Web portal zajednice ugađanja automobila

Strmečki, Stjepan

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Stručni studij računarstva

WEB PORTAL ZAJEDNICE UGAĐANJA AUTOMOBILA

Završni rad

Stjepan Strmečki

Osijek, 2024.

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

Obrazac Z1S: Obrazac za imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju

Osijek, 17.02.2024.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju

Ime i prezime Pristupnika:	Stjepan Strmečki
Studij, smjer:	Računarstvo
Mat. br. Pristupnika, godina upisa:	AR 4757, 19.07.2019.
OIB Pristupnika:	73885355067
Mentor:	Robert Šojo, mag. ing. comp.
Sumentor:	,
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Marina Peko, dipl. ing.
Član Povjerenstva 1:	Robert Šojo, mag. ing. comp.
Član Povjerenstva 2:	mr. sc. Željko Štanfel
Naslov završnog rada:	Web portal zajednice ugađanja automobila
Znanstvena grana završnog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada	Web portal omogućava vođenje, kreiranje i prikaz različitih članaka kao i natjecanja za zajednicu ugođenih automobila. Samo registrirani korisnik ima mogućnost postavljanja informacija o sebi i svome automobilu te ima mogućnost sudjelovanja u ocjenjivanju, komentiranju i prijavi za sudjelovanje u natjecanju za određenu kategoriju. Gost ima mogućnost samo uvid u prikaz različitih članaka i objavljene događaje. Web portal omogućava prikaz automobila za različite kategorije prikaza (vanjska kozmetika, kokpit, ugađanje performansi motora). Student treba razviti funkcionalnu aplikaciju koristeći različite web tehnologije. Rezervirano za studenta: Stjepan Strmečki
Prijedlog ocjene pismenog dijela ispita (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:	17.02.2024.

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, 27.02.2024.

Ime i prezime studenta:	Stjepan Strmečki
Studij:	Računarstvo
Mat. br. studenta, godina upisa:	AR 4757, 19.07.2019.
Turnitin podudaranje [%]:	8

Ovom izjavom izjavljujem da je rad pod nazivom**: Web portal zajednice ugađanja automobila**

izrađen pod vodstvom mentora Robert Šojo, mag. ing. comp.

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

Ovaj rad proizlazi iz potrebe da se olakša interakcija autoentuzijasta putem online objava i poboljšavanje iskustva organiziranja druženja i natjecanja. Druženja uživo postaju sve veća te organizacije istih postaju sve kompleksnije. Objavljivanje svojih automobila u raznim izdanjima sve su češće pojave na internetu, te publika za iste sve više raste. Stvaranje ove web aplikacije jest s ciljem spajanja korisnika u jednu zajednicu te poticanje posjetitelja na zbližavanje s drugim korisnicima kroz komentare ili događanja. U svrhu rješavanja ovih problema, ovaj se rad temelji na upotrebi niza tehnologija, uključujući Firebase bazu podataka, HTML, CSS, JavaScript React biblioteka. Kombinacijom ovih tehnologija omogućeno je stvaranje web portala koji pruža sveobuhvatno iskustvo korisnicima. Korištenje Firebase okoline te Visual Studio Code kao razvojne okoline daje efikasno razvijanje i održavanje projekta. Poglavlja su podijeljena u četiri ključna dijela. U drugom poglavlju se analiziraju postojeća rješenja i web stranica gdje se mogu pronaći pojedini oblici implementacije vezane za objave, organiziranje događaja ili čitanje novosti. Treće se poglavlje bavi detaljnim opisom korištenih tehnologija pri izradi web aplikacije. U četvrtom se poglavlju opisuju implementirane funkcionalnosti te potkrepljuju logikom korištenom pri izradi web aplikacije. Konačno, u petom poglavlju, analizira se korištenje web aplikacije, prolazeći kroz perspektive registriranog korisnika i posjetitelja.

1.1. Zadatak završnog rada

Kreiranje web portala koji okuplja ljubitelje automobila, automobilskih susreta te sveukupnog automobilskog svijeta. Na jednome mjestu prikazuje galeriju brojnih automobila te omogućava korisnicima organizaciju brojnih natjecanja i događanja diljem Hrvatske. Portal nudi korisnicima koji nisu registrirani pregled galerije prepune slika koje registrirani korisnici objavljuju te čitanje najnovijih vijesti iz auto-industrije. Registrirani korisnici imaju mogućnost objavljivanja fotografija, glasanje te komentiranje istih. Registrirani korisnici također imaju mogućnost organiziranja događanja poput stilističkih i utrka automobila. Organizatori događanja imaju uvid u korisničke prijave za događanja. Administratorska uloga ima mogućnost brisanja neprimjerenih komentara i objava te pregled u sva događanja na stranici te postavljanje novosti na početnoj stranici.

2. POSTOJEĆA RJEŠENJA

Postojeća rješenja koja imaju funkcionalnost kao određene mogućnosti na stranici su:

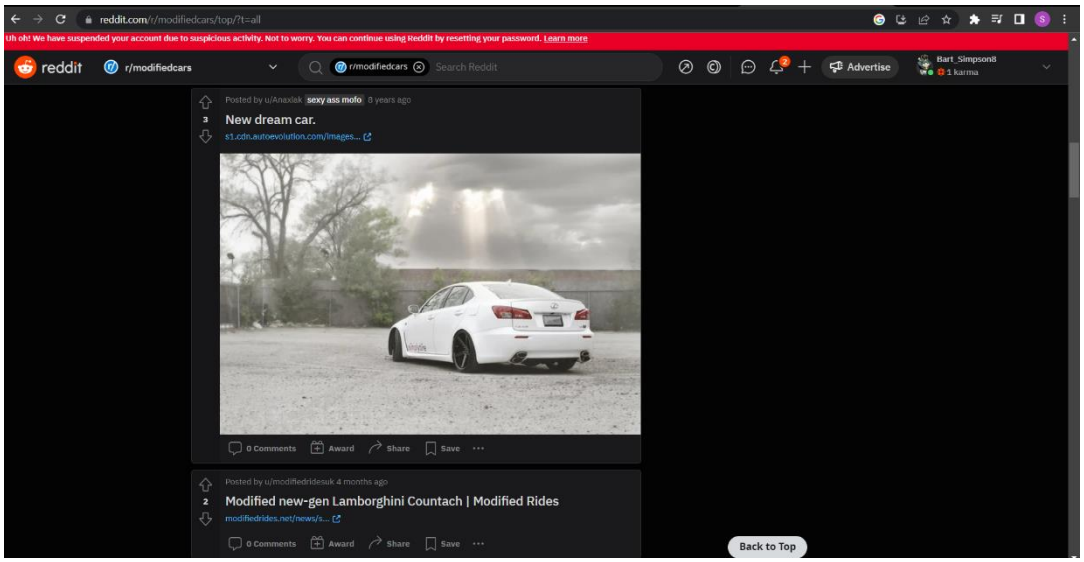
- Reddit
- Car 0Throttle
- Facebook - events

2.1. Reddit

Reddit je dinamičan internetski portal koji pruža prostor za dijeljenje i raspravu o različitim temama putem "subreddita", korisničkih komentara i glasanja. Reddit je organiziran u "subreddite", koji su podforumi posvećeni određenim temama. Postoje subredditi za različite interese kao što su tehnologija, vijesti, sport, umjetnost, znanost, auto-moto industrija i mnogi drugi. Korisnici mogu pregledavati subreddite, postavljati pitanja, dijeliti članke, slike, videozapise i druge vrste sadržaja, te komunicirati s drugim korisnicima putem komentara.

Jedna od ključnih značajki Reddita je sustav glasanja. Korisnici mogu glasati "gore" ili "dolje" za sadržaj koji se dijeli. Ovisno o broju glasova gore ili dolje, sadržaj dobiva veću ili manju vidljivost na portalu.

Subreddit vezan za modificiranje automobila prikazan je na slici 2.1. [1].

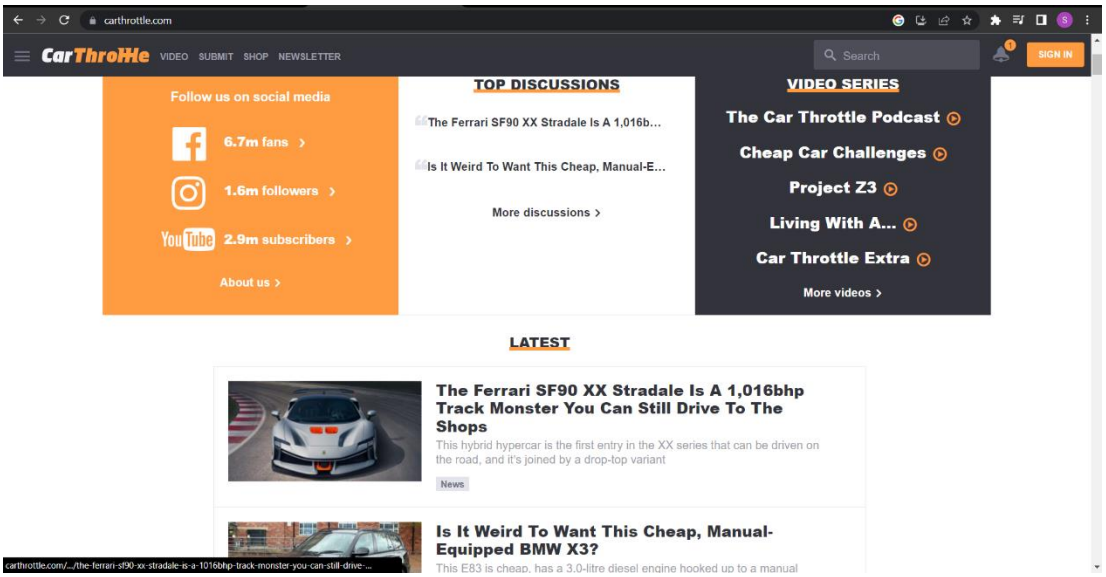


Slika 2.1. *Subreddit vezan za modificiranje automobila.*

2.2. Car Throttle

Car Throttle je popularan web portal posvećen automobilima i svijetu automobila. Osnovan je 2009. godine i brzo je stekao veliku bazu vjernih čitatelja i obožavatelja. Car Throttle pruža raznolik sadržaj koji uključuje vijesti, recenzije, članke, videozapise i druge vrste sadržaja vezane uz automobile. Portal se fokusira na sve elemente automobilske kulture. Jedna od ključnih značajki Car Throttlea je angažirana zajednica. Korisnici mogu stvarati svoje profile, sudjelovati u raspravama, komentirati sadržaj i dijeliti vlastite priče i projekte. Car Throttle također organizira razne natjecanja, događanja i tematske izazove za svoje članove.

Car Throttle se ističe svojim pristupom informiranju o automobilima s dozom humora i neformalnosti, privlačeći mlađu publiku koja dijeli strast prema automobilima. Slika 2.2. predstavlja početnu stranicu Car Throttle web portala [2].



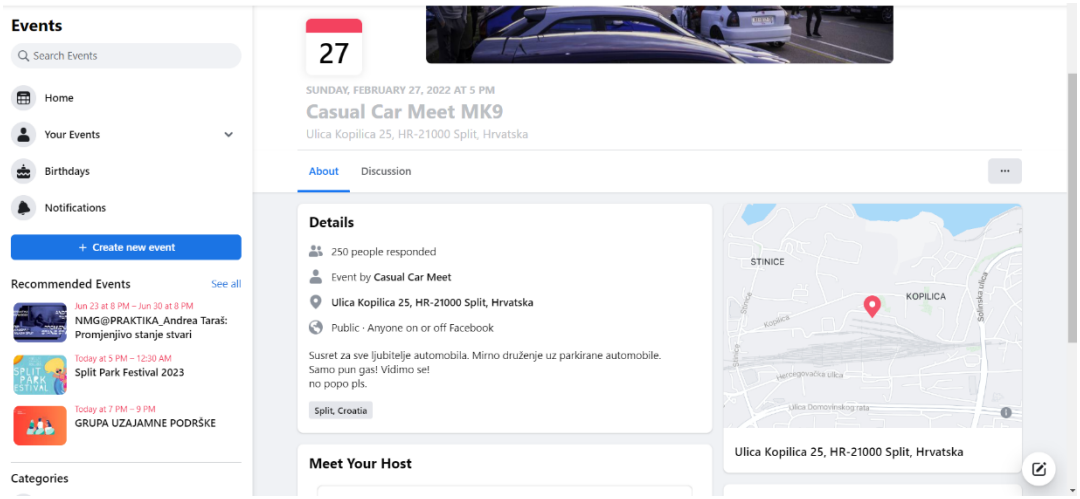
Slika 2.2. *Prikaz početne stranice Car Throttle web portala.*

2.3. Facebook - events

Facebook je društvena mreža koja krisnicima nakon registracije olakšava međusobno povezivanje putem poruka, objava, grupa, te organizacijom događaja - “event ”. Facebookova organizacija događaja omogućuje korisnicima da stvaraju, promoviraju i upravljaju događajima putem Facebook platforme.

Nekoliko ključnih značajki Facebook organizacije događaja:

- Stvaranje događaja: Korisnici mogu jednostavno stvoriti događaj i unijeti sve relevantne informacije poput naziva događaja, datuma, vremena, mjesta, opisa i slika.
- Interakcija i angažman: Sudionici događaja mogu postavljati pitanja, komentirati, dijeliti slike i videozapise te izražavati svoje zanimanje ili prisutnost.
- Pregled događaja: Organizatori imaju pregled statistika događaja, uključujući broj zainteresiranih i prisutnih sudionika te opću angažiranost s događajem [3].



Slika 2.3. *Prikaz Facebook-ovog “Car Meet” događaja.*

3. TEHNOLOGIJE KORIŠTENE PRI IZRADI WEB APLIKACIJE

Pri izradi web aplikacija moguće je koristit brojene programske jezike i tehnologije. U nastavku su objašnjene one tehnologije koje su korištene pri izradi web portala za ljubitelje automobilske industrije. Spomenute tehnologije se mogu podijeliti u dva dijela, a to su *frontend* i *backend*.

Frontend web stranice je ono što se vidi i s čime se koristi putem preglednika također se naziva i klijentska strana, uključuje sve sa čime korisnik može imati direktnu interakciju: tekst i boja poveznica, slika, obrazaca te raznih izbornika. Iako se *frontend* bavi interakcijom web stranice s korisnikom te služi kako bi korisniku olakšao rukovanje sučeljem, UX (engl. *User Experience*) / UI (engl. *User Interface*) dizajneri se bave izgledom web stranice. To znači da prije nego što *frontend* developer krene s izradom web aplikacije, izgled najprije dobije u obliku .sketch datoteke koji onda realizira u samu stranicu. Može se reći da *frontend* developeri oživljuju osmišljeni dizajn te dizajn pretvaraju u interaktivnu stranicu koristeći programske jezike kao što su React, Angular te strukturne jezike kao primjerice HTML (engl. *HyperText Markup Language*) i CSS (engl. *Cascading Style Sheets*).

Backend ili serverska strana je dio web stranice koji nije vidljiv korisnicima. *Backend* je odgovoran za pohranjivanje, manipulaciju te organizaciju podataka, također pomaže *frontendu* na način da mu šalje sve podatke potrebne za prikaz na samome ekranu. Određenim interakcijama korisnik svjesno – ispunjavanjem nekakve forme ili nesvjesno – inicijalnim dolaskom na stranicu, šalje zahtjev poslužiteljskoj strani koja vraća određene informacije, slike i drugo koje onda *frontend* obradi i preglednik prikaže korisniku. Kako bi stranica bila interaktivna i dinamična *frontend* i *backend* moraju raditi u skladu kako bi omogućili korisnički unos, pohranu, izmjenu te brisanje određenih podataka ih nekakvog spremnika, koji se u ovome slučaju nalazi na serverskoj strani aplikacije. Sama manipulacija i interakcija sa serverskom stranom aplikacije je razlika između interaktivne/dinamične aplikacije i statičke koja nema nikakve interakcije sa serverskom stranom ili najčešće ju niti nema. *Backend* programeri moraju koristiti jezike koji će imati mogućnost komunikacije s bazom podataka, a neki od njih su PHP, Java Spring Boot [4].

3.1. HTML

HTML je standardni jezik za izgradnju i strukturiranje web stranica. Koristi se za definiranje strukture i sadržaja web stranica putem posebnih oznaka ili "tagova". HTML samo određuje strukturu web stranice, stoga on sam nije dovoljan da web stranica izgleda vizualno dobro i interaktivno. Kako bi se HTML uljepšao i prikazao korisniku na najljepši način, uz njega se još koriste CSS i JavaScript. Najlakša interpretacija odnosa bi bila zamisliti da je HTML ljudski kostur, JavaScript srce/mozak te CSS koža.

Važni dijelovi HTML-a su oznake “tagovi” i atributi. HTML koristi tagove za označavanje elemenata na web stranici. Svaki tag ima svoje ime i okružuje određeni dio sadržaja. Primjeri tagova su <h1> (naslov prvog reda), <p> (paragraf), <a> (hiperveza) itd.. Oznake omogućuju stvaranje takvog rasporeda. HTML koristi hijerarhijsku strukturu koja se naziva DOM (engl. *Document Object Model*) kako bi organizirao sadržaj web stranice. To znači da elementi mogu biti ugniježđeni jedan u drugome, stvarajući stablo elemenata. Tagovi se mogu proširiti atributima koji pružaju dodatne informacije o elementima. Na primjer, <a> tag koristi atribut "href" za definiranje odredišne adrese hiperveze. Oni uzimaju vrijednosti, koje prenose više informacija o elementu i pomažu pri izradi stvari kao što su stiliziranje i manipulacija JavaScriptm. Ukratko HTML je osnova svake web stranice i važan je za razumijevanje kako se web stranice strukturiraju i prikazuju u preglednicima. Logo HTML-a prikazan je na slici 3.1 [5].

Slika 3.1. *Logo HTML-a.*

3.2. CSS

CSS je jezik koji se koristi za stiliziranje i oblikovanje web stranica. Dok HTML definira strukturu i sadržaj web stranica, CSS se koristi za kontroliranje izgleda i prezentacije tih elemenata. CSS se koristi za definiranje stilova za web stranice, uključujući dizajn, izgled i varijacije prikaza za različite uređaje i veličine zaslona. CSS koristi selektore kako bi odabrali određene HTML elemente kojima će se primijeniti stilovi. Selektori mogu biti bazirani na imenu elementa, ID-u, klasi, atributima i drugim karakteristikama elemenata. CSS koristi stilski pravila koja se sastoje od selektora i deklaracija. Selektor odabire elemente, dok deklaracije definiraju stilove koji se primjenjuju na te elemente. CSS omogućava dodjeljivanje klasa i ID-ova elementima radi preciznijeg odabira i stiliziranja. Klase se mogu dodijeliti više elemenata, dok ID-ovi trebaju biti jedinstveni za svaki element na stranici. Logo CSS-a prikazan je na slici 3.2 [6].

Slika 3.2. *Logo CSS-a.*

3.3. JavaScript

JavaScript je najraširenij skriptni jezik. JavaScript je programski jezik koji se najčešće koristi za izradu interaktivnih elemenata na web stranicama. To je skriptni jezik, što znači da se izvršava izravno u pregledniku korisnika i može interaktivno mijenjati sadržaj i ponašanje web stranice. JavaScript omogućava izradu interaktivnih elemenata na web stranicama. Može se koristiti za obradu događaja poput klika mišem, pritiska tipke ili prijenosa miša preko elemenata. Također se može koristiti za dinamičko ažuriranje sadržaja stranice, manipuliranje HTML elementima i obavljanje animacija. JavaScript koristi sličnu sintaksu kao i drugi programski jezici poput C-a ili Jave. JavaScript je snažan alat za manipulaciju DOM-a koji predstavlja strukturu

HTML-a. Može se koristiti za pristupanje HTML elementima, mijenjanje njihovih svojstava, dodavanje ili uklanjanje elemenata i manipuliranje sadržajem stranice. Postoji mnogo "frameworka" (poput Reaca Angulara ili Vuea) koji se grade na vrhu JavaScripta i olakšavaju izradu složenijih web aplikacija. Svaki veći web preglednik u sebi ima svoju posebnu implementaciju JavaScripta [7]. JavaScript se koristi ne samo za izradu web stranica, već i za razvoj mobilnih aplikacija (putem "frameworka" poput React Nativea) Na slici 3.3. prikazan je logo JavaScripta.

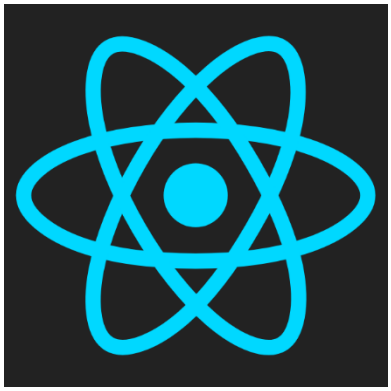
Slika 3.3. *Logo JavaScripta*

3.4. React

React je JavaScript okvir otvorenog koda za izradu korisničkog sučelja koji je razvila tvrtka Facebook. Koristi se za izgradnju skalabilnih i reaktivnih web aplikacija. React se temelji na komponentnoj arhitekturi, što znači da se aplikacija gradi iz više manjih i ponovno upotrebljivih komponenti. Svaka komponenta ima svoje stanje (engl. *statea*) i svojstva (engl. *propsa*) koja omogućuju dinamičko ažuriranje i reaktivnost. React koristi Virtualni DOM kako bi efikasno ažurirao korisničko sučelje. Umjesto direktnog manipuliranja stvarnim DOM-om, React stvara virtualnu reprezentaciju DOM-a koja se ažurira i uspoređuje s pravim DOM-om, a zatim se samo promijenjene dijelove primjenjuju na stvarni DOM. To poboljšava performanse i brzinu aplikacije. React koristi JSX (engl. *JavaScript Extension*), ekstenziju JavaScripta koja omogućava pisanje HTML sličnog koda u JavaScriptu. JSX olakšava pisanje komponenti i njihovih prikaza, te omogućava kombiniranje logike i prikaza u istom kodu. React promiče koncept jednosmjernog podataka, što znači da podaci teku samo u jednom smjeru - od vršne komponente prema potkomponentama. Ovo olakšava praćenje stanja aplikacije i upravljanje podacima. React

komponente imaju različite metode životnog ciklusa koje se automatski pozivaju u određenim fazama života komponente. React je vrlo fleksibilan i moćan *framework* koji omogućava izgradnju modernih, reaktivnih i skalabilnih korisničkih sučelja za različite platforme [8].

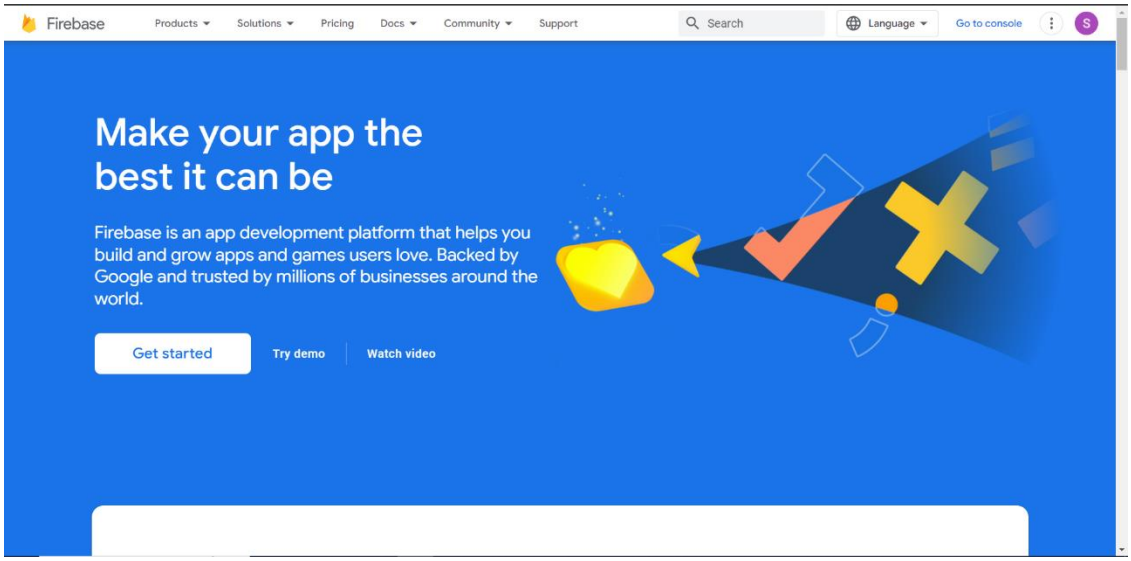
React.js logo prikazan je na slici 3.4.



Slika 3.4. *React logo.*

3.5. Firebase

Firebase je platforma za razvoj aplikacija u oblaku koju je razvio Google [9]. Pruža različite usluge i alate za izgradnju, poboljšanje i upravljanje web i mobilnih aplikacija. Firebase Realtime Database je cloud-based baza podataka koja omogućava sinkronizaciju podataka u stvarnom vremenu između različitih korisnika ili uređaja. Podaci se mogu ažurirati i dohvaćati u stvarnom vremenu, što je idealno za izradu aplikacija s trenutačnim ažuriranjima. Firebase pruža autentifikaciju korisnika putem različitih davatelja usluga kao što su Google, Facebook, Twitter, GitHub i drugi. To olakšava implementaciju sigurnog sustava za prijavu i registraciju korisnika u aplikacijama. Firebase Cloud Functions omogućava pisanje i izvršavanje backend koda u oblaku. Možete koristiti Cloud Functions za obradu događaja ili izvršavanje skripti, to omogućava skalabilnost i fleksibilnost u izgradnji *backend* funkcionalnosti aplikacija. Firebase *Cloud-Storage* je usluga za pohranu i upravljanje datotekama u oblaku. Može se koristiti za spremanje i dijeljenje slika, videozapisa, dokumenta i drugih medija. Pruža skalabilnost, sigurnost i jednostavno upravljanje datotekama. Firebase je popularan među programerima zbog svoje jednostavnosti korištenja, skalabilnosti i integriranih usluga. Početna stranica Firebasea na kojoj se vide brojne mogućnosti prikazana je na slici 3.5.



Slika 3.5. *Početna stranica firebase-a na kojoj se vide brojne mogućnosti.*

3.7. Visual Studio Code

Microsoft Visual Studio Code (VS Code) je besplatan, otvorenog koda i univerzalni tekstni uređivač koji je razvio Microsoft. Dolazi s ugrađenom podrškom za JavaScript, TypeScript i Node.js te ima bogat ekosustav ekstenzija za druge programske jezike (kao što su C++, C#, Java, Python, PHP i Go) također je dostupan na velikom broju operativnih sustava kao što su Windows, MAC OS, Debian os i drugi. Microsoft Visual Studio Code dolazi s bogatom podrškom za sintaksu, automatsko dovršavanje koda, refaktoriranje, debugiranje, integraciju s kontrolnim sustavima verzija i mnoge druge značajke koje olakšavaju razvojni proces. VS Code je integriran s raznim alatima i servisima koji olakšavaju razvojni proces. Pruža integraciju s kontrolnim sustavima verzija poput Git-a, alatima za upravljanje paketima poput npm-a ili pip-a, alatima za kontinuiranu integraciju (CI) poput Azure Pipelines-a i još mnogo toga. Visual Studio Code ima veliku zajednicu korisnika i developera. Zajednica stalno razvija nova proširenja, teme i dijeli svoja iskustva i znanje putem foruma, blogova i društvenih mreža. Ovo pruža bogat izvor podrške i resursa za sve korisnike VS Codea. Logo Visual Studio Codea prikan je na slici 3.6. [10].

Slika 3.6. *Logo Visual Studio Codea.*

3.8. GitHub

GitHub je web platforma koja pruža usluge za upravljanje izvorima koda pomoću sistema za kontrolu verzija Git. Omogućuje programerima zajednički rad na projektima, praćenje promjena, upravljanje problemima, praćenje zadatka te olakšava distribuciju softvera. Glavne prednosti GitHuba mogućnost su suradnje timova na udaljenim lokacijama, stvaranje grananja za testiranje novih značajki ili popravke problema nastalih u kodu te integracija s raznim alatima za automatizaciju i kontinuiranu integraciju. GitHub također pruža prostor za pohranu i dijeljenje projekata, s naglaskom na otvorenom kodu, što doprinosi transparentnosti i zajedničkom razvoju softvera. Razvijatelji mogu pridonositi projektima putem *pull* zahtjeva, pregledavati i komentirati kod te pratiti aktivnosti unutar zajednice. GitHub logo prikazan je na slici 3.7. [11].

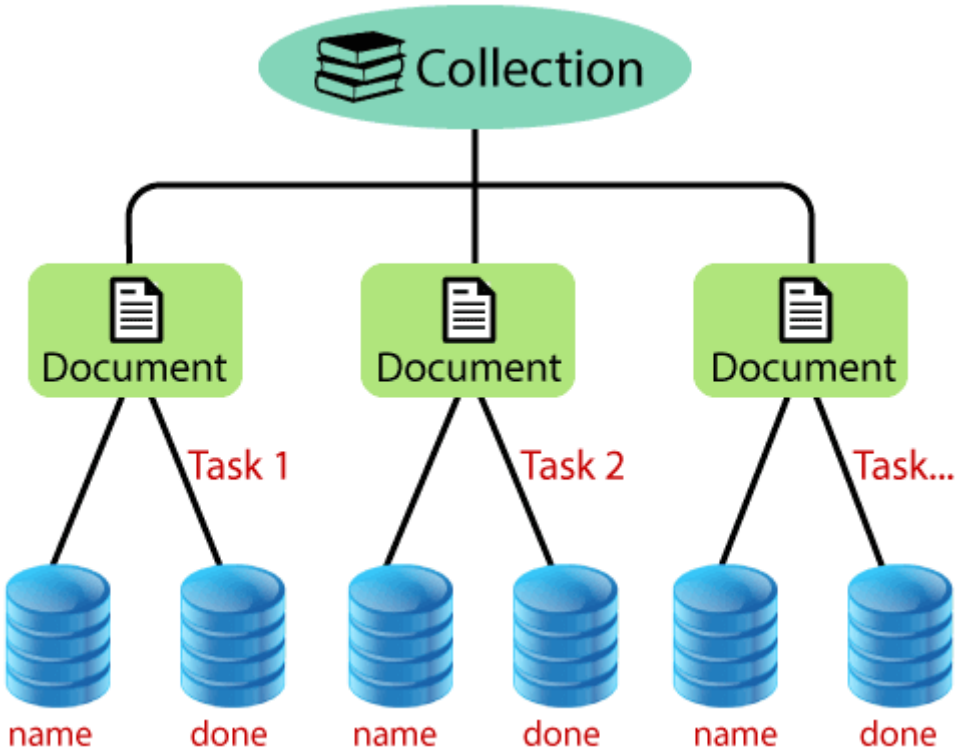
Slika 3.7. *GitHub logo.*

4. FUNKCIONALNOST WEB APLIKACIJE

U ovom poglavlju pobliže se opisuju funkcije korištene pri izradi web portala zajednice ugađanja automobila.

4.1. Baza podataka

Za kreiranje baze podataka korišten je sustav Firebase. Konkretno, Firebase Realtime Database, Firebase Storage te Firebase Authentification. Korišteno je NoSQL načelo spremanja i korištenja podataka, to znači da su baze podataka fleksibilne i omogućavaju jednostavno spremanje i dohvat podataka u JSON (engl. *JavaScript Object Notation*) formatu, što je karakteristično za NoSQL baze podataka. Firebase Realtime Database modelira se kao kolekcija dokumenata, dopušta korisnicima stvaranje kolekcija unutar kolekcija što intenzivno povećava njezinu korisnost te olakšava iščitavanje njezinih podataka. Na slici 4.1. je prikazan model zapisa podataka u NoSQL bazi podataka.

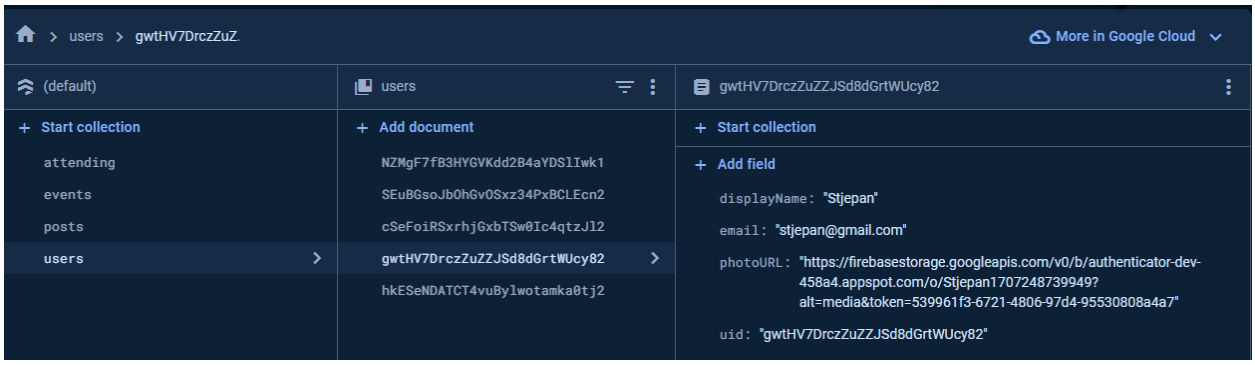


Slika 4.1. *Model zapisa podataka u NoSQL bazi podataka.*

4.2. Registracija i prijava

Web portal zamišljen je kako bi ponudio razne funkcionalnosti svim korisnicima, ovisno o njihovoj ulozi nude im se određene mogućnosti. Korisnik koji nije prošao proces autentifikacije je u ulozi gosta te su mu ograničene mogućnosti na stranici.

Proces autentifikacije zahtjeva od korisnika da obavi registraciju te da se prijavi u aplikaciju. Registracija zahtjeva od korisnika da unese svoje korisničko ime, svoj email, kreira zaporku te da u privitak doda sliku avatara. Nakon unosa podataka klikom na gumb *Sign up* u bazi podataka u kolekciji *users* kreira se dokument sa korisničkim podatcima registriranog korisnika. Slika 4.2 prikazuje zapis korisnika u bazi podataka.



Slika 4.2. *Prikaz zapisa registriranog korisnika u bazi podataka.*

```
try {
  const res = await createUserWithEmailAndPassword(auth, email, password);
  const date = new Date().getTime();
  const storageRef = ref(storage, `${displayName + date}`);
  await uploadBytesResumable(storageRef, file).then(() => {
    getDownloadURL(storageRef).then(async (downloadURL) => {
      try {
        await updateProfile(res.user, {
          displayName,
          photoURL: downloadURL,
        });
        await setDoc(doc(db, "users", res.user.uid), {
          uid: res.user.uid,
          displayName,
          email,
          photoURL: downloadURL,
        });
        setAvatarSelected(true);
        setLoading(false);
        navigate("/");
      } catch (err) {
        setErr(true);
        setLoading(false);
      }
    });
  });
} catch (err) {
  setErr(true);
  setLoading(false);
}
```

Slika 4.3. *Prikaz logike registriranja novog korisnika.*

Ukoliko se korisnik već registrirao, ima opciju prijave u aplikaciju putem *login*. Na login opciji korisnik mora unijeti email i lozinku kako bi pokrenuo proces autentifikacije. Ukoliko su korisnički podatci točni korisnika se propušta u aplikaciju. Ukoliko korisnik nije u mogućnosti zadovoljiti proces autentifikacije, mora se javiti administratoru stranice kako bi se otkrila priroda problema. Slika 4.4. prikazuje logiku prijave i izgled forme na *login* stranici.

```jsx
const Login = () => {

    const [err, setErr] = useState(false);
    const navigate = useNavigate();

    const handleSubmit = async (e) => {
        e.preventDefault();
        const email = e.target[0].value;
        const password = e.target[1].value;

        try {
            await signInWithEmailAndPassword(auth, email, password);
            navigate("/")
        } catch (err) {
            setErr(true);
        }
    };

    return (
        <div className='formContainer'>
            <div className='formWrapper'>
                <span className='logo'>GearsAndMotors</span>
                <span className='title'>Login</span>
                <form onSubmit={handleSubmit}>
                    <input type='email' placeholder='email'/>
                    <input type='password' placeholder='password'/>
                    <button>Login</button>
                    {err && <span>Something went wrong</span>}
                </form>
                <p><Link to={"/register"}>Sign up</Link> / <Link to={"/home"}>continue as a guest</Link></p>
            </div>
        </div>
    )
}
```

Slika 4.4. *Login forma i logika prijave.*

Firebase nudi administratoru opciju praćenja korisničkih interakcija sa autentifikacijskim procesima. Administrator može vidjeti kada je korisnik kreirao račun, kada se zadnji puta prijavio te može omogućiti korisniku resetiranje korisničke zaporke kao i mogućnost brisanja korisničkog računa. Slika 4.5 prikazuje administrativne mogućnosti sa korisničkim računima.

Slika 4.5. *Prikaz administrativnog panela.*

4.2. Odjava

Odjava iz aplikacije obavlja se klikom na gumb *Logout* koji se nalazi na desnoj strani navigacijske trake. Odjavom se brišu korisnički podatci iz lokalnog spremišta preglednika te se korisnik vodi na *Login* stranicu gdje se može ponovno prijaviti ili nastaviti pregledavati aplikaciju kao gost. Na slici 4.6. je prikazan proces odjave i preusmjeravanje na *Login* stranicu.

```
21  const handleLogout = async () => {
22    await logoutUser();
23    setAuthenticated(false);
24    navigate("/login");
25  };
```

Slika 4.6. *Logika odjavljivanja iz aplikacije.*

4.3. Prijavljeni korisnici

Prolaskom kroz proces registracije i prijave korisnici postaju članovi web portala te imaju slobodu interakcije sa sadržajem na stranici. Ovaj proces je važan ako korisnici žele stvarati objave, komentirati objave te organizirati događaje. Kreiranje objave se odvija odabirom *Create post* opcije koju je moguće vidjeti na padajućem izborniku klikom na *Rate my car* opciju koja se nalazi na navigacijskoj traci. Slika 4.7. prikazuje logiku korištenu za pohranjivanje objava u bazu podataka.

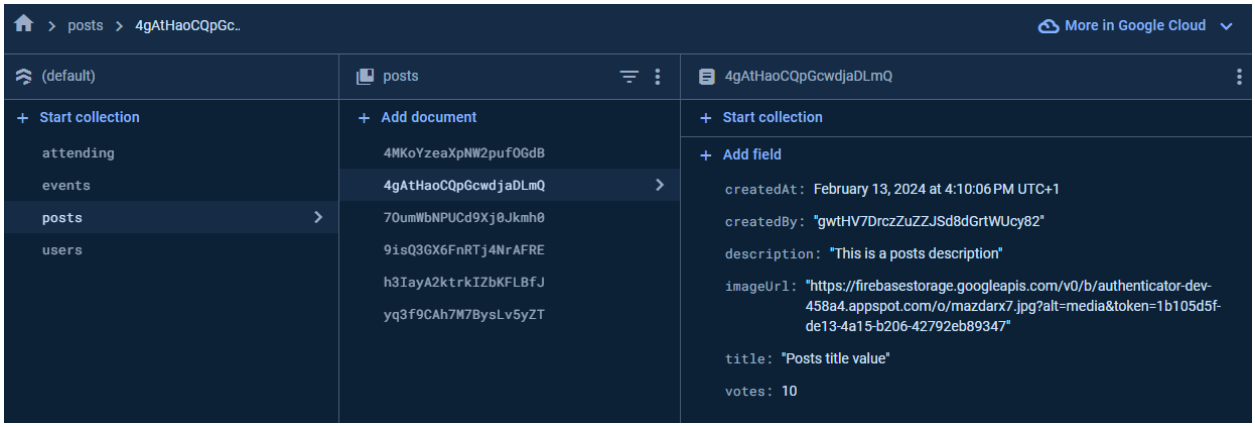
```
30  const handlePostCreation = async () => {
31    if (!title || !image) {
32      setAlertStyle(true);
33      setShowAlert(true);
34      return;
35    }
36    const currentUser = auth.currentUser;
37    const storageRef = ref(storage, image.name);
38    await uploadBytes(storageRef, image);
39    const imageUrl = await getDownloadURL(storageRef);
40    await addDoc(collection(db, "posts"), {
41      title,
42      description,
43      imageUrl,
44      createdBy: currentUser.uid,
45      createdAt: new Date(),
46      votes: 0,
47    });
48    navigate("/");
49  };
```

Slika 4.7. *Logika kreiranja objave.*

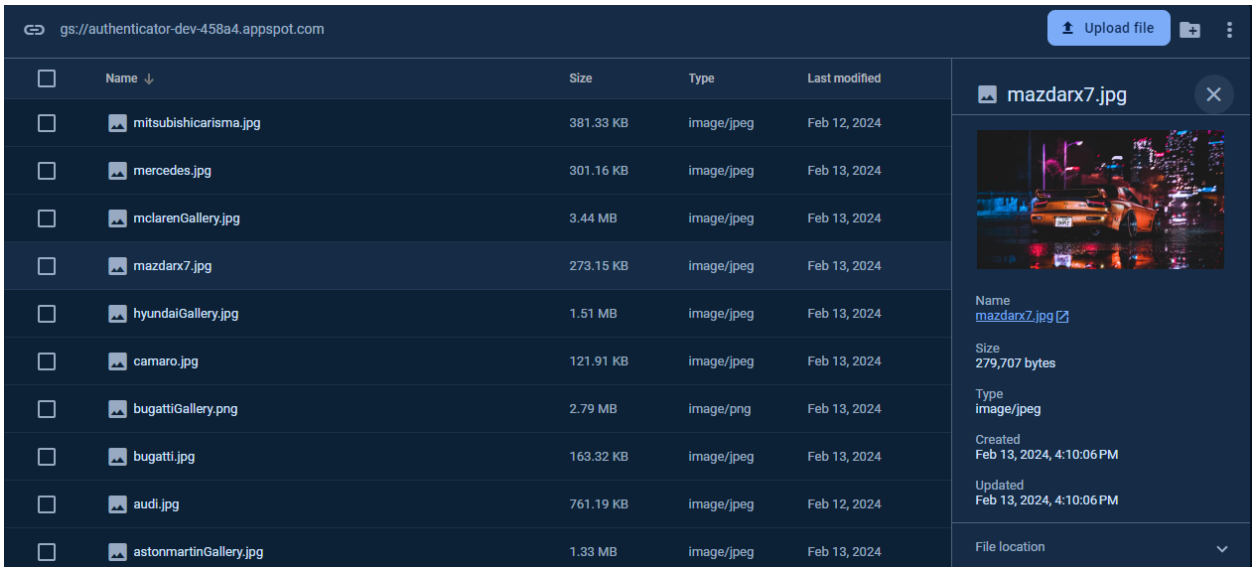
Ukoliko su svi uvjeti kreiranja objave zadovoljeni i komunikacija sa Firebase bazom podataka je uspješna, kreirana objava se sprema u dva djela:

- Podatci spremljeni u Firebase Realtime Database,
- Komprimirana fotografija spremljena u Firebase Storage.

U Firebase Realtime Database se spremaju specifični podatci autora te specifični podatci objave. Komprimirana fotografija sprema se u Firebase Storage. Na slici 4.8. prikazana je spremljena objava, a na slici 4.9. prikazano je spremište komprimiranih fotografija.

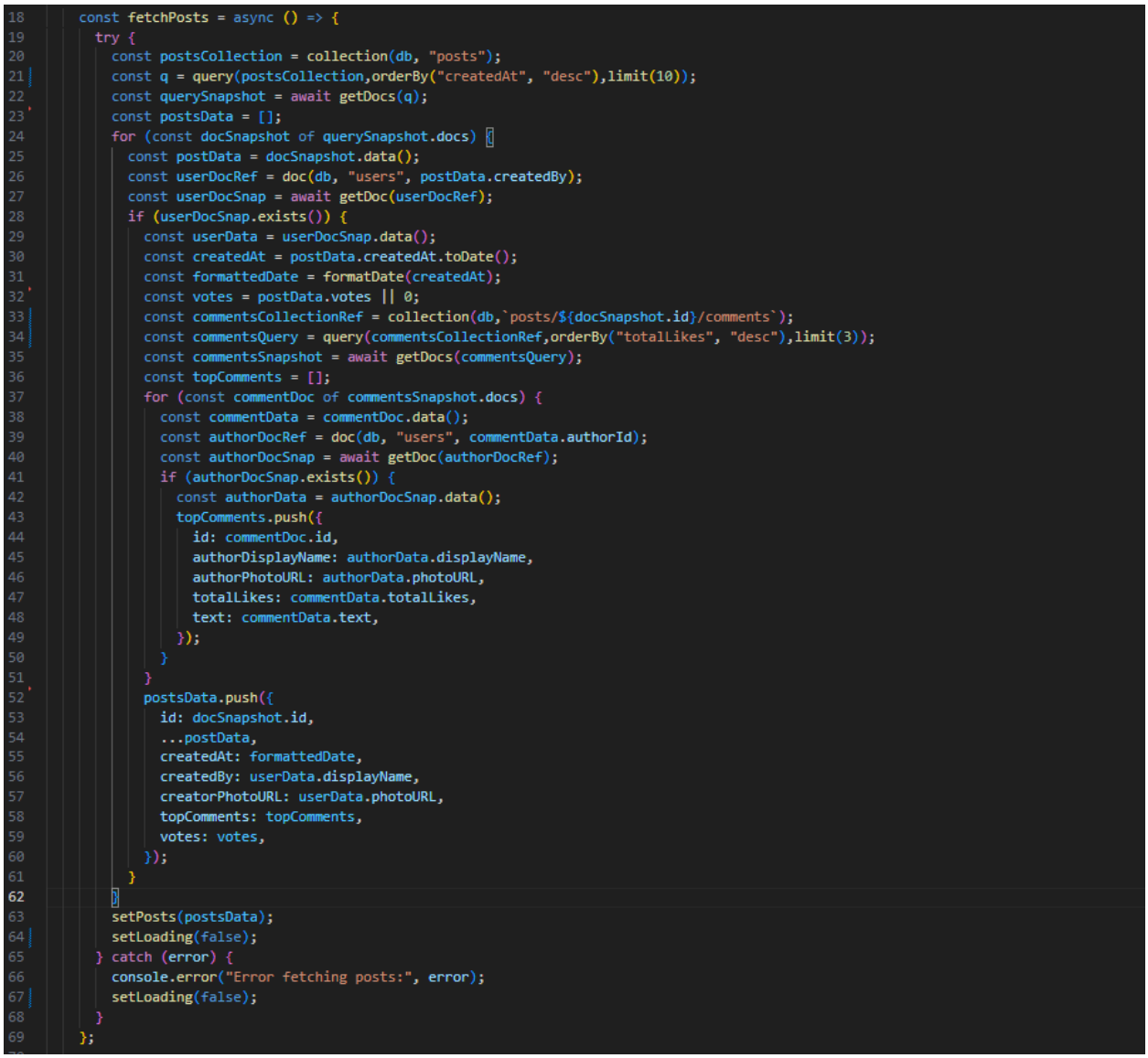


Slika 4.8. *Izgled spremljene objave u Firebase Realtime Database bazi.*



Slika 4.9. *Izgled komprimirane fotografije u Firebase Storage bazi.*

Nakon uspješnog spremanja u bazu i kreiranja objave, korisnika se odvodi na *Home* stranicu gdje on ima mogućnost vidjeti svoju objavu te objavu drugih korisnika odlaskom na *View posts* stranicu. Na slici 4.10. prikazana je logika dohvaćanja svih korisničkih objava te prikaz prva tri komentara s najviše interakcija.



```
const fetchPosts = async () => {
  try {
    const postsCollection = collection(db, "posts");
    const q = query(postsCollection,orderBy("createdAt", "desc"),limit(10));
    const querySnapshot = await getDocs(q);
    const postsData = [];
    for (const docSnapshot of querySnapshot.docs) {
      const postData = docSnapshot.data();
      const userDocRef = doc(db, "users", postData.createdBy);
      const userDocSnap = await getDoc(userDocRef);
      if (userDocSnap.exists()) {
        const userData = userDocSnap.data();
        const createdAt = postData.createdAt.toDate();
        const formattedDate = formatDate(createdAt);
        const votes = postData.votes || 0;
        const commentsCollectionRef = collection(db,`posts/${docSnapshot.id}/comments`);
        const commentsQuery = query(commentsCollectionRef,orderBy("totalLikes", "desc"),limit(3));
        const commentsSnapshot = await getDocs(commentsQuery);
        const topComments = [];
        for (const commentDoc of commentsSnapshot.docs) {
          const commentData = commentDoc.data();
          const authorDocRef = doc(db, "users", commentData.authorId);
          const authorDocSnap = await getDoc(authorDocRef);
          if (authorDocSnap.exists()) {
            const authorData = authorDocSnap.data();
            topComments.push({
              id: commentDoc.id,
              authorDisplayName: authorData.displayName,
              authorPhotoURL: authorData.photoURL,
              totalLikes: commentData.totalLikes,
              text: commentData.text,
            });
          }
        }
        postsData.push({
          id: docSnapshot.id,
          ...postData,
          createdAt: formattedDate,
          createdBy: userData.displayName,
          creatorPhotoURL: userData.photoURL,
          topComments: topComments,
          votes: votes,
        });
      }
    }
    setPosts(postsData);
    setLoading(false);
  } catch (error) {
    console.error("Error fetching posts:", error);
    setLoading(false);
  }
};
```

Slika 4.10. *Logika dohvaćanja svih objava.*

Klikom na neku od objava, korisnik ima pravo pobliže vidjeti objavu. Na slici 4.11. prikazana je logika koja propušta prijavljene korisnike u detaljni prikaz objave. Korisniku se otvaraju detaljniji opis objave, podatci autora te svi komentari postavljeni na odabranu objavu. Na slici 4.12. prikazana je logika dohvaćanja detalja objave. Korisniku se nudi mogućnost interakcije s objavom. Interakciju s objavom korisnik može ostvariti kroz direktno glasanje za određenu objavu. Korisnik može dati objavi *upvote* ili *downvote* što će utjecati na ukupan broj glasova za pojedinu objavu. Na slici 4.13. prikazana je logika dodjeljivanja glasova za pojedinu objavu.

```
87  const handlePostClick = (postId) => {
88    const auth = getAuth();
89    onAuthStateChanged(auth, (user) => {
90      if (user) {
91        navigate(`/post-view`, { state: { postId } });
92      } else {
93        setShowAlert(true);
94        setAlertStyle(true);
95      }
96    });
97  };
```

Slika 4.11. *Logika propuštanja korisnika na prikaz detalja objave.*

```
64  const fetchPost = async () => {
65    const postId = location.state?.postId;
66    if (!postId) {
67      navigate("/posts-view");
68      return;
69    }
70
71    try {
72      const postDoc = await getDoc(doc(db, "posts", postId));
73      if (postDoc.exists()) {
74        const postData = postDoc.data();
75        const createdAtTimestamp = postData.createdAt.toDate();
76        const createdAtString = formatCreatedAtDate(createdAtTimestamp);
77        const userRef = doc(db, "users", postData.createdBy);
78        const userSnapshot = await getDoc(userRef);
79        if (userSnapshot.exists()) {
80          const userData = userSnapshot.data();
81          setPost({
82            id: postDoc.id,
83            ...postData,
84            createdBy: userData.displayName,
85            creatorPhotoURL: userData.photoURL,
86            createdAt: createdAtTimestamp,
87          });
88        }
89      } else {
90        console.log("Post not found");
91      }
92    } catch (error) {
93      console.error("Error fetching post:", error);
94    }
95  };
```

Slika 4.12. *Logika dohvaćanja detalja objave.*

```
const handleVote = async (type) => {
  if (processingVote) return;

  try {
    setProcessingVote(true);
    const user = auth.currentUser;
    const userVoteRef = doc(db, `posts/${post.id}/userVotes`, user.uid);
    const userVoteDoc = await getDoc(userVoteRef);
    let voteChange = 0;
    if (userVoteDoc.exists()) {
      const userVoteData = userVoteDoc.data();
      if (userVoteData.vote === type) {
        await deleteDoc(userVoteRef);
        voteChange = type === "upvote" ? -1 : 1;
      } else {
        await updateDoc(userVoteRef, { vote: type });
        voteChange = type === "upvote" ? 2 : -2;
      }
    } else {
      await setDoc(userVoteRef, { vote: type });
      voteChange = type === "upvote" ? 1 : -1;
    }
    await updateDoc(doc(db, "posts", post.id), {
      votes: post.votes + voteChange,
    });
    const postRefetch = await getDoc(doc(db, "posts", post.id));
    if (postRefetch.exists()) {
      const postData = postRefetch.data();
      const createdAtTimestamp = postData.createdAt.toDate();
      setPost({
        id: postRefetch.id,
        ...postData,
        createdBy: post.createdBy,
        creatorPhotoURL: post.creatorPhotoURL,
        createdAt: createdAtTimestamp,
      });
    }
    setUserVote(type);
    setProcessingVote(false);
  } catch (error) {
    console.error("Error handling vote:", error);
    setProcessingVote(false);
  }
};
```

Slika 4.13. *Logika dodjeljivanja glasova za pojedinu objavu.*

Prijavljeni korisnici imaju mogućnost interakcije s komentarima za odabranu objavu. Korisnici mogu postavljati svoje komentare koji se prikazuju odmah po objavi, te mogu komentare drugih korisnika označiti sa *upvote.* Na slici 4.14. je prikazana logika iza objavljivanja komentara.

```
const handleCommentSubmit = async () => {
  try {
    if (comment.length > 100) {
      console.error("Comment exceeds 100 characters.");
      return;
    }
    const user = auth.currentUser;
    const newCommentRef = await addDoc(
      collection(db, `posts/${post.id}/comments`),
      {
        authorId: user.uid,
        text: comment,
        createdAt: new Date(),
        totalLikes: 0,
        updating: false,
      }
    );
    setComment("");
    const newCommentDoc = await getDoc(newCommentRef);
    if (newCommentDoc.exists()) {
      const newCommentData = newCommentDoc.data();
      const userRef = doc(db, "users", newCommentData.authorId);
      const userSnapshot = await getDoc(userRef);
      if (userSnapshot.exists()) {
        const userData = userSnapshot.data();
        setCommentsList([
          {
            id: newCommentDoc.id,
            ...newCommentData,
            authorDisplayName: userData.displayName,
            authorPhotoURL: userData.photoURL,
            likes: [],
          },
          ...commentsList,
        ]);
      }
    }
  } catch (error) {
    console.error("Error adding comment:", error);
  }
};
```

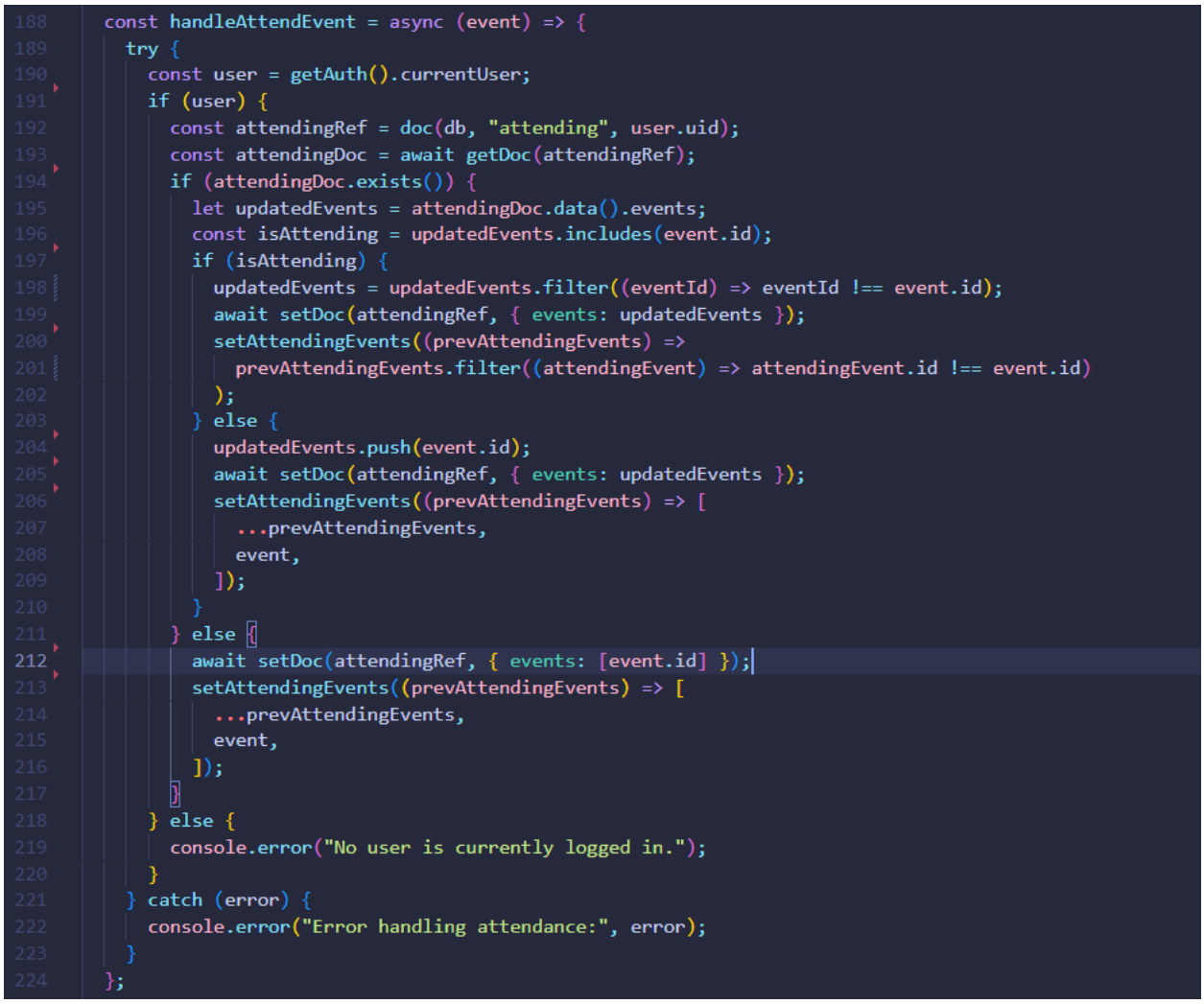
Slika 4.14. *Logika postavljanja komentara.*

Web portal nudi korisnicima pregled i organiziranje natjecanja. Tipovi natjecanja su *style* i *race*. Prijavljeni korisnici imaju mogućnost vidjeti organizirane događaje te kreiranje istih. Slika 4.15. prikazuje logiku iza dohvaćanja organiziranih događaja. Prijavljeni korisnici

pritiskom na pribadaču koja predstavlja događaj imaju opciju prijavljivanja za odabrani događaj. Slika 4.16. prikazuje logiku iza prijave korisnika za određeni događaj.

```
const fetchEvents = async () => {
  try {
    const eventsCollection = collection(db, "events");
    const eventsSnapshot = await getDocs(eventsCollection);
    const currentDate = new Date();
    const validEvents = [];
    const userPromises = [];
    eventsSnapshot.forEach((doc) => {
      const eventData = doc.data();
      const eventDateTime = new Date(eventData.eventDateTime);
      if (eventDateTime > currentDate) {
        const userPromise = getUserDisplayName(eventData.authorId).then(
          (displayName) => {
            validEvents.push({
              id: doc.id,
              ...eventData,
              authorDisplayName: displayName,
              formattedDateTime: eventDateTime.toLocaleDateString("en-US", {
                day: "numeric",
                month: "short",
                year: "numeric",
                hour: "numeric",
                minute: "numeric",
              }),
            });
          }
        );
        userPromises.push(userPromise);
      }
    });
    await Promise.all(userPromises);
    setMarkers(validEvents);
  } catch (error) {
    console.error("Error fetching events:", error);
  }
};
```

Slika 4.15. *Metoda za dohvaćanje organiziranih događaja.*



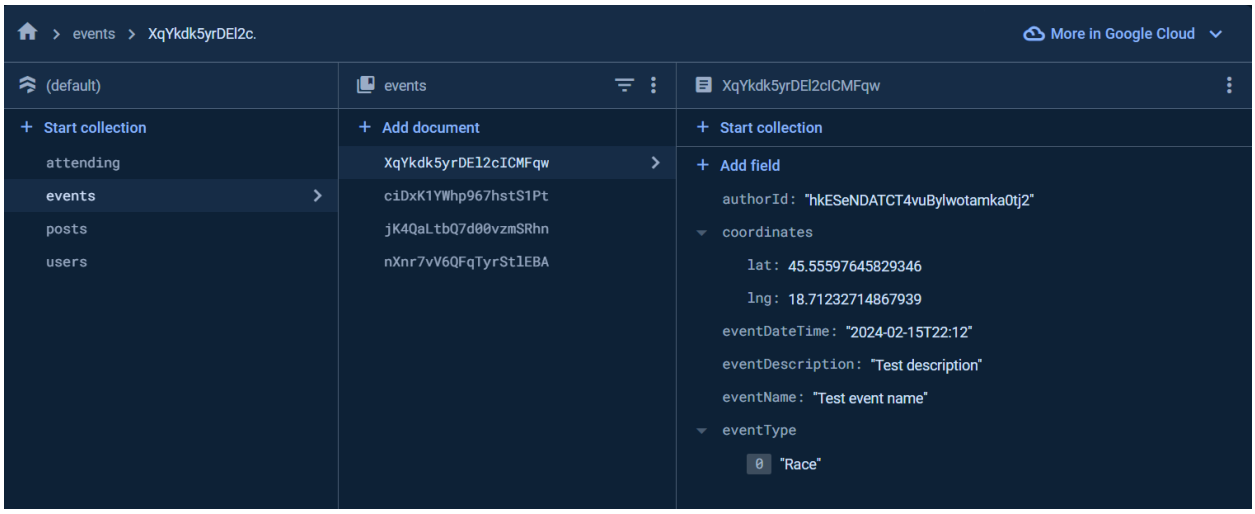
```
const handleAttendEvent = async (event) => {
  try {
    const user = getAuth().currentUser;
    if (user) {
      const attendingRef = doc(db, "attending", user.uid);
      const attendingDoc = await getDoc(attendingRef);
      if (attendingDoc.exists()) {
        let updatedEvents = attendingDoc.data().events;
        const isAttending = updatedEvents.includes(event.id);
        if (isAttending) {
          updatedEvents = updatedEvents.filter((eventId) => eventId !== event.id);
          await setDoc(attendingRef, { events: updatedEvents });
          setAttendingEvents((prevAttendingEvents) =>
            prevAttendingEvents.filter((attendingEvent) => attendingEvent.id !== event.id)
          );
        } else {
          updatedEvents.push(event.id);
          await setDoc(attendingRef, { events: updatedEvents });
          setAttendingEvents((prevAttendingEvents) => [
            ...prevAttendingEvents,
            event,
          ]);
        }
      } else {
        await setDoc(attendingRef, { events: [event.id] });
        setAttendingEvents((prevAttendingEvents) => [
          ...prevAttendingEvents,
          event,
        ]);
      }
    } else {
      console.error("No user is currently logged in.");
    }
  } catch (error) {
    console.error("Error handling attendance:", error);
  }
};
```

Slika 4.16. *Metoda prijave korisnika na događaj.*

Organiziranje natjecanja moguće je samo prijavljenim korisnicima i zahtijeva od korisnika da navede ime događaja, tip događaja, datum i vrijeme, kratki opis te mjesto. Na slici 4.17. je prikazana logika iza kreiranja događaja. Događaji su u bazi podataka predstavljeni kao kolekcija događaja te svaki događaj ima svoje osnovne informacije prikazane slikom 4.18.

```
const handleSubmit = async (e) => {
  e.preventDefault();
  if (
    !eventName ||eventType.length === 0 ||!eventDateTime ||!currentCoordinates ||!validateDescription()
  ) {
    alert(
      "Please fill in all required fields, including current coordinates and description."
    );
    return;
  }
  try {
    const user = getAuth().currentUser;
    if (user) {
      const eventRef = await addDoc(collection(db, "events"), {
        eventName,
        eventType,
        eventDateTime,
        eventDescription,
        coordinates: currentCoordinates,
        authorId: user.uid,
      });
      const eventDoc = await getDoc(eventRef);
      if (eventDoc.exists()) {
        console.log("Event added to Firestore:", eventDoc.data());
        setSubmissionSuccess(true);
      } else {
        console.error("Error adding event to Firestore.");
      }
    } else {
      console.error("No user is currently logged in.");
    }
  } catch (error) {
    console.error("Error submitting event:", error);
  }
  setEventName("");
  setEventType([]);
  setEventDateTime("");
  setEventDescription("");
  setMarkers([]);
  setCurrentCoordinates(null);
};
```

Slika 4.17. *Metoda za kreiranje događaja.*



Slika 4.18. *Prikaz osnovnih informacija događaja.*

Korisnici i posjetitelji imaju mogućnost pogledati polaznike koji su se prijavili na događaj. Logika iza pregleda polaznika na događaj prikazana je slikom 4.19.

```
const fetchAttendees = async () => {
  try {
    const attendeesRef = collection(db, "attending");
    const attendeesSnapshot = await getDocs(attendeesRef);
    const attendeesData = [];
    for (const docs of attendeesSnapshot.docs) {
      const userEventIds = docs.data().events;
      if (userEventIds.includes(eventId)) {
        const userId = docs.id;
        const userDoc = await getDoc(doc(db, "users", userId));
        const userData = userDoc.data();
        attendeesData.push({ id: userId, ...userData });
      }
    }
    setAttendees(attendeesData);
  } catch (error) {
    console.error("Error fetching attendees:", error);
  }
};
```

Slika 4.19. *Logika dohvaćanja polaznika pojedinog događaja.*

Neprijavljeni korisnici imaju samo mogućnost pregleda web portala, korisničkih objava i kreiranih događaja. Na određenim dijelovima portala ukoliko korisnik pokuša napraviti nešto što mu nije dozvoljeno jer nije obavio proces autentifikacije, bit će potaknut na prijavu ili registraciju. Provjera ispravnosti korisničkih podataka vrši se uz pomoć ugrađene Firebase *getAuth* metode te u ovisnosti o ispravnosti podataka korisnika se propušta dalje ili preusmjerava na *Login* stranicu. Slika 4.20. prikazuje logiku iza provjere autentičnosti korisničkih podataka.

```
const isUserAuthenticated = async () => {
  try {
    const user = auth.currentUser;
    if (user) {
      const userDocRef = doc(db, "users", user.uid);
      const userDoc = await getDoc(userDocRef);
      return userDoc.exists();
    } else {
      return false;
    }
  } catch (error) {
    console.error("Error checking user authentication:", error);
    return false;
  }
};
```

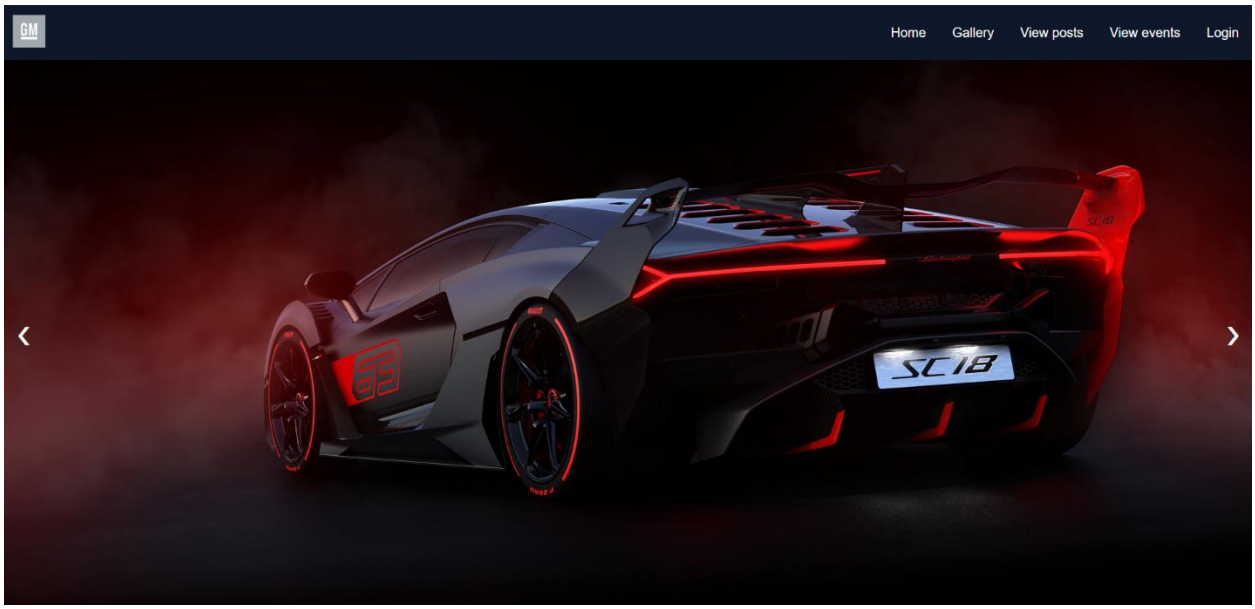
Slika 4.20. *Metoda provjere autentičnosti korisničkih podataka.*

5. KORIŠTENJE WEB APLIKACIJE

Web portal za ljubitelje automobila sadrži poglede vidljive samo registriranim korisnicima te one vidljive svim gostima portala. Gostima portala dostupne su mogućnosti uvida u novosti, objave korisnika, organiziranje događanja, no ne i interakcija sa istima. Ukoliko posjetitelji žele provesti ikakav oblik interakcije sa sadržajem na web portalu moraju proći proces registracije i prijave. Prijavljeni korisnici imaju mogućnost kreirati svoje objave te interaktirati sa objavama drugih korisnika. Također imaju opciju kreiranja događaja ili natjecanja, te interakciju sa događajima ili natjecanjima organiziranih od strane drugih korisnika. Administrator ima ulogu kao i ostali registrirani korisnici, no on ima uvid u bazu te tamo ima pravo ograničiti korisnike na određene načine ili im po potrebi pomoći.

5.1 Početni prikaz stranice

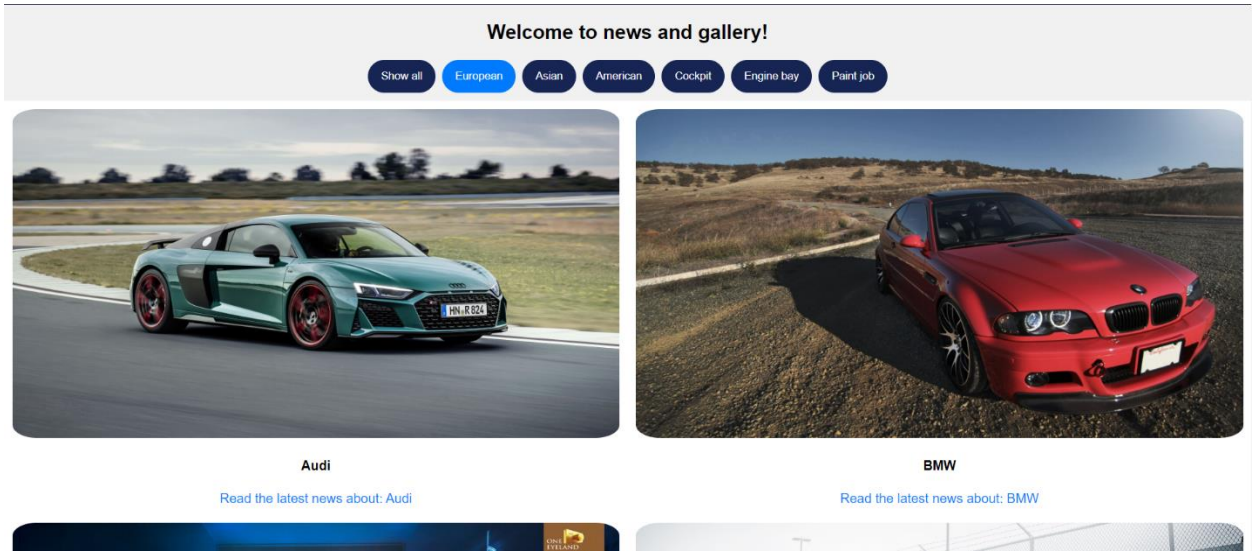
Početna stranica sastoji se od animiranog zaslona na kojemu se izmjenjuju okom ugodne slike, kratak opis web portala, administratorove poruke, te navigacijska traka. Ovisno o tome je li korisnik gost ili je prijavljeni korisnik navigacijska traka ima ograničeni prikaz ili puni prikaz mogućnosti. Neprijavljeni korisnik ima ograničeni prikaz trake na kojoj su mu ponuđene mogućnosti *Home*, *Gallery*, *View posts*, *View events* i *Login*. Prijavljeni korisnik ima puni prikaz navigacijske trake te su mu ponuđene mogućnosti *Home*, *Gallery*, *Create post*, *View posts*, *View events*, *Organize an event* i *Logout*. Navigacijska traka je responzivna te gledajući na malim zaslonima prelazi u padajući izbornik. Klikom na jednu od opcija na navigacijskoj traci korisnika se odvede na odgovarajuću stranicu. Klikom na tipku *Home* korisnika se vraća na početnu stranicu. Na slici 5.1. prikazana je stranica *Home*.



Slika 5.1. *Prikaz stranice Home.*

5.2 Gallery

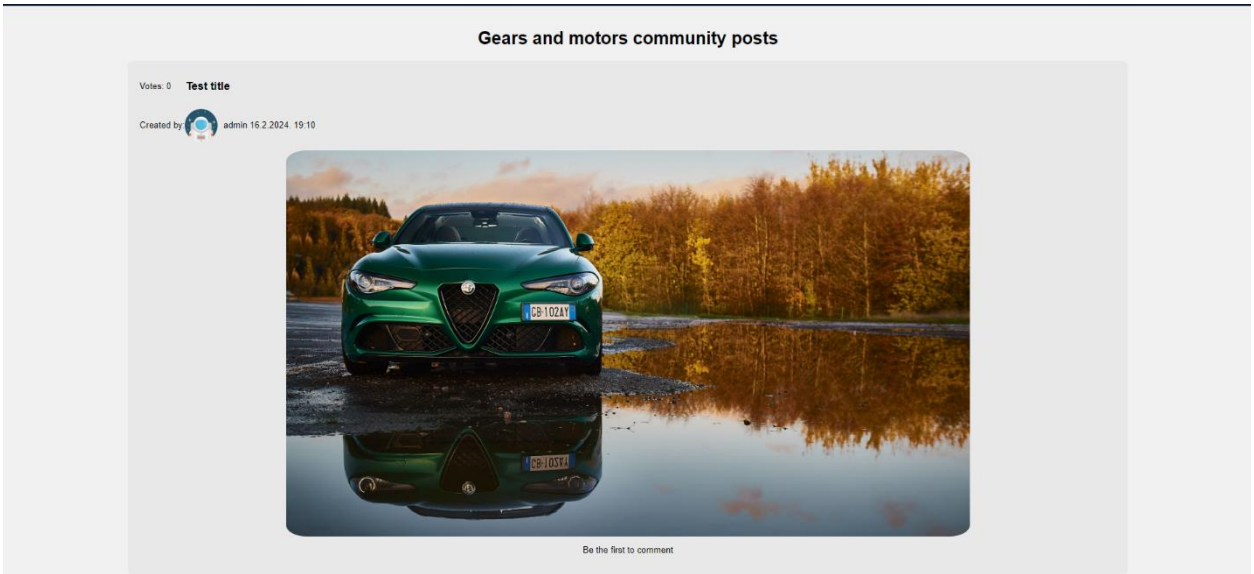
Gallery stranica sastoji se od responzivne galerije fotografija automobila. Dozvoljava sortiranje automobila po jednoj od 7 opcija *American*, *European*, *Asian*, *Cockpit*, *Engine bay*, *Paint job* te *Show all*. Odabirom pojedine opcije radi se filtriranje prikaza fotografija te su slike povezane s odabranim pojmom prikazane na korisniku. Zanimljivost galerije je što fotografije ispod sebe imaju hipervezu na odgovarajuću stranicu proizvođača automobila, gdje se mogu pročitati najnovije vijesti vezane baš uz tu marku automobila. Na slici 5.2. prikazana je stranica *Gallery*.



Slika 5.2. *Prikaz stranice Gallery.*

5.3 View posts

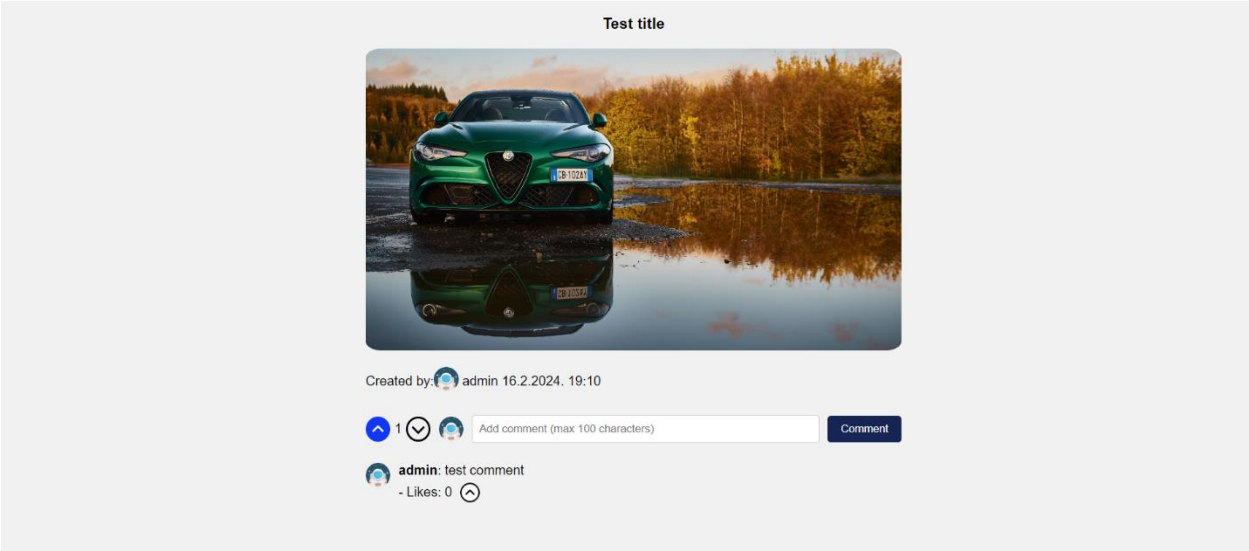
View posts stranica sastoji se od responzivne kolekcije objava registriranih korisnika. Svaka objava ima prikazan naslov, broj glasova, informacije o autoru, fotografiju objave te ovisno o postojanju komentara prikazuje tri najbolje ocijenjena komentara. Prijavljeni korisnik klikom na određenu objavu ima mogućnost odlaska na stranicu *View post* te pobliže vidjeti objavu, dok će neprijavljeni korisnik klikom na određenu objavu biti obavješten o tome kako je za daljnju interakciju potrebna prijava. Na slici 5.3. prikazana je stranica *View posts*.



Slika 5.3. *Prikaz stranice View posts.*

5.4. View post

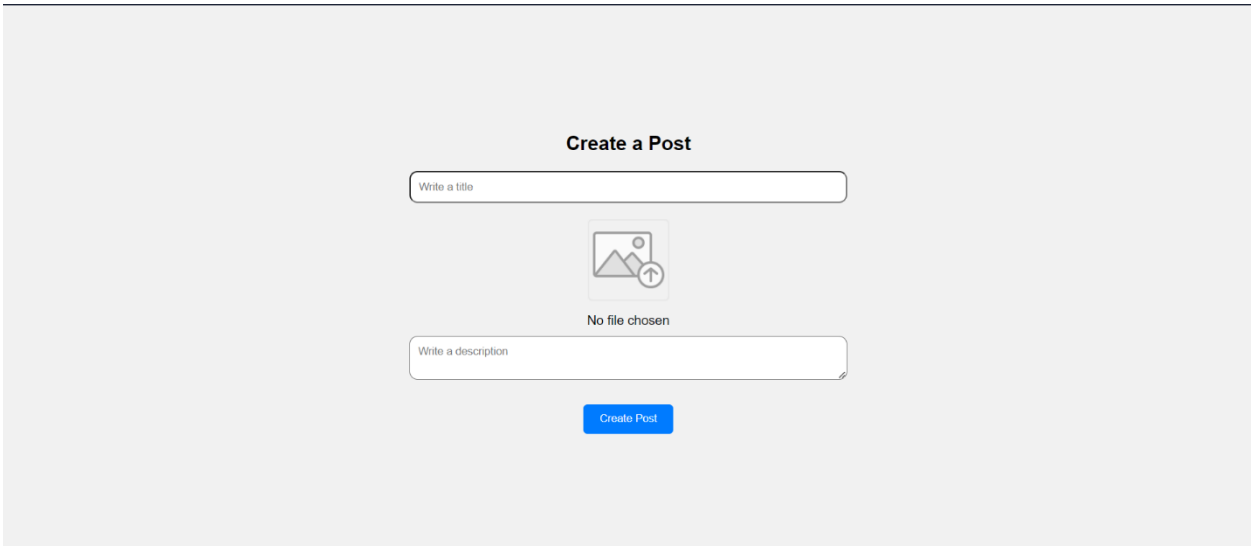
View post stranica sastoji se od responzivnog prikaza objave pojedinog korisnika. Prikaz ove strnaice moguć je samo prijavljenim korisnicima. Ulaskom na određenu objavu otvaraju se mogućnosti pobližeg upoznavanja objave te interakcija s istom. Korisniku je vidljiv naslov objave, fotografija, njezin opis, autor, vrijeme kreiranja, otvorene su mu opcije glasanja *upvote* i *downvote* te opcija komentiranja. Opcija *upvote* povećava broj glasova na objavi te postavlja pozitivan oblik povratne informacije autoru dok *downvote* predstavlja suprotno. Ukoliko za objavu već postoje komentari korisniku je ponuđena mogućnost davanja glasa komentarima koji se ističu malo više od drugih i zaslužuju biti pohvaljeni. Korisnik također može otvoriti i pregledati sliku koju je autor objavio u većem prikazu. Na slici 5.4. prikazana je stranica *View post*.



Slika 5.4. *Prikaz stranice View post.*

5.5 Create post

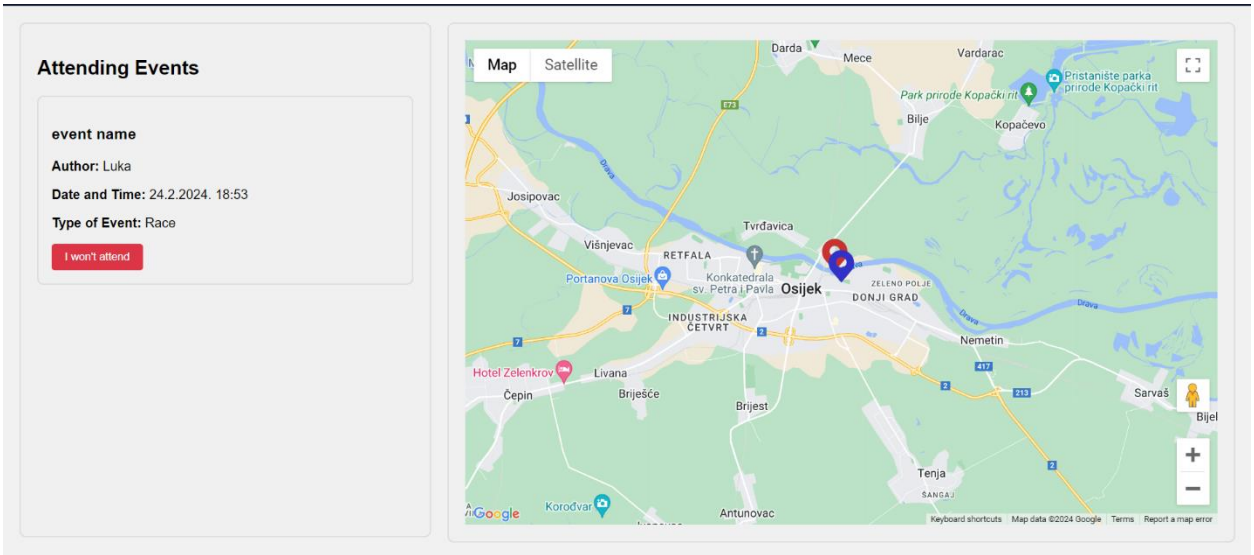
Create post stranica sastoji se od responzivnog prikaza obrasca koji je potrebno popuniti kako bi se mogla stvoriti objava. Sastoji se od polja za unos naslova, polja za unos fotografije te polja za unos opisa fotografije. Polja za unos naslova i fotografije su obavezna te ukoliko korisnik pokuša napraviti objavu bez unesene vrijednosti za jedno od ta dva polja, bit će obaviješten padajućom porukom sa vrha ekrana. Popunjavanjem obaveznih polja te neobaveznim popunjavanjem opisa slike, korisniku je omogućeno kreiranje objave klikom na gumb *Create Post*. Uspješnim stvaranjem objave korisnika se vodi na početnu stranicu gdje korisnik ulaskom na *View posts* može vidjeti svoju objavu. Na slici 5.5. prikazana je stranica *Create post*.



Slika 5.5. *Prikaz stranice Create post.*

5.6 View events

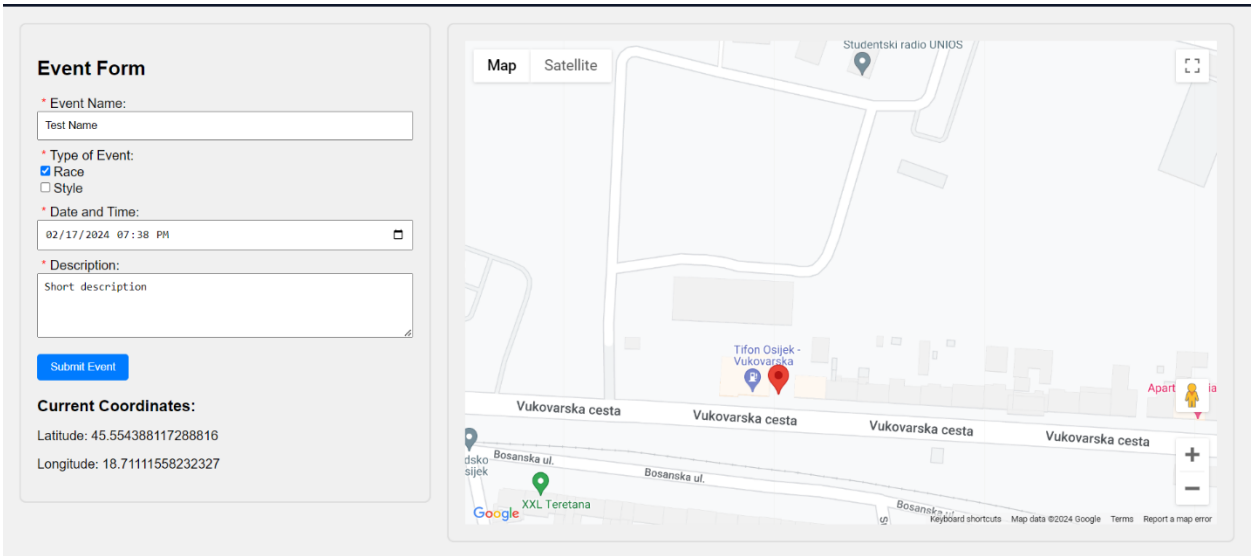
Stranica *View events* vidljiva je prijavljenim i neprijavljenim korisnicima. Stranica je responzivnog dizajna te se sastoji od dvije glavne sastavnice, *Attending events* sastavnice te Google karte na kojoj su vidljivi organizirani događaji sa određenom pribadačom za lokaciju na karti. Razlika u funkcionalnosti *View events* stranice prijavljenih i neprijavljenih korisnika je što sastavnica *Attending events* zahtjeva korisnika da bude prijavljen u aplikaciju. Prijavljeni korisnik može pratiti na koja događanja se prijavio te može pratiti vlastita događanja koja je organizirao. Na karti događanja postoje tri vrste pribadača za lokaciju koje predstavljaju događanja: crvena, plava i zelena. Crvena pribadača predstavlja događanja na koja se korisnik nije prijavio, plava predstavlja događanja na koja se korisnik prijavio dok zelena predstavlja događanja koja je korisnik organizirao. Pritiskom na određeno događanje prikazuju se osnovne informacije događanja naziv, tip, autor, datum i vrijeme te lokacija. Na slici 5.6. prikazana je stranica *View events*.



Slika 5.6. *Prikaz stranice View events.*

5.7. Organize event

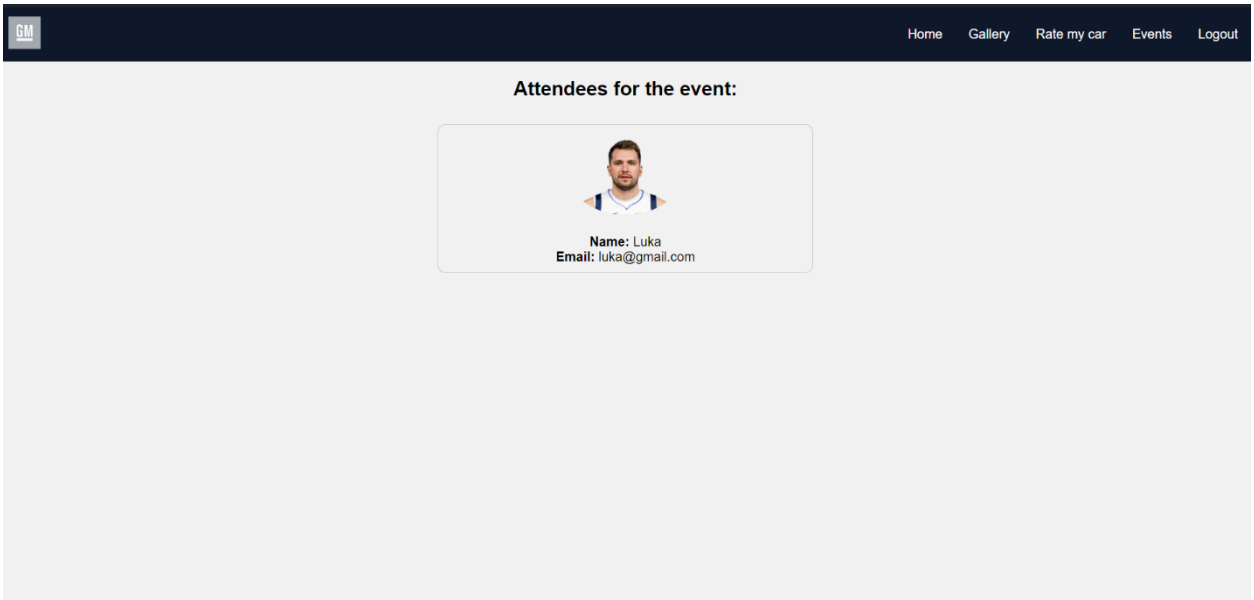
Organize event stranica vidljiva je samo prijavljenim korisnicima te se sastoji od obrasca i karte. Kako bi korisnik mogao organizirati događaj mora ispuniti obrazac sa sljedećim podatcima: ime događaja, tip događaja, vrijeme, kratki opis, te pritiskom na kartu korisnik postavlja mjesto lokacije događaja. Karta pruža satelitski način pregleda ili pregled u obliku karte puta. Ukoliko su uvjeti za organiziranje zadovoljeni korisniku se dopušta kreiranje događaja te je istog moguće vidjeti na stranici *View events*. Na slici 5.7. prikazana je stranica *Organize event*.



Slika 5.7. *Prikaz stranice Organize event.*

5.8. Event attendees

Klikom na pribadaču za lokacijom te odabirom na *View attendees* posjetitelji na zasebnoj stranici mogu dobiti informaciju o korisnicima koji su prijavili dolazak na događaj. Logika korištena za dohvaćanje polaznika na događanje prikazana je na slici 5.8.



Slika 5.8. *Prikaz stranice View attendees.*

6. ZAKLJUČAK

U okviru ovog završnog rada, napravljen je web portal za ljubitelje automobila koji nudi praćenje vijesti iz auto svijeta, postavljanje objava, interakcija s istima i postavljanje događaja. Potrebne su registracija i prijava kako bi korisnik imao sve mogućnosti aplikacije. Portal sadrži galeriju koja pruža novosti iz svijeta te je dostupna svim posjetiteljima stranice. Portal također sadržava kreiranje objava te kreiranje događaja. U današnje vrijeme broj ljubitelja automobila sve više raste, stoga je ovaj portal mjesto gdje će istomišljenici moći razmijeniti iskustva, te se sastati uživo na događajima koja su im u blizini. Već postoje mnoga rješenja koja nude organizaciju, komunikaciju i objavljivanje objava ljubitelja automobila, no ni jedno rješenje trenutno ne nudi sve mogućnosti zajedno. Način na koji bi se ova aplikacija unaprijedila bilo bi dodavanje načina za privatnu komunikaciju (engl. *private messaging system*).

LITERATURA

[1] Reddit, https://www.reddit.com , 09.06.2023.

[2] Car Throttle , https://www.carthrottle.com, 09.06.2023.

[3] Facebook- event, https://web.facebook.com/ccm1950/?_rdc=1&_rdr , 09.06.2023.

[4] Difference between frontend and backend, https://careerfoundry.com/en/blog/web-development/whats-the-difference-between-frontend-and-backend, 9.06.2023.

[5] HTML, https://www.javatpoint.com/what-is-html , 9.06.2023.

[6] CSS, https://www.w3schools.com/css/css_intro.asp, 9.06.2023.

[7] JavaScript, https://www.tutorialspoint.com/javascript/index.htm#, 9.06.2023.

[8] React, https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs , 9.06.2023.

[9] Firebase, https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0 , 9.06.2023.

[10] Visual Studio Code, https://www.infoworld.com/article/3666488/what-is-visual-studio-code-microsofts-extensible-code-editor.html, 12.06.2023.

[11] GitHub, https://github.com/topics/documentation, 12.06.2023.

SAŽETAK

Web portal zajednice ugađanja automobila pruža sučelje za pregled, objavljivanje, komentiranje i razmjenu mišljenja o ugođenim automobilima uz organiziranje događaja poput utrka i stilskih natjecanja. Web portal razvijen je koristeći većinski dvije tehnologije – React i Firebase. Uz pomoć Firebase Authentication alata, implementirane su funkcionalnosti registracije i prijave korisnika, no korisnik i u ulozi gosta ima mogućnost interakcije s ograničenim brojem funkcionalnosti.

Ključne riječi: baza podataka, društvena mreža, organizacija događaja, ugođeni automobili, web tehnologije

ABSTRACT

The Car Tuning Community Web Portal gives an interface for viewing, posting, commenting and exchange of opinions about tuned cars along with organizing events such as races or style competitions. The web portal was developed using mainly two technologies – React and Firebase. Helped along with the Firebase Authentication tool, registration and login functionalities have been implemented, however the user can interact with a limited number of functionalities in the guest role.

Key words: database, event organization, tuned cars, social network, web technologies

PRILOG

Poveznica na GitHub repozitorij: https://github.com/strmi8/gears-and-motors