCart(product, shade){
    setCart(cart.filter(function(element) {
        return element._id !== product._id || element._id ===
        product._id && element.shade._id !== product.shade._id;
    })))
}
```

Ispis 22 - Funkcija za uklanjanje elementa iz košarice

3.5. Korisničko sučelje za neregistriranog korisnika

Gost je uloga s najmanjim brojem funkcionalnosti u aplikaciji. Može pregledati ponudu aplikacije, ali ne može napraviti nikakvu funkcionalnost nad proizvodom. Njegova stranica s ponudom proizvoda prikazana je na Slici 26.



Slika 26 – Ponuda iz perspektive neregistriranog korisnika

S ulogom gosta, proizvodi se mogu samo gledati, pretraživati ili filtrirati po cijeni. Gost je osoba koja nema kreirani korisnički račun u aplikaciji i koja nije prijavljena u aplikaciju. Drugim riječima, u lokalnoj pohrani ne postoji spremljen JWT token. Pomoću postojanja JWT tokena u lokalnoj pohrani određuje se uloga trenutnog korisnika, a ako postoji JWT token onda i dostupne funkcionalnosti za prijavljenog korisnika.

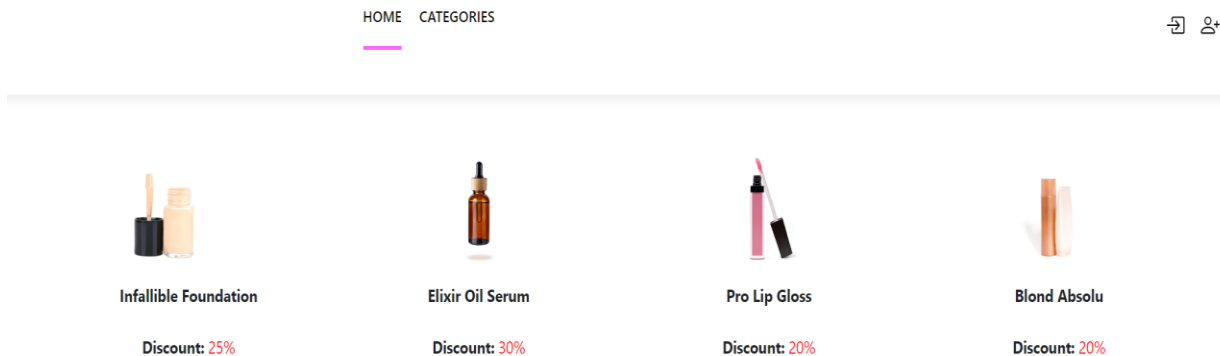
Neregistrirani korisnik, kao i registrirani korisnik, može pratiti proizvode koji su na sniženju odnosno koji su sniženi više od 15%. Klikom na poveznicu „Best attack“ prikažu se proizvodi čiji postotak sniženja je veći od 15%. Na ovaj način korisnik/gost može vidjeti postoje li neki

isplativi proizvodi za njega. Ispis 23 prikazuje kôd pomoću kojega su filtrirani proizvodi s postotkom sniženosti većim od 15%.

```
productRouter.get('/bestAttack', cors(), (req, res)=>{
    Product.find({discount: { $gte: parseInt(15) }}, (err,
products)=>{
        if(err){
            console.log(err)
        }
        else{
            console.log(products)
            return res.json(products);
        }
    })
})
```

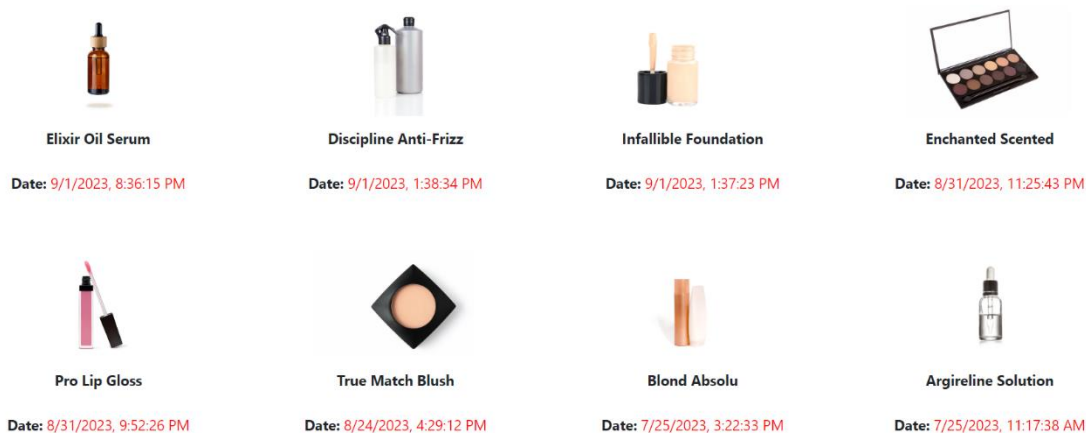
Ispis 23 - Filtriranje proizvoda prema postotku sniženosti

Nakon što su podaci filtrirani, prikazani su na stranici s označenim postotkom sniženosti. Na Slici 27 vidljivo je kako gost nema mogućnost naručiti navedene proizvode, jer nema kreiran korisnički račun. Prikaz filtriranih proizvoda s većim postotkom sniženosti na jednom mjestu može mu privući pažnju, navesti ga da izradi korisnički račun i na kraju kupi proizvod.



Slika 27 - Prikaz proizvoda s većim postotkom sniženosti

Osim sniženih proizvoda može vidjeti i neke nove proizvode, ali ni nad njima ne može izvesti nikakvu funkcionalnost dok ne izradi korisnički račun. Proizvodi su prikazani na sličan način kao i prethodno prikazani, samo što su različito filtrirani. Slika 28 prikazuje proizvode filtrirane po datumu.



Slika 28 - Prikaz novih proizvoda za neregistriranog korisnika

Već i na samoj početnoj stranici, neregistriranom korisniku kao i ostalima, omogućeno je vidjeti snižene proizvode kako bi se povećala mogućnost kupnje nekih proizvoda. Proizvodi na početnoj stranici isto su prikazani s istaknutim postotkom sniženosti kako bi prilikom samog dolaska na stranicu navelo posjetitelja na zainteresiranost. Slika 29 prikazuje dio sniženih proizvoda na početnoj stranici.



Slika 29 - Isticanje postotka sniženja u svrhu zainteresiranosti

Postotak sniženosti uvećan je i prikazan upečatljivim dizajnom kako bi neregistrirani korisnik prilikom pregleda obratio pozornost na snižene proizvode. Dizajn je postignut pomoću CSS-a. Ispis 24 prikazuje na koji način je postignut dizajn postotka sniženosti u CSS-u.

```
.saleDiscount {  
    border-radius: 50%;  
    position: relative;  
    top: -270px;  
    right: -350px;  
    background-color: #ff66ff;  
    color: #fff;  
    height: 50px;  
    width: 50px;  
}
```

Ispis 24 - Uređivanje postotka sniženosti na upečatljiv način

Gost ima mogućnost registracije, odnosno kreiranja korisničkog računa kako bi dobio veće ovlasti. Prilikom registracije potrebno je popuniti formu za registraciju ispravnim podacima. Ispravni podaci definirani su pomoću regularnih izraza. Primjer je lozinka, treba imati najmanje osam znakova, uključujući jedno slovo i jedan broj. Isto tako elektronička adresa mora ispuniti pravila definiranog regularnog izraza. Lozinka i ponovljena lozinka se moraju podudarati. Dok sva pravila nisu ispunjena i polja popunjena korisnik ne može izraditi korisnički račun. Nakon uspješne izrade korisničkog računa, korisnika se preusmjeri na stranicu za prijavu. Ako je prijava uspješna, ima sve funkcionalnosti registriranog korisnika. Slika 30 prikazuje izgled forme za registraciju.

Signup

Name

Surname

Username

Email

Password

Repeat password

Register

Slika 30 – Prikaz forme za registraciju

Prilikom kreiranja korisničkog računa, prije spremanja podataka u bazu podataka, lozinku je potrebno pretvoriti u šifrirani oblik. U aplikaciji za pretvaranje izvornog teksta lozinke u šifrirani oblik korišten je bcrypt modul. Primjer pretvaranja lozinke u prigodan oblik prikazan je u Ispisu 25.

```
const saltRounds = 10;

bcrypt.genSalt(saltRounds, function(err, salt) {

  bcrypt.hash(req.body.password, salt, function(err, hash) {

    let user = new User({name: req.body.name,
      surname:req.body.surname, username:req.body.username,
      email: req.body.email, password: hash, role:
      req.body.role, purchasesNum: req.body.purchasesNum,
      discount: req.body.discount });

    user.save();

    return res.json(user);});

});
```

Ispis 25 – Pretvaranje lozinke u šifrirani oblik

4. Zaključak

U radu su prikazane važnije funkcionalnosti aplikacije i izgled korisničkog sučelja. Priložene su slike sučelja kako bi se lakše vizualizirale specifične funkcionalnosti, ali i izgled same aplikacije. Pojednostavljena je kupnja željenih proizvoda tako da korisnik može što jednostavnije doći do željenog proizvoda. Dizajn aplikacije je minimalistički i korisnički orijentiran. Boje i palete su birane u odnosu na temu (sve je u ružičastim nijansama), ali sama pozadina je bijela kako cijeli dojam aplikacije ne bi bio napadan za korisnika. Važniji botuni su označeni upadljivijim bojama kako bi privukle pozornost promatrača. Aplikacija je intuitivna za korištenje te je princip sličan kao i kod ostalih web trgovina. Glavne funkcionalnosti izvode se na intuitivan način (dodaje se u košaricu klikom na botun, plaća se klikom, unose se podaci o kartici) kako bi ovaj način kupnje bio pristupačan za svakog korisnika bez obzira na dobnu skupinu kojoj pripada.

React biblioteka je praktična za izradu ovakvog tipa aplikacije. Koristeći stanja prilikom kodiranja moguće je stvoriti dinamične aplikacije na kojima se korisniku odmah ažurira svaka promjena na stranici. Ovakve dinamične stvari vidljive su u aplikaciji, a konkretan primjer je košarica. Nakon unosa aktivnog koda za popust smanji se cijena košarice ili prilikom dodavanja proizvoda u košaricu iznad ikone košarice ažurira se broj proizvoda. Na taj način je jednostavnije pratiti što se događa na stranici, ali i je li određena funkcionalnost odrađena ili postoji neki problem zbog kojega se funkcionalnost ne može izvesti.

Osim glavnih funkcionalnosti koje su navedene u razradi, implementirane su i neke dodatne kao što su mogućnost komentiranja ili recenziranja kupljenih proizvoda. Naravno, uvijek postoji mjesta za poboljšanje i napredak aplikacije. Neka korisna poboljšanja bila bi dodavanje chat opcije za dodatne informacije o proizvodu u realnom vremenu ili primjerice implementacija praćenja statusa dostave u obliku karte tako da korisniku bude vidljiva i lokacija trenutnog tijeka dostave, a ne samo status.

Literatura

- [1] Gillis, A: “MongoDB”, <https://www.techtarget.com/searchdatamanagement/definition/MongoDB> (posjećeno 22.07.2023.)
- [2] “Node.js Introduction”, https://www.w3schools.com/nodejs/nodejs_intro.asp (posjećeno 22.07.2023.)
- [3] Sufian, T: “What is Node.js: A Comprehensive Guide”, <https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs> (posjećeno 22.07.2023)
- [4] Herbert, D: “What is React?”, <https://blog.hubspot.com/website/react-js> (posjećeno 22.07.2023.)
- [5] Deshpande, C: “The Best Guide to Know What Is React”, <https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs> (posjećeno 25.07.2023.)
- [6] “React: The Virtual DOM”, <https://www.codecademy.com/article/react-virtual-dom> (posjećeno 01.09.2023.)
- [7] “React Single Page Application”, <https://www.bairesdev.com/blog/react-spa-single-page-application/> (posjećeno 01.09.2023.)
- [8] “CSS – Cascading Style Sheets”, <https://developer.mozilla.org/en-US/docs/Web/CSS> (posjećeno 25.07.2023.)
- [9] “CSS Syntax, https://www.w3schools.com/css/css_syntax.asp (posjećeno 25.07.2023.)
- [10] “Karnik, N: Introduction to Mongoose for MongoDB”, <https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/> (posjećeno 01.09.2023.)
- [11] “Node.js - Express framework ”, https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm (posjećeno 01.09.2023.)
- [12] “Postman Tutorijal ”, <https://www.javatpoint.com/postman> (posjećeno 01.09.2023.)