Aplikacija za aukcije automobila

Mrdalj, Denis

Undergraduate thesis / Završni rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:145649

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-21**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

Aplikacija za aukcije automobila

Završni rad

Denis Mrdalj

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 12.09.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Denis Mrdalj
Studij, smjer:	Računalno inženjerstvo
Mat. br. Pristupnika, godina upisa:	R 4406, 12.10.2021.
OIB Pristupnika:	48797048948
Mentor:	izv. prof. dr. sc. Alfonzo Baumgartner
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Aplikacija za aukcije automobila
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rad:	[Rezervirano: Denis Mrdalj] Izraditi aplikaciju koja korisnicima omogućuje praćenje i sudjelovanje u aukcijama automobila. Omogućiti vremenski istek aukcije i automatsko produljenje na kraći vremenski period ako dođe nova ponuda u zadnjoj minuti. Aukcije imaju prostor za komentare dostupan samo za registrirane korisnike. Pretraga automobila sadrži razne filtre i načine sortiranja rezultata.
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	12.09.2023.
Datum potvrde ocjene od strane Odbora:	24.09.2023.

Potvrda mentora o predaji konačne verzije rada:	Mentor elektronički potpisao predaju konačne verzije.
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 25.09.2023.

Ime i prezime studenta:	Denis Mrdalj
Studij:	Računalno inženjerstvo
Mat. br. studenta, godina upisa:	R 4406, 12.10.2021.
Turnitin podudaranje [%]:	8

Ovom izjavom izjavljujem da je rad pod nazivom**: Aplikacija za aukcije automobila**

izrađen pod vodstvom mentora izv. prof. dr. sc. Alfonzo Baumgartner

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD **1**

1.1. Zadatak završnog rada **1**

2. PREGLED PODRUČJA TEME **2**

2.1. eBay Motors **2**

2.2. Bring a Trailer **3**

2.3. Cars and Bids **4**

2.4. Collecting Cars **4**

2.5. Copart **5**

3. KORIŠTENE TEHNOLOGIJE **6**

3.1. TypeScript **6**

3.2. React **7**

3.3. HTML **8**

3.4. CSS **8**

3.5. Firebase **9**

3.6. REST API **9**

3.7. Visual Studio Code **10**

4. IMPLEMENTACIJA APLIKACIJE **11**

4.1. Frontend **13**

4.2. Backend **22**

5. TESTIRANJE RADA APLIKACIJE **25**

6. ZAKLJUČAK **32**

LITERATURA **33**

SAŽETAK **34**

ABSTRACT **35**

1. UVOD

Tema ovog završnog rada je web aplikacija za aukcije automobila. Aplikacija korisnicima pruža mogućnost pregleda, pretraživanja te sudjelovanja u trenutno aktivnim aukcijama. Pregled i pretraživanje automobila aktivnih aukcija dostupno je za sve korisnike pa tako i neregistrirane. Međutim, ako korisnik želi licitirati i dodavati svoje aukcije mora biti registriran. Aukcije su prikazane u kartičnom obliku s osnovnim informacijama, a dodatne informacije o vozilu nalaze se na stranici aukcije. Aplikacija je namijenjena svim ljubiteljima zanimljivih i rijetkih automobila koji su zainteresirani za kupovinu novog automobila kroz vrlo jednostavno i intuitivno sučelje.

Za izradu ove web aplikacije korištene su tehnologije poput programskog jezika *TypeScript* te dodatnih biblioteka *React* i *Firebase*. Biblioteke poput *React-a* znatno olakšavaju stvaranje web aplikacija. *Firebase* nudi razne alate koji su vrlo korisni za stvaranje baze podataka ove aplikacije. Za strukturu i dizajn web aplikacije korišten je opisni jezik HTML (engl. *Hypertext Markup Language*) te stilski jezik CSS (engl. *Cascading Style Sheets*) uz dodatnu biblioteku *Tailwind* CSS.

U drugom poglavlju opisane su web stranice za aukcije automobila koje su služile kao inspiracija ove aplikacije.Treće poglavlje opisuje odabir tehnologija poput programskih jezika i razvojnih okvira koje su korištene u razvoju ove aplikacije. U četvrtom poglavlju detaljno je objašnjen kod za ostvarivanje raznih funkcionalnosti aplikacije. Peto poglavlje rada prikazuje izgled i način rada aplikacije iz perspektive korisnika.

1.1. Zadatak završnog rada

Zadatak ovog završnog rada je napraviti *React* web aplikaciju za aukcije automobila. Aplikacija podržava registracije i prijave novih korisnika unosom e-mail adrese i lozinke. Neregistrirani korisnici imaju mogućnost pregleda i pretraživanja dostupnih aukcija, ali u njima ne mogu sudjelovati ako nisu registrirani. Registrirani korisnici imaju mogućnost postavljanja ponuda na aukcije i dodavanja novih. Svaka aukcija sadrži galeriju fotografija te detaljan opis karakteristika vozila. Prilikom dodavanja nove aukcije, korisnik može odabrati vrijeme trajanja aukcije.

2. PREGLED PODRUČJA TEME

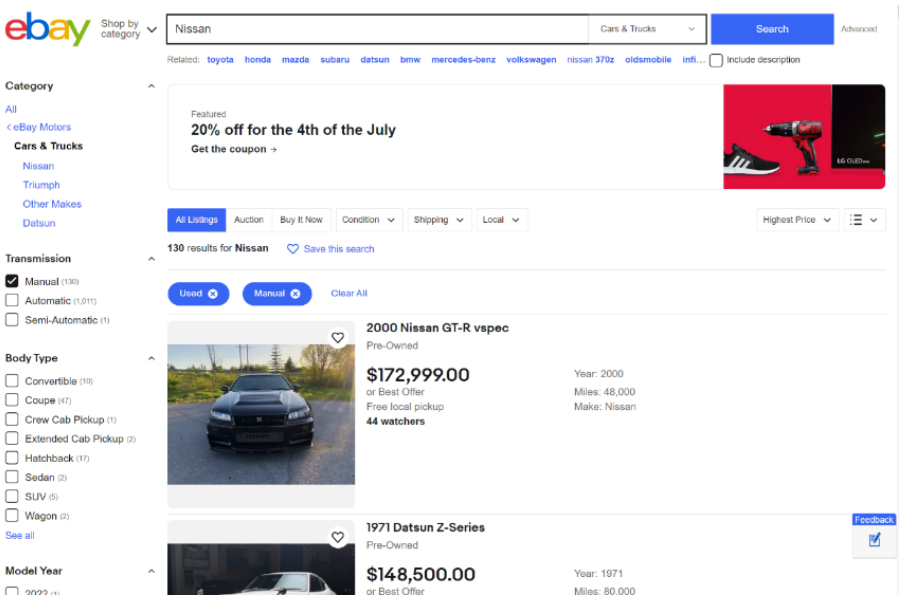
Web stranice za aukcije automobila postaju sve popularnije. Kupci i prodavači sve se više okreću aukcijskom obliku kupovine i prodaje vozila zbog jednostavnosti korištenja te brze prodaje i kupovine. Zbog relativno kratkog trajanja aukcije, ovakve web stranice brzo izgrade veliku zajednicu korisnika koji vrlo aktivno provjeravaju stanje dostupnih aukcija i sudjeluju u razgovorima s ostalim korisnicima stranice. Ovako visoka razina aktivnosti i angažmana korisnika vrlo je važna prodavačima koji mogu biti sigurni da će njihovu aukciju vidjeti gotovo svi korisnici stranice. Zbog natjecateljske prirode aukcija, vozila se često prodaju po neočekivano visokim cijenama. Model online aukcija vozila pokazao se vrlo profitabilan.

Jedna od ključnih prednosti korištenja web stranica za aukcije automobila je mogućnost pristupa velikom broju vozila raznih tipova bez potrebe fizičkog prisustva. Također, aukcije sadrže vrlo detaljne informacije o vozilu, uključujući tehničke podatke, povijest vozila te veliki broj fotografija. Neke stranice nude i mogućnost provjere vozila putem izvještaja o povijesti vozila, što je vrlo korisno za kupce koji žele biti sigurni u stanje vozila prije sudjelovanja u aukciji.

U nastavku je navedeno nekoliko najpoznatijih web stranica za aukcije automobila.

2.1. eBay Motors

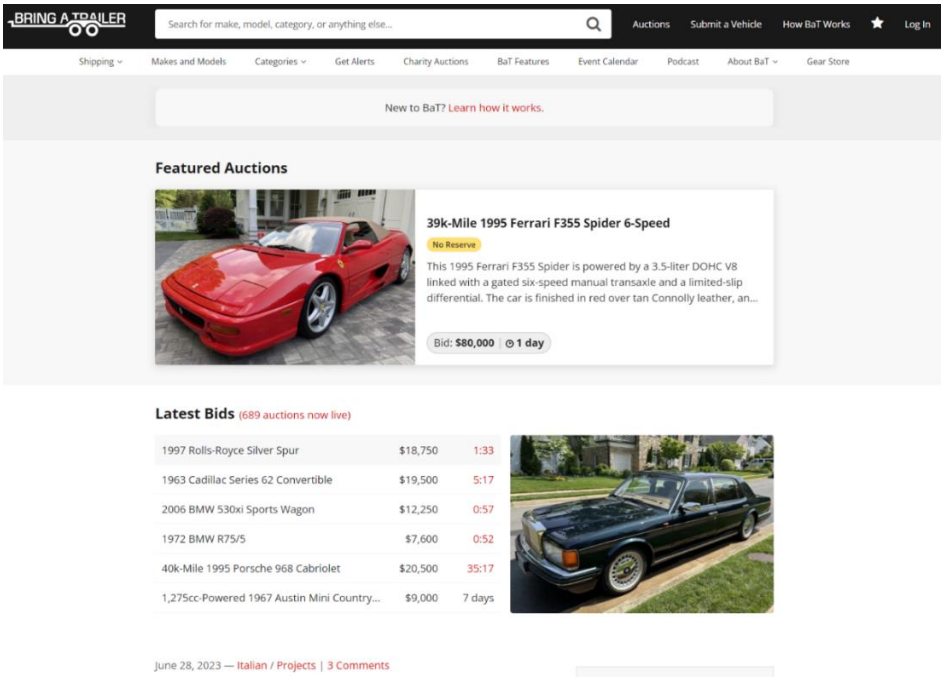
eBay Motors je vodeća, a ujedno i najveća platforma za aukcije automobila. Osim novih i rabljenih vozila, omogućuje i jednostavno pretraživanje auto dijelova. *eBay Motors* dostupan je svim registriranim korisnicima *eBay* platforme. Vozila mogu prodavati privatnici i auto kuće, a kupci ne plaćaju proviziju prilikom kupovine. Princip aukcije je jednostavan. Najviša ponuda pobjeđuje aukciju. Kao i u ovoj aplikaciji, korisnik mora imati korisnički račun kako bi sudjelovao u licitaciji. Primjer pretrage vozila prikazan je na slici 2.1.



Slika 2.1. Pretraga *Nissan* vozila na web stranici *eBay Motors*

2.2. Bring a Trailer

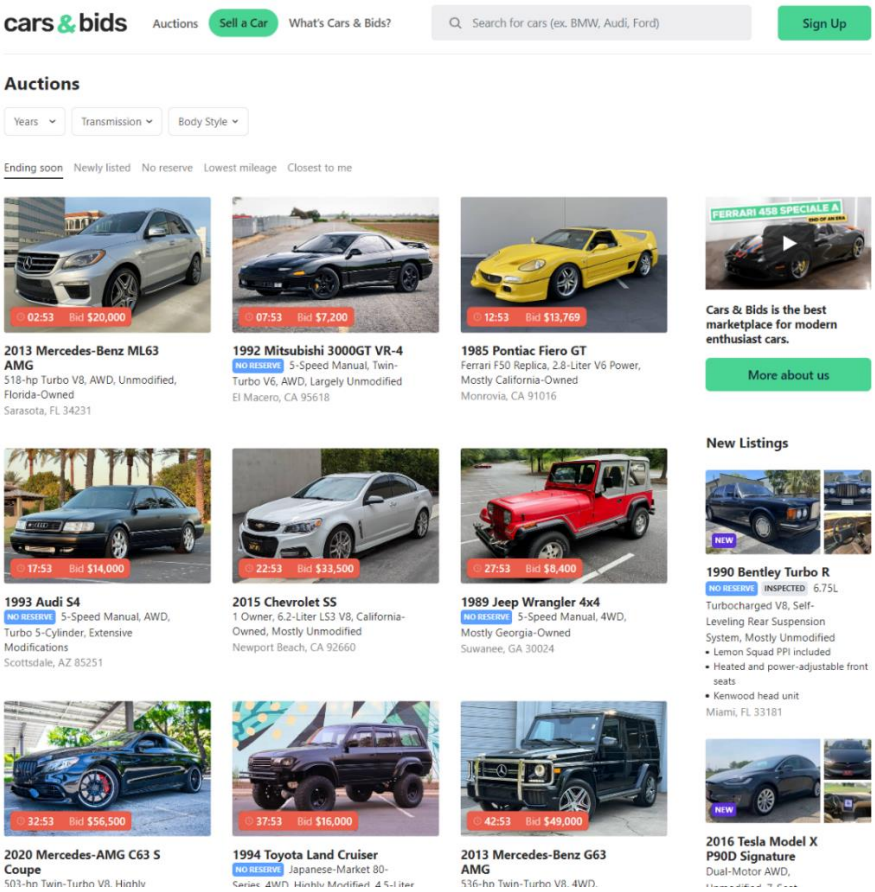
Bring a Trailer je popularna web stranica za aukcije automobila koja se ističe po svojoj usredotočenosti na klasične i starinske automobile. Ova platforma je stekla kultni status među ljubiteljima automobila i kolekcionarskih vozila. Stranica pruža korisnicima pregled, licitiranje i kupnju raznih automobila, uključujući klasične, sportske i jedinstvene automobile. Također su dostupna i druga vozila poput motocikala, kamiona i nekih posebnih vozila. Izgled početne stranice prikazan je na slici 2.2.



Slika 2.2. Početna stranica na stranici *Bring a Trailer*

2.3. Cars and Bids

Cars and Bids je relativno nova platforma za aukcije automobila s naglaskom na moderna, sportska i rijetka vozila. Na stranici se često nalaze ekskluzivni primjerci automobila, kao i rijetki modeli koji se teško mogu pronaći na drugim mjestima. Jedna od ključnih značajki stranice *Cars and Bids* je pažljivo odabrana ponuda vozila. Svaki automobil na aukciji je detaljno pregledan i opisan, a prati ga velik broj visokokvalitetnih fotografija. *Cars and Bids* također ima aktivnu zajednicu ljubitelja automobila koji mogu sudjelovati u raspravama, dijeliti mišljenja i iskustva, te povezati s drugim članovima zajednice. Prikaz aukcija koje uskoro istječu nalazi se na slici 2.3.

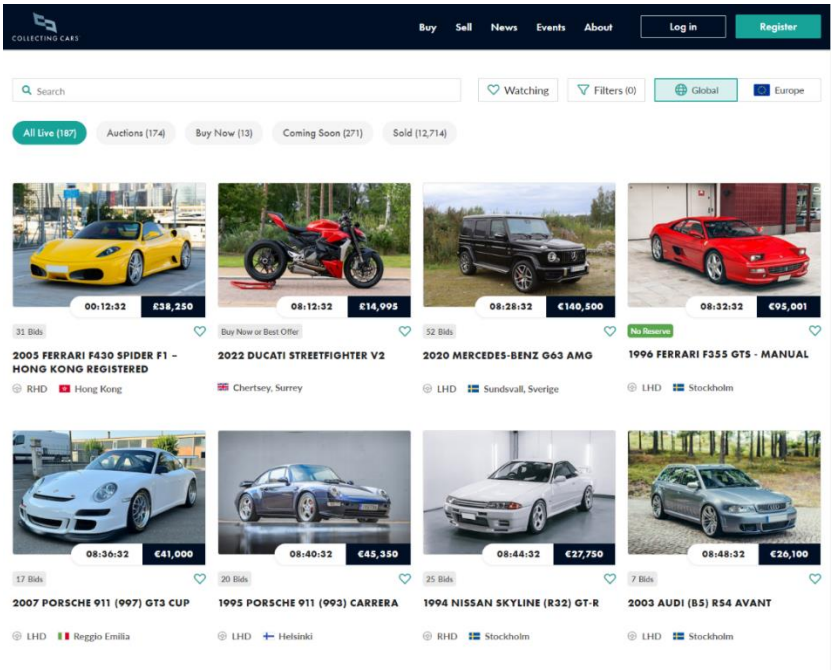


Slika 2.3. Aukcije koje uskoro istječu na stranici Cars and Bids

2.4. Collecting Cars

Collecting Cars je nova platforma za aukcije s naglaskom na kolekcionarska vozila. Iako je prvo započela u Ujedinjenom Kraljevstvu, brzo je prerasla u globalno tržište s uredima u Europi, Australiji i Sjedinjenim Američkim Državama. Platforma nudi niz jedinstvenih mogućnosti poput Buy Now opcije koja omogućuje kupnju vozila čak i nakon isteka aukcije. Provizije za kupce

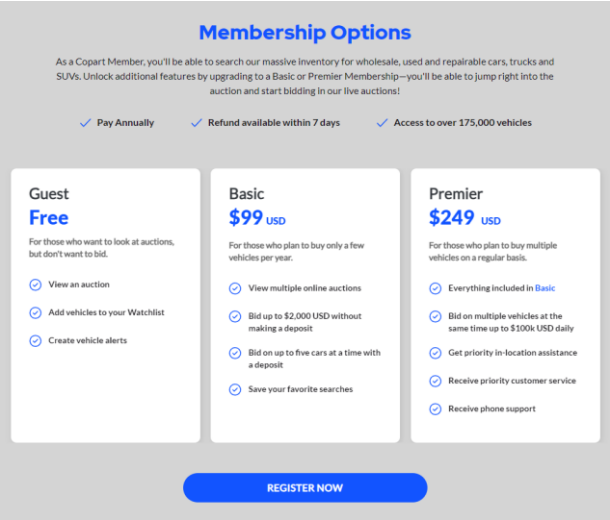
kreću se od 6% do 7.2% ovisno o regiji. Korisničko sučelje ove platforme poslužilo je kao inspiracija za korisničko sučelje ove aplikacije. Izgled sučelja prikazan je na slici 2.4.



Slika 2.4. Aukcije na stranici *Collecting Cars*.

2.5. Copart

Copart je globalna aukcijska platforma za automobile, kamione, brodove, motocikle te razna druga vozila. Ističe se po sustavu plaćenog članstva koji nudi razne prednosti u odnosu na korisnike s besplatnom članarinom. Svako vozilo je detaljno provjereno te je na svakoj aukciji izlistano stanje vozila i potencijalni nedostatci. Korisnici mogu zatražiti obavijesti za aukciju koju žele pratiti. Pregled članskih ponuda nalazi se na slici 2.5.



Slika 2.5. Članske ponude na stranici *Copart*.

3. KORIŠTENE TEHNOLOGIJE

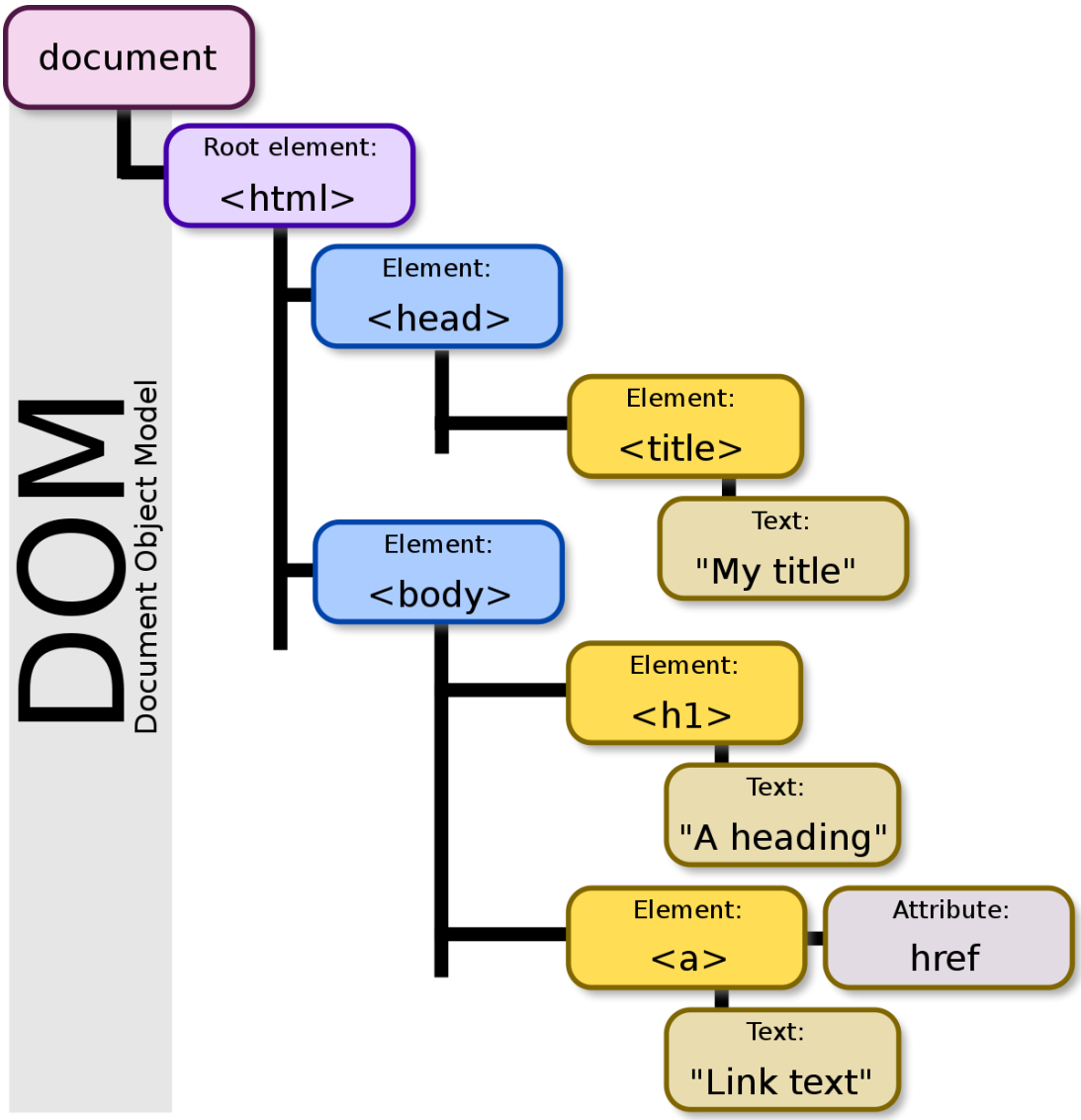
3.1. TypeScript

TypeScript je vrlo popularan programski jezik koji je osmišljen kao nadogradnja programskog jezika *JavaScript*. Iako je sintaksa vrlo slična između ta dva jezika, *TypeScript* donosi niz prednosti za razvoj složenijih aplikacija. Glavna je prednost statička tipizacija koja omogućava definiranje prilagođenih tipova za varijable, funkcije, povratne vrijednosti te parametre koji će se koristiti. Definiranje tipova doprinosi boljoj organizaciji koda te lakšem održavanju. *TypeScript* također podržava sve što i *JavaScript*, uključujući objektno orijentirano programiranje, *Document Object Model* (DOM) te sve ostale značajke koje čine *JavaScript* jednim od najkorištenijih programskih jezika za razvoj web aplikacija. DOM predstavlja strukturu stabla elemenata HTML dokumenta gdje je svaki element objekt. Iako uvodi mnoge korisne značajke *JavaScript-u*, *Typescript* se prilikom izvođenja prevodi u *JavaScript* kod pa je rezultat *JavaScript* datoteka koja se može izvršiti u svim modernim web preglednicima. Primjer definiranja prilagođenog tipa podatka prikazan je na slici 3.1.

```
type Person = {
    name: string;
    age: number;
};

function greet(person: Person) {
    return "Hello " + person.name;
}
```

Slika 3.1 Primjer definiranja prilagođenog tipa podatka *Person* i korištenja unutar funkcije.

JavaScript, pa isto tako i *TypeScript* koristi se za izradu interaktivnih i dinamičnih web stranica te stvaranje složene logike i funkcionalnosti što nije moguće ostvariti čistim HTML kodom. Primjeri uključuju prikazivanje i sakrivanje elemenata, promjenu boje ili fonta, validaciju obrazaca i slično. Sadržaj web stranice reagira na korisničke radnje te se dinamički mjenja. Osnovni je jezik koji se izvršava u web preglednicima i omogućuje interakciju s korisnicima. Primjer strukture stabla DOM modela nalazi se na slici 3.2.



Slika 3.2. Primjer strukture stabla DOM modela

3.2. React

React je popularna *JavaScript* biblioteka za izradu korisničkog sučelja u web aplikacijama. Razvijen je od strane *Facebook-a* kako bi olakšao razvoj interaktivnih, složenih i brzih korisničkih sučelja. Glavni element *React* biblioteke je komponenta. Komponente su modularne, samostalne jedinice koje mogu prikazivati dinamičke podatke i reagirati na korisničke interakcije. Svaka komponenta sadrži vlastiti HTML kod, CSS stil i logiku. Korištenjem komponenti razvoj i održavanje aplikacije je jednostavno i organizirano.

Jedna od ključnih značajki biblioteke *React* je *Virtual Document Object Model* (virtualni DOM). Dok *JavaScript* samostalno stvara svoj DOM, *React* koristi virtualni DOM kako bi učinkovito ažurirao samo izmjenjene dijelove korisničkog sučelja umjesto cijelog dokumenta.

Još jedna prednost korištenja *React-a* je upotreba *JavaScript Syntax Extension-a* (JSX). JSX je proširenje *JavaScript*-a koje omogućuje pisanje HTML koda unutar *JavaScript* datoteka. Na taj način olakšava razvoj korisničkog sučelja. Analogno, *TypeScript* također podržava upotrebu sličnog proširenja *TypeScript Syntax Extension* (TSX). TSX znatno olakšava definiranje HTML elemenata i CSS stilova za pojedine elemente te je omogućen u svim datotekama s *.tsx* nastavkom.

3.3. HTML

HTML (engl. *Hyper Text Markup Language*) je osnova svakog web razvoja i ključan je za stvaranje struktuirane web stranice. Odgovoran je za organizaciju sadržaja i povezivanje različitih elemenata na web stranici. Iako je sličan programskom jeziku, HTML je opisni jezik za opis strukture sadržaja stranice uporabom posebnih oznaka. Svaka oznaka (engl. *tag*) ima otvarajuću i zatvarajuću oznaku između kojih se nalazi određeni sadržaj. Primjerice, oznaka <p> označava paragraf, <h1> označava naslov prvog stupnja, <img> označava prikaz slika i tako dalje. Važna značajka HTML-a je mogućnost dodavanja atributa elementima. Atributi pružaju dodatne informacije i funkcionalnosti elementima.

HTML kod piše se unutar HTML dokumenta koji započinje osnovnom oznakom <html>. Struktura složene web stranice sastoji se od zaglavlja, navigacijske trake, glavnog sadržaj, bočne trake i podnožja. Za definiranje sadržaja zaglavlja koristi se oznaka <header>, za navigaciju <nav>, za glavni sadržaj <main> u kombinaciji s oznakama <article>, <section> i , bočna traka definira se oznakom <aside>, a podnožje oznakom <footer>. U ovoj aplikaciji kreirane su posebne *React* komponente za navigacijsku traku i podnožje koje koriste <nav> i <footer> oznake.

3.4. CSS

Za stvaranje vizualnog izgleda stranice, uz HTML se često koristi i CSS (engl. *Cascading Style Sheets*). CSS je stilski jezik koji omogućuje kontrolu nad stilovima, fontovima, bojama, tablicama, veličinama i raznim drugim elementima web stranice. Dok se HTML koristi za definiranje strukture sadržaja web stranice, CSS se koristi za definiranje izgleda elemenata stranice.

U ovoj aplikaciji korišten je razvojni okvir *Tailwind* CSS. HTML i CSS obično su odvojene datoteke, međutim *Tailwind* omogućuje stiliziranje elemenata putem klasa koje se koriste direktno u HTML kodu. *Tailwind* koristi takozvani „*utility-first*“ pristup što znači da pruža gotove utilitarne klasne atribute za stiliziranje elemenata poput *text-center, bg-blue, font-bold* i razne druge. Korištenjem *Tailwind*-a razvoj aplikacije mnogo je brži i lakši jer ne postoji potreba za posebnim

definiranjem klasa u CSS datoteci. Naravno, korištenje tradicionalnog CSS-a još uvijek je dostupno i u potpunosti podržano.

CSS se sastoji od pravila koja se primjenjuju na određene elemente. Svako pravilo sastoji se od selektora i deklaracija. Selektor određuje na koje se elemente primjenjuje stil, a deklaracija definira konkretne stilove. CSS također podržava stvaranje klasa i identifikatora za ciljano uređivanje specifičnog HTML elementa. Klasa se elementu dodjeljuje atributom *class*, dok se identifikator dodjeljuje atributom *id*. CSS također podržava niz naprednih stilova poput gradijenata, animacija, tranzicija, fleksibilnih tablica i drugih.

Upotreba CSS-a značajno poboljšava estetski dojam web stranice. Odvajanje stilova od strukture sadržaja omogućuje fleksibilnost u promjeni izgleda stranice bez potrebe za izmjenom HTML koda.

3.5. Firebase

Firebase nudi razne funkcionalnosti za web i mobilne aplikacije koje koriste baze podataka. Platforma pruža širok spektar alata i usluga koji olakšavaju razvoj aplikacija bez potrebe za složenom konfiguracijom infrastrukture te održavanjem *backend* koda. Pojam *backend* odnosi se na dio aplikacije koji se brine o pozadinskim funkcionalnostima kojih korisnik nije svijestan poput baze podataka, autentifikacije korisnika, sigurnosti i slično.

Neki od ključnih servisa Firebase platforme su *Realtime* baze podataka, autentifikacija korisnika, *Web Hosting*, *Cloud Firestore*, *Cloud Functions* i druge. Korištenjem *Firebase* servisa, developeri ne moraju sami stvarati vlastite servere, API-je (*Application Programming Interface*) i slično. Vrlo je jednostavan za korištenje, podržava niz funkcionalnosti i integraciju s drugim *Google* uslugama. Osim navedenog, platforma također pruža usluge analitike, testiranja, upravljanja korisnicima te mnoge druge.

3.6. REST API

API je skup alata i protokola koji omogućuje međusobnu komunikaciju između aplikacija. API nudi programerima pristup funkcionalnostima ili podacima izvan svoje aplikacije bez potrebe za razumijevanjem ili implementacijom složenih detalja i načina rada aplikacije.

REST API (*Representational State Transfer Application Programming Interface*) je vrsta API-ja koja slijedi principe REST arhitekture te omogućuje komunikaciju aplikacije sa poslužiteljem putem HTTP (*Hypertext Transfer Protocol*) metoda. REST API sadrži resurse poput podataka,

slika ili procesa kojima se pristupa preko jedinstvenog URL (*Uniform Resource Locator*) identifikatora. To znači da klijent može dobiti resurse s poslužitelja u raznim formatima poput XML (*Extensible Markup Language*), JSON (*JavaScript Object Notation*), HTML i slično.

Za potrebe demonstracije aplikacije, fotografije automobila dohvaćene su od strane vanjskog API-ja *Imagin Studio* koji nudi velik broj profesionalnih fotografija svih novih automobila.

3.7. Visual Studio Code

Visual Studio Code je jedno od najkorištenijih besplatnih razvojnih okruženja razvijeno od strane *Microsoft-a.* Iako je jednostavan za korištenje, nudi bogate mogućnosti za razvoj složenih aplikacija poput automatskog dovršavanja koda, isticanja i ispravljanja pogrešaka, pristup terminalu te raznolike podrške za programske jezike i okruženja. Zahvaljujući ovim prednostima te izuzetnoj preglednosti, Visual Studio Code postao je omiljen među programerima.

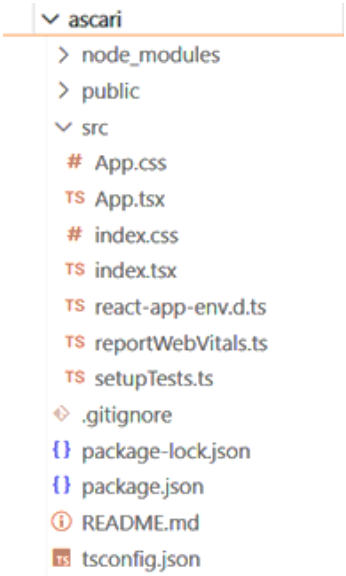
4. IMPLEMENTACIJA APLIKACIJE

Prvi korak u stvaranju *React* aplikacije je stvaranje projekta naredbom *npx create-react-app* nakon koje slijedi proizvoljan naziv aplikacije. NPX(*Node Package Execute*) se odnosi na upravitelj paketa koji će poslužiti za instalaciju raznih biblioteka tijekom razvoja aplikacije. *Create-react-app* je temeljna naredba koja stvara *React* projekt unutar definirane datoteke. *Template typescript* je dodatak naredbi koji *React-u* govori da će se koristiti *TypeScript*. Naredba se može pokrenuti u *cmd* i *PowerShell* prozorima ili direktno kroz terminal uređivača koda *Visual Studio Code*. Primjer stvaranja nove React aplikacije prikazan je na slici 4.1.

```
PS C:\Users\Dena\Documents> npx create-react-app ascari --template typescript

Creating a new React app in C:\Users\Dena\Documents\ascari.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template-typescript...

[#########.........] | idealTree:ascari: timing idealTree:#root Completed in 1616ms
```

Slika 4.1. Primjer stvaranja nove *React* aplikacije naziva *ascari*.

Nakon instalacije svih potrebnih paketa i ovisnosti, dobivamo gotovu strukturu aplikacije koja je prikazana na slici 4.2.



Slika 4.2. Dobivena struktura *ascari* aplikacije unutar *Visual Studio Code-a*.

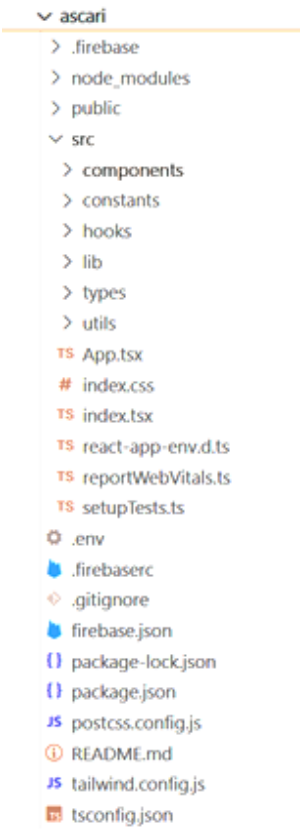
Većina koda potrebnog za ovu aplikaciju nalazit će se unutar *src* datoteke dok će *public* datoteka sadržavati sve potrebne resurse poput fotografija i vektora koji će se korsititi. Najvažnija datoteka je *App.tsx*. Unutar te datoteke definirane su rute (*Routes*) iz biblioteke *react-router-dom*. Biblioteku je potrebno dodatno instalirati naredbom *npm i react-router-dom*. Ova biblioteka

ključna je za navigiranje korisnika kroz aplikaciju. Kako bi *react-router-dom* pravilno radio, potrebno je unutar *index.tsx* omotati cijelu aplikaciju pozivom komponente *BrowserRouter*. Definiranje ruta prikazano je na slici 4.3.

```
function App() {
  return (
      <Routes>
        <Route path="/" element={<Search/>}/>
        <Route path="/cars/:id" element={<Auction/>}/>
        <Route path="/profile" element={<Profile/>}/>
        <Route path="/signin" element={<SignIn/>}/>
        <Route path="/signup" element={<SignUp/>}/>
        <Route path="/create" element={<Create/>}/>
        <Route path="/about" element={<About/>}/>
      </Routes>
  );
}
```

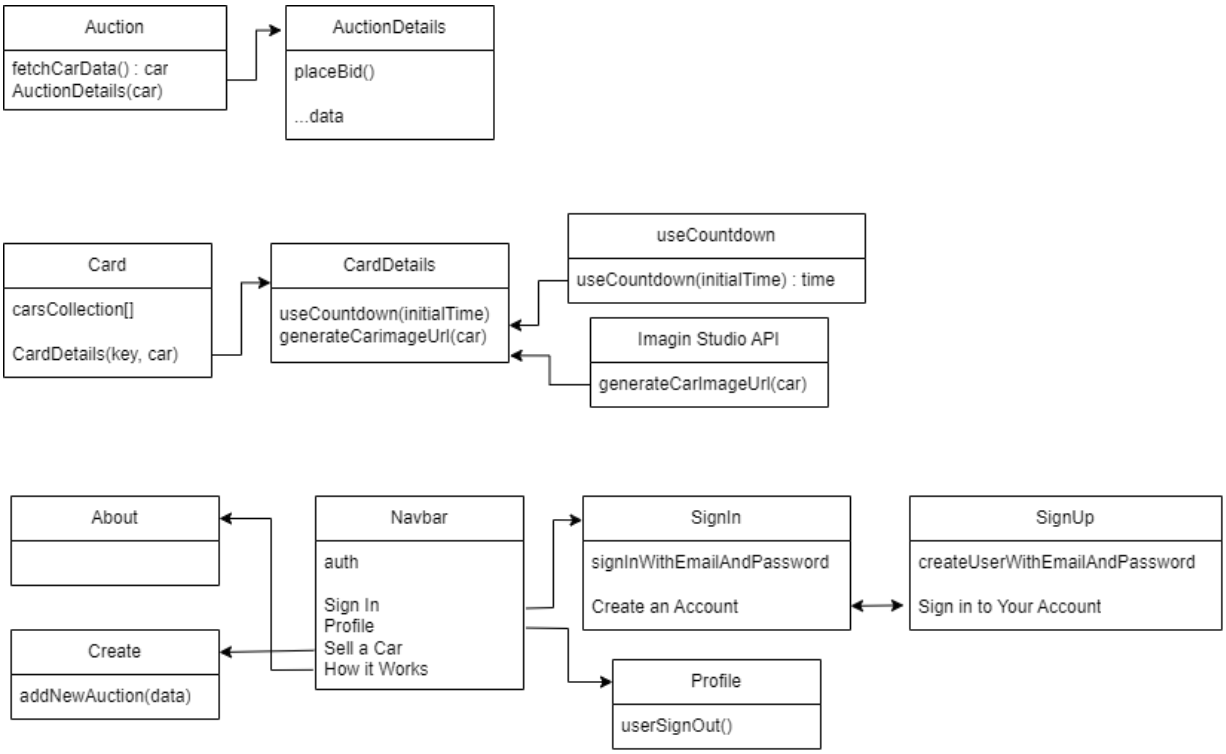
Slika 4.3. Definiranje ruta unutar *App.tsx*.

Struktura aplikacije proširena je dodatnim datotekama poput *components* koja sadrži sve korištene *React* komponente, *firebase* koja sadrži svu logiku potrebnu za *Firebase* funkcionalnosti, *types* u kojoj se nalaze svi personalizirani tipovi podataka te raznim drugim datotekama koje će biti dodatno objašnjene. Proširena struktura aplikacije prikazana je na slici 4.4.



Slika 4.4. Proširena struktura *ascari* aplikacije unutar *Visual Studio Code-a*.

Struktura i interakcija između komponenti aplikacije prikazana je dijagramom na slici 4.5.



Slika 4.5. Dijagram komponenti aplikacije.

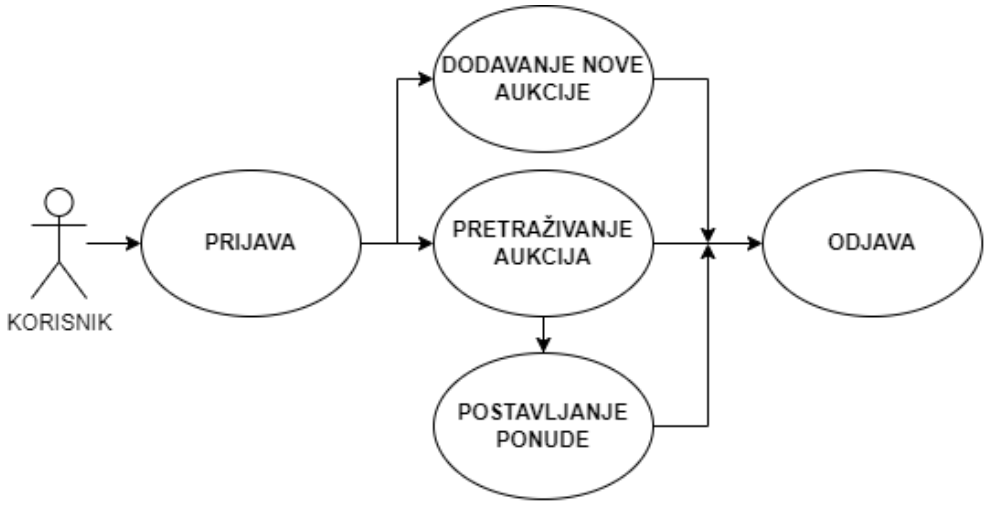
4.1. Frontend

Frontend aplikacije odnosi se na dio koda koji je odgovoran za stvaranje svih elemenata s kojima je korisnik u interakciji. To uključuje korisničko sučelje (*User Interface*) i iskustvo (*User Experience*), a dobiva se kombinacijom HTML-a, CSS-a i *TypeScript-a*. *Frontend* je ključan dio svake aplikacije jer utječe na zadovoljstvo korisnika i njihov angažman što indirektno doprinosi i uspjehu same aplikacije. Elementi korišteni u ovoj aplikaciji stilizirani su pomoću *Tailwind* CSS-a. Primjer stiliziranja prikazan je na slici 4.6.

```
<Link to="/"
    className='bg-white border-2 border-black text-black font-bold
    px-4 py-2 mt-6 rounded-full hover:bg-black hover:text-white'>Explore Live Auctions</Link>
```

Slika 4.6. Primjer stiliziranja *Link* elementa pomoću *Tailwind-a*

Korisnik ne mora biti prijavljen kako bi pristupio početnoj stranici aplikacije. Na početnoj stranici prikazane su sve aukcije te ih korisnik može pogledati. Također ima opciju pretraživanja aukcija prema raznim kriterijima, a rezultate pretrage moguće je i sortirati. Primjer scenarija korištenja aplikacije prikazan je dijagramom na slici 4.7.



Slika 4.7. Dijagram scenarija korištenja aplikacije.

Navigacijska traka aplikacije razlikuje se ovisno o tome je li korisnik prijavljen ili ne. Ako korisnik nije prijavljen, vidljive su samo dvije mogućnosti: *Sign In* gumb i *How it Works* poveznica. Dinamička navigacijska traka ostvarena je provjerom *authUser* koja vraća vrijednost true ako je korisnik prijavljen ili false ako nije. Komponenta navigacijske trake prikazana je na slici 4.8.

```
<nav className="bg-white bg-opacity-70 border-b-black-100 border flex-between p-8 h-28 sticky top-0">
  <Link to='/' className='justify-end items-end'>
    <img src="/logo.svg" alt="logo" width={130}/>
  </Link>
  <div className="flex">
    <Link  to="/about"
      className="flex text-[18px] items-center hover:text-gray-400 py-2 px-4 text-black font-bold">
      How it Works
    </Link>
    {authUser?
    <div className="flex">
      <Link  to="/create"
        className="flex text-[18px] items-center hover:text-gray-400 py-2 px-4 text-black font-bold">
        Sell a Car
      </Link>
      <Link  to="/profile"
        className="flex text-[18px] items-center bg-black
        hover:bg-white hover:border-black-2 hover:text-black mx-4 px-4 text-white font-bold rounded-full">
        Profile
      </Link>
    </div>
    :
      <Link  to="/profile"
        className="flex text-[18px] items-center bg-black
        hover:bg-white hover:border-black-2 mx-4 px-4 text-white font-bold rounded-full">
        Sign In
      </Link>}
  </div>
</nav>
```

Slika 4.8. Komponenta *Navbar.tsx*

Klikom na *Sign In* gumb, korisnik je prebačen na stranicu gdje može unijeti svoju e-mail adresu i lozinku. Ako nije registriran, postoji i poveznica koji ga preusmjerava na stranicu za registraciju. Sva programska logika odgovorna za registraciju i prijavu korisnika nalazi se unutar komponenti

SignIn.tsx i *SignUp.tsx*. *How it Works* poveznica vodi na stranicu koja sadrži dodatne informacije i upute o načinu rada aukcije. Komponenta za prijavu korisnika prikazana je na slici 4.9.

```
<div className="grid place-items-center my-20">
  <form onSubmit={signIn} className="flex flex-col items-center">
    <h1 className="text-[25px] text-center font-bold mb-10">Sign In To Your Ascari Account</h1>
    <div>
      <label className="text-black-100 text-[18px] mb-1 font-semibold">E-mail address</label>
      <input
        type="email" required
        className="block bg-white border border-black-100 text-black-100 rounded-lg w-80 px-2 py-2 mb-4"
        placeholder="Enter your e-mail"
        value={email}
        onChange={(e) => setEmail(e.target.value)}>
      </input>
    </div>
    <div>
      <label className="text-black-100 text-[18px] block mb-1 font-semibold">Password</label>
      <input
        type="password" required
        className="bg-white border border-black-100 text-black-100 rounded-lg w-80 px-2 py-2"
        placeholder="Enter your password"
        value={password}
        onChange={(e) => setPassword(e.target.value)}>
      </input>
      <p className="mt-6 text-gray-600">New to Ascari?</p>
      <Link to="/signup" class="text-gray-400">Create an Account</Link>
      <div class="flex justify-center">
        <button
          type="submit"
          class="bg-black-100 text-white rounded-full w-fit px-4 py-2 mt-2">
          Continue
        </button>
      </div>
    </div>
  </form>
</div>
```

Slika 4.9. Komponenta *SignIn.tsx*

Prilikom registracije korisnik ima mogućnost izabrati korisničko ime (*username*). Također postoji veza koja vodi na *SignIn.tsx* komponentu u slučaju da je korisnik već registriran. Komponenta za registraciju korisnika prikazana je na slici 4.10.

```
<div className="grid place-items-center my-20">
  <form onSubmit={signUp} className="flex flex-col items-center">
    <h1 className="text-[25px] text-center font-bold mb-10">Create Your Ascari Account</h1>
    <div>
      <label className="text-black-100 text-[18px] mb-1">Username</label>
      <input
        type="username" required
        className="block bg-white border border-black-100 text-black-100 rounded-lg w-80 px-2 py-2 mb-4"
        placeholder="Choose a username"
        value={username}
        onChange={(e) => setUsername(e.target.value)}>
      </input>
    </div>
    <div>
      <label className="text-black-100 text-[18px] mb-1">E-mail address</label>
      <input
        type="email" required
        className="block bg-white border border-black-100 text-black-100 rounded-lg w-80 px-2 py-2 mb-4"
        placeholder="Enter your e-mail"
        value={email}
        onChange={(e) => setEmail(e.target.value)}>
      </input>
    </div>
    <div>
      <label className="text-black-100 text-[18px] block mb-1">Password</label>
      <input
        type="password" required
        className="bg-white border border-black-100 text-black-100 rounded-lg w-80 px-2 py-2"
        placeholder="Enter your password"
        value={password}
        onChange={(e) => setPassword(e.target.value)}>
      </input>
      <p className="mt-6 text-gray-600">Already registered?</p>
      <Link to="/signin" className=" text-gray-400">Sign in to my account</Link>
      <div className="flex justify-center">
        <button
          type="submit"
          className=" bg-black-100 text-white rounded-full w-fit px-4 py-2 mt-6">
          Continue
        </button>
      </div>
    </div>
  </form>
</div>
```

Slika 4.10. Komponenta *SignUp.tsx*

Pretraga aukcija ostvarena je filtriranjem kolekcije aukcija dohvaćene iz *firebase firestore* baze podataka. Nakon ispunjavanja željenih kriterija, pritiskom na gumb *Search* poziva se metoda *handleSearch* koja sadrži potrebnu logiku filtriranja kolekcije aukcija. U slučaju da nema aukcija koje odgovaraju kriterijima pretrage, aplikacija prikazuje sve aukcije. Metoda za pretragu aukcija prikazana je na slici 4.11.

```
const handleSearch = () => {
  setFilteredCars(cars.filter((car) => {
    return (
      car.make?.toLowerCase().includes(searchMake.toLowerCase()) &&
      car.model?.toLowerCase().includes(searchModel.toLowerCase()) &&
      car.year?.toString().includes(filterYear.toLowerCase()) &&
      car.transmission?.toLowerCase().includes(filterTransmission.toLowerCase())
      )
  }))
}
```

Slika 4.11. Metoda *handleSearch* unutar komponente *Search.tsx*

Sortiranje rezultata pretrage implementirano je na način da se uspoređuju vrijednosti dva elementa. Točan kriterij sortiranja moguće je odabrati proizvoljno pomoću izbornika. Izbornik je ostvaren korištenjem *select* elementa prikazanog na slici 4.12.

```
<select
  className='bg-white rounded-full border-2 border-black px-4 py-2'
  defaultValue={''}
  onChange={(e) => setSort(e.target.value)}>
  <option value="">Sort By</option>
  <option value="endingSoon">Ending soon</option>
  <option value="newest">Newest</option>
  <option value="year">Year</option>
  <option value="mileage">Mileage</option>
</select>
```

Slika 4.12. *Select* element s nizom opcija (*option*) za sortiranje

Odabir kriterija sortiranja sprema se unutar *sort* stanja (*state*) te se na temelju njega vrši sortiranje ugrađenom *TypeScript* metodom *sort* koja je prikazana na slici 4.13.

```
<div className='px-6 home__cars-wrapper'>
  {filteredCars.length ?
    (filteredCars?.sort((a,b)=>{
      if(sort === 'endingSoon') {
        return a.timeRemaining - b.timeRemaining
      } else if (sort === 'newest'){
        return b.timeRemaining - a.timeRemaining
      } else if (sort === 'year'){
        return b.year - a.year
      } else if (sort === 'mileage'){
        return a.mileage - b.mileage
      } else return 0
    }).map((car) => (
      <div key={car.id} >
        <CardDetails key={car} car={car}/>
      </div>
    ))): (
      <div className='w-screen pr-10'>
        <Card/>
      </div>
  )}
</div>
```

Slika 4.13. Dio koda odgovoran za sortiranje

Komponenta *Card.tsx* koristi se za dohvaćanje i prikazivanje aukcija u kartičnom obliku. Svaka kartica generirana je pozivom komponente *CardDetails.tsx* koja sadrži sve informacije vezane uz aukciju poput fotografije, lokacije, trenutne cijene, preostalog vremena te drugih detalja. Fotografije automobila dohvaćene su pomoću API-ja. Logika korištenja API-ja sadržana je unutar metode generateCarImageUrl koja je prikazana na slici 4.14. Funkcionira na način da se na dobiveni URL dodaju paramteri poput marke, modela i godine proizvodnje automobila. Također postoji parametar za kut (*angle*) koji omogućuje dobivanje fotografije automobila iz različitog kuta gledanja.

```
import { CarType } from "../types/CarType"

export const generateCarImageUrl = (car: CarType, angle?: string) => {
    const url = new URL("https://cdn.imagin.studio/getimage");
    const { make, model, year } = car;

    url.searchParams.append('customer', 'hrdenis-mrdalj');
    url.searchParams.append('make', make);
    url.searchParams.append('modelFamily', model.split(" ")[0]);
    url.searchParams.append('zoomType', 'fullscreen');
    url.searchParams.append('modelYear', `${year}`);
    url.searchParams.append('angle', `${angle}`);

    return `${url}`;
}
```

Slika 4.14. Metoda za dohvaćanje fotografija s API-ja uređivanjem URL-a

Brojač vremena implementiran je pomoću ugrađene *React* metode *setInterval*. Preostalo vrijeme računa se računanjem razlike između trenutnog vremena i završnog vremena koje je definirano i uneseno u bazu podataka prilikom stvaranja aukcije. Zahvaljujući takvom načinu računanja, brojač radi u stvarnom vremenu što znači da gašenje aplikacije ne utječe na brojač. Ako je brojač došao do nule, prikazuje se poruka *Auction ended*. Logika brojača vremena apstrahirana je unutar prilagođene *hook* metode *useCountdown* što znači da se može koristiti u svakom dijelu aplikacije jednostavnim pozivom. Metoda je prikazana na slici 4.15.

```
export const useCountdown = (endTime: number, callback: () => void, interval = 1000) => {
    const starter = Date.now()
    const [time, setTime] = useState(endTime)

    useEffect(()=> {
        const customInterval = setInterval(()=> {
            if(time>0) {
                setTime((prev) => prev - interval)
            }
        }, interval)

        if(time===0) callback()
        return () => clearInterval(customInterval)

    },[time])

    return time
}
```

Slika 4.15.Prilagođena *hook* metoda unutar *useCountdown.tsx* komponente

Pomakom pokazivača miša na karticu aukcije pojavljuje se gumb *Go to Auction* koji usmjerava korisnika na novu stranicu aplikacije koja se sastoji od komponenata *Auction.tsx* i *AuctionDetails.tsx*. Unutar *Auction.tsx* komponente nalazi se logika odgovorna za dohvaćanje (*fetch*) podataka iz *firestore* baze podataka i prosljeđivanje dohvaćenih podataka komponenti *AuctionDetails.tsx* u obliku objekta. Komponenta aukcije prikazana je na slici 4.16.

```
useEffect(() => {
    const fetchCarData = async () =>{
        setIsLoading(true)
        const docSnap = await getDoc(getCar)
        if(docSnap.exists()){
            const newCarObj = {
                id: docSnap.id,
                ...docSnap.data()
            }
            setCar(newCarObj)
            setIsLoading(false)
        } else {
            console.log("No document")
        }
    }
    fetchCarData()
},[])
if(isLoading) return <div className='text-center'>Loading</div>


return (
    <div className=''>
        {Object.keys(car) && Object.keys(car).length ? (
            <AuctionDetails car={car}/>
        ): null}
    </div>
)
```

Slika 4.16. Komponenta *Auction.tsx*

Komponenta *AuctionDetails.tsx* prima kao parametar objekt prilagođenog tipa *CarType* koji je definiran unutar datoteke *types*. Svrha ove komponente je prikazati sve informacije i detalje o aukciji i ponuditi opciju licitiranja prijavljenim korisnicima. Sve informacije vezane za aukciju prikazuju se pomoću objekta koji je primljen kao parametar. Opcija licitiranja je skrivena za neprijavljene korisnike i zamijenjena gumbom *Sign in to Bid* koji preusmjerava korisnika na stranicu za registraciju i prijavu. Ovo je također ostvareno provjerom stanja *authUser* koje nam govori stanje autentifikacije korisnika. U slučaju da je korisnik prijavljen, pojavljuje se gumb *Place Bid* koji otvara obrazac u kojem korisnik upisuje željenu svotu te gumb *Confirm* za predaju obrazca. Nakon predaje obrazca, baza podataka se ažurira s novom svotom. Također se provjerava je li trenutno prijavljen korisnik autor aukcije.U slučaju da je, prikazuje se poruka „*The seller*

cannot place bids“. Kada aukcija završi, prikazuje se poruka u kojoj piše korisničko ime korisnika s najvišom ponudom. Logika provjere stanja autentifikacije korisnika prikazana je na slici 4.17.

```
{authUser && auth.currentUser?.displayName!==car.sellerUsername ? (
<div className='flex-col items-center'>
    {time>0?(
    <button className="bg-black text-white py-2 px-10 h-10 mb-4 flex items-center rounded-full"
        onClick={handleClick}>
            Place Bid
    </button>) : (
        <p className='text-[18px]'>
            Auction has ended, won by <span className='font-bold'>{car.bidder}</span> for <span className='font-bold text-green-900'>${car.bid}</span>
            </p>
    )
    )
    {show ? (
        <div className="flex-col items-center'>
            <div className="flex items-center'>
                <p className="pl-2 pr-2 text-[20px] text-green-800 font-bold">${car.bid}</p>
                <div className="flex items-center'>
                    <p className="block items-center pl-2 pr-2 text-[20px] font-bold">+</p>
                    <input
                        type="bid" required
                        className="text-center bg-gray-100 border-b border-black-100 text-black-100 w-20 ml-2 px-2"
                        placeholder="Enter your bid"
                        value={bid}
                        onChange={(e) => setBid(parseInt(e.target.value))}>
                    </input>
                    <p className="block items-center pl-2 pr-2 text-[20px] font-bold">=</p>
                    <p className="block items-center pl-2 pr-2 text-[20px] font-bold">${car.bid+bid}</p>
                    <button onClick={placeBid} className="ml-6 bg-black text-white py-2 px-10 rounded-full">
                        Confirm
                    </button>
                </div>
            </div>
        </div>
    ) : (
        <div></div>
    )}
</div>
) : (
    <div>
        <p className="mb-2">The seller cannot place bids</p>
        <Link  to="/profile"
        className="flex items-center bg-black hover:bg-gray-400 w-fit py-2 px-6 text-white rounded-full">
        Sign in to Bid
        </Link>
    </div>
    )
]
```

Slika 4.17. Logika skrivanja gumba za licitiranje ovisno o stanju autentifikacije korisnika

Metoda *placeBid* poziva se klikom na gumb *Confirm*. Ova metoda odgovorna je za računanje nove svote aukcije i ažuriranje dokumenta unutar firestore baze podataka metodom *updateDoc*. Metoda je prikazana na slici 4.18.

```
const placeBid = () => {
    newBid = bid+ car.bid!
    highBidder = auth.currentUser?.displayName
    updateDoc(doc(firestore, 'cars', car.id), {bid: newBid, bidder: highBidder})

    return (
        bids.push(newBid),
        bidders.push(highBidder!))
}
```

Slika 4.18. Metoda *placeBid*

Kada se korisnik uspješno prijavi, navigacijska traka nudi opciju *Sell a Car* koja nudi mogućnost postavljanja nove aukcije. Klikom na gumb korisnik je preusmjeren na stranicu za ispunjavanje obrazca aukcije. Polja obrazca uključuju marku i model automobila, prijeđene kilometre, opis automobila i slično. Korisnik također ima mogućnost postaviti početnu cijenu aukcije od koje želi započeti licitaciju. Uz to može odrediti i vrijeme trajanja svoje aukcije. Većinu polja korisnik popunjava sam, međutim postoji nekoliko atributa aukcije koji se generiraju u pozadini kao što su datum početka i završetka aukcije, postavljanje kontakt informacija o korisniku u bazu podataka i slično. Klikom na gumb *Continue*, podaci se dodaju u *firestore* bazu podataka, a korisnik je preusmjeren na početnu stranicu gdje može vidjeti svoju aukciju. Metoda za dodavanje nove aukcije prikazana je na slici 4.19.

```
const addNewAuction = (e: React.FormEvent<HTMLFormElement>) => {
    e.preventDefault()
    const bidder = ''
    const startDate = Date.now()
    const endDate = startDate+(auctionTime)
    const timeRemaining = endDate - startDate
    const user = auth.currentUser
    if(user !== null) {
      sellerUsername=user.displayName!
      sellerEmail=user.email!}
    addAuction({
      year,
      make,
      model,
      bodyStyle,
      drive,
      mileage,
      engine,
      transmission,
      fuel,
      vin,
      location,
      description,
      bid,
      bidder,
      sellerUsername,
      sellerEmail,
      startDate,
      endDate,
      timeRemaining
    })
    console.log("successfuly added")
    navigate("/")
}
```

Slika 4.19. Metoda *addNewAuction*

Uz navigacijsku traku, svaka stranica aplikacije sadrži i podnožje (footer) s nekoliko poveznica koje sadrže dodatne informacije o uvjetima korištenja, odluku o privatnosti i slično. Glavni HTML element koji je korišten za stvaranje podnožja je *footer*. Komponenta podnožja prikazana je na slici 4.20.

```
<div className="flex justify-between
items-center flex-wrap mt-10 border-t
border-gray-100 sm:px-16 px-6 py-10">
  <p>@2023 Ascari Auctions. All Rights Reserved</p>
  <div
  className="footer__copyrights-link">
    <Link to="/"
    className="text-gray-500">
      Privacy Policy
    </Link>
    <Link to="/"
    className="text-gray-500">
      Terms of Use
    </Link>
  </div>
</div>
```

Slika 4.20. Komponenta *Footer.tsx*

4.2. Backend

Kao što je već rečeno, pojam backend odnosi se na dio koda aplikacije koji se odvija u pozadini. Iako korisnik nije svijestan što se događa u pozadini, backend je neophodan za svaku složenu aplikaciju. Aspekti backend-a su obično baza podataka, poslužitelj, poslovna pravila, sigurnosni protokoli, upravljanje korisnicima i slično.

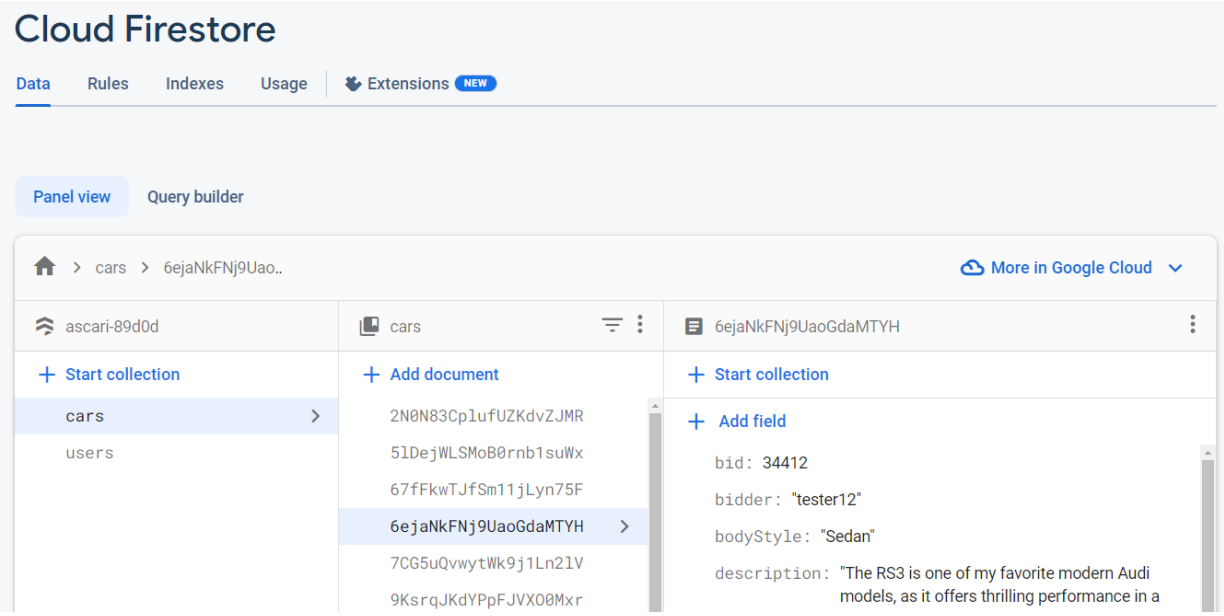
Za potrebe ove aplikacije korištena je gotova platforma *firebase* koja znatno olakšava implementaciju *backend-a*. *Firebase* nudi razne usluge, a za ovu aplikaciju su korišteni *firestore* baza podataka i *authentication* sustav autentifikacije korisnika. U početku je potrebno stvoriti novi *firebase* projekt i povezati ga s aplikacijom. To se ostvaruje pomoću naredbe *firebase login* koja se upisuje u terminal i služi za prijavu korisnika *firebase* računom. Nakon toga, upisuje se naredba *firebase init* kojom se započinje povezivanje projekta. *Firebase* zatim nudi upitnik s nekoliko opcija koje se mogu proizvoljno odabrati poput željenih usluga, naziva projekta i slično. Po završetku izvođenja naredbe *firebase init* stvara se konfiguracijska datoteka *firebase.ts* u kojoj se

nalaze svi potrebni ključevi za uspješno povezivanje projekta s aplikacijom. Datoteka je prikazana na slici 4.21.

```
const firebaseConfig = {
    apiKey: "AIzaSyAaptRDf_-eS1oqt3cY6Pt8LdnaKem10T0",
    authDomain: "ascari-89d0d.firebaseapp.com",
    projectId: "ascari-89d0d",
    storageBucket: "ascari-89d0d.appspot.com",
    messagingSenderId: "983467910478",
    appId: "1:983467910478:web:3d9435b42a960840a6ea1f",
    measurementId: "G-XBXTFSMP3F"
  };

  // Initialize Firebase
  export const app = initializeApp(firebaseConfig);
  export const auth = getAuth(app)
```

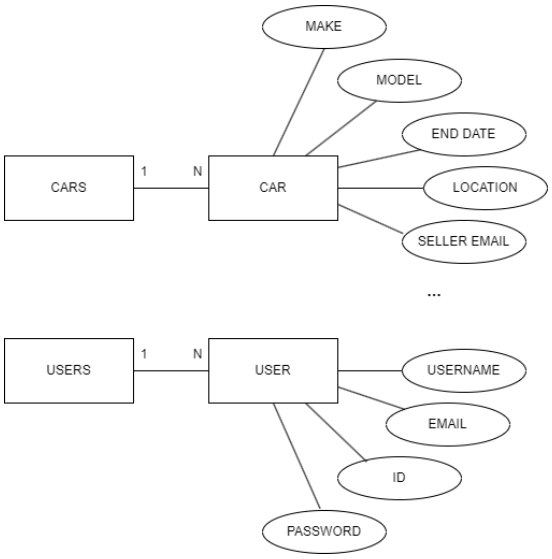
Slika 4.21. Komponenta *firebase.ts*

Nakon povezivanja, potrebno je stvoriti kolekciju (*collection*) unutar *Cloud firestore* usluge. Za ovu aplikaciju potrebne su dvije kolekcije. *Cars* kolekcija sadrži sve aukcije, a *users* kolekcija sadrži registrirane korisnike. Struktura baze podataka prikazana je na slici 4.22.



Slika 4.22. Struktura baze podataka.

ER (*Entity-relationship*) dijagram baze podataka prikazan je na slici 4.23.



Slika 4.23. ER dijagram baze podataka.

Svaka aukcija spremljena je unutar posebnog dokumenta (*document*) s jedinstvenim identifikatorima. Komponenta *controller.ts* sadrži metode za stvaranje potrebnih referenci kolekcija i dvije metode *addAuction* i *addUser* za dodavanje novih dokumenata u bazu podataka. Komponenta je prikazana na slici 4.24.

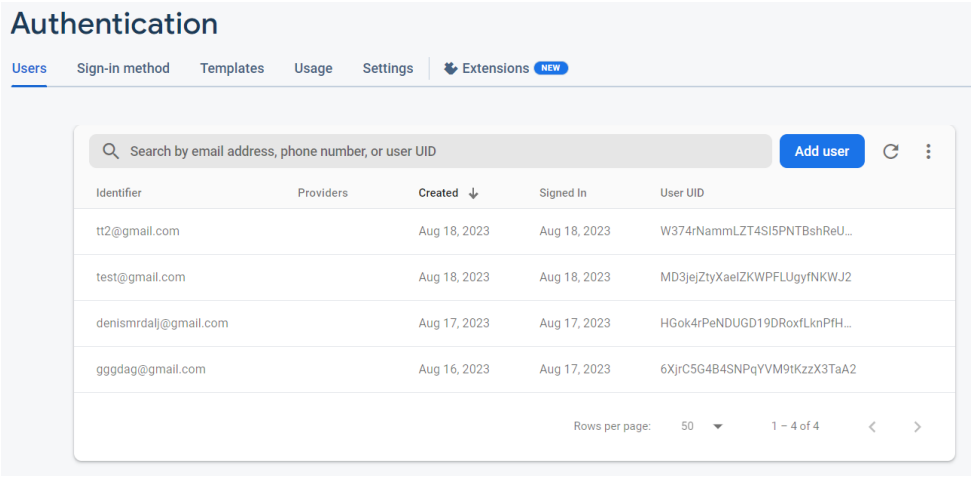
```
export const firestore = getFirestore(app)

export const carsCollection = collection(firestore, 'cars')
export const usersCollection = collection(firestore, 'users')

export const addAuction = async(auctionData: AddAuctionType) => {
    const newAuction = await addDoc(carsCollection, {...auctionData})
}

export const addUser = async(userData: AddUserType) => {
    const newUser = await addDoc(usersCollection, {...userData})
}
```

Slika 4.24. Dohvaćanje stvorenih kolekcija i dodavanje dokumenata u bazu podataka

Prilikom stvaranja nove aukcije prikupljaju se podaci uneseni od strane korisnika i stvara se novi dokument u bazi podataka pomoću ugrađene metode *addDoc*. Metoda addUser poziva se prilikom registracije korisnika. Iako *firebase* sadrži poseban odjeljak za autentifikaciju, korisno je imati i vlastitu kolekciju korisnika kako bi se mogli dodavati razni atributi.

Autentifikacija korisnika odvija se pomoću firebase/auth biblioteke koja sadrži razne gotove metode za registraciju i prijavu korisnika poput *createUserWithEmailAndPassword* i *signInWithEmailAndPassword*. *Firebase* nudi niz načina autentifikacije korisnika. Osim standardne autentifikacije e-mail adresom i lozinkom, moguća je i autentifikacija putem *Google-*

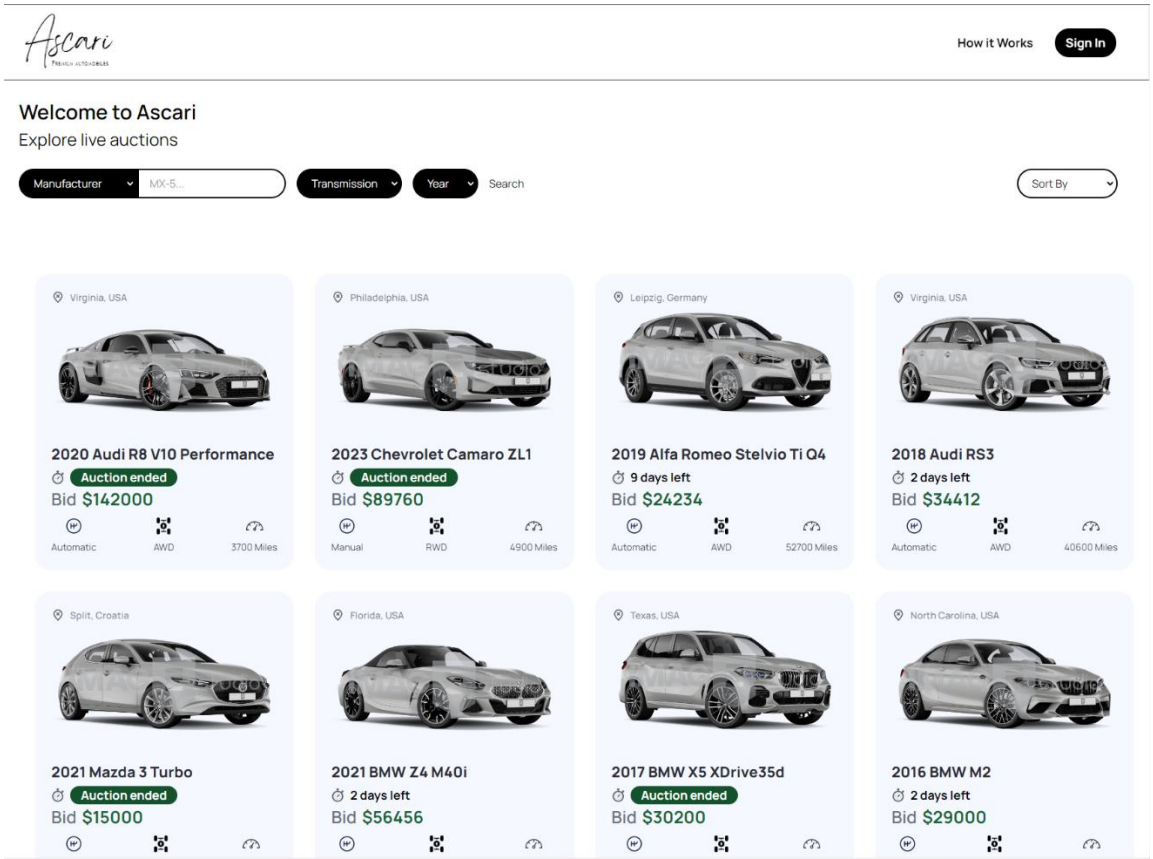
a te raznih društvenih mreža. Za potrebe ove aplikacije dovoljna je autentifikacija e-mail adresom i lozinkom. Registrirani korisnici vidljivi su unutar *Authentication* dijela *firebase-a* na slici 4.25.



Slika 4.25. Popis svih registriranih korisnika

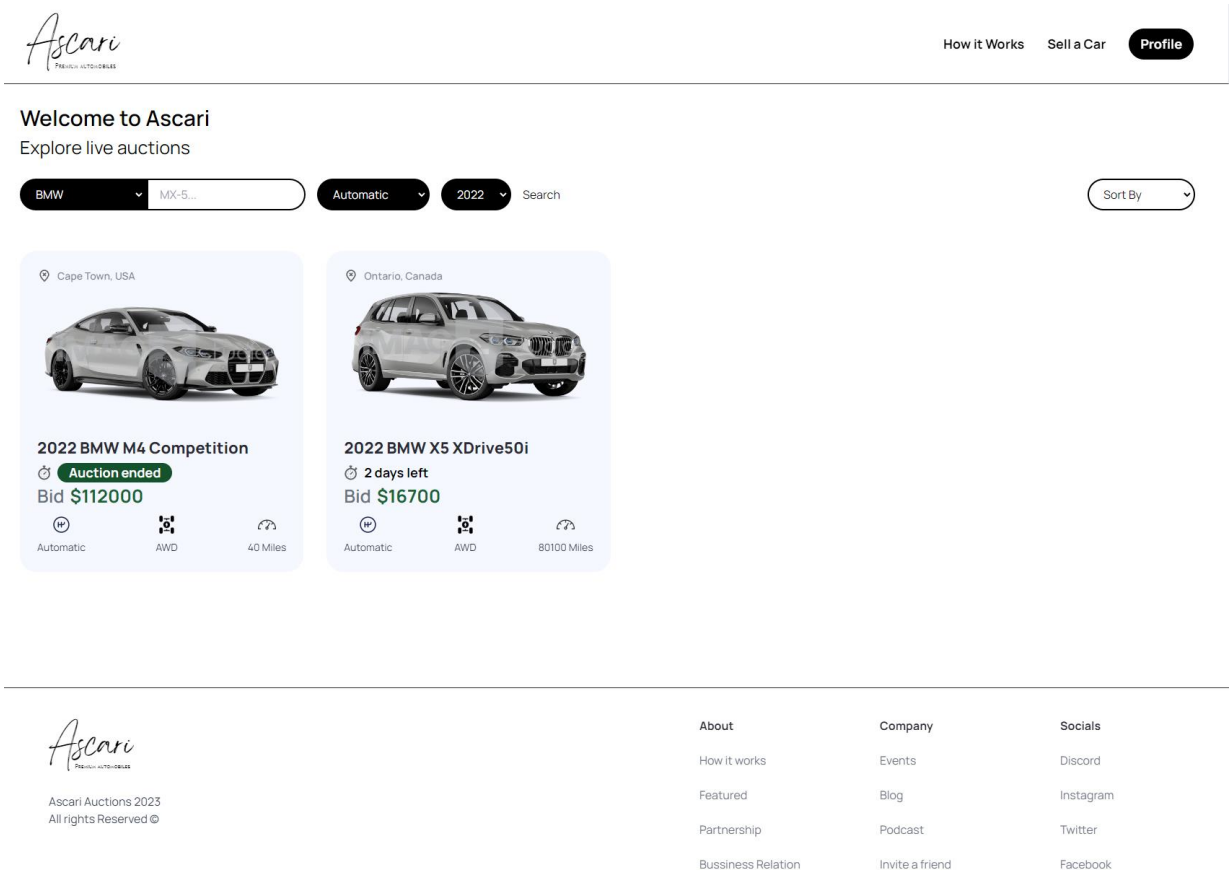
5. OPIS RADA APLIKACIJE

Pokretanjem aplikacije pojavljuje se početna stranica prikazana slikom 5.1. Početna stranica sadrži sve aukcije, opcije filtriranja i sortiranja te navigacijsku traku.



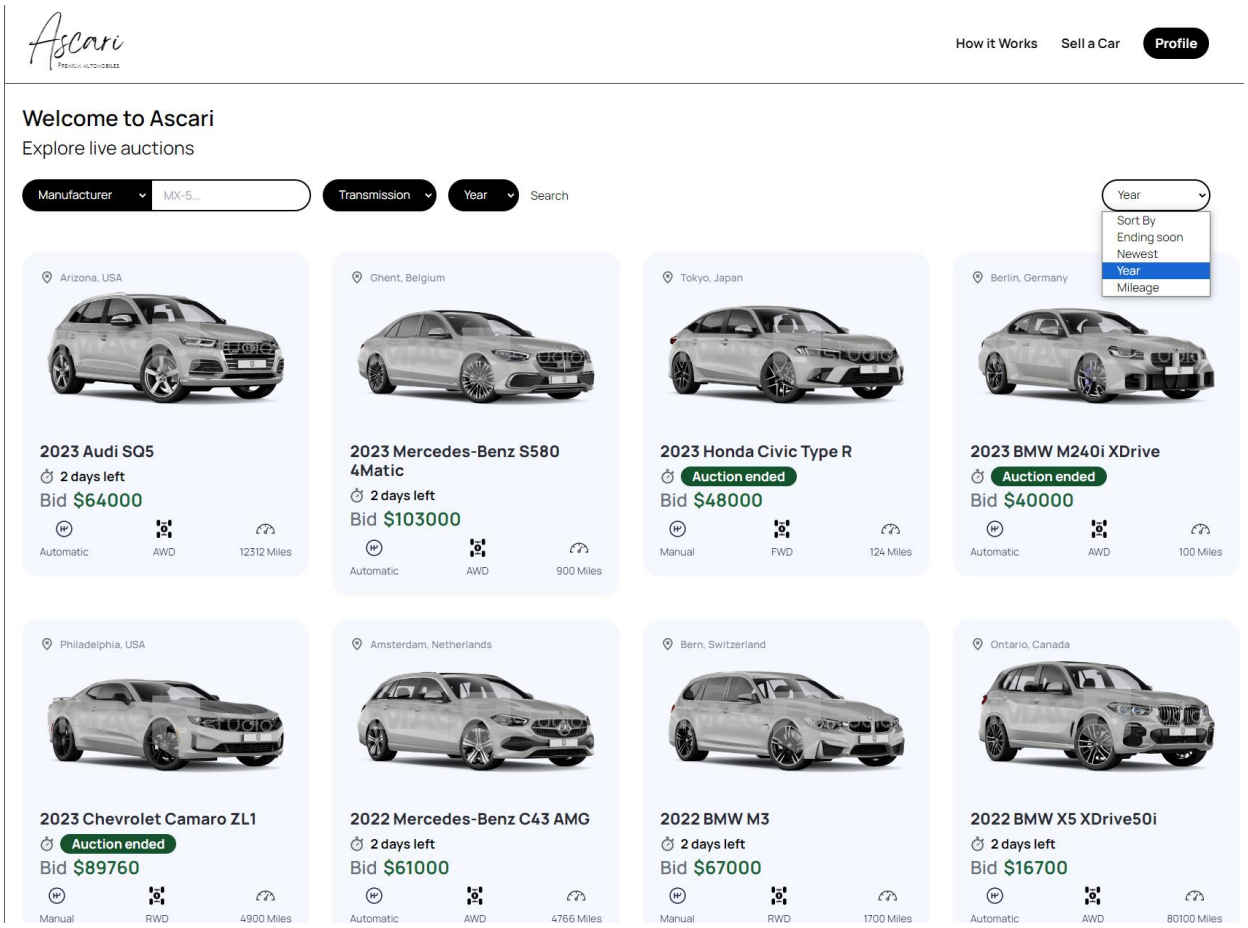
Slika 5.1. Početna stranica neprijavljenog korisnika

Slika 5.2. demonstrira funkciju filtriranja vozila. Moguće je filtrirati aukcije prema marki i modelu vozila, tipu mjenjača te godini proizvodnje.



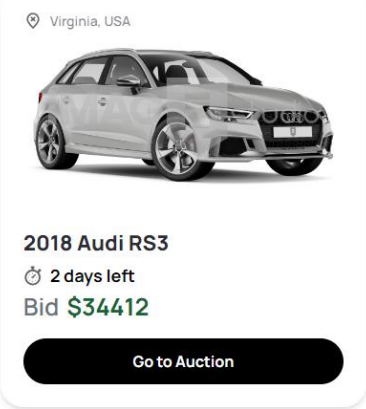
Slika 5.2. Pretraga vozila marke BMW s automatskim mjenjačem godine proizvodnje 2022.

Sortiranje dobivenih rezultata može se odrediti klikom na izbornik *Sort By*. Postoji nekoliko izbora, a to su završava uskoro (*Ending soon*), najnovije (*Newest*), godina proizvodnje (*Year*) te kilometraža (*Mileage*). Sortiranje je prikazano na slici 5.3.

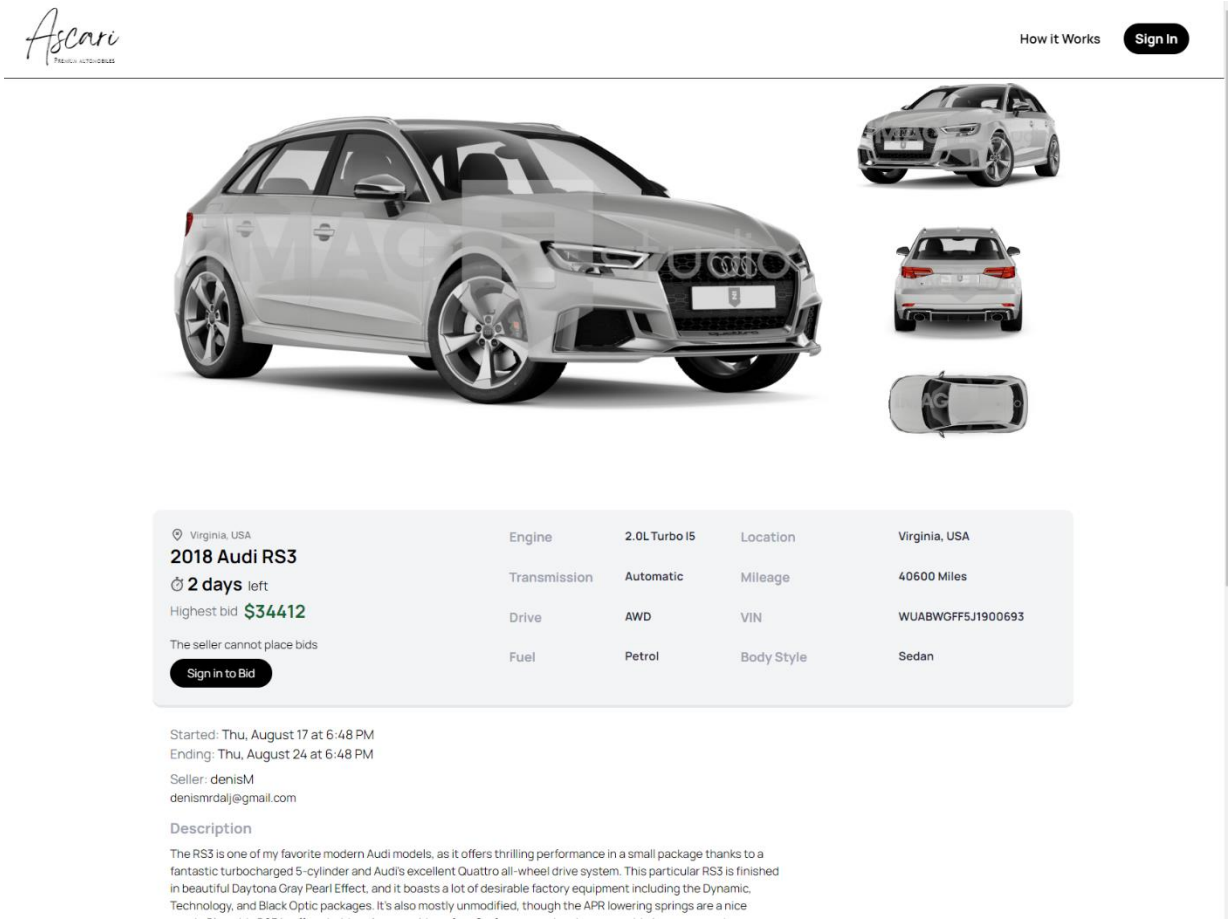


Slika 5.3. Sortiranje aukcija prema godini proizvodnje.

Aukcije su prikazane kao kartice na kojima se nalaze informacije o lokaciji vozila, marki i modelu, vremenu isteka, najvišoj ponudi te dodatne informacije o tipu mjenjača, pogona i pređenih milja. Pomicanjem pokazivača miša na željenu karticu pojavljuje se gumb koji preusmjerava korisnika na stranicu same aukcije. Izgled kartice prikazan je na slici 5.4.

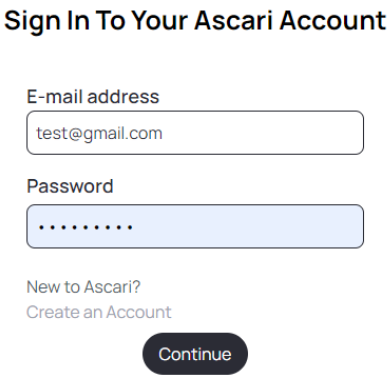


Slika 5.4. Izgled kartice s gumbom *Go to Auction*.



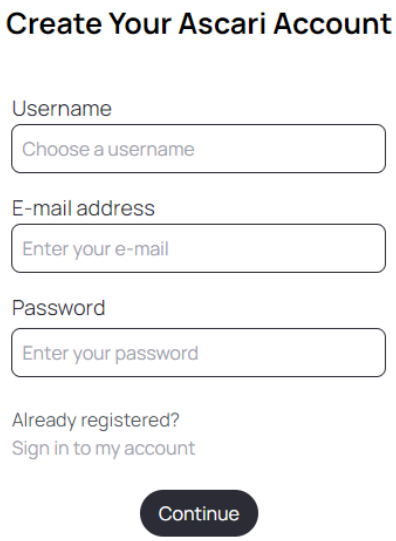
Slika 5.5. Izgled stranice aukcije.

Svaka stranica aukcije sadrži galeriju fotografija generiranih pomoću *Imagin Studio* API-ja, tablicu specifikacija vozila, informacije o datumu početka i isteka aukcije, najviša ponuda te opis vozila. Izgled stranice aukcije prikazan je na slici 5.5. Za nerpijavljene korisnike pojavljuje se gumb *Sign in to Bid* i poruka s upozorenjem da autor aukcije ne može sudjelovati u licitaciji. Klikom na gumb pojavljuje se obrazac za prijavu korisnika. Izgled obrazca za prijavu korisnika prikazan je na slici 5.6.



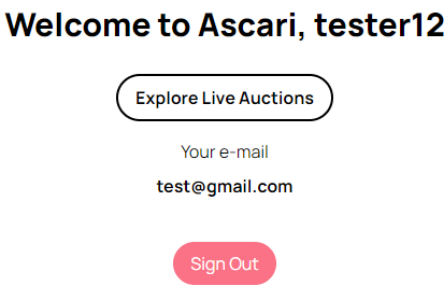
Slika 5.6. Izgled obrazca za prijavu.

Klikom na poveznicu Create an Account, korisnik je preusmjeren na obrazac za registraciju koji je prikazan na slici 5.7.



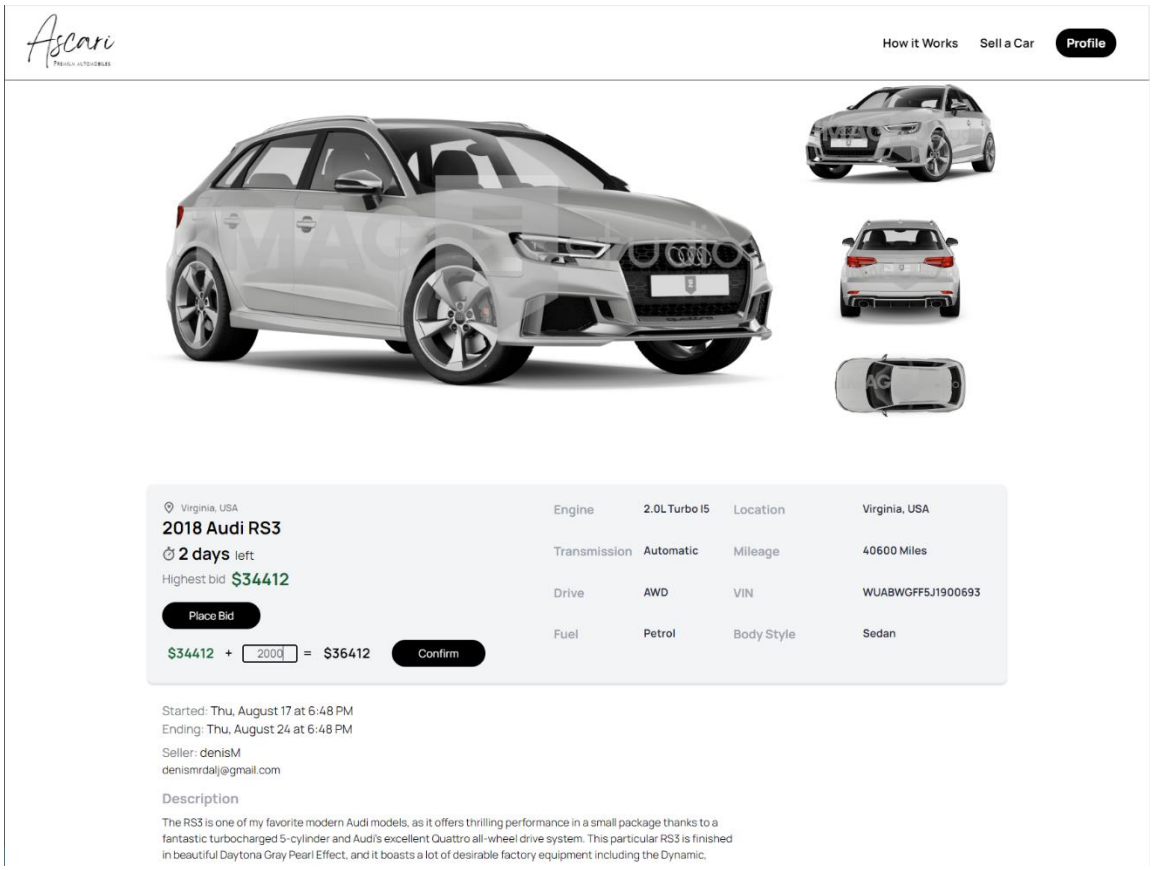
Slika 5.7. Izgled obrazca za registraciju.

Klikom na gumb Continue korisnik je prijavljen i preusmjeren na *Profile* stranicu. Na stranici profila prikazano je korisničko ime i e-mail adresa. Gumb *Explore Live Auctions* vodi korisnika na početnu stranicu aplikacije. Postoji i gumb *Sign Out* za odjavu korisnika. Izgled stranice profila prikazan je na slici 5.8.



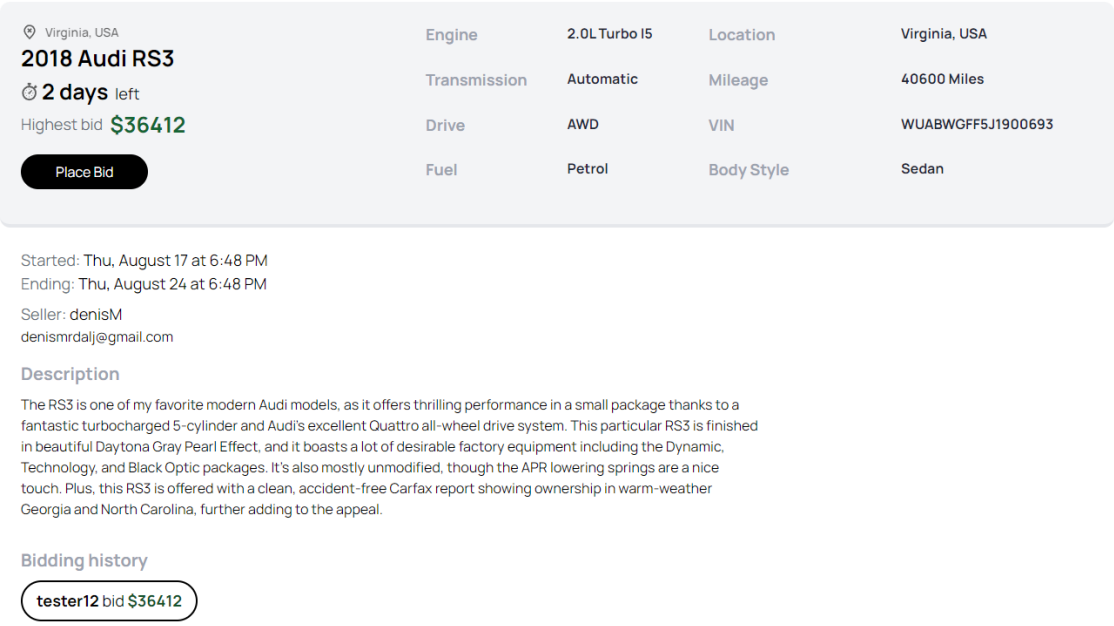
Slika 5.8. Izgled stranice profila.

Nakon što se korisnik uspješno prijavio, navigacijska traka se promijenila te sada sadrži mogućnost postavljanja aukcije (*Sell a Car*) i gumb *Profile* koji vodi na stranicu profila. Stranica aukcije sada nudi mogućnost licitacije. Klikom na gumb *Place Bid* otvara se kratak obrazac za unos željene svote. Gumb *Confirm* sprema novu svotu i ažurira sučelje. Izgled stranice aukcije iz perspektive prijavljenog korisnika prikazan je na slici 5.9.



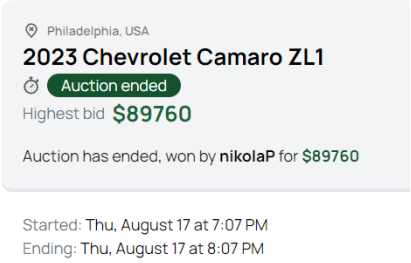
Slika 5.9. Izgled stranice aukcije i navigacijske trake za prijavljenog korisnika.

Nakon što je korisnik poslao svoju ponudu, njegovo korisničko ime prikazano je na dnu stranice zajedno s trenutno najvišom ponudom koja je vidljiva na slici 5.10.



Slika 5.10. Izgled stranice aukcije.

Nakon što aukcija završi, opcija licitiranja nestaje te se prikazuje poruka s korisničkim imenom pobjednika aukcije. Aukcija je još uvijek vidljiva na početnoj stranici aplikacije i ne briše se iz baze podataka. Završena aukcija vidljiva je na slici 5.11.



Slika 5.11. Izgled završene aukcije.

Prijavljeni korisnici također imaju pristup gumbu *Sell a Car* na navigacijskoj traci. Klikom na gumb, otvara se obrazac koji je potrebno ispuniti kako bi se aukcija dodala u bazu podataka. Korisnik ispunjava osnovne informacije o svom automobilu. Obrazac kombinira padajuće izbornike i polja za upisivanje teksta. Korisnik ima mogućnost odabrati početnu cijenu aukcije te vrijeme trajanja aukcije. Padajući izbornik *Auction type* nudi opcije od 14 dana, 7 dana, 24 sata i 1 sat. Nakon što je svako polje popunjeno, pritiskom na gumb *Continue* aukcija je dodana u *firestore* bazu podataka u obliku novog dokumenta s jedinstvenim identifikatorom. Aukcija je odmah vidljiva na početnoj stranici. Obrazac za novu aukciju prikazan je na slici 5.12.

Slika 5.12. Izgled obrazca za novu aukciju.

6. ZAKLJUČAK

Cilj ovog završnog rada bio je napraviti web aplikaciju za aukcije automobila pomoću *React* okvira, *TypeScript* programskog jezika i *Firebase* baze podataka. Aplikacija korisnicima omogućuje pregled aktivnih i završenih aukcija, filtriranje i sortiranje rezultata, sudjelovanje u licitaciji te dodavanje vlastitih aukcija. Korisničko sučelje i iskustvo aplikacije pažljivo je osmišljeno kako bi apliakcija bila jednostavna, intuitivna i responzivna. Na sve više zasićenom tržištu online aukcija, istču se one aplikacije koje što više funkcionalnosti objedine u što jednostavnije sučelje. Aplikaciju je moguće proširiti na niz načina poput uključivanja funkcionalnosti plaćanja kreditnim karticama, omogućavanje korisniku da dodaje vlastite fotografije te dodavanje odjeljka za komentare ispod svake aukcije. Za potrebe ove aplikacije bilo je potrebno upoznati se s *React* okvirom, *TypeScript* programskim jezikom, osnovama HTML-a i CSS-a te *Tailwind* bibliotekom. Također, za potrebe baze podataka, autentifikacije korisnika te sigurnosnih protokola bilo je potrebno ukomponirati i *Firebase*.

LITERATURA

[1] Službena web stranica *eBay Motors* platforme dostupno na: https://www.ebay.com/b/Auto-Parts-and-Vehicles/6000/bn_1865334 [Datum zadnjeg pregleda: 20.kolovoz 2023.]

[2] Službena web stranica *Bring a Trailer* dostupno na: https://bringatrailer.com/ [Datum zadnjeg pregleda: 20.kolovoz 2023.]

[3] Službena web stranica *Cars & Bids* dostupno na: https://carsandbids.com/ [Datum zadnjeg pregleda: 20.kolovoz 2023.]

[4] C. Rippon, Learn React with TypeScript: A beginner's guide to reactive web development with React 18 and TypeScript, Packt Publishing, Birmingham 2023.

[5] Službena web stranica *React* dokumentacije dostupno na: https://legacy.reactjs.org/docs/getting-started.html#learn-react [Datum zadnjeg pregleda: 20.kolovoz 2023.]

[6] J. Duckett, HTML & CSS: Design and Build Web Sites, John Wiley & Sons, Indianapolis 2011.

[7] Službena web stranica *Tailwind* CSS dokumentacije dostupno na: https://console.firebase.google.com/u/0/ [Datum zadnjeg pregleda: 21.kolovoz 2023.]

[8] Službena web stranica *Firebase* dostupno na: https://console.firebase.google.com/u/0/ [Datum zadnjeg pregleda: 21.kolovoz 2023.]

[9] Službena web stranica dokumentacije *Visual Studio Code* dostupno na: https://code.visualstudio.com/docs [Datum zadnjeg pregleda: 21.kolovoz 2023.]

SAŽETAK

U ovom završnom radu bilo je potrebno napraviti web aplikaciju koja korisnicima omogućava pregled, dodavanje i sudjelovanje u aukcijama automobila. Aplikacija je namijenjena svim ljubiteljima automobila koji žele kupiti ili prodati svoj automobil na aukcijski način kroz moderno sučelje vrlo jednostavno za korištenje. Za izradu ove aplikacije korištene su tehnologije: React, TypeScript programski jezik, HTML i CSS, biblioteka *Tailwind* CSS te *Imagin Studio* API.

Ključne riječi: aukcija, React, TypeScript

ABSTRACT

Car auction application

This graduation thesis required disegning a web application which would allow users to browse, add and participate in car auctions. The application is intended for all car enthusiasts interested in buying or selling their car at an auction through a modern, very easy to use interface. This application was developed using React, TypeScript, HTML and CSS, the Tailwind CSS library and the Imagin Studio API.

Keywords: auction, React, TypeScript