Univerzitet u Sarajevu
Elektrotehnički fakultet
Odsjek za računarstvo i informatiku

Razvoj društvene mreže korištenjem React okruženja

Završni rad

I ciklus studija

Mentor:
Prof. dr. Dženana Đonko

Kandidat:
Delila Dević

Sarajevo, 2018. godina

Postavka rada

Cilj:

-Analiza glavnih karakteristika okruženja React za razvoj web aplikacija
-Razvoj društvene mreže koristeći React okruženje

Opis:

Ovaj rad će obuhvatiti analizu okruženja React za razvoj *frontenda* web aplikacija. Potom će se razviti pokazna web aplikacija Wonderland, čijim razvojem će se ilustrirati primjena pomenutog okruženja.
Rad će se sastojati iz sljedećih poglavlja:

- Poglavlje 1 je „*Uvod*“ koje sadrži uvod u temu rada, ,glavne ciljeve i strukturu rada.
- Poglavlje 2 je „*Analiza okruženja React*“ i predstavlja analizu načina rada i osnovnih koncepata okruženja React.
- Poglavlje 3 je „*Specifikacija web aplikacije Wonderland*“ u kojem je definisana namjena i funkcionalnosti pokazne web aplikacije.
- Poglavlje 4 je „*React u praksi*“ u kojem je detaljno opisana implementacija podstabla NewPost.
- Poglavlje 5 je „*Sigurnost web aplikacije Wonderland*“ u kojemje prikazana implementacija nefunkcionalnih sigurnosnih zahtjeva.
- Poglavlje 6 je „*Zaključak*“ u kojem je sumirano sve što je predstavljeno u radu i rezultati dobijeni analizom okruženja React za razvoj koričničkih interfejsa web aplikacija.

Očekivani rezultati:

- Analiza okruženja React za razvoj *frontenda* web aplikacija
- Razvijena web aplikacija koristeći prednosti okruženja React

Polazna literatura:

- Daniel Galin, Software Quality Assurance : From Theory to Implementation, Addison Wesley 2003
- Robin Wieruch, The road to learn React, 2017

Mentor:
Prof. dr. Dženana Đonko

_____

Univerzitet u Sarajevu
Naziv fakulteta/akademije: Elektrotehnički fakultet
Naziv odsjeka i/ili katedre: Računarstvo i informatika
Predmet: Verifikacija i validacija softvera

Izjava o autentičnosti radova

Seminarski rad, završni (diplomski odnosno magistarski) rad za I i II ciklus studija i integrirani studijski program I i II ciklusa studija, magistarski znanstveni rad i doktorska disertacija[1]

Ime i prezime: Delila Dević
Naslov rada: Razvoj društvene mreže koristenjem React okruženja
Vrsta rada: Završni rad I ciklusa studija
Broj stranica: ...

Potvrđujem:

- da sam pročitao/la dokumente koji se odnose na plagijarizam, kako je to definirano Statutom Univerziteta u Sarajevu, Etičkim kodeksom Univerziteta u Sarajevu i pravilima studiranja koja se odnose na I i II ciklus studija, integrirani studijski program I i II ciklusa i III ciklus studija na Univerzitetu u Sarajevu, kao i uputama o plagijarizmu navedenim na web stranici Univerziteta u Sarajevu;
- da sam svjestan univerzitetskih disciplinskih pravila koja se tiču plagijarizma;
- da je rad koji predajem potpuno moj, samostalni rad, osim u dijelovima gdje je to naznačeno;
- da rad nije predat, u cjelini ili djelimično, za stjecanje zvanja na Univerzitetu u Sarajevu ili nekoj drugoj visokoškolskoj ustanovi;
- da sam jasno naznačio prisustvo citiranog ili parafraziranog materijala i da sam se referirao na sve izvore;
- da sam dosljedno naveo korištene i citirane izvore ili bibliografiju po nekom od preporučenih stilova citiranja, sa navođenjem potpune reference koja obuhvata potpuni bibliografski opis korištenog i citiranog izvora;
- da sam odgovarajuće naznačio svaku pomoć koju sam dobio pored pomoći mentorice i akademskih tutora.

Mjesto, datum _____
Potpis _____

[1] U radu su korišteni slijedeći dokumenti: *Izjava autora* koju koristi Elektrotehnički fakultet u Sarajevu; *Izjava o autentičnosti završnog rada* Centra za interdisciplinarne studije – master studij „Evropske studije", *Izjava o plagijarizmu* koju koristi Fakultet političkih nauka u Sarajevu.*plagijarizmu* koju koristi Fakultet političkih nauka u Sarajevu.

Sadržaj

Sažetak

Cilj ovog rada je analiza prednosti i olakšanja koja nudi biblioteka React. Fokus je stavljen na primjenu tih prednosti i pisanje ponovo iskoristivih komponenti pri razvoju modernih i kompleksnih korisničkih interfejsa.

Komponente predstavljaju ključni dio svake React aplikacije,pa su njihova struktura, različiti načini pisanja, tok podataka pomoću parametara *props*, kao i *state* objekti komponente posebno detaljno opisani i potkrijepljeni primjerima iz pokazne aplikacije Wonderland. Posebna pažnja je posvećena aspektu sigurnosti React aplikacija.

Predstavljeni su funkcionalni i nefunkcionalni zahtjevi web aplikacije Wonderland, kao i njena namjena. U radu je prikazana implementacija dijela aplikacije, kako bi se na najbolji način prezentovalo stečeno znanje za razvoj aplikacija koristeći React. Pokazna aplikacija je otvorena za dalja poboljšavanja i nadogradnje.

Abstract

The aim of this document is to analyze the benefits and facilitation provided by React. The focus is on applying these benefits and writing reusable components in the development of modern and complex user interfaces.

Components represent a key part of every React application, so their structure, different writing modes, data flow using props parameters, and state component objects are particularly detailed and substained by examples from the sample application Wonderland. Special attention is paid to the aspect of React applications security.

The functional and non-functional requirements of the Wonderland web application as well as its purpose have been presented. The document presents the implementation of the part of the application in order to best present the acquired knowledge for the development of applications using React. The sample application is open for further enhancements and upgrades.

1. Uvod

Većina današnjih web aplikacija su *Single page* aplikacije koje se zasnivaju na interakciji sa korisnicima, pa zbog toga programiranje njihovih korisničkih interfejsa može biti veoma teško. Promjena korisničkog interfejsa, prilikom interakcije, postiže se kombinovanjem funkcija JavaScript jezika, JavaScript framework-a i biblioteka.

Razvojem sve kompleksnijih web aplikacija dolazi do problema prevelikog korištenja serverskih resursa. Nastojanjem rješavanja ovog problema Facebook-ovi inžinjeri su razvili JavaScript biblioteku koja se naziva React. React je deklarativna, efikasna i fleksibilna JavaScript biblioteka za kreiranje korisničkih interfejsa. Ukoliko se koristi MVC (*Model View Controller*) arhitekturalni patern, može se reći da on predstavlja samo 'V' unutar MVC. React je *open-source* komponenta zadužena samo za dio koji je vidljiv korisniku.

U ovom radu analizirana je struktura i način rada pomenute biblioteke. Fokus je na poboljšanjima i olakšavajućim aspektima koje ova biblioteka nudi. Da bi se što bolje analizirali pomenuti elementi, razvijena je i pokazna web aplikacija Wonderland.

1.**1. Cilj**

Cilj ovog rada je pokazati način razvoja web aplikacija koristeći React. Fokus je stavljen na analizu glavnih karakteristika i poboljšanja u odnosu na razvoj koristeći druge tehnologije. Da bi se manifestovali zadani ciljevi, razvija se i pokazna web aplikacija Wonderland.

1.2. Struktura rada

Rad se sastoji iz šest poglavlja:

- Poglavlje 1 je „*Uvod*“ koje sadrži uvod u temu rada, ,glavne ciljeve i strukturu rada.
- Poglavlje 2 je „*Analiza okruženja React*“ i predstavlja analizu načina rada i osnovnih koncepata okruženja React
- Poglavlje 3 je „*Specifikacija web aplikacije Wonderland*“ u kojem je definisana namjena i funkcionalnosti pokazne web aplikacije.
- Poglavlje 4 je „*React u praksi*“ u kojem je detaljno opisana implementacija podstabla NewPost.
- Poglavlje 5 je „*Sigurnost web aplikacije Wonderland*“ u kojemje prikazana implementacija nefunkcionalnih sigurnosnih zahtjeva.
- Poglavlje 6 je „*Zaključak*“ u kojem je sumirano sve što je predstavljeno u radu i rezultati dobijeni analizom okruženja React za razvoj *frontenda* web aplikacija.

1.3. Očekivani rezultati

- Analiza okruženja React za razvoj *frontenda* web aplikacija
- Razvijena web aplikacija koristeći prednosti okruženja React

1.4. Metodologije rada

Za realizaciju postavljenih ciljeva rada, bilo je potrebno pročitati polaznu literaturu i upoznati se sa teorijskim osnovama za razvoj korisničkog interfejsa web aplikacija koristeći React. Nakon sticanja potrebnih osnovnih znanja, definisana je arhitektura pokazne web aplikacije Advena koristeći MVC arhitekturalni *patern.* Zatim je implementirana web aplikacija Advena sa očekivanim funkcionalnostima.

1.5. Napomena

Za izradu ovog rada korišten je font *Calibri*. Veličina teksta je 12pt. Tehnički termini, za koje ne postoji odgovarajući prijevod na bosanskom jeziku, su napisani u *Italic* stilu. Za referenciranje izvora korištenih prilikom izrade rada korišten je *Harvard* stil referenciranja.

2. Analiza okruženja React

U ovom poglavlju izvršena je analiza okruženja za razvoj korisničkog interfejsa React. Opisan je nastanak React-a te svi njegovi bitni dijelovi.

2.1. Historija JavaScript-a i nastanak React-a

React je prvobitno predstavljao JavaScript port za XHP, verziju PHP-a koju je objavio Facebook 2010. godine. XHP je uglavnom zadužen za minimizaciju *Cross Site Scripting* (XSS) napada. *Cross site scripting* je vrsta sigurnosnog napada u kojemnapadač ubrizgava podatke unutar pouzdanih stranica. Ova vrsta napada se dešava kada se nepouzdanom izvoru dopusti da ubaci svoj sopstveni kod u web aplikaciju, najčešće u obliku JavaScript koda koji se izvršava na žrtvinom pretraživaču. Na taj način se ne napada određena žrtva direktno, već se iskorištava ranjivost stranice.

Ipak, problem koji XHP nije uspio riješiti predstavljale su dinamičke web aplikacije koje zahtijevaju mnogo serverskih resursa. Zbog toga, inžinjer Facebook-a Jordan Walke odlučuje da način pisanja interfejsa iz XHP-a uključi u JavaScript i za to je imao svega šest mjeseci. Rezultat njegovog rada je React. [1]

Nakon zvanične objave 2011. godine, React nije dobio veliku podršku, jer se smatralo da predstavlja veliki korak unazad, ali se situacija drastično mijenja nakon „React turneje“. React je prvi put upotrijebljen za prikazivanje Facebook objava, a kasnije se počinje koristiti i u mnogim drugim poznatim web aplikacijama, poput Inastagram-a, Netflix-a, WhatsApp-a, Codeacademy-ja. Vremenom se razvijao, te Facebook odlučuje da React postane u potpunosti *open source* u maju 2013. godine. [2]

2.2. Uvod u React

React je deklarativna, efikasna i fleksibilna JavaScript biblioteka za pravljenje korisničkih interfejsa. On predstavlja samo 'V' unutar MVC (*Model View Controller*). React je *open-source* komponenta bazirana na frontendu, zadužena samo za dio koji je vidljiv korisniku.

React koristi deklarativni pogled koji olakšava da aplikacija koja se gradi bude i efikasna i fleksibilna. Deklarativni pogled čini kod predvidljivijim, jednostavnijim i olakšava debagiranje. [3]

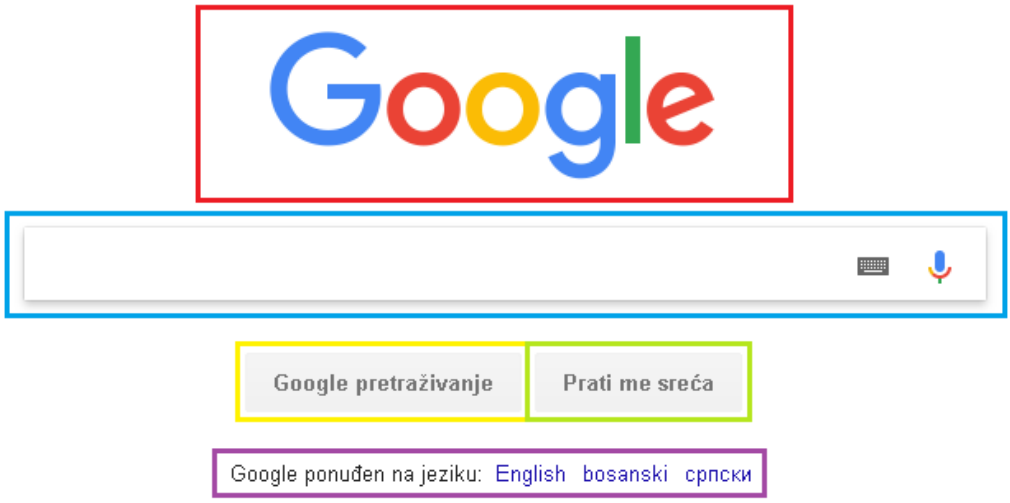
Kreiran je od strane softver inžinjera Facebook-a, u svrhu razvoja web aplikacija koje su skalabilne, održive i robusne.

Skalabilnost podrazumijeva sposobnost sistema da funkcioniše na predviđen način, neovisno o promjeni njegove velilčine, dok je robusnost sposobnost sistema da se nakon pada brzo i lahko vrati u operativno stanje, pri tome ne oštećujući podatke. Održivost softvera je jedna od njegovih najbitnijih karakteristika, a predstavlja proces modifikacije softvera ili njegovih komponenti zbog ispravljanja grešaka, promjene u okruženju ili dodavanja novih funkcionalnosti, a promjene u softveru su neizbježne.

Pri razvoju React-a korišten je drugačiji pristup od onog koji je korišten u kreiranju Angulara i ostalih MVC okruženja. U tim okruženjima moguće je izgraditi cijelu aplikaciju, bez potrebe za dodatnim bibliotekama, jer pružaju HTML predloške, kontrolere i servise. S druge strane, React je fokusiran samo na *view* sloj aplikacije, pa se za izgradnju funkcionalne aplikacije moraju koristiti i druge biblioteke.

2.3. React komponente

React je način izgradnje korisničkih interfejsa. Ono što olakšava njihovu izgradnju i omogućava ponovnu iskoristivost je podjela stranice na manje dijelove. Ti dijelovi stranice se nazivaju komponente.



Slika 2.1. Primjer podjele stranice na komponente

Na slici 2.1. svaka označena particija se smatra komponentom. Dakle, komponenta je kod koji predstavlja dio web stranice.

React aplikacija je napravljena od više komponenti od kojih je svaka zadužena za rad sa malim dijelom HTML-a koji se može koristiti. Komponente mogu sadržavati druge komponente da bi se postigla izgradnja kompleksne aplikacije iz jednostavnijih blokova.

Postoje tri načina pisanja React komponenti [5]:

- Upotreba varijabilne funkcije – na početku razvoja React-a ovo je bio jedini način pisanja komponenti. U sljedećem primjeru varijabla Post predstavlja React komponentu koju je moguće koristiti bilo gdje na web stranici.

```
var Post = React.createClass({
        render(){
                return(
                        <div>
                                ...
                        </div>
                )
        }});
```

Listing 2.1. Primjer kreiranja komponente upotrebom varijabilne funkcije

- Upotreba klasa – nakon uvođenja klasa u JavaScript 2015. godine postaje moguće kreirati JavaScript klasu koja je naslijeđena iz React.Component, kao u sljedećem primjeru.

```
class Post  extends React.Component{
        render(){
                return(
                        <div>
                                ...
                        </div>
                )
}}
```

Listing 2.2. Primjer kreiranja komponente upotrebom klase

- Upotreba komponenti bez stanja – predstavljaju jednostavne funkcije koje vraćaju React komponente. Ovakav način pisanja komponenti je koristan samo za one komponente koje se fokusiraju na prezentaciju podataka, ali ne i manipulaciju njima.

```
constPost = () => {
        render(){
                return(
                        <div>
                                ...
                        </div>
                )
}}
```

Listing 2.3. Primjer kreiranja komponente bez stanja

2.4. Props i state

U React aplikacijama postoji hijerarhija komponenti, te u skladu s tim postoje *parent* i *child* komponente. *Child* komponente su smještene unutar *render()* funkcije *parent* komponente koja im može proslijediti potrebne podatke. Prosljeđivanje podataka je moguće korištenjem parametara props.

Props parametri komponenti mogu prestavljati varijablu ili funkciju. Veoma je bitno da su funkcije koje se prosljeđuju kao props *pure* JavaScript funkcije. Takve funkcije ne mijenjaju vrijednosti svojih parametara i uvijek vraćaju iste rezultate za iste vrijednosti ulaznih parametara. U sljedećem primjeru komponenta Parent prosljeđuje Child komponenti kao props parametre varijablu title i funkciju sum(a,b).

```
class Parent extends React .Component {
        sum(a,b){
                return a+b;
        }
        render () {
                const  title = "sum";
                return (
                        < Child
                                title = {title}
                                function = { this.sum.bind(this) }
                        / >
        ) ;
}}
```

Listing 2.4. Primjer prosljeđivanja props parametara

Pored props objekata koje komponenta ne može mijenjati, veliku važnost imaju i state objekti. State objekti omogućavaju kreiranje dinamičkih i interaktivnih komponenti, te osiguravaju usklađenost korisničkog interfejsa i trenutnih podataka. State predstavlja privatni atribut jedne određene komponente. Ukoliko se state objekat promijeni, primjenom funkcije setState(), komponenta se ponovo renderuje.

Prije upotrebe state objekta, s ciljem izbjegavanja pristupa neinicijalizovanom objektu, potrebno je postaviti njegovo početno stanje. To se obično radi u metodi constructor(), dok se podaci sadržani u state objektu popunjavaju nakon dodavanja komponente na DOM, koristeći metodu componentDidMount(). U primjeru koji slijedi je prikazano postavljanje inicijalnog stanja, te njegovo mijenjanje u funkciji componentDidMount().

```
class Post extends React .Component
        constructor ( props ) {
                super ( props ) ;
                this . state = {
                        data : []
                };
         }
         componentDidMount () {
                this . setState ({
                        data : [1 , 2 , 3]
                }) ;
        }
        ...
```

Listing 2.3. Primjer postavljanja i mijenjanja state objekta

2.5. Virtuelni DOM

Za razliku od prethodnih JavaScript okruženja za razvoj korisničkog interfejsa, React ne koristi pretraživačev DOM, nego gradi virtuelnu prezentaciju DOM-a, u obliku stabla JavaScript objekata. Pisanjem koda u React-u se ne manipuliše DOM-om, nego njegovom virtuelnom reprezentacijom, a za reflektovanje tih promjena na pretraživačevom DOM-u je zadužen React. [4]

Pri manipulaciji DOM-a potrebno je prvo locirati element, a zatim direktno promijeniti taj element koristeći *innerHTML*. Ovakav način razvoja može biti problematičan zbog otežanog praćenja trenutnog stanja svakog od elemenata i zbog toga što je manipulacija DOM-a skupa operacija i može usporiti aplikaciju koja se razvija. Rješenje ovog problema je korištenje virtuelnog DOM-a.

Virtuelni DOM je stablo JavaScript objekata koji predstavljaju stvarni DOM. Prilikom promjene podataka pravi se novi virtuelni DOM. Ova operacija je veoma brza i neće utjecati na performanse aplikacije. Reactova implementacija virtualnog DOM-a dolazi sa bitnim optimizacijama performansi. Inžinjeri Facebook-a koji su radili na razvoju React-a su uspjeli da smanje kompleksnost *diff* algoritam koji se koristi za ažuriranje virtualnog DOM-a sa $O(n^3)$ na $O(n)$.[2] Pored toga što korištenje virtualnog DOM-a povećava performanse modifikacije DOM-a, ažuriranje virtuelnog DOM-a je olakšano, jer se događa samo onda kada se state objekat promijeni, kao što je već pomenuto.

[2] Big-O notacijom se definiše gornja granica složenosti algoritma

2.6. JSX

JSX (engl. *JavaScript Syntax Extension*) predstavlja sintaksu, napravljenu od strane Facebook inžinjera, kako bi se omogućilo pisanje komponenti pomoću HTML-a. U već navedenim primjerima se može primijetiti metoda render koja vraća kod napisan u JSX-u.

Upotreba JSX-a nije obavezna pri kreiranju React komponenti, ali pruža određene prednosti u odnosu na korištenje čistog JavaScript-a. Jedna od glavnih prednosti jeste optimizacija pokretanja koda prilikom prevođenja JSX-a u JavaScript koristeći predprocesor. Pored toga, omogućeno je pisanje JavaScript-a unutar JSX-a. Jedna od najčešćih situacija upotrebe JavaScript-a unutar JSX-a jeste uslovno renderovanje komponenti. Kao što je u sljedećem primjeru prikazano, komponenta Admin se prikazuje, ukoliko je korisnik prijavljen kao administrator.

```
        . . .
render () {
        const  userLevel = this.props.userLevel;
                return (
                        . . .
                        {userLevel === 'admin' && <AdminPanel/>}
                        . . .
                ) ;
}}
```

Listing 2.3. Primjer upotrebe JavaScript-a unutar JSX-a

Unutar ovog poglavlja opisani su principi rada i kreiranja React aplikacija, te njihova sintaksa. Detaljno su opisane React komponente, te njihovi glavni dijelovi.

3. Specifikacija web aplikacije Wonderland

U okviru ovog poglavlja opisana je namjena, funkcionalni i nefunkcionalni zahtjevi web aplikacije Wonderland, koja se razvija u radu. Ova aplikacija će služiti kao pokazna aplikacije da bi se predstavile mogućnosti i olakšice koje nudi React biblioteka. U ovom poglavlju će biti prikazani i prototipi same aplikacije.

3.1. Namjena aplikacije

Web aplikacija Wonderland predstavlja društvenu mrežu koja je namijenjena prvenstveno turistima. Dok većina glavnih društvenih mreža, kao što su Facebook ili Twiter najviše pažnje posvećuju medijima, sajtovi poput Wonderland, specifični za putovanja, mogu biti izuzetno korisni i zanimljivi za one koji žele da podijele svoje fotografije i priče, da se sretnu sa nekim na putovanjima ili da vide iskustva drugih turista.

3.2. Pregled sadržaja aplikacije

Web aplikacija Wonderland je zamišljena kao spoj Facebook-a i Instagram-a, sa stavljanjem akcenta na turizam, dijeljenja lokacija, priča i slika sa putovanja. Kao takva, ova društvena mreža se sastoji od nekoliko glavnih dijelova, koji će biti objašnjeni u nastavku.

- Početna stranica – predstavlja stranicu za prijavu i registraciju korisnika

- Naslovna stranica – ovaj dio služi za pregled objava osoba koje korisnik prati i objava vezanih za lokacije koje ga zanimaju, sadrži polje za pretragu, te segment za objavljivanje statusa i slika.

- Profilna stranica – u ovom dijelu su prikazane osnovne informacije o korisniku, njegove fotografije i objave.

3.3. Funkcionalni zahtjevi

U ovom dijelu su prikazane sve funkcionalnosti društvene mreže Wonderland.

- Prijava korisnika – Da bi se korisnik mogao prijaviti mora biti registrovan. Prijava se vrši na osnovu email-a ili korisničkog imena i šifre koja se u bazi čuva *heširana*. Šifra se može promijeniti klikom na „Zaboravili ste šifru?“ dugme. Svi uneseni podaci se moraju validirati.

- Registracija korisnika – Korisnik može kreirati račun ukoliko ne postoji niti jedan drugi račun koji koristi isti email. Prilikom kreiranja korisničkog računa potrebno je unijeti d ime, prezime, email, datum rođenja i šifra. Sva polja su obavezna. Dužina šifre mora biti veća od 8 karaktera,te mora sadržavati barem jedno veliko slovo, jedan broj i jedan od znakova ~, !, @, #, $, %, ^, *, -; =, ., ;, '.

- Objava statusa i slika – Korisnik ima mogućnost objavljivanja slika i informacija o trenutnoj lokaciji, raspoloženju i vremenskim prilikama, te označavanja prijatelja na objavi. Nakon odabira željenih opcija, klikom na dugme Objavi aplikacija generiše smislenu rečenicu, te objavu prikazuje na profilu korisnika.

- Brisanje objava – U slučaju da se korisniku ne sviđa objava ima mogućnost da je obriše.

- Pregled objava – Korisnik ima mogućnost pregleda objava svih korisničkih računa koje prati, te objava vezenih za lokcije koje prati na svojoj naslovnoj stranici.

- Praćenje korisničkih računa i lokacija – Registrovani korisnik ima mogućnost da prati druge korisnike, ali i lokacije, kao što su država, grad, univerzitet, trgovina, restoran i sl. Korisnik ima mogućnost pretrage korisničkih računa i lokacija, kako bi što lakše pronašao ono što ga interesuje.

- Pretraga – Korisnik ima mogućnost da vrši pretragu drugih korisničkih računa prema imenu i prezimenu ili lokacija po nazivu. Kada se pretraga završi rezultati pretrage se prikazuju na ekranu u vidu tabele. Pored imena i prezimena korisnika ili naziva lokacije prikazuje se i dugme za praćenje/prestanak praćenja.

- Dodavanje/uređivanje ličnih podataka – Ova funkcionalnost predstavlja dio izgradnje korisničkog računa. Korisnik bi trebao da bude u mogućnosti da unese ili promijeni lične podatke, kao što su jezik, spol, posao, obrazovanje i broj telefona. Ova polja mogu ostati i prazna. Da bi korisnik mogao urediti ova polja mora biti prijavljen.

- *Upload* slike profila – Potrebno je omogućiti korisnicima upload slike profila. Pri kreiranju korisničkog računa postavlja se preddefinisana slika profila. Profilna slika je vidljiva svima.

- *Drag and drop* slike – pri promjeni slike profila ili objave slike, korisnik treba biti u mogućnosti da *drag and drop*-a željenu sliku sa svog računara.

- Postavka vidljivosti korisničkog računa – Ova funkcionalnost omogućava korisniku da vidljivost ličnih podataka, detalja korisničkog računa i objava postavi na javnu ili privatnu, prema njegovoj želji. Inicijalna vrijednost za vidljivost navedenih podataka je javna.

- Prikaz profila – Bilo koji prijavljeni korisnik može pregledati svoj profil ili profil svakog od korisnika ove društvene mreže. U ovisnosti od postavljene vidljivosti prikazuju se određeni podaci. U slučaju da je korisnički račun privatni, lične podatke i objave mogu vidjeti samo oni korisnici koji ga prate.

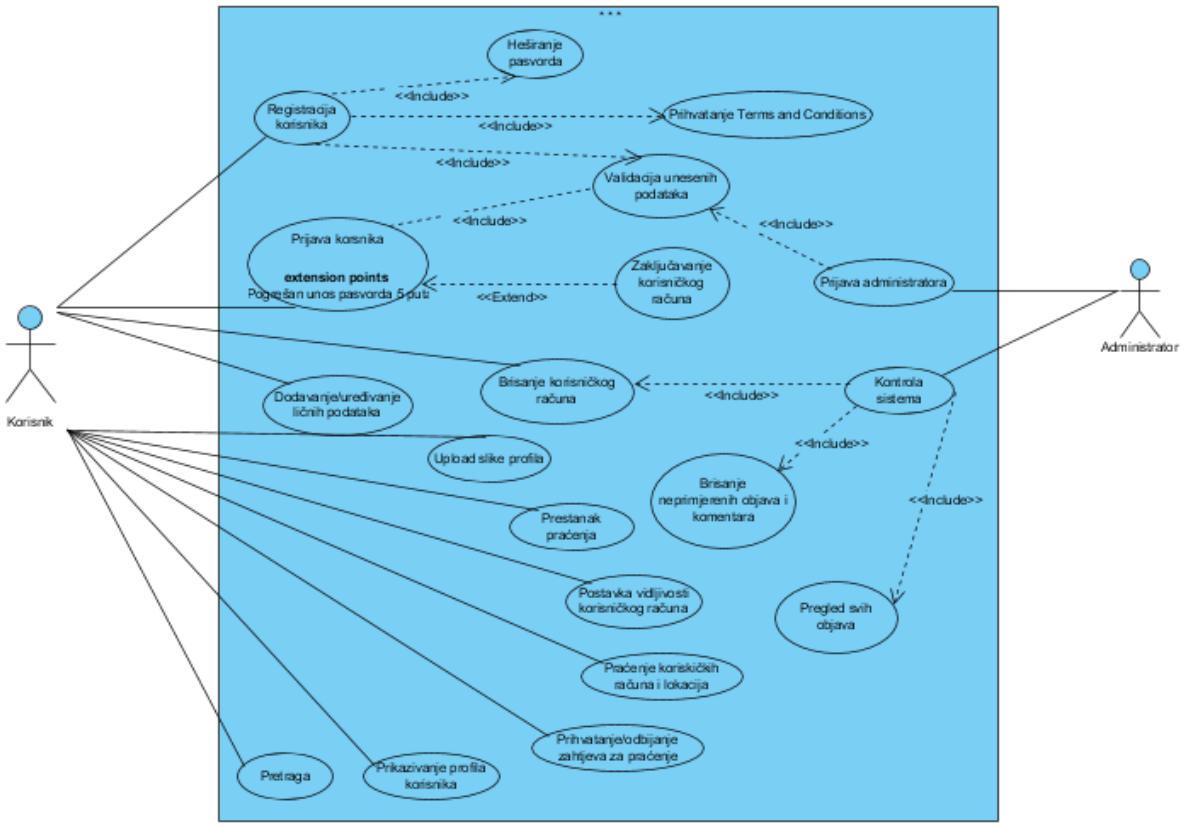
- Prihvatanje/odbijanje zahtjeva za praćenje – Ukoliko korisnik sa privatnim računom primi zahtjev za praćenje od strane drugog korisnika, bira između dvije opcije, da prihvati ili odbije zahtjev. Ako primalac odluči da prihvati zahtjev, podnosilac zahtjeva se dodaje u listu pratitelja primaoca. U suprotnom, pošiljalac prima obavijest da je zahtjev za praćenje odbijen i isti se uklanja iz liste pristiglih zahtjeva kod primaoca.

- Prestanak praćenja – Svaki registrovani korisnik treba imati mogućnost prestanka praćenja profila ili lokacija koje trenutno prati.

- Dodavanje/uređivanje opisa – Ova funkcionalnost predstavlja dio izgradnje korisničkog računa. Korisnik bi trebao da bude u mogućnosti da unese ili promijeni opis na svom profilu koji može sadržavati dodatne informacije o korisniku. Ova polja mogu ostati i prazna. Da bi korisnik mogao urediti ova polja mora biti prijavljen.

- Odjava sa korisničkog računa – Korisnik ima mogućnost odjave sa korisničkog računa klikom na odgovarajuću ikonu. Nakon uspješnog odjavljivanja, korisnik je preusmjeren na početnu stranicu.

Na slici 3.1. predstavljen je dijagram slučajeva upotrebe aplikacije Wonderland, sa grafički prikazanim funkcionalnostima.



Slika 3.1. Dijagram slučajeva upotrebe sistema Wonderland

3.4. Detaljna specifikacija određenih funkcionalnih zahtjeva aplikacije

Naziv zahtjeva	**Registracija korisnika**
Ocjena važnosti	Pripada Must-be Quality kategoriji Kano modela.
Preduslov	Ne postoji račun koji koristi isti email.
Ulaz	Korisnički podaci: ime i prezime, email, šifra i datum rođenja.
Procesiranje	Kreiranje korisničkog računa i spremanje njegovih podataka u bazu.
Izlaz	Obavijest o uspješnoj registraciji i preusmjeravanje na naslovnu stranicu.

Naziv zahtjeva	**Prijava korisnika**
Ocjena važnosti	Pripada Must-be Quality kategoriji Kano modela.
Preduslov	Korisnik je registrovan.
Ulaz	Korisnički email i šifra.
Procesiranje	Provjera korisničkih podataka i početak korisničke sesije.
Izlaz	Prijava greške u slučaju unosa pogrešnih korisničkih podataka ili preusmjeravanje na naslovnu stranicu

Naziv zahtjeva	**Praćenje koriskičkih računa i lokacija**
Ocjena važnosti	Pripada Must-be Quality kategoriji Kano modela.
Preduslov	Korisnik je prijavljen.
Ulaz	Korisnički račun ili oznaka lokacije koja se želi pratiti.
Procesiranje	Dodavanje korisničkog računa ili lokacije na spisak pratitelja i upis u bazu.
Izlaz	Obavijest o uspješnom praćenju korisničkog računa ili lokacije.

Naziv zahtjeva	**Prikaz profila**
Ocjena važnosti	Pripada Must-be Quality kategoriji Kano modela.
Preduslov	Korisnik je prijavljen.
Ulaz	Klik na odgovarajuće dugme za pregled vlastitog profila ili klik na ime drugog korisnika za pregled njegovog profila.
Procesiranje	Povlačenje podataka vezanih za određeni korisnički račun i učitavanje sadržaja na odgovarajuću formu.
Izlaz	Prikaz korisničkog profila.

Naziv zahtjeva	**Pretraga**
Ocjena važnosti	Pripada Must-be Quality kategoriji Kano modela.
Preduslov	Korisnik je prijavljen.
Ulaz	Tekst za pretragu.
Procesiranje	Pretraga baze podataka na osnovu unesenog teksta za pretragu i prikaz rezultata pretrage.
Izlaz	Prikaz rezultata pretrage na odgovarajućoj formi.

Naziv zahtjeva	**Objava statusa i slika**
Ocjena važnosti	Pripada Must-be Quality kategoriji Kano modela.
Preduslov	Korisnik je prijavljen.
Ulaz	Slika ili odabir ponuđenih opcija za informisanje drugih korisnika o trenutnoj lokaciji, vremenu ili raspoloženjima.
Procesiranje	Odabir slike iz galerije ili odabir ponuđenih opcija o trenutnoj lokaciji, vremenu ili raspoloženjima, upis u bazu i prikaz objave na profilu korisnika i naslovnim stranicama pratilaca.
Izlaz	Prikaz objave na profilu korisnika i naslovnim stranicama pratilaca.
Prioritet	

3.5. Akteri

U sistemu Wonderland postoje dvije vrste aktera:

- Administrator - osoba koja je zadužena za održavanje web stranice. Ima mogućnost pristupa određenim informacijama svih korisničkih računa, te mogućnost brisanja komentara, objava ili korisničkih računa.

- Korisnik – osoba koja je registrovana na stranicu i ima mogućnost objavljivanja slika i statusa i praćenja drugih korisničkih računa i lokacija.

3.6. Nefunkcionalni zahtjevi

Za razliku od funkcionalnih zahtjeva koji opisuju šta aplikacija treba da radi, nefunkcionalni zahtjevi definišu koliko dobro aplikacija mora da funkcioniše. Ove zahtjeve ćemo podijeliti u tri kategorije koje će u nastavku biti objašnjene.

3.6.1. Zahtjevi performansi

- Brzina – Sistem bi trebao biti brz i ne bi se trebao usporavati sa porastom broja korisnika. Vrijeme odziva sistema prilikom korištenja ne smije biti veće od 10 sekundi.

3.6.2. Sigurnosni zahtjevi

- Prilikom registracije uneseni email je potrebno validirati.
- Šifra bi trebala imati najmanje osam karaktera, sadržavajući barem jedno malo slovo, jedno veliko, jedan broj.
- Šifru je potrebno smjestiti u bazu podataka kao *hash.*
- Osigurati zaštitu od Cross Site Scriptign-a

3.6.3. Atributi kvalitete softvera

- Korisnost – Korisnički interfejs bi trebao biti jasan i jednostavan i ne treba sadržavati nebitne detalje koji mogu zbuniti korisnika.
- Pristupačnost – Sistem bi trebao uvijek biti dostupan. Za bolje korisničko iskustvo potrebno je osigurati minimalan broj padova sistema. Sistem treba biti pouzdan i tačno izvršavati određene akcije koje korisnik odabere.
- Testabilnost – Sistem je potrebno implementirati tako da bude pogodan za testiranje, u svrhu otkrivanja i uklanjanja eventualnih grešaka.
- Održavanje – Sistem bi trebao biti razvijan i ažuriran stalno. Trebalo bi biti lahko dodavanje novih ili prilagođavanje već postojećih funkcionalnosti.

3.7. Prototip aplikacije

U ovoj sekciji će biti prikazan dizajn interaktivnog prototipa web aplikacije Wonderland. Prototip je rađen u alatu Balsamiq.

Na slici 3.2. je prikazan željeni izgled stranice za prijavu, odnosno registraciju korisnika.

Slika 3.2. Prototip stranice za prijavu/registraciju

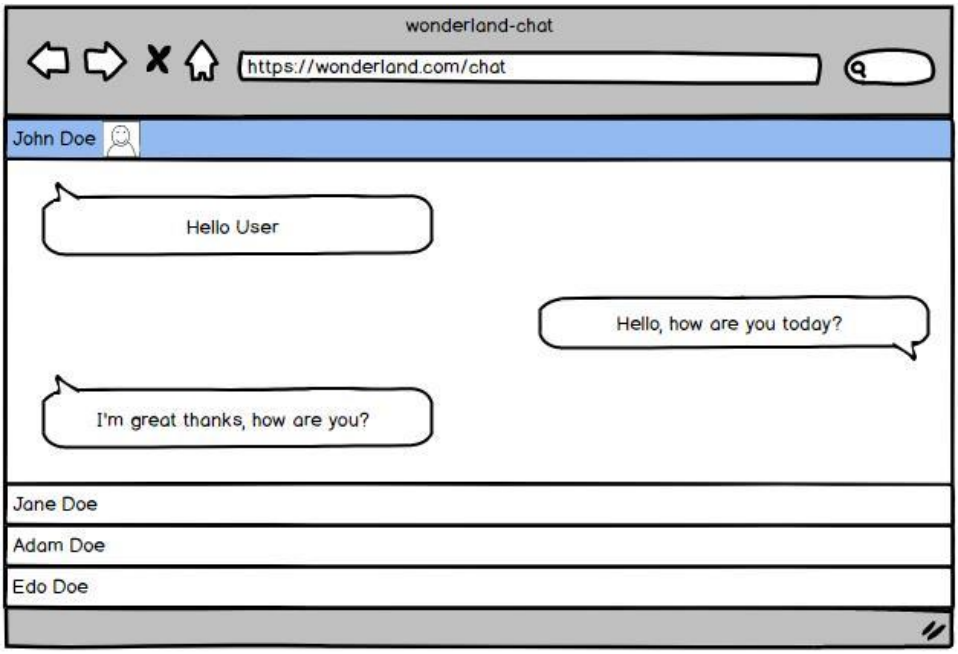
Na slici 3.3. je prikazan željeni izgled naslovne stranice web aplikacije Wonderland, koja se prikazuje nakon uspješne prijave korisnika.

Slika 3.3 Prototip naslovne stranice web aplikacije Wonderland

Na slici 3.4. je prikazan željeni izgled profilne stranice korisnika John Doe, sa prikazanom jednom njegovom objavom.

Slika 3.4. Prototip profilne stranice we aplikacije Wonderland

Na slici 3.5. je prikazan prototip izgleda stranice na kojoj se nalazi chat jednog korisnika.



Slika 3.5. Prototip stranice na kojoj je predstavljen chat

3.8. Odabir tehnologije za razvoj

Za razvoj klijentske strane web aplikacije Wonderland koristit će se okruženje React, dok će se za razvoj serverske strane koristiti Microsoft .NET okruženje. Baza podataka koja će se koristiti je MySQL baza podataka,a alat za administraciju baze MySQL Express.

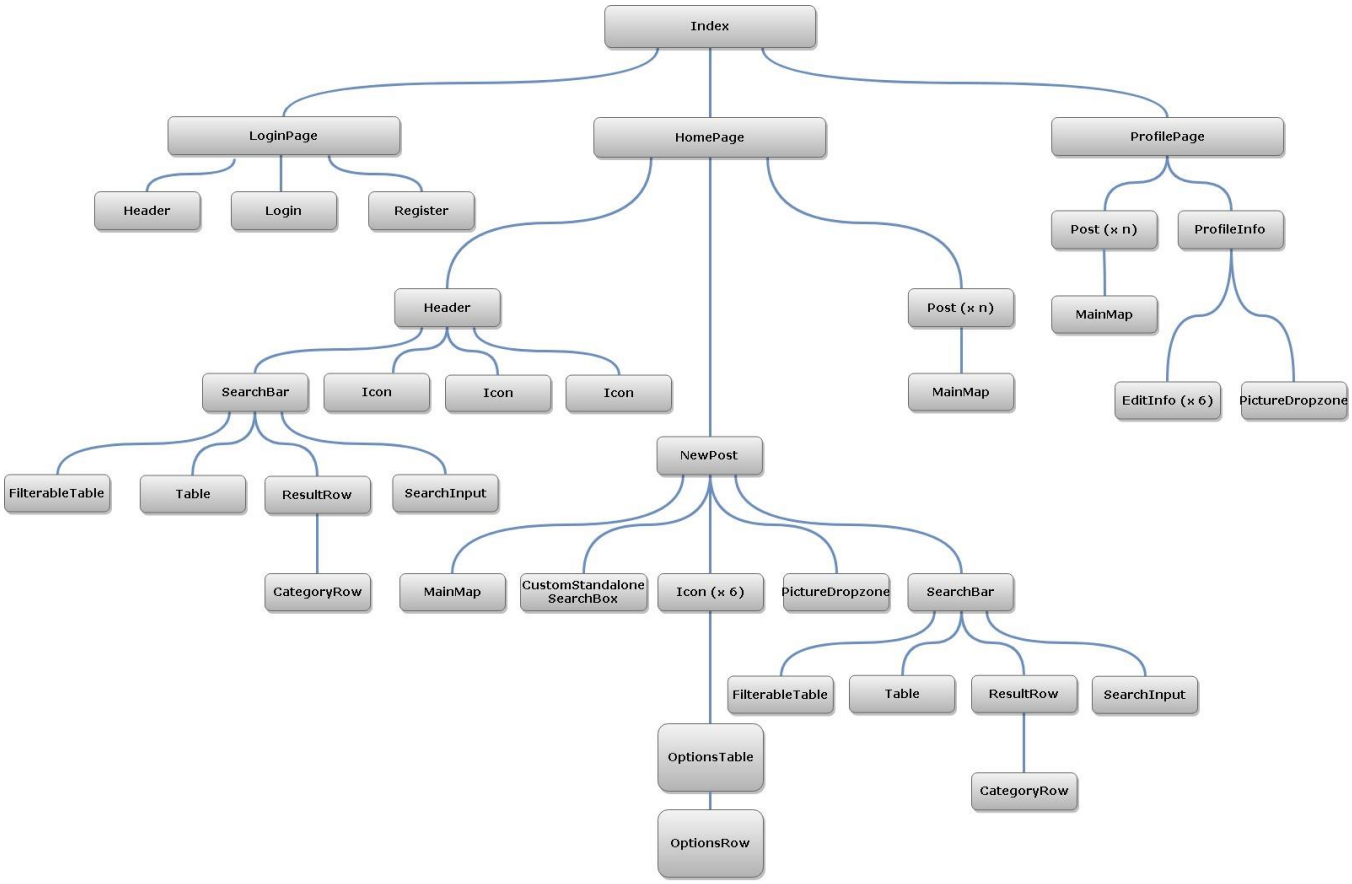
U ovom poglavlju data je specifikacija pokazne web aplikacije Wonderland. Sprovedena je detaljna analiza funkcionalnih i nefunkcionalnih zahtjeva, a zatim su odabrane tehnologije za razvoj aplikacije. Pored toga, predstavljen je i prototip web aplikacije Wonderland.

4. React u praksi

U ovom poglavlju je prikazana hijerarhija svih komponenti u aplikaciji, te detaljno opisana implementacija podstabla NewPost. Naglasak je stavljen na ponovnu iskoristivost komponenti i korištenje najboljih praksi u implementaciji.

4.1. Hijerarhija komponenti

Izgradnja komponenti koje se mogu iskoristiti u različitim aplikacijama i sa različiitim podacima je glavna ideja i dobra praksa prilikom razvoja *Single page* aplikacija. Poštujući ovu praksu, web aplikacija Wonderland se sastoji od ugnježdenih, ponovo iskoristivih komponenti, pa je veoma važno jasno definisati komponente i njihove veze. Dijagram koji prikazuje hijerarhiju komponenti prikazan je na sljedećoj slici:



Slika 4.1. Prikaz komponenti aplikacije Wonderland

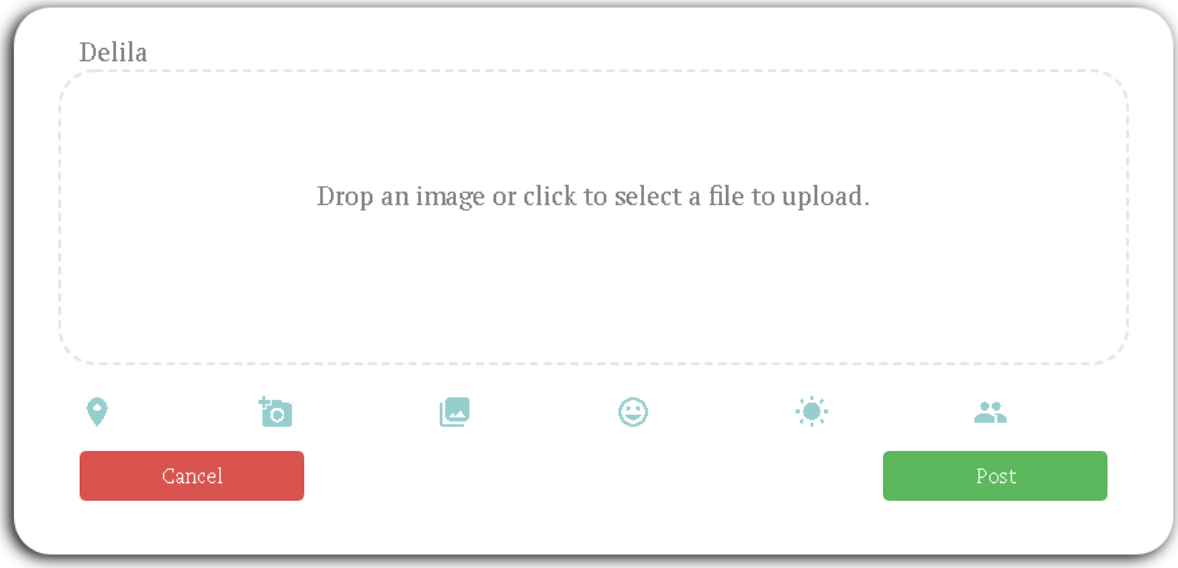
Pri raspoređivanju komponenti bitno je obratiti pažnju na *state* i *props* objekte svake komponente, kako bi oni sadržavali samo neophodne podatke. Upravo zbog toga je dodana Index komponenta koja sadrži informacije o svim korisnicima, registrovanom korisniku, njegovim pratiteljima i njihovim objavama u *state* objektu, kako bi ih mogla adekvatno proslijediti komponentama LoginPage, HomePage i ProfilePage.

4.2. Implementacija podstabla *NewPost*

Komponenta *NewPost* je najkompleksnija komponenta pokazne web aplikacije Wonderland, te će zbog toga njena implementacija biti prikazana u ovoj sekciji. NewPost komponenta služi za kreiranje i dijeljenje nove objave, što je jedan od glavnih dijelova društvene mreže Wonderland. U sklopu nove objave korisniku se pruža mogućnost dijeljenja slike, lokacije, vremenskih prilika, raspoloženja i označavanja prijatelja. U nastavku će se pojedinačno obraditi ove mogućnosti i njihova implementacija. Zbog veličine pomenute komponente, njena implementacija neće u cjelosti biti prikazana, već samo njeni najbitniji dijelovi.

Komponenta za kreiranje nove objave se nalazi na početnoj stranici web aplikacije Wonderland. Izgled ove komponente je prikazan na slici 4.2., a nakon klika na komponentu prikazuju se dodatne mogućnosti vezane za kreiranje objave, kao što je prikazano na slici 4.3.

Slika 4.2. Početni izgled komponente NewPost



Slika 4.3. Izgled komponente NewPost nakon klika

Na početnom izgledu komponente NewPost prikazuje se tekst koji korisnika potiče da kreira novu objavu. U ovom slučaju je to rečenica „*Share photos with friends!*“, ali to je samo jedna od četiri rečenice koje se mogu naći na tom mjestu. Izbor ovih rečenica je nasumičan, čime se postiže određena dinamika i izbjegava monotonost aplikacije.

Na listingu 4.1. prikazana je struktura komponente NewPost. U prvih sedam linija koda uključeni su moduli od kojih komponenta zavisi, uključujući i komponente SearchBar, MainMap, CustomStandaloneSearchBox, Icon i PictureDropzone koje ova komponenta koristi.

```
import React, { Component } from 'react';
import { Jumbotron, Button, Container } from 'reactstrap';
import Icon from './Icon';
import MainMap from './MainMap';
import CustomStandaloneSearchBox from './CustomStandaloneSearchBox';
import PictureDropzone from './PictureDropzone';
import SearchBar from '../SearchBar/SearchBar';

const CLOUDINARY_UPLOAD_PRESET = 'n9wsbdne';
const CLOUDINARY_ URL ='https://api.cloudinary.com/v1_1/dykzjdkwy/upload';

class NewPost extends Component {
        constructor(props) { ... }

        handlePostClick(e) { ... }

        onLocationIconClick(e) { ... }

        onLocationSubmit(text) { ... }

        showMap(lat, lng) { ... }

        onFriendsIconClick() { ... }

        onTaggedFriend(name) { ... }

        changeSequence(sequence) { ... }

        onImageUpload(url) { ... }

        onCancelButtonClick(e) { ... }

        onPostButtonClick(e) { ... }

        render() { ... }

        }
export default NewPost;
```

Listing 4.1. Struktura komponente NewPost

Prva komponenta koja je importovana je Icon, komponenta koja se koristi na najviše mjesta u aplikaciji Wonderland, pa predstavlja najbolji primjer ponovo iskoristive komponente. Kod dijela ove komponente prikazan je na listingu 4.2.

```
import React, { Component } from 'react';
import { Jumbotron, Button, Container } from 'reactstrap';
import OptionsTable from './OptionsTable';

class Icon extends Component {
constructor(props) {
        . . .
        this.state = {
        visible: "none",
                }
}
handleIconClick(e) {
        if (this.props.displayOptions) {
                if (this.state.visible === "none")
                        this.setState({
                        visible: "block"
                        });
                else
                        this.setState({
                        visible: "none"
                        });
        }
        else
                this.props.handleIconClick();
  }
  changeSequence(sequence) { . . . }
  render() {
    return (
      <span className={`${this.props.class}`} onClick={this.handleClick} >
        <i className={`material-icons ${this.props.class}`} > {this.props.name}  </i>
        <div id="moodAndWeather" style={{ display: this.state.visible }} >
          <OptionsTable data={this.props.data}
           changeSequence={this.changeSequence}></OptionsTable>
        </div>
      </span>
    );}}
```

Listing 4.2. Prikaz dijela komponente Icon

Slika 4.4. Primjeri različitih komponenti Icon

Primjer izgleda različitih verzija komponente *Icon* prikazan je na slici 4.4. Kako bi se razvila komponenta koja može imati različite ikone i različite funkcionalnosti nakon klika na njih, komponenti Icon se kao *props* parametri prosljeđuju *name* od kojeg zavisi izgled ikone koja će biti prikazana, class od kojeg zavisi boja ikone, funkcije handleIconClick i changeSequence koje osiguravaju da se nakon klika na ikonu izvrše željene akcije, definisane unutar komponente kojoj Icon predstavlja *child* komponentu. Parametar displayDiv se prosljeđuje komponenti kao bool vrijednost, true ukoliko se na klik ikone treba prikazati komponenta OptionsTable, u suprotnom false.

Prikazivanjem komponente OptionsTable korisniku se nudi mogućnost odabira trenutnog raspoloženja i vremenskih prilika. Izgled ove komponente prikazan je na slici 4.5. Kako bi se omogućilo prikazivanje proizvoljnog broja opcija u ovoj tabeli, red tabele treba da predstavlja komponentu. U tu svrhu je implementirana i OptionsRow komponenta.

Slika 4.5. Izgled komponente OptionsTable

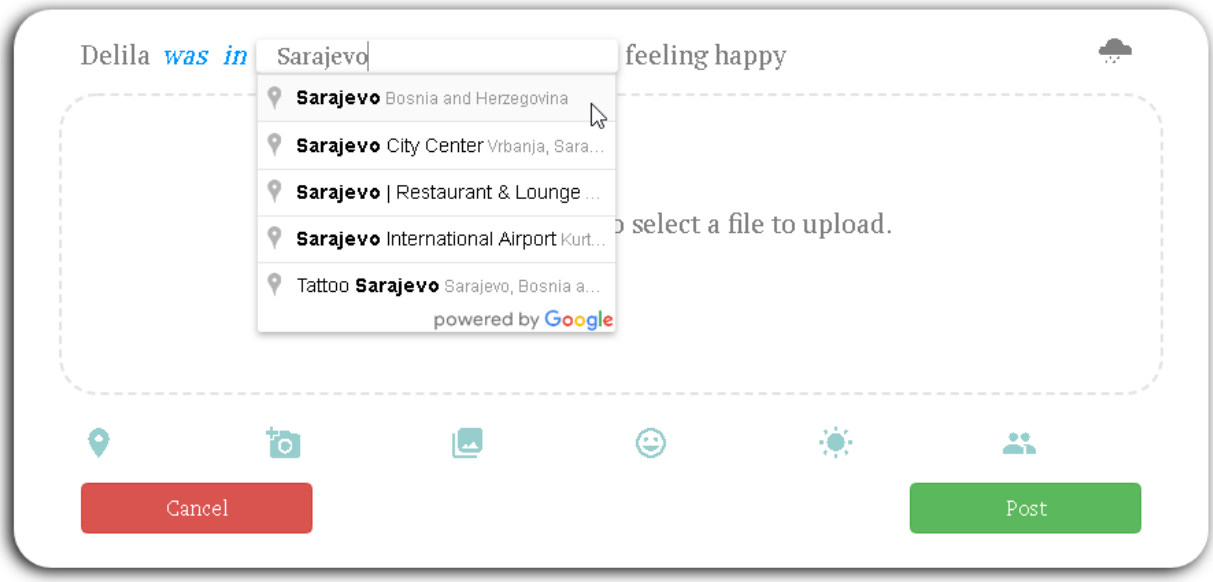
Klikom na neku od opcija raspoloženja, sistem mijenja rečenicu koja će biti objavljena, dok se klikom na opciju vremenskih prilika dodaje odabrana ikona, kao što je prikazano na slici 4.6.

Slika 4.6. Prikaz promjene komponente NewPost

Kako bi se omogućila opcija dijeljanja informacija o lokaciji implementirane su komponente CustomStandaloneSearchBox i MainMap. Obje komponente koriste Google Maps JavaScript API, pa je prije implementacije potrebno registrovati projekat na Google API konzoli i na taj način dobiti ključ koji omogućava korištenje navedenog API-ja.

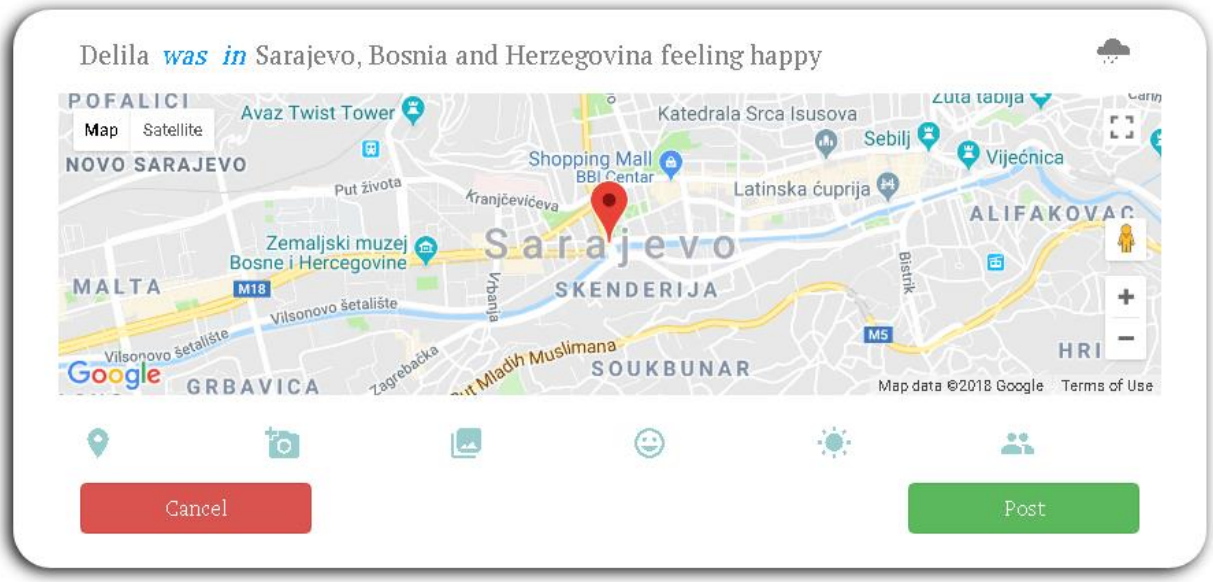
U obje komponente importovana je biblioteka react-google-maps koja olakšava implementaciju željenih funkcionalnosti pretrage i prikazivanja lokacije na mapi i integraciju React aplikacije sa Google Maps JavaScript API-jem.

Funkcionalnost pretrage i odabira željene lokacije postignuta je implementacijom komponente CustomStandaloneSearchBox, kao što je prikazano na slici 4.7. Ova komponenta koristi biblioteku react-google-maps za pronalazak svih mjesta koja u nazivu sadržavaju uneseni tekst.



Slika 4.7. Prikaz pretrage i odabira lokacije

Rezultat ove pretrage je JSON objekat koji između ostalog sadrži objekat location, pomoću kojeg je moguće dobiti geografsku dužinu i širinu odabrane lokacije. Navedeni podaci se prosljeđuju komponenti MainMap koja je zadužena za prikazivanje mape na kojoj je prikazana odabrana lokacija, što je prikazano na slici 4.8.



Slika 4.8. Prikaz Google mape unutar aplikacije

React-ova komponenta GoogleMap, importovana iz navedene biblioteke react-google-maps pruža velik spektar atributa koji olakšavaju vizuelnu prezentaciju raznih stilova mapa, od kojih su najvažniji:

- defaultCenter – geografska širina i dužina lokacije
- defaultZoom – inicijalna širina prikaza lokacije

Na listingu 4.3. prikazana je komponenta MainMap.

```
import React, { Component } from 'react';
 import { compose, withProps, lifecycle } from 'recompose';
 import { withScriptjs, withGoogleMap, GoogleMap, Marker } from 'react-google-maps';

 const MainMap = withScriptjs(withGoogleMap((props) =>
        <GoogleMap
              defaultZoom={14}
            defaultCenter={{ lat: props.lat, lng: props.lng }}
        >
     {props.isMarkerShown && <Marker position={{ lat: props.lat, lng: props.lng }} />}

   </GoogleMap>
 ));

export default MainMap;
```

Listing 4.3. Prikaz koda komponente MainMap

Sljedeća *child* komponenta komponente NewPost je PictureDropzone. Ova komponenta omogućava korisnicima da učitaju sliku koju žele da objave. Funkcionalnost učitavanja slike omogućena je i klikom na odgovarajuću ikonu koja predstavlja galeriju, ali navedena komponenta pruža dodatnu mogućnost *drag and drop* slike.

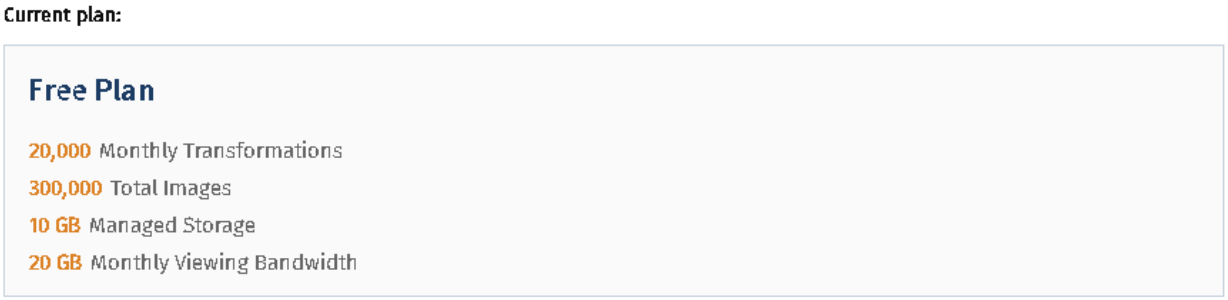
Drag and Drop slike je postignunt korištenjem biblioteke react-dropzone. Ova biblioteka pruža komponentu Dropzone koja kao *props* parametar može primiti funkciju koja definiše ponašanje sistema nakon učitane slike. . Kod ove funkcije je prikazan na listingu 4.4.

```
const CLOUDINARY_UPLOAD_PRESET = 'n9wsbdne';
 const CLOUDINARY_UPLOAD_URL = 'https://api.cloudinary.com/v1_1/dykzjdkwy/upload';

   . . .

   handleImageUpload(file) {

      let upload = request.post(CLOUDINARY_UPLOAD_URL)
         .field('upload_preset', CLOUDINARY_UPLOAD_PRESET)
         .field('file', file);

      upload.end((err, response) => {
         if (response.body.secure_url !== '') {
            this.setState({
               uploadedFileCloudinaryUrl: response.body.secure_url
            });
            this.props.imageUploaded(response.body.secure_url);
         }
      });
   }
```

Listing 4.4. Funkcija za učitavanje slike

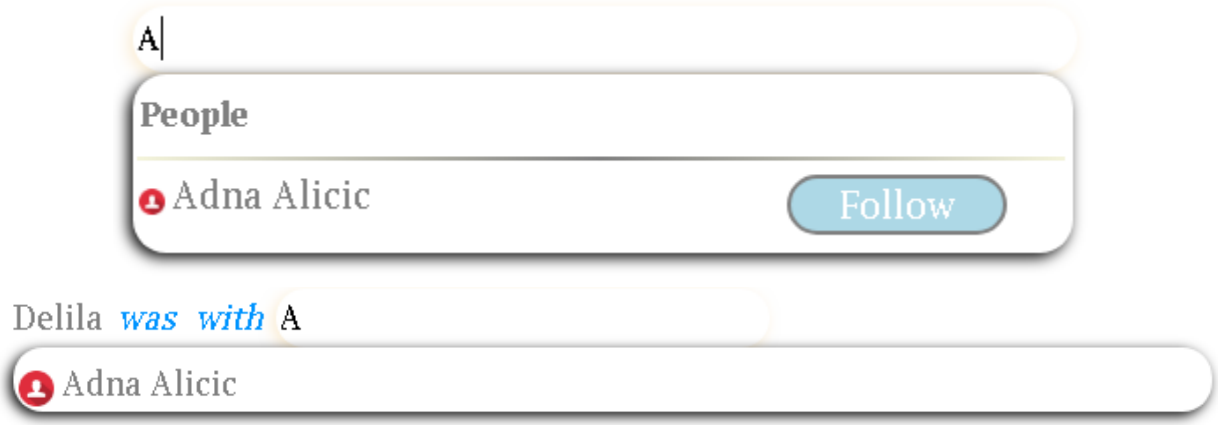
Sve slike učitane sa klijentskog računara se *uploaduju* na Cloudinary, koji predstavlja jednostavno rješenje za upravljanje slikama bilo koje web aplikacije. Cloudinary pokriva prenos, skladištenje, manipulaciju i isporuku prvenstveno slika i videa, ali i drugih oblika datoteka. U bazi podataka aplikacije Wonderland se čuva URL slike koja je postavljena na Cloudinary, čime se olakšava pristup i prikazivanje slika.

Za potrebe manipulacije slikama aplikacije Wonderland iskorištena je besplatna verzija ovog servisa koja nudi 10 GB prostora za skladištenje slika. Svi podaci vezani za besplatni plan su prikazani na sljedećoj slici.



Slika 4.9. Prikaz dostupnih Cloudinary resursa

Posljednja komponenta iskorištena za realizaciju komponente NewPost je SearchBar, čija implementacija neće biti prikazana, ali je važno napomenuti da je ova komponenta prvenstveno implementirana za potrebe pretrage korisnika i lokacija, te praćenja istih. Prosljeđivanje različitih *props* parametara omogućava ponovnu iskoristivost navedene komponente u svrhu označavanja osoba na objavi. Komponenta SearchBar ima različit izgled u dva navedena slučaja, što je prikazano na slici 4.9.



Slika 4.10. Prikaz komponente SearchBar iskorištene na dva različita mjesta

U ovom poglavlju predstavljena je implementacija NewPost komponente i njenih podkomponenti, a samim tim i analizirana implementacija funkcionalnog zahtjeva vezanog za dijeljenje objava.

5. Sigurnost web aplikacije Wonderland

U okviru ovog poglavlja opisani su postupci zaštite web aplikacije Wonderland, odnosno implementacija nefunkcionalnih sigurnosnih zahtjeva aplikacije koji su navedeni u sekciji 3.6.2. Detaljno je prikazan način kodiranja korisničke šifre, kao i razlozi koji su doveli do potrebe za jakom zaštitom šifre.

Sigurnost web aplikacija je neizostavan aspekt razvoja i održavanja svake web stranice. Web aplikacije postaju sve popularnije, s obzirom na široku primjenu i dostupačnost sa bilo kojeg mjesta, računara ili mobitela, što ih čini pogodnom metom za različite napade, bili oni namjerni ili nenamjerni. Korištenje aplikacija za internet bankarstvo, trgovinu, društvene mreže i sl. obično podrazumijeva razmjenu povjerljivih informacija, što može biti veliki sigurnosni problem. Nepoduzimanje određenih mjera zaštite takvih web aplikacija može dovesti do njihovog *kraha* ili curenja povjerljivih informacija.

Neki od najčešćih oblika napada su: SQL injection, *brute force* i već pomenuti Cross Site Scripting. Sve tri vrste napada se mogu izbjeći pravilnom provjerom podataka koji se šalju serveru. Potrebno je provjeriti dvije stvari: ispunjavaju li podaci zadanu formu i sadrže li uneseni podaci u sebi sadržaj koji bi aplikacija mogla protumačiti kao kod koji je potrebno izvršiti. U web aplikaciji Wonderland su primijenjene osnove zaštite web aplikacija od napada. Važni dijelovi sigurnosti aplikacije su kriptiranje šifre i provjera podataka prije upisa u bazu podataka.

Spašavanje orginalne šifre unutar baze podataka se smatra lošom praksom. Jedna od opcija, iako ne i najbolja, jeste *heširanje* šifre. *Hešir*ane šifre nisu jedinstvene, zbog deterministične prirode *heš* funkcije koja za isti ulaz uvijek daje isti izlaz. Zbog toga, ako dva korisnika, Korisnik1 i Korisnik4, izaberu istu šifru npr. JEDRZEJCZYK njihov heš će biti isti, kao što je prikazano na tabeli 6.1. Ako napadač uspije pronaći orginalu šifru jednog korisnika, tu šifru može iskoristiti da pristupi i svim ostalim računima koji koriste istu šifru.

Korisničko ime	*Heširana* šifra
Korisnik1	a8c4d89305cd3cc438984d23b4436265
Korisnik2	482c811da5d5b4bc6d497ffa98491e38
Korisnik3	f3b88fc050a430ad2042c351973c3188
Korisnik4	a8c4d89305cd3cc438984d23b4436265
Korisnik5	562f71f117590df1ea68d58386b97339

Tabele 6.1. Primjer heširanja MD5 algoritmom

Najjednostavniji način otkrivanja orginalne šifre koji napadač može iskoristiti je dictionary napad. Upotrebom unaprijed pripremljene liste riječi i njihovih *heširanih* vrijednosti, koristeći riječi iz npr. bosanskog rječnika, napadač može jednostavno porediti *heširane* stringove iz ukradene tabele koja sadrži sve šifre sa svakim *heširanim* stringom iz liste. Ukoliko se dvije vrijednosti popudaraju, pronađena je orginalna šifra. Ipak, za otkrivanje nestandardne šifre, poput one koju koriste Korisnik1 i Korisnik4 napadač mora koristiti neki specijalni rječnik, kao što je leetspeak.

Ovakav način napada zahtijeva izračunavanje *heširane* vrijednosti u realnom vremenu, za što je potrebno mnogo vremena, ukoliko se koristi dobra *heš* funkcija. Napadač može koristiti rainbow tabele kako bi zaobišao ovaj problem. Rainbow tabele predstavljaju tabele koje sadrže riječi iz nekog riječnika i *random* riječi i njihove *heširane* vrijednosti. Kako su ove tabele unaprijed pripremljene napadač može samo izvršiti poređenje *heširanih* vrijednosti, bez gubljenja vremena na *heširanje.* Napadač obično prvo nastoji otkriti šifre koje je najviše ponavljaju.

Bolja opcija kriptiranja šifre, kojom se umanjuje opasnost od navedenih napada, je dodavanje salt stringa na šifru. Prema OWASP-u salt je kriptografski jaka nasumična vrijednost fiksne dužine koja se dodaje uzalu *heš* funkcije, kako bi se stvorio jedinstven *heš* za svaki ulaz, bez obzira na to da li je ulaz jedinstven.[6] Najbolje rješenje autentifikacije korisnika predstavlja korištenje salt stringa, zajedno sa *heširanjem.*Neke od najpoznatijih kriptografskih funkcija koje koriste salt i *heširanje* su SHA-2 i SHA-3. Najveći problem SHA funkcija je njihova brzina. Ove funkcije su dizajnirane da rade veoma brzo, što omogućava napadaču brže i lakše otkrivanje šifre korisnika. Iz tog razloga, u pokaznoj aplikaciji Wonderland korišten je znatno sporiji algoritam bcrypt.

Bcrypt je algoritam za heširanje, dizajniran od strane i Niels Provosa i David Mazièresa. Baziran je na Blowfish deterministčkom algoritmu koji koristi simetrični ključ. Blowfish algoritam je veoma brz, osim u slučajevima kada se pomenuti ključ mijenja. Svaki novi ključ zahtijeva proces obrade ekvivalentan enkripciji 4KB tekst. Ova osobina Blowfish algoritma je veoma korisna za algoritme *heširanja* koji je koriste, jer usporava napade i na taj način pomaže u zaštiti podataka. Još jedna od prednosti bcrypt-a je to korištenje salt stringa dužine 128 bita.

Web aplikacija Wonderland koristi bcrypt tokom upisa i provjere šifre u bazi podataka, kao što je prikazano na listingu 6.1. i listingu 6.2. Time se sprječava neovlašten pristup korisničkim računima. Korisnik ne može povratiti šifru ukoliko je zaboravi, jer je bcrypt jednosmjerni algoritam, što znači da nije moguće dobiti orginalnu šifru iz *heširane.* U tom slučaju se korisniku nudi mogućnost kreiranja nove šifre.

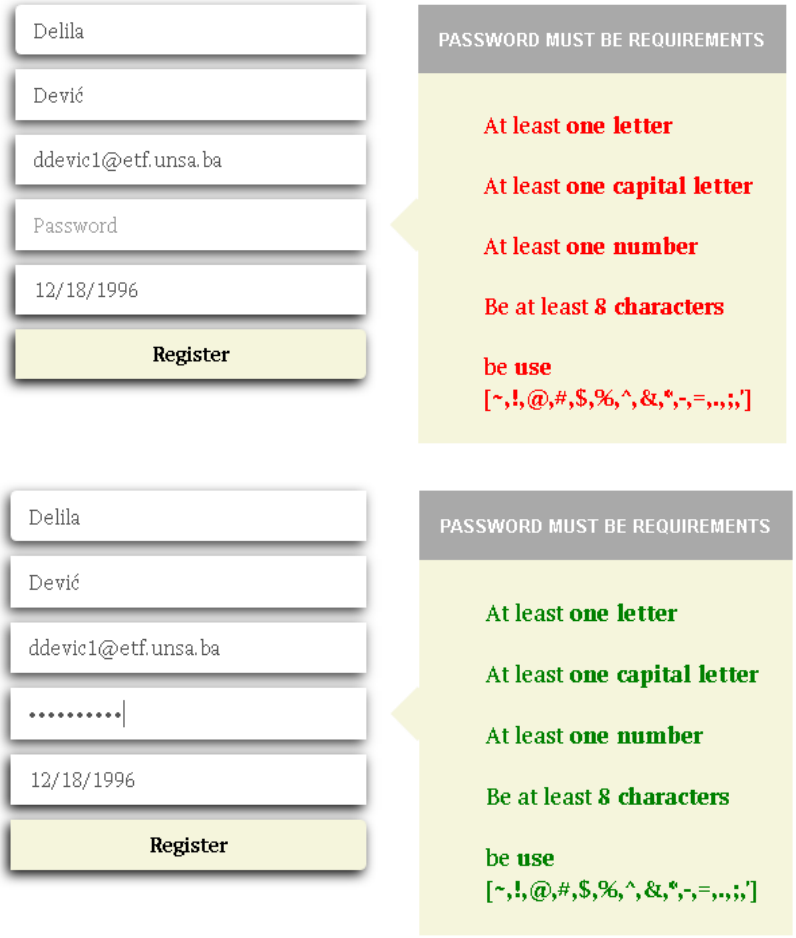
```
hashPassword(password, callback) {

      bcrypt.genSalt(SALT_FACTOR, function (error, salt) {
        if (!error)
          bcrypt.hash(this.refs.password.value, salt, function (error, hash) {
            if(!error)      callback(null, hash);
              else    callback(error, null);
             }
          else    callback(error, null);
           }
   }

 // Funkcija koja se šalje kao callback parametar funkciji hashPassword
 handleRegistration (error, password){

        if(!error){  /* Upis korisnika u bazu podataka */ }
        else{   /* Obavijest o neuspješnoj registraciji */  }
 }
```

Listing 6.1. Prikaz korištenja bcrypt-a za heširanje šifre

```
comparePassword(candidatePassword, originalPassword, callback) {

      bcrypt.compare(candidatePassword, originalPassword, function (error, isMatch) {
        if(!error)    callback(null, isMatch)
        else    callback(err, null)
       });
 }

 // Funkcija koja se šalje kao callback parametar funkciji comparePassword
 handleLogin(error, isMatch)

        if(!error){ /* Preusmjeravanje na početnu stranicu */  }
        else{ /* Obavijest o pogrešnoj kominaciji korisničkog imena i šifre */  }
  }
```

Listing 6.2. Prikaz provjere unesene šifre

Jedan od načina sprečavanja dictionary napada u aplikaciji Wonderland je podsticanje korisnika na korištenje šifre koja sadrži kombinaciju velikih i malih slova, brojeva i drugih znakova. Nakon klika na mjesto predviđeno za upis šifre, korisniku se desno od forme za registraciju prikazuju navedini zahtjevi vezani za šifru. Boja fonta zahtjeva se mijenja iz crvene u zelenu, nakon što je on zadovoljen. Na slici 6.1. je prikazana ta promjena prije i nakon unosa validne šifre 'Delila00@'.



Slika 5.1. Prikaz validacije šifre

Ova funkcionalnost je implementirana koristeći biblioteku jQuery, radi jednostavnije implementacije. Pri korištenju jQuery-ja u React aplikaciji potrebno je paziti da React i jQuery ne ažuriraju iste DOM elemente. Primjer korištenja jQuery-ja je prikazan na listingu 6.3.

```
if (pswd.length < 8) {
        $('#length').removeClass('valid').addClass('invalid');
 } else {
        $('#length').removeClass('invalid').addClass('valid');
 }
```

Listing 6.3. Korištenje jQuery-ja za promjenu boje fonta zahtjeva za dužinom šifre

Za validaciju svih polja za unos korištene su biblioteke react-easy-validation i validator. Biblioteka react-easy-validation obezbjeđuje komponente ValidationIn i ValidationOut koje predstavljaju omotače oko standardnih komponenti koje je potrebno validirati Prednosti korištenja komponenti ove biblioteke su: [8]

- Jednostavna upotreba
- Podržavaju komponente koje posjeduju upravljanje greškama
- Podržavaju validaciju skupa komponenti, dodjeljujući im kao atribut naziv grupe
- Mogu pristupiti vrijednosti unutar komponente oko koje su omotani
- Mogu se koristiti sa bibliotekama poput react-toolbox ili material-ui

Biblioteka validator pruža funkcije koje olakšavaju validaciju. Primjer korištenja navedenih biblioteka za validaciju emaila prikazan je na listingu 6.4.

```
<ValidationIn
        groups={['form', 'email']}
        validators={[{
        rule: value => value,
        hint: 'Required'
        }, {
        rule: value => validator.isEmail(value),
        hint: 'Not valid'
        }]}>

        <TextField
                id="email" ref="email"
                floatingLabelText="Email"
                fullWidth
                onChange={this.handleEmail}
                value={this.state.email}
        />

</ValidationIn>
```

Listing 6.4. Validacija polja za unos email-a

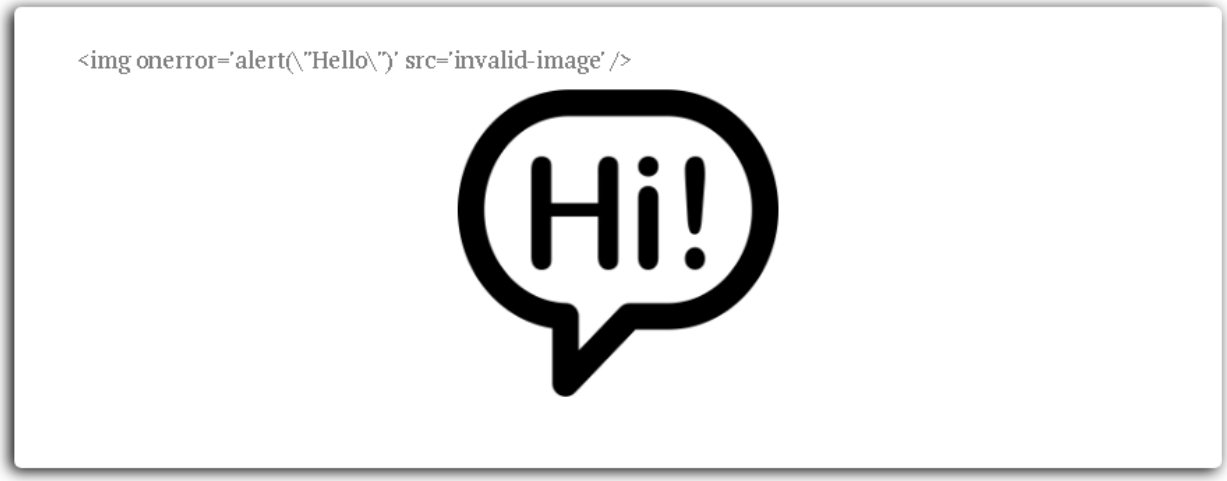
Na slici 6.2. prikazan je izgled procesa validacije email-a, početni izgled polja za unos, izgled polja za unos pri unosu nevalidne, a zatim validne vrijednosti.

Slika 5.2. Prikaz validacije email-a

React-ov dobar dizajn osigurava zaštitu web aplikacije Wonderland od XSS napada. Odbrana od XSS napada predstavlja posljednji nefunkcionalni sigurnosni zahtjev koji će u ovom radu biti obrađen.

U slučaju da je aplikacija Wonderland implementirana bez korištenja React biblioteke i bez poduzimanja dodatnih mjera za zaštitu, napadač bi bio u mogućnosti da pri registraciji u bilo koje od polja za unos, uzmimo za primjer polje za unos imena, upiše JavaScript kod. Uneseni kod bi se izvršavao na pretraživaču svakog korisnika koji pogleda sadržaj web aplikacije koji sadrži napadačevo ime.

Dakle, ukoliko bi napadač u polje za ime unio „<img src='invalid-image' onerror='alert(\“ Hello \“)'“, te ukoliko bi drugi korisnik pogledao njegovu objavu, na ekranu bi se pojavio *pop-up* prozor sa tekstom Hello. React automatski rješava ove probleme, izbjegavajući izvršavanje HTML i JavaScript koda u stringovima. Umjesto izvršavanja koda, React će jednastavno prikazati čitav uneseni tekst. Primjer objave napadača koji je unio navedeni kod umjesto imena je prikazan na slici 6.3.



Slika 5.3. Prikaz objave XSS napadača

Iako React automatski rješava većinu napada, postoje problemi na koje se treba obratiti pažnja. Jedan od takvih je korištenje React-ove funkcije dangerouslysetinnerHTML koja predstavlja zamjenu za funkciju innerHTML , gdje treba paziti da tekst unutar funkcije ne sadrži JavaScript kod. Drugi od problema se javlja kada je korisnik u mogućnosti unosa URL-a neke stranice, te proslijeđivanja istog atributu href a *tag*-a. Ovo omogućava unos JavaScript koda koji će se izvršavati na pretraživaču drugih korisnika koji otvore taj link umjesto URL-a stranice. Iako se ovaj problem može riješiti izbacivanjem podstringa „javascript:“ iz unesenog URL-a, napadač ipak može iskoristiti base64 šeme za kodiranje koda koji želi da unese. Treći problem je dozvoljavanje korisnicima da definišu vlastite objekte koji se proslijeđuju kao props parametri.

Pri implementaciji aplikacije Wonderland su uzeti u obzir navedeni problemi, tako da je izbjegnutno korištenje pomenutih problematičnih funkcija, unosa proizvoljnih URL-ova i definisanje vlastitih objekata. Dodatni način odbrane od Cross Site Scripting napada u aplikaciji Wonderland predstavlja karakterističan dizajn komponente za kreiranje nove objave. Pri kreiranju nove objave korisnik je ograničen na izbor ponuđenih opcija. Onemogućavanje korisnika da unosi proizvoljan tekst u novu objavu sprečava dijeljenje *hoaxes* i urbanih legendi koje predstavljaju priče sa zanimljivim naslovom i netačnim sadržajem koji često nije povezan sa naslovom.

U ovom poglavlju opisana je sigurnost web aplikacije Wonderland. Detaljno je opisan način kodiranja korisničke šifre, validacija svih polja za unos podataka, a posebno validacija šifre i način zaštite od Cross Site Scripting napada. *Iako React sam rješava većinu Cross Site Scripting napada prikazane su situacije u kojima se moraju poduzeti dodatne mjere zaštite.*

6. Zaključak

Cilj ovog rada je bila analiza načina rada i prednosti koje nudi biblioteka za razvoj korisničkog interfejsa React, pa će ovaj rad biti završen prikazom najvažnijih prednosti pomenute biblioteke.

Rad sa DOM-om može biti veoma težak. Jedna od glavnih prednosti React biblioteke je manipulacija virtuelnom reprezentacijom DOM-a, umjesto stvarnim DOM-om pretraživača. Za reflektovanje promjena na pretraživačevom DOM-u je zadužen React. Dakle, React se ponaša kao posrednik između programera i pretraživača. Reactova implementacija virtualnog DOM-a dolazi sa optimizacijom performansi algoritma koji se koristi za njegovo ažuriranje.

Za React se kaže da je deklarativna biblioteka, jer služi za opisivanje korisničkih interfejsa i modeliranje njihovih stanja. Deklarativno programiranje u Reactu omogućava programeru jednostavno definisanje različitih stanja komponenti, dok je sam React zadužen za ažuriranje korisničkog interfejsa.

Ovdje je važno pomenuti i način ažuriranja korisničkog interfejsa. Naziv React (reagovati) je jednostavno objašnjenje za ovaj koncept. Sve promjene stanja komponente ili podataka koje komponenta koristi reflektuju se na korisnički interfejs. React reaguje na promjene i automatski ažurira samo određene dijelove korisničkog interfejsa, a ne cijelu web stranicu.

Ukoliko se pravilno koristi, React pruža mnoge povlastice unutar aplikacija. Ovo se odnosi na brzinu, jednostavnost implementacije i male potrebne resurse, ali i na dobre prakse čije korištenje potiče, što rezultira implementacijom brze aplikacije sastavljene od ponovo upotrebljivih, lahko razumljivih komponenti.

Kroz ovaj rad opisana je i implementacija društvene mreže Wonderland kako bi se praktično prikazale prednosti React biblioteke. Web aplikacija Wonderland je intuitivna i lahka za korištenje, te nisu potrebna dodatna pojašnjenja za njeno korištenje. Bitno je istaknuti da su poduzete određene mjere zaštite sigurnosti podataka, jer je sigurnost neophodan aspekt razvoja svake aplikacije.

Pokazna web aplikacija Wonderland pruža još mjesta za nadogradnju u vidu dodatnih funkcionalnosti, koje mogu biti mnogobroje s obzirom na činjenicu da sve društvene mreže konstantno rastu pružajući nove usluge svojim korisnicima.

7. Reference

[1] Bill Fisher, 11.02.5015. – https://www.quora.com/React-JS-Library/How-was-the-idea-to-develop-React-conceived-and-how-many-people-worked-on-developing-it-and-implementing-it-at-Facebook/answer/Bill-Fisher-17
[Datum preuzimanja: 20.06.2018.]

[2] The road to learn React, Robin Wieruch – https://github.com/the-road-to-learn-react/the-road-to-learn-react/blob/master/manuscript/chapter1.md
[Datum preuzimanja: 05.06.2018.]

[3] React – Overview – https://reactjs.org/
[Datum preuzimanja: 06.06.2018.]

[4] Fullstack React, Anthony Accomazzo, Ari Lerner, David Guttman, Nate Murray, Clay Allsopp and Tyler McGinnis – http://opencarts.org/sachlaptrinh/pdf/900001.pdf
[Datum preuzimanja: 1.07.2018.]

[5] React – Components and Props – https://reactjs.org/docs/components-and-props.html
[Datum preuzimanja: 4.07.2018.]

[6] OWASP Use a cryptographically strong credential-specific salt – https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet#Use_a_cryptographically_strong_credential-specific_salt
[Datum preuzimanja: 11.08.2018.]

[7] Usenix documentation https://www.usenix.org/legacy/publications/library/proceedings/usenix99/full_papers/provos/provos_html/node5.html
[Datum preuzimanja: 05.08.2018.]

Prilog – Prikaz pokazne aplikacije Wonderland

Pokazna aplikacija Wonderland razvijena je kako bi se praktično prikazale prednosti i način rada React biblioteke. Web aplikacija Wonderland je intuitivna i lahka za korištenje, te nisu potrebna dodatna pojašnjenja za njeno korištenje.

Web aplikacija Wonderland predstavlja društvenu mrežu koja je namijenjena prvenstveno turistima. Može biti izuzetno korisna i zanimljiva za one koji žele da podijele svoje fotografije i priče, da se sretnu sa nekim na putovanjima ili da pronađu iskustva drugih korisnika. Tokom razvoja aplikacije velika pažnja je posvećena osiguravanju dobrog korisničkog iskustva.

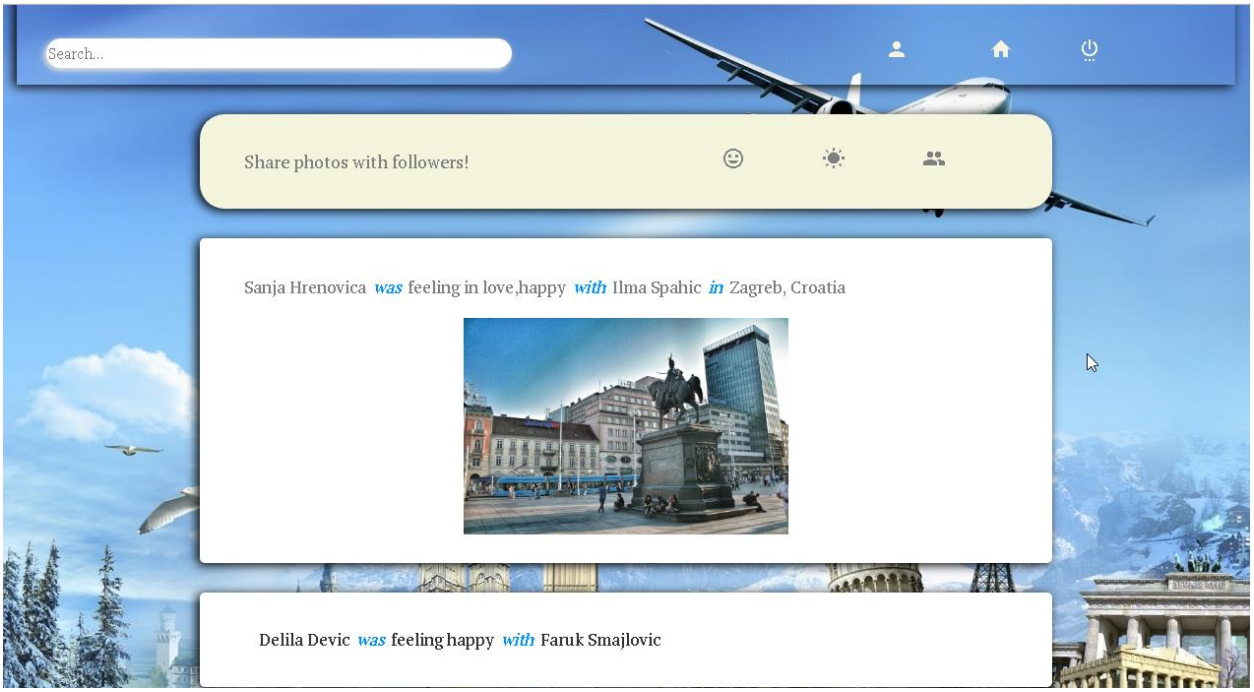
Namjena, pregled sadržaja, funkcionalni i nefunkcionalni zahtjevi aplikacije Advena navedeni su u poglavlju 3. U nastavku će biti prikazane implementirane funkcionalnosti uz kratak opis istih.

Nakon pokretanja aplikacije korisniku se prikazuju forme za prijavu i registraciju. Prilikom registracije potrebno je unijeti ime, prezime, datum rođenja, email i šifru. Prilikom unosa zahtijevanih podataka, vrši se validacija podataka i korisnik se obaviještava o uspješnoj registraciji ili greški koja se desila.

Prilog – Slika 1. Početna stranica

Prilikom prijave, od korisnika se traži unos korisničkog imena i šifre. Nakon unosa podataka, vrši se njihova validacija i u slučaju ispravne prijave korisnik se preusmjerava na naslovnu stranicu.

Naslovna stranica služi za pregled objava osoba koje korisnik prati i objava vezanih za lokacije koje ga zanimaju, sadrži polje za pretragu, te segment za objavljivanje statusa i slika.

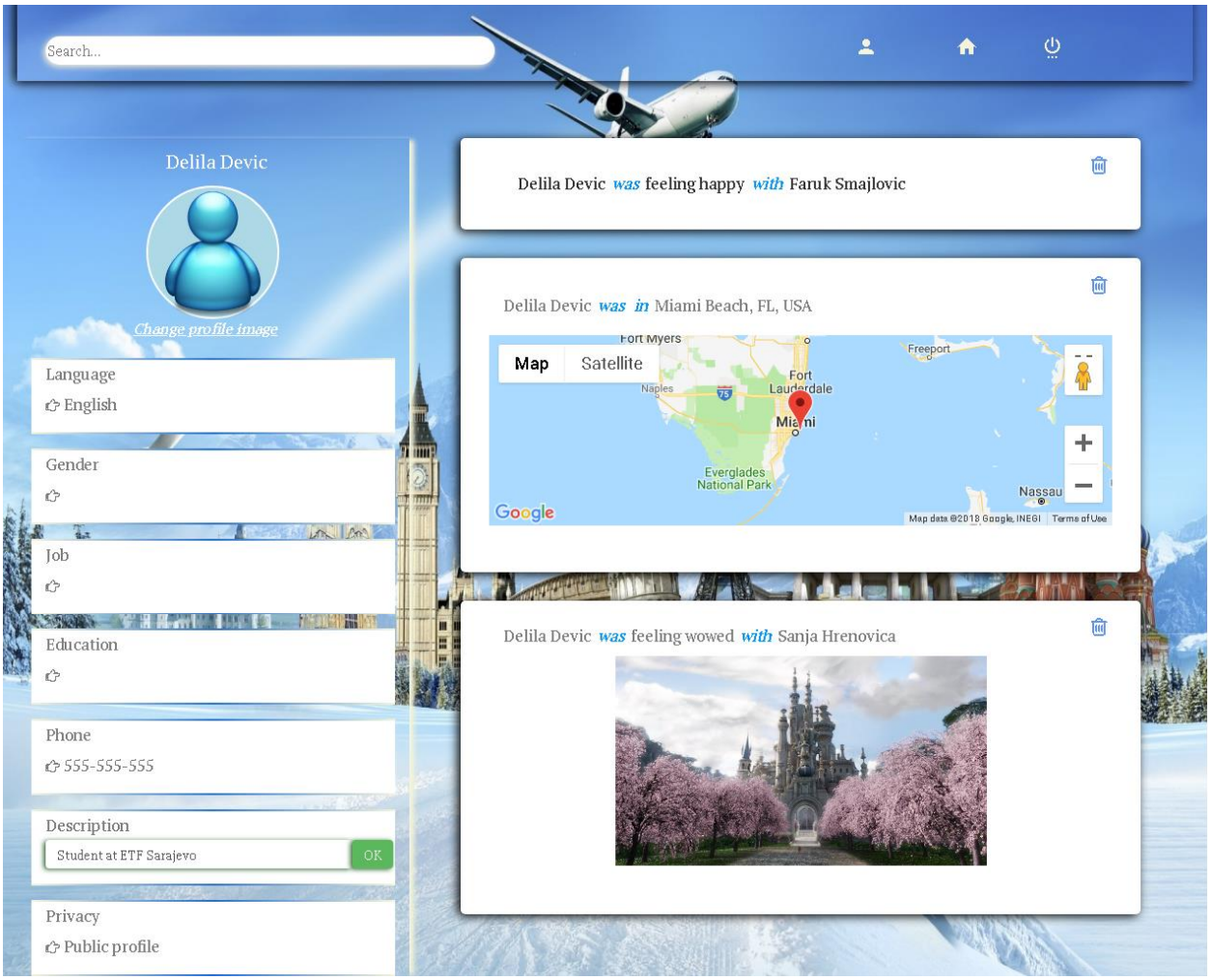


Prilog – Slika 2. Naslovna stranica

Nakon klika na segment za objavljivanje statusa prikazuju se dodatne mogućnosti vezane za kreiranje objave. Dodatne mogućnosti obuhvataju dodavanje slike, označavanje pratitelja, dijeljenja informacija o trenutnoj lokaciji, raspoloženju i vremenskim prilikama. Proces kreiranja nove objave detaljno je objašnjen u poglavlju 4.

Prilog – Slika 3. Kreiranje nove objave

Klikom na ikonu profila korisnik može pregledati sve svoje objave, te uređivati lične informacije ukoliko želi. Na profilnoj stranici korisnik ima mogućnost promjene profilne slike klikom na „Change profile picture“ ili *drag and drop* slike sa vlastitog računara. Ukoliko se neka od objava ne dopada korisniku ima mogućnost da je obriše.



Prilog – Slika 4. Profilna stranica

Bilo koji prijavljeni korisnik može pregledati svoj profil ili profil svakog od korisnika ove društvene mreže. U ovisnosti od postavljene vidljivosti prikazuju se određeni podaci. U slučaju da je korisnički račun privatni, lične podatke i objave mogu vidjeti samo oni korisnici koji ga prate, dok se korisnici koji ne prate taj korisnički račun obavještavaju o privatnosti profila koji žele posjetiti.

Prilog – Slika 5. Profilna stranica privatnog profila

Na profilu korisnika sa privatnim računom prikazani su i zahtjevi za praćenjem, koje korisnik može prihvatiti ili odbiti. Ako primalac odluči da prihvati zahtjev, podnosilac zahtjeva se dodaje u listu pratitelja primaoca. U suprotnom, pošiljalac prima obavijest da je zahtjev za praćenje odbijen i isti se uklanja iz liste pristiglih zahtjeva kod primaoca.

Prilog – Slika 6. Pregled pristiglih zahtjeva za praćenje

Popis slika