IZRADA APLIKACIJE DRUŠTVENE MREŽE - APLIKACIJA YOUSTAR

Vlaić, Damjan

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Specijalistički diplomski stručni studij Primijenjeno računarstvo

DAMJAN VLAIĆ

Z A V R Š N I R A D

Izrada aplikacije društvene mreže - Aplikacija YouStar

Split, lipanj 2023.

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Specijalistički diplomski stručni studij Primijenjeno računarstvo

Predmet: Agilno vođenje projekata

Z A V R Š N I R A D

Kandidat: Damjan Vlaić

Naslov rada: Izrada aplikacije društvene mreže - Aplikacija YouStar

Mentor: mr. sc. Ivica Ružić, viši predavač

Split, lipanj 2023.

Sadržaj

Sažetak

U ovom završnom radu izrađena je društvena mreža (eng. *web*) aplikacija slična aplikaciji Instagram. Aplikacija omogućava korisnicima pretraživanje korisnika te uvid u njihove profile, dopisivanje između različitih korisnika te kreiranje ili komentiranje objava. Korisnik također može postaviti oznaku like na pojedinu objavu ili komentar te mijenjati određene detalje vlastitog korisničkog računa.

Aplikacija koristi *MVC* (*Model View Controller*) arhitekturu. Poslužiteljska strana (eng. *backend*) napravljena je u razvojnom okviru *Node.js*, a korisničko sučelje (eng. *frontend*) je izrađeno koristeći razvojni okvir React, *HTML* (*HyperText Markup Language*), *CSS* (*Cascading Style Sheets*) te biblioteku *Boostrap*. Za pohranu podataka korišten je sustav za upravljanje bazama podataka *MongoDB*. Kôd je napisan u integriranom razvojnom okruženju (eng. *Integrated Development Enviroment*) *Visual Studio Code*. Kôd aplikacije podijeljen je u dva dijela, jedan za poslužiteljsku, a drugi za korisničku stranu.

Ključne riječi: web aplikacija, društvena mreža, *Node.js*, *React*, *MongoDB*, *MVC*

Summary

Application YouStar

In this final work, the social media web application like Instagram has been built. The application allows users to search and view profiles of other users, exchange messages between them and to create or comment posts. The user can also like individual posts or comments and change different details of its user account.

The MVC architecture was used for the structure of application. Server side of application was made in Node.js framework, while the user interface (frontend) was made using frontend framework React, HTML (HyperText Markup Language), CSS (Cascading Style Sheets) and Bootstrap library. Microsoft SQL Server (the system used for managing databases) was used to store the data. The code is written in the integrated development environment Visual Studio Code. The application code is separated into two parts, one represents the server, and the other the client side.

Keywords: web application, social media, Node.js, React, MongoDB, MVC

1. Uvod

Ideja ovog završnog rada je izrada društvene web aplikacije koju će korisnici moći jednostavno koristiti. Prvi oblici društvenih mreža pojavljuju se 90.-ih godina 20. stoljeća te su tijekom godina nastajale različite implementacije istih. Usprkos tome, kod većine društvenih mreža, popularnost pojedinih korisnika se i dalje iskazuje putem broja korisnika koji ih prate te brojem like-ova na objavama. Društvena mreža YouStar nastoji uvesti drugačiji pristup koji će omogućiti korisnicima da svoju popularnost iskazuju i brojem zvjezdica koji predstavlja ukupan broj like-ova na svim korisnikovim objavama.

Razvoj modernih tehnologija kao što su JavaScript, React, i CSS, omogućio je programerima da razviju suvremene web aplikacije. Razvojno okruženje React omogućuje da se pojedini dijelovi mogu koristiti više puta unutar aplikacije. Time se kôd piše brže te je kao takav čitljiviji drugim programerima. Na pozadinskom servisu aplikacije biblioteke poput Node.js-a i Express-a omogućuju jednostavno kreiranje servera te povezivanjem s istim. Također kreiranje podataka, manipuliranje istima te njihovo spremanje jako je olakšano s programima i paketima kao što je MongoDB. CSS je omogućio stiliziranje i animiranje apsolutno svakog elementa HTML kôda, od jednostavnog klizećeg teksta do animacija koje su prije bile nezamislive za jednu web stranicu.

Nakon uvoda, u drugom poglavlju su opisane korištene tehnologije. U trećem poglavlju su detaljno prikazane funkcionalnosti aplikacije kao i sama izrada, a u posljednjem poglavlju je dan zaključak.

2. Korištene tehnologije

Aplikacija je podijeljena u dva dijela. Serverski dio je izrađen u Node.js razvojnom okviru, dok je klijentska strana izrađena koristeći React, razvojni okvir za izradu korisničkog sučelja. Oba razvojna okvira kao podlogu koriste Javascript programski jezik. Node.js je objavljen 2009. godine kako bi se Javascript programski jezik mogao koristiti i na poslužiteljskoj strani. React biblioteka objavljena je 2013. godine i jedna je od najpopularnijih razvojnih komponenti za izradu korisničkog sučelja.

Objekti objava i ostalih klasa spremljeni su u bazu podataka MongoDB. Serverska strana komunicira s bazom podataka kako bi prikupljene informacije prenijela na klijentsku stranu.

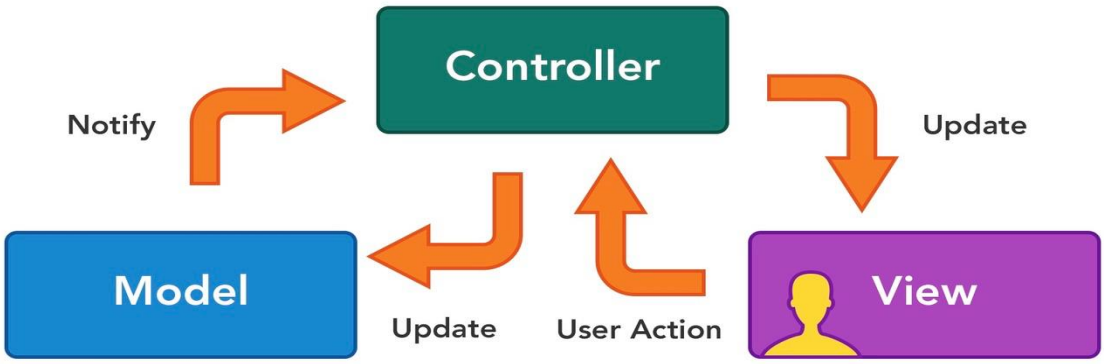
Klijentska strana napravljena je koristeći Javascript, HTML i CSS, uvelike se oslanjajući na Bootstrap koji aplikaciji daje responzivan dizajn.

2.1. Node.js

Node.js je razvojni okvir koji se koristi za izradu mrežnih aplikacija sa mogućnošću da se dinamička web stranica izradi u potpunosti prije nego što je poslana korisniku. Omogućava programerima jednostavno prosljeđivanje podataka između klijentske i poslužiteljske strane. Kao podlogu koristi programski jezik JavaScript.

Nadalje, Node.js pruža programeru veliki broj već napisanih klasa i biblioteka koje znatno olakšavaju izradu mrežne aplikacije. U slučaju da pojedina biblioteka nije po standardu uključena u projekt, moguće je jednostavno uključiti dodatne biblioteke ili klase, pomoću „npm install {ime paketa}“ naredbe.

Model klase imaju ulogu mapirati podatke iz stvarnog svijeta koji su potrebni za rad aplikacije. U slučaju aplikacije YouStar, to su na primjer objave koje su predstavljene klasom *Post*. Kontroler služi kao posrednik između poslužiteljske i klijentske strane aplikacije. Zaslužan je za prijenos podataka koji će se HTML dokument prikazati nakon klika na određenu poveznicu te slanje podataka iz baze na grafičko sučelje.



Slika 1: *Prikaz MVC model arhitekture*

Klijentska strana se odnosi na sučelje pomoću kojega korisnik ostvaruje interakciju s aplikacijom. React pomoću komponenti prikazuje određene dijelove korisničkog sučelja pružajući programeru mogućnost pisanja JavaScript kôda unutar HTML sadržaja. Dizajn MVC model arhitekture prikazan je na slici 1.

2.2. React

React.js je biblioteka otvorenog kôda (engl. open source library) temeljena na programskom jeziku JavaScript. Biblioteku je kreirala korporacija Facebook. Koristi se za brzu i učinkovitu izgradnju interaktivnih korisničkih sučelja i web aplikacija. U React-u aplikacije se razvijaju stvaranjem komponenti za višekratnu upotrebu koje se mogu zamisliti kao neovisni blokovi. Komponente čine pojedinačne dijelove vizualnog sučelja koje kada se sastave čine cjelokupno korisničko sučelje aplikacije. Primarna uloga Reacta je što učinkovitije upravljati prikazom, učitavajući komponente na što efikasniji način. React to postiže manipulacijom DOM elemenata pomoću Javascripta te se tako pomoću njega rade dinamične i responzivne web aplikacije.

2.3. MongoDB

Aplikacija koristi MongoDB sustav za upravljanje bazama podataka. Mongoose je objektno orijentirana javascript biblioteka pomoću koje se ostvaruje konekcija između serverske strane i baze podataka. Mongoose pruža potpuni skup operacija nad svim podatcima u bazi (stvaranje, čitanje, mijenjanje i brisanje) te tablice u bazi stvara na temelju mongoose shema u kôdu. Takav pristup naziva se *code-first*.

3. Funkcionalnosti aplikacije

U ovom poglavlju opisan je praktični dio rada. Pojašnjava se za što je zadužen klijentski, a za što poslužiteljski dio aplikacije te je istaknuta razlika u mogućnostima koje korisnici imaju ovisno u ulozi.

3.1. Poslužiteljski dio aplikacije

Poslužiteljski dio aplikacije (engl. *backend*) je serverska strana web aplikacije. Koristi se za pohranu, raspoređivanje podataka te osigurava da klijentska strana ima pristup podatcima u bazi podataka. Korisnici neizravno pristupaju dijelovima serverskog kôda pomoću akcija na korisničkom sučelju.

3.1.1. Navigacija

Navigacijska traka se nalazi na vrhu stranice. Izgled trake kada korisnik nije prijavljen vidljiv je na slici 2.

Slika 2: *Izgled navigacijske trake kada korisnik nije prijavljen*

Poveznica *YouStar* korisnika preusmjerava na stranicu s objavama svih korisnika koje prijavljeni korisnik prati. U desnom dijelu trake nalaze se poveznice *Signup* i *Signin* koje se pojavljuju samo dok korisnik nije prijavljen. Klikom na poveznicu *Signup* korisnika se usmjerava na formu, kako bi unošenjem svojih podataka napravio novi račun. Sukladno tome, *Signin* poveznica služi kako bi se korisnik koji već ima račun, mogao prijaviti u sustav sa svojim podatcima.

Slika 3: *Izgled navigacijske trake dok je korisnik prijavljen*

Ako je korisnik prijavljen, na traci prikazanoj na slici 3. pojavljuje se poveznica *Chats* putem koje se mogu vidjeti svi razgovori prijavljenog korisnika s drugim korisnicima te poveznica *Saved Posts* putem koje korisnik može vidjeti spremljene objave. Poveznica *Settings* korisnika vodi na formu preko koje se mogu mijenjati pojedini podatci korisnikovog računa. Uz to se i poveznice *Signup* i *Signin* mijenjaju s poveznicom *Logout* pomoću koje se korisnik može odjaviti iz sustava.

Dio navigacije izdvojen je unutar podnožja stranice (*engl. footer*). Ako korisnik nije prijavljen, podnožje samo sadrži dugme putem kojega se mogu pretraživati korisnici aplikacije što je vidljivo na slici 4.

Slika 4: *Prikaz podnožja dok korisnik nije prijavljen*

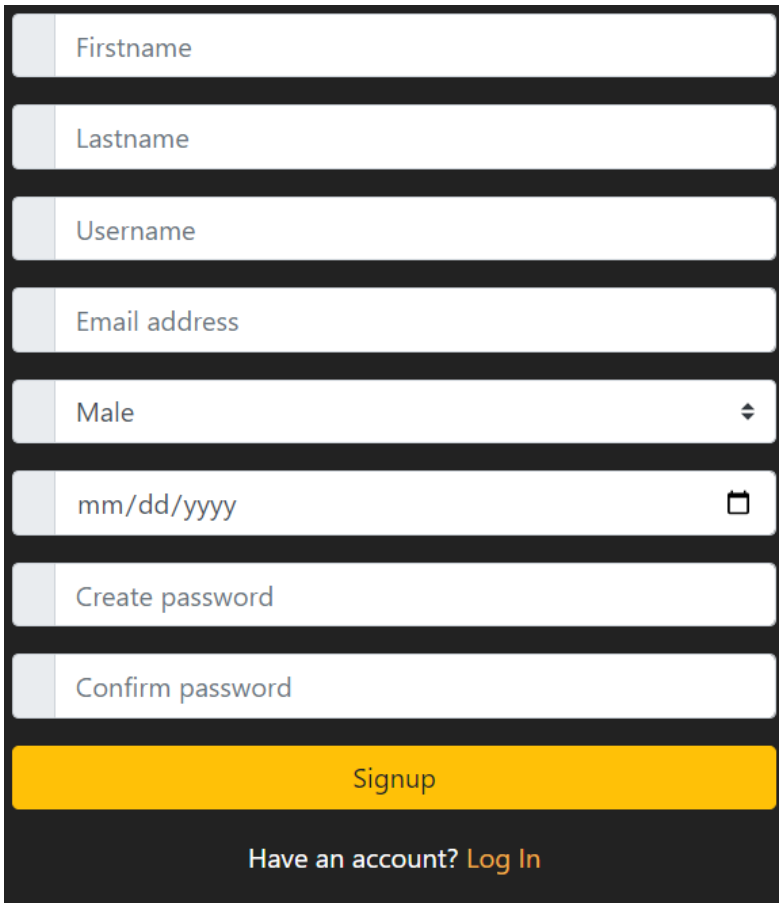
Ako je korisnik prijavljen, podnožje aplikacije dobiva 2 dodatna dugmeta kao što prikazuje slika 5.

Jedan služi kao poveznica preko koje korisnik može pristupiti svom profilu, a drugo dugme vodi na formu pomoću koje se mogu objaviti nove objave.

Slika 5: *Prikaz podnožja dok je korisnik prijavljen*

3.1.2. Autentikacija

Kako bi korisnik napravio novi račun u aplikaciji, koristi postupak registracije. Forma za registraciju sadrži atribute: ime, prezime, korisničko ime, adresa elektroničke pošte, spol, datum rođenja i šifru što je vidljivo na slici 6.



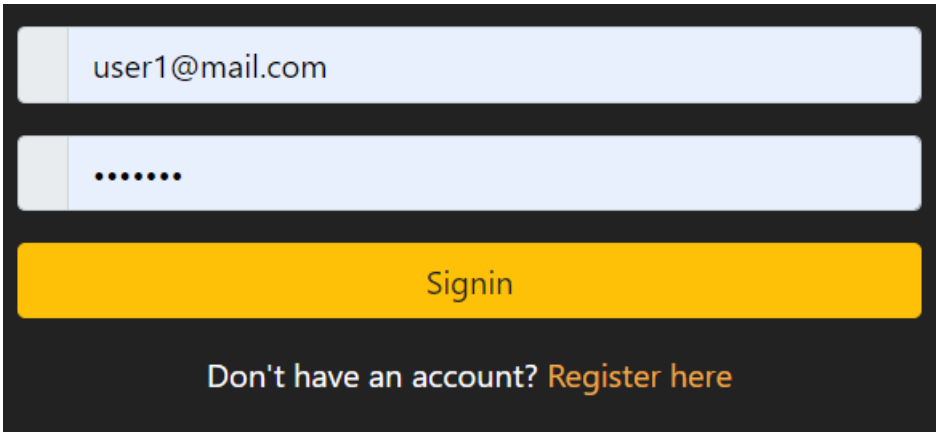
Slika 6: *Izgled forme za registraciju*

Nakon klika na dugme *Signup*, popunjeni podaci o korisniku se šalju *signupConroller* metodi unutar *auth* kontrolera kako bi se kreirao novi *User* objekt. Kako bi se izbjegla situacija da se slučajno kreiraju 2 korisnika s istom adreesom elektroničke pošte, prvo je potrebno provjeriti da li se u bazi već nalazi takav korisnik. Ako je nova adresa elektroničke pošte jedinstvena, šifra korisnika se enkriptira uz *salt*, podaci o novom korisniku se pohranjuju u bazu čime je postupak registracije gotov što je vidljivo iz ispisa 1.

```
exports.signupController = async (req, res) => {
  const { firstname, lastname, username, email, birthday, gender,
password } = req.body;
  try {
    const user = await User.findOne({ email });
    if (user) {
      return res.status(400).json({
        errorMessage: "Email already exists",
      });
    }
    const newUser = new User();
    newUser.firstname = firstname;
    newUser.lastname = lastname;
    newUser.username = username;
    newUser.email = email;
    newUser.birthday = birthday;
    newUser.isBanned = false;
    newUser.status = "";
    newUser.gender = gender;
    const salt = await bcrypt.genSalt(10);
    newUser.password = await bcrypt.hash(password, salt);
    await newUser.save();
    res.json({ successMessage: "Registration success. Please signin." });
```

Ispis 1: *Prikaz signupController funkcije*

Ako korisnik nije prijavljen, klikom na poveznicu *Signin* u navigacijskoj traci, otvara se forma prikazana na slici 7, putem koje se korisnik prijavljuje u sustav s postojećim računom.



Slika 7: *Signin forma*

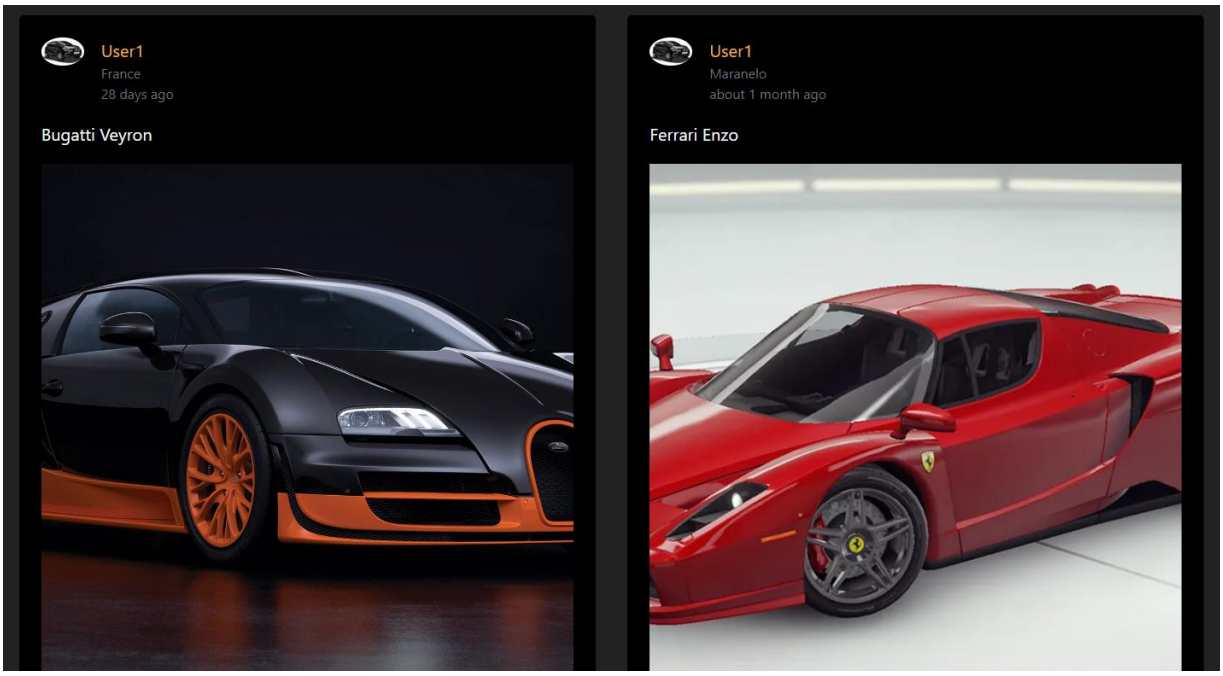
Klikom na dugme *Signin*, *signinController* funkciji *auth* kontrolera se šalju korisnička adresa elektroničke pošte i šifra. Prvo se provjerava da li u bazi podataka već trenutno postoji korisnik s tom adresom elektroničke pošte. Ako je pronađen takav korisnik, proslijeđena šifra se uspoređuje sa šifrom dohvaćenog korisnika. Ako se šifre podudaraju korisnik se uspješno prijavljuje u sustav te se unutar lokalne memorije web preglednika (*engl. local storage*) pohranjuju određeni podaci o prijavljenom korisniku. Također se korisniku dodjeljuje kolačić (*engl. cookie*) koji sadrži token, odnosno vrijeme koje definira koliko dugo korisnik može ostati prijavljen u aplikaciji prije nego što ga sustav automatski odjavi. Odjavom iz sustava brišu se kolačići i podaci u lokalnoj memoriji web preglednika. Prikaz *signinController* funkcije vidljiv je u ispisu 2.

```
exports.signinController = async (req, res) => {
  const { email, password } = req.body;
  try {
    const user = await User.findOne({ email });
    if (!user) {
      return res.status(400).json({
        errorMessage: "Invalid credentials",
      });
    }
    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) {
      return res.status(400).json({
        errorMessage: "Invalid credentials",
      });
    }
    const payload = {
      user: {
        _id: user._id,
      },
    };
    jwt.sign(payload, jwtSecret, { expiresIn: jwtExpire }, (err, token)
=> {
      const { _id, username, email, role } = user;
      res.json({
        token,
        user: { _id, username, email, role },
      });
    });
```

Ispis 2: *Prikaz signinController funkcije*

3.1.3. Izvor vijesti

Na početnoj stranici aplikaciju vidljivoj na slici 8, prikazana je lista objava svih korisnika koje trenutno prijavljeni korisnik prati sortirani od najranije prema kasnijoj. Objave su prikazane unutar *Home* komponente i definirane su klasom *Post*.



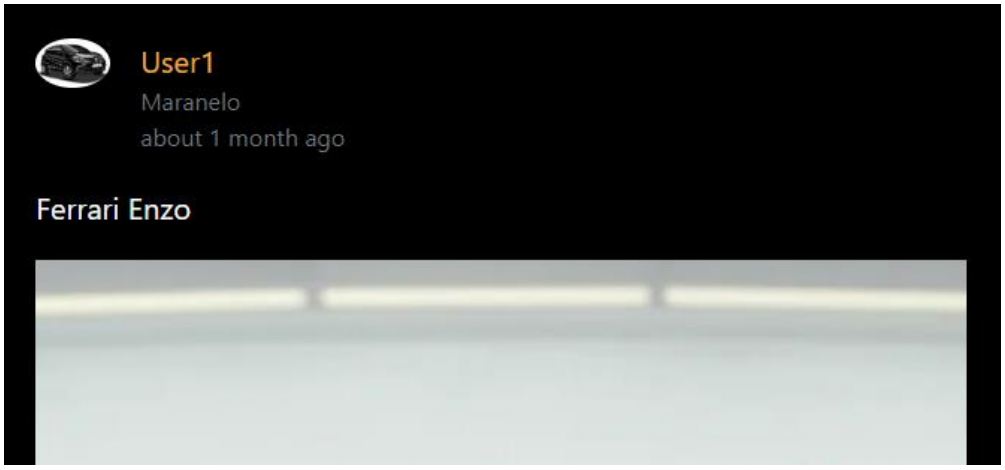
Slika 8: *Izgled liste filmova dok korisnik nije prijavljen*

Klasa *Post* sadrži polja: *stars*, *likers*, *comments*, *description*, *image*, *location*, *creatorUsername*, *creator* i autogenerirana Mongoose *polja _id, createdAt, i updatedAt*. Deklaracija klase *Post* prikazana je u ispisu 3.

```
const postSchema = new Schema(
  {
    description: { type: String, required: true },
    image: { type: String, required: true },
    location: { type: String, required: false },
    creatorUsername: { type: String, required: false },
    stars: { type: Number, default: 0 },
    creator: { type: Schema.Types.ObjectId, ref: "User", required: true
},
    likers: [
      {
        type: Schema.Types.ObjectId,
        ref: "User",
      },
    ],
    comments: [
      {
        type: Schema.Types.ObjectId,
        ref: "Comment",
      },
    ],
  },
  { timestamps: true }
);
```

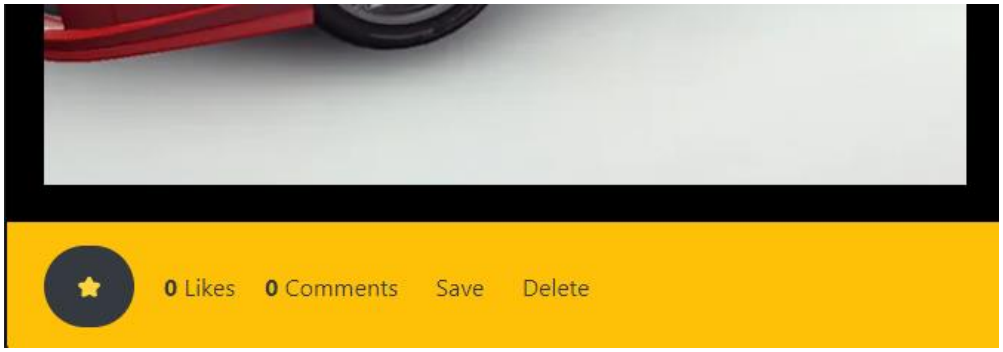
Ispis 3: *Klasa Post*

U zaglavlju kartice svake prikazane objave prikazane su informacije kao što su: profilna slika vlasnika objave, vlasnikovo korisničko ime, lokacija, opis i vrijeme od kada je objava objavljena. Klikom na profilnu sliku ili korisničko ime, prijavljenog korisnika se usmjerava na profilnu stranicu vlasnika objave. Izgled zaglavlja prikazan je na slici 9.



Slika 9: *Prikaz zaglavlja objave*

Ispod zaglavlja nalazi se sama slika objave te ispod nje dugmeta pomoću kojih korisnik može ostvariti interakciju s objavom. Izgled dna objave prikazan je na slici 10.



Slika 10: *Prikaz dugmeta u dnu objave*

Klikom na dugme sa zvjezdicom, broj likeova (zvjezdica) na toj objavi se povećava za jedan ako prijavljeni korisnik već nije like-ao tu objavu. Ako je objava već like-ana od strane prijavljenog korisnika, broj like-ova na toj objavi će se prilikom klika smanjiti. Procedura je ostvarena tako da se klikom na dugme sa zvjezdicom poslužiteljskoj strani šalju id like-ane objave i id korisnika koji ju je like-ao. *Post controller* na poslužiteljskoj strani dohvaća te informacije preko *likePost* rute te provjerava da li se id korisnika nalazi u listi korisnika koji su like-ali objavu. Ako se nalazi korisnik se briše iz liste, u suprotnom se dodaje što je vidljivo u ispisu 4.

```
exports.likePost = async (req, res) => {
  let isLiked = true;
  const postId = req.params.postId;
  const likerId = req.params.likerId;
  const post = await Post.findById(postId);
  const postOwnerId = post.creator;
  const postOwner = await User.findById(postOwnerId);

  for (let i = 0; i < post.likers.length; i++) {
    if (post.likers[i] == likerId) {
      post.likers.remove(post.likers[i]);
      isLiked = false;
      break;
    }
  }

  if (isLiked) {
    post.stars += 1;
    post.likers.push(likerId);
    postOwner.stars += 1;
  } else {
    post.stars -= 1;
    postOwner.stars -= 1;
  }

  await post.save();
  await postOwner.save();

  res.json({
    successMessage: "Post liked.",
  });
```

Ispis 4: *Prikaz likePost rute*

3.1.4. Komentiranje objava

Dugme *Comments* korisnika preusmjerava na listu svih komentara koji su objavljeni pod tom objavom. Iznad liste komentara nalazi se forma pomoću koje korisnik može objaviti novi komentar što je vidljivo na slici 11.

Slika 11: *Izgled liste komentara*

Komentari su definirani klasom *Comment* koja sadrži polja: *text*, *stars*, *owner*, *comments*, *likers* i autogenerirana *Mongoose* polja *_id*, *createdAt*, *updatedAt*. Prikaz klase *Comment* prikazan je u ispisu 5.

```
const schema = new Schema(
  {
    text: { type: String, required: true },
    stars: { type: Number, required: false },
    owner: { type: Schema.Types.ObjectId, ref: "User" },
    comments: [
      {
        type: Schema.Types.ObjectId,
        ref: "Comment",
      },
    ],
    likers: [
      {
        type: Schema.Types.ObjectId,
        ref: "User",
      },
    ],
  },
  { timestamps: true }
);
```

Ispis 5: *Prikaz Comment klase*

Upisom teksta u formu i klikom na dugme *Post*, šalje se *post* zahtjev na poslužiteljsku stranu s informacijama o sadržaju komentara, objavitelju i objavi na kojoj će komentar biti objavljen. Poslužiteljska strana dohvaća zahtjev pomoću *postComment* funkcije kojom dodaje objavljeni komentar u listu komentara za tu objavu. Komentar se stvara s 0 zvjezdica te je svakom korisniku dopušteno samo jedanput like-ati komentar. Ako komentar pripada prijavljenom

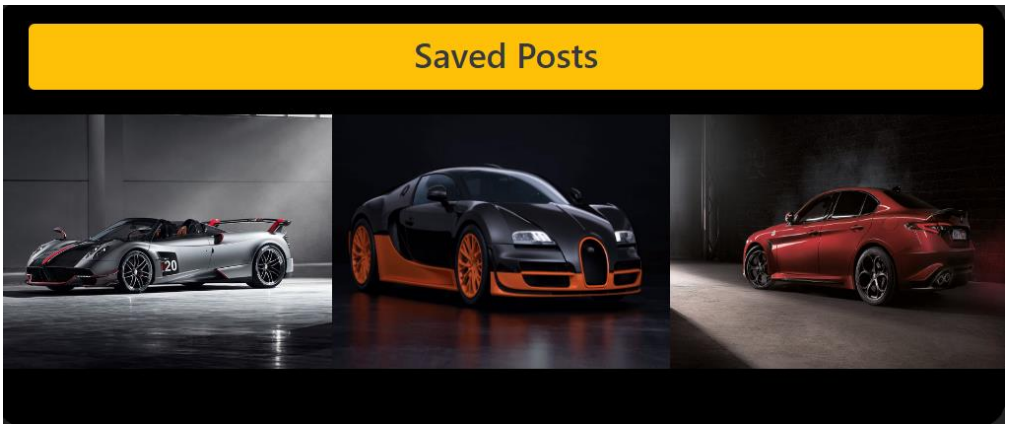
korisniku, na komentaru će se prikazati *Delete* dugme preko kojega se komentar može izbrisati. Prikaz *postComment* funkcije prikazan je u ispisu 6.

```
exports.postComment = async (req, res) => {
  try {
    const postId = req.params.postId;
    const commenterId = req.params.userId;

    const post = await Post.findById(postId);
    const commenter = await User.findById(commenterId);
    const { comment } = req.body;

    let newComment = new Comment();
    newComment.text = comment;
    newComment.stars = 0;
    newComment.owner = commenter;
    await newComment.save();

    post.comments.unshift(newComment);

    await post.save();

    res.status(200).json({
      successMessage: `Posted successfully!`,
    });
```

Ispis 6: *prikaz postComment funkcije*

3.1.5. Spremanje objava

Dugme *Save* omogućava da korisnik pohrani objavu kako bi joj u budućnosti mogao lakše pristupiti. Klikom na dugme *Saved Posts* unutar navigacijske trake moguće je vidjeti sve objave koje je prijavljeni korisnik dosad spremio. Izgled liste spremljenih objava prikazan je na slici 12.

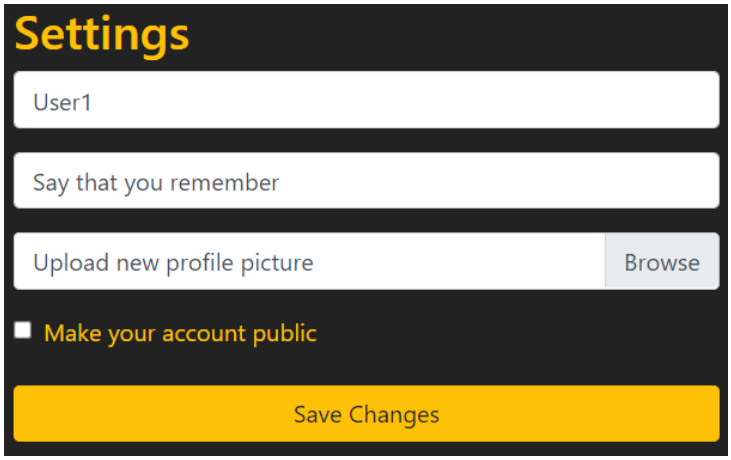


Slika 12: *Prikaz spremljenih objava*

Ako je objava već spremljena, uz objavu se umjesto dugmeta *Save* pojavljuje dugme *Unsave* preko kojega se objava može izbaciti iz liste spremljenih.

3.1.6. Postavke

Poveznica *Settings* korisnika preusmjerava na formu preko koje korisnik može personalizirati podatke o svom profilu. Moguće je mijenjati korisničko ime, status, sliku profila ili pristupačnost profila čineći ga privatnim ili javnim za korisnika koji ne prate prijavljenog korisnika.Prikaz forme za promjenu podataka o korisniku vidljiv je na slici 13.



Slika 13: *Prikaz forme sa postavkama profila*

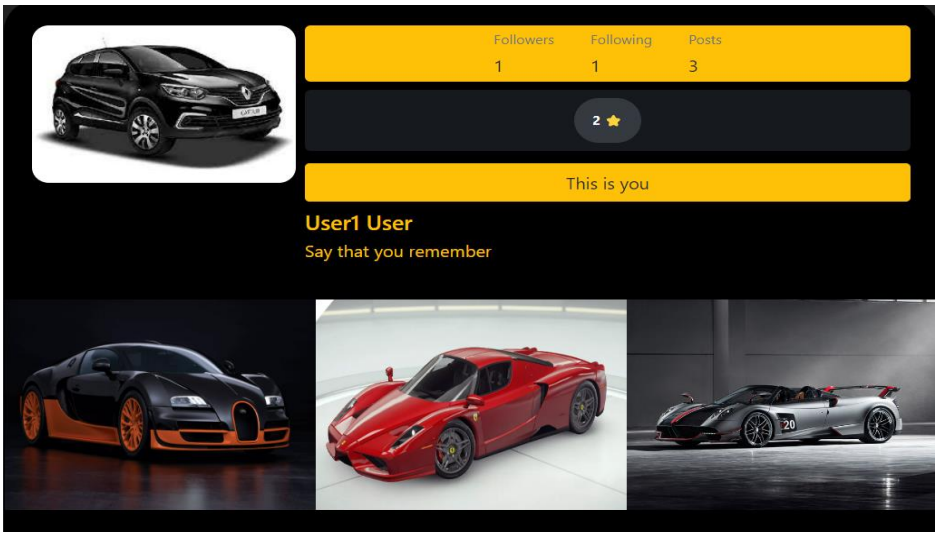
Settings komponenta preko id-a prijavljenog korisnika dohvaća njegova podatke kako bi popunila formu s njegovim podatcima kako je prikazano u ispisu 7.

```
const Settings = (props) => {
  const userId = props.match.params.userId;
  const [user, setUser] = useState({});
  const [username, setUsername] = useState("");
  const [status, setStatus] = useState("");
  const [isAccountPublic, setIsAccountPublic] = useState(false);
  const [profileImage, setProfileImage] = useState(null);

  useEffect(() => {
    getUser(userId).then((response) => {
      setUser(response.data);
      setUsername(response.data.username);
      setStatus(response.data.status);
      setIsAccountPublic(response.data.isAccountPublic);
    });
  }, [userId]);
```

Ispis 7: *Prikaz popunjavanja forme korisnikovim podatcima*

3.1.7. Korisnički profil

Informacije o pojedinom korisniku mogu se vidjeti unutar *UserDetails* komponente. Na korisnikovom profilu prikazane su informacije: ime i prezime korisnika, korisničko ime, profilna slika, broj zvjezdica, broj pratitelja i praćenih korisnika, slike objava i njihov ukupan broj.



Slika 14: *Prikaz izgleda korisničkog profila*

Slike objava se nalaze u dnu korisničkog profila i sortirane su od najnovije prema najstarijoj ovisno o vremenu objavljivanja. Detalje o pojedinoj objavi moguće je vidjeti klikom na sliku. Redak ispod broja zvjezdica, prikazan na slici 15, pruža prijavljenom korisniku određene akcije nad drugim korisnicima. Unutar retka mogu biti vidljiva 2 ili 3 dugmeta ovisno o vrsti korisnika čiji se profil promatra. Dva dugmeta su uvijek *Chat* (preko kojega se može pristupiti razgovoru s korisnikom) i *Follow* ili *Unfollow* ovisno o tome da li prijavljeni korisnik prati korisnika čiji je profil prikazan.

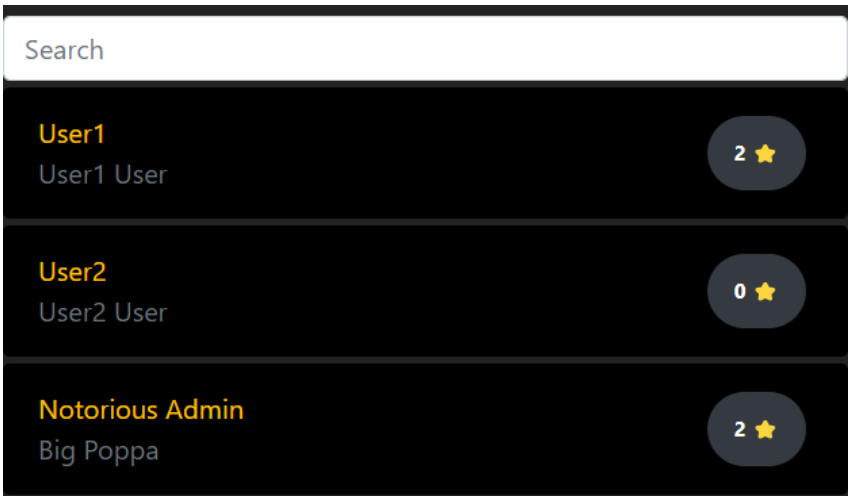
Slika 15: *Prikaz retka dok korisnik promatra tuđi profil*

Treće dugme s natpisom *Ban*, prikazan slikom 16,pojavljuje se ako je administrator prijavljen u sustav te omogućuje da se određenom korisniku onemogući bilo kakvo kreiranje novog sadržaja.

Slika 16: *Prikaz retka dok je prijavljen administrator*

3.1.8. Pretraživanje korisnika

Do liste svih korisnika dolazi se putem *Search* dugmeta u dnu stranice. Svaki korisnik u listi prikazan je s njegovim korisničkim imenom, imenom, prezimenom i brojem zvjezdica te se klikom na pojedini redak otvara profil odgovarajućeg korisnika. Iznad liste nalazi se forma putem koje se mogu filtrirati željeni korisnici.

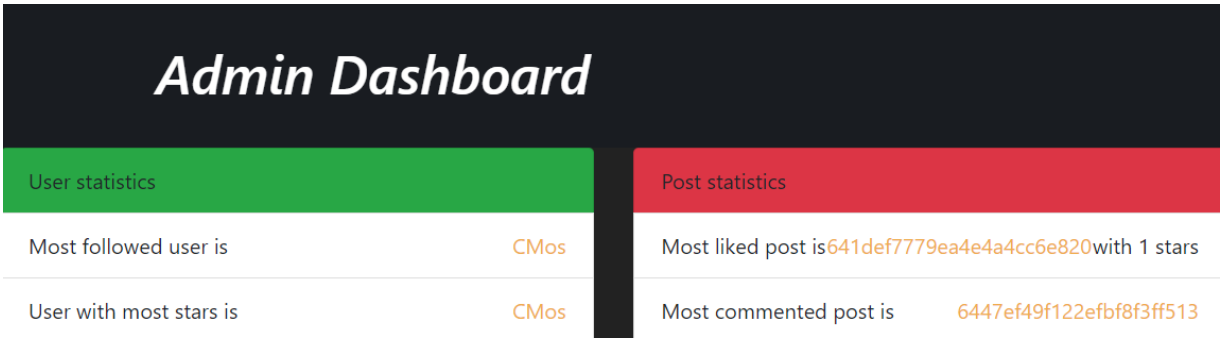


Slika 17: *prikaz liste korisnika*

Filtriranje funkcionira tako da se upisom svakog novog slova u formu izdvajaju oni korisnici čije korisničko ime sadrži to slovo bez obzira na kapitalizaciju. Listi se također može pristupiti preko poveznica *Followers* i *Following* unutar profila korisnika. Preko *Followers* poveznice u listu korisnika se učitavaju samo oni korisnici koji prate tog korisnika, dok se preko *Following* poveznice u listi prikazuju korisnici koje taj korisnik prati.

3.1.9. Administratorsko sučelje

Administrator je korisnik kojega se u bazi podataka može prepoznati po tome što mu je polje *role* postavljeno na vrijednost 1 umjesto 0 kao kod običnih korisnika. Razlika postoji i u tome da administrator ima pristup administratorskom sučelju kojemu se može pristupiti putem poveznice *Dashboard* u navigacijskoj traci. Sučelje prikazuje različite statistike o korisnicima i objavama: najpraćeniji korisnik, korisnik i objava s najviše zvjezdica te najkomentiranija objava. Izgled administratorskog sučelja prikazan je na slici 18.



Slika 18: *Prikaz administratoskog sučelja*

Svaka od prikazanih statistika sadrži poveznicu preko koje se mogu vidjeti detalji o njenom rezultatu. Dodatne mogućnosti administratora mogu se također vidjeti i kroz mogućnost obavljanja akcija koje nemaju obični korisnici: mogućnost brisanja tuđih objava i blokiranja korisničkih računa. Korisnik s blokiranim računom ne može komentirati objave, kreirati novu objavu, mijenjati osobne podatke preko postavki. Također, blokiranom korisniku i ostalim korisnicima nije dozvoljeno pregledavanje vlastitih objava.

3.1.10. Dopisivanje

Aplikacija pruža privatnu komunikaciju između 2 korisnika putem sustava za dopisivanje (*engl. chat*) čije je grafičko sučelje prikazano na slici 19. Ako se prijavljeni korisnik i korisnik kojemu se želi poslati poruka dosad nikada nisu dopisivali, potrebno je prvo pristupiti

korisnikovu profilu i kliknuti na *Chat* dugme preko kojega otvara sučelje gdje su zabilježene sve poruke od korisnika koji sudjeluju u komunikaciji.

Slika 19: *Prikaz korisničkog sučelja sa praznim razgovorom*

Slanje poruke obavlja se tako da se klikom na strelicu u donjem desnom kutu kreira novi objekt klase *Message*. Klasa *Message* sadrži podatke o tome tko je poslao poruku (*sender*), sadržaj poruke (*message*) i vrijeme slanja poruke (*creationTime*). Prikaz *Message* klase sadržan je u ispisu 8.

```
const MessageSchema = new Schema(
  {
    sender: { type: Schema.Types.ObjectId, ref: "User", required: true },
    message: { type: String, required: true },
    creationTime: { type: String, required: true },
  },
  { timestamps: true }
);
```

Ispis 8: *Prikaz Message klase*

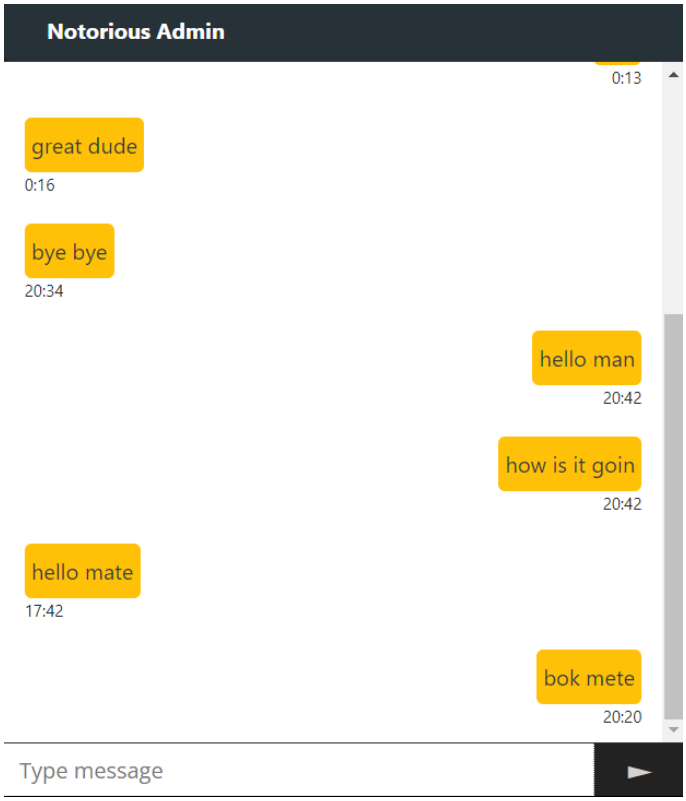
Sučelje za razgovor u kojemu nema poruka, nema pridodijeljen vlastiti *Chat* objekt. Uloga *Chat* klase je okupiti poruke između dva korisnika na jednom mjestu. *Chat* objekt sadrži podatke o

sudionicima u razgovoru (*member1* i *member2*) te listu u koju se spremaju pristigle poruke za taj razgovor (*messages*).

```
const ChatSchema = Schema(
  {
    member1: { type: Schema.Types.ObjectId, ref: "User", required: true
},
    member2: { type: Schema.Types.ObjectId, ref: "User", required: true
},
    messages: [
      {
        type: Schema.Types.ObjectId,
        ref: "Message",
      },
```

Ispis 9: *Prikaz Chat klase*

Prilikom slanja prve poruke u razgovoru, najprije se kreira novi *Chat* objekt sa *member1* poljem postavljenim na id korisnika koji je poslao poruku, *member2* poljem sadrži id korisnika kojemu je poruka upućena te praznom listom poruka. Zatim se kreira pristigla poruka i njezin id se sprema unutar *messages* liste novokreiranog razgovora čime se poruka pojavljuje unutar korisničkog sučelja. Poruke prijavljenog korisnika prikazana su s desne, dok su poruke od suprotnog korisnika prikazane s lijeve strane korisničkog sučelja što je prikazano na slici 20.



Slika 20: *Prikaz razgovora sa porukama*

Svi razgovori u kojima je prijavljeni korisnik sudjelovao mogu se vidjeti preko poveznice *Chats* unutar navigacijske trake. Razgovori su prikazani kao lista čiji svaki element na sebi ima napisano ime korisnika s kojim prijavljeni korisnik dijeli razgovor.To je vidljivo na slici 21.

Slika 21*: Prikaz liste razgovora*

Klikom na pojedini element otvara se razgovor s odgovarajućim korisnikom. Elementi liste su dobiveni učitavanjem svakog *Chat* objekta čiji id se nalazi unutar liste *chats* prijavljenog korisnika što se može zaključiti iz ispisa 10.

```
const userId = req.params.userId;
const allChats = await Chat.find({});
const promise = allChats.map(async (cId) => {
  const chatObject = await Chat.findById(cId);
  if (chatObject.member1 == userId || chatObject.member2 == userId) {
    return {
      chatObj: chatObject,
      member1Obj: await User.findById(chatObject.member1),
      member2Obj: await User.findById(chatObject.member2),
    };
  }
});
const chats = await Promise.all(promise);
let chatsWithoutUndefinedObjects = [];
chats.forEach((c) => {
  if (c != undefined) {
    chatsWithoutUndefinedObjects.push(c);
  }
});
res.json(chatsWithoutUndefinedObjects);
```

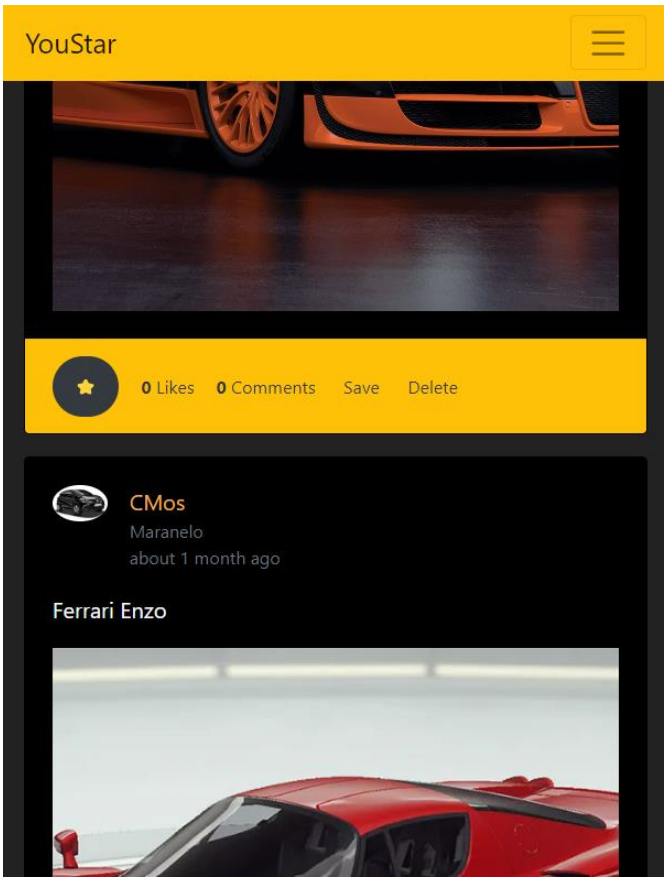
Ispis 10: *Prikaz pronalaska liste rezgovora prijavljenog korisnika*

3.2. Korisničko sučelje

3.2.1. Dizajn

Izgled korisničkog sučelja izrađen je koristeći HTML i CSS.

Dizajn aplikacije uvelike je baziran na Bootstrapu. To je paket koji omogućuje responzivan izgled stranice. Na primjer, izgled liste novih objava će se automatski prilagoditi veličini i obliku ekrana uređaja na kojemu je aplikacija pokrenuta. Slika 22. prikazuje kako lista filmova izgleda na mobilnom uređaju.



Slika 22: *Prikaz liste objava kada se aplikacija pokreće na mobitelu*

Na samom vrhu stranice nalazi se navigacijska traka koja olakšava snalaženje kroz aplikaciju. Slika 23. prikazuje izgled navigacijske trake ako je aplikacija pokrenuta na laptopu.

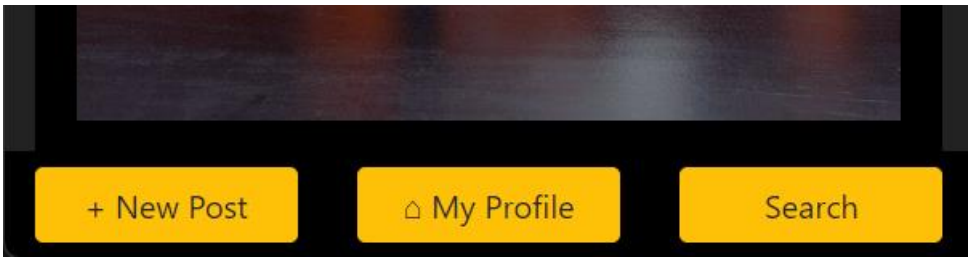
Slika 23: *Izgled navigacijske trake na ekranu laptopa*

Kako je dizajn navigacijske trake baziran na *Bootstrapu*, moguće je mijenjati njezin izgled ovisno o širini ekrana. Zbog toga će se poveznice *Dashboard*, *Chats*, *Saved Posts*, *Settings* i

Logout prilikom pokretanja aplikacija na manjem ekranu sakupiti u spuštajuću listu proširivu klikom na dugme u krajnjem desnom kutu navigacijske trake prikazanom na slici 24.

Slika 24: *Izgled navigacijske trake na ekranu mobitela*

Dio navigacije također se nalazi u podnožju aplikacije i vidljiv je na slici 25.



Slika 25: *Izgled navigacije u podnožju aplikacije*

3.2.2. React

React funkcionira tako da se određene cjeline korisničkog sučelja razdvajaju u vlastite komponente. Komponente uglavnom sadržavaju dinamičan HTML kôd koji prikazuje sadržaj dohvaćen preko http upita prema poslužiteljskoj *Node.js* aplikaciji. Primjer toga bila bi *UserDetails* komponenta koja čija zadaća je dohvatiti različite podatke o korisniku. Kako bi se odredilo čiji profil će se učitati preko *UserDetails* komponente potrebno je proslijediti id korisnika što je vidljivo u ispisu 11.

```
const UserDetails = (props) => {
  const userId = props.match.params.userId;
```

Ispis 11: *Prikaz prosljeđivanja id-a korisnika u UserDetails komponentu*

Podatke koje komponenta prikazuje potrebno je spremiti unutar useState stanja. Svako od stanja, prikazano u ispisu 12, sadrži ime varijable i funkciju preko koje se varijabla može postaviti na određenu vrijednost.

```
const [user, setUser] = useState({});
const [numOfFollows, setNumOfFollows] = useState(0);
const [numOfFollowers, setNumOfFollowers] = useState(0);
const [numOfPosts, setNumOfPosts] = useState(0);
const [visitorFollows, setVisitorFollows] = useState([]);
const [usersPosts, setUsersPosts] = useState([]);
const [loggedInUserId, setLoggedInUserId] = useState(null);
const [loggedInUser, setLoggedInUser] = useState({});
```

Ispis 12: *Prikaz useState stanja u UserDetails komponenti*

Kako bi se useState stanja popunila podatcima iz baze podataka potrebno je poslati upite poslužiteljskoj strani. Upiti su određene metode koje preko http protokola šalju određene zahtjeve poslužiteljskoj strani koja zatim komunicira s bazom podataka. Postoje 4 vrste http upita: GET, POST, PUT i DELETE. GET upit se koristi kako bi se klijent dohvatio podatke iz baze podataka i prikazao ih na korisničkom sučelju. POST upit služi kako bi se podatci uneseni na korisničkoj strani spremili u bazu podataka, PUT kako bi se određeni podatci u bazi izmijenili, dok DELETE služi za brisanje podataka. Prikaz GET, POST, PUT i DELETE upita prikazan je u ispisu 13.

```
export const createPost = async (data) => {
  const response = await axios.post("/api/post", data);
  return response;
};


export const getPost = async (postId, loggedInUserId) => {
  const response = await
axios.get(`/api/post/${postId}/${loggedInUserId}`);
  return response;
};


export const updatePost = async (postId, formData) => {
  const response = await axios.put(`/api/post/${postId}`, formData);
  return response;
};


export const deletePost = async (postId, ownerId) => {
  const response = await axios.delete(`/api/post/${postId}/${ownerId}`);
  return response;
};
```

Ispis 13: *Prikaz GET, POST, PUT i DELETE http upita*

4. Zaključak

U današnjem svijetu ljudi svakodnevno koriste društvene mreže kako bi se međusobno upoznali, razmijenili informacije, iskustva i uspomene. Rezultat ovog završnog rada je društvena mreža koja sadrži sve navedene karakteristike pružajući korisnicima dodatnu aktivnost u skupljanju što većeg broja zvjezdica.

Aplikacija omogućuje korisnicima pregled objava korisnika koje prate, like-anje, komentiranje, spremanje te kreiranje novih objava. Osim aktivnosti specifičnih za objave, korisnici mogu pregledavati profile korisnika kojih prate ili onih s javnim profilom te personalizirati vlastiti korisnički račun. Nadalje, korisnicima je omogućeno međusobno dopisivanje čime se ostvaruje efektivna komunikacija u aplikaciji. Što se tiče poboljšanja aplikacije, uvijek postoje razne ideje koje se mogu dodatno implementirati. Neke od njih su mogućnost objavljivanja više slika ili videa pod jednom objavom i mogućnost slanja slika ili video snimki putem poruke.

Pokazalo se kako su za izradu ove aplikacije Node.js i React tehnologije bile odličan izbor. Oba su razvojna okvira, iako imaju različite svrhe, usko povezana iz zato što koriste Javascript kao programski jezik što omogućuje intuitivno pisanje kôda na poslužiteljskoj i klijentskoj strani aplikacije. Korištenjem responzivnog dizajna, većina korisničkog sučelja aplikacije može se prilagoditi različitim veličinama i dimenzijama ekrana, što poboljšava korisničko iskustvo.

Društvene mreže u potpunosti su promijenile odnose među ljudima čineći komunikaciju jednostavnom i zabavnom. S napretkom moderne tehnologije čovječanstvo s godinama ostvaraju sve veći životni standard čineći svijet boljim i povezanijim mjestom.

Literatura

[1] dokumentacija za samu izradu React aplikacije https://www.udemy.com/course/react-the-complete-guide-incl-redux (*posjećeno 20.2.2023.*)

[2] dokumentacija za samu izradu Node.js aplikacije https://www.udemy.com/course-dashboard-redirect/?course_id=1879018 (*posjećeno 22.2.2023.*)

[3] MVC dokumentacija https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller (*posjećeno 22.2.2023.*)

[4] Node.js dokumentacija https://nodejs.dev/en/learn/

(*posjećeno 2.3.2023.*)

[5] dokumentacija za izradu korisničkog sučelja, https://www.w3schools.com (*posjećeno 6.3.2021.*)